YEDITEPE UNIVERSITY
FACULTY OF ENGINEERING

SMART SHOPPING BASKET DESIGN with RASPBERRY PI and PRODUCT RECOGNITION MODULE

GÜLESİN AKÇIN

GRADUATION PROJECT REPORT

Department of Electrical and Electronics Engineering

Supervisor

Prof. Dr. ADNAN KAVAK

ISTANBUL,2024

YEDITEPE UNIVERSITY
FACULTY OF ENGINEERING

SMART SHOPPING BASKET with RASPBERRY PI and PRODUCT

RECOGNITION MODULE

by

GÜLESİN AKÇIN

MAY 16, 2024, Istanbul

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

YEDITEPE UNIVERSITY

Signature of Author(s) ……

Department of Electrical and Electronics Engineering

Certified By ……

Project Supervisor, Department of Electrical and Electronics Engineering

Accepted By ……

Head of the Department of Electrical and Electronics Engineering

ACKNOWLEDGEMENTS

I'd want to express my sincere gratitude to everyone who contributed to the preparation and completion of this thesis. I'd like to thank Prof. Dr. Adnan Kavak, my thesis advisor and I am grateful to my valuable professors and friends from Yeditepe University's Department of Electrical and Electronics Engineering that helped me during my study.

I'd like to thank my family for always being there for me. Their patience, understanding, and encouragement were indispensable in the effective completion of this study. I am grateful to them for always believing in me on this journey and making me feel special and lucky.

May, 2024

Gülesin Akçın

Table of Contents

ABSTRACT

The barcode technology widely used in the shopping market today is known to be inefficient and outdated. Customers only use simple shopping baskets to carry products, which causes big problems especially during sale periods . This study presents a smart shopping system developed using Raspberry Pi and RFID technology. The system automatically identifies products using passive RFID cards and updates prices instantly, allowing customers to manage their budgets. The LCD display shows the customer's total costs in real time, which speeds up the shopping process and avoids long queues. The goal of effective use of RFID technology is to reduce queues that often form in supermarkets during peak hours and provide customers with informed shopping. Inventory management is greatly improved when an RFID reader provides automatic scanning and payment, reducing the workload of retailers and creating a better shopping experience for customers. The invoicing process speeds up the transaction process by eliminating time-consuming manual processes. This solution attempts to improve the shopping experience by using a technology-based, cost-effective and business-efficient approach. This project also contributes to the concept of "smart city", which invests in information and communication technologies to use limited resources more effectively and more efficiently, saves money as a result of these investments, improves the service and quality of life it provides with this savings, reduces the carbon footprint it leaves in nature, respects the environment and natural resources, does all this with innovative and sustainable methods.

LIST OF SYMBOLS

V: Voltage

A: Ampere

Ω: Ohm

W: Watt

Hz: Hertz

ABBREVIATIONS

RFID: Radio-Frequency Identification

Pi: Raspberry Pi

LCD: Liquid Crystal Display

UID: Unique Identifier

I2C: Inter-Integrated Circuit

SPI: Serial Peripheral Interface

GPIO: General-Purpose Input/Output

PHP: Hypertext Preprocessor

OS: Operating System

USD: United States Dollar

IoT: Internet of Things

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Today, technological transformation causing significant changes in the retail industry. While digitization enables businesses to respond to customer requests more quickly and efficiently, it also provides a competitive edge by improving operational efficiency. Changes in consumer preferences are altering the shopping experience, necessitating the use of modern technology such as RFID.

RFID technology is a system that can identify and track items via radio waves. RFID systems include an RFID reader and RFID tags. RFID tags transfer data to the reader via radio waves, which is then processed in the database. This technology is commonly used for inventory management, logistics, and customer service. RFID is an effective technology in the retail business for improving inventory accuracy, supply management, and customer experience.

Today's retail systems are highly challenged due to the lack of branded technology and flexible shopping baskets that provide efficient product delivery. Since barcode scanning is a manual process, the process is time-consuming and error-prone. Long lines at cashiers, especially during peak shopping periods, can negatively affect customer satisfaction and disrupt store operations. Better solutions are needed to reduce the time consumers spend shopping, increase customer loyalty, costumer profitability and simplify the work of retailers.

1.1. Thesis Content

This thesis project intends to give new solutions to these difficulties through the development of a smart shopping system using Raspberry Pi and RFID technology, as well as to make important contributions to the retail industry by improving customer experience and operational efficiency. The smart shopping system recognizes the things that shoppers add to their baskets and instantaneously updates the total amount and enabling them to pay. This speeds up the buying experience while increasing client happiness. These prospects given by technological innovation in the retail industry boost the ability to respond to client needs more quickly and effectively, allowing retailers to maintain a competitive advantage.

2. RESEARCH OBJECTIVE

2.1 Problem Definition and Necessity of the Study

Long cashier lines in the retail industry, particularly during peak shopping hours, are a major issue that has a detrimental impact on customer satisfaction. Current shopping systems are insufficient to address these issues due to the inefficiency of barcode technology and basic shopping baskets that simply provide product transportation functions. Because barcode scanning is done manually, it takes time and is prone to errors. Long cashier lines, particularly during peak shopping hours, lower consumer happiness and have a detrimental impact on store operations. More effective solutions are required to reduce the amount of time people spend shopping and alleviate retailers' workloads. In this environment, an innovative solution is required to reduce the time lost by clients throughout the purchasing process as well as the operational issues faced by shops.

2.2 Purpose of the Study

This thesis project attempts to give creative solutions to the aforementioned challenges with a smart retail system built with Raspberry Pi and RFID technology. The system's major goal is to make customers' shopping experiences easier and more effective, eliminate cashier lines, and enhance the satisfaction of consumers. The smart shopping system improves the shopping experience by providing services such as automatic product identification, the cost updates, and instant total computation. In addition, if the system can be improved further, automatic inventory management will allow retail stores to track their stocks more successfully and reduce operational expenses.

2.3 Detailed Aims and Expected Results

This thesis study's specific objectives and anticipated outcomes can be grouped under a few broad topics.

First, RFID technology is intended for **product tracking and identification**. Products may be automatically recognized and tracked thanks to RFID tags. Inventory management is made easier by the RFID reader's transmission of tag data to the database. This increases the accuracy of inventories, maximizes stock.

Its second goal is to **enhance the customer experience**. While shopping, customers can quickly recognize the products they add to their basket and view their real-time total cost. Because there are no longer any lines of customers waiting to pay, shopping is quicker and more pleasurable with this approach. Customers have more control over their spending and a more transparent shopping experience thanks to real-time pricing updates and information about overall spend.

Its third goal is to boost **operational effectiveness**. Inventory tracking that is automated enhances stock management and accuracy. Retailers enable staff to concentrate more on customer service by minimizing burden. Because there is less chance of theft and loss, the RFID-based technology reduces operating costs. This makes it possible to run retail operations more successfully and efficiently.

Fourthly, the goal is to **support the advancement of technology**. Especially with the digital transformation, the use of technology like RFID and Raspberry Pi has gained different meanings in the retail industry, and this project enables retailers to create business models that are more adaptable and dynamic. This initiative helps the retail sector adapt to technology by creating a roadmap to meet the needs of its customers better and faster. Technological advancements boost consumer loyalty and satisfaction while making the retail sector more competitive. It allows companies that make the right investments to be differentiated positively.

Lastly, **further testing and analysis** are scheduled. To guarantee the precision and effectiveness of the system, a thorough testing and analysis process will be undertaken. The outcomes will assess the system's functionality and pinpoint areas in need of development. The purpose of these tests is to assess the system's performance under real-world scenarios.

The smart shopping system presented in this thesis study was created to enhance consumer satisfaction and boost operational effectiveness in the retail sector. Using RFID technology and a Raspberry Pi, the system seeks to improve speed, efficiency, and user-friendliness of the shopping experience. According to these goals, the insights gathered throughout the system's development and deployment would facilitate technology advancements in the retail sector and boost operational effectiveness and consumer happiness.

3. LITERATURE REVIEW

3.1 Introduction to Literature Review

The retail industry's adoption of technology has revolutionized both consumer experience and operational efficiency. Among these technologies, the Raspberry Pi and Radio Frequency Identification (RFID) have distinguished themselves as creative ways to enhance the shopping experience. This examination of the literature will compare different technologies, critically assess current smart retail systems, and highlight the advancements made by this personalized project.

3.2 Review of Current Technologies

3.2.1 RFID Technology

RFID(Radio-FrequencyIDentification) is an electronic device with small circuits incorporating a small chip and antenna. RFID consists of two parts: a tag and a reader. We may also include an RFID printer, an RFID antenna, and software for the system. An RFID tag comprises of a chip, power source, and antenna. The tag may so connect with RFID readers, transmitting and receiving data.

Tags are classified as either active or passive based on how they receive energy. Active RFID tags have a source of energy for communication and operating, whereas passive RFID tags obtain their energy from the reader. The antenna on an RFID tag allows the reader to connect with the chip. In this project, passive rfid cards will be used.

RFID chips themselves have a unique ID code and all kinds of information about the objects to be recognized can be recorded in it. Memory capacities of RFID chips can be determined according to application/need. Name of objects, product code, etc. Information can be decoded with a maximum memory capacity of 1K. High memory capacity is required when you want to load a lot of information about the object or, depending on the application, to track objects or to continuously record tracking information.

RFID chips have a unique ID number and can store a variety of information about the things to be detected. RFID chips' memory capacity can be determined based on their use or necessity. Names of things, product codes, etc. Information may be decoded using a maximum memory capacity of

1K. High memory capacity is essential when loading a large amount of information about an item or, depending on the application, monitoring objects or continually recording tracking information.

RFID chips are extremely hard for copying. Each chip has a unique ID number assigned by the manufacturer. RFID tags can have many levels of protection. Security technologies can be used to prohibit access to the information in the chip, lock it, or render it useless.

Retail operations have undergone a major transformation thanks in large part to RFID technology. The ability to wirelessly identify and monitor tangible items enhances customer service and inventory management.

Item Tracking: RFID tags enable easy tracking of products along the supply chain by providing a unique identity for each one. RFID readers in stores can determine whether products have been removed from the shelf, reducing the risk of theft and facilitating quick resolution of inventory problems.

Inventory Management: By automatically updating stock levels in real time without the need for human counting, RFID increases inventory accuracy. When products need to be reordered, automated systems keep track of stock levels and send out alerts.

Customer Experience: Personalized shopping experiences could be provided via RFID technology. things that have RFID tags, for instance, can activate sales and suggest things to customers as soon as they walk into the store. RFID-enabled smart shopping baskets save checkout times by automatically scanning merchandise.

3.2.2 Raspberry Pi

Raspberry Pi is a low-cost, high-performance British-made microcomputer. Raspberry Pi, which was first introduced in 2012, runs an open source operating system and is intended for a variety of applications.

The Raspberry Pi's hardware is based on a Broadcom-developed ARM CPU. The Raspberry Pi uses an ARM11 CPU that runs at 700 MHz, with the most recent models including a 64-bit quad-core ARM Cortex-A53 processor that can reach 1.4 GHz.Raspberry Pi is commonly equipped with a Broadcom BCM2835 chip. It comes with GB of RAM.

Raspberry Pi supports a variety of operating systems. These include Raspbian, the Raspberry Pi's own operating system, Debian, Ubuntu, Android, Chrome OS, and others.

Raspberry Pi's inexpensive cost and tiny size make it suitable for a wide range of projects. Raspberry Pi is widely utilized, particularly among students, manufacturers, and hobbyists. Raspberry Pi may be used to build home automation systems, media centers, gaming consoles, robotics, sensors, camera systems, and a variety of other electrical applications.

A small and quiet gadget accomplishes the duties of a laptop and a desktop computer. Small computers are a better for the environment and more ethical alternative to larger computers, which require more energy and complicated cooling systems.

Because of its versatility and affordability, Raspberry Pi is commonly employed in retail innovation.

Cost Effective Solutions: Compared to commercial software and hardware solutions, Raspberry Pi is an affordable option for small and medium-sized retail enterprises, requiring a minimal initial investment.

Programmability and Application Areas: Because Raspberry Pi can run on a variety of operating systems, retailers can create solutions that are specifically tailored to their needs. It facilitates the gathering and analysis of data in fields including inventory management, customer service, and the creation of shopping experiences.

Innovative Applications: The Raspberry Pi is employed in interactive kiosks, smart retail systems, and personal shopper assistants. Raspberry Pi-equipped smart shopping baskets are able to recognize additional products and instantly determine the overall cost. Customers can access product information and exclusive offers through digital signage.

As shown **Table 1**, below shows the previous studies of the similar projects and their comparative advantages and disadvantages.

3.3 Comparison of Previous Studies

Table 1 Comparison of Previous Studies

Study	Method	Advantages	Disadvantages
Automated Shopping Trolley Using Raspberry Pi (Chougule et al., 2019) [1]	Integrating RFID readers for automatic billing	Enhanced billing generation	Complexity and high cost
Design and Implementation of a Smart Shopping Trolley Using RFID Technology (Al-Hakmani & Sajan, 2020) [2]	Controlling customers' shopping within their budgets	Budget control	High Cost
Smart Shopping System Using RFID (2020) [3]	Automatic reading of RFID-tagged products	Fast product identification	Consistency issues with RFID tags
IoT Based Smart Shopping System (2020) [4]	Integrating RFID and IoT technologies	Enhanced customer experience	High internet dependency and system stability issues
Enhanced Smart Shopping System with User Profiling (Lee Meng Xian, 2018) [5]	System learning user shopping preferences and providing suggestions	Personalized experience	User privacy and data security concerns
RFID and IoT Integration for Retail Operations (2017) [6]	Integration of RFID and IoT technologies	Improved inventory management	Technological infrastructure complexity and cost
Adaptive Smart Shopping Solutions Using Raspberry Pi (2021) [7]	Raspberry Pi-based adaptive smart shopping solution	User interface and interaction	Energy consumption and performance balance challenges
Cartsmart: Customer-friendly shopping for modern times (2021) [8]	Human-following shopping cart, automatic billing	Human-following and obstacle detection	Complex structure and implementation difficulties

3.4 Critical Evaluation of Previous Studies

Despite the fact that earlier researches has significantly advanced the application of RFID and Raspberry Pi technology, a number of important problems still exist:

Cost and Complexity: These technologies must be utilized in small and medium-sized enterprises due to their high price and complex systems. restricts retailers' adoption of it.

Energy Efficiency: High energy consumption is a concern for many systems, particularly for stores that are always in operation.

System Stability: It can be challenging to deliver dependable, continuous service, particularly for systems that rely heavily on an internet connection.

User Privacy: Data security and privacy are issues that are brought up by systems that use user profiling.

3.5 Contributions and Innovations of This Project

This study addresses the shortcomings of earlier research in order to concentrate on cost effectiveness, energy efficiency, and system stability:

Cost-effectiveness: This initiative applies the technology to small and medium-sized businesses, making it available to large-scale retailers, by using inexpensive components and open-source software.

Energy Efficiency: By utilizing low-energy-consuming components, the system reduces energy expenses.

System Stability: The project may be utilized consistently in retail environments because it has been tuned to deliver dependable and continuous performance. is essential to how things work.

Improved User Experience: By lowering checkout times and enhancing the shopping experience, the intelligent shopping basket system automatically scans products and updates the total cost in real time.

4. DESIGN

4.1. Realistic Constraints and Conditions

Economy: Cost-effective solutions are offered using Raspberry Pi and RFID technology. The low cost of these components allows minimizing initial investment and increasing wide applicability. It has been stated that the project indirectly contributes to revenue increase by reducing operating costs in the retail sector and accelerating customer service processes.

Cost Analysis: Start-up and operating costs were analyzed in detail. Initial costs include hardware components such as Raspberry Pi, RFID readers, and LCD displays; It has been statedthat operating costs include permanent expenses such as maintenance, updates and energy consumption.

Environmental Issues: Because of the system's electronic data tracking and digital transactions, paper usage has been reduced, thus reducing the amount of waste. It is stated that RFID tags are produced from recyclable materials and the environmental impact is minimized.

Sustainability: It is emphasized that the components used in the project (for example, RFID tags) are produced from recyclable materials, thus contributing to environmental sustainability.

Manufacturability: Raspberry Pi and RFID readers are standardized and widely used technologies; It has been stated that these components can be easily procured and integratedinto the system, thus increasing the overall producibility of the project.

Ethical: It has been stated that the protection of customer data is one of the most important ethical dimensions of the project. The system fully complies with privacy standards and personal data protection laws during data collection and processing.

Health & Safety : RFID technology operates at low energy levels and does not have any harmful effects on human health. It has been scientifically proven that it is not. In addition, ithas been stated that all components used in the system comply with health and safety standards such as RoHS and REACH.

Social and Political: The system provides social benefits by increasing customer satisfaction and reducing checkout queues in the retail sector. At the same time, productivity increases support the competitiveness of retail businesses and contribute to the local economy.

4.2. Cost of the Design

This project's cost analysis backs up the goal of providing technical advancements at an affordable cost. The three primary cost categories that were analyzed at were initial, operating, and operational.

Startup Costs: The main components used in the project include Raspberry Pi 3 Model B+ microcontrollers, RC522 RFID reader modules and LCD screens. The Raspberry Pi 3 Model B+ costs around $35 per unit, with RFID readers costing $5 and small LCD screens costing $10. Overall, these hardware expenditures need an initial investment of around $50 per shopping basket. This estimation anticipates a total beginning cost of $500 for ten shopping baskets during the project's first phase.

Operating Costs: A one-time technical service fee of around $200 is expected for system configuration and installation. The projected cost of maintenance and upgrades, which will occur once a year, is $20 per basket. This adds up to an extra $200 in spending every year.

Operational Expenses: The system is always consuming energy, but energy expenses are substantially reduced due to the Raspberry Pi's low power consumption. Calculated at around $1 per month in energy use. Staff training is calculated at a one-time cost of $50 per employee to ensure that they can efficiently use the system. Other recurring expenses include software licensing and technical support, which total $100 per year.

These expenses are intended to maximize the system's scalability and economic efficiency. A full analysis of the expenses is required to analyze the project's economic sustainability and demonstrate that it is financially feasible, particularly for small and medium-sized retail businesses.

As shown **Table 2**, it shows the total cost breakdown.

Table 2 Cost Breakdown

Cost Category	**Description**	**Unit Cost (USD)**	**Quantity**	**Total Cost (USD)**
Initial Costs				
Raspberry Pi 3 Model B+	Microcontroller units	35	10	350
RC522 RFID Readers	RFID reader modules	5	10	50
LCD Displays	Small size LCD displays	5	10	50
Operational Costs				
System Installation	Technical service fee for setup	200	1	200
Annual Maintenance	Yearly maintenance and updates	20	10	200
Operating Expenses				
Energy Consumption	Monthly energy cost	1	12	12
Staff Training	Training fee per staff member	50	2	100
Software Licenses	Annual software license fees	100	1	100
Total Cost				1062

4.3. Engineering Standards

Multiple standards were used in this project to verify that the design and development procedures maintained international engineering standards. These standards promote the dependable and efficient use of technology by improving system security, compatibility, and effectiveness.

Bluetooth Standards: The Raspberry Pi modules used in the project rely heavily on Bluetooth connections in order to communicate with other devices. This connection complies with the Bluetooth SIG's specifications. These standards promote energy economy and data security, while also maximizing device compatibility.

Wireless Standards: RFID technology and Raspberry Pi correspond to IEEE wireless communication standards. These standards enables effective use of the frequency spectrum, signal integrity, and communication security, allowing data to be sent safely and uninterrupted in the retail environment.

Microprocessor Standards: The system uses Raspberry Pi microprocessors that are chosen in compliance with IEEE microprocessor standards. These standards allow devices to be optimized in terms of processing performance, power consumption, and multitasking capabilities.

Software Standards: The system's software has been created in accordance with ANSI and other industrial software engineering standards. While these standards increase software quality, security and maintenance simplicity, they also need to provide guidance on critical issues such as protection of user data and system stability.

4.4. Details of the Design

4.4.1 Hardware Components

Table 3 Hardware Components List

Component Name	Quantity
Raspberry Pi 3 Model B+	1
RC522 RFID NFC Module, Card and Keychain Kit	1
20x4 LCD Screen - I2C Soldered, Blue Display	1
3.56 MHz NFC Tag - ISO14443A, Ntag 213, 25 mm	5
Micro HDMI Cable	1
CAT5 Ethernet Cable - 1 M - Grey	1
13.56 MHz NFC Card - ISO14443A, 1K	1
40 Pin Male-Female M-F Jumper Cable-200 mm	1
30 cm 40 Pin Male-Male M-M Jumper Cable-300 mm	1
40 Pin Female-Female F-F Jumper Cable-200 mm	1
Micro USB Power Transfer Cable - 50cm	1

As seen in **Table 3**, the hardware equipment used in the project is listed.

Raspberry Pi 3 Model B+: A compact and powerful microcomputer **(Figure 1)** , this device is equipped with a Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit clocked at 1.4 GHz. It acts as the central unit of the smart shopping system, managing the inputs from the RFID reader and updating the display accordingly.

Figure 1 Raspberry Pi 3 Model B+ Source: [9]

RC522 RFID Module: This module **(Figure 2)** is used to read RFID tags attached to products. It runs at 13.56 MHz and communicates with the Raspberry Pi via the SPI interface. This allows the system to identify the products in the shopping basket.

Figure 2 RC522 RFID Module Source : [10]

LCD Screen: The LCD screen **(Figure 3)** which is an essential output component of the system, shows the names, prices and total cost of the products added to the basket. Increases user engagement by providing real-time feedback when products are added or removed.

Figure 3 16x2 LCD Screen Source : [11]

RFID Tags: Passive RFID tags **(Figure 4)** attached to products in the store, each tag contains a unique identifier that matches product information. The RC522 module scans these tags to identifyeach product

Figure 4 Passive RFID Tags Source : [12]

Other Components: These include jumper cables used for connections between components, Ethernet cable, MicroSD Card , 3A 5V Power Adapter and a personal computer(laptop) used for the development and programming of system components.

4.4.2 Software and Programming

Table 4 Software Components

Category	Component/Technology	Description
Programming Languages	Python	Main scripting and programming language
	Bash (Shell Script)	Used for various scripts via shell commands
Libraries and Modules	RPi.GPIO	Controls GPIO pins on the Raspberry Pi
	signal	Used for handling operating system signals
	time	Used for time-based functions
	I2C_LCD_driver	Controls LCD display via I2C protocol
	MFRC522	Used for RFID interactions
Database and Interfaces	MySQL	Database management system
	phpMyAdmin	Web-based tool to manage MySQL databases
Network and Remote Access Tools	PuTTY	SSH and Telnet client
	RealVNC Viewer	Used for graphical desktop sharing and remote access
Operating Systems and Distributions	Raspbian OS	Debian-based operating system customized for Raspberry Pi
Development Tools and Environments	Thonny IDE	IDE used for developing and running Python code
	Git	Version control system
Configuration and Environment Management	Python virtual environments	Manages dependencies and isolation for Python environments
Communication Protocols	I2C	Used for data communication between sensors and devices

As seen in the **Table 4**, the software components are listed.

Raspbian OS: Raspbian **(Figure 5)**, a Debian-based operating system modified for the Raspberry Pi 3 Model B+, is the project's primary software platform. It includes basic management of system features including hardware drivers, network settings, and security updates.

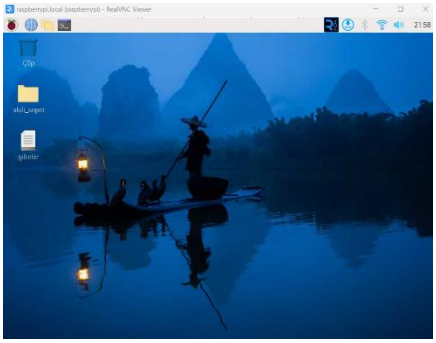

Figure 5 Screenshot of the Raspian Os Screen

Python: Python is the primary programming language utilized in this project. Basic tasks like data processing, information display on the LCD panel, and communication with the RFID module are all done with Python. Python's large library makes it easier to use the software overall and speeds up the development process.

MFRC522 Python Library: This is a customized library for communicating with the RC522 RFID module. This library streamlines the process of reading and processing data from RFID tags.

I2C_LCD_driver Python Library: This allows you to use Python to control the LCD display. This library simplifies activities like as updated and publishing text on the screen, allowing users to engage with the user interface.

PhpMyAdmin & MySQL: This allows to use to handle data such as information about the product, costs, and availability. You may manage your MySQL database using PhpMyAdmin, a web-based utility. This combination streamlines the process of updating, querying, and maintaining product data.

SSH (Secure Shell): A secure protocol for remote access and system management. It allows developers to securely connect to the Raspberry Pi via any network and do tasks like software updates, debugging, and system configuration.

VNC (Virtual Network Computing): It allows developers to remotely access the Raspberry Pi's graphical interface. This simplifies software development and testing procedures, which is especially beneficial for teams operating in many locations.

These software components enable the project to communicate smoothly with the hardware and user interface, manage data flow, and provide customers with relevant offers and shopping experiences. Each was chosen specifically to increase system reliability and effeciency.

4.5 Flow Charts

4.5.1 Smart Shopping Trolley System Flow Chart

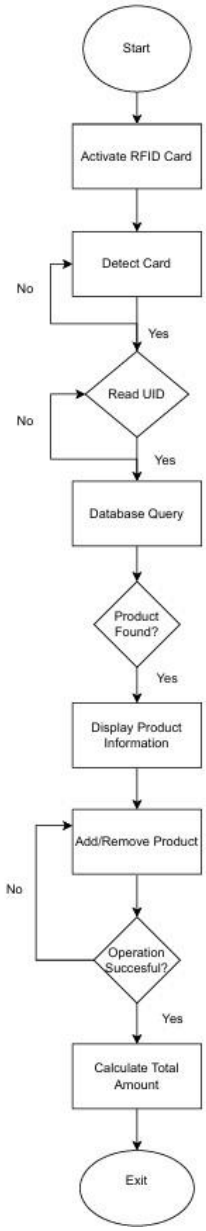

Figure 6 System Flow Chart

This flowchart **(Figure 6)** illustrates how the smart shopping basket system operates. The user activates the RFID card to start the operation. The unique identification number (UID) of the card is read when the system detects the board. If product information is found, this UID is displayed after being queried in the database. The user can add or remove items from their basket after viewing the product details. Following a successful transaction, the user can complete their purchasing by knowing the total amount in their basket.

4.5.2 User Flow Chart

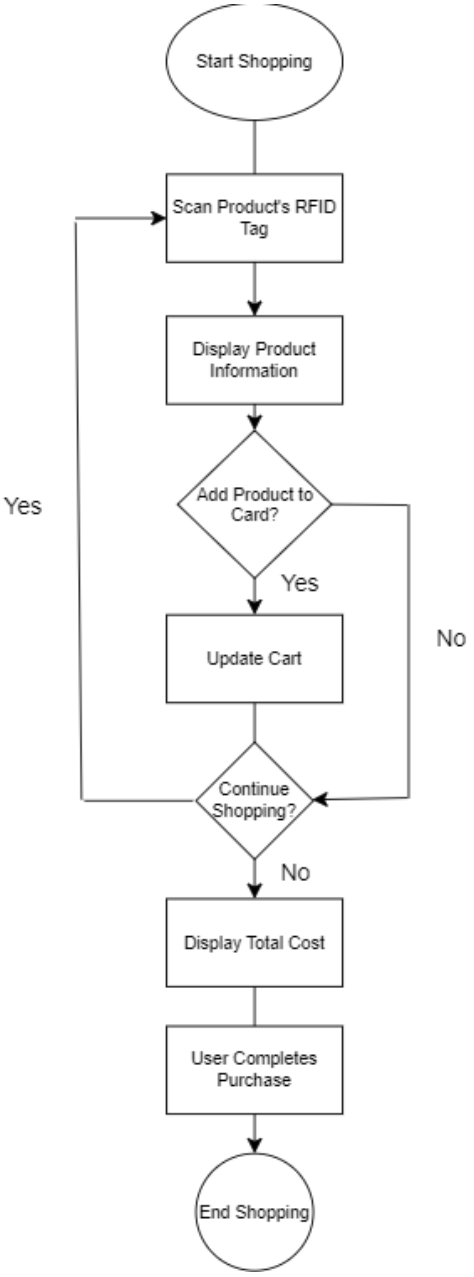

Figure 7 User Flow Chart

The user flow chart **(Figure 7)** illustrates how a consumer uses an smart shopping basket system to make purchases. The consumer scans the product's RFID tag as soon as they begin purchasing. The product details are retrieved by the system from the scanned label and shown on the screen. The customer decides whether or not to add the item to basket after seeing this information. The consumer is prompted to proceed with their purchase after the item has been placed to their basket. The customer's payment method is initiated and the total cost of the basket is computed if they decide to stop shopping.

4.5.3 Admin Panel Flow Chart

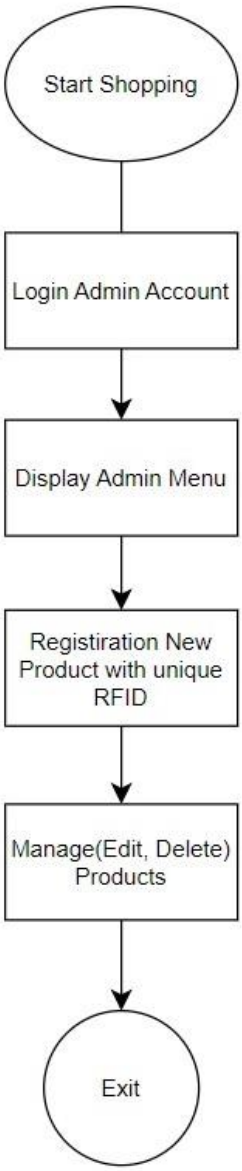

Figure 8 Admin Panel Flow Chart

The flowchart for the Admin Panel **(Figure 8)** illustrates how administrative users handle system activities. As soon as the administrator logs onto the system, the procedure begins. The admin menu is displayed after logging in, and this is where new product registration is done. Each new product is given a unique RFID tag upon registration. The administrator can alter or remove already-existing products after registration. The administrator is free to exit the system safely after the operations are finished. The steps for performing in-system navigation and product management are outlined in this flow.

4.6 Design Specifications

4.6.1 System Specifications

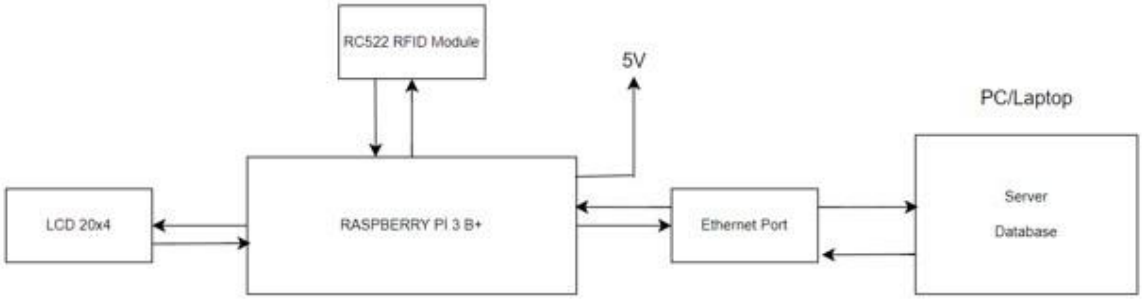

Figure 9 System Specifications Block Diagram

This **Figure 9** shows that the block diagram of the system specifications.

4.6.2 Requirements Specifications

Functional Requirements

RFID Tag Reading: The application needs to be able to read essential product data from the database by scanning RFID tags.
Data Logging: The application has to be able to enter the information from the read RFID tag into the database.
Product Management : Using the system, administrators should be able to add new products and modify or remove information about current items.
Product Information Display: The LCD screen should show the product name and price that were retrieved from the scan.

Non-Functional Requirements

User Interface Display: The program should be able to display the total number of products in the shopping list.
Product Information Presentation: The product name and price should be clearly displayed on the LCD screen within 3 seconds.
Usability: The system is suitable for non-technical users. It should also be easily usable by.
Software Development: The program should be written using the Python programming language and PHP script.

5. METHODS

5.1 Hardware Setup

5.1.1 Wiring Raspberry Pi & RC522 RFID Module

This section will examines at how to connect a Raspberry Pi 3 Model B to an RC522 RFID module, as well as the basic procedures involved in the installation process. The RC522 module reads RFID tags and converts the information into a digital format that can be processed by a microcontroller such as the Raspberry Pi.

Required Materials

- Raspberry Pi 3 Model B
- RC522 RFID Module
- Jumper cables (Male-Female)

Connection Diagram

In Table 5, the connections between Raspberry Pi and RC522 RFID Module are shown.

The suggested communication mechanism for connecting the RFID module to the Raspberry Pi was **SPI (Serial Peripheral Interface)**. A essential protocol called SPI enables fast data transfer between microcontrollers and other peripherals. The following are the actions that are taken in the connection diagram:

3.3V and GND Pins: For the RC522's energy requirements, connections to ground (GND) and 3.3V are made. This phase makes sure the system maintains electrical stability and that the module gets energy.

SPI Pins: The Raspberry Pi's GPIO pins are connected to the MISO, MOSI, SCK, and SS (Slave Select) pins. These pins are essential for the module and microcontroller's data exchange and synchronization activities. and bring it back up gradually.

Reset Pin: Software restarts and controlled resets of the module are made feasible by connecting the RST pin to the Raspberry Pi.

Table 5 RPi & RC522 Connections

RC522 RFID Module	Raspberry Pİ 3 Model B+
SDA	Pin 24/ GPIO8(CEO)
SCK	Pin 23/ GPIO11(SCKL)
MOSI	Pin 19 / GPIO10 (MOSI)
MISO	Pin 21 / GPIO9 (MISO)
IRQ	-
GND	Pin6 (GND)
RST	Pin 22/ GPIO25
3.3V	Pin 1 (3.3V)

5.1.2 Wiring Raspberry Pi & LCD Display

In this section **(Table 6)** , the connection established between Raspberry Pi 3 Model B+ and the LCD screen and the steps to realize this connection are discussed. The LCD screen is used to display user interface information and communicates with the Raspberry Pi via the I2C protocol.

Connection Details

Connections between the Raspberry Pi and the LCD screen are made using the I2C communication protocol. I2C is a two-wire serial bus system and is preferred for low-speed data transmission between microcontrollers and various peripheral devices. The pins required to establish the connections and their corresponding connections are listed below:

5V Power Connection: A connection is made from the 5V output pin (Physical Pin 2) of the Raspberry Pi to the VCC pin of the LCD screen. This connection guarantees the power required for the operation of the LCD screen.

Ground Connection (GND): A connection is made from the ground pin (Physical Pin 6) of the Raspberry Pi to the GND pin of the LCD screen. This ensures that the system remains electrically stable.

SDA (Serial Data Line) Connection: A connection is established from the SDA pin (Physical Pin 3) of the Raspberry Pi to the SDA pin of the LCD screen. This connection enables data communication.

SCL (Serial Clock Line) Connection: A connection is made from the SCL pin (Physical Pin 5) of the Raspberry Pi to the SCL pin of the LCD screen. This connection provides the clock signal required for data transmission in the I2C protocol.

Table 6 RPi & LCD Display Connections

RPi 5V Pin (Physical Pin 2)	LCD VCC
RPi GND Pin (Physical Pin 6)	LCD GND
RPi SDA (Physical Pin 3)	LCD SDA
RPi SCL(Physical Pin 5)	LCD SCL

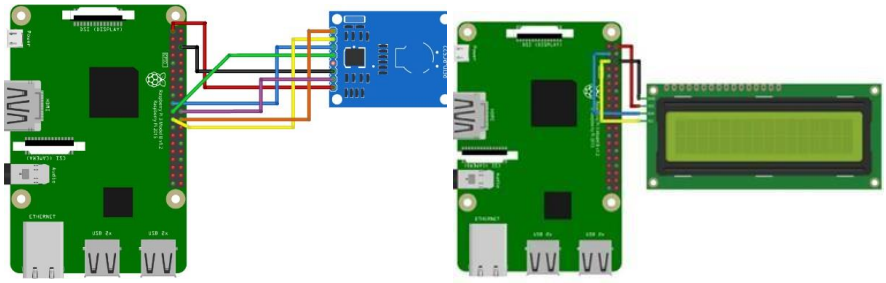

Figure 10 RPi & RC522 Circuit Diagram Source [13] *Figure 11 RPi & LCD Display Circuit Diagram Source [14]*

Figures 10 and **11** show the diagram of the system and connections.

5.2 Software Setup

5.2.1 Raspian OS on Raspberry Pi 3 & PuTTY & VNC Server

5.2.1.1 Raspbian OS Download:

The Raspbian operating system is downloaded from the official website of the Raspberry Pi Foundation. The latest version of the operating system is selected and downloaded from the "Raspberry Pi OS (previously called Raspbian)" section.

5.2.1.2 Printing Image to SD Card:

The downloaded Raspbian OS image file must be written to the SD card. An image printing tool such as Balena Etcher can be used for this process. After the program is opened, the downloaded image file is selected, the SD card is determined as the target and the printing process is started.

5.2.1.3 SSH and VNC Activation:

To enable SSH and VNC, enter the root directory of the SD card with an empty file named ssh. A file and a file named wpa_supplicant.conf are created. The wpa_supplicant.conf file contains the Wi-Fi network settings so the Raspberry Pi can connect to the network on startup: country=US

ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev update_config=1

network={

ssid="YOUR_WIFI_SSID" psk="YOUR_WIFI_PASSWORD"

key_mgmt=WPA-PSK

}

These files are added to the SD card and then the SD card is inserted into the Raspberry Pi.

5.2.1.4 Starting Raspberry Pi and Connecting to the Network:

Raspberry Pi is connected to the power supply with the SD card inserted. It automatically connects to the network and becomes accessible via SSH.

5.2.1.5 SSH Connection with PuTTY:

The IP address of the Raspberry Pi can be accessed through your router's DHCP client list or through an IP scanner tool.

PuTTY is an SSH client and can be downloaded and installed from the official website. After PuTTY is opened, the IP address of the Raspberry Pi or "**raspberrypi.local**" is written in the "**Session**" tab and the connection is initiated by clicking the "**Open**" button.

After the connection is established, the Raspberry Pi username (**'pi'**) and password (**'raspberry'**) appears on the terminal screen.

5.2.1.6 Graphical Access with VNC Server:

After entering the Raspberry Pi terminal via SSH connection, VNC is 'Enabled' under 'Interfacing Options' by running the *sudo raspi-config* command.

VNC Viewer from RealVNC's website is downloaded and installed. After VNC Viewer is opened, the IP address of the Raspberry Pi is entered and the graphical desktop is accessed by clicking the "**Connect**" button.

5.2.1.7 Installing Python and Required Libraries:

Python Installation: Raspbian OS uses Python by default. Installs as . However, if it is necessary to install the latest version or a specific version, the following commands are run through the terminal:

sudo apt-get update sudo apt-get install python3

Required Python Libraries: Libraries to be used in the project can be easily installed with pip. For example, for libraries such as **RPi.GPIO, MFRC522, I2C_LCD_driver**:

pip3 install RPi.GPIO pip3 install mfrc522

pip3 install i2c_lcd_driver

phpMyAdmin Installation:

phpMyAdmin is a web-based tool used to manage MySQL databases. For installation:

sudo apt-get install phpmyadmin

During phpMyAdmin installation, Apache is selected as the server and the MySQL database password is entered. Once the installation is complete, it can be accessed at raspberrypi website.

5.2.2.10 Git:

Git, the version control system, is necessary for managing code changes. Its setup is as follows:

sudo apt-get install git

Communication Protocols:

I2C Activation: I2C is used to communicate with devices such as LCD screens via Raspberry Pi. The raspi-config tool is used to activate I2C:

sudo raspi-config

I2C is selected and enabled from the "**Interfacing Options**" menu.

5.3 Implementation and Testing

5.3.1 I2C LCD Driver

```
# LCD Address
ADDRESS = 0x27

import smbus
from time import sleep

class i2c_device:
    def init (self, addr, port=I2CBUS):
    self.addr = addr
    self.bus = smbus.SMBus(port)
```

```
# Tek bir komut gönder
def write_cmd(self, cmd):
    self.bus.write_byte(self.addr, cmd)
    sleep(0.0001)

# Komut ve argüman gönder
def write_cmd_arg(self, cmd, data):
    self.bus.write_byte_data(self.addr, cmd, data)
    sleep(0.0001)

# Veri bloğu gönder
def write_block_data(self, cmd, data):
    self.bus.write_block_data(self.addr, cmd, data)
    sleep(0.0001)

# Tek bir bayt oku
def read(self):
    return self.bus.read_byte(self.addr)

# Veri oku
def read_data(self, cmd):
    return self.bus.read_byte_data(self.addr, cmd)

# Veri bloğu oku
def read_block_data(self, cmd):
    return self.bus.read_block_data(self.addr, cmd)

# Komutlar
LCD_CLEARDISPLAY = 0x01
LCD_RETURNHOME = 0x02
LCD_ENTRYMODESET = 0x04
LCD_DISPLAYCONTROL = 0x08
LCD_CURSORSHIFT = 0x10
LCD_FUNCTIONSET = 0x20
LCD_SETCGRAMADDR = 0x40
LCD_SETDDRAMADDR = 0x80

# Giriş modunun bayrakları
 LCD_ENTRYRIGHT = 0x00
 LCD_ENTRYLEFT = 0x02
 LCD_ENTRYSHIFTINCREMENT = 0x01
 LCD_ENTRYSHIFTDECREMENT = 0x00

# Gösterge açma/kapama kontrol bayrakları
LCD_DISPLAYON = 0x04
LCD_DISPLAYOFF = 0x00
LCD_CURSORON = 0x02
LCD_CURSOROFF = 0x00
LCD_BLINKON = 0x01
LCD_BLINKOFF = 0x00

# Gösterge/kursor kaydırma bayrakları
LCD_DISPLAYMOVE = 0x08
LCD_CURSORMOVE = 0x00
LCD_MOVERIGHT = 0x04
LCD_MOVELEFT = 0x00

# Fonksiyon set bayrakları
LCD_8BITMODE = 0x10
LCD_4BITMODE = 0x00
LCD_2LINE = 0x08
```

```
LCD_1LINE = 0x00
LCD_5x10DOTS = 0x04
LCD_5x8DOTS = 0x00

# Arka ışık kontrol bayrakları
LCD_BACKLIGHT = 0x08
LCD_NOBACKLIGHT = 0x00

En = 0b00000100 # Enable bit
Rw = 0b00000010 # Read/Write bit
Rs = 0b00000001 # Register select bit

class lcd:
    def init (self):
        # LCD cihazını başlat
        self.lcd_device = i2c_device(ADDRESS)

# LCD'yi başlat
 self.lcd_write(0x03)
 self.lcd_write(0x03)
 self.lcd_write(0x03)
 self.lcd_write(0x02)

self.lcd_write(LCD_FUNCTIONSET | LCD_2LINE | LCD_5x8DOTS |
LCD_4BITMODE)
self.lcd_write(LCD_DISPLAYCONTROL | LCD_DISPLAYON)
self.lcd_write(LCD_CLEARDISPLAY)
self.lcd_write(LCD_ENTRYMODESET | LCD_ENTRYLEFT)
sleep(0.2)

# Komutu LCD'ye yaz
    def lcd_strobe(self, data):
        self.lcd_device.write_cmd(data | En | LCD_BACKLIGHT)
        sleep(0.0005)
        self.lcd_device.write_cmd(((data & ~En) | LCD_BACKLIGHT))
        sleep(0.0001)

# LCD'ye 4 bit yaz
def lcd_write_four_bits(self, data):
    self.lcd_device.write_cmd(data | LCD_BACKLIGHT)
    self.lcd_strobe(data)

# LCD'ye komut yaz
def lcd_write(self, cmd, mode=0):
    self.lcd_write_four_bits(mode | (cmd & 0xF0))
    self.lcd_write_four_bits(mode | ((cmd << 4) & 0xF0))

# LCD'ye karakter yaz
def lcd_write_char(self, charvalue, mode=1):
    self.lcd_write_four_bits(mode | (charvalue & 0xF0))
    self.lcd_write_four_bits(mode | ((charvalue << 4) & 0xF0))

# LCD'ye metin yaz
def lcd_display_string(self, string, line=1, pos=0):
    if line == 1:
        pos_new = pos
    elif line == 2:
        pos_new = 0x40 + pos
    elif line == 3:
        pos_new = 0x14 + pos
```

```
    elif line == 4:
        pos_new = 0x54 + pos

    self.lcd_write(0x80 | pos_new)

    for char in string:
        self.lcd_write(ord(char), Rs)

# LCD'yi temizle ve başlangıç konumuna getir
def lcd_clear(self):
    self.lcd_write(LCD_CLEARDISPLAY)
    self.lcd_write(LCD_RETURNHOME)

# Arka ışığı aç/kapa
def backlight(self, state): # for state, 1 = on, 0 = off
    if state == 1:
        self.lcd_device.write_cmd(LCD_BACKLIGHT)
    elif state == 0:
        self.lcd_device.write_cmd(LCD_NOBACKLIGHT)

# Özel karakterler ekle (0-7)
def lcd_load_custom_chars(self, fontdata):
    self.lcd_write(0x40)
    for char in fontdata:
        for line in char:
            self.lcd_write_char(line)
```

This Python code is used to control an LCD display via I2C protocol with the Raspberry Pi. I2C is a bus system used to communicate with low-speed peripherals. At the beginning of the code, the I2C address of the LCD screen and which data bus will be used are determined. The**smbus** library is used to enable I2C communication, and the sleep function is used to add shortdelays between certain operations.

The code consists of two classes: **i2c_device** and **lcd**. The i2c_device class contains the necessary functions to send and receive commands and data over the I2C bus. In this class, there are operations such as sending a single command **(write_cmd),** sending commands and data **(write_cmd_arg),** sending a data block **(write_block_data),** reading a single byte (read),reading data **(read_data)** and reading a data block **(read_block_data).**

The lcd class includes functions related to initializing and controlling the LCD screen. At the beginning of the class **(_init_),** the commands required to initialize the LCD screen are sent.The **lcd_strobe** function sends a short signal while writing data to the LCD. **lcd_write_four_bits** function is used to write 4 bit data. **lcd_write** and **lcd_write_char** functions are used to write commands and characters respectively. The **lcd_display_string** function writes a string of text to the screen in a specific line and position. The **lcd_clear** function clears the screen and returns the cursor to the starting position. The backlight function turns the backlight of the LCD screen on and off. The **lcd_load_custom_chars**function allows adding special characters.

In this way, this code allows controlling an LCD screen using an I2C connection with the Raspberry Pi and can perform operations such as displaying text on the screen, clearing the screen and managing the backlight.

5.3.2 RFID with LCD Setup

```
#!/usr/bin/env  python3
# -*- coding: utf-8 -*-

#import RPi.GPIO as GPIO
import MFRC522
import signal
import I2C_LCD_driver
from time import *

continue_reading = True

mylcd = I2C_LCD_driver.lcd()

# Function to read uid and convert it to a string
def uidToString(uid):
    mystring = ""

    for i in uid:
        mystring = format(i, '02X') + mystring
    return mystring

# Capture SIGINT for cleanup when the script is aborted
def end_read(signal, frame):
    global continue_reading

    print("Ctrl+C captured, ending read.")
    continue_reading        =        False
    GPIO.cleanup()

#          Hook          the          SIGINT
signal.signal(signal.SIGINT, end_read)

# Create an object of the class MFRC522
MIFAREReader = MFRC522.MFRC522()

# Welcome message
print("Welcome to the MFRC522 data read example")
print("Press Ctrl-C to stop.")
```

```
# This loop keeps checking for chips. If one is near it will get the UID
and authenticate

while continue_reading:
    #  Scan  for  cards
    (status, TagType) =

MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)

    # If a card is found
    if status == MIFAREReader.MI_OK:
        print("Card detected")

        # Get the UID of the card

        (status, uid) = MIFAREReader.MFRC522_SelectTagSN()


        # If we have the UID, continue
        if status == MIFAREReader.MI_OK:
            print("Card read UID: %s" % uidToString(uid))
            mylcd.lcd_display_string(uidToString(uid), 1)

        else:

            print("Authentication              error")
            mylcd.lcd_display_string("Error!")
```

This Python program reads the UIDs (unique identification numbers) of RFID cards using an RFID reader and LCD screen and displays these UIDs on the screen. The program uses the **MFRC522** module to communicate with the RFID reader, the signal module to capture program interrupt, and the **I2C_LCD_driver** module to control the LCD display. The **continue_reading** variable controls whether the program continues to run or not, while the **mylcd** variable represents the object that controls the LCD screen. The **uidToString** function takes the UID of the RFID card and converts it to a string in hexadecimal format. The end_read function ensures that the program ends properly when the user presses **Ctrl+C** and clears the GPIO pins. When the program is started, the **signal.signal(signal.SIGINT, end_read)** line captures the **Ctrl+C** signal and runs the **end_read** function. The main loop of the program continuously scans RFID cards. When a card is detected, the card's UID is read and printed on the screen. If the UID is read successfully, this UID is also shown on the LCD screen. If an error occurs, the error message is printed on the screen and LCD. This loop continues until the **continue_reading** variable becomes False.

5.3.3 phpMyAdmin Database Management

Figure 12 phpMyAdmin Database Management Login Page

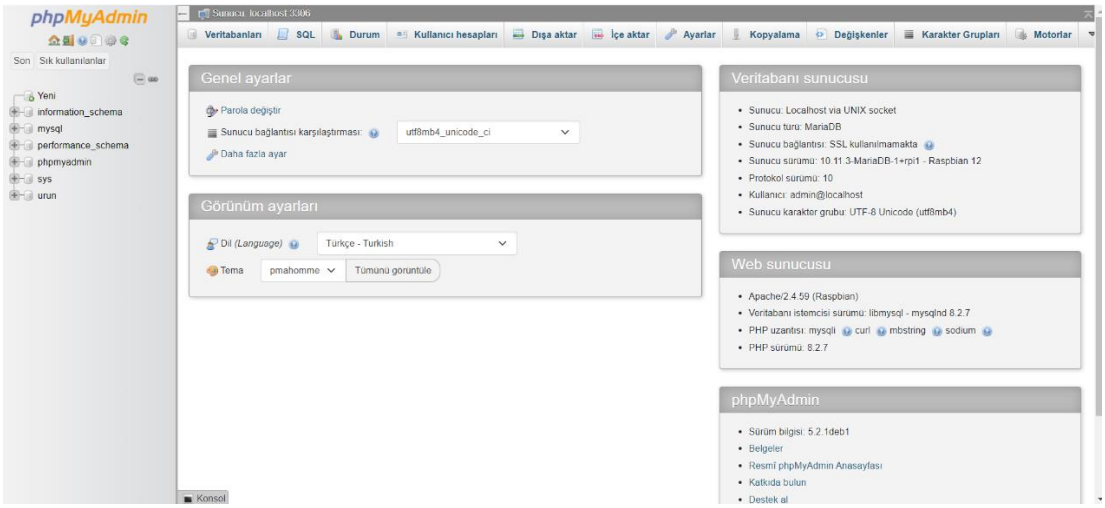

Figure 13 phpMyAdmin main page

In this process, first login to web page by entering Raspberry Pi wifi IP address then log in to the phpMyAdmin login screen by entering the username "admin" and password. On the main screen, language and theme settings can be made and available databases are listed. A database named "**product**" has been created and it contains tables named "**product_table**" and "**basket**".

In the "**product_table**" table, there are id, **product_name**, **price, product_id** and **date_time** columns for each product. This table stores the names, prices and other information of the products. For example, products such as "Tomato", "Cola", "Water", "Apple" and their prices are in this table. In thetable structure, the id column is set to AUTO_INCREMENT and the date_time column gets the currenttimestamp by default.

To add a new product, go to the "product_table" table and click the "**Add**" option. Then, you are asked to enter information such as product name, price and product_id. After entering this information, youcan add the product to the table by clicking the "**Go**" button.

The "**basket**" table is used to keep track of the products added to the shopping basket. To add a product tothe basket, the product's RFID tag is scanned and the system automatically runs the **INSERT INTO** basket **(product_id) VALUES ('[value]')** query. This process allows the scanned product to be added to the basket.

If you want to remove a product from the basket, simply scan the RFID tag of the product. The system checks whether the scanned product is in the basket and, if so, removes the product from the basket by running the query DELETE FROM basket WHERE **product_id = '[value]'**.

These operations are necessary for the management of the products and the basket in the RFID-based smart shopping system project. Users can add and remove products using the phpMyAdmin interfaceand manage their baskets in real time with the RFID reader.

5.3.4 Smart Shopping Basket Implementation

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

#import RPi.GPIO as GPIO
import MFRC522
import signal
import I2C_LCD_driver
from time import *
import pymysql

continue_reading = True
mylcd = I2C_LCD_driver.lcd()

Host = "localhost"
User = "admin"
Password = "Leonel@01"
database = "urun"

conn = pymysql.connect(host=Host, user=User, password=Password,
db=database)
cur = conn.cursor()

top = 0

# function to read uid and convert it to a string
def uidToString(uid):
    mystring = ""
```

```
    for i in uid:
        mystring = format(i, '02X') + mystring
    return mystring

# Capture SIGINT for cleanup when the script is aborted
def end_read(signal, frame):
    global continue_reading
    print("Ctrl+C captured, ending read.")
    continue_reading = False
    GPIO.cleanup()

# Hook the SIGINT
signal.signal(signal.SIGINT, end_read)

# Create an object of the class MFRC522
MIFAREReader = MFRC522.MFRC522()

# Welcome message
mylcd.lcd_display_string("Akilli sepeti", 1)
mylcd.lcd_display_string("hosgeldiniz", 2)
print("Akilli sepeti hosgeldiniz!")
time.sleep(2)
mylcd.lcd_clear()
mylcd.lcd_display_string("Sepet'e", 1)
mylcd.lcd_display_string("Urun ekleyin", 2)
print("Sepete urun ekleyin!")

# main code
while continue_reading:
    # Scan for cards
    (status, TagType) =
MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)

    # If a card is found
    if status == MIFAREReader.MI_OK:
        # Get the UID of the card
        (status, uid) = MIFAREReader.MFRC522_SelectTagSN()

        # If we have the UID, continue
        if status == MIFAREReader.MI_OK:
            #print("Card detected")
            #print("Card read UID: %s" % uidToString(uid))
            kart_id = uidToString(uid)
            #print(kart id)#for debug id card
            query = f"SELECT urun_adi,fiyat FROM urun_table WHERE (urun_id
= %s)"
            result = cur.execute(query,(kart_id,))
            conn.commit()
            if result == 1:#urun veri tabanina olup olmadigini kontrol
ediyor ve eger urun varsa
                data = cur.fetchone()
                urun_adi, f = data
                fiyat = int(f)
                if fiyat == 0:#onaylama karti veri tabani fiyati 0 olarak
tanimlanmis ve sistem onu sonlandiriyor
                    mylcd.lcd_clear()
                    mylcd.lcd_display_string("odeniyor...", 1)
                    print("odeniyor ...")
                    time.sleep(2)
                    mylcd.lcd_clear()
                    mylcd.lcd_display_string("sepet bos", 1)
```

```
                    mylcd.lcd_display_string("Urun ekleyin", 2)
                    print("Sepet bos Urun ekleyin")
                    delete = f"DELETE FROM sepet"
                    delt = cur.execute(delete)
                    conn.commit()
                    top = 0#toplam sifirlaniyor
                else:#alisveris devam ediyor
                    query1 = f"SELECT urun_id FROM sepet WHERE (urun_id =
%s)"#check if object(urun) is on basket
                    result1 = cur.execute(query1,(kart_id,))
                    if result1 == 1:#urun sepete olup olmadigini kontrol
ediyor ve eger varsa
                        delete = f"DELETE FROM sepet WHERE (urun_id = %s)"
                        delt = cur.execute(delete,(kart_id,))
                        conn.commit()
                        mylcd.lcd_clear()
                        mylcd.lcd_display_string(urun_adi, 1)
                        mylcd.lcd_display_string("cikariliyor...", 2)
                        print(urun_adi + " cikariliyor...")
                        time.sleep(1)
                        top = top - fiyat#sepeten urun fiyati toplamdan
cikartiliyor
                    elif result1 == 0:#urun sepete olup olmadigini kontrol
ediyor ve eger yoksa
                        add = f"INSERT INTO sepet (urun_id) VALUES
('{kart_id}')"
                        cur.execute(add)
                        conn.commit()
                        mylcd.lcd_clear()
                        mylcd.lcd_display_string(urun_adi + " : " +
str(fiyat) + " TL", 1)
                        mylcd.lcd_display_string("eklendi", 2)
                        print(urun_adi + " : " + str(fiyat) + " TL
eklendi")
                        time.sleep(1)
                        top = top + fiyat#sepete urun fiyati toplama
ekleniyor

                    mylcd.lcd_clear()
                    mylcd.lcd_display_string("sepet toplam :", 1)
                    mylcd.lcd_display_string(str(top), 2)
                    print("sepet toplam :" + str(top))
            else:#urun veri tabanina olup olmadigini kontrol ediyor ve eger
urun yoksa
                mylcd.lcd_clear()
                mylcd.lcd_display_string("bu urun yok", 1)
                print("Bu urun vridabanina yok...")
                time.sleep(1)
                mylcd.lcd_clear()
                mylcd.lcd_display_string("sepet toplam :", 1)
                mylcd.lcd_display_string(str(top), 2)
                print("sepet toplam :" + str(top))
```

```
else:
    print("Authentication error")
    mylcd.lcd_display_string("Error!")
```

This Python program reads the UIDs of RFID cards using an RFID reader and LCD screen, pulls productinformation from a database, and displays this information on the screen. The program uses the **MFRC522** module to communicate with the RFID reader, the signal module to catch the program interrupt, the I2C_LCD_driver module to control the LCD screen, and the **pymysql** module to connect with the MySQL database. The **continue_reading** variable controls whether the program continues torun or not, while the **mylcd** variable represents the object that controls the LCD screen.

The database connection is established with **pymysql.connect** and the cur variable manages the queries. The top variable holds the total price of the items in the basket. The **uidToString** function takes the UID of the RFID card and converts it to a string in hexadecimal format. The **end_read** function ensures that the program ends properly when the user presses **Ctrl+C** and clears the GPIO pins. Whenthe program is started, a welcome message is printed on the screen and LCD screen and the user is toldto add items to his/her basket.

The main loop continuously scans the RFID cards. If the card is detected, the UID is read and the database is checked to see if there are any products with this UID. If the product is found in the database, the product name and price are retrieved. If the price is 0 (confirmation card), the basket is emptied and the user is told to add a new product. If shopping continues, it is checked whether the product is already in the basket. If the product is in the basket, it is removed; if not, it is added to the basket. The basket total is updated and printed on the LCD screen. If the product is not found in the database,the user is notified that the product was not found and the basket total is shown. If the UID cannot be read, the error message is printed on the screen and LCD. This loop continues until the **continue_reading** variable becomes False. In this way, the program serves to add or remove productsto a basket by reading RFID cards, calculate the total price and display this information on the LCD screen. Product information is obtained and basket transactions are managed using the database connection.

6. RESULTS AND DISCUSSION

6.1 Results

6.1.1 Product Scanning & Adding

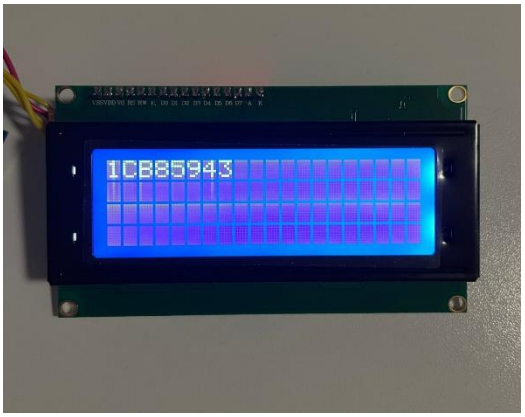

Figure 14 Screenshot of RC522 reading card ID

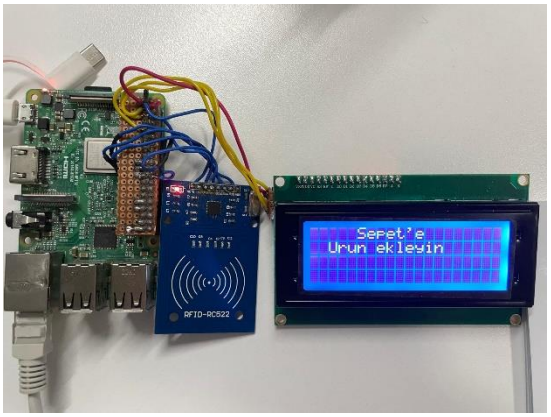

Figure 15 Screenshot of "Add Product to Basket"

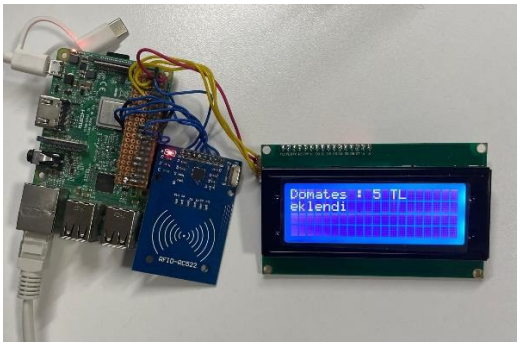

Figure 16 Screenshot of the adding products to basket

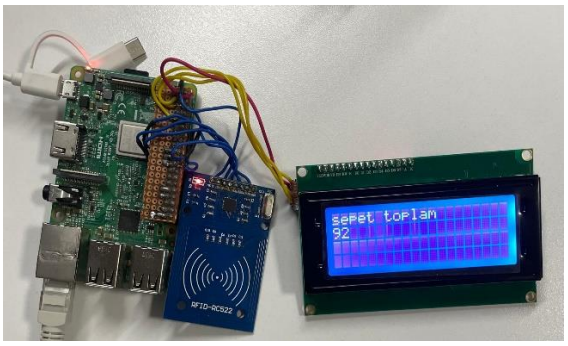

Figure 17 Screenshot of the basket total

As seen in **Figure 14-15-16-17**, by scanning the RFID tags on products, the system recognizes them and shows them on the LCD screen. The user can immediately view transactions on the LCD screen while adding or removing items from the basket. During the initial phase, the product's unique identification number (UID) is shown on the LCD screen once the system has read the RFID tag. This stage shows that the product's identification and scanning were successful. The "Add Item to Basket" notification then shows on the LCD screen when the user adds a new product to the basket, signaling that the system is prepared to recognize a new product.

When the user scans the RFID tag, the name and price of the product are displayed on the LCD screen. For example, the message "Tomato: $5 - Added" indicates that the tomato has been successfully added to the basket and its price is $5. When the user wants to see the total amount of the products in the basket, the message "Basket Total: XX TL" appears on the LCD screen. This shows the user the total cost of all added products. This process provides instant feedback tothe user, making the shopping process more controllable and efficient.

6.1.2 Product Removal

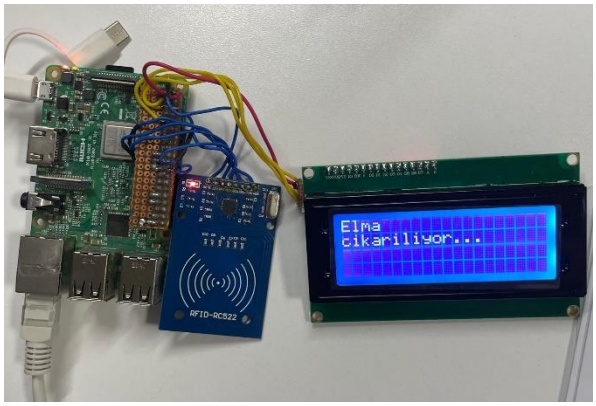

Figure 18 Screenshot of the product removal

When the user removes an item from the basket, the system scans the RFID tag and identifies the item. During this process, the message "Removing apple..." appears on the LCD screen **(Figure 18)**. This message indicates that the user removed the apple from the basket and the system successfully saved this change. Such instant feedback gives the user full control over the shopping process and confirms that transactions were carried out correctly.

6.1.3 Payment & Confirmation

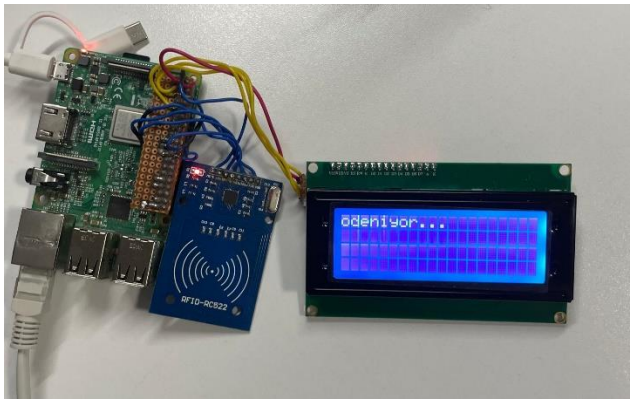

Figure 19 Screenshot of the checkout process

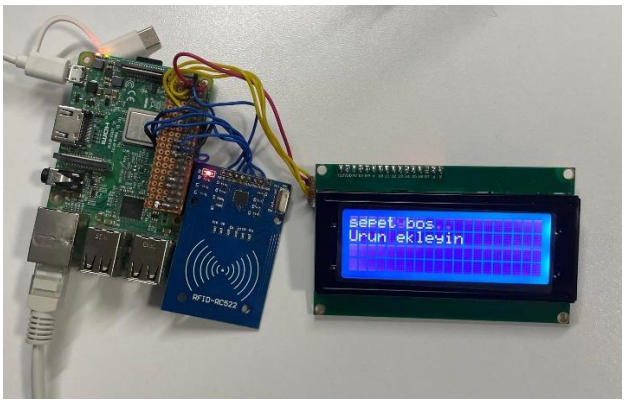

Figure 20 Screenshot of the after checkout process

In order for the shopping process to be completed and the "Paying..." message to appear **(Figure 19)**,the user scans a specially designated confirmation card (or payment card) in the system. This card allows the system to proceed to the payment stage. The verification card, unlike other productRFID tags, is programmed to trigger a specific action in the system. The logic of this card is to indicate that the shopping has ended and the payment process must begin.

When the system reads the RFID tag of this verification card, it checks the identification number (UID) of the card in the database. The UID of this card matches a predetermined UID in the system. If this match is verified, the system initiates the payment process and displays the message "Paying..." on the LCD screen. This stage indicates that the user has completed his shopping and proceeded to the payment process. Once the checkout process is completed,the system resets the basket and shows the user the "Basket Empty - Add Item" message **(Figure 20)** so the user can start a new shopping. The following functions have been successfully achieved with the developed smart shopping basket system:

- Displaying product information on the screen by reading RFID tags.
- Accurately tracking the products added and removed from the basket.
- Instant calculation and display of the total balance.
- Providing a user-friendly interface.

As a result of the tests, it was observed that the system has a high accuracy rate in reading RFID tags and displaying product information accurately. In addition, it has been determined that total balance calculations and product addition/removal operations are carried out quickly and accurately.

6.2 Discussion

6.2.1 Interpretation of the Results

The collected results are used to address the problem raised in the first stage of the project and appear to be suitable for its resolution. The smart shopping basket solution has improved the client experience while also increasing operational efficiency in the retail sector. However, some aspects of the work require additional improvements.

6.2.2 Satisfactory Parts

The solution produced satisfactory parts in terms of improving customer experience and increasing operational efficiency. In terms of RFID reading accuracy, the system successfully read RFID tags and displayed accurate product information on the screen. This guaranteed that customers received accurate information throughout the purchase process. Furthermore, the instant calculation and presentation of the total balance of products added and deleted from the allowed clients to more successfully manage their budgets. The user interface is intuitive and simple to use, giving users an excellent shopping experience.

6.2.3 Unsatisfactory Parts

However, it has been observed that the system is insufficient in certain places. There have been issues with the RFID scanning range, particularly at long distances and when there are obstructing objects, resulting in lower reading accuracy. It is believed that this problem can be remedied by boosting the power of the RFID reader or utilizing stronger antennas. In addition, some slowdowns in database performance have been reported during intensive system usage. This could be a disadvantage when employing the system in large-scale retail settings. This issue can be resolved by optimizing the database or upgrading to a more capable server infrastructure.

7. CONCLUSION

7.1 Summary of the Study

This thesis describes the design and execution of a smart shopping basket system based on Raspberry Pi and RFID technologies. By tackling the inefficiencies of standard barcode systems and the constraints of simple shopping baskets, the study proposes a solution that enhances both consumer experience and operational efficiency in retail settings. The system automatically identifies products with RFID tags, updates pricing in real time, and presents the total amount on an LCD screen, reducing payment time and enhancing customer satisfaction.

Chapter 1 - Introduction: This chapter discusses existing retail business difficulties and how they can be handled using RFID and Raspberry Pi technology. The study's goal was to improve customer satisfaction while also ensuring operational efficiency.

Chapter 2 - Research Objective: The purpose of the research is to identify the inefficiencies of existing shopping systems and to offer innovative solutions to these problems with an RFID-based system. It is explained how RFID technology can contribute to automatic identificationof products and instant updating of prices.

Chapter 3 - Literature Review: This chapter reviewed existing technologies and previous studies, determining the study's innovative features and contributions. The usage of RFID and Raspberry Pi in the retail industry is explained, as well as how this study is different from others.

Chapter 4 - Design: The system's hardware and software components are thoroughly discussed, together with realistic limitations and conditions, cost analysis, engineering standards, and design details. This section covers technical elements such as connecting the Raspberry Pi to an RFID reader, using the LCD screen, and managing the database.

Chapter 5 - Methods: The hardware installation and software development actions are described step by step. The integration of RFID and LCD screens, database management, and smart shopping basket implementation are all discussed in detail. This section tested the system and evaluated its performance.

Chapter 6 - Results and Discussion: The functionality of the system was tested, and important achievements included RFID tag reading accuracy, total balance calculation, and user interface. However, multiple limitations, such as RFID scanning range and database efficiency are stated. It has been indicated that while the study was overall successful in meeting its goals, several areas required improvement.

7.2 Impact, Importance and Contribution

The possible effects of this study for the retail sector are influential. The RFID-based smart shopping basket technology, which replaces outdated barcode systems, represents an important improvement in the industry by speeding up in-store operations and enhancing consumer experience. Customers may simply identify products and quickly view their total cost during the purchasing process, making the shopping experience enjoyable and affordable. Additionally, this solution benefits retailers in optimizing managing their inventory and reducing operating costs. RFID technology offers reducing employee workload and automatic scanning, which improves operational efficiency by minimizing manual operations.

This thesis advances considerably to the development of a smart shopping basket system using RFID technology and Raspberry Pi. The outcomes indicate that the system can read RFID tags with high precision, display product information correctly, and calculate the total balance immediately. These contributions enable substantial progress towards driving innovation in the retail sector and enhancing customer experience. Furthermore, the system's user-friendly interface and affordable components provide a good investment for small and medium-sized retailers.

7.3 Future Work

Improving RFID Reader Performance: More powerful RFID readers or improved antenna design can be used to improve reading range and accuracy. This ensures accurate tag detection even in tough situations.

Database Performance Optimization: The database management system should be optimized to accommodate large transaction volumes. This may involve indexing, query optimization, and upgrading to a stronger database architecture.

User Interface Improvement: The user interface may be improved further to improve usability. This might involve providing multilingual help, additional product information, and features that are interactive. A bigger screen may be used, and payment can be done by QR code.

Integration with Mobile Applications: Additional functions such as remote basket management, personalized promotions and real-time notifications can be provided by developing a complementary mobile application. The payment screen can be opened with a QR code. Customers can make payment from their phone by entering their card information. Or, when the products are added to the basket, a personalized QR code containing the list of products is given to the customer by pressing a button, and time can be saved by scanning it at unmanned cash registers. It may be a good development, especially for customers with health problems.

Huskeylens: When the artificial intelligence image processing sensor is included, error-free reading can be achieved with RFID. Customer habits can be monitored for the workplace.

Security Enhancements: Ensuring data security and user privacy is critical. Future efforts should focus on implementing strong security protocols to protect customer data and prevent unauthorized access.

REFERENCES

[1] V. Chougule, A. Keste, R. Koshti, R. Magdum and M. M. Chimanna, "Automated Shopping Trolley Using Raspberry Pi B+ Model," *MAT JOURNALS, Journal of Communication Engineering and Its Innovations ,* vol. 5, no. 2, p. 5, 2019.

[2] A. K. S. Al-Hakmani and A. Sajan, "Design and Implementation of a Smart Shopping Trolley Using RFID Technology," in *Fourth Middle East College Student Research Conference*, Muscat, Sultanate of Oman, 2020.

[3] K. S. Pathangay, . D. Manukonda, . R. Mandamanedi, . S. Konakala and B. Pandhi, "IOT-Based Smart Shopping Cart Using RFID," Department of Computer Science , Vadodara, India , 2024.

[4] A. A. Algur, R. N. Gavade, S. S. Kundale, A. M. Kurkute and Y. V. Sawant, "IoT Based Smart Shopping System," *International Journal of Research in Engineering, Science and Management,* vol. 3, no. 5, pp. 461-463, 2020.

[5] L. M. XIAN, "WISE SHOPPING WITH RADIO FREQUENCY IDENTIFICATION (RFID) BASED," in *BIT (Hons) Communication and Networking* , UTAR , 2018.

[6] K. Devi, T. A. Kaarthik, K. K. Selvi, K. Nandhini and S. Priya, "Smart Shopping Trolley Using RFID," *International Journal of Innovative Research in Computer,* vol. 5, no. 3, pp. 5392-5398, 2017.

[7] Shailesh, P. S. Deb, R. Chauhan and V. Tyagi, "Smart Trolley," in *International Conference on Advance Computing and Innovative Technologies on Engineering(ICACITE)*, Gr. Noida, India, 2021.

[8] J. Francis, M. P. Tony, A. Thomas and M. M, "Cartsmart: Customer-friendly shopping for modern," Dept. of Electronics and Communication Engineering , Chengannur, Alappuzha, Kerala, India, 2021.

[9] Anonim, "Raspberry Pi 3 Model B+," Robotistan, 2024. [Online]. Available: https://www.robotistan.com/raspberry-pi-3-model-b-plus. [Accessed 5 May 2024].

[10] Robolink, "RC522 RFID NFC Kiti," RoboLink, 2024. [Online]. Available: https://www.robolinkmarket.com/rc522-rfid-nfc-kiti-1356mhz. [Accessed 5 May 2024].

[11] Amazon, "16x2 1602 LCD Display Screen Blue + IIC I2C Module Interface Adapter for Raspberry pi 2 Pack," Amazon, 2024. [Online]. Available: https://www.amazon.com/JANSANE-Arduino-Display-Interface-Raspberry/dp/B07D83DY17. [Accessed 5 May 2024].

[12] Indiamart, "13.56 MHz RFID Card," Indiamart, 2024. [Online]. Available: https://www.indiamart.com/proddetail/rfid-card-15275819148.html. [Accessed 5 May 2024].

[13] M. Toorani, "Raspberry Pi ile RFID Etiketi Okuma," Samm Blog, 21 October 2019. [Online]. Available: https://blog.samm.com/raspberry-pi-ile-rfid-etiketi-okuma/. [Accessed 5 May 2024].

[14] I. Starters, "Interfacing 16×2 LCD with Raspberry Pi," IoT Starters, 3 December 2021. [Online]. Available: https://iotstarters.com/interfacing-16x2-lcd-with-raspberry-pi/. [Accessed 5 May 2024].

APPENDICES

I2C LCD DRIVER CODE

```
# LCD Address
ADDRESS = 0x27

import smbus
from time import sleep

class i2c_device:
    def init (self, addr, port=I2CBUS):
    self.addr = addr
    self.bus = smbus.SMBus(port)

# Tek bir komut gönder
def write_cmd(self, cmd):
    self.bus.write_byte(self.addr, cmd)
    sleep(0.0001)

# Komut ve argüman gönder
def write_cmd_arg(self, cmd, data):
    self.bus.write_byte_data(self.addr, cmd, data)
    sleep(0.0001)

# Veri bloğu gönder
def write_block_data(self, cmd, data):
    self.bus.write_block_data(self.addr, cmd, data)
    sleep(0.0001)

# Tek bir bayt oku
def read(self):
    return self.bus.read_byte(self.addr)

# Veri oku
def read_data(self, cmd):
    return self.bus.read_byte_data(self.addr, cmd)

# Veri bloğu oku
def read_block_data(self, cmd):
    return self.bus.read_block_data(self.addr, cmd)

# Komutlar
LCD_CLEARDISPLAY = 0x01
LCD_RETURNHOME = 0x02
LCD_ENTRYMODESET = 0x04
LCD_DISPLAYCONTROL = 0x08
LCD_CURSORSHIFT = 0x10
LCD_FUNCTIONSET = 0x20
LCD_SETCGRAMADDR = 0x40
LCD_SETDDRAMADDR = 0x80

# Giriş modunun bayrakları
 LCD_ENTRYRIGHT = 0x00
 LCD_ENTRYLEFT = 0x02
 LCD_ENTRYSHIFTINCREMENT = 0x01
 LCD_ENTRYSHIFTDECREMENT = 0x00
```

```
# Gösterge açma/kapama kontrol bayrakları
LCD_DISPLAYON = 0x04
LCD_DISPLAYOFF = 0x00
LCD_CURSORON = 0x02
LCD_CURSOROFF = 0x00
LCD_BLINKON = 0x01
LCD_BLINKOFF = 0x00

# Gösterge/kursor kaydırma bayrakları
LCD_DISPLAYMOVE = 0x08
LCD_CURSORMOVE = 0x00
LCD_MOVERIGHT = 0x04
LCD_MOVELEFT = 0x00

# Fonksiyon set bayrakları
LCD_8BITMODE = 0x10
LCD_4BITMODE = 0x00
LCD_2LINE = 0x08
LCD_1LINE = 0x00
LCD_5x10DOTS = 0x04
LCD_5x8DOTS = 0x00

# Arka ışık kontrol bayrakları
LCD_BACKLIGHT = 0x08
LCD_NOBACKLIGHT = 0x00

En = 0b00000100 # Enable bit
Rw = 0b00000010 # Read/Write bit
Rs = 0b00000001 # Register select bit

class lcd:
    def init (self):
        # LCD cihazını başlat
        self.lcd_device = i2c_device(ADDRESS)

# LCD'yi başlat
 self.lcd_write(0x03)
 self.lcd_write(0x03)
 self.lcd_write(0x03)
 self.lcd_write(0x02)

self.lcd_write(LCD_FUNCTIONSET | LCD_2LINE | LCD_5x8DOTS |
LCD_4BITMODE)
self.lcd_write(LCD_DISPLAYCONTROL | LCD_DISPLAYON)
self.lcd_write(LCD_CLEARDISPLAY)
self.lcd_write(LCD_ENTRYMODESET | LCD_ENTRYLEFT)
sleep(0.2)

# Komutu LCD'ye yaz
    def lcd_strobe(self, data):
        self.lcd_device.write_cmd(data | En | LCD_BACKLIGHT)
        sleep(0.0005)
        self.lcd_device.write_cmd(((data & ~En) | LCD_BACKLIGHT))
        sleep(0.0001)

# LCD'ye 4 bit yaz
def lcd_write_four_bits(self, data):
    self.lcd_device.write_cmd(data | LCD_BACKLIGHT)
    self.lcd_strobe(data)
```

```
# LCD'ye komut yaz
def lcd_write(self, cmd, mode=0):
    self.lcd_write_four_bits(mode | (cmd & 0xF0))
    self.lcd_write_four_bits(mode | ((cmd << 4) & 0xF0))

# LCD'ye karakter yaz
def lcd_write_char(self, charvalue, mode=1):
    self.lcd_write_four_bits(mode | (charvalue & 0xF0))
    self.lcd_write_four_bits(mode | ((charvalue << 4) & 0xF0))

# LCD'ye metin yaz
def lcd_display_string(self, string, line=1, pos=0):
    if line == 1:
        pos_new = pos
    elif line == 2:
        pos_new = 0x40 + pos
    elif line == 3:
        pos_new = 0x14 + pos

    elif line == 4:
        pos_new = 0x54 + pos

    self.lcd_write(0x80 | pos_new)

    for char in string:
        self.lcd_write(ord(char), Rs)

# LCD'yi temizle ve başlangıç konumuna getir
def lcd_clear(self):
    self.lcd_write(LCD_CLEARDISPLAY)
    self.lcd_write(LCD_RETURNHOME)

# Arka ışığı aç/kapa
def backlight(self, state): # for state, 1 = on, 0 = off
    if state == 1:
        self.lcd_device.write_cmd(LCD_BACKLIGHT)
    elif state == 0:
        self.lcd_device.write_cmd(LCD_NOBACKLIGHT)

# Özel karakterler ekle (0-7)
def lcd_load_custom_chars(self, fontdata):
    self.lcd_write(0x40)
    for char in fontdata:
        for line in char:
            self.lcd_write_char(line)
```

RFID with LCD SETUP CODE

```
#!/usr/bin/env  python3
# -*- coding: utf-8 -*-

#import RPi.GPIO as GPIO
import MFRC522
import signal
import I2C_LCD_driver
from time import *

continue_reading = True

mylcd = I2C_LCD_driver.lcd()

# Function to read uid and convert it to a string
def uidToString(uid):
    mystring = ""

    for i in uid:
        mystring = format(i, '02X') + mystring
    return mystring

# Capture SIGINT for cleanup when the script is aborted
def end_read(signal, frame):
    global continue_reading

    print("Ctrl+C captured, ending read.")
    continue_reading        =        False
    GPIO.cleanup()

#          Hook          the          SIGINT
signal.signal(signal.SIGINT, end_read)

# Create an object of the class MFRC522
MIFAREReader = MFRC522.MFRC522()

# Welcome message
```

```
print("Welcome to the MFRC522 data read example")
print("Press Ctrl-C to stop.")

# This loop keeps checking for chips. If one is near it will get the UID
and authenticate

while continue_reading:
    #  Scan  for  cards
    (status, TagType) =

MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)

    # If a card is found
    if status == MIFAREReader.MI_OK:
        print("Card detected")

        # Get the UID of the card

        (status, uid) = MIFAREReader.MFRC522_SelectTagSN()


        # If we have the UID, continue
        if status == MIFAREReader.MI_OK:
            print("Card read UID: %s" % uidToString(uid))
            mylcd.lcd_display_string(uidToString(uid), 1)

        else:

            print("Authentication              error")
            mylcd.lcd_display_string("Error!")
```

Smart Shopping Basket Implementation Code

```
#!/usr/bin/env python3 # -*- coding: utf-8 -*-

#import RPi.GPIO as GPIO
import MFRC522
import signal
import I2C_LCD_driver from time import * import pymysql

continue_reading = True
mylcd = I2C_LCD_driver.lcd()

Host = "localhost" User = "admin" Password = "Leonel@01" database = "urun"

conn = pymysql.connect(host=Host, user=User, password=Password, db=database)
cur = conn.cursor() top = 0
# function to read uid and convert it to a string
def uidToString(uid): mystring = ""

for i in uid:
mystring = format(i, '02X') + mystring
return mystring

# Capture SIGINT for cleanup when the script is aborted
def end_read(signal, frame): global continue_reading
print("Ctrl+C captured, ending read.") continue_reading = False GPIO.cleanup()

# Hook the SIGINT signal.signal(signal.SIGINT, end_read)

# Create an object of the class MFRC522 MIFAREReader = MFRC522.MFRC522()

# Welcome message mylcd.lcd_display_string("Akilli sepeti", 1)
mylcd.lcd_display_string("hosgeldiniz", 2) print("Akilli sepeti hosgeldiniz!")
time.sleep(2)
mylcd.lcd_clear() mylcd.lcd_display_string("Sepet'e", 1)
mylcd.lcd_display_string("Urun ekleyin", 2) print("Sepete urun ekleyin!")

# main code
while continue_reading: # Scan for cards (status, TagType) =
MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)

# If a card is found
if status == MIFAREReader.MI_OK: # Get the UID of the card
(status, uid) = MIFAREReader.MFRC522_SelectTagSN()


= %s)"

# If we have the UID, continue
if status == MIFAREReader.MI_OK: #print("Card detected")
#print("Card read UID: %s" % uidToString(uid)) kart_id = uidToString(uid)
#print(kart id)#for debug id card
query = f"SELECT urun_adi,fiyat FROM urun_table WHERE (urun_id

result = cur.execute(query,(kart_id,)) conn.commit()
if result == 1:#urun veri tabanina olup olmadigini kontrol
```

```
ediyor ve eger urun varsa
data = cur.fetchone() urun_adi, f = data fiyat = int(f)
if fiyat == 0:#onaylama karti veri tabani fiyati 0 olarak tanimlanmis ve sistem
onu sonlandiriyor
mylcd.lcd_clear() mylcd.lcd_display_string("odeniyor...", 1) print("odeniyor
...")
time.sleep(2) mylcd.lcd_clear()
mylcd.lcd_display_string("sepet bos", 1)

mylcd.lcd_display_string("Urun ekleyin", 2) print("Sepet bos Urun ekleyin")
delete = f"DELETE FROM sepet" delt = cur.execute(delete) conn.commit()
top = 0#toplam sifirlaniyor
else:#alisveris devam ediyor
query1 = f"SELECT urun_id FROM sepet WHERE (urun_id =
%s)"#check if object(urun) is on basket
result1 = cur.execute(query1,(kart_id,))
if result1 == 1:#urun sepete olup olmadigini kontrol

ediyor ve eger varsa cikartiliyor

ediyor ve eger yoksa ('{kart_id}')"


delete = f"DELETE FROM sepet WHERE (urun_id = %s)" delt =
cur.execute(delete,(kart_id,)) conn.commit()
mylcd.lcd_clear() mylcd.lcd_display_string(urun_adi, 1)
mylcd.lcd_display_string("cikariliyor...", 2) print(urun_adi + "
cikariliyor...") time.sleep(1)
top = top - fiyat#sepeten urun fiyati toplamdan
elif result1 == 0:#urun sepete olup olmadigini kontrol add = f"INSERT INTO sepet
(urun_id) VALUES cur.execute(add)
conn.commit() mylcd.lcd_clear()
mylcd.lcd_display_string(urun_adi + " : " +

str(fiyat) + " TL", 1)
eklendi") ekleniyor


mylcd.lcd_display_string("eklendi", 2) print(urun_adi + " : " + str(fiyat) + "
TL

time.sleep(1)
top = top + fiyat#sepete urun fiyati toplama



mylcd.lcd_clear() mylcd.lcd_display_string("sepet toplam :", 1)
mylcd.lcd_display_string(str(top), 2) print("sepet toplam :" + str(top))
else:#urun veri tabanina olup olmadigini kontrol ediyor ve eger

mylcd.lcd_clear() mylcd.lcd_display_string("bu urun yok", 1) print("Bu urun
vridabanina yok...") time.sleep(1)
mylcd.lcd_clear() mylcd.lcd_display_string("sepet toplam :", 1)
mylcd.lcd_display_string(str(top), 2) print("sepet toplam :" + str(top))

else:
print("Authentication error") mylcd.lcd_display_string("Error!")
```