**Smart Shopping Cart with Automated Billing System**

**A Project Report**

***Submitted by:***

**MOHAMMAD AAQIB AHANGER (18205135052)**

**YASIR NISAR TELI (18205135034)**

**JUNAID UL ISLAM (18205135044)**

***in partial fulfilment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

|  |
| --- |
|  |

**at**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SSM COLLEGE OF ENGINEERING**

**PARIHASPORA PATTAN, BARAMULAH**

**2018-2022**

|  |
| --- |
|  |

CERTIFICATE

This is to certify that the project entitled

**Smart Shopping Cart with Automated Billing System**

Submitted by

**MOHAMMAD AAQIB AHANGER Roll No: 18205135052**

**YASIR NISAR TELI Roll No: 18205135034**

**JUNAID UL ISLAM Roll No: 18205135044**

is the bonafide work carried out by them under supervision of Mrs Mehwish, and is approved for the partial fulfillment of the requirement for the award of the degree of Bachelor of engineering (**Computer Engineering**) of SSM College of Engineering, Kashmir affiliated to University of Kashmir, Srinagar during the academic year 2022.

This Project Report has not been earlier submitted to any other institute or university for the award of the Degree.

**Mrs Mehvish Mrs. YASMEEN**

**Internal Guide Head**

**Department of CSE Department of CSE**

**Dr. SAJAD HUSSAIN DIN**

**Principal**

ACKNOWLEDGEMENT

We take this opportunity to express our profound sense of gratitude and respect to all those who helped us throughout the duration of this project. We acknowledge the effort of those who have contributed significantly to our project. First of all we are very thankful to our God for providing us such a great opportunity to do the Project in this college and also very thankful to our parents for their regular support and guidance. We feel privileged to offer our sincere thanks and deep sense of gratitude to **DR. SAJAD HUSSAIN DIN, MRS. YASMEEN and MRS. MEHWISH** for expressing their confidence and support amd in us by letting us work on a project of this magnitude and using latest technologies and providing their support, help & encouragement in implementing this project. Last but not the least, we are grateful to all our friends for providing critical feedback & support whenever required. There are times in such projects when clock beats you time and you run out of energy, you just want to finish it once and forever, Parents and Friends made us endure such times with their unfailing humor & warm wishes.

ABSTRACT

This project targeted to reduce queue at billing counter in a shopping complex. The main objective of the system is to provide a technology oriented, low-cost, easily scalable and rugged system for assisting shopping in person. The RFID powered electronic shopping cart is built to enhance the overall shopping experience for customers in shopping mall. Upon placing the item in the cart, the customer can access an array of product information like its price and expiry date. If the product date is expired the buzzer will beep. After the completion of the process, the customer presses the finish key, the total bill is sent to the mobile application through the GSM and the customer can pay the bill through online or can also pay the bill in the cash counter. The person can also plan his shopping by buying the essential commodities as the total bill is being displayed on the screen resulting in enhanced savings. This system also has the feature to delete the scanned products by scanning it again to further optimize the shopping experience of the customer. The RFID mounted trolley is defined as “Smart Cart” and the shopping items are tagged using RFID Tags.

Keywords - Smart Cart, RFID Technology, ESP32, Buzzer, Firebase cloud, Arduino, Android Studio.

Table of Contents

[CERTIFICATE i](#_Toc121904283)

[ACKNOWLEDGEMENT ii](#_Toc121904284)

[ABSTRACT iii](#_Toc121904285)

[List of Figures vi](#_Toc121904286)

[List of Tables vii](#_Toc121904287)

[Chapter 1 INTRODUCTION 1](#_Toc121904288)

[Chapter 2 RELATED WORKS 3](#_Toc121904289)

[Chapter 3 PRODUCT DESCRIPTION 6](#_Toc121904290)

[3.1 ESP32 configuration 6](#_Toc121904291)

[3.2 LCD Interfacing 6](#_Toc121904292)

[3.3 RFID Interfacing 7](#_Toc121904293)

[3.4 Firebase Cloud 7](#_Toc121904294)

[Chapter 4 METHODOLOGY 8](#_Toc121904295)

[4.1 ESP32 Wi-Fi Module 8](#_Toc121904296)

[4.2 RFID Reader 8](#_Toc121904297)

[4.3 Arduino IDE 9](#_Toc121904298)

[4.4 Liquid Crystal Display (LCD) 9](#_Toc121904299)

[4.5 Android Studio 10](#_Toc121904300)

[4.6 Java Language 11](#_Toc121904301)

[4.7 XML: (Extensible Markup Language) 12](#_Toc121904302)

[4.8 ANDROID 13](#_Toc121904303)

[4.9 FIREBASE 13](#_Toc121904304)

[Chapter 5 CODING 15](#_Toc121904305)

[5.1 Arduino Coding 15](#_Toc121904306)

[5.2 Android Coding 25](#_Toc121904307)

[5.2.1 Android Manifest 25](#_Toc121904308)

[5.2.2 Main Activity 26](#_Toc121904309)

[5.2.3 activity\_main.xml 27](#_Toc121904310)

[Chapter 6 SCREENSHOTS 29](#_Toc121904311)

[Chapter 7 RESULTS 30](#_Toc121904312)

[REFERENCES 32](#_Toc121904313)

List of Figures

[Figure ‎3.1 ESP32 WiFi Module 6](#_Toc121904314)

[Figure ‎4.1 RFID Card Reader 8](#_Toc121904315)

[Figure ‎4.2 RFID reader interfaced with the ESP32 9](#_Toc121904316)

[Figure ‎4.3 16x2 LCD Display Module 10](#_Toc121904317)

[Figure ‎6.1 Firebase Realtime Database 29](#_Toc121904318)

[Figure ‎7.1 Smart Cart with all the components assembled 30](#_Toc121904319)

[Figure ‎7.2 Reading Data in realtime from Firebase 31](#_Toc121904320)

List of Tables

[Table ‎7.1 Connection between ESP32 and RFID Reader 30](#_Toc121904321)

[Table ‎7.2 Connection between ESP32 and I2C Module 30](#_Toc121904322)

# INTRODUCTION

In the world of Internet of Things (IoT), interactions among physical objects have become a reality. Day to day items would now be able to be outfitted with computing power and communication functionalities, permitting objects everywhere to be associated with one another. This has bought a new revolution in industrial, financial and environmental systems and triggered great challenges in data management, wireless communications and real-time decision making. There are many researches of IoT on different applications. One of the biggest IoT applications is the Smart shopping cart. The Smart Shopping system comes with the smart which is an embedded device with RFID reader for scanning the RFID tag of products, an LCD display for displaying the bill, a ESP32 wi-fi module for manipulation and sending data to server and a GSM module for wireless communication. People tend to overshoot their budget when they are shopping at a big shopping center. Moreover, they end up in long queues at the end of their shopping waiting for the products to be scanned and billed. The Smart Shopping Cart addresses the above problems with ease. It helps the customer in ensuring that he does not overshoot his pre decided budget and only buys the essential commodities actually needed by him, also the system aids in eliminating the long queues at the billing counter. It also aids in eliminating long queues at the billing counter as the items are already scanned and the customer has to just pay the bill through the mobile application.

The Smart Shopping Cart has RFID reader to scan the product where the product details are stored in the Firebase Cloud. And LCD which displays the total bill, Then the customer pays the bill through the different modes provided. The System not only displays the total cost of the commodities in the cart it also has a feature to remove any product if the customer wishes to do so. The Smart cart also eliminates the tedious process of scanning the products at the counter as this process is already done by the customer during the shopping itself. The product is also beneficial for the shopping centers as it helps them in optimizing the total workforce at their place resulting in profits in the long run.

The traditional shopping carts which are available in shopping markets are nothing but carts with a steel frame moving on wheels. Till now there has been no incorporation of electronics in order to aid the customers and enhance their shopping experience. Though there have been a lot of attempts to modernize the shopping carts all of these attempts are aimed at finding the products in the shopping market in lesser time using web servers and other utilities.

This system is aimed at doing the above in a cost-effective manner so that it is feasible to implement it in real-time. There are lot of changes that have been incorporated into the traditional shopping system.

# RELATED WORKS

A paper entitled “Arduino based smart cart” here group of people designed the smart cart in interesting way; they also implemented the feature of security in it. The design of smart cart was like mail box when item is dropped in it the door used to get closed automatically, they will open only when the mount of the product was paid, but this system had many drawbacks such as once the item is dropped and if the customer do not want to purchase it the will not unless payment is done. This system did not gain much popularity because of this drawback[1]. A paper entitled “RFID technology for IOT- based healthcare systems in smart spaces, “The group of authors has explained about the use of RFID for personal healthcare, the device is used for personal healthcare, the device is used for monitoring the users health and gives remote assistances, the users can interact through any wireless communication medium, the sensors are able to detect the physical parameter of environment such as humidity, temperature and presence of toxic agent[2].

In the world of Internet of Things (IoT), interactions among physical objects have become a reality. Day to day items would now be able to be outfitted with computing power and communication functionalities, permitting objects everywhere to be associated with one another. This has brought a new revolution in industrial, financial and environmental systems and triggered great challenges in data management, wireless communications and real-time decision making[3]. There have been various attempts which were carried out in the past to eliminate lengthy shopping lines in retail stores. One of the famous approaches is the introduction of self-checkouts where customer convenience has been improved drastically[4 ].

A paper entitled “Future internet, the internet of things application, key challenges and architecture”. The author in this paper talked about the existing development trends, the generic architecture of IoT, its distinguish features and future application they also explained about key challenge associated with future IoT such as naming and standardization, information privacy, object security, data encryption and network security. The author also discussed about ubiquitous computing where processing of information is linked with each object that is encountered and IoT Security [5]. There are multiple attempts made in 2003. *Shanmuqaprivan et al.* proposed a basic design using RFID and a barcode reader for product identification, while using ZigBee for communication [6].

Their design was similar to a mail receptacle: a chute where items are inserted and scanned, then dropped into a closed chamber. The chamber had a door on the top which can only be opened if the user has paid for the items. The design indirectly guarded against wireless communication security threats by not allowing any wireless communication - the cart was physically wired up to a point-of-sales system to pay when the user was done shopping. Ali et al. designs a smart cart system with navigation[7]. In [8] the authors have presented their work in which each commodity in the Mall will be attached with a RFID tag, and each trolley will be attached with RFID Reader which would be working on the ZigBee. A centralized system would be there for any help and queries and for the billing transaction of the products by the customers. Even the exit gates of the mall will be laced up with the RFID readers for detecting any theft. There is no user interface and hence it is not a user-friendly system. *Vrinda et al* in [9] have featured a cart equipped with an RFID reader, a ZigBee transceiver and an LCD display. This smart shopping cart keeps an account of the bill made by keeping running total of their purchases. LCD screen will show the total bill of the items present in the cart. System does not have a user interface and ZigBee is used instead of WiFi module.

In 2009, the University of Arkansas Information Technology Research Institute completed a study to determine the business value of RFID item-level tagging for day-to-day operations at a major luxury retailer. The chain’s management evaluated the use of RFID tags in the denim category. The results demonstrated that overall inventory accuracy improved by more than 27 percent, under stocks decreased by 21 percent, and overstocks decreased by 6 percent. The study also compared how long it took to count items using RFID vs. a barcode reader. With RFID, scanning 10,000 items took two hours; scanning with a barcode reader took 53 hours. This translated into an average of 4,767 counted items per hour using RFID, and 209 items per hour using a barcode system—a 96 percent reduction in cycle-counting time [10]. Public awareness of RFID was heightened in recent years when the U.S. Department of Defense (DoD) and retail giant Wal-Mart required their suppliers to use RFID technology. In January of 2005 Wal-Mart’s CIO stated that using RFID has resulted in a 26 percent reduction in out of stocks in the stores with RFID capabilities, and out of stock items that are replenished three times faster than those items not RFID tagged [11]. There is earlier work done on modernizing shopping carts. These works mainly focused on autonomous movement of the cart and finding the location of the desired product inside the shopping complex. Though these features are useful and help in reducing the time for shopping, there is a fundamental flaw that they are very expensive to implement[12].

A paper entitled “Intelligent shopping cart” here the shopping cart for speeding the billing process and used the Arduino microcontroller, and RFID tag and reader for selecting the items. Practical usage of these tag in shopping mall is expensive and requires lots of maintains [13]. A paper entitled “Integration of wearable devices in a wireless sensor network for an e-health monitoring” The author in this paper discussed about use of WSN in IoT based application, the use of WSN ine-health or human physiological monitoring, this application was used in firefighting and sports, the user is provided with interface to suggest a series of exercise to improve a sports man’s/women’s condition depending upon their context and profile, the user can interact using different interface [14].

# PRODUCT DESCRIPTION

The project is to design a smart shopping cart which helps users with their shopping. The microcontroller used to achieve the functions required is an ESP32 Wi-Fi module. It has been divided into five broad areas to achieve the targeted functionality:

## ESP32 configuration

This involves writing the code in embedded C which will enable the microcontroller to perform the various functionalities of the smart shopping cart. Figure ‎3.1 shows the development board of ESP32.

|  |
| --- |
| Figure . ESP32 WiFi Module |

## LCD Interfacing

A Liquid Crystal Display is included to display the total cost continuously. It also displays the name of the product and its price.

## RFID Interfacing

A RFID card reader is interfaced with the Arduino Uno present on the shopping cart so that the customer is able to scan the products he/she intends to buy. The card reader is also equipped with a buzzer which indicates if a product is expired.

## Firebase Cloud

Firebase is a Google’s database platform which is used to store the data of the products in the shopping complex. It also enables to send messages to the customer. All the details of the products are stored in Firebase cloud and after the customer completes the shopping, the bill is being generated and a link to pay the bill is sent through a SMS. The customer can pay the bill by clicking on the link which redirects to the payment gateway

With the above-mentioned goals in mind, the Smart Shopping Cart will have the following functions-

* The cart is initialized automatically once it is powered on. The ESP32 microcontroller starts all the devices interfaced with it.
* The LCD starts displaying the welcome message with the initial cost as “Rs.0”.
* The customer can now start his/her shopping by scanning the RFID cards attached to the products against the RFID reader one at a time.
* The system triggers a buzzer whenever it detects a product which is being expired.
* Once the product is detected by the RFID reader the ESP32 sends the card details to the central database (Firebase cloud)
* The ESP32 compares the information with the database and then returns the cost of the product which is transmitted wirelessly back to the cart.
* The LCD shows the name of the product scanned and adds its cost to the total cost counter.
* In case a product has been scanned multiple times accidentally or the customer changes his mind he/she can delete the product from the Total cost counter by scanning the same product twice

# METHODOLOGY

An ESP32 WiFi module has been used in this system. It is interfaced with an RFID reader, LCD display, GSM and buzzer. The Google’s Firebase cloud is used as a database to store all the product information. Through the GSM an SMS is sent as a payment link and list of the products and the total bill is displayed in the mobile application.

## ESP32 Wi-Fi Module

The ESP32 is used to transfer the information from the RFID reader to the cloud as the product information is to be compared with the product list in database and the corresponding data is to be displayed on the LCD. The ESP32 can be controlled from the local WiFi network. The code for the above task is written in the Arduino IDE and is being uploaded to ESP32.

## RFID Reader

The RFID reader is attached to the shopping cart which detects any tag which comes in its vicinity. The tag has a unique number assigned to it. Once the reader reads the number it passes it to the ESP32 which further communicates it for further processing. The RFID reader is connected to the serial rxd (pin 0) of the ESP32.

|  |
| --- |
| Figure . RFID Card Reader |

|  |
| --- |
|  |
| Figure . RFID reader interfaced with the ESP32 |

## Arduino IDE

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

## Liquid Crystal Display (LCD)

The conventional 16x2 character LCD is used. This type of LCD is the most ideal display device which is used popularly with the Arduino microcontrollers. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on. This LCD has two registers namely Command and Data. The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. The library name for LCD is “Liquidcrystal.h”. The library incorporates a host of functions like clear screen, scroll select, special character display, auto scroll, serial display and text direction. The contrast of the LCD can be adjusted with the help of a potentiometer. The LCD displays the initial “Smart Cart “message when the system is powered on along with the “total cost”. When the customer begins shopping the LCD displays the name of the product and the cost whenever an item is scanned. The data to display is sent to the LCD using the pins D4, D5, D6 and D7. Pins 15 and 16 are used for the backlight while pins 1 and 2 are the ground and supply respectively. Pin 6 acts as an enable necessary to actuate the LCD. Figure ‎4.3 shows the LCD interface.

|  |
| --- |
| Figure . 16x2 LCD Display Module |

## Android Studio

* Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as primary IDE for native Android application development.
* Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0. The current stable version is 3.2, which was released in September 2018.
* Features: - The following features are provided in the current stable version:
  + Android-specific refactoring and quick fixes
  + Lint tools to catch performance, usability, version compatibility and other problems
  + Pro-Guard integration and app-signing capabilities
  + Template-based wizards to create common Android designs and components
  + A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations
  + Support for building Android Wear apps
  + Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine
  + Android Virtual Device (Emulator) to run and debug apps in the Android studio.
  + Gradle-based build support
* Android Studio supports all the same programming languages of IntelliJ, and PyCharm e.g., Python, and Kotlin and Android Studio 3.0 supports "Java 7 language features and a subset of Java 8 language features that vary by platform version." External projects backport some Java 9 features

## Java Language

* Java is the name of a programming language created by Sun Microsystems in 1995. This company was bought out by Oracle Corporation, which continues to keep it up to date. The latest version is Java SE 9, which came out in 2017.
* Java, which was called Oak when it was still being developed, is object oriented, meaning it is based on objects that work together to make programs do their jobs. Java code looks like C, C++, or C#, but code written in those languages will not work in Java in most cases without being changed.
* Java runs on many different operating systems, including Android, the world's most popular mobile operating system. This makes Java platform independent. It does this by making the Java compiler turn code into Java bytecode instead of machine code. This means that when the program is executed, the Java Virtual Machine interprets the bytecode and translates it into machine code.
* Java Concepts

Java was developed to achieve 5 main goals. These are:

* + It should be simple, object-oriented, distributed and easy to learn.
  + It should be robust and secure.
  + It should be independent of a given computer architecture or platform.
  + It should be very performant.
  + It should be possible to write an interpreter for the language. The language should also support parallelism and use dynamic typing.

## XML: (Extensible Markup Language)

* In computing, Extensible Mark-up Language (XML) is a mark-up language that defines a set of rules for encoding documents in a format that is both human-readable and machine- readable. The W3C's XML 1.0 Specification and several other related specifications all of them free open standards define XML.
* The design goals of XML emphasize simplicity, generality, and usability across the Internet. It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, the language is widely used for the representation of arbitrary data structures such as those used in web services.
* Several schema systems exist to aid in the definition of XML-based languages, while programmers have developed many applications programming interfaces (APIs) to aid the processing of XML data.

## ANDROID

* Android is a mobile operating system developed by Google, based on a modified version of the Linux kernel and other open-source software and designed primarily for touch screen mobile devices such as smartphones and tablets. In addition, Google has further developed Android TV for televisions, Android Auto for cars, and Wear OS for wrist watches, each with a specialized user interface. Variants of Android are also used on game consoles, digital cameras, PCs and other electronics.
* Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, with the first commercial Android device launched in September 2008. The operating system has since gone through multiple major releases, with the current version being 9.0 "Pie", released in August 2018. The core Android source code is known as Android Open-Source Project (AOSP) and is primarily licensed under the Apache License.
* Android is also associated with a suite of proprietary software developed by Google, including core apps for services such as Gmail and Google Search, as well as the application store and digital distribution platform Google Play, and associated development platform. These apps are licensed by manufacturers of Android devices certified under standards imposed by Google, but AOSP has been used as the basis of competing Android ecosystems, such as Amazon.com, Fire OS, which use their own equivalents to the Google Mobile Services.
* Android has been the best-selling OS worldwide on smartphones since 2011 and on tablets since 2013. As of May 2017, it has over two billion monthly active users, the largest installed base of any operating system, and as of June 2018, the Google Play store features over 3.3 million apps.

## FIREBASE

* Firebase is a Backend-as-a-Service   that started as a YC11 start-up and grew up into a next-generation app-development platform on Google Cloud Platform.
* Firebase frees developers to focus crafting fantastic user experiences. You don’t need to manage servers. You don’t need to write APIs. Firebase is your server, your API and your datastore, all written so generically that you can modify it to suit most needs. Yeah, you’ll occasionally need to use other bits of the Google Cloud for your advanced applications. Firebase can’t be everything to everybody. But it gets pretty close.
* Real-time data is the way of the future. Nothing compares to it.
* Most databases require you to make HTTP calls to get and sync your data. Most databases give you data only when you ask for it.
* When you connect your app to Firebase, you’re not connecting through normal HTTP. You’re connecting through a Web-Socket. Web-Sockets are much, much faster than HTTP. You don’t have to make individual Web-Socket calls, because one socket connection is plenty. All of your data syncs automagically through that single Web-Socket as fast as your client’s network can carry it.
* Firebase sends you data as soon as it’s updated. When your client saves a change to the data, all connected clients receive the updated data almost instantly.
* Firebase Storage provides a simple way to save binary files — most often images, but it could be anything — to Google Cloud Storage directly from the client!!!
* Firebase Storage has its own system of security rules to protect your G-Cloud bucket from the masses, while granting detailed write privileges to your authenticated clients.

# CODING

## Arduino Coding

#include <Arduino.h>

#if defined(ESP32)

#include <WiFi.h>

#elif defined(ESP8266)

#include <ESP8266WiFi.h>

#endif

#include <Firebase\_ESP\_Client.h>

#include <SPI.h>

#include <MFRC522.h>

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#define SS\_PIN 5

#define RST\_PIN 22

//Provide the token generation process info.

#include "addons/TokenHelper.h"

//Provide the RTDB payload printing info and other helper functions.

#include "addons/RTDBHelper.h"

// Insert your network credentials

#define WIFI\_SSID "ssm8050"

#define WIFI\_PASSWORD "aaqib123"

// Insert Firebase project API Key

#define API\_KEY "AIzaSyB-fObEQ5kv6G-kFOOkNYEn-LX5pAkgrHo"

// Insert RTDB URLefine the RTDB URL \*/

#define DATABASE\_URL "smart-shopping-5a6f8-default-rtdb.firebaseio.com"

//Define Firebase Data object

FirebaseData fbdo;

FirebaseAuth auth;

FirebaseConfig config;

MFRC522 rfid(SS\_PIN, RST\_PIN); // Instance of the class

LiquidCrystal\_I2C lcd(0x27, 16, 2); // set the LCD address to 0x27 for a 16 chars and 2 line display

MFRC522::MIFARE\_Key key;

// Init array that will store new NUID

byte nuidPICC[4];

bool signupOK = false;

void setup() {

lcd.init(); // initialize the lcd

lcd.backlight();

lcd.setCursor(0, 0);

lcd.print("Welcome to Smart");

lcd.setCursor(0, 1);

lcd.print("Shopping Cart");

Serial.begin(9600);

SPI.begin(); // Init SPI bus

rfid.PCD\_Init(); // Init MFRC522

for (byte i = 0; i < 6; i++) {

key.keyByte[i] = 0xFF;

}

Serial.println(F("This code scan the MIFARE Classsic NUID."));

Serial.print(F("Using the following key:"));

printHex(key.keyByte, MFRC522::MF\_KEY\_SIZE);

WiFi.begin(WIFI\_SSID, WIFI\_PASSWORD);

Serial.print("Connecting to Wi-Fi");

while (WiFi.status() != WL\_CONNECTED) {

Serial.print(".");

delay(300);

}

Serial.println();

Serial.print("Connected with IP: ");

Serial.println(WiFi.localIP());

Serial.println();

/\* Assign the api key (required) \*/

config.api\_key = API\_KEY;

/\* Assign the RTDB URL (required) \*/

config.database\_url = DATABASE\_URL;

/\* Sign up \*/

if (Firebase.signUp(&config, &auth, "", "")) {

Serial.println("ok");

signupOK = true;

}

else {

Serial.printf("%s\n", config.signer.signupError.message.c\_str());

}

/\* Assign the callback function for the long running token generation task \*/

config.token\_status\_callback = tokenStatusCallback; //see addons/TokenHelper.h

Firebase.begin(&config, &auth);

Firebase.reconnectWiFi(true);

}

void loop() {

// Reset the loop if no new card present on the sensor/reader. This saves the entire process when idle.

if ( ! rfid.PICC\_IsNewCardPresent())

return;

// Verify if the NUID has been readed

if ( ! rfid.PICC\_ReadCardSerial())

return;

Serial.print(F("PICC type: "));

MFRC522::PICC\_Type piccType = rfid.PICC\_GetType(rfid.uid.sak);

Serial.println(rfid.PICC\_GetTypeName(piccType));

// Check is the PICC of Classic MIFARE type

if (piccType != MFRC522::PICC\_TYPE\_MIFARE\_MINI &&

piccType != MFRC522::PICC\_TYPE\_MIFARE\_1K &&

piccType != MFRC522::PICC\_TYPE\_MIFARE\_4K) {

Serial.println(F("Your tag is not of type MIFARE Classic."));

return;

}

int num = nuidPICC[0] + nuidPICC[1] + nuidPICC[2] + nuidPICC[3];

if (rfid.uid.uidByte[0] != nuidPICC[0] ||

rfid.uid.uidByte[1] != nuidPICC[1] ||

rfid.uid.uidByte[2] != nuidPICC[2] ||

rfid.uid.uidByte[3] != nuidPICC[3] ) {

Serial.println(F("A new card has been detected."));

// Store NUID into nuidPICC array

for (byte i = 0; i < 4; i++) {

nuidPICC[i] = rfid.uid.uidByte[i];

}

Serial.println(F("The NUID tag is:"));

Serial.print(F("In hex: "));

printHex(rfid.uid.uidByte, rfid.uid.size);

Serial.println();

Serial.print(F("In dec: "));

printDec(rfid.uid.uidByte, rfid.uid.size);

if (num == 180) {

if (Firebase.ready() && signupOK ) {

Firebase.RTDB.setInt(&fbdo, "Cart1/Item1/Price", 10);

Firebase.RTDB.setInt(&fbdo, "Cart1/Item1/Qty", 1);

}

else {

Serial.println(fbdo.errorReason());

}

}

if (num == 221) {

if (Firebase.ready() && signupOK ) {

Firebase.RTDB.setInt(&fbdo, "Cart1/Item2/Price", 20);

Firebase.RTDB.setInt(&fbdo, "Cart1/Item2/Qty", 1);

}

else {

Serial.println(fbdo.errorReason());

}

}

if (num == 770) {

if (Firebase.ready() && signupOK ) {

Firebase.RTDB.setInt(&fbdo, "Cart1/Item3/Price", 30);

Firebase.RTDB.setInt(&fbdo, "Cart1/Item3/Qty", 1);

}

else {

Serial.println(fbdo.errorReason());

}

}

if (num == 184) {

if (Firebase.ready() && signupOK ) {

Firebase.RTDB.setInt(&fbdo, "Cart1/Item4/Price", 40);

Firebase.RTDB.setInt(&fbdo, "Cart1/Item4/Qty", 1);

}

else {

Serial.println(fbdo.errorReason());

}

}

}

else {

Serial.println(F("Card read previously."));

// Store NUID into nuidPICC array

for (byte i = 0; i < 4; i++) {

nuidPICC[i] = rfid.uid.uidByte[i];

}

if (num == 180) {

if (Firebase.ready() && signupOK ) {

Firebase.RTDB.setInt(&fbdo, "Cart1/Item1/Qty", 0);

}

else {

Serial.println(fbdo.errorReason());

}

}

if (num == 221) {

if (Firebase.ready() && signupOK ) {

Firebase.RTDB.setInt(&fbdo, "Cart1/Item2/Qty", 0);

}

else {

Serial.println(fbdo.errorReason());

}

}

if (num == 770) {

if (Firebase.ready() && signupOK ) {

Firebase.RTDB.setInt(&fbdo, "Cart1/Item3/Qty", 0);

}

else {

Serial.println(fbdo.errorReason());

}

}

if (num == 184) {

if (Firebase.ready() && signupOK ) {

Firebase.RTDB.setInt(&fbdo, "Cart1/Item4/Qty", 0);

}

else {

Serial.println(fbdo.errorReason());

}

}

}

// Halt PICC

rfid.PICC\_HaltA();

// Stop encryption on PCD

rfid.PCD\_StopCrypto1();

}

/\*\*

Helper routine to dump a byte array as hex values to Serial.

\*/

void printHex(byte \*buffer, byte bufferSize) {

for (byte i = 0; i < bufferSize; i++) {

Serial.print(buffer[i] < 0x10 ? " 0" : " ");

Serial.print(buffer[i], HEX);

}

}

/\*\*

Helper routine to dump a byte array as dec values to Serial.

\*/

void printDec(byte \*buffer, byte bufferSize) {

for (byte i = 0; i < bufferSize; i++) {

Serial.print(buffer[i] < 0x10 ? " 0" : " ");

Serial.print(buffer[i], DEC);

}

}

## Android Coding

### Android Manifest

*<?*xml version="1.0" encoding="utf-8"*?>*<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools">  
  
 <application  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.SmartShopping"  
 tools:targetApi="31">  
 <activity  
 android:name=".MainActivity"  
 android:exported="true">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
  
 <meta-data  
 android:name="android.app.lib\_name"  
 android:value="" />  
 </activity>  
 </application>  
  
</manifest>

### Main Activity

package ech.android.smartshopping;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.TextView;  
  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
  
public class MainActivity extends AppCompatActivity {  
  
 TextView tv\_item;  
 Button btn\_unload;  
 FirebaseDatabase firebaseDatabase;  
 DatabaseReference databaseReference;  
 String billDisplay="";  
 int total=0;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 tv\_item = findViewById(R.id.*tv\_item*);  
 btn\_unload = findViewById(R.id.*btn\_unload*);  
 firebaseDatabase = FirebaseDatabase.*getInstance*();  
 databaseReference = firebaseDatabase.getReference("Cart1");  
  
 btn\_unload.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 databaseReference.removeValue();  
 billDisplay="";  
 total=0;  
 }  
 });  
  
 *// Read from the database* databaseReference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(DataSnapshot dataSnapshot) {  
 for(DataSnapshot ds: dataSnapshot.getChildren()){  
 Items items = ds.getValue(Items.class);  
 if(items.getQty()!=0){  
 billDisplay = billDisplay+"\n\nItem: "+ds.getKey()+"\n\tPrice: Rs."+items.getPrice()+"\n\tQty: "+items.getQty();  
 total = total+ items.getPrice();  
 }  
 }  
 tv\_item.setText("");  
 tv\_item.setText(billDisplay+"\n\n Grand Total: Rs "+total+" only.");  
 }  
  
 @Override  
 public void onCancelled(DatabaseError error) {  
  
 }  
 });  
  
 }  
}

### activity\_main.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="#ECE5E5"  
 tools:context=".MainActivity">  
  
 <TextView  
 android:id="@+id/textView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="16dp"  
 android:layout\_marginEnd="8dp"  
 android:text="Cart # 1"  
 android:textColor="#0F6CDD"  
 android:textSize="25sp"  
 android:textStyle="bold"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <androidx.cardview.widget.CardView  
 android:id="@+id/cardView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="16dp"  
 android:layout\_marginEnd="8dp"  
 android:backgroundTint="#E4C4C4"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical">  
  
 <TextView  
 android:id="@+id/tv\_item"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="TextView"  
 android:textSize="20sp" />  
 </LinearLayout>  
 </androidx.cardview.widget.CardView>  
  
 <Button  
 android:id="@+id/btn\_unload"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:text="Unload Cart"  
 app:layout\_constraintEnd\_toEndOf="@+id/cardView"  
 app:layout\_constraintStart\_toStartOf="@+id/cardView"  
 app:layout\_constraintTop\_toBottomOf="@+id/cardView" />  
</androidx.constraintlayout.widget.ConstraintLayout>

# SCREENSHOTS

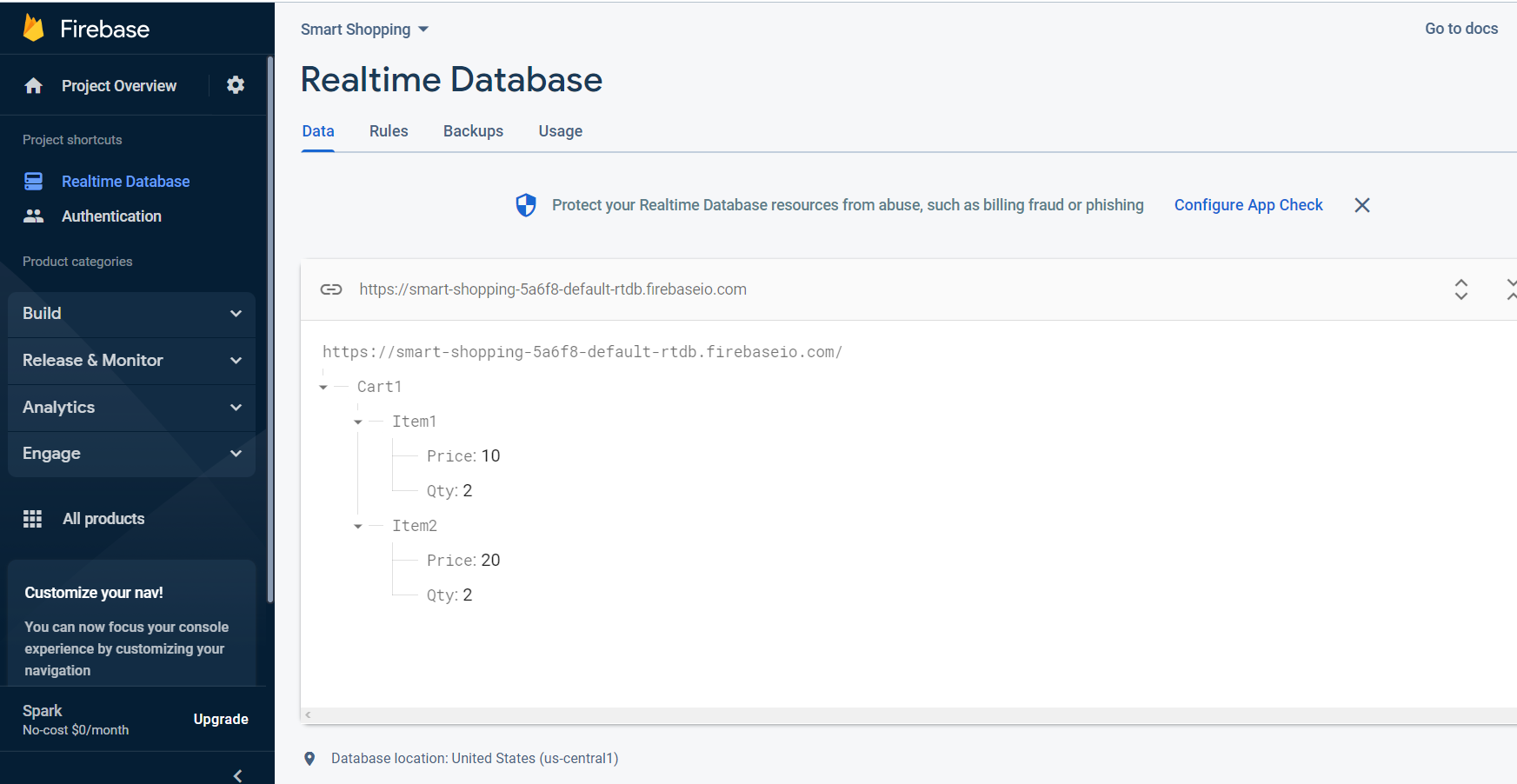


Figure . Firebase Realtime Database

# RESULTS

Table ‎7.1 Connection between ESP32 and RFID Reader

|  |  |
| --- | --- |
| ESP32 | RFID |
| 3.3v | 3.3v |
| D22 | RST |
| GND | GND |
|  | IRQ |
| D19 | MISO |
| D23 | MOSI |
| D18 | SCK |
| D5 | SDA |

Table ‎7.2 Connection between ESP32 and I2C Module

|  |  |
| --- | --- |
| ESP32 | I2C Module |
| VIN | VCC |
| GND | GND |
| D21 | SDA |
| D22 | SCL |

|  |
| --- |
| Figure . Smart Cart with all the components assembled |

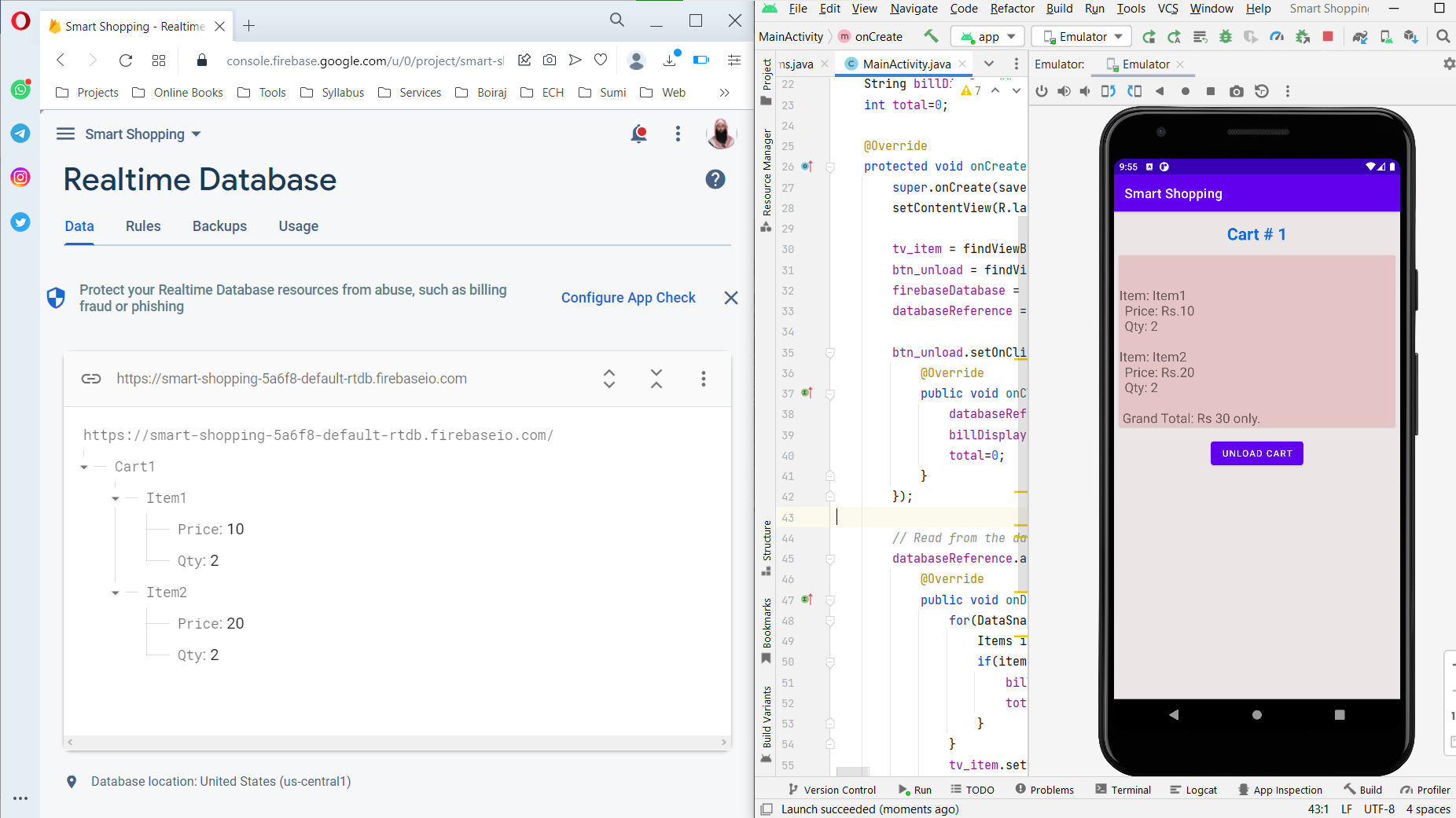


Figure . Reading Data in realtime from Firebase

REFERENCES

[1] IoT applications on Secure Smart Shopping System Ruinian Li, Tianyi Song, Nicholas Capurso, Jiguo Yu, JasonCitationinformation:DOI10.1109/JIOT.2017.2706698, IEEE Internet of Things Journal.

[2] T. Song, R. Li, X. Xing, J, Yu, and X. Cheng ,”A privacy preserving communicated protocol for iot applications in smart homes,”in to appear in International conference on Identification ,Information and Knowledge in the Internet of Things(IIKI) 2016,2016.

[3] F. Xia, L. T. Yang, L. Wang, and A. Vinel, “Internet of things,” International Journal of Communication Systems, vol. 25, no. 9, p. 1101, 2012. Dr. Mary Cherian , Disha DH, Chaithra KB,

*[4] C. N. Megan Griffith-Greene / Marketplace. (28 Jan 2016, 22 June 2017). Self Check Outs. Available:* <http://www.cbc.ca/news/business/marketplace-are-youbeing-served-1.3422736>

[5] D.Klabjan and J. Pei, “In-store one-to-one marketing,” Journal of Retailing and Consumer Services, vol. 18, no. 1, pp. 64–73, 2011.

[6] T. Shanmugapriyan, “Smart cart to recognize objects based on user intention,” International Journal of Advanced Research in Computer and Communication Engineering, vol. 2, no. 5, 2013.

[7] Z. Ali and R. Sonkusare, “Rfid based smart shopping and billing,” International Journal of Advanced Research in Computer and Communication Engineering, vol. 2, no. 12, pp. 4696–4699, 2013.

[8] Mr.P. Chandrasekar and Ms.T. Sangeetha “Smart Shopping Cart with Automatic Billing System through RFID and ZigBee”, IEEE, 2014.

[9] Ms.Vrinda, Niharika, “Novel Model for Automating Purchases using Intelligent Cart,” e-ISSN: 2278-0661, pISSN:;1; 2278-8727Volume16,Issue 1, Ver. VII (Feb. 2014), PP 23-30.

[10] [2] A.Sarac,N.Absi, S.Dauzere-Peres, ―A Literature Review of impact of RFID technologies in Supply Chain Management‖, France, March 2009.

[11] Ferguson, Renee Boucher. ―Wal-Mart's CIO Dishes on RFID at NRFTech Conference.‖ E-Week.com,Aug. 9, 2006.

[12] Johnsen, Edward L. "Shopping cart." U.S. Patent 5,250,789, issued October 5, 1993.

[13] P. Castillejo, J.-F.Martinez, J. Rodriguez-Molina, and A. Cuerva,“Integration of wearable devices in a wireless sensor network for an e-health application,” IEEE Wireless Communications, vol. 20, no. 4, pp. 38–49, 2013.

[14] N. Mitton, S. Papavassiliou, A. Puliafito, and K. S. Trivedi, “Combining cloud and sensors in a smart city environment,” EURASIP journal on Wireless Communications and Networking, vol. 2012, no. 1, p. 1, 2012.