IMPLEMENTATION OF A SMART SUPERMARKET USING RFID TECHNOLOGY

Dissertation submitted in partial fulfillment of the requirement for the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

By

RISHABH AGGARWAL
131065

UNDER THE GUIDANCE OF

DR. ASHWANI SHARMA

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

TABLE OF CONTENTS

DECLARATION BY THE SCHOLAR

I hereby declare that the work reported in the B-Tech thesis entitled **"IMPLEMENTATION OF A SMART SUPERMARKET USING RFID TECHNOLOGY"** submitted at **Jaypee University of Information Technology,Waknaghat, India,** is an authentic record of my work carried out under the supervision of DR**. ASHWANI SHARMA**. I have not submitted this work elsewhere for any other degree or diploma.

(Signature of the Scholar)

RISHABH AGGARWAL

Department of **Electronics and Communication Engineering**

Jaypee University of Information Technology, Waknaghat, India

Date-

SUPERVISOR'S CERTIFICATE

This is to certify that the work reported in the B-Tech. thesis entitled "**IMPLEMENTATION OF A SMART SUPERMARKET USING RFID TECHNOLOGY**", submitted by **RISHABH AGGARWAL** at **Jaypee University of Information Technology,Waknaghat**, **India** is a bonafide record of his / her original work carried out under my supervision. This work has not been submitted elsewhere for any other degree or diploma.

(Signature of Supervisor)

Dr. Ashwani Sharma

Assistant Professor

Date-

ACKNOWLEDGEMENT

I am at a loss of words to depict the transformative effect my Project supervisor, Dr.Ashwani Sharma, has had on me. Over the complete period of my learning he provided food for contemplation, challenged me to become a better person, taught me with examples the art of the scientific method, and guided my cause with utmost determination and honesty. I've learned key lessons from him that would help me all through my life.

I also express my sincere gratitude towards my Project Lab assistant Mr. Kamlesh Srivastava. He persistently encouraged me, provided thoughtful opinion to advance my work, was liberal with his schedules, and passionately answered my various queries.

LIST OF ACRONYMS & ABBREVIATIONS

Cm-centimeters

GHz-Gigahertz

HF-High Frequency

KHz-kilohertz

LF- Low Frequency

M-meters

MHz-Megahertz

NFC-Near Field Communication

PC-Personal Computer

RFID-Radio Frequency Identification

TM-Trade Mark

UHF-Ultra High Frequency

USA-United States of America

Wi-Fi-Wireless Fidelity

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Supermarket is a place where people buy their daily provisions ranging from food products, clothing, electrical appliances etc. Now a days number of large as well as small supermarkets has increased throughout the world due to increasing public demand & spending. Sometimes customers have problems regarding the partial information about the product on sale and waste their time needlessly at the billing counters. Continuous improvement is required in the conventional shopping methods and billing system to enhance the quality of shopping experience for the customers.

To overcome the problems stated above and to improve upon the existing system, we aim to develop a smart shopping system for best customer experience, where a customer just enter to the supermarket and grabs a smart cart and just starts shopping. The customer will browse the items traditionally and just drop them in the smart cart, the items will be automatically billed, and once the customer is done with the shopping, the customer can just checkout by making the payment using his smartphone and without any frustration of payment queues and delays.

To achieve this goal, first, we aim to design a smart shopping cart using RFID technology. This can be done by attaching RFID tags to the products and a RFID reader on the shopping cart, and also by pairing the customer's smart phone with the cart. With this system customer will have the information about price of every item that is scanned in, total price of the item and also brief about the product. This system will save time of customers and manpower required in supermarkets and also the running expenses incurred.

CHAPTER-1

INTRODUCTION

1.1 Overview

In the early times when people wanted to possess any item, they had to barter i.e. exchange any item which they owned or possessed with the item they wanted to purchase. With barter, an entity having any surplus of value, such as a measure of food or a number of livestock could directly trade that for something perceived to have comparable or greater worth or usefulness, such as a clay pot or a tool. The ability to carry out barter dealings is restricted in that it depends on a coincidence of wants. The seller of food grain has to find the buyer who wants to buy grain and who also could present in return something the seller wants to buy. There is no agreed standard measure into which both seller and buyer could exchange merchandise according to their relative value of all the various goods and services offered by other possible barter partners.

As a result, this system of purchasing became obsolete and was replaced by monetary transactions in which the goods were bought or sold for a specified or mutually agreed upon amount of money and the money thus used in the transaction can be used elsewhere to buy anything else.

But in the modern times where everyone is in a hurry, when people go to markets, they look out for ease and convenience of shopping. When they step out of their homes for purchasing desired items, they expect a good shopping experience and can get easily turned off on seeing large crowds. As a result, looking for convenience and ease of shopping, people turn to online shopping. As a result, e-commerce giants like Amazon™ and eBay™ are experiencing very huge annual growth in the number of users shopping online. This booming e-commerce market has also resulted in the adoption of e-payment options like using debit/credit cards, net banking and using third party e-wallets like PayPal™, Paytm™ etc by the people globally. These e-payment options have eased the payment methods adopted for online shopping.

But still people prefer to go to brick and mortar stores for making daily purchases like groceries, food items, fruits and vegetables etc. Today, majority of the people have a very decent standard of living as compared to about 100 years ago. So, they prefer shopping in supermarkets. But they are becoming very crowded now-a-days due to a slow checkout process which involves using barcodes, which is a 50 year old technology still very much in use till date. Continuous enhancement is required in the long-established shopping methods and billing scheme to enhance the quality of shopping experience for the customers. Also sometimes customers face trouble regarding the partial information about the merchandise on sale and waste needless time at the

billing counters. This results in large lines at the checkout counters of the supermarkets which makes the shopping experience of the customers very chaotic and frustrating. To rise above these problems stated above and to enhance the existing system, we aim to develop a smart shopping system for best customer experience, where a customer just enter to the supermarket and grabs a smart cart and carry on for shopping. The customer will browse the items traditionally and just drop them in the smart cart, the items will be automatically billed, and once the customer is done with the shopping an automatic debit from his account will be conducted. The customer will just checkout without any frustration of payment queues and delays. To achieve this goal, first, we aim to design a smart shopping cart using RFID technology. This can be done by attaching RFID tags to the merchandise and a RFID reader on the shopping cart, and also by pairing the customer's smart phone with the cart.

With this system in place, customer will have the information about price of every item that is scanned in, total price of the item and also brief about the product at his fingertips. This system will save time of customers and manpower required in supermarkets and also the running expenses incurred.

1.2 What is a RFID System?

A *RFID system* is always made up of two components (Figure 1.1):
- the *transponder*, which is located on the object to be identified;
- the interrogator or *reader*, which, depending upon the design and the technology used, may be a read or write/read device (in this book — in accordance with normal colloquial usage — the data capture device is always referred to as the *reader*, regardless of whether it can only read data or is also capable of writing).

A practical example is shown in Figure 1.2. A reader typically contains a radio frequency module (transmitter and receiver), a control unit and a coupling element to the transponder. In addition, many readers are fitted with an additional interface (RS 232, RS 485, etc.) to enable them to forward the data received to another system (PC, robot control system, etc.).

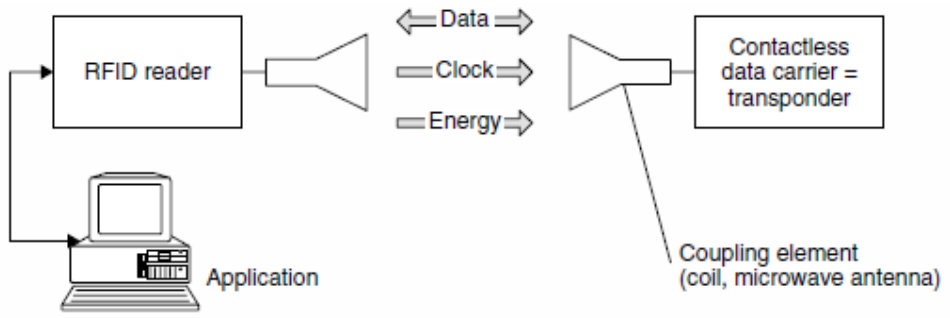

Figure 1.1 The reader and transponder are the main components of every RFID system [10]

Figure 1.2 RFID reader and contactless smart card in practical use [10]

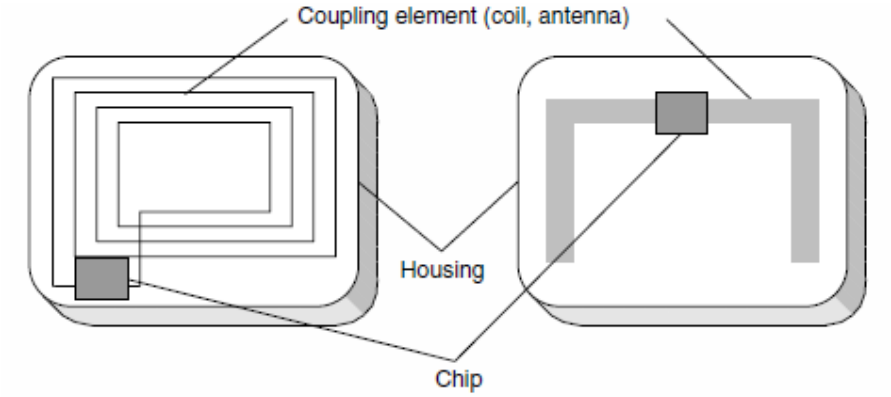

Figure 1.3 Basic layout of the RFID data-carrying device, the transponder. Left, inductively coupled transponder with antenna coil; right, microwave transponder with dipolar antenna [10]

1.2.1 Classification of RFID System

The transponder, which represents the actual *data-carrying device* of an RFID system, normally consists of a *coupling element* and an electronic *microchip* (Figure 1.3).

When the transponder, which does not usually possess its own voltage supply (battery), is not within the interrogation zone of a reader it is totally passive. The transponder is only activated when it is within the interrogation zone of a reader. The power required to activate the transponder is supplied to the transponder through the coupling unit (contactless), as are the timing pulse and data.

The transponders are differentiated by following properties-

- ***Power supply***- They are of two types-
  - *Passive Transponders*- They do not have their own power supply. Therefore, all power required for the operation of a passive transponder must be drawn from the (electrical/ magnetic) field of the reader.
  - *Active Transponders*- They incorporate a battery, which supplies all or part of the power for the operation of a microchip.

- ***Operating Frequency***- The operating frequency of a RFID system is the frequency at which the reader transmits. The transmission frequency of the transponder is disregarded. In most cases it is the same as the *transmission frequency* of the reader (load modulation, backscatter). However, the transponder's transmitting power' may be set several powers of ten lower than that of the reader.
  The different transmission frequencies are classified into the three basic ranges-

  - LF (low frequency, 30–300 kHz)
  - HF (high frequency)/RF radio frequency (3–30MHz)
  - UHF (ultra high frequency, 300MHz–3 GHz)/microwave (>3 GHz).
- ***Range***- The different types of coupling are –
  - close-coupling (0–1 cm)
  - remote-coupling (0–1 m)
  - long-range (>1m)

- ***Data Communication***- The different procedures for sending data from the transponder back to the reader can be classified into three groups-
  - The use of reflection or backscatter (the frequency of the reflected wave corresponds with the transmission frequency of the reader → frequency ratio 1:1)
  - Load modulation (the reader's field is influenced by the transponder → frequency ratio 1:1)
  - The use of subharmonics (*1/n* fold) and the generation of harmonic waves (*n*-fold) in the transponder.

1.3 Major Drawbacks of Using RFID Technology

- **Universal standardization-**The frequencies used for UHF RFID in the USA are at present mismatched with those of Europe or Japan. In addition, no upcoming standard has yet become as universal as the barcode. To tackle international trade concerns, it is compulsory to use a tag that is operational within all of the global frequency domains.

- **RFID Reader Collision-**Reader collision occurs when the signals from two or more readers overlap. The tag is unable to respond to simultaneous queries. Systems must be carefully set up to avoid this problem; many systems use an anti-collision protocol (also called a singulation protocol). Anti-collision protocols enable the tags to take turns in transmitting to a reader.

- **RFID systems can be easily disrupted-** Since RFID systems make use of the electromagnetic spectrum (like WiFi networks or cellphones), they are relatively easy to jam using energy at the right frequency. Though this would only be a nuisance for consumers in stores, it could be catastrophic in other environments where RFID is increasingly used. Also, active RFID tags can be repeatedly interrogated to wear the battery down, unsettling the system.

1.4 Organization of Thesis

The thesis is organized as follows:

Chapter-2 describes the desirable properties of RFID readers suitable for this application.

Chapter-3compares the different types of tags and readers found and choosing the one best suited for this application.

Chapter-4 describes the code used to make the android application which would act as an interface between the end-users and the shopping cart.

CHAPTER-2

DESIRABLE FEATURES OF READER & TAG TYPE

We aim to develop a system to make the supermarkets smarter by customizing a smart shopping cart for self-readable items and smart payment options using RFID technology and to help the customers in faster shopping and making faster checkouts at the supermarkets.

There has been an enormous upsurge in the popularity of RFID systems in recent years. The technical requirements of the fields of application often overlap, which means that the clear classification of suitable systems is no simple matter anymore. To make matters worse, no binding standards are as yet in place for RFID systems. It is difficult even for a specialist to retain an overview of the range of RFID systems currently on offer. Therefore, it is not always easy for users to select the system best suited to their needs.

2.1 Physical Considerations

Physical considerations for choosing the RFID system are-

- ***Data transfer rate-*** High data transfer rate is required for seamless performance of the system
- ***Multiple reads capability***-As many items are expected to be placed together in the shopping cart, the system should be able to read multiple tags almost instantaneously
- ***Cost-*** Lower cost to set up this system will lead to widespread adoption of this system by the market
- ***Performance in close proximity to liquids and metals-*** Supermarkets have many items of metal and liquids. The presence of these materials is not friendly towards transmission of radio waves. So, a system which can work well in the presence of these materials is to be chosen.

2.2 Non-Physical Considerations

Here are some non-physical considerations when selecting RFID systems.

2.2.1 Operating Frequency

RFID systems that use frequencies between approximately 100 kHz and 30 MHz operate using inductive coupling. By contrast, microwave systems in the frequency range 2.45–5.8 GHz are coupled using electromagnetic fields. The specific *absorption rate* (damping) for water or non-conductive substances is lower by a factor of 100 000 at 100 kHz than it is at 1 GHz. Therefore, virtually no absorption or damping takes place. Lower frequency HF systems are primarily used due to the better penetration of objects.

2.2.2 Range

The required range of an application is dependent upon several factors:

- the positional accuracy of the transponder;
- the minimum distance between several transponders in practical operation;
- the speed of the transponder in the interrogation zone of the reader.

2.2.3 Security Requirements

Security requirements to be imposed on a planned RFID application, i.e. *encryption* and *authentication*, should be assessed very precisely to rule out any nasty surprises in the implementation phase. For this purpose, the incentive that the system represents to a potential attacker as a means of procuring money or material goods by manipulation should be evaluated.

2.2.4 Memory Capacity

The chip size of the data carrier — and thus the price class — is primarily determined by its *memory capacity*. Therefore, permanently encoded read-only data carriers are used in price-sensitive mass applications with a low local information requirement.

CHAPTER-3

COMPARISON OF READER & TAG TYPE

This table compares all the properties of the different types of RFID technology available in the market today.

	Low Frequency(LF)	High Frequency(HF)	Ultra High Frequency(UHF)
Frequency Range:	125kHz, 134.2kHz	13.56 MHz (Global)	865 – 928 MHz (Regionally dependent)
Typical Read Range:	Up to 8cm for Texas Instruments 32mm glass, up to 7cm for EM4102 50mm disc (transponder and antenna dependant)	Between 5cm and 8cm (transponder and antenna dependant)	Between 1.5m and 2.0m (transponder and antenna dependant)
ISO Standards:	ISO 11784, ISO 11785, ISO 18000-2	ISO 15693, ISO 14443	ISO 18000-6C
Data transmission rate:	Slow data transmission rate	Higher data read rate than LF tags	Fast data transmission rate
Multiple reads capability:	Usually only single reads	Good	Excellent multiple reads capability
Supported Tags:	A wide variety of manufacturer specific transponders including NXP (Philips) HITAG, EM Microelectronic and Texas Instruments	A wide variety of transponders at 13.56 MHz including ISO 15693, ICODE (I & II) and the complete Mifare family of ISO14443 (A & B)	EPC Class 1 Gen 2 Transponders
Tag Suppliers:	NXP, Sokymat, EM Microelectronics, Texas Instruments	ACG, HID, Toshiba, iDTRONIC, Invengo, Tagsys, UPM Raflatac, X-ident and many more	Alien, Avery Dennison, Avonwood Eureka , Caen, Confidex, iDTRONIC, Intermec, Invengo, Omni-ID, Toshiba, TI, UPM Raflatac, X-ident and many more
Tag Cost:	Relatively expensive	Varies depending on type of tag	UHF tags can be very low cost (at high volumes) due to the simpler manufacturing process .

Reader Cost:	Lower (more established technology)	Lower (more established technology)	Higher (newer and more complex technology)
Reader Antenna size:	Short range mobile LF readers require only a small antenna	Short range mobile HF readers require only a small antenna	Mobile UHF reader antennas are relatively large, reduced antenna sizes can be used if compromising on read range
Read field:	Small Read Field, but easier to define – ideal for reading unique items at close range	Small Read Field, but easier to define – ideal for reading unique items at close range	Read field is much larger than LF or HF, but the radio waves can bounce off objects farther away. Excellent performance in environments with high tag density
Tag memory capacity:	Smaller memory sizes in comparison to passive HF RFID tags	Capable of relatively high memory capacity, typically 256 bits to 8 Kbytes	Smaller memory sizes in comparison to passive HF RFID tags, typically 96 bits to 1 Kbits
Performance in close proximity to liquids and metals:	Performance unaffected by surrounding water or metals	Proven track record of reliable and accurate performance of HF tags on liquids and metals	Unless properly engineered, UHF tags can be detuned by proximity to metals, liquids and human tissue. However, mount on metal UHF tags exist and in some cases outperform their HF counterparts
Security:	Low encryption capabilities	Multiple encryption/security features	Read/write protection and anti-cloning, low encryption capabilities

Table 3.1 Properties of RFID technology for different frequency bands

The following tables show the different characteristics, ISO standards and common parameters for different frequencies as shown below-

Frequency	Mode	Range	Transfer Rate	Penetrating Capability
125-135kHz	Passive	Short range (upto 0.5m)	Low	Liquid
13.56MHz	Passive	Medium range (upto 1.5m)	Moderate	Liquid
860-930MHz	Passive	Medium range (upto 5m)	Moderate to High	Liquid and Metal
433MHz	Active	Ultra long (upto 100m)	High	Liquid and Metal
2.45GHz	Active	Long range (upto 10m)	Very High	Liquid and Metal

Table 3.2 RFID characteristics for different frequencies

RFID: Radio-Frequency Identification								
by ISO(Internatinal_Organization_for_Standardization) and IEC(International_Electrontechnical_Commission)								
A complete RFID system, combined by Reader, Tags,Transponder and application software.								
Common RFID and Parameters								
Name	Chinese Items	Working Frequency	Band	Range	Standard		Chip	Application
RFID	ID	125 KHz	LF	10cm	ISO/IEC 18000-2	ISO11785	TaiWan SYRIS EM	EM 4100
	IC	13.56 MHz	HF	10cm	ISO/IEC 18000-3	ISO/IEC 14443 (Mifare 1)	Philips(Siemen)	Mifare 1 S50/70
						ISO15693		
	Bluetooth	433 MHz	UHF (Microwave)	1–25m	ISO/IEC 18000-7			
	Microwave	902-928 MHz		1–10m(ARPT)	ISO/IEC 18000-6			
	2.45G	2.45GHz		3-100M	ISO/IEC 18000-4			
		5.8GHz		200 M	ISO/IEC 18000-5			
Remark	IC and ID card	1) IC Card-Integrated Circuit(s) Card/Smart Card: data can be read and write with big capacity; Encrypted 2) ID Card-Identification Card: Read only, no data write						

Table 3.3 RFID ISO-Standards

RFID (2)								
Common RFID and Parameters								
Name	Band		Chinese Items	Regulations	Working FRQCY	Data Speed	International Standard Name	
RFID	120–150 kHz (LF)		ID	Unregulated	125 KHz	Low	Proximity Card or prox card (PICC)	EM 4100
	13.56 MHz (HF)		IC	ISM band worldwide	13.56 MHz	Low to moderate		Mifare 1 S50/70
	433 MHz		Bluetooth	Short Range Devices ISM band	433 MHz	Moderate		
	860～960MHz	865-868 MHz (Europe)	Microwave	ISM band	865-868 MHz	Moderate to high		
		902-928 MHz (North America)			902-928 MHz			
	2450-5800 MHz		2.45G	ISM band	2.45GHz	High		
	3.1–10 GHz			Ultra wide band	5.8GHz	High		

Table 3.4 Common RFID Parameters

Comparing all the current available RFID tag and reader types in the market, we have concluded that-

- **Using HF Tags and Reader (13.56 MHz) -** This is by far the best one which is suited for this particular application. Reasons are-
  - they are the cheapest available tags
  - can read multiple tags simultaneously
  - good (but not the best) data transfer rate
  - reader is relatively inexpensive
  - reliable performance in the presence of metals and liquids
  - tags have higher memory capacity
  - can support multiple security/encryption features because it is inductive coupled. It is more secure as compared to UHF readers as they are coupled using electromagnetic waves
  - Reader can be easily paired with a smartphone using either NFC or Bluetooth™

- But it also has some shortcomings which are as follows-
  - Very small read field
  - Data transfer rate not the best
  - The customers will have to scan each object they desire to buy manually by passing the tag close to the reader
  - Tag reading will be difficult for large objects

CHAPTER-4

ANDROID APPLICATION

The android app consists of the following functionalities-

- ***Support***- Devices running Android Version 4.0 or later having Bluetooth™ and internet connectivity

- ***Bluetooth***- To be used to connect the customer's android device with the shopping cart and the real-time status of the cart is displayed on the app

- ***Internet***- To be used to process the payment options once the customers are finished with their shopping

- ***User-Interface***- It will display the real-time status of the cart i.e., all the details regarding the items placed in the cart

The code used in the application is as below-

```
package com.example.android.bluetoothconnect;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothServerSocket;
import android.bluetooth.BluetoothSocket;
import android.content.Context;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;

import com.example.android.common.logger.Log;

import java.io.IOException;
import java.io.InputStream;
```

```
import java.io.OutputStream;
import java.util.UUID;

/**
* This class does all the work for setting up and managing Bluetooth
* connections with other devices. It has a thread that listens for
* incoming connections, a thread for connecting with a device, and a
* thread for performing data transmissions when connected.
*/
public class BluetoothConnectService {
// Debugging
private static final String TAG = "BluetoothConnectService";

// Name for the SDP record when creating server socket
private static final String NAME_SECURE = "BluetoothConnectSecure";
private static final String NAME_INSECURE = "BluetoothConnectInsecure";

// Unique UUID for this application
private static final UUID MY_UUID_SECURE =
UUID.fromString("fa87c0d0-afac-11de-8a39-0800200c9a66");
private static final UUID MY_UUID_INSECURE =
UUID.fromString("8ce255c0-200a-11e0-ac64-0800200c9a66");

// Member fields
private final BluetoothAdapter mAdapter;
private final Handler mHandler;
private AcceptThread mSecureAcceptThread;
private AcceptThread mInsecureAcceptThread;
private ConnectThread mConnectThread;
private ConnectedThread mConnectedThread;
private int mState;
private int mNewState;
```

```
// Constants that indicate the current connection state
public static final int STATE_NONE = 0; // we're doing nothing
public static final int STATE_LISTEN = 1; // now listening for incoming connections
public static final int STATE_CONNECTING = 2; // now initiating an outgoing connection
public static final int STATE_CONNECTED = 3; // now connected to a remote device

/**
* Constructor. Prepares a new BluetoothConnect session.
*
* @param context The UI Activity Context
* @param handler A Handler to send messages back to the UI Activity
*/
public BluetoothConnectService(Context context, Handler handler) {
mAdapter = BluetoothAdapter.getDefaultAdapter();
mState = STATE_NONE;
mNewState = mState;
mHandler = handler;
}

/**
* Update UI title according to the current state of the chat connection
*/
private matched void updateUserInterfaceTitle() {
mState = getState();
Log.d(TAG, "updateUserInterfaceTitle() " + mNewState + " -> " + mState);
mNewState = mState;

// Give the new state to the Handler so the UI Activity can update
```

```
mHandler.obtainMessage(Constants.MESSAGE_STATE_CHANGE,
mNewState, -1).sendToTarget();
}
/**
* Return the current connection state.
*/
public matched int getState() {
return mState;
}
/**
* Start the chat service. Specifically start AcceptThread to begin a
* session in listening (server) mode. Called by the Activity onResume()
*/
public matched void start() {
Log.d(TAG, "start");
// Cancel any thread attempting to make a connection
if (mConnectThread != null) {
mConnectThread.cancel();
mConnectThread = null;
}
// Cancel any thread currently running a connection
if (mConnectedThread != null) {
mConnectedThread.cancel();
mConnectedThread = null;
}
// Start the thread to listen on a BluetoothServerSocket
```

```
if (mSecureAcceptThread == null) {
mSecureAcceptThread = new AcceptThread(true);
mSecureAcceptThread.start();
}
if (mInsecureAcceptThread == null) {
mInsecureAcceptThread = new AcceptThread(false);
mInsecureAcceptThread.start();
}
// Update UI title
updateUserInterfaceTitle();
}

/**
 * Start the ConnectThread to initiate a connection to a remote device.
 *
 * @param device The BluetoothDevice to connect
 * @param secure Socket Security type - Secure (true) , Insecure (false)
 */
 public matched void connect(BluetoothDevice device, boolean secure) {
 Log.d(TAG, "connect to: " + device);

 // Cancel any thread attempting to make a connection
 if (mState == STATE_CONNECTING) {
 if (mConnectThread != null) {
 mConnectThread.cancel();
 mConnectThread = null;
}
}

// Cancel any thread currently running a connection
if (mConnectedThread != null) {
```

```
mConnectedThread.cancel();
mConnectedThread = null;
}

// Start the thread to connect with the given device
mConnectThread = new ConnectThread(device, secure);
mConnectThread.start();
// Update UI title
updateUserInterfaceTitle();
}

/**
* Start the ConnectedThread to begin managing a Bluetooth connection
*
* @param socket The BluetoothSocket on which the connection was made
* @param device The BluetoothDevice that has been connected
*/
public matched void connected(BluetoothSocket socket, BluetoothDevice
device, final String socketType) {
Log.d(TAG, "connected, Socket Type:" + socketType);

// Cancel the thread that completed the connection
if (mConnectThread != null) {
mConnectThread.cancel();
mConnectThread = null;
}

// Cancel any threa d currently running a connection
if (mConnectedThread != null) {
mConnectedThread.cancel();
mConnectedThread = null;
}
```

```
// Cancel the accept thread because we only want to connect to one device
if (mSecureAcceptThread != null) {
mSecureAcceptThread.cancel();
mSecureAcceptThread = null;
}
if (mInsecureAcceptThread != null) {
mInsecureAcceptThread.cancel();
mInsecureAcceptThread = null;
}

// Start the thread to manage the connection and perform transmissions
mConnectedThread = new ConnectedThread(socket, socketType);
mConnectedThread.start();

// Send the name of the connected device back to the UI Activity
Message msg =
mHandler.obtainMessage(Constants.MESSAGE_DEVICE_NAME);
Bundle bundle = new Bundle();
bundle.putString(Constants.DEVICE_NAME, device.getName());
msg.setData(bundle);
mHandler.sendMessage(msg);
// Update UI title
updateUserInterfaceTitle();
}

/**
* Stop all threads
*/
public matched void stop() {
Log.d(TAG, "stop");
```

```
if (mConnectThread != null) {
mConnectThread.cancel();
mConnectThread = null;
}
if (mConnectedThread != null) {
mConnectedThread.cancel();
mConnectedThread = null;
}
if (mSecureAcceptThread != null) {
mSecureAcceptThread.cancel();
mSecureAcceptThread = null;
}
if (mInsecureAcceptThread != null) {
mInsecureAcceptThread.cancel();
mInsecureAcceptThread = null;
}
mState = STATE_NONE;
// Update UI title
updateUserInterfaceTitle();
}
/**
* Write to the ConnectedThread in an unmatched approach
*
* @param out The bytes to write
* @see ConnectedThread#write(byte[])
*/
```

```
public void write(byte[] out) {
// Create temporary object
ConnectedThread r;
// Synchronize a copy of the ConnectedThread
matched (this) {
if (mState != STATE_CONNECTED) return;
r = mConnectedThread;
}
// Perform the write unmatched
r.write(out);
}

/**
* Indicate that the connection attempt failed and notify the UI Activity.
*/
private void connectionFailed() {
// Send a failure message back to the Activity
Message msg = mHandler.obtainMessage(Constants.MESSAGE_TOAST);
Bundle bundle = new Bundle();
bundle.putString(Constants.TOAST, "Unable to connect device");
msg.setData(bundle);
mHandler.sendMessage(msg);

mState = STATE_NONE;
// Update UI title
updateUserInterfaceTitle();

// Start the service over to restart listening mode
BluetoothConnectService.this.start();
}

/**
```

```
* Indicate that the connection was lost and notify the UI Activity.
*/
private void connectionLost() {
// Send a failure message back to the Activity
Message msg = mHandler.obtainMessage(Constants.MESSAGE_TOAST);
Bundle bundle = new Bundle();
bundle.putString(Constants.TOAST, "Device connection was lost");
msg.setData(bundle);
mHandler.sendMessage(msg);

mState = STATE_NONE;
// Update UI title
updateUserInterfaceTitle();

// Start the service over to restart listening mode
BluetoothConnectService.this.start();
}

/**
* This thread runs while listening for incoming connection. It behaves
* like a server-side client. It runs until a connection is accepted
* (or until cancelled).
*/
private class AcceptThread extends Thread {
// The local server socket
private final BluetoothServerSocket mmServerSocket;
private String mSocketType;

public AcceptThrea (boolean secure) {
BluetoothServerSocket tmp = null;
mSocketType = secure ? "Secure" : "Insecure";

// Create a new listening server socket
```

```
try {
if (secure) {
tmp = mAdapter.listenUsingRfcommWithServiceRecord(NAME_SECURE,
MY_UUID_SECURE);
} else {
tmp = mAdapter.listenUsingInsecureRfcommWithServiceRecord(
NAME_INSECURE, MY_UUID_INSECURE);
}
} catch (IOException e) {
Log.e(TAG, "Socket Type: " + mSocketType + "listen() failed", e);
}
mmServerSocket = tmp;
mState = STATE_LISTEN;
}

public void run() {
Log.d(TAG, "Socket Type: " + mSocketType +
"BEGIN mAcceptThread" + this);
setName("AcceptThread" + mSocketType);

BluetoothSocket socket = null;

// Listen to the server socket if we're not connected
while (mState != STATE_CONNECTED) {
try {
// This is a blocking call and will only return on a
// successful connection or an exception
socket = mmServerSocket.accept();
} catch (IOException e) {
Log.e(TAG, "Socket Type: " + mSocketType + "accept() failed", e);
break;
```

```
}
// If a connection was accepted
if (socket != null) {
matched  (BluetoothConnectService.this) {
switch (mState) {
case STATE_LISTEN:
case STATE_CONNECTING:
// Situation normal. Start the connected thread.
connected(socket, socket.getRemoteDevice(),
mSocketType);
break;
case STATE_NONE:
case STATE_CONNECTED:
// Either not ready or already connected. Terminate new socket.
try {
socket.close();
} catch (IOException e) {
Log.e(TAG, "Could not close unwanted socket", e);
}
break;
}
}
}
}
Log.i(TAG, "END mAcceptThread, socket Type: " + mSocketType);
}
public void cancel() {
Log.d(TAG, "Socket Type" + mSocketType + "cancel " + this);
```

```
try {
mmServerSocket.close();
} catch (IOException e) {
Log.e(TAG, "Socket Type" + mSocketType + "close() of server failed", e);
}
}
}

/**
* This thread runs while attempting to make an outgoing connection
* with a device. It runs straight through; the connection either
* succeeds or fails.
*/
private class ConnectThread extends Thread {
private final BluetoothSocket mmSocket;
private final BluetoothDevice mmDevice;
private String mSocketType;

public ConnectThread(BluetoothDevice device, boolean secure) {
mmDevice = device;
BluetoothSocket tmp = null;
mSocketType = secure ? "Secure" : "Insecure";

// Get a BluetoothSocket for a connection with the
// given BluetoothDevice
try {
if (secure) {
tmp = device.createRfcommSocketToServiceRecord(
MY_UUID_SECURE);
} else {
tmp = device.createInsecureRfcommSocketToServiceRecord(
```

```
MY_UUID_INSECURE);
}
} catch (IOException e) {
Log.e(TAG, "Socket Type: " + mSocketType + "create() failed", e);
}
mmSocket = tmp;
mState = STATE_CONNECTING;
}

public void run() {
Log.i(TAG, "BEGIN mConnectThread SocketType:" + mSocketType);
setName("ConnectThread" + mSocketType);

// Always cancel discovery because it will slow down a connection
mAdapter.cancelDiscovery();

// Make a connection to the BluetoothSocket
try {
// This is a blocking call and will only return on a
// successful connection or an exception
mmSocket.connect();
} catch  (IOException e) {
// Close the socket
try {
mmSocket.close();
} catch  (IOException e2) {
Log.e(TAG, "unable to close() " + mSocketType +
" socket during connection failure", e2);
}
connectionFailed();
return;
```

```
}
// Reset the ConnectThread because we're done
matched (BluetoothConnectService.this) {
mConnectThread = null;
}
// Start the connected thread
connected(mmSocket, mmDevice, mSocketType);
}
public void cancel() {
try {
mmSocket.close();
} catch (IOException e) {
Log.e(TAG, "close() of connect " + mSocketType + " socket failed", e);
}
}
}
/**
* This thread runs during a connection with a remote device.
* It handles all incoming and outgoing transmissions.
*/
private class ConnectedThread extends Thread {
private final BluetoothSocket mmSocket;
private final InputStream mmInStream;
private final OutputStream mmOutStream;
public ConnectedThread(BluetoothSocket socket, String socketType) {
Log.d(TAG, "create ConnectedThread: " + socketType);
mmSocket = socket;
InputStream tmpIn = null;
```

```
OutputStream tmpOut = null;

// Get the BluetoothSocket input and output streams
try {
tmpIn = socket.getInputStream();
tmpOut = socket.getOutputStream();
} catch (IOException e) {
Log.e(TAG, "temp sockets not created", e);
}

mmInStream = tmpIn;
mmOutStream = tmpOut;
mState = STATE_CONNECTED;
}

public void run() {
Log.i(TAG, "BEGIN mConnectedThread");
byte[] buffer = new byte[1024];
int bytes;

// Keep listening to the InputStream while connected
while (mState == STATE_CONNECTED) {
try {
// Read from the InputStream
bytes = mmInStream.read(buffer);

// Send the obtained bytes to the UI Activity
mHandler.obtainMessage(Constants.MESSAGE_READ, bytes, -1, buffer)
.sendToTarget();
} catch (IOException e) {
Log.e(TAG, "disconnected", e);
connectionLost();
```

```
break;
}
}
}
/**
* Write to  the connected  OutStream.
*
* @param buffer The bytes to write
*/
public void write(byte[] buffer) {
try {
mmOutStream.write(buffer);
// Share the  sent message  back to  the UI  Activity
mHandler.obtainMessage(Constants.MESSAGE_WRITE, -1, -1, buffer)
.sendToTarget();
} catch (IOException e) {
Log.e(TAG, "Exception during write", e);
}
}
public  void  cancel() {
try {
mmSocket.close();
} catch (IOException e) {
Log.e(TAG, "close() of connect  socket  failed", e);
}
}
}
}
```

REFERENCES

[1]S.Sai Ganesh, S.Akhila, "RFID Based Shopping Cart", International Journal of Innovative Research in Engineering & Management (IJIREM)
ISSN: 2350-0557, Volume-2, Issue-3, May-2015.

[2] Sowmyashree M S, Saritha I G, Thejaswini S, Surekha R Gondkar, "
RFID Based Automated Billing System Using Zigbee Embedded Into Shopping Cart", IOSR Journal of Business and Management (IOSR-JBM) e-ISSN: 2278-487X, p-ISSN: 2319-7668. Volume 14, Issue 6 (Nov. - Dec. 2013), PP 21-28.

[3] El Mahboul Abdelaziz, "Smart Shopping Cart System -A New Innovation For Grocery Industry", Bachelor´S Thesis |Turku University Of Applied Sciences.

[4]Raju Kumar, K. Gopalakrishna, K. Ramesha," Intelligent Shopping Cart", International Journal of Engineering Science and Innovative Technology (IJESIT) Volume 2, Issue 4, July 2013.

[5]Galande Jayshree, Rutuja Gholap, Preeti Yadav, "RFID Based Automatic Billing Trolley", International Journal of Emerging Technology and Advanced Engineering, Volume 4, Issue 3, March 2014.

[6]Kirti Chawla,"An RFID-Based Object Localization Framework and System", PhD Thesis|University of Virginia.

[7]Murulidhara N, SreeRajendra, "Automated Shopping and Billing with product Inventory Management System", July 2015, IJIRT, Volume 2, Issue 2.

[8]Udita Gangwal, Sanchita Roy, Jyotsna Bapat, "Smart Shopping Cart for Automated Billing", SENSORCOMM 2013.

[9] Mrs.Lekshmy S, Ms.Aniltta T C, Ms.Nivya Shaji, " RFID Based Shopping Trolley", International Journal of Computer Engineering In Research Trends, Volume 2, Issue 12, December-2015, pp. 1096-1099.

[10]Klaus Finkenzeller,"RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification,Second Edition",John Wiley & Sons, Ltd. ISBN: 0-470-84402-7

[11][WEB]Amazon Go-Frequently Asked Questions (FAQs) URL-https://www.amazon.com/b?ie=UTF8&node=16008589011#

[12][WEB] Data transfer via bluetooth between paired Android and Raspberry PI URL-http://stackoverflow.com/questions/39839568/data-transfer-via-bluetooth-between-paired-android-and-raspberry-pi

[13][WEB] https://github.com/googlesamples/android-BluetoothChat/blob/master/Application/src/main/java/com/example/android/bluetoothchat/BluetoothConnectionService.java

[14][WEB] RFID-Wikipedia URL-https://en.wikipedia.org/wiki/Radio-frequency_identification

[15] [WEB] RFID Comparison URL-https://www.tsl.com/support/rfid-comparison/