Abstract

An innovative inventory management system for the centralised accounting and management of various inventory items for a network of counter less/ cashier less and also for the management of grocery and vegetable stores in a family.

The above system and software can be used to manage the inventory of grocery and vegetable stock in a family. This system can be linked to the network of any supermarket chains of the customer. Here the information about emptying of the store and the probable requirement of various items can be identified by the network system and at any time the customer can be warned by messaging. The store can be ready with the items that can be delivered on confirmation of the customer order.

Contents

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

.

Item	Description
RFID	Radio Frequency IDentification
V	Voltage
A	Amperes
R1	Strain gauge 1
R2	Strain gauge 2
R3	Strain gauge 3
R4	Strain gauge 4
ADC	Analog to Digital Converter
MbPS	Megabits Per Second
DC	Direct Current
AC	Alternating Current
IDE	Integrated Development Environment
PC	Personal Computer

Chapter 1

INTRODUCTION

In the supermarkets and malls, the various items stores have currently been accounted and managed manually. A large manpower is required to monitor the movement of various items and for forecasting the requirements of replenishing certain fast-moving products. Many times, the sales assistants won't be able to track certain items that may get exhausted abruptly for various unexpected reasons. In such a situation, the mall may not be able to provide the product to the customer even if they have enough stocks at a remote store house.

1.1 Problem Identification

The existing system uses manual system for managing inventory processing which results in time waste to taking stock and updating the stock on daily basis. On the other hand, it also leads to inefficiency and in accuracy in keeping record of the items in the store. So, this process requires the human power to maintain stock level of every item. The highs and lows of stock levels should be updated every time. This bring lot of problems which includes too much of paper work and difficulty in updating every time a change is made. To overcome this problem, the automated inventory system is needed.

1.2 Problem Formulation

Two basic components are required for monitoring such a system:
1. The product identification system.

2. A system is required to ID and record each product and corresponding weight/ number when it goes to a particular rack.
Similarly, when the customer/sales person takes away any packet/bottle, the system and software had to ID and record the change in the weight/volume. Now using this system, the incoming and outgoing movement of the product can be monitored and recorded in the local computer.

1.3 Problem Statement & Objectives

An innovative inventory management system for the centralised accounting and management of various inventory items for a network of counter less/ cashier less and also for the management of grocery and vegetable stores in a family.
The main objectives are:
Identification of Product Labels: Stores must be able to identify and label the individual food products with which a user interacts.
No Additional Human Effort: To support unobtrusive tracking, Stores must not require the user to perform any additional actions.
Need to Estimate Residual Amount in a Container: Past studies have shown that users who are aware of the amount of unconsumed food items in their store make less wasteful consumption decisions.

1.4 Limitations

This project has faced with a problem of getting slow server response. Therefore the refresh rate will be slow throughout the process.
In this project the server cant detect non RFID weights and can only add one item in the fridge at a time.

Chapter 2

RESEARCH METHODOLOGY

This smart inventory can be used in shops and supermarkets for stock management. In this smart inventory system, any change in the weight of the goods is precisely calculated by a computer program and logical decisions are made whether the inventory require more stock or not. The RFID system scans the products in the inventory. To gather data and deliver it to the server, it is connected to a wireless network.

RFID is used in this system to scan and detect the products. The RFID system generates and maintains a magnetic field around its coil by using alternating current. The RFID system consists of two components: an RFID reader and an RFID tag. RFID tags are detected by the RFID reader by creating a magnetic field. Therefore, RFID tags can be attached to the goods which need to be detected by the RFID reader. When used in an inventory system, an RFID reader sends out signals continuously until it detects the needed modulated signal. The product is listed in the inventory if the signal received is correct. This method works when the product is on the store's shelves.

In this system, we are using load cells that are basically metal spring elements which consists of Strain gauges. The resistance of a Strain gauge change when a load or weight is applied on it. The load cells can convert the change in resistance of strain gauges to a much usable form such as voltage difference and also this voltage is analog in nature. To amplify this difference and convert it to digital form before processing it through a program, an analog to digital converter is required. In this system we are using dual channel HX711 pressure sensor module which has an inbuilt amplifier. The digital output from this module can be fed directly to the inputs of the Arduino UNO, which can process the information based on the requirement of inventory system using a computer program. The load cell can be attached beneath the base of a storage crate, the output of this load cell is given to the HX711 pressure sensor module which outputs the digital values to the Arduino UNO. Arduino UNO can be programmed to calculates the number of items in the crate based on their weight and RFID tag.

The details of products in the inventory are stored in a database. Each product entry's time and date are likewise saved in the database. All the details of the inventory can be viewed from a web application regardless the device or the location. Also, the user can check if any products are missing and has the option to buy products from third-party vendors. The user can check the web application anything to see the list of products in the inventory and details including product name, weight, product id and, date and time.

Chapter 3

LITERATURE SURVEY AND REVIEW

Other papers and articles were referred for the development of our automated inventory management system. Each papers have different methods in implementing their project. The benefits and difficulties in every paper is considered in our project.

3.1 Literature Collection & Segregation

”Smart Shelf for Smart Kitchen” by Apoorva Verma This article proposes that the Smart Shelf can measure grocery goods. The Smart Shelf is divided into two parts: weight sensing and level sensing. The level sensing part consists of a fixed-size container with an RFID tag that defines container size and product description, an RFID tag reader, and an ultrasonic sensor that measures the level of the container’s contents. The RFID tag with similar container specifications and content identification, the RFID tag reader, and the weight sensor measuring all contents on the shelf make up weight sensing.

”Research and Design of the Intelligent Inventory Management System Based on RFID” by Xiaojun Jing and Peng Tang They suggested a system that combined RFID and ZIGBEE. ZIGBEE focuses on detecting indicators in small areas, and RFID can identify the type and location of some sensor nodes. The fundamental disadvantage of RFID is that its detection range is limited. It is also difficult for the RFID reader to read the signal precisely if numerous loads are on the same palette.

“IoT Based Smart Inventory Management System for Kitchen Using Weight Sensors, LDR, LED, Arduino Mega and NodeMCU (ESP8266) Wi-Fi Module with Website and App” by Sifat Rezwan, Wasit Ahmed, Mahrin Alam Mahia and Mohammad Rezaul Islam Smart Kitchen Inventory Management System (SIMS) is an IoT-based solution that makes managing kitchen, medication, and restaurant inventory more efficient and convenient. This will not only alert users of their existing inventory, but it will also place an order for additional things if the number runs out. Users can also manually order products online and have them delivered to their doorstep using the SIMS app. The user can also create a list for a specific timeframe in order to keep track of their spending. Additionally, users can utilise the website to follow their order status and history. People may forget about the hassles of food shopping with the help of SIMS’ Smart Kitchen Inventory (SKI), which allows them to order anything they need from anywhere using the website or the Android app.

“Smart Inventory Management Using Electronic Sensor Based Computational Intelligence” by Kunal Singh Electronic sensor-based technologies combined with this intelligence can create a remarkable inventory management system that can not only keep a thorough track of the goods required, but is also very easy to manage, making these sensors-based smart inventory management systems very efficient in terms of preventing any mismanagement of goods. The author goes over several of these smart systems in depth in this essay. The disadvantages of traditional inventory management systems and the impact of computational intelligence on their improvisation have been thoroughly studied. The operation of RFID (Radio Frequency Identification), Load Cells, and Ultrasonic sensor-based inventory management systems are covered in detail. In addition, several recent advances in inventory management, such as the author’s suggested three-dimensional scanning system and other sensor-based smart inventory systems, have been thoroughly examined. Methods for enhancing the mobility and efficiency of such systems in terms of simplicity of use, which can be achieved using Bluetooth or wireless radio transmission, have been studied in length.

“Internet of Things based system for Smart Kitchen” by Jyotir Moy Chatterjee Smart Kitchen is defined in this paper as using an IoT system with numerous sensors, including a weight sensor, a gas sensor, and a temperature sensor. The gas sensor is used to detect gas leaks in the system, while the weight sensor is used to measure the cylinder’s weight. The current room temperature is detected using a temperature sensor. The threshold value is set in the kitchen, and when it crosses the settings, it notifies the user of the weight of a gas cylinder as well as any gas cylinder leaks. The server and the user can communicate via an Android device. The server can send a notification to the user by email or SMS, which will appear on their Android devices.

”Smart Refrigerator using the Internet of Things(IoT)” by M.K. Sangole This paper proposes a method in which a load cell is put beneath the refrigerator’s vegetable tray and continuously measures the weight of the vegetables in the tray. Because the weight of the vegetable tray falls below the threshold, it detects the absence of veggies. It will be accompanied by a low signal, which will be delivered to the user via the mobile app.

3.2 Critical Review of Literature

These papers and articles based on IoT based systems helped us to improve and study our project more efficiently. These papers are all based on inventory management with IoT which is similar to our project. The problems and benefits of these papers help to plan, strategize and improve our project better. Studying these papers helped us to understand the problems and developments of their projects and implement it in our project. Developments were made in our project by studying the drawbacks in these papers. Our Automated Inventory Management help users to detect products in their inventory using load cells and RFID. The products details are displayed in their device through a web application. These papers also use weight sensors and RFID which helped to compare our project with them. In addition, these papers also mentioned using ultrasonic sensor and wireless radio transmission systems.

Chapter 4

ACTUAL WORK

Upon completion of identifying & formulating the research problem, and carrying out the necessary literature survey and review, the actual work on the project is taken-up. This chapter is dedicated to the actual work done by students. Hence, the chapter name and sub-chapter names are not fixed. It is left to the discretion of the students with appropriate guidance from their respective supervisors. However, one or more of the following aspects (as applicable) shall be covered in this chapter:

- Methodology of the study or actual work (different from research methodology)
- Modeling, Analysis and Design
- Prototype and testing

4.1 Methodology for the Study

Stocking these days became very hard because of busy schedule. The previous system uses human power in tracking the stocks which makes a customer to visit the shop repeatedly in a month for buying items. In this system weight of the item can be seen on the user device and if the weight goes below the expected value, it allows the user to make stock replenishment. This system is super cost effective to identify and record the usage of the items. User can see real-time value of the item. This is an essential prototype and can also be implemented in smart home initiative. Inventory management keeps a thorough track of items required and this system is also easy to manage and became very efficient in terms of preventing mismanagement of items.

4.2 Modeling, Analysis & Design

4.2.1 Hardware Requirements

- Arduino UNO
- Node MCU
- Tags
- Buzzer
- LEDs
- Jumper Wires
- Bread board
- USB Cable
- Power Supply

4.2.2 Software Requirements

- Arduino IDE
- JSON
- Flask Server
- SQLite

4.2.3 Requirement Specifications

- Arduino UNO: Arduino UNO has 16 digital input/output pins. 6 analog pins. It contains 16MHZ Quartz Crystal. 1 USB Connection. and a power jack. It is also having an ICSP header and reset button.

FIGURE 4.1: Arduino UNO

- NodeMCU: The operating voltage of nodemcu is 3.3V. The input voltage voltage is 7-12V. NodeMCU has 16 digital input and output pins, 1 analog input pin and It contains a flash memory of 4MB.

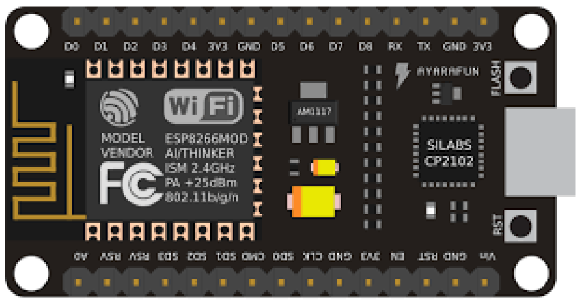


FIGURE 4.2: NodeMCU

- Buzzer: Buzzer is a signalling device which gives a beep sound. The DC voltage is given to buzzer and used in devices.

FIGURE 4.3: Buzzer

- LED: Light Emitting Diode is a light source that emits light when electricity passes through it.

FIGURE 4.4: LED

- Jumper Wires: Jumper wires are the wires used to connect different sensors. There are 3 types of jumper wires. Male-male, female-female, male-female.

FIGURE 4.5: Jumper Wires

- Bread Board: Bread boards are used as constructing base to stimulate the electronic devices. It requires no soldering. So, the changing of connections would easy in bread board.

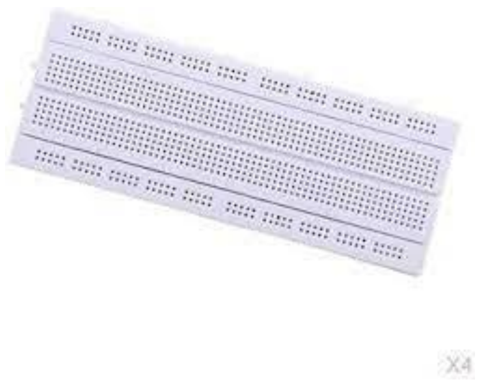


FIGURE 4.6: Bread Board

- USB Cable: USB cable wire is used to connect Arduino and laptop. It communicates the data between computer and peripheral devices at 480Mbps. These cables are very easy to afford.

FIGURE 4.7: USB Cable

- Power Supply: Power supply changes the flow of electric current from a source to right voltage and flow. These are also used as electric force converters. 5V 2A Power adapter takes an AC input of 100-240V and gives 5V 2A DC yield.

4.2.4 Module Description

RFID: Radio Frequency Identification is the best method for automated inventory system. It requires No calculation and it is very easy to implement. RFID consists of two components. one is RFID reader and the other one is RFID tag. The RFID has three frequency levels. The low frequency is 125-134kHz, high frequency is 13.56 MHz and ultra high frequency is 860-960MHz. Lower the frequency low is the range of operation

of the system.
RFID reader detects the RFID tag component, which is attached to the items which need to be detected by RFID reader. In case of bar code scanners, this RFID tag and RFID reader is not required. The passive tags which totally rely on the power which is generated by transponder to run the circuit within the tag and also generation of feedback signal. This type of water body prof and it is most recommended for automated inventory systems. Also this doesn't require power supply which makes less expensive. In this tag one part is used as transponder and the other part is used for signal generation.

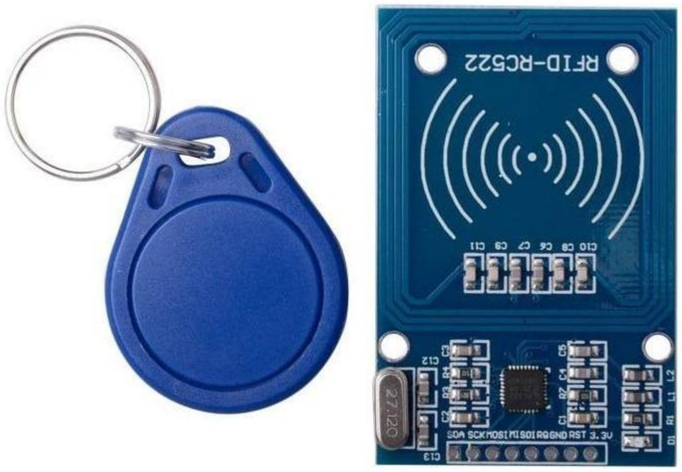


FIGURE 4.8: On the left is an RFID tag with inexpensive RFID reader on right

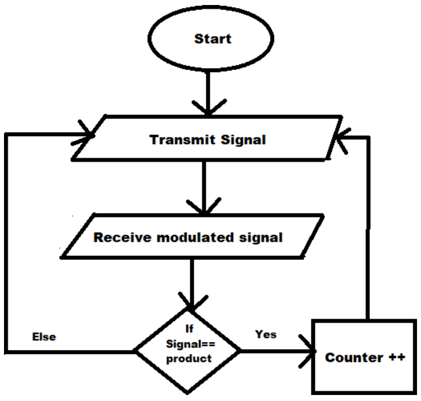


FIGURE 4.9: Algorithm for application for RFID

From the algorithm, it can be clearly observed that RFID reader when attached to an inventory system, it transmits signals continuously until it detects the required modulated signal. The product is incremented by one if the signal received is correct. This reader is attached to the door of the inventory rack and the item should pass through this reader. Based on this algorithm, the item will automatically get register in the inventory.

Load cell: Load cell is a type of force transducer which converts tension or compression or pressure or torque into a miserable electric signal. The strength of produced electrical signal is directly proportional to the force applied. There are many types of load cells available. Hydraulic load cell, pneumatic load cell, strain gauge load cell etc. The most commonly used type of load cell in industrial applications is the strain gauge load cell.
This type of load cell is accurate and cot effective. It comprised of solid metal body or spring element on which strain gauge has been secured. Its body is made of aluminium or alloy stell or stainless steel. But this body must be minimally elastic. When load is applied, the body is slightly deformed due to applied load. The body of load cell always returns to its original shape. When load is removed the body shape changes and also the stain gauge changes. This causes the change in electrical resistance of strain gauge which can be measured as a voltage change. The change in output is proportional to the load applied.

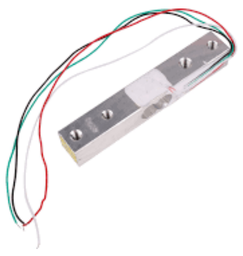


FIGURE 4.10: Load Cell

Strain gauge works on the principle of wheat stone bridge. When strain gauge is subjected to strain, their resistance changes by very small amount of stain. In order to measure the change strain accurately, the most popular circuit used is wheat stone bridge. This is a simple circuit where four resistors are connected end to end from a square having four corners. Across one pair of diagonal corners an excitation voltage is applied and across the other pair the output of the bridge is measured designated as V. V will be zero when the ratio R1/R3=R2/R4, where R1, R2, R3, R4 are the respective strain gauges in a load cell. The gauges are located in such a way that R1 and R4 experience tensile stress and R2 and R3 experience compressive stress. In this condition the sensitivity of the bridge is maximum.

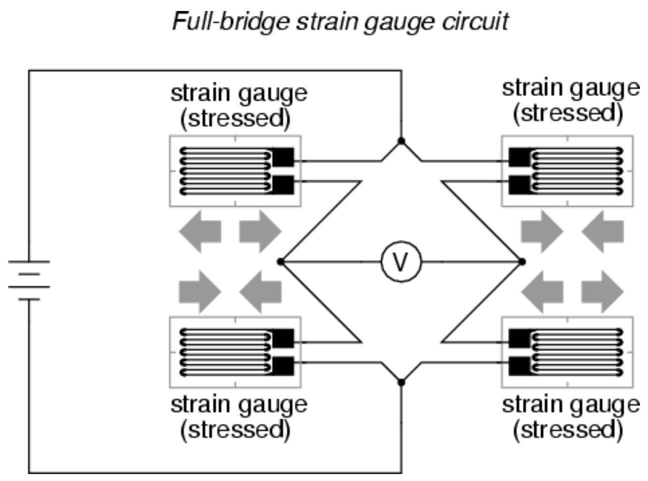


FIGURE 4.11: Full Bridge Strain Gauge Circuit

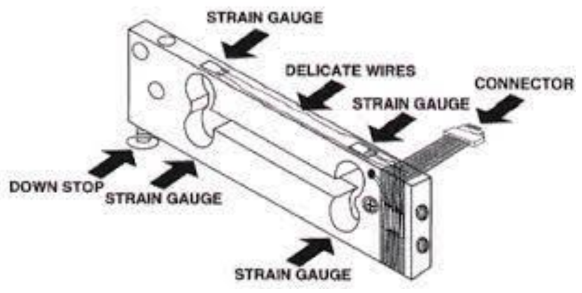


FIGURE 4.12: Load Cell Transducer

HX771: Since the voltage difference is tiny, we need HX711 amplifier module to make it readable for Arduino. HX771 is a analog to digital converter specifically used in weighing scale machines. It is a 24-bit precision ADC so that you can get more accurate value in terms of grams you can able to measure. HX711 is directly interfaced with a bridge sensor. It has two select-able differential input channels. Select-able output data rates from 10 sample per second to 80 samples per second. It requires no programming for internal registers. This module doesn't require any power supply and can easily interface to any micro controller to measure the weight. This chip is cost effective and also fast responsive and improves the total performance and reliability.

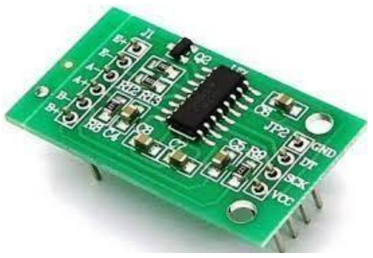
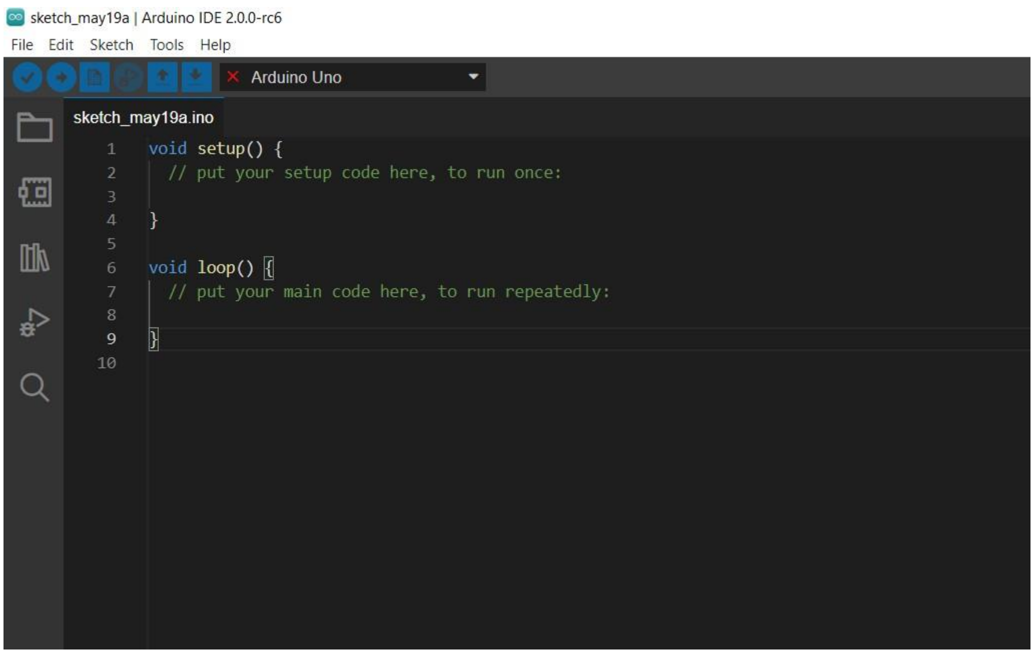


FIGURE 4.13: HX711

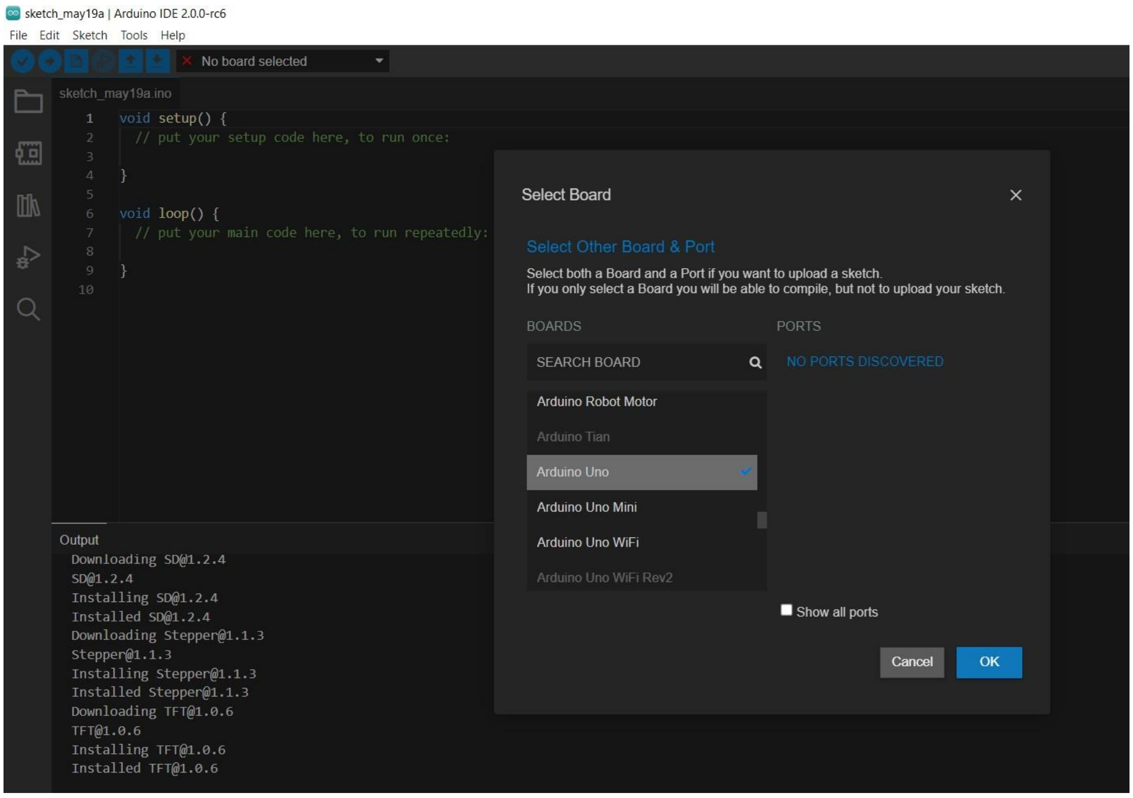
Arduino IDE: Arduino above all else is an open-source PC equipment and programming organization. The Arduino Local area alludes to the undertaking and client local area that plans and uses microcontroller-based advancement sheets. These advancement sheets are known as Arduino Modules, which are open-source prototyping stages. The rearranged microcontroller board arrives in an assortment of improvement board bundles. The most widely recognized programming approach is to utilize the Arduino IDE, which uses the C programming language. This gives you admittance to a tremendous Arduino Library that is continually developing because of opensource local area. Download Arduino Integrated Design Environment (IDE) here (Most late form: 1.6.5): https://www.arduino.cc/en/Main/Software This is the Arduino IDE whenever it's been opened. It opens into a clear sketch where you can begin programming right away. In the first place, we ought to arrange the board and port settings to permit us to transfer code. Interface your Arduino board to the PC through the USB cable.


sketch_may19a | Arduino IDE 2.0.0-rc6
File Edit Sketch Tools Help
Arduino Uno
sketch_may19a.ino

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

FIGURE 4.14: Arduino IDE default window

Board Setup: You need to tell the Arduino IDE what board you are transferring to. Select the Tools pulldown menu and go to Board. This rundown is populated naturally with the right now accessible Arduino Boards that are created by Arduino. In the event that you are utilizing an Uno or an Uno-Compatible Clone (ex. Funduino, SainSmart, IEIK, and so on), select Arduino Uno. On the off chance that you are utilizing another board/clone, select that board.



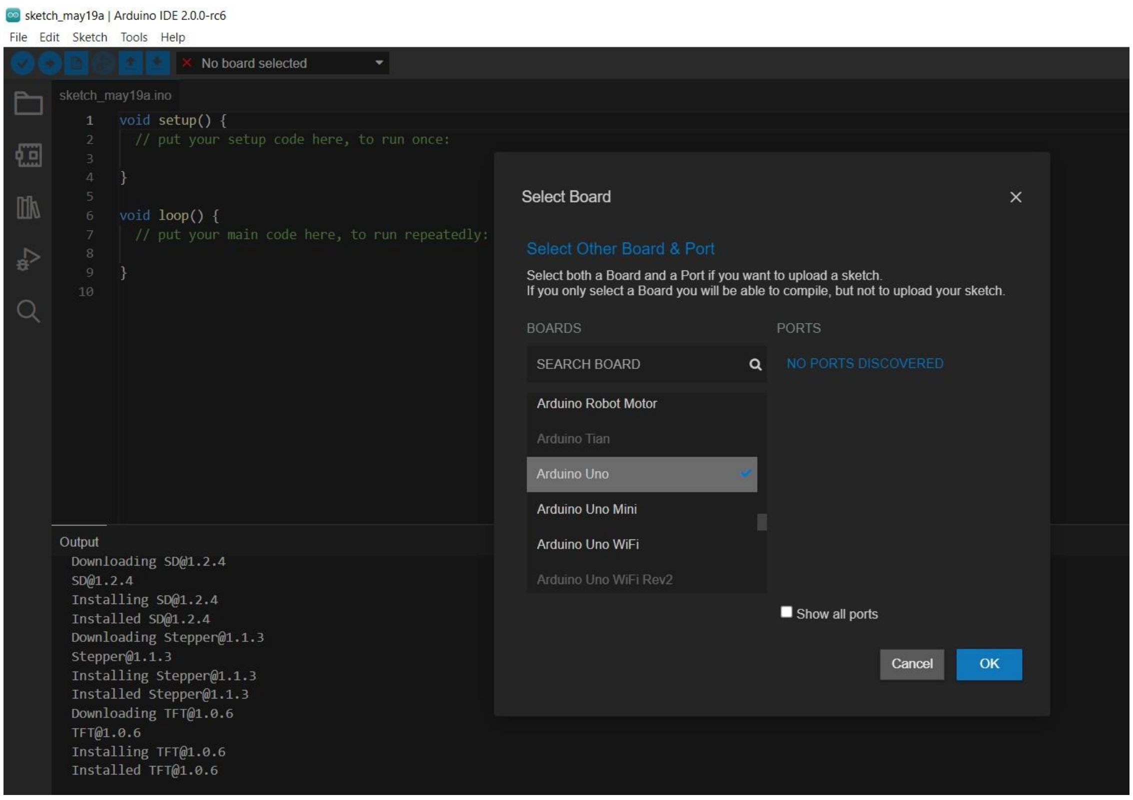
```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Select both a Board and a Port if you want to upload a sketch.
If you only select a Board you will be able to compile, but not to upload your sketch.

```
Downloading SD@1.2.4
SD@1.2.4
Installing SD@1.2.4
Installed SD@1.2.4
Downloading Stepper@1.1.3
Stepper@1.1.3
Installing Stepper@1.1.3
Installed Stepper@1.1.3
Downloading TFT@1.0.6
TFT@1.0.6
Installing TFT@1.0.6
Installed TFT@1.0.6
```

FIGURE 4.15: Board Setup

IDE: COM Port Setup In the event that you downloaded the Arduino IDE prior to connecting your Arduino board, when you connected the board, the USB drivers ought to have introduced consequently. The latest Arduino IDE ought to perceive associated sheets and name them with which COM port they are utilizing. Select the Tools pull-down menu and afterward Port. Here it should list all open COM ports, and if there is a perceived Arduino Board, it will likewise give its name. Select the Arduino board that you have associated with the PC. In the event that the arrangement was fruitful, in the base right of the Arduino IDE, you should see the board type and COM number of the board you intend to program. Note: the Arduino Uno involves the following accessible COM port; it won't generally be COM3.



sketch_may19a | Arduino IDE 2.0.0-rc6

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Select Board

Select Other Board & Port

Select both a Board and a Port if you want to upload a sketch.
If you only select a Board you will be able to compile, but not to upload your sketch.

BOARDS: Arduino Uno ✓

PORTS: NO PORTS DISCOVERED

```
Output
Downloading SD@1.2.4
SD@1.2.4
Installing SD@1.2.4
Installed SD@1.2.4
Downloading Stepper@1.1.3
Stepper@1.1.3
Installing Stepper@1.1.3
Installed Stepper@1.1.3
Downloading TFT@1.0.6
TFT@1.0.6
Installing TFT@1.0.6
Installed TFT@1.0.6
```

FIGURE 4.16: COM Port

4.2.5 Design

This project is designed to replace the old methods and operations involved in calculating stocks of item. Designing is a drawing or pattern that shows something to be done. Design is also called as purposeful; planning. It starts with an login method, which allows user to go into various area of the program. Thus, it provides a quick access to different area of the program.

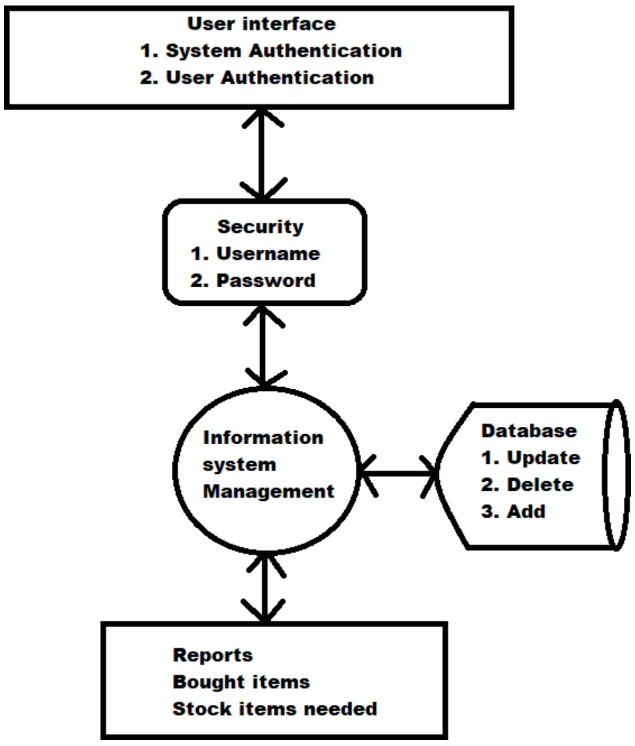


FIGURE 4.17: Architecture diagram for proposed system

UML Diagrams UML stands for Unified Modelling Language, is a general-purposed modelling language in the field of object-oriented software engineering. The goal of UML is to become a common language for creating models of object-oriented computer software. The advantage of UML is in the future some form of methods or process can also be added to it. It is a standard form of visualization, constructing and documenting the artefacts of the system. UML is very important for developing object oriented software and the software development process. Graphical notations to express the design of software projects is used in UML diagrams.
The advantages of UML design are:

1. It provides a easy understanding of modelling language.
2. UML encourages the growth of Object-oriented tools market.
3. UML gives ready-to-use and expressive modelling language so that user can develop and exchange meaningful models.
4. Integrates the best practices.

Data flow Diagram The Data Flow Diagram is also called as bubble chart. It is a basic graphical formalism that can be utilized to address a framework as far as the info information to the frame work, different handling completed on this information, and the yield information is created by the framework.

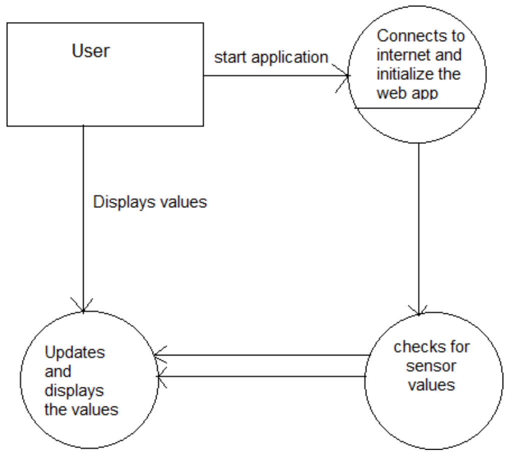

User
start application
Connects to internet and initialize the web app
Displays values
Updates and displays the values
checks for sensor values

FIGURE 4.18: Data flow diagram which depicts the data flow

The data flow diagram represents the flow of the data in the system. The above diagram consists of user, update and display values wi-fi and sensors, checking sensor values. The diagram shows that the user starts the application by connecting the wi-fi and initializing web app. Then check the sensor values and the values are displayed to the user. Data flow diagrams can be divided into logical and physical. The logical data flow diagram describes flow of data through a system to perform certain functionality of a business. The physical data flow diagram describes the implementation of the logical data flow.

Use Case Diagram Use case diagrams are usually referred as behavior diagrams which is used to describe a set of actions that some system or systems should or perform in collaboration with one or more external users of the system.each use case should provide some observable and result to the actors or other stakeholders of the system.

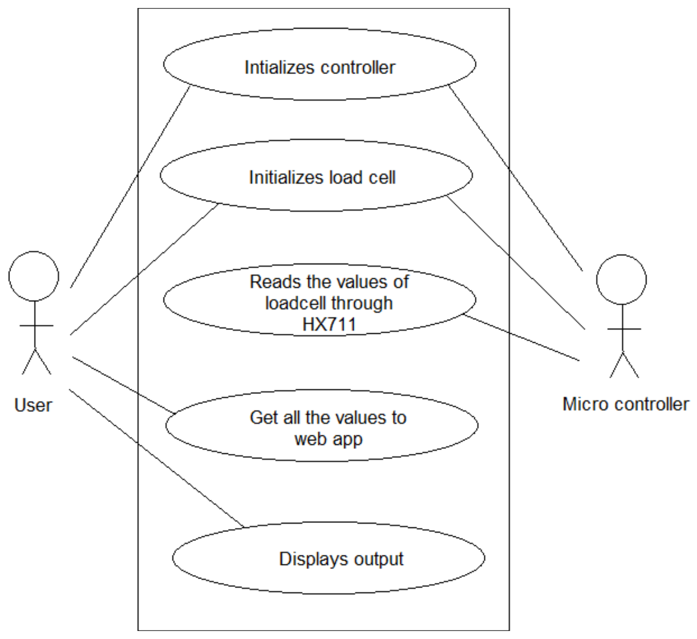


FIGURE 4.19: use case diagram which depicts the interaction between user and the system.

In the above use case diagram, there are two members namely user and micro controller. There are total five use cases that represent the specific functionality of the system. Each actor interacts with a particular use case. It is not necessary that each actor should interact with all the use cases, but it happens. These interaction between the user and micro controller together sums up the application.

Activity Diagram Activity diagram focuses on behaviour of a system instead of implementation. It is also called object-oriented flow chart. It is made up of actions which apply to behavioral modeling technology.

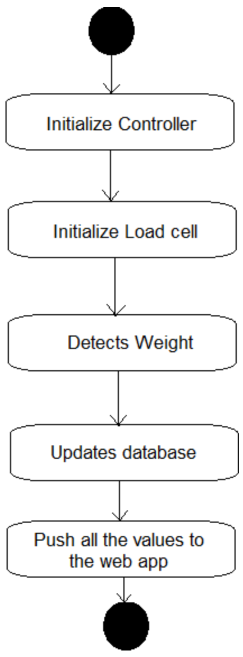


FIGURE 4.20: Activity diagram focuses on the execution and flow of the behavior of a system.

The above diagram shows the activity diagram. The activity diagram consists of the steps and process present in the system and how they are evaluated. It starts with initializing micro controller, initializing load cell and read values, detects weight, updates database, push all the values to server.

Sequence Diagram Sequence diagrams are interaction diagrams that detail how operations are carried out. They capture The interaction between objects in the context of a collaboration. Sequence diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are send and when the messages are sent. It is most commonly used interaction diagram. An interaction diagram is used to show the interactive behavior of the system. We use different types of interaction diagrams to capture various features and aspects of interaction in a system.

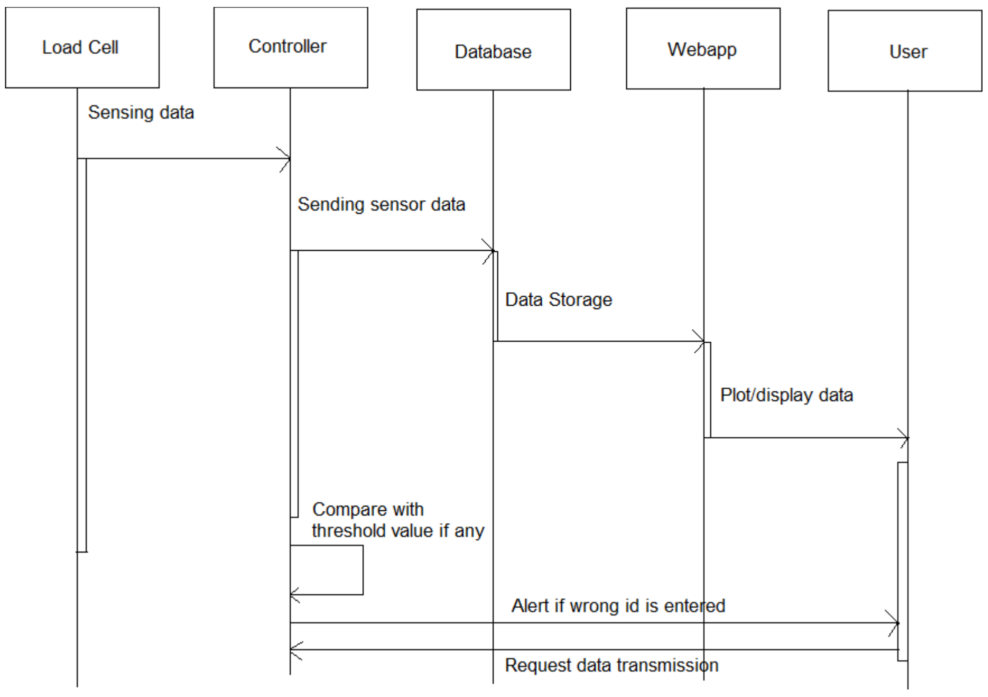


FIGURE 4.21: Sequence diagram depicts interaction between objects in a sequential order

The above sequence diagram shows an interaction among a set of objects participated in a collaboration arranged in a order. It shows the objects participating in the interaction by their lifelines and the messages that they send to each other. In the diagram, the actors are the load cell, controller, database, web app, user.
The load cell detects the weight and HX711 collects the data.
Controller sends the load cell data to database.
Database stores and sends the data which it got from controller to web app
Web app plots or displays the data to the user.
Web app requests controller for data transmission at the last step.

4.2.6 Database

The databases used in this project are:

```
CREATE TABLE users (id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
username TEXT NOT NULL, hash TEXT NOT NULL);
```

```
CREATE TABLE history(id INTEGER PRIMARY KEY AUTOINCREMENT,
‘user_id‘ INTEGER NOT NULL, ‘badge_number‘ TEXT, ‘product_name‘ TEXT,
amount INTEGER, ‘now_amount‘ INTEGER, ‘exp_date‘ DATE,
‘bought_date DATE‘, ‘last_used_time‘ TIME, ‘last_used_date‘ DATE,
status BOOL, ‘bought_how‘ TEXT,
FOREIGN KEY(‘user_id‘) REFERENCES users(id));

CREATE TABLE tag(id INTEGER PRIMARY KEY AUTOINCREMENT,
‘tag_id‘ TEXT NOT NULL, status BOOL, UNIQUE (‘tag_id‘));

CREATE TABLE items(id INTEGER PRIMARY KEY AUTOINCREMENT,
‘product_name‘ TEXT, cost INTEGER, weight INTEGER,
‘exp_date‘ DATE, UNIQUE (‘product_name‘));

CREATE TABLE ‘master_weight‘(id INTEGER PRIMARY KEY AUTOINCREMENT,
‘master_weight‘ INTEGER, code TEXT, UNIQUE(code));

CREATE TABLE data(id INTEGER PRIMARY KEY AUTOINCREMENT,
current INTEGER, privous INTEGER, reduce INTEGER, total INTEGER,
code TEXT, UNIQUE(code));

CREATE TABLE ‘old_weight‘(id INTEGER PRIMARY KEY AUTOINCREMENT,
‘old_weight‘ INTEGER, code TEXT, UNIQUE(code));

CREATE TABLE ‘scaned_tag‘(id INTEGER PRIMARY KEY AUTOINCREMENT,
‘scaned_tag‘ TEXT, ‘product_code‘ INTEGER, UNIQUE (‘scaned_tag‘));

CREATE TABLE products(id INTEGER PRIMARY KEY AUTOINCREMENT,
‘product_name‘ TEXT, pin INTEGER, UNIQUE(pin));

CREATE TABLE transactions(id INTEGER PRIMARY KEY AUTOINCREMENT,
‘user_id‘ INTEGER NOT NULL, ‘badge_number‘ TEXT, ‘product_name‘ TEXT,
amount INTEGER, ‘now_amount‘ INTEGER, ‘exp_date‘ DATE,
‘bought_date‘ DATE, ‘last_used_time‘ TIME, ‘last_used_date DATE‘,
status TEXT, ‘bought_how‘ TEXT,
FOREIGN KEY(‘user_id‘)  REFERENCES  users(id));

CREATE TABLE repository(id INTEGER PRIMARY KEY AUTOINCREMENT,
‘product_weight‘ INTEGER, ‘product_total‘ INTEGER,
‘product_rfid‘ TEXT, ‘product_code‘ INTEGER,
‘last_used_date_time‘  DATE,  UNIQUE  (‘product_rfid‘));
```

4.3 Implementation Details

The automated inventory system is integrated with RFID, load cell, HX711, node MCU. The RFID reader detects the RFID tag when the item is weighed on load cell. This load cell sends the data to nodeMCU through HX711. Our system was developed using application software i.e., SQLite which was used in the creation of a database and tables. HTML CSS, Hyper Text Markup Language Cascading Style Sheets used to design user interface. JSON which means Java Script Object Notation used to pass information from Arduino Uno to NodeMCU. Flask server is used to host our web app.

4.4 Prototype & testing

The main purpose of testing is to identify errors. Testing is the process of discovering any faults or weakness in a product. It is a way of checking whether the functionality of components, assemblies and sub assemblies in a product are finished or not. Testing is made to ensure that the software system meets its requirement and user expectations doesn’t fail in any circumstances.
Test Objectives

- All field entries must work properly.
- Pages must be activated from the identifies link.
- The entry screen, response must not be delayed.
- Verify that the entries are of correct format.
- No duplicate entries should be allowed.

Test case	1
Name of Test	RFID reader
Input	RFID Tag
Expected Output	Tag number
Actual Output	Tag number
Result	successful

TABLE 4.1: RFID Reader Test Case

Test case	2
Name of Test	Load Cell
Input	Weight of an item
Expected Output	Actual weight of an item
Actual Output	Accurate weight of an item
Result	successful

TABLE 4.2: Load Cell Test Case

Test case	3
Name of Test	LED
Input	Electric signal from controller port.
Expected Output	Proving light output
Actual Output	An external observation is done and LEDs glow as per the program logic.
Result	successful

TABLE 4.3: LED Test Case

Chapter 5

RESULTS, DISCUSSIONS AND CONCLUSIONS

Here, the results of the project work (literature survey and review along with actual work) shall be listed and discussed in detail with appropriate arguments (result analysis) leading to logical conclusions. The list of conclusions should sync with the project objectives. The scope for future research and development in the field of the current project work must also be included in this chapter.

5.1 Results & Analysis

We have tested the inventory system after the design and programming were completed. The inventory system is working well. We have also considered all the scenarios that occur and it was successful. In the present system, each packet/bottle will be tagged with an RFID/QR code, when it goes to the rack. The item is identified and its volume/weight and number are recorded using load sensor attached to the bottom of the rack. Now using this system, the incoming and outgoing movement of the product can be monitored and recorded in the local computer. These local computers are networked at various levels and finally linked to the global HQ. An AI or a proper software (AI based) can monitor the rate of movement of various products and predict the requirements to replenish the item. This can alarm the product managers to make requirements to get various items from a remotely located store house so that it can be supplied to the stores located at various parts of city/town, well before any item gets exhausted.

5.2 Comparative Study

We have incorporated the current methods and technologies referred to as described in the literature part. We have tried to implement new ideas into our project. In this module the new technology RFID is implemented for scanning the ID and weight of an item.

SL.NO	FEATURES	EXISTING TECHNOLOGIES	OUR PROJECT
1	RFID	No RFID is used in existing systems	RFID is used
2	Human Power	Human power is required to store the item details in database	Item details are automatically stored in database
3	User Payments	Doesn’t have any support on user payments	User payment possible using NFC technology
4	Tags	Not possible to give unique tag to each product	Unique tag is given to each product

TABLE 5.1: Comparison analysis

5.3 Discussions

This project will reduce the problems with the manual inventory system. It can also be used as a template to develop the real-life database system for various supermarkets. Our project can be served as a reference to other researchers and interested parties. Information system plays major role in managing data in organization. In organizations that use inventory, this system will reduce your time lead, increases your work speed and completely avoids misalignment from your process, eliminates variation in inventory replenish based on the market demand. This project will help managers to know what items and how much amount of stock items left in the rack

5.4 Cost Estimation Model

The cost of automated inventory system will depend on different parameters which includes no.of users who needs to access to the system, functionalities off modules in terms of internal processes, no.of products to be managed and hardware integration. In this project we have used a small RFID which was Rs.120 and it is affordable. In case of very large industries, different type of RFID should be use which may cost of around Rs. 50,000.

5.5 Conclusions

The above system and software can be used to manage the inventory of grocery and vegetable stock in a family. This system can be linked to the network of any supermarket chains of the customer. Here the information about emptying of the store and the probable requirement of various items can be identified by the network system and at any time the customer can be warned by messaging. The store can be ready with the items that can be delivered on confirmation of the customer order.

5.6 Scope for Future Work

The present system is a preliminary logical hardware display of the basic system required for establishing a global network of store management which is a very complex system that requires innovation in the product ID system (RFID) and their readers and also the weight/number monitoring system. The product ID system has to be cheap and of enough range and it has to record fat enough. Similarly, the weight/number monitoring system has to be accurate, rigid, stable and cost efficient. The software has to incorporate various customer-based parameters and algorithms like data optimisation (e.g.: travelling salesman) needs to be implemented, so that tracking is easier. This same system can be adopted for a domestic inventory management that can be linked to a network of stores.

Bibliography

[1] Xiajun Jing, Peng Tang, *Research and Design of the Intelligent Inventory Management System Based on RFID,* Sixth International Symposium on Computational Intelligence and Design, 2013.

[2] S. Jayanth, M.B. Poorvi, M.P. Sunil, *Inventory Management System using IoT.*, Springer Science Business Media Singapore, 2017

[3] M.K. Sangole, Bhushan S. Nasikkar, Dhananjay V. Kulkarni, Gitesh K. Kakuste, *Smart Refrigerator using Internet of Things (IoT)* .International Journal of Advance Research, Ideas and Innovations in Technology, Vol.3, Issue.1

[4] Murali.N.G, Aarthi.S, Ethiraj.M, Baghavathi Priya.S, *IoT Based Interactive Smart Refrigerator,* 3rd International Conference on Computers and Management (ICCM 2017)

[5] Subero, A *Serial communication protocols. In: Programming PIC Microcontrollers with XC8.* Apress, Berkeley, CA (2018)

[6] McGrath, M.J., Scanaill*Sensing and sensor fundamentals. In: Sensor Technologies.*, Apress, Berkeley, CA (2013)

[7] Kakeeto.K, *Project report entitled Inventory Management System for Graphic systems Uganda limited.*.(2003)

[8] Wallin C,, Rungtusanatham, M.J.,and Rabinovich, *what is the right inventory management approach for a purchased item?,* International Journal of Operations and Production Management, Bradford: Vol 26. issue 1/2, pg 50,19 pgs

[9] YingBai*Practical Database Programming with visualbasic.net 2nd edition*John Wiley Sons Inc., Publications, 2012.

[10] *Interface Circuits Data Book*, Texas Instruments, Austin, Texas, 1993.

Appendix A

Appendix

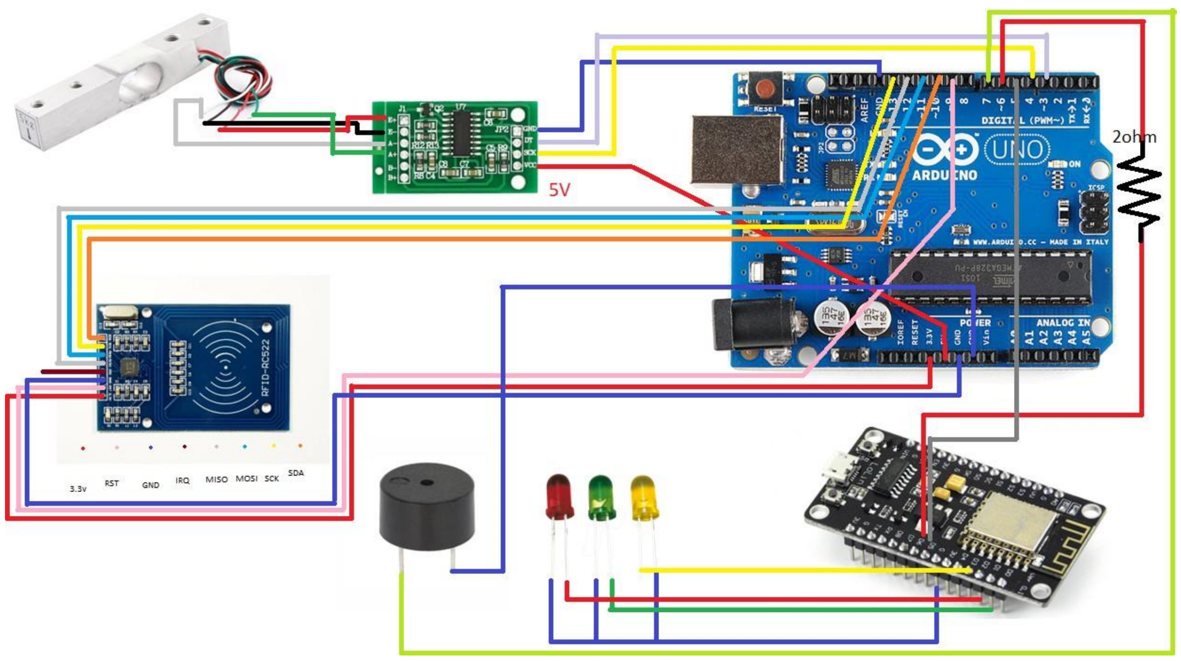


FIGURE A.1: Pin Diagram

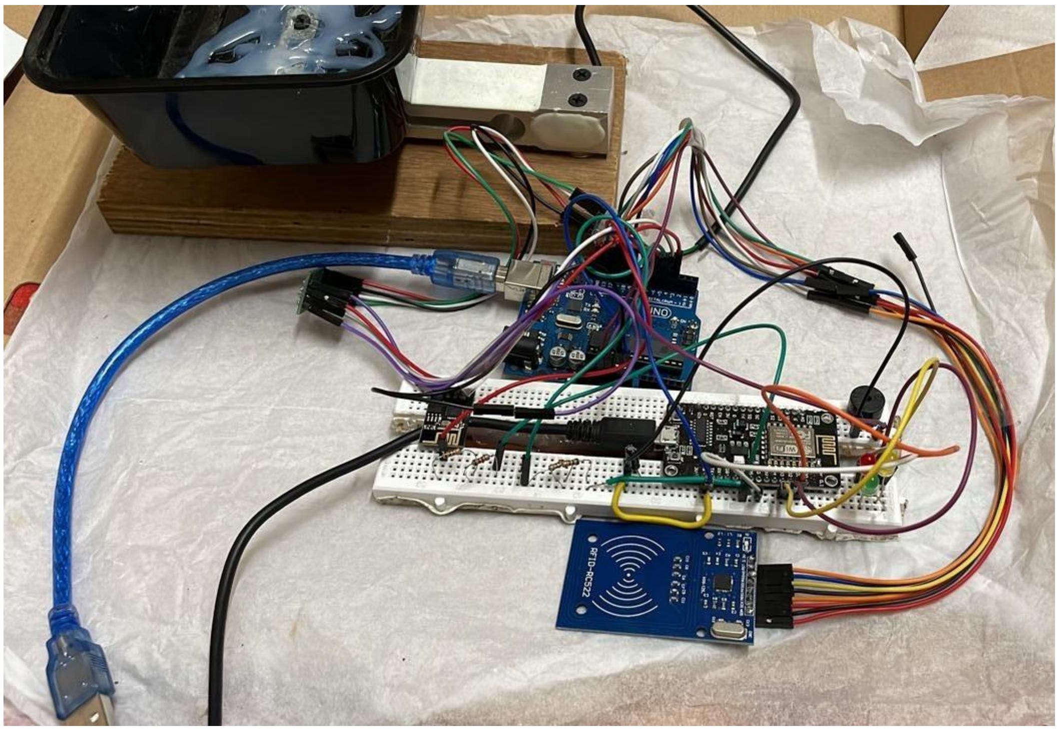


FIGURE A.2: Product Image

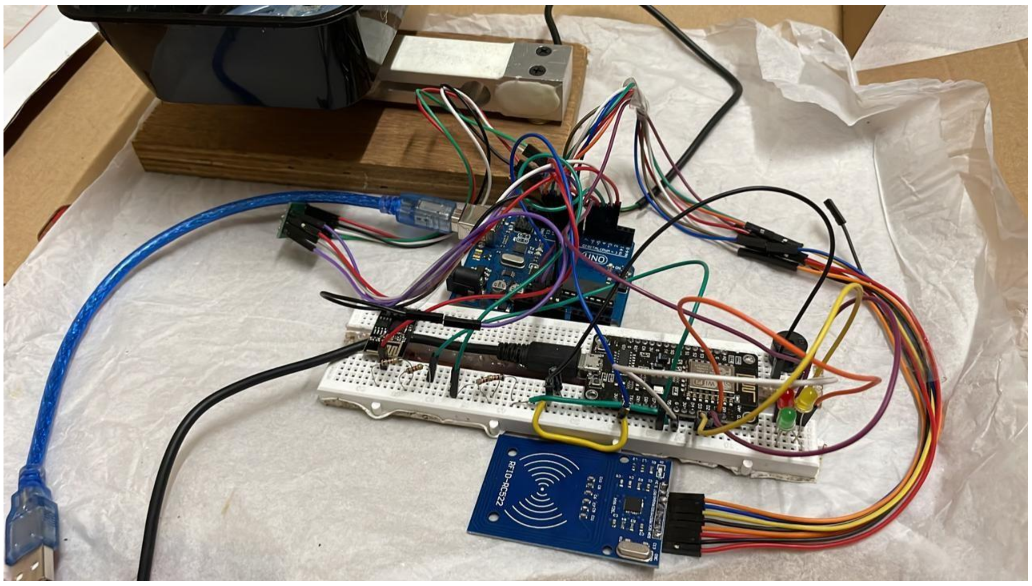
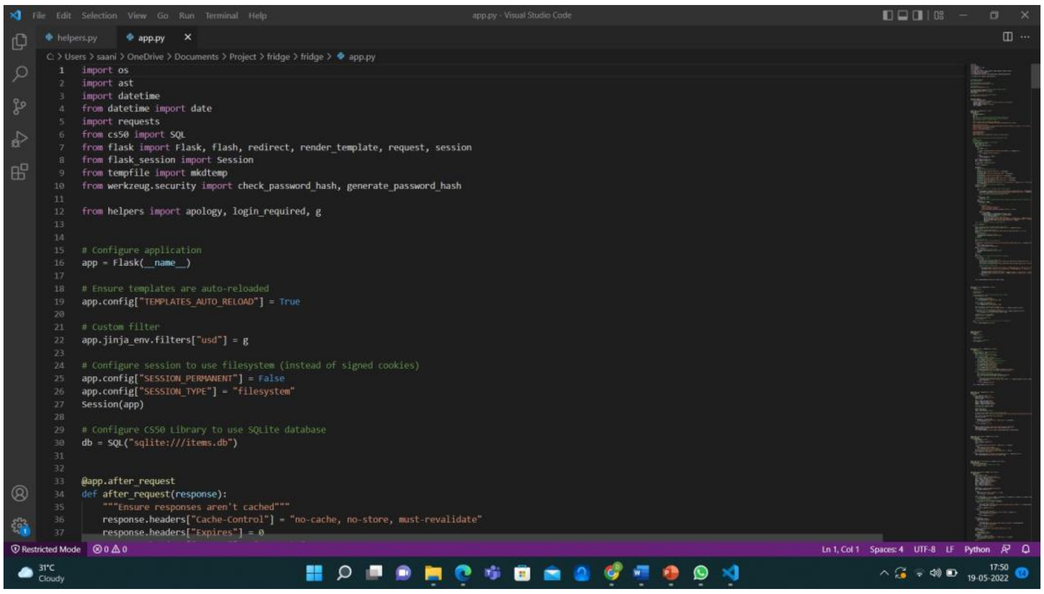


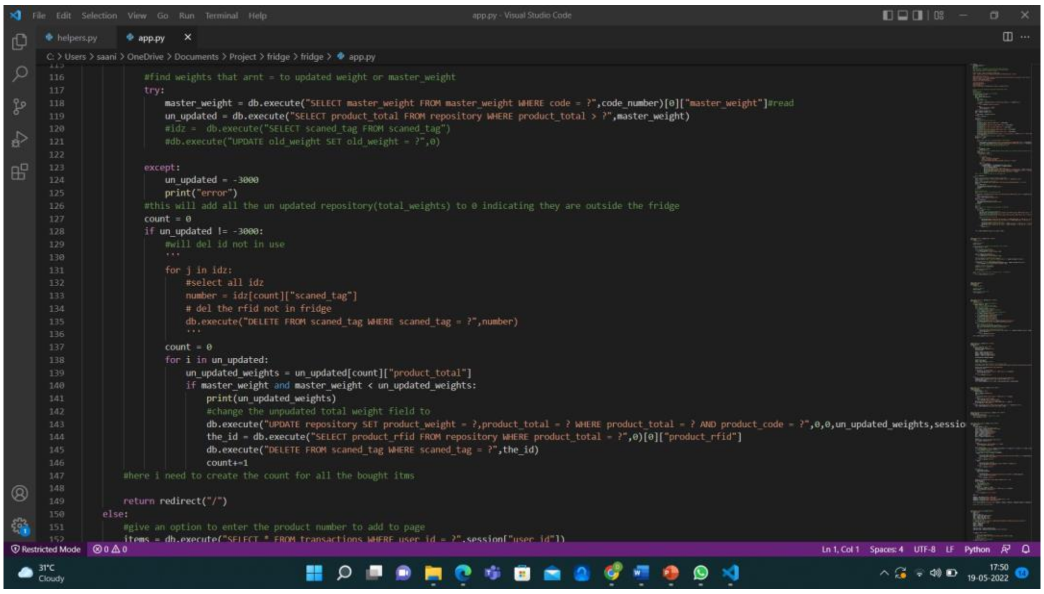
FIGURE A.3: Product Image

A.1 Web App Program Inclusion



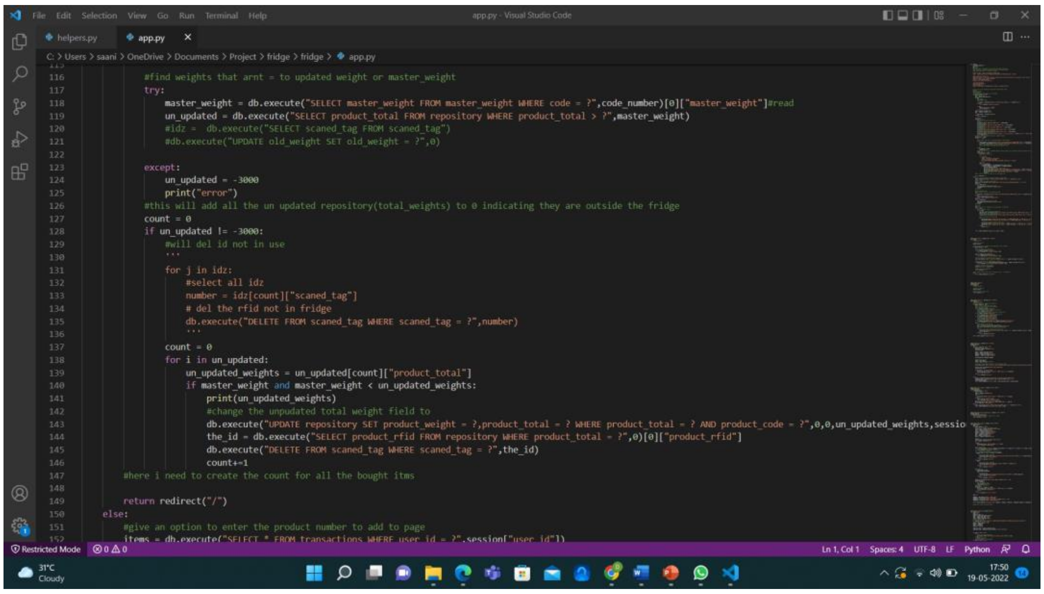
```python
import os
import ast
import datetime
from datetime import date
import requests
from cs50 import SQL
from flask import Flask, flash, redirect, render_template, request, session
from flask_session import Session
from tempfile import mkdtemp
from werkzeug.security import check_password_hash, generate_password_hash

from helpers import apology, login_required, g


# Configure application
app = Flask(__name__)

# Ensure templates are auto-reloaded
app.config["TEMPLATES_AUTO_RELOAD"] = True

# Custom filter
app.jinja_env.filters["usd"] = g

# Configure session to use filesystem (instead of signed cookies)
app.config["SESSION_PERMANENT"] = False
app.config["SESSION_TYPE"] = "filesystem"
Session(app)

# Configure CS50 Library to use SQLite database
db = SQL("sqlite:///items.db")


@app.after_request
def after_request(response):
    """Ensure responses aren't cached"""
    response.headers["Cache-Control"] = "no-cache, no-store, must-revalidate"
    response.headers["Expires"] = 0
```

FIGURE A.4: Web app program



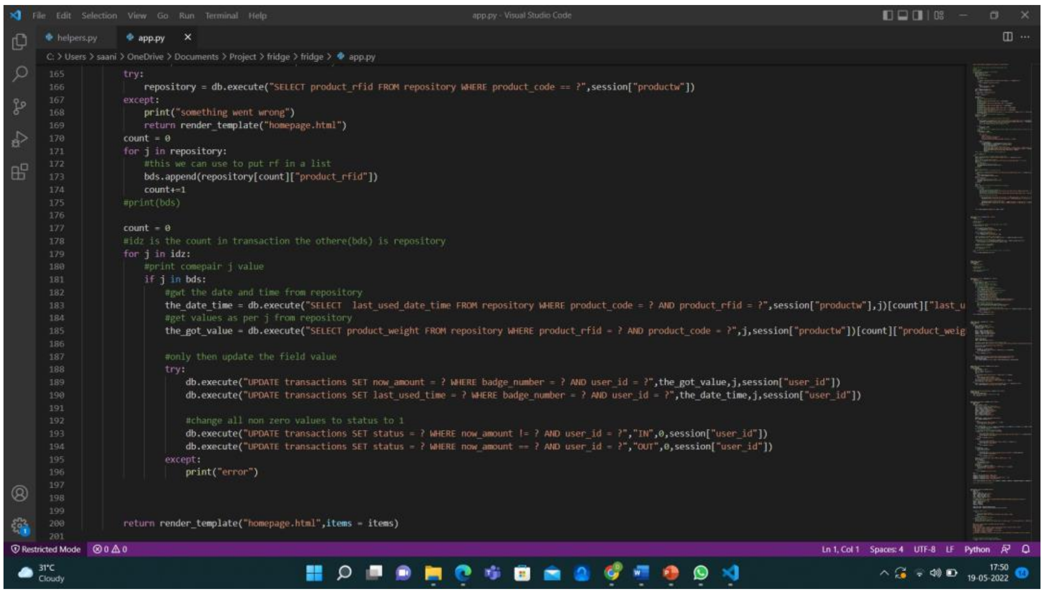
```python
                #find weights that arnt = to updated weight or master_weight
                try:
                    master_weight = db.execute("SELECT master_weight FROM master_weight WHERE code = ?",code_number)[0]["master_weight"]#read
                    un_updated = db.execute("SELECT product_total FROM repository WHERE product_total > ?",master_weight)
                    #idz =  db.execute("SELECT scaned_tag FROM scaned_tag")
                    #db.execute("UPDATE old_weight SET old_weight = ?",0)

                except:
                    un_updated = -3000
                    print("error")
                #this will add all the un updated repository(total_weights) to 0 indicating they are outside the fridge
                count = 0
                if un_updated != -3000:
                    #will del id not in use
                    '''
                    for j in idz:
                        #select all idz
                        number = idz[count]["scaned_tag"]
                        # del the rfid not in fridge
                        db.execute("DELETE FROM scaned_tag WHERE scaned_tag = ?",number)
                        '''
                    count = 0
                    for i in un_updated:
                        un_updated_weights = un_updated[count]["product_total"]
                        if master_weight and master_weight < un_updated_weights:
                            print(un_updated_weights)
                            #change the unpudated total weight field to
                            db.execute("UPDATE repository SET product_weight = ?,product_total = ? WHERE product_total = ? AND product_code = ?",0,0,un_updated_weights,sessio
                            the_id = db.execute("SELECT product_rfid FROM repository WHERE product_total = ?",0)[0]["product_rfid"]
                            db.execute("DELETE FROM scaned_tag WHERE scaned_tag = ?",the_id)
                            count+=1
            #here i need to create the count for all the bought itms

            return redirect("/")
        else:
            #give an option to enter the product number to add to page
            items = db.execute("SELECT * FROM transactions WHERE user_id = ?",session["user_id"])
```

FIGURE A.5: web app program



```python
                #find weights that arnt = to updated weight or master_weight
                try:
                    master_weight = db.execute("SELECT master_weight FROM master_weight WHERE code = ?",code_number)[0]["master_weight"]#read
                    un_updated = db.execute("SELECT product_total FROM repository WHERE product_total > ?",master_weight)
                    #idz =  db.execute("SELECT scaned_tag FROM scaned_tag")
                    #db.execute("UPDATE old_weight SET old_weight = ?",0)

                except:
                    un_updated = -3000
                    print("error")
                #this will add all the un updated repository(total_weights) to 0 indicating they are outside the fridge
                count = 0
                if un_updated != -3000:
                    #will del id not in use
                    '''
                    for j in idz:
                        #select all idz
                        number = idz[count]["scaned_tag"]
                        # del the rfid not in fridge
                        db.execute("DELETE FROM scaned_tag WHERE scaned_tag = ?",number)
                        '''
                    count = 0
                    for i in un_updated:
                        un_updated_weights = un_updated[count]["product_total"]
                        if master_weight and master_weight < un_updated_weights:
                            print(un_updated_weights)
                            #change the unpudated total weight field to
                            db.execute("UPDATE repository SET product_weight = ?,product_total = ? WHERE product_total = ? AND product_code = ?",0,0,un_updated_weights,sessio
                            the_id = db.execute("SELECT product_rfid FROM repository WHERE product_total = ?",0)[0]["product_rfid"]
                            db.execute("DELETE FROM scaned_tag WHERE scaned_tag = ?",the_id)
                            count+=1
            #here i need to create the count for all the bought itms

            return redirect("/")
        else:
            #give an option to enter the product number to add to page
            items = db.execute("SELECT * FROM transactions WHERE user_id = ?",session["user_id"])
```

FIGURE A.6: web app program



```python
            try:
                repository = db.execute("SELECT product_rfid FROM repository WHERE product_code == ?",session["productw"])
            except:
                print("something went wrong")
                return render_template("homepage.html")
            count = 0
            for j in repository:
                #this we can use to put rf in a list
                bds.append(repository[count]["product_rfid"])
                count+=1
            #print(bds)

            count = 0
            #idz is the count in transaction the othere(bds) is repository
            for j in idz:
                #print comepair j value
                if j in bds:
                    #gwt the date and time from repository
                    the_date_time = db.execute("SELECT  last_used_date_time FROM repository WHERE product_code = ? AND product_rfid = ?",session["productw"],j)[count]["last_u
                    #get values as per j from repository
                    the_got_value = db.execute("SELECT product_weight FROM repository WHERE product_rfid = ? AND product_code = ?",j,session["productw"])[count]["product_weig

                    #only then update the field value
                    try:
                        db.execute("UPDATE transactions SET now_amount = ? WHERE badge_number = ? AND user_id = ?",the_got_value,j,session["user_id"])
                        db.execute("UPDATE transactions SET last_used_time = ? WHERE badge_number = ? AND user_id = ?",the_date_time,j,session["user_id"])

                        #change all non zero values to status to 1
                        db.execute("UPDATE transactions SET status = ? WHERE now_amount != ? AND user_id = ?","IN",0,session["user_id"])
                        db.execute("UPDATE transactions SET status = ? WHERE now_amount == ? AND user_id = ?","OUT",0,session["user_id"])
                    except:
                        print("error")



            return render_template("homepage.html",items = items)
```

FIGURE A.7: web app program

A.1.1 Output

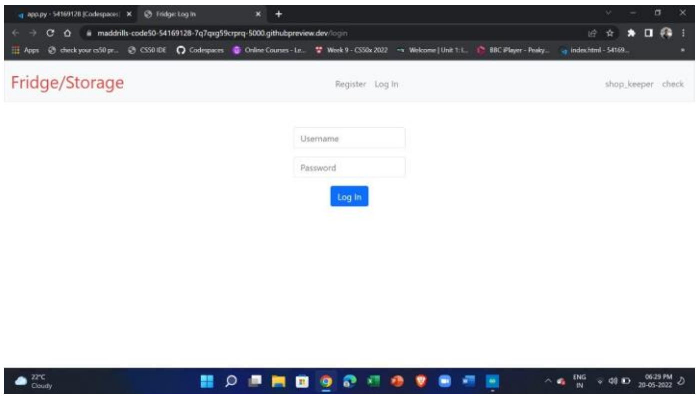


FIGURE A.8: User login

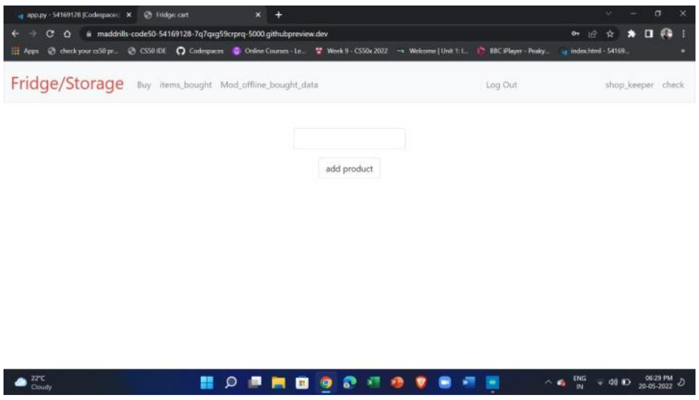
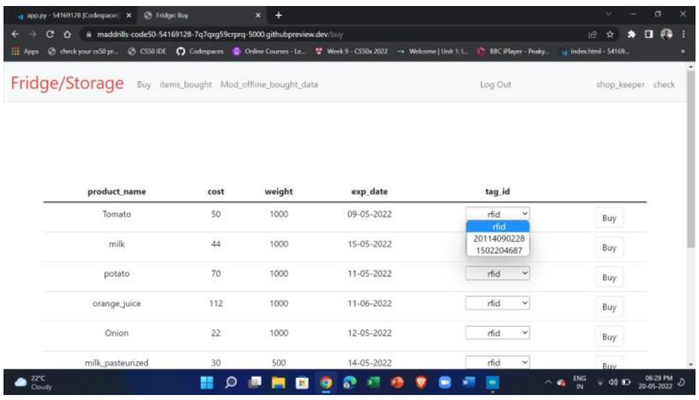
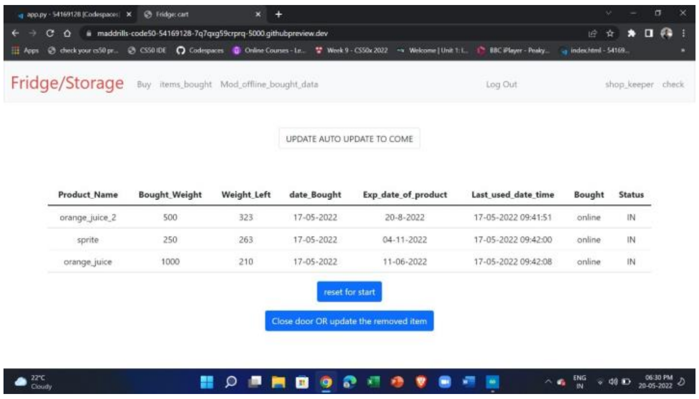


FIGURE A.9: Product column



product_name	cost	weight	exp_date	tag_id	
Tomato	50	1000	09-05-2022	rfid	Buy
milk	44	1000	15-05-2022		Buy
potato	70	1000	11-05-2022	rfid	Buy
orange_juice	112	1000	11-06-2022	rfid	Buy
Onion	22	1000	12-05-2022	rfid	Buy
milk_pasteurized	30	500	14-05-2022	rfid	Buy

FIGURE A.10: Items in the Store



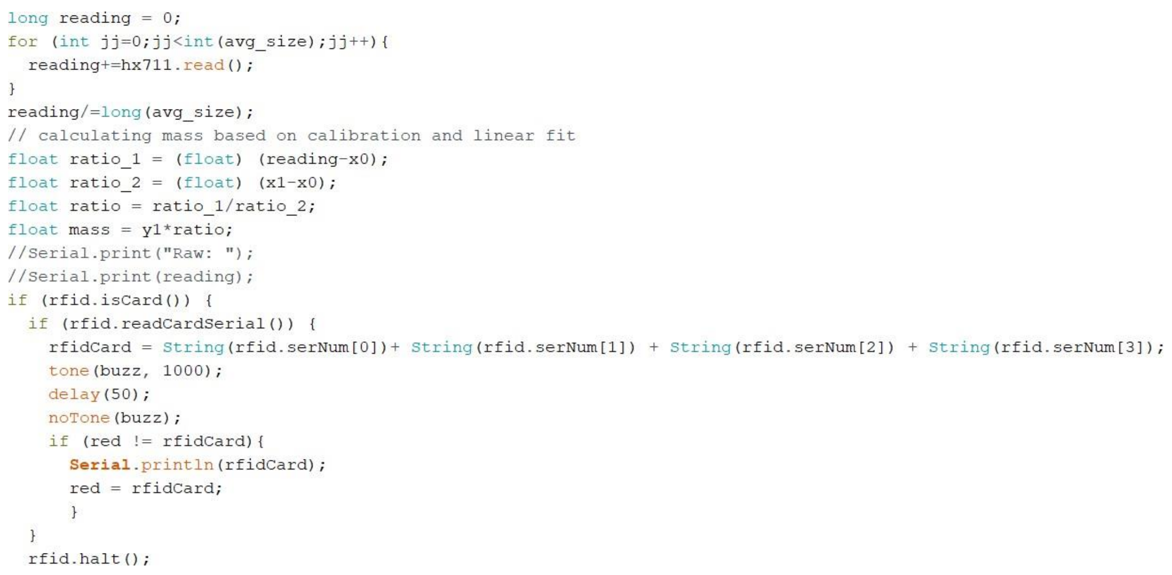
Product_Name	Bought_Weight	Weight_Left	date_Bought	Exp_date_of_product	Last_used_date_time	Bought	Status
orange_juice_2	500	323	17-05-2022	20-8-2022	17-05-2022 09:41:51	online	IN
sprite	250	263	17-05-2022	04-11-2022	17-05-2022 09:42:00	online	IN
orange_juice	1000	210	17-05-2022	11-06-2022	17-05-2022 09:42:08	online	IN

FIGURE A.11: Items Bought

Appendix B

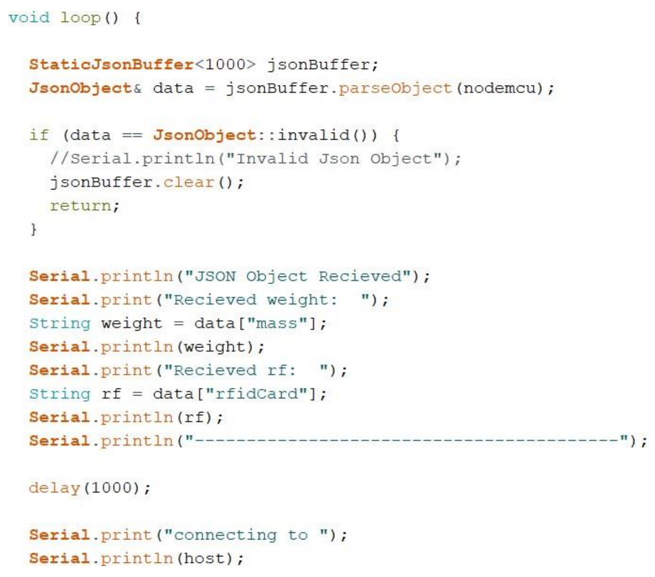
Appendix B

B.1 Arduino Programming



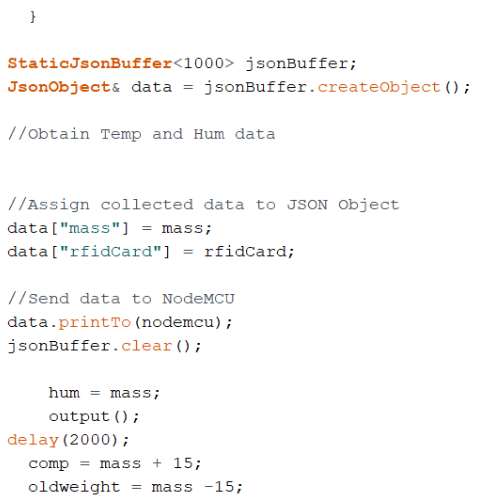
```
long reading = 0;
for (int jj=0;jj<int(avg_size);jj++){
  reading+=hx711.read();
}
reading/=long(avg_size);
// calculating mass based on calibration and linear fit
float ratio_1 = (float) (reading-x0);
float ratio_2 = (float) (x1-x0);
float ratio = ratio_1/ratio_2;
float mass = y1*ratio;
//Serial.print("Raw: ");
//Serial.print(reading);
if (rfid.isCard()) {
  if (rfid.readCardSerial()) {
    rfidCard = String(rfid.serNum[0])+ String(rfid.serNum[1]) + String(rfid.serNum[2]) + String(rfid.serNum[3]);
    tone(buzz, 1000);
    delay(50);
    noTone(buzz);
    if (red != rfidCard){
      Serial.println(rfidCard);
      red = rfidCard;
      }
  }
  rfid.halt();
```

FIGURE B.1: get hx711 weight data and rfid data



```
void loop() {

  StaticJsonBuffer<1000> jsonBuffer;
  JsonObject& data = jsonBuffer.parseObject(nodemcu);

  if (data == JsonObject::invalid()) {
    //Serial.println("Invalid Json Object");
    jsonBuffer.clear();
    return;
  }

  Serial.println("JSON Object Recieved");
  Serial.print("Recieved weight:  ");
  String weight = data["mass"];
  Serial.println(weight);
  Serial.print("Recieved rf:  ");
  String rf = data["rfidCard"];
  Serial.println(rf);
  Serial.println("-----------------------------------------");

  delay(1000);

  Serial.print("connecting to ");
  Serial.println(host);
```

FIGURE B.2: get data from uno and as JSON



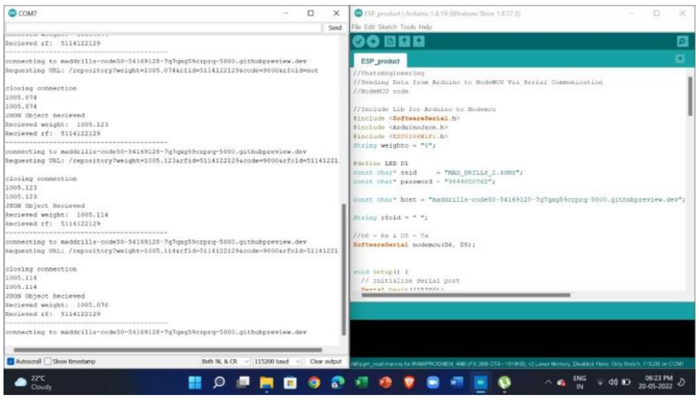
```
    }

  StaticJsonBuffer<1000> jsonBuffer;
  JsonObject& data = jsonBuffer.createObject();

  //Obtain Temp and Hum data


  //Assign collected data to JSON Object
  data["mass"] = mass;
  data["rfidCard"] = rfidCard;

  //Send data to NodeMCU
  data.printTo(nodemcu);
  jsonBuffer.clear();

      hum = mass;
      output();
  delay(2000);
    comp = mass + 15;
    oldweight = mass -15;
```

FIGURE B.3: send data to mcu from uno

```
    Serial.print("Requesting URL: ");
    Serial.println(url);

    // This will send the request to the server
    client.print(String("POST ") + url + " HTTP/1.1\r\n" +
                 "Host: " + host + "\r\n" +
                 "Connection: close\r\n\r\n");

    Serial.println();
    Serial.println("closing connection");

    digitalWrite(LED,HIGH);
    delay(50);
    digitalWrite(LED,LOW);

    rfold = rf;
    weighto = weight;
    Serial.println(weighto);
    Serial.println(weight);
  }
```

FIGURE B.4: sent as a post request to server

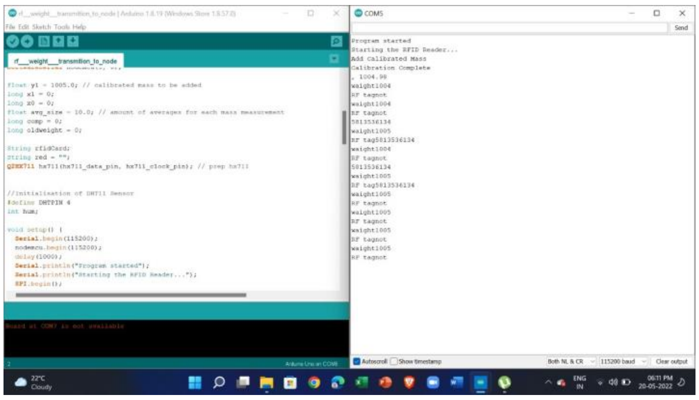
B.1.1 Output



```
Recieved rf:  5114122129
----------------------------------------
connecting to maddrills-code50-54169128-7q7qxg59crprq-5000.githubpreview.dev
Requesting URL: /repository?weight=1005.074&rfid=5114122129&code=9800&rfold=not

closing connection
1005.074
1005.074
JSON Object Recieved
Recieved weight:  1005.123
Recieved rf:  5114122129
----------------------------------------
connecting to maddrills-code50-54169128-7q7qxg59crprq-5000.githubpreview.dev
Requesting URL: /repository?weight=1005.123&rfid=5114122129&code=9800&rfold=51141221

closing connection
1005.123
1005.123
JSON Object Recieved
Recieved weight:  1005.114
Recieved rf:  5114122129
----------------------------------------
connecting to maddrills-code50-54169128-7q7qxg59crprq-5000.githubpreview.dev
Requesting URL: /repository?weight=1005.114&rfid=5114122129&code=9800&rfold=51141221

closing connection
1005.114
1005.114
JSON Object Recieved
Recieved weight:  1005.076
Recieved rf:  5114122129
----------------------------------------
connecting to maddrills-code50-54169128-7q7qxg59crprq-5000.githubpreview.dev
```

```
//ThatsEngineering
//Sending Data from Arduino to NodeMCU Via Serial Communication
//NodeMCU code

//Include Lib for Arduino to Nodemcu
#include <SoftwareSerial.h>
#include <ArduinoJson.h>
#include <ESP8266WiFi.h>
String weighto = "0";

#define LED D1
const char* ssid     = "MAD_DRILLS_2.4GHz";
const char* password = "9449050762";

const char* host = "maddrills-code50-54169128-7q7qxg59crprq-5000.githubpreview.dev";

String rfold = " ";

//D6 = Rx & D5 = Tx
SoftwareSerial nodemcu(D6, D5);


void setup() {
  // initialize serial port
```

FIGURE B.5: Arduino output on Serial Monitor



```
float y1 = 1005.0; // calibrated mass to be added
long x1 = 0;
long x0 = 0;
float avg_size = 10.0; // amount of averages for each mass measurement
long comp = 0;
long oldweight = 0;

String rfidCard;
String red = "";
Q2HX711 hx711(hx711_data_pin, hx711_clock_pin); // prep hx711


//Initialization of DHT11 Sensor
#define DHTPIN 4
int hum;

void setup() {
  Serial.begin(115200);
  nodemcu.begin(115200);
  delay(1000);
  Serial.println("Program started");
  Serial.println("Starting the RFID Reader...");
  SPI.begin();
```

```
Board at COM7 is not available
```

```
Program started
Starting the RFID Reader...
Add Calibrated Mass
Calibration Complete
, 1004.98
waight1004
RF tagnot
waight1004
RF tagnot
5813536134
waight1005
RF tag5813536134
waight1004
RF tagnot
5813536134
waight1005
RF tag5813536134
waight1005
RF tagnot
waight1005
RF tagnot
waight1005
RF tagnot
waight1005
RF tagnot
```

rf__weight__transmition_to_node | Arduino 1.8.19 (Windows Store 1.8.57.0)
File Edit Sketch Tools Help
rf__weight__transmition_to_node
Arduino Uno on COM8
COM5
Send
Autoscroll
Show timestamp
Both NL & CR
115200 baud
Clear output
22°C
Cloudy
ENG
IN
06:11 PM
20-05-2022

FIGURE B.6: Arduino output on Serial Monitor