IoT- BASED SMART SHOPPING CART

Project Report

<u>Project by:</u>

Mitanshu Kumar

Electronics and Communication Engg.

Samrat Ashok Technological Institute, Vidisha, M.P.

https://linkedin.com/in/mitanshukr
https://github.com/mitanshukr

Suggestions and Corrections are always welcome.
Hope you find this report helpful.

Intentionally left blank

Index

Table of Contents

Abstract

1. Abstract

One of the major challenges of the Offline shopping industry is crowd control, especially long non-ending queues on billing counters. Companies and merchants lose huge sales potential due to the queues on the Billing counters; also it imprints a bad shopping experience on the customers as well.

Our smart IoT enabled shopping cart is well developed to handle this issue and ease out this scenario for customers as well as sellers.
It majorly works on Radio frequency signals to track the purchasing products using RFID System (RC- 522) and IEEE 802.11 based Wi-Fi network to fetch data from server using Wi-Fi Module (ESP83266) and display on screen, controlled by Arduino Mega 2560 microcontroller.

In this project we have worked on the idea that we can attach RFID tags on each product in a supermarket or a grocery store and with the help of tag's unique ID (UID) we can retrieve specific product's information like weight and selling price, etc from a locally hosted server database.

Getting the Total sum amount of products added in the cart, Removing a product from the cart, and Generating bill, etc. are some added features in the product/device to make the overall product market-ready, dynamic and user responsive.

Introduction

2. Introduction

IoT- based Smart Shopping Cart is a device having potential to revolutionize the world of the shopping industry with its unique feature and, idea to provide hassle-free, time-saving and fast shopping experience.

On and off, we often see long and possibly never-ending queues on the billing counter, which ruins the shopping desires of customers, making the sellers face huge loss in sales potential. Huge amount of rush plus the cashier preparing the bill with the barcode scanner is very time consuming and results in a heavy crowd as well. **Research organized on the US population shows, they spend approx 113 hours per person per year in the queue, and companies lose approx 50% of customers because of long queues on the billing counter every year.** (Source: Qminder.com).

Our smart IoT enabled shopping cart is well developed to handle these issues and provide an excellent solution to customers as well as shopkeepers.
It majorly works on Radio frequency signals to track the purchasing products using RFID System (RC- 522) and IEEE 802.11 based Wi-Fi network to fetch data from server using Wi-Fi Module (ESP83266) and display on screen, controlled by Arduino Mega 2560 microcontroller.

In our project we have worked on the idea that we can attach RFID tags on each product in a supermarket or a grocery store. The respective tag's ID should be stored on the database server with basic information about the product, for e.g.; Product Name, its weight, its MRP, selling price, offer on the particular product, if any, etc.

Every time a user will scan the product through RFID reader, the respective tag Id will be sent to the server and product details will be fetched back from the matching data from the server. Then the fetched data can be displayed on the display screen followed by displaying total product added in the cart and total amount, etc.

The power of Arduino Mega-2560 microcontroller opened up a door for us to include additive features in the product for example; operating logical operations **(Getting Total sum amount of products added in the cart, Removing a product from the cart, Generating bill, etc.)** along with control over other major components like LEDs and buzzers to make the overall product dynamic and user responsive.

We confidently believe that this product/device has potential to change the world of the shopping industry with its unique feature and idea to provide hassle-free, time-saving and fast shopping experience.

Project Requirements

Sub-Topics:

3. Project Requirements

3.1 Components Used:

- ➢ Arduino Mega 2560
- ➢ RFID Module & Cards (RC-522)
- ➢ Wi-Fi-Module (ESP8266)
- ➢ 16x2 LCD Display
- ➢ Potentiometer (10k ohm)
- ➢ LEDs and Resistors
- ➢ Push buttons and Buzzer
- ➢ PCB and Breadboard
- ➢ Jumper Wires and Connectors

3.2 Languages/Technologies Used:

i). <u>Hardware programming:</u>

- ➢ Arduino Programming (C/C++)
- ➢ AT Commands (ESP8266)

ii). <u>Backend Development:</u>

- ➢ SQL and phpMyAdmin (RDMS)
- ➢ PHP

iii). <u>Frontend development:</u>

- ➢ HTML/CSS
- ➢ JavaScript

Images, and Schematic diagrams

<u>Sub-Topics:</u>

4. Images and Schematic diagrams

4.1 Prototype: Smart Shopping Cart

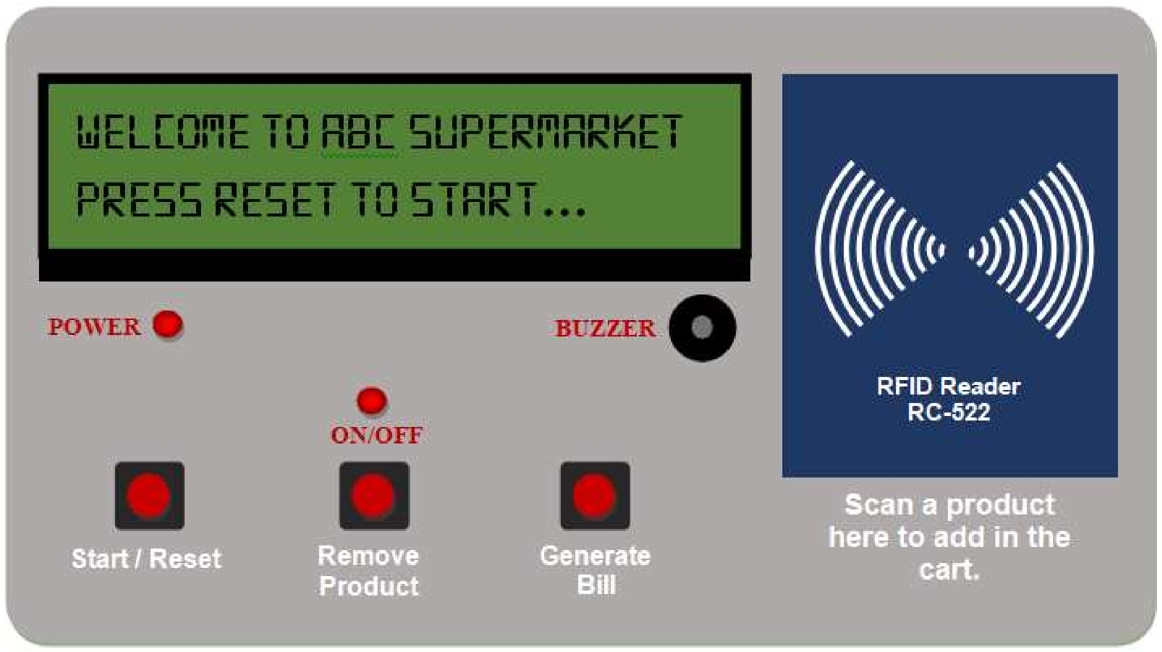


Figure: Smart Shopping Cart (Prototype)
Design by @Mitanshu

Fig. 4.1
Smart Shopping Cart
(Prototype)

4.2 Rough Schematic diagram of Smart Shopping Cart

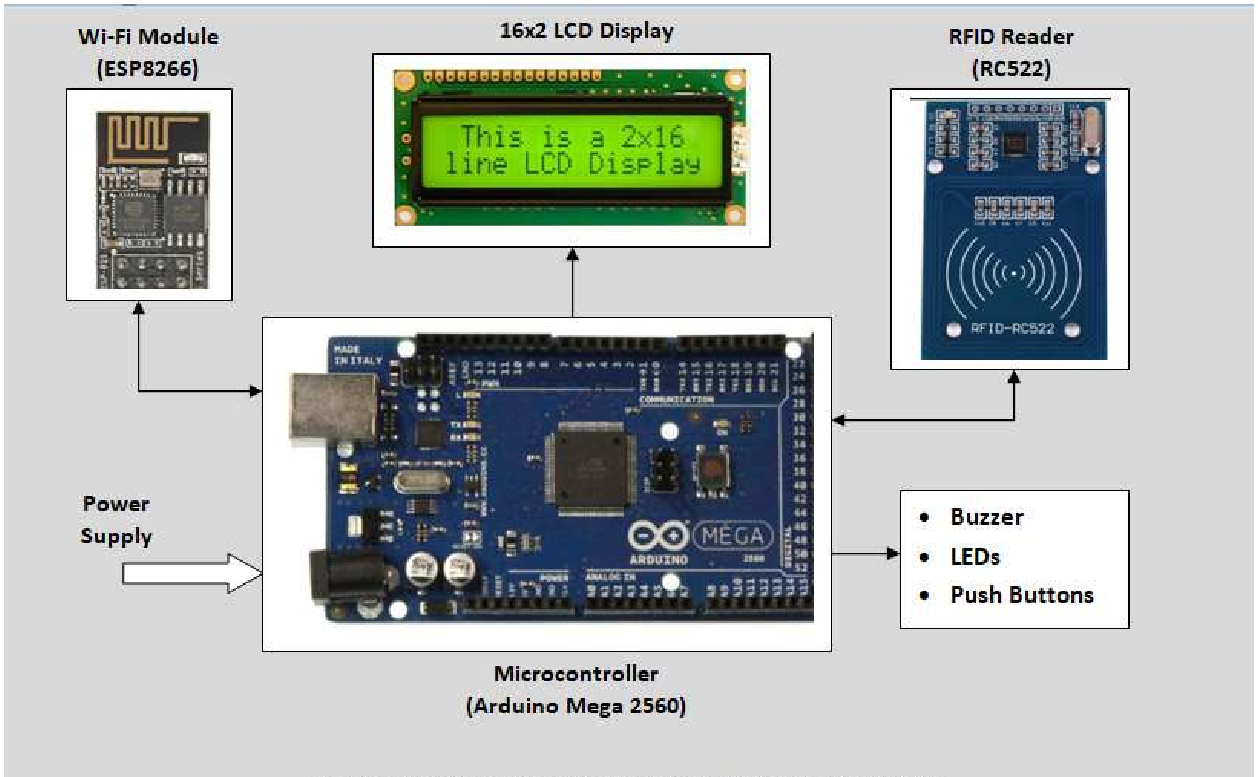


Figure: *Rough Schematic diagram of Smart Shopping Cart*

Fig. 4.2
Rough Schematic Diagram
Of Smart Shopping Cart

4.3 Schematic Diagram of Smart Shopping Cart

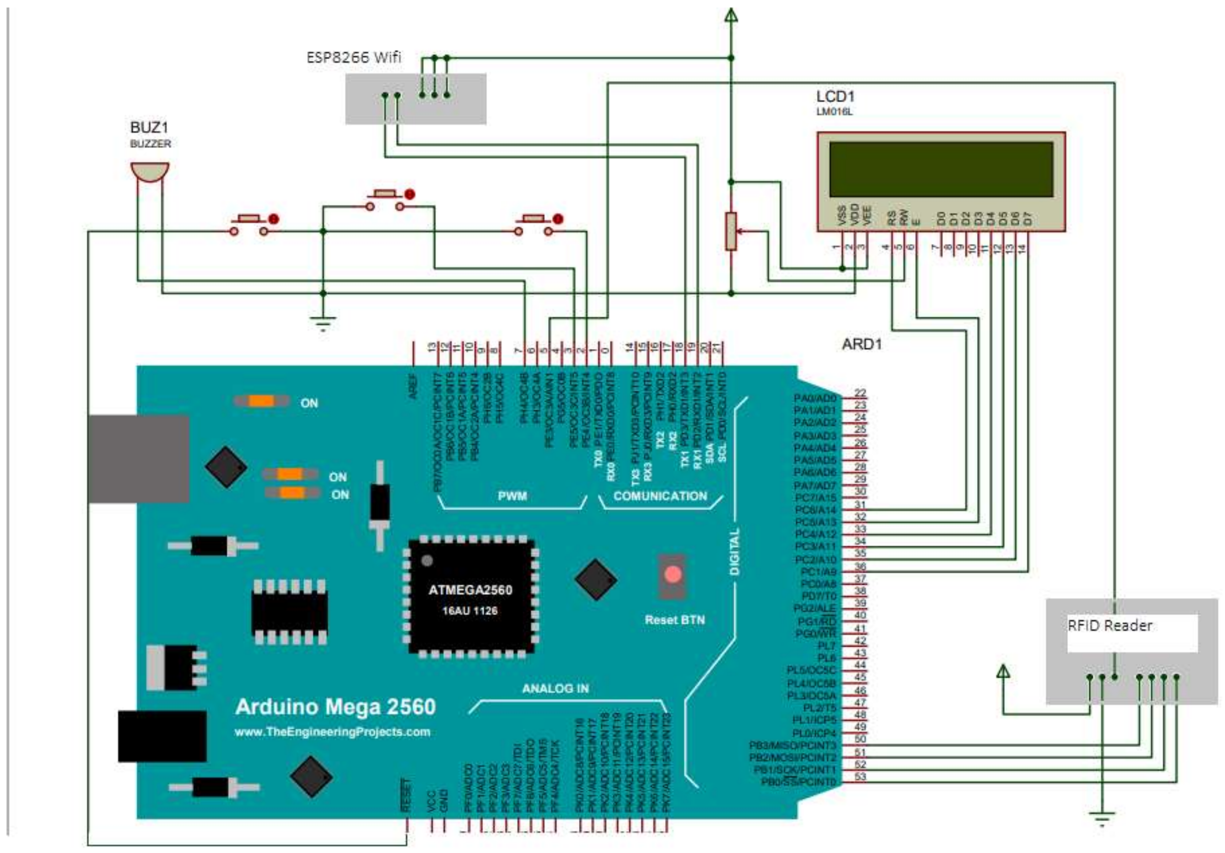


Figure: Schematic Diagram of Smart Shopping

Fig. 4.3
Schematic Diagram of
Smart Shopping Cart.
(Designed on Proteus Design Suite)

Working Principle

Sub-Topics:

5. Working Principle

5.1 How the device works?
(Its functions on surface level)

This IoT enabled Smart shopping cart consists of a Wi-Fi Module, RFID reader, and LCD display which is controlled by Arduino. So, whenever the customer places any product in the cart, it is detected by the RFID module and is displayed on the LCD along with the price of that product (*The product details, like name and Price gets fetched from the database server with the help of Wi-Fi module, more in Section 5.3*). As the customer goes on adding products, all products are detected by the module and therefore the price adds up accordingly. In case, if a customer changes their mind and doesn't want any product added in the cart they can remove it by pressing **Remove button** and the price added will be deducted automatically. At the end, the customer can press the button **Generate Bill (section 5.5)** which when pressed adds the entire product along with their price and gives the total amount to be paid with **unique *Order Id*** (Section 5.4), generated from the database server. At the exit, for payment and verification, customer should share the *unique Order Id* with the person at billing counter, who afterward can generate a soft copy of bill from **Bill generation portal** (section 5.5). Hence this technique is an appropriate method to be used in places like supermarkets and grocery stores. This will help in reducing manpower and helps in making a better shopping experience for customers.

5.2 Role of different components and their function:

5.2.1 RFID Reader and Tags

Radio-frequency identification (RFID) uses electromagnetic fields to automatically identify and track tags attached to the objects. An RFID tag consists of a tiny radio transponder; a radio receiver and transmitter. Since RFID tags can be attached to cash, clothing or possessions we've used RFID technology in our Major project to detect/scan products.

RFID module uses input of 3.3 volts and a frequency of 13.5 MHz. When an RFID tag is shown near the reader, electromagnetic induction takes place between the coils and powers the chip inside the tag. This chip then sends data electromagnetically to the reader. The reader receives this electromagnetically transferred data and outputs it serially. Every RFID reader comes with Serial output pins. We can collect the read data through these serial pins using Arduino or any other microcontroller.

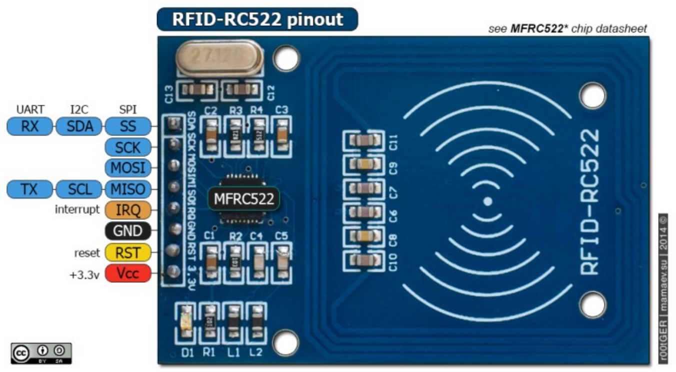


Figure 5.2.1:
RFID Reader
Pin description

In our project we have worked on the idea that we can attach RFID tags to the products in a supermarket or a grocery store. These tags are then scanned by an RFID module which is powered and controlled by an Arduino Mega.

When triggered by an electromagnetic interrogation pulse from a nearby RFID reader device which is connected to the Arduino, tag transmits digital data, usually an identifying inventory number back to the reader. This number is used to inventory goods/products. Using an Arduino Mega we display the shopping essentials like weight of the product, its MRP, its discounted price on an LCD.

5.2.2 Wi-Fi Module (ESP8266)

ESP8266 is an UART Wi-Fi transparent transmission module with ultralow power consumption, specially designed for the needs of a new connected world. It offers a complete and self-contained Wi-Fi networking solution, allowing it to either host the application or to offload all Wi-Fi networking functions from another application processor.

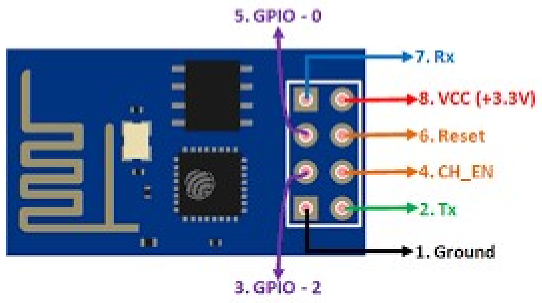


Figure 5.2.2:
ESP8266
Pin description

In our project, ESP8266 will serve as a Station Mode (STP Mode), meaning it will search for the hosted network nearby and connect with the Wi-Fi network. The purpose of the ESp8266 in our project is to provide wireless connectivity to the database server and Arduino Board. (More in section 5.3)

Note: ESP8266 works in AP mode (Access Point) also where it creates its own hotspot and other devices connect with it.

5.2.3 Arduino Mega 2560 Microcontroller

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It is based on the microcontroller Atmega328p of the Atmel Series. Basically it has its own CPU along with a control unit and ALU arithmetic and logic unit (for various mathematical operations) and memory. It has its own software (Arduino IDE) where you can work which is based on C/C++ languages. It has various digital and analog ports where you can play around with a lot of sensors.

We've used Arduino Mega in our project to control and power the other components and sensors. The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.

(Please Turn Over)

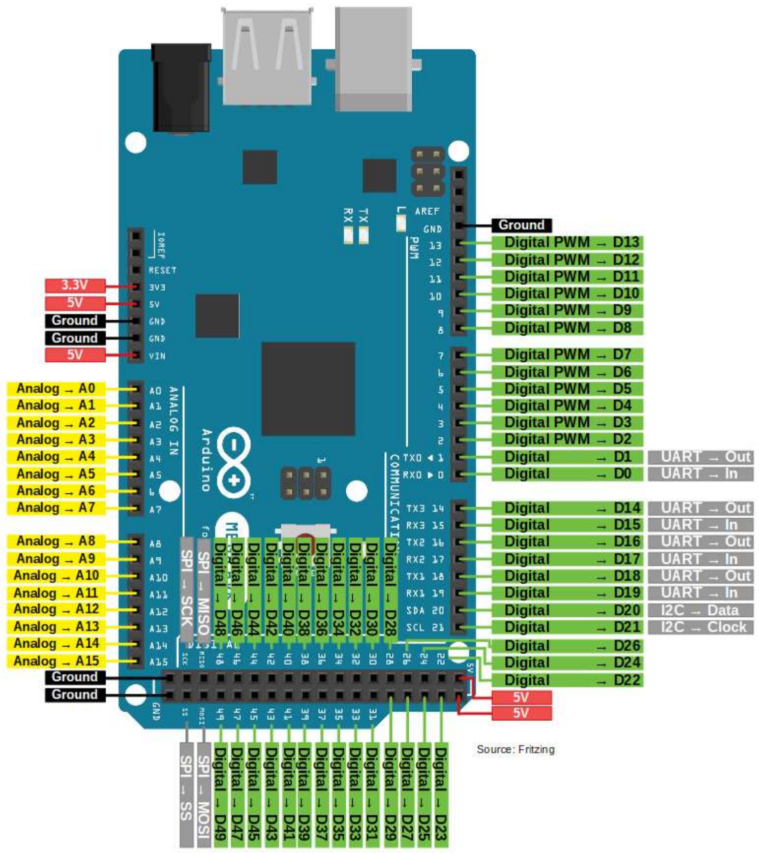


Figure 5.2.3:
Arduino Mega 2560
Pin description

5.2.4 16x2 LCD Display

LCD displays utilize two sheets of polarizing material with a liquid crystal solution between them. An electric current passed through the liquid causes the crystals to align so that light cannot pass through them.

LCD PINS:

- **GND** or **V_{SS}:** Ground or 0V
- **V_{CC}** or **V_{DD}:** Supply Voltage 5V

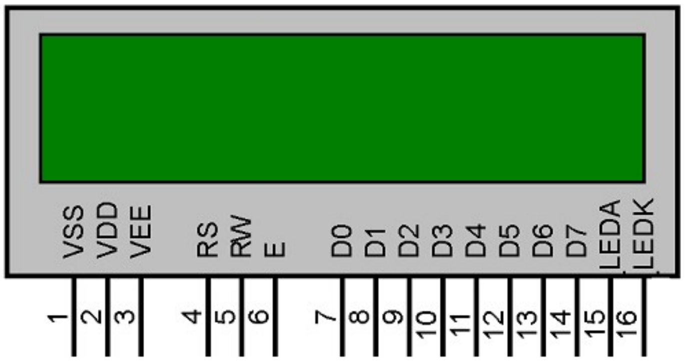


Figure 5.2.2:
LCD Display 16x2
Pin description

- **V_{EE}:** Contrast adjustment through a variable resistor
- **RS:** Register Select.
- **R/W:** Read/Write.
    - When ***RW=1***, data is read from LCD, and
    - When ***RW=0***, writes the data to LCD.
- **EN:** Enable. Sends data to data pins when a high to low pulse is given.
- **Eight Data pins (DB0 to DB7):** These 8-Data pins carry 8-bit data or command from an external unit such as microcontroller.
- **Led+ve:** Back light of the LCD which should be connected to V_{cc} or 5V.
- **Led-ve:** Back light of the LCD which should be connected to Ground or 0V.

5.3 How the device communicates with the host server?

The component majorly responsible for communication with the host server is ESP8266 Wi-Fi Module (Section 5.2.2) which after powered on starts looking for Hotspot Wi-Fi network which must be hosted with given credentials. Only after successful connection, data can be received or transmitted over the hosted Wi-Fi network.

The Hotspot Wi-Fi network should have the same name and password as mentioned in the Arduino code. For e.g.;

```
String wifiName = "ABC_SuperMarket";
String password = "123@abc";
```

Please Refer Arduino Code (Section 9.1)

The ESP modules come with some pre-build command lines (called AT Commands) to send and receive commands and operations. For e.g;

```
AT              // Command to Test whether the ESP module
                   is working properly or not.
```

```
OK              // After sending AT command, we get back a
                   message OK from the server, denoting the
                   module is working properly.
```

Similarly, there’s a full list of commands mentioned in the datasheet of ESP8266 to operate different functions.

To connect with hosted Wi-Fi network, we pass the following command;

```
AT+CWJAP= ”wifiName”,”password”
```

Similarly, to connect with the http server (In our case; Local Server), the following command line must be passed;

```
AT+CIPSTART=0,”TCP”,”serverHTTP:port”
```

Each AT command responds back with a certain message to depict Success and Failure.

Now after successful connection with hosted network and http or local server comes the major task of Wi-Fi Module; *To Send and Receive data.*

For this purpose there’s also a special command available in ESP instruction set, as shown below:

```
AT+CIPSEND=0,”httplink.length”
```

... Followed by HTTP link of character length *httplink.length*.

After sending the above command successfully, comes the role of backend side programs, JavaScript and PHP, which must be available in the server database files.

ESP8266 Wi-Fi module requests the host server using HTTP GET or POST request by sending HTTP link using above command. Below example shows the Arduino code to send HTTP GET request to the host server.

(Please Turn Over)

```
String serverHTTP = "GET /example.com/php?input=456";
sendCommand("AT+CIPSEND=0," + String(serverHTTP.length() + 2),
5, ">");
Serial1.println(serverHTTP);
```

Please Refer Arduino Code (Section 9.1)

This can be understood as; we open a certain site on the browser (suppose Google.com) and then the server responds back to our request and shows a result in the webpage.

Similarly, When AT+CIPSEND sends a http link with GET request to the server, the server processes the respective PHP script file on the host server and accordingly action is taken depending on commands/actions mentioned in the PHP file.

The below code snippet from Arduino code summarizes the whole process of sending HTTP GET request to the server and storing the server output at appropriate location.

(Please Turn Over)

```
String wifiName = "ABC_SuperMarket";
String password = "123@abc";
String serverHost = "192.168.43.8";      //localhost IP
String port = "80";

//sendCommand("AT Command", "WaitCount", "ServerResponse")
sendCommand("AT", 5, "OK");  //testing server
sendCommand("AT+CWMODE=1", 5, "OK");  //setting network mode
sendCommand("AT+CWJAP=\"" + wifi + "\",\"" + pass + "\"", 10, "OK");
                                                //connecting to host network
String serverHTTP = "GET /example.com/php?input=456";
sendCommand("AT+CIPMUX=1", 5, "OK");
sendCommand("AT+CIPSTART=0,\"TCP\",\"" + serverHost + "\"," + port,
10, "OK");
sendCommand("AT+CIPSEND=0," + String(serverHTTP.length() + 2),
5, ">");
Serial1.println(serverHTTP);
readServer();  //reading the server response i.e. receiving data from the
server.
```
Please Refer Arduino Code (Section 9.1)

5.4 Getting Started.

(***Creation of Order-ID*, *Addition*** and ***Removal of Product*** from Host server.)

To get started, it's pretty interesting that in our program we are retrieving as well as storing new data in the server database. Every time user starts the device i.e. presses Start/Reset button, a new Order-ID gets generated in the server database and also the same gets fetched to the Arduino memory. All the purchases that happen in that session get linked with that same Order-ID, and all the user actions i.e. adding/removing products from the cart, keeps getting updated in a new database table (shopping Bill) linked with that same order-ID in parallel.
Here is a code snippet that explains it more clearly;

```
String orderId = sendToServer("GET /Test_site/generateOrderID.php?");
```

Please Refer Arduino code (section 9.1) for **sendToServer()** function.

The function **sendToServer()** returns a unique **order Id** and that gets stored in a variable *order-Id.*

You may be wondering what happens on the backend side after the *GET* request is passed by the Arduino code. The PHP file that processes the task of Order-Id generation is named as ***generateOrderID.php*** which commands the server database to create a random Order-Id and store it in the database table.
After that, whenever a user scans a product to add in or remove from the cart, Arduino sends the GET request to the server with the same order-ID which processes the respective PHP file and takes action.

When a user adds a product in the cart, the below command gets passed to the host server from Arduino (Wi-Fi module) with respective order-Id and product-Id as shown below, which adds the product in respective database table linked with the given order-Id.

```
String productData = sendToServer("GET/
Test_site/addProduct.php?orderId="+ orderId + "&productId=" +
tagID);
```

Please Refer Arduino code (section 9.1) for **sendToServer()** function.

Similarly, when a user removes data, the below code snippet gets passed to the host server with the same Order-Id.

```
String deleteResult = sendToServer("GET
/Test_site/removeProduct.php?orderId="+ orderId + "&productId=" +
tagID);
```

Please Refer Arduino code (section 9.1) for **SendToServer()** function.

The respective PHP script files then do their jobs to add or remove data from the database (which majorly consists SQL Syntaxes used to handle data in the database) and respond back to the Arduino with a certain message.

Note: *You can see the PHP script files in the Code Snippet Section (9.2).*

5.5 Bill Generation Functionality

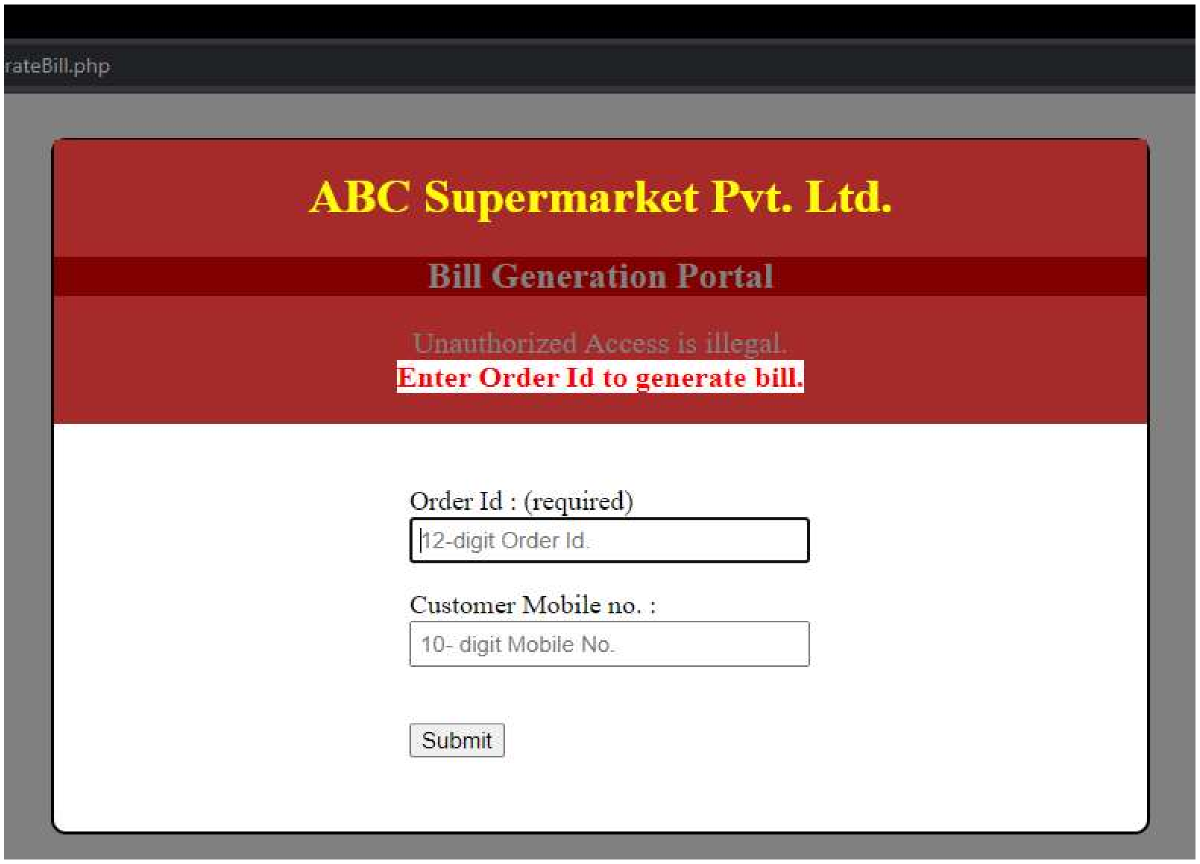
How Bill generation works in Smart Shopping Device?

Every shopping ends with Bill generation and payments. To make the product market ready we have implemented a button which will generate Bill in our device which restricts the system/user to add or remove products any further.

When the user presses a ***Generate Bill button***, the Order Id generated in the previous section gets displayed on the display screen with Total amount (Example fig. in User manual Section) and with that Order Id, an administration at billing counter generates bill and accepts payment.

The role of Arduino and Hardware components are done after getting Order-Id displayed on the display screen. After that, a web page helps to retrieve or generate a soft copy of Bill as shown below (Fig.5.5a). It should be understood that the whole process takes place with the help of PHP script and user interface is designed with the help of HTML, CSS and JavaScript.

(Please Turn Over)



ABC Supermarket Pvt. Ltd.

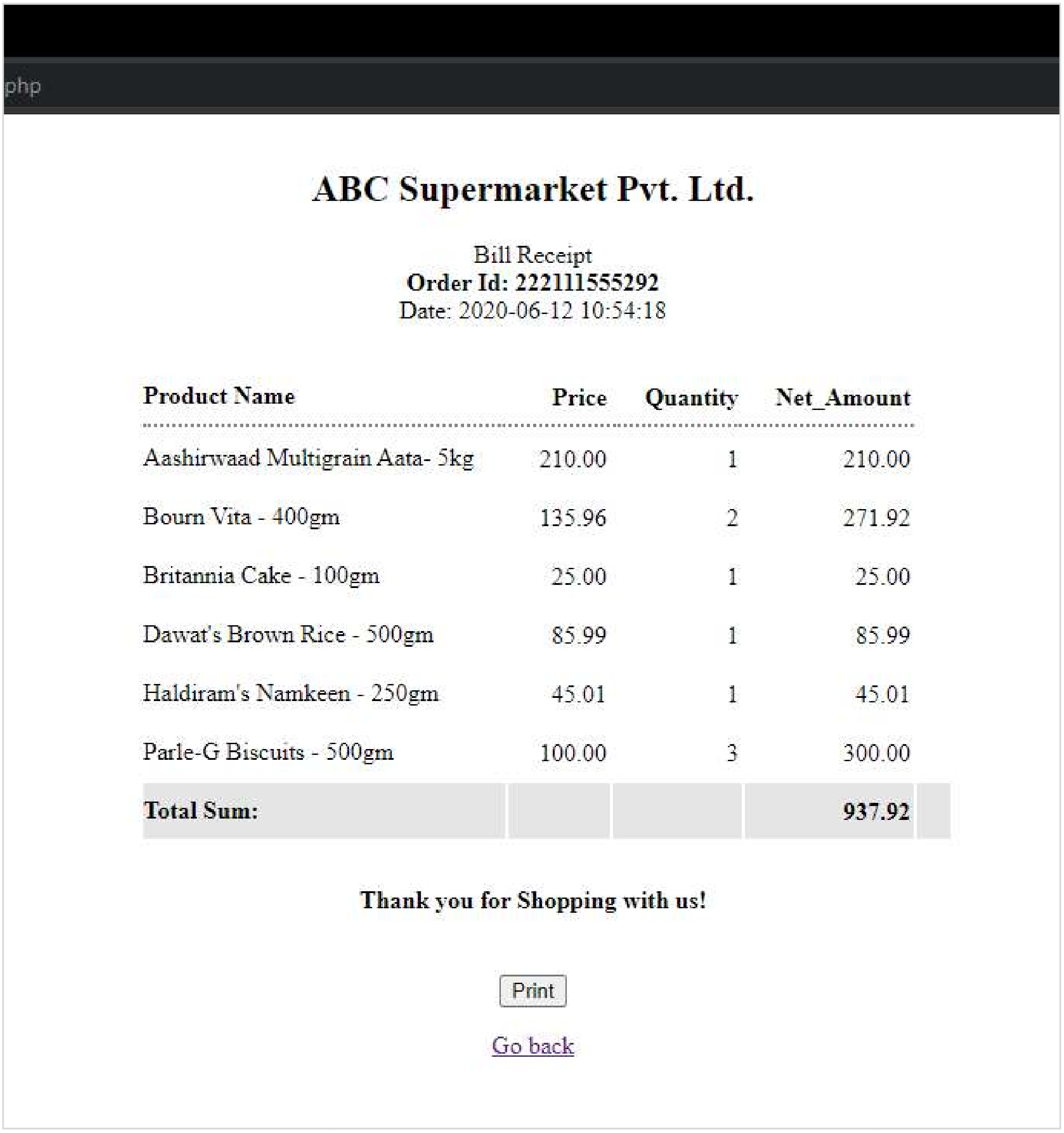
Bill Generation Portal

Unauthorized Access is illegal.
Enter Order Id to generate bill.

Order Id : (required)
12-digit Order Id.

Customer Mobile no. :
10- digit Mobile No.

Fig. 5.5a (I) Bill Generation Portal


php

ABC Supermarket Pvt. Ltd.

Bill Receipt
Order Id: 222111555292
Date: 2020-06-12 10:54:18

Product Name	Price	Quantity	Net_Amount
Aashirwaad Multigrain Aata- 5kg	210.00	1	210.00
Bourn Vita - 400gm	135.96	2	271.92
Britannia Cake - 100gm	25.00	1	25.00
Dawat's Brown Rice - 500gm	85.99	1	85.99
Haldiram's Namkeen - 250gm	45.01	1	45.01
Parle-G Biscuits - 500gm	100.00	3	300.00
Total Sum:			**937.92**

Thank you for Shopping with us!

Fig. 5.5a (II)
Sample Bill; Generated after submitting a sample Order Id on Bill generation Portal.

Further Space for Improvement

6. Further Space for Improvement

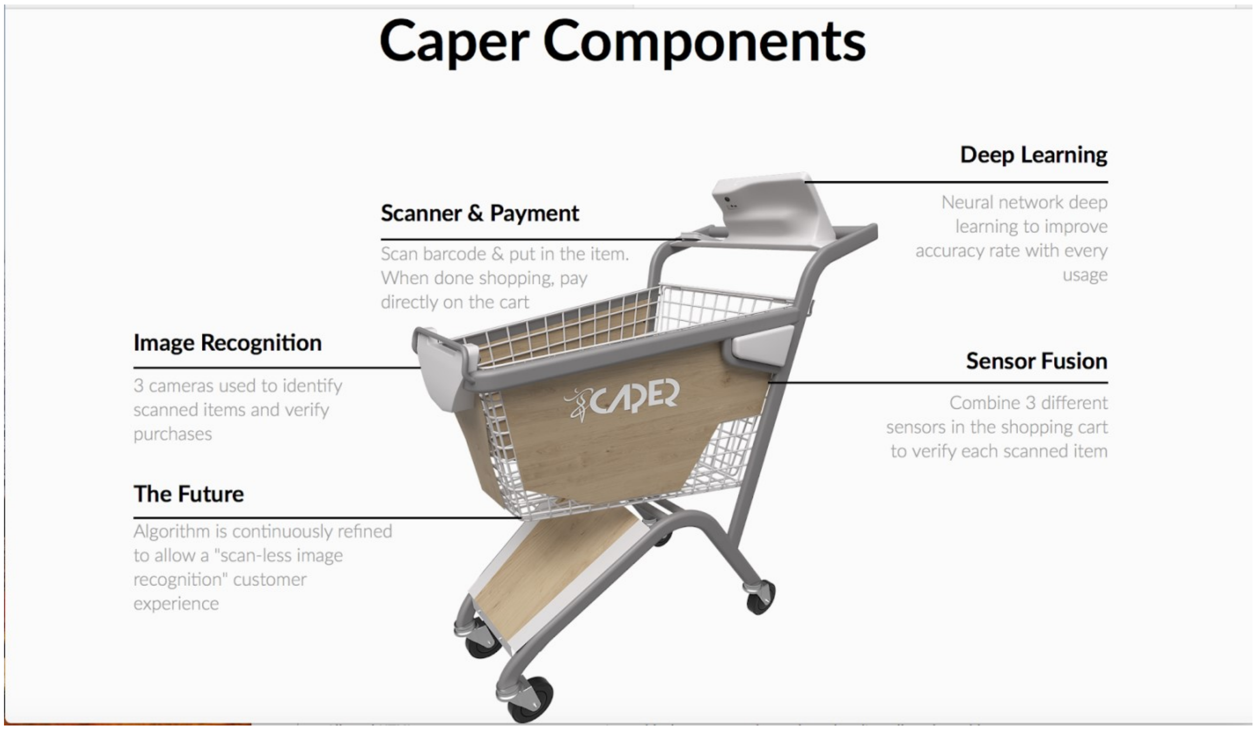
The world is changing so fast that everyday something new emerges in the field of Science and Technology. With increasing demand of Automation and lightning-fast processing, everything needs to keep updating with new Technologies.

Our Prototype device, '*Smart Shopping Cart*' should also need to be regularly updated to maintain pace with the fast emerging digital world.

Experts say the coming decades are full of Automation, Artificial Intelligence and Machine Learning. Keeping that in view, many developed countries have already started testing 'Smart Shopping Cart' to make the shopping experience more streamlined. The AI and Machine Learning algorithms are making the device even smarter, by giving product insights, shopping analysis, predictions and personalized shopping suggestions. We can acquire Machine Learning models and algorithms into our project to make it more smart and efficient.

Below picture shows an example of a Smart shopping cart, developed by ***CAPER Inc.,*** an US-based private software company, which includes Deep Learning, AI, and Image Recognition to their system, provides an ultimate smart shopping solution.

(Please Turn Over)



Caper Components

Deep Learning
Neural network deep learning to improve accuracy rate with every usage

Scanner & Payment
Scan barcode & put in the item. When done shopping, pay directly on the cart

Image Recognition
3 cameras used to identify scanned items and verify purchases

Sensor Fusion
Combine 3 different sensors in the shopping cart to verify each scanned item

The Future
Algorithm is continuously refined to allow a "scan-less image recognition" customer experience

Figure 6.1:
AI Enabled Smart Cart
Source: https://caper.ai/

Conclusion

7. Conclusion

Smart shopping cart is indeed a great technological advancement in the field of shopping and market industry. With increasing digitization, especially in our country, Digital India initiatives are playing a major role in making India advance with each passing moment in the field of Science and Technology.

Our project 'Smart shopping cart' with Internet of Things and smart sensors surely passes all the required parameters to contribute to emerging Digital India.

In this project report, we saw in-depth working and processes of Smart shopping cart; how it works, how data fetches from the server database, how RFID helps to detect a product and respond accordingly, how Arduino and ESP8266 together create a Internet enabled ecosystem, i.e. IoT, and a lot more.

We expect that the report on 'IoT-based Smart shopping cart' helps a reader to familiarize themselves with the smart, hassle-free and time-saving Smart shopping cart, and help them to apply the information to the more practical approaches and in the improvement of the Tech and shopping industry.

Here is a picture of our project under Prototype testing, in working condition. The other picture shows the result output on serial monitor.

(Please Turn Over)

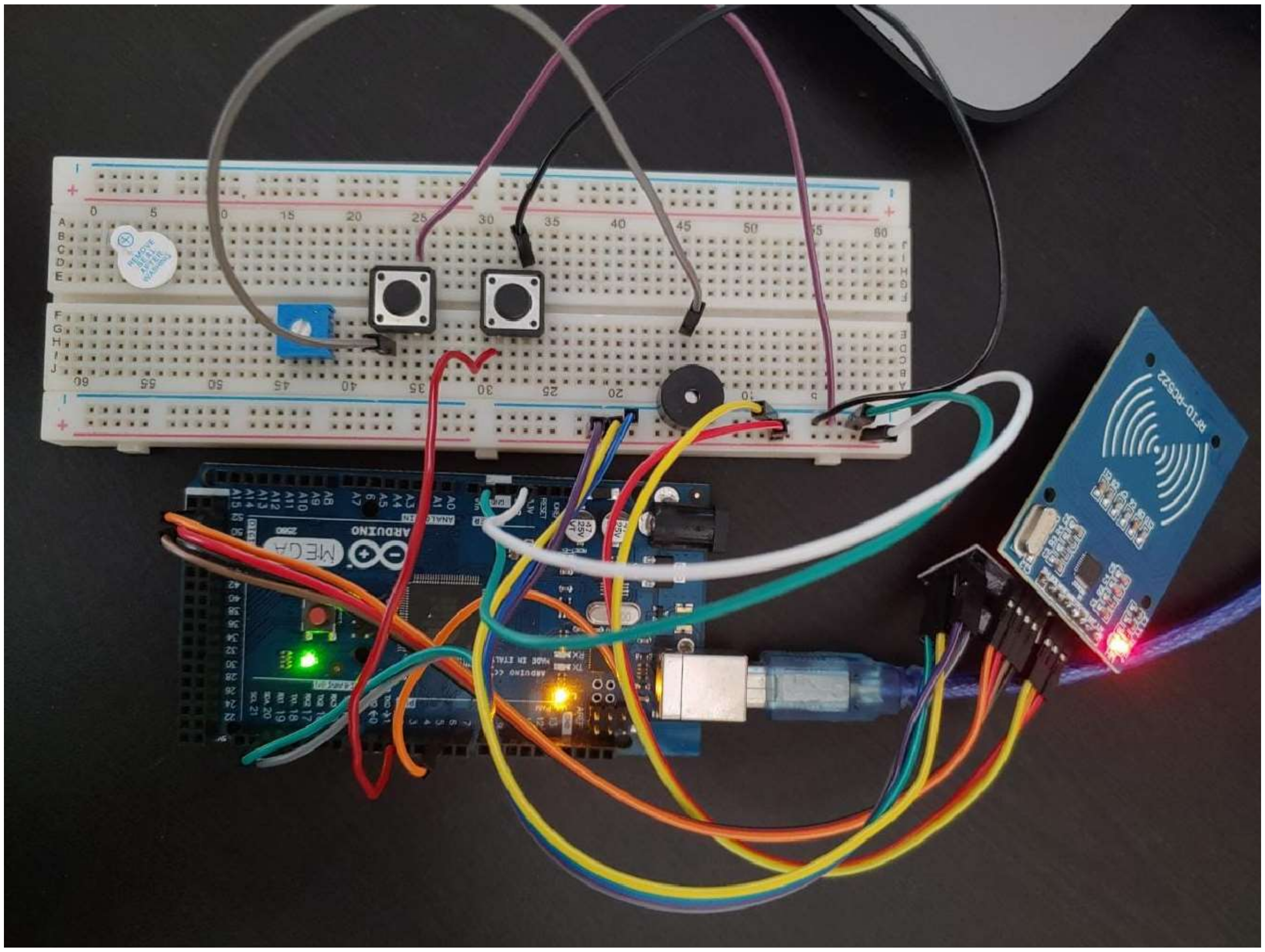
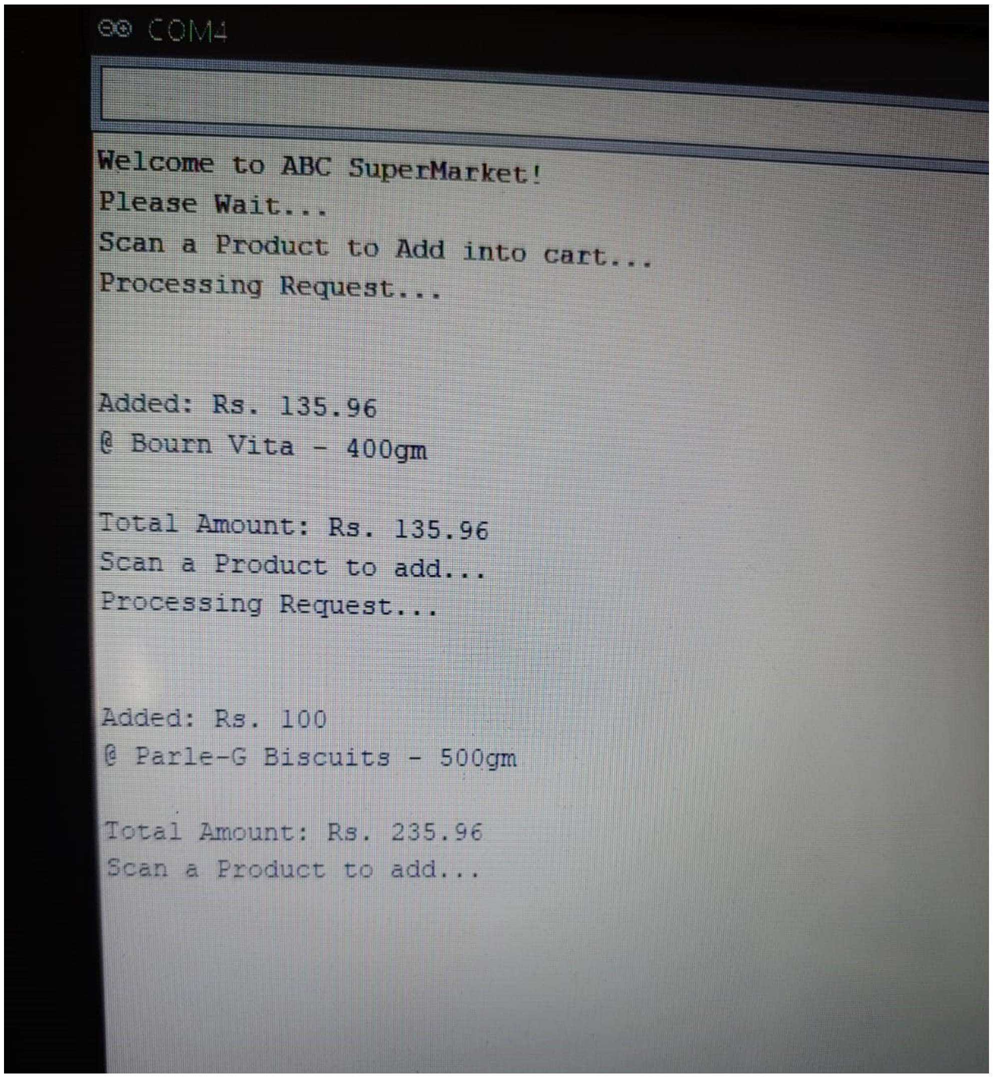


Figure 7.1
Working Project
(Prototype testing)



```
Welcome to ABC SuperMarket!
Please Wait...
Scan a Product to Add into cart...
Processing Request...


Added: Rs. 135.96
@ Bourn Vita - 400gm

Total Amount: Rs. 135.96
Scan a Product to add...
Processing Request...


Added: Rs. 100
@ Parle-G Biscuits - 500gm

Total Amount: Rs. 235.96
Scan a Product to add...
```

Figure 7.2
Result output on Serial Monitor.
(Prototype testing)

References

8. References

https://www.arduino.cc/en/Tutorial/HomePage

https://dev.mysql.com/doc/refman/8.0/en/sql-syntax.html

https://dev.mysql.com/doc/refman/8.0/en/

https://medium.com/@cgrant/using-the-esp8266-wifi-module-with-arduino-uno-publishing-to-thingspeak-99fc77122e82

https://www.espressif.com/sites/default/files/documentation/4b-esp8266_at_command_examples_en.pdf

https://nevonprojects.com/smart-shopping-trolley-with-automated-billing-using-arduino/

https://innovate.mygov.in/innovation/smart-shopping-cart-2/#:~:text=The%20Automated%20Shopping%20Cart%2C%20%E2%80%9CSmart,experience%20of%20every%20buyer.

https://d0wnl0ads.files.wordpress.com/2011/08/sql-the-complete-reference-third-edition-sep-2009.pdf

https://www.php.net/docs.php

https://www.udemy.com/course/the-complete-web-developer-course-2/

https://www.qminder.com/cost-customer-service-wait-times/

https://www.caper.ai

https://www.stackoverflow.com/

https://www.google.com/

Coding

Sub-Topics:

9. Coding

9.1 Arduino/Hardware Program.

Arduino Code: Should be uploaded to Arduino Mega Board.

```
#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 53
#define RST_PIN 5
MFRC522 mfrc522(SS_PIN, RST_PIN);   // Create MFRC522 instance.
const int buzzer = 7; // buzzer to arduino pin 9

int removeProductButton = 2;    //remove product
int generateBillButton = 3;   //generate bill
boolean removeItem = false;


String wifiName = "V40 ThinQ_3650";
String password = "mkmk78789";

String serverHost = "192.168.43.8";
String port = "80";

float totalPrice = 0.0;
String orderId = "";
String tagID = "";

void setup() {
  Serial.begin(9600);   // Initiate a serial communication
  SPI.begin();      // Initiate  SPI bus
  mfrc522.PCD_Init();   // Initiate MFRC522
  pinMode(generateBillButton, INPUT_PULLUP);
  pinMode(removeProductButton, INPUT_PULLUP);
  pinMode(buzzer,OUTPUT); // set buzzer pin 7 as an output

  Serial1.begin(115200);  //esp8266 serial begin
  sendCommand("AT", 5, "OK");
  sendCommand("AT+CWMODE=1", 5, "OK");
  sendCommand("AT+CWJAP=\"" + wifiName + "\",\"" + password +
"\"", 10, "OK");

  Serial.println("Welcome to ABC SuperMarket!");
  Serial.println("Please Wait...");
  //Generating Unique Order Id.
  orderId = sendToServer("GET /Test_site/generateOrderID.php?");
  int backTickIndex = orderId.indexOf('`');
  orderId = orderId.substring((backTickIndex + 1),(backTickIndex + 13));
  //Serial.println("Order ID: " + orderId);

  Serial.println("Scan a Product to Add into cart...");
  //LCD Welcome Message.
  //LCD Display("Scan a product")***********
}

void loop(){
  int removeButtonVal = digitalRead(removeProductButton);
  int generateBillVal = digitalRead(generateBillButton);
  if(removeButtonVal == LOW)
  removeProduct();

  if(generateBillVal == LOW)
  generateBill();

  if (!mfrc522.PICC_IsNewCardPresent()) {
```

```
 return;
  }
  // Select one of the cards
  if (!mfrc522.PICC_ReadCardSerial()) {
    return;
  }

  Serial.println("Processing Request...");
  tone(buzzer, 5000, 100);

  // Serial.print("UID tag: ");
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    tagID += String(mfrc522.uid.uidByte[i], HEX);
  }
  tagID.toUpperCase();
  // Serial.print(tagID);
  Serial.println();

  //String productdata = wifiCommand("GET
/Test_site/RFID.php?tagID=",tagID);
  String productdata= sendToServer("GET
/Test_site/addProduct.php?orderId=" + orderId + "&productId=" +
tagID);


trimOutput(productdata);
}

String sendToServer(String url) {
  //place LCD Print("Processing...") if delay in final output.

  String serverHTTP = url;
```

```
sendCommand("AT+CIPMUX=1", 5, "OK");
  sendCommand("AT+CIPSTART=0,\"TCP\",\"" + serverHost + "\"," +
port, 10, "OK");
  sendCommand("AT+CIPSEND=0," + String(serverHTTP.length() + 2),
5, ">");
  Serial1.println(serverHTTP);
  String serverOutput = readServer();
  return serverOutput;
}

String readServer() {
  String ch = "";
  while (Serial1.available()) {
    ch = Serial1.readStringUntil('\0');
    //Serial.print(ch);
    //delay(10);
  }
  return ch;
}

void trimOutput(String ch){
  int backTick1 = ch.indexOf('`');
  int backTick2 = ch.indexOf('`', backTick1 + 1);
  int backTick3 = ch.indexOf('`', backTick2 + 1);



  String price = ch.substring(backTick1 + 1, backTick2);
  String product = ch.substring(backTick2 + 1, backTick3);

  Serial.println();
  if(removeItem == false){
  Serial.println("Added: Rs. " + price + "\n@ " + product);
  //LCD Print ; This is the final output to the user.
  //Added: Rs. [ProductPrice]
  //@ [ProductName]
  }
  else if(removeItem == true){
    if(price.equals("0"))
    Serial.println("Product Not Found");
    else
    Serial.println("Removed: Rs. " + price + "\n@ " + product);
  //LCD Print ; This is the final output to the user.
  //Removed: Rs. [ProductPrice]
  //@ [ProductName]
  }
  //Success Buzzer place here.
  tone(buzzer, 8000, 50);
  delay(10);
  tone(buzzer, 5000, 100);
  totalSum(price);
  tagID="";
}

void totalSum(String price){
   if(removeItem == false){
  totalPrice += price.toFloat();
   }else if(removeItem == true){
      totalPrice -= price.toFloat();
      Serial.println("Remove Mode OFF...");



 }
  Serial.println("\nTotal Amount: Rs. " + String(totalPrice));
  Serial.println("Scan a Product to add...");
  //Print to LCD as well.
```

```
}
void removeProduct(){
  Serial.println("\nRemove Mode ON...");
  Serial.println("Scan a product to remove...");

  while(!mfrc522.PICC_IsNewCardPresent());
  while(!mfrc522.PICC_ReadCardSerial());

  Serial.println("Processing Request...");
  tone(buzzer, 5000, 100);

  // Serial.print("UID tag: ");
  tagID = "";
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    tagID += String(mfrc522.uid.uidByte[i], HEX);
  }
  tagID.toUpperCase();
  //mk Serial.print(tagID);
  Serial.println();
  String deleteResult = sendToServer("GET
/Test_site/removeProduct.php?orderId=" + orderId + "&productId=" +
tagID);
  removeItem = true;
  //Serial.println(deleteResult);
  trimOutput(deleteResult);
  removeItem = false;
}

void generateBill(){
  Serial.println("Generating Bill...");
  Serial.println("Order Id:" + orderId + "\n@ Total amount: Rs." +
totalPrice);
  tone(buzzer, 10000, 1000);
```

```
 Serial.println("Thank you for Shopping.");
  while(1);
}

void sendCommand(String command, int waitCount, char readReply[]) {
  boolean flag = true;
  Serial1.println(command);
  int count = 0;
  while (!Serial1.find(readReply)) {
    count++;
    if (count == waitCount) {
      Serial.println("Fail: " + command);
      flag = false;

      if(command.equals("AT+CWJAP=\"" + wifiName + "\",\"" +
password + "\"")
      || command.equals("AT+CIPSTART=0,\"TCP\",\"" + serverHost +
"\"," + port)){
        Serial.println("Server Down. Please press RESET...");
        while(1){
          tone(buzzer, 1000);
          delay(500);
          noTone(buzzer);
          delay(500);
        }
        //LCD Print here.
      }
      else{


 Serial.println("Error: Please Scan Again.");
        //LCD Print here.
        //Failure Buzzer place here.
    tone(buzzer, 10000, 1000);
      }
      break;
    }
  }
  if(flag){
  //Serial.println("Command Success: " + command);
  }
}
```

9.2 Server -Side Program

9.2.1 PHP Scripts

(i). generateOrderId.php: Generates unique Order Id in server database, create a virtual Bill receipt, and also fetch the Order Id to Arduino memory, when triggered.

```
<?php

$link = mysqli_connect("localhost","root", "", "abc_supermarket");

if (mysqli_connect_error()) {
  die ("There was an error connecting to the database");
}
/*else{
echo "Connection Successful!";
}*/

date_default_timezone_set('Asia/Kolkata');
```

```
$date = date("Y-m-d h:i:s");

$insert = "INSERT INTO `orderdetails` (`orderId`, `userMobile`, `date`)
VALUES (NULL, NULL, '$date')";
mysqli_query($link, $insert);

$query = "SELECT `orderId` FROM `orderdetails` WHERE `date` =
'$date'";

if ($result = mysqli_query($link, $query)) {
  $row = mysqli_fetch_array($result);

  echo "`".$row['orderId']."`";
}
else{
  echo "Error in Server";
}

?>
```

(ii). addProduct.php: Processes Arduino request to add new product in the cart i.e. fetch product detail from the server database and display on LCD screen. Also, add product in the virtually created Bill receipt.

```
<?php

$link = mysqli_connect("localhost","root", "", "abc_supermarket");

if (mysqli_connect_error()) {
  die ("There was an error connecting to the database");
}
/*  else{
echo "Connection Successful!";
}*/
$orderId = $_GET['orderId'];
$productId = $_GET['productId'];

$serverRead = "SELECT * FROM productdata WHERE productId =
'$productId'";

if ($result = mysqli_query($link, $serverRead)) {
  $row = mysqli_fetch_array($result);
  echo "`".$row['productPrice']."`".$row['productName']."`";

  $query = "INSERT INTO `purchaselist`(`order_Id`, `product_Id`)
VALUES ('$orderId', '$productId')";
  mysqli_query($link, $query);
}
else{
  echo "Error in Server";
}

?>
```

(iii). removeProduct.php: Processes Arduino request to remove product from the cart i.e. remove product Virtual Bill receipt and display on LCD screen.

```
<?php

$link = mysqli_connect("localhost","root", "", "abc_supermarket");

if (mysqli_connect_error()) {
  die ("There was an error connecting to the database");
}
/*  else{
echo "Connection Successful!";
}*/
$orderId = $_GET['orderId'];
$productId = $_GET['productId'];

$query="SELECT `order_Id`, `productId`,`productName`,`productPrice`
FROM `purchaselist`,`productdata`
WHERE `order_Id`= $orderId and `product_Id` = `productId` and
`product_Id`='$productId' LIMIT 1";

if ($result = mysqli_query($link, $query)) {
  $row = mysqli_fetch_array($result);
  if($row['productPrice'])
  echo "`".$row['productPrice']."`".$row['productName']."`";
  else {
    echo "`"."0"."`"."Product not found."."`";
  }
  $delete = "DELETE FROM `purchaselist` WHERE `order_Id`=
'$orderId' and `product_Id`= '$productId' LIMIT 1";
  mysqli_query($link, $delete);
}
else{
  echo "Error in Server";
 }

 ?>
```

(iv). generateBill.php: Webpage of Bill generation Portal and gateway to generate soft copy of Bill receipt.

```
<!DOCTYPE html>
<html>
<head>
  <title>Generate Bill - ABC SuperMarket</title>
  <style>
  body{
    background-color: grey;
    padding: 0;
    margin: 0;
  }
  #main{
    background-color: white;
    width: 720px;
    height: 100%;
    margin: 0 auto;
    text-align:center;
    border: 2px solid;
    border-radius: 10px;
    padding-bottom: 50px;
    margin-top: 30px;
  }
  #form-div{
```

```
 width: 35%;
    margin: 0 auto;
    padding-top: 20px;
  }
  #form{
    text-align: left;
  }
  #head{
    background-color: brown;
    position: relative;
    top:-22px;
    margin-bottom: -20px;
  }
  h1{
    padding:20px 0 0 0;
    color:yellow;

  }
  p{
    font-size:20px;
    padding-bottom: 20px;
    color:grey;
  }
  .field{
    padding: 5px;
    width: 250px;
    font-size: 15px;
  }
  label{
    font-size: 18px;
  }
</style>
```

```
<body>
  <div id="main">
    <div id="head">
      <h1>ABC Supermarket Pvt. Ltd.</h1>
      <h2 style="background-color:maroon; color: grey"> Bill Generation
Portal </h2>
      <p> Unauthorized Access is illegal.</br>
        <span style="color:red; background-color:white;"><strong>Enter
Order Id to generate bill.
        </strong></span></p>
      </div>
      <div id="form-div">
        <form id="form" action="customerBill.php" method="post">
          <label>Order Id : (required)</label><br>
          <input class="field" name="orderId" placeholder='12-digit Order
Id.' type='text' pattern ='[0-9]{12}' required><br><br>
          <label>Customer Mobile no. :</label><br>
          <input class="field" name="mobileNo" placeholder='10- digit
Mobile No.' type='text' pattern ='[0-9]{10}'
autocomplete="off"><br><br>

          <input style="font-size: 15px; margin-top: 20px" name="submit"
type='submit' value='Submit'>
        </div>

        <?php
        if(isset($_POST['submit'])){
          // Fetching variables of the form which travels in URL
          $orderId = $_POST['orderId'];
          $phone = $_POST['mobileNo'];
          if($orderId !='')
          {
            //  To redirect form on a particular page
```

```
 header("Location:customerBill.php");
            //echo "<script> window.open('');</script>";
          }
          else{
            echo "error";
          }
        }
        ?>
      </form>
    </div>

  </body>
  </html>
```

(v). customerBill.php: Prints soft copy of Bill receipt in the web browser.

```
<style>

table, th, td{
  margin-
left: auto;
  margin-right: auto;
  padding: 10px 2px 8px 20px;
  text-align: right;
}
th{
  border-bottom: 2px dotted grey;
}
.first_clm{
  padding: 5px 20px 5px 0;
  text-align: left;
}
table tr:last-child{
  font-weight: bold;
  background-color: #e5e5e5;
}
#bill{
  width: 100%;
  margin: 0 auto;
  margin-top: 35px;
  text-align: center;
}

</style>
<?php

$link = mysqli_connect("localhost", "root", "", "abc_supermarket");

if (mysqli_connect_error()) {

  die ("There was an error connecting to the database");
}

$data = $_POST['orderId'];

$query= "select productName,
cast(productPrice as decimal(10,2)) Price,
count(*) qty,
cast((productPrice*count(*)) as decimal (10,2)) Net_Amt
from productdata, purchaselist
```

```
where productId = product_Id and order_Id =$data
group by productName, productPrice
union all
select 'Total Sum:',' ',' ',sum(totalSum)
from (
  select productName, productPrice,
  count(*) quantity,
  cast((productPrice*count(*)) as decimal (10,2)) totalSum
  from productdata, purchaselist
  where productId = product_Id and order_Id =$data
  group by productName, productPrice)
  t";
  if($result = mysqli_query($link, $query)){

    date_default_timezone_set('Asia/Kolkata');
    $date = date("Y-m-d h:i:s");

    echo'<div id="bill"><h2>ABC Supermarket Pvt. Ltd.</h2>';
    echo'<p>Bill Receipt<br>';
    echo '<strong>Order Id: '.$data.'</strong><br>';
    echo 'Date: '.$date.'</p>';

    echo "<table class=\"table_style\"><tr><th class=\"first_clm\"
>Product
Name</th><th>Price</th><th>Quantity</th><th>Net_Amount</th></tr
>";
    while ($row = mysqli_fetch_array($result)) {
      echo '<tr><td
class="first_clm">'.$row["productName"]."</td><td>".$row["Price"]."<
/td><td>".$row["qty"]."</td><td>".$row["Net_Amt"]."</td><td>";
    }

    echo "</table><h4>Thank you for Shopping with us!</h4>";
```

```
 echo '<br><button onclick="window.print()">Print</button>';
  echo '<p><a href="generateBill.php">Go back</a><p></div>';
}
else{
  echo "Order Id not found!...";
}

?>
```

User manual

Sub-Topics:

10. User manual

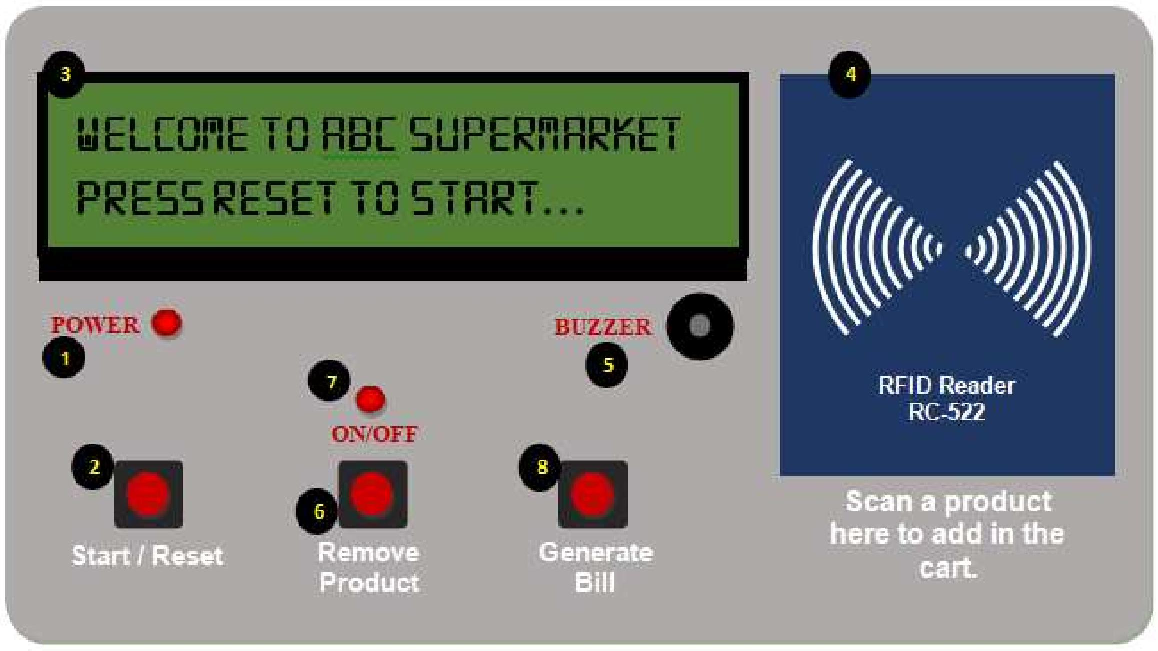


Figure: Smart Shopping Cart (Prototype)
Design by @Mitanshu

10.1 Key Indications:

1. POWER Indicator LED: Main Power Supply.

2. Start/Reset Button: To start or reset the device.

3. LCD Display: Displays Messages from Device.

4. Product Scanner (RFID): Scans the RFID tag attached to the product.

5. Sound (Buzzer): Buzzer Sounds to inform activity and confirmations.

6. Remove Product Button: Button to ON/Activate Removal mode.

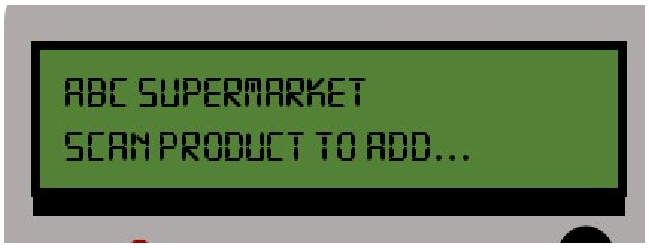
7. Remove Product Indicator (ON/OFF): Indicates the status of Remove button.

8. Generate Bill Button: Button to end shopping and generate Bill.

10.2 Instructions to Use:

Step1. Press Start/ Reset button before starting adding product to cart.

The display screen will show a message, Scan product…



ABC SUPERMARKET
SCAN PRODUCT TO ADD...

Figure 10.2

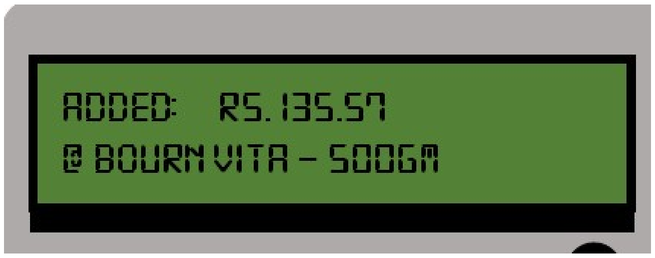
Remember, Pressing Start/Reset button in between your shopping will omit all the previous added products.

Step2. Fill your shopping cart with your favorite items by scanning the product to the scanner.

The first beep sound indicates the product is successfully scanned.

And the second beep sound indicates the product is successfully added.

The display screen will show message as shown below:



ADDED: RS.135.57
@ BOURNVITA - 500GM

Figure 10.3

Step3. In case, you change your mind and wish to remove any product, however we don't suggest that; press the Remove Product button; the LED indictor will turn ON, and the display screen will show Remove mode on, as shown below:

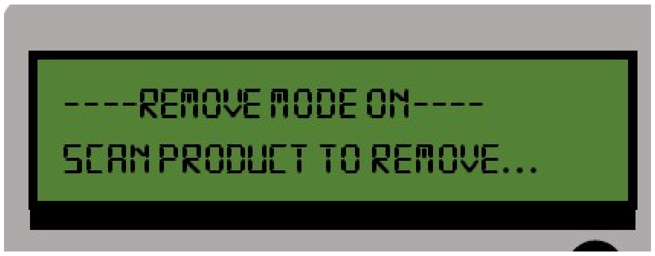

----REMOVE MODE ON----
SCAN PRODUCT TO REMOVE...

Figure 10.3

Scan the product you wish to remove and put it back on its appropriate market space.

After successful removal the screen will display a message as below:

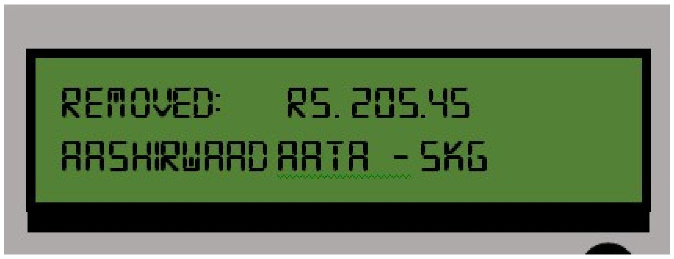

REMOVED: RS. 205.45
AASHIRWAAD AATA - 5KG

Figure 10.4

Note: You can remove only one product at a time.

If you try to remove a product which is not added in your cart, Product Not found message will be printed on the screen as shown:

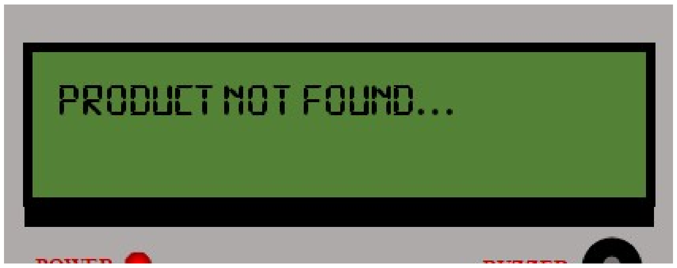
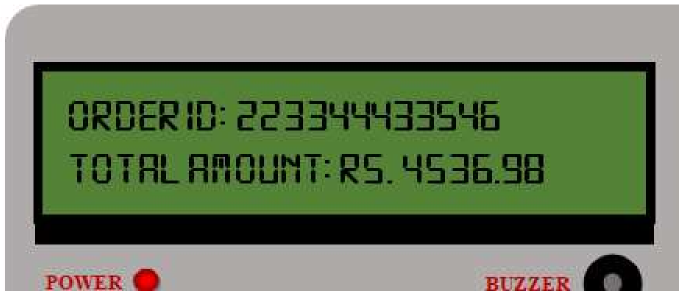

PRODUCT NOT FOUND...

Figure 10.5

<u>Step4.</u> Finally, when you are done! Press Generate Bill button. The display screen will print an *<u>Order Id</u>* and *<u>Total amount</u>* to be paid as shown:



```
ORDER ID: 223344433546
TOTAL AMOUNT: RS. 4536.98
```

<u>Figure 10.6</u>

Remember, after pressing this button, no further product can be added or deleted.

Go to billing counter and share the ***<u>order Id</u>*** with our Executive, followed by bill payment and checkout.

Thank you for shopping with ABC Supermarket.

10.3 Error messages and Troubleshooting:

1. Display Screen print: SERVER DOWN with continuous pulsed beep sound.

In this case, try resetting the device by pressing Start/Reset button. This possibly occurs because of no internet connection or issue in connecting with the database server.

2. Display Screen print: ERROR: SCAN AGAIN! With one long beep sound.

In this case, try scanning the product again. This might be happening because of issue in scanning your product or weak network connection.

3. Beep sound generated but product not added to the cart.

Try scanning the product again. It might be happening because of issue in scanning product.

For any other issue and query, contact *Smart Cart Help desk.*

****END****