An-Najah National University

Faculty of Engineering and Information Technology

Department of Computer Engineering

# Graduation Project II - 10636582/7

## Multifunctional Beverage Preparation and Dispensing System
## iDrink Station

Supervisors
1. Dr. Hanal Abu-Zant

Team members
1. Ahmad Ghassan Atallah
2. Khaled Mohammad Saadeh

Presented in partial fulfillment of the requirements for Bachelor degree in Computer Engineering.

June 18, 2025

# Dedication

This project is dedicated to our parents and family, who taught us to trust in Allah. In addition to their kindness also for supporting and encouraging us to believe in ourselves. Without them, we would not be here now.

# Acknowledgment

We grabbed this opportunity to express our gratitude and thanks to all who contributed to helping us complete this project as it should have been done.

We would like to express our very great appreciation and sincere gratitude to our supervisor Dr. Hanal Abu-Zant, for his extraordinary support, and valuable suggestions during the planning and development of this project, and for his patience and motivation. he is helping us and giving us advice, guidance, and counseling. This project would have been impossible without the support of the head of the computer engineering department Dr. Manar Qamheyeh and Dr. Abdullah Rashed. For this, we are very grateful to them.

We also want to thank friends and colleagues for their continued support throughout the study period. We are also indebted to all teaching staff and assistants for their great help and give us all the information that we need, and for the continuous support they have given us throughout our time in graduate university.

# Disclaimer

Students at the Computer Engineering Department, Faculty of Engineering and Information Technology, An-Najah National University wrote this report. It has not been altered or corrected, other than editorial corrections, because of assessment and it may contain language as well as content errors. The views expressed in it together with any outcomes and recommendations are solely those of the students. An-Najah National University accepts no responsibility or liability for the consequences of this report being used for a purpose other than the purpose for which it was commissioned.

# Table of Contents (TOC)

# List of Figures (LOF)

# Abstract


## Abstract

This project aims to provide a practical and sustainable solution in the world of vending beverages by designing and building a machine that prepares cold and hot drinks using simple electronic and mechanical components. It focuses on facilitating and accelerating the automatic beverage preparation process, providing a smooth and fast user experience, making it ideal for offices, cafes, educational centers, public spaces, and even kiosks and small businesses.

The machine operates using an Arduino controller, which precisely manages its various components, regulating the operation of sensors and controlling the movement of pumps and motors. It also includes an ESP module to enable communication with a mobile application. The beverage type is selected either manually via an electronic keypad or electronically through the app, offering flexibility in control. Additionally, the system can be activated using an RFID card or directly from the mobile application.

Operating data, such as the beverage type (cold or hot), the selected flavor, temperature, and the number of cups prepared, is displayed on an LCD screen, providing the user with a comprehensive and clear experience.

The mechanism involves pumping water from the tank, heating it as needed, selecting the appropriate beverage type, then pumping the desired concentrate or powder and mixing it into the cup, serving it to the user within a few minutes. These sequential steps allow the user to prepare a variety of drinks, such as mojitos, Vimto, iced tea, fruit juices, Zahra juice, coffee, Nescafe, or tea, with just the push of a button.

This project represents a practical model for the concept of digital transformation in quick-service restaurants, combining simplicity in design with performance efficiency. The machine not only saves time and effort for users and workers, but also opens the way for the development of innovative small businesses in the beverage sector, especially with the approach of summer and the increased demand for cold and ready-to-drink beverages.

# Chapter One

# Introductory

# Chapter One: Introductory

**Chapter Contents:**

# Chapter One: Introductory

## 1.1: Introduction.

The increasing need for fast, hygienic, and autonomous beverage service has led to the evolution of smart vending systems. These systems offer users a seamless way to access their preferred drinks without human interaction, making them ideal for workplaces, campuses, and public spaces. This project, titled iDrink Station, presents a cost-effective and modular smart machine designed to prepare and dispense both hot and cold beverages using embedded systems and automation.

The machine is based on a microcontroller-driven design, utilizing an Arduino Mega to control all mechanical actions and an ESP32 module for wireless communication. Users interact with the system either manually through a 4x4 keypad or digitally via a mobile app. Additionally, the machine can be activated using an RFID card, providing flexibility in access. The LCD screen displays all operational details, such as drink selection, temperature, and number of cups prepared.

The internal layout is organized into two racks. The first rack contains six containers: three for concentrated juice flavors (lemon, blueberry, and orange), one for cold water, one for hot water, and one for milk. Each liquid is connected to an individual pump and flow sensor, enabling precise measurement and delivery. The second rack includes two mixers—one for cold drinks and another for hot drinks—alongside a water heater for preparing coffee, Nescafe, or tea. Dedicated cleaning pumps are also installed in both racks to maintain hygiene after each cycle.

Upon user selection, the system initiates a sequence: pumps draw the required ingredients, flow sensors monitor the volume, mixers blend the contents, and the result is dispensed into a cup. The process is fast, efficient, and customizable. Available drinks include a variety of hot and cold options, all accessible with a single button press or app tap.

The iDrink Station demonstrates an applied integration of IoT, automation, and mechatronics, offering a smart, affordable, and scalable solution for beverage service. Its modular design makes it easy to maintain or expand, while its dual-interface control (manual and app) ensures versatility. This system serves as a real-world application of embedded technologies and aligns with the goals of digital transformation in modern services.

Most importantly, the system provides consistency in drink quality, reduces service time, and minimizes human error or contamination risks. With minimal maintenance needs and intuitive operation, it can be deployed in various environments where quick, reliable beverage access is essential.

# Chapter One: Introductory

## 1.2: Problems and Work Limitations.

As university students still in the learning phase, we faced several challenges during the development of the iDrink Station. One of the main limitations was lack of experience—especially in dealing with hardware connections, integrating physical components, and writing code that brings everything together. Each part of the project required research, testing, and trial-and-error to get it right.

Another big issue was the limited budget. We had to work with basic components and avoid expensive modules, which forced us to make creative decisions and sometimes change plans based on what we could afford. Also, due to the large number of components involved, the system became complex and needed extra effort to manage wiring, mounting, and proper layout in a small space.

We also experienced technical problems like tubing leaks and dripping after the pump stops, most likely caused by osmotic pressure. Fixing these required physical adjustments and sometimes extra valves or clamps. Managing synchronization between many parts, like pumps, flow sensors, mixers, and the heater, was tricky and required clean logic in the code.

Time was a major constraint, as we were working under a tight academic schedule. Balancing the project with other university tasks limited our ability to test the system in more real-life scenarios or polish it further. Despite all of this, we were driven by our passion to build something innovative. Even with limited tools, parts, and skills, we challenged ourselves to design, build, and control a working prototype that combines both hardware and software in one automated system.

# Chapter One: Introductory

## 1.3: Objectives and work significance.

The main goal of the iDrink Station project was to design and build a fully functional machine that can prepare and serve both hot and cold drinks automatically. We wanted to create a system that is simple, affordable, and practical for real-life use. From the beginning, the idea was to make the process of getting a drink faster, cleaner, and more consistent, without the need for a human operator. The project also gave us a chance to apply what we've learned in university and push our skills in electronics, programming, and system integration.

One of the key objectives was to make the machine work with both manual and digital input, allowing the user to control it using either a keypad or a mobile app. We also wanted to include RFID support, so that the machine can be activated securely or used for access control in places like offices or schools. Giving users more than one way to interact with the system makes it flexible and useful in many different environments.

Another important objective was to make the machine handle multiple ingredients, including three types of concentrated juice, cold water, hot water, and milk. Simplest vending machines can only serve one or two options. We wanted to go further and allow the system to prepare different drinks using separate tanks, pumps, and mixers. This required building a setup that is both organized and modular, so we could manage everything efficiently and add more options in the future if needed.

We also focused on automating the entire drink preparation process, from selecting the drink to mixing and serving it in a cup. To do this, we had to program the Arduino to control all pumps, read flow sensors, activate mixers, and manage timing carefully. On the other hand, the ESP32 was used to handle communication with the mobile app, making it possible to send commands wirelessly and check the machine's status. Adding an LCD screen allowed us to give real-time feedback to the user, such as the selected drink, the number of cups, or if the machine is cleaning.

The significance of this project goes beyond the technical achievement. It gave us real experience in working as a team, managing time, solving unexpected problems, and dealing with limitations like budget or hardware availability. More importantly, the project solves real issues - saving time, ensuring hygiene, and offering convenience for users in places like schools, offices, or even small businesses. It also shows that with a bit of creativity and effort, smart solutions can be built using simple, low-cost tools.

# Chapter One: Introductory

## 1.4: Scope of the work.

The iDrink Station project provides a range of features that cover the full process of preparing and serving both hot and cold drinks automatically. The system includes hardware, control logic, and a mobile interface to deliver a complete and flexible experience. The scope of the work includes the following main functions:

1. **Cold and Hot Drink Support:**
   The system prepares both cold drinks (like juices and iced tea) and hot drinks (like coffee and tea) using separate mixing units and a heating element for precise temperature control.

2. **Multi-Liquid Ingredient Control:**
   It supports 6 input sources: 3 juice concentrates, cold water, hot water, and milk. Each is delivered using a dedicated pump and measured with flow sensors to ensure consistent portioning.

3. **Full Automation:**
   Once a drink is selected, the system autonomously controls all required components—activating pumps, mixers, heaters, and timing mechanisms—to deliver the final product into a cup without user intervention.

4. **Dual Control Interface:**
   Users can operate the system either through a physical keypad or via a mobile application connected through ESP32. This makes the system accessible from close range and remotely.

5. **RFID Card Activation:**
   The system can optionally be started using an RFID card, useful for controlled environments such as schools, labs, or employee-only areas.

6. **Live System Feedback via LCD:**
   A 20x4 LCD provides real-time feedback to the user, including selected drink, system status (idle, filling, cleaning), and number of cups served.

7. **Built-in Cleaning Routine:**
   After each use, or on demand, the system can run a cleaning cycle that flushes the mixers and pipes to reduce contamination between drinks.

8. **Mobile Application Services:**
   This integration adds modern usability and allows the system to fit into digital environments. The companion mobile app allows users to:
   - ✓ Select drink type and quantity remotely.
   - ✓ Start and stop the system.
   - ✓ Monitor temperature, status, and usage.
   - ✓ View live updates during operation.

9. **Organized Modular Structure:**
   The machine is physically divided into two racks: one for ingredients and delivery, and another for processing (mixing and heating). This structure improves maintenance and upgrades.

Beyond the technical components, the iDrink Station is designed to offer a complete beverage service with minimal user effort. It automates the entire process from input to output, while giving users flexibility in how they interact with the machine - manually or digitally. The project not only solves practical problems like time-saving and hygiene, but also demonstrates how basic, low-cost components can be combined to build a smart, scalable, and user-friendly product. This combination of hardware, software, and real functionality defines the true scope and impact of the system.

# Chapter One: Introductory

## 1.5: Report Organization.

This report is structured to clearly explain the development, features, and implementation of the iDrink Station project—an automated smart beverage dispensing system. It follows a logical format inspired by engineering project documentation, covering all stages from problem identification to final testing and recommendations.

At the start of the project, we worked as a team to define the idea and understand the needs we aim to address. Regular collaboration was maintained through meetings and shared documents, allowing us to distribute tasks and track progress. Platforms like Google Sheets and Trello were used to manage time and responsibilities efficiently. The work began with technical research and component identification, followed by system design and implementation. We then moved to software integration, testing, and documentation.

The report is organized into several key sections:

1. **Introduction:**
   Describes the idea behind the project, its goals, and how it fits into real-life scenarios like offices and smart kiosks.

2. **Problems and Work Limitations:**
   Discusses challenges we faced during development such as limited experience, budget constraints, hardware integration, and time pressure.

3. **Objectives and Work Significance:**
   Outlines the purpose of the system and the real-world problems it solves—automation, hygiene, and user convenience.

4. **Scope of the Work:**
   Highlights the key features, such as hot and cold drink preparation, dual control interface (keypad + app), RFID access, auto-cleaning, and modular design.

5. **Constraints and Standards:**
   Notes development limitations and engineering standards followed during implementation.

6. **Literature Review:**
   Presents related systems and prior research, and how our solution differs in scale, function, and cost.

7. **Methodology:**
   Details the process of building the system, including component selection, wiring, programming, system logic, and mobile app integration.

8. **Results and Discussion:**
   Evaluates system performance based on functionality, stability, and user experience.

9. **Conclusion and Recommendations:**
   Summarizes what was achieved and suggests areas for future improvement, such as adding a payment module or touchscreen UI.

This organization ensures the reader can follow each development stage, understand the technical background, and see the significance of the iDrink Station in solving real-world problems.

| # | Task | Start At | End At | Duration (days) |
|---|---|---|---|---|
| 1 | Idea and Goal Definition | 20-Feb-25 | 28-Feb-25 | 9 |
| 2 | Component Research & Selection | 28-Feb-25 | 10-Mar-25 | 11 |
| 3 | Hardware Collection & Assembly | 10-Mar-25 | 25-Mar-25 | 15 |
| 4 | System Design & Wiring | 25-Mar-25 | 5-Apr-25 | 11 |
| 5 | Arduino + ESP32 Programming | 5-Apr-25 | 25-Apr-25 | 20 |
| 6 | Mobile App Design & Integration | 25-Apr-25 | 10-May-25 | 15 |
| 7 | System Testing and Troubleshooting | 10-May-25 | 25-May-25 | 15 |
| 8 | Adjustments and Feature Tuning | 25-May-25 | 2-Jun-25 | 8 |
| 9 | Final Report Preparation | 2-Jun-25 | 10-Jun-25 | 8 |

*Figure 1: Report Organization and Project Management.*

# Chapter Two

# Constraints and Standards

# Chapter Two: Constraints and Standards

**Chapter Contents:**

**2.1: Constraints and work limitations.**

**2.2: Standards / Codes.**

**2.3: Earlier coursework.**

# Chapter Two: Constraints and Standards

## 2.1: Constraints and work limitations.

Throughout the development of the iDrink Station project, we faced a number of technical, practical, and logistical challenges that shaped the direction and complexity of the work. These limitations are summarized as follows:

❖ **Technical Constraints:**

1. **Hardware Integration:**
   Due to our limited experience, connecting various hardware components such as pumps, flow sensors, drivers, Arduino Mega, and ESP32 required significant time to understand wiring, logic, and power management.

2. **Tubing and Liquid Flow Issues:**
   One major problem was the dripping of liquids after dispensing, caused by residual pressure in the tubes. Without access to high-end solenoid valves or flow regulators, we could not fully resolve this issue.

3. **Limited Hardware Capabilities:**
   Using basic components introduced memory and performance limitations. The Arduino could not handle multi-threaded tasks, which forced us to simplify some logic.

❖ **Time and Resource Constraints:**

1. **Tight Project Timeline:**
   Starting from February 20th and finishing by June 10th, we had limited time to plan, learn, design, build, test, and document the project. This restricted the addition of advanced features and deep testing.

2. **Budget Limitations:**
   As university students, we faced financial constraints. All components were purchased using personal funds, which affected our ability to acquire high-quality or spare parts. We often had to reuse materials and accept lower precision.

3. **Component Availability:**
   Some parts were hard to find in the local market, requiring design changes or substitutions. This added delays and required flexibility in the mechanical layout.

❖ **Development and Team Constraints**

1. **Small Team Size:**
   With limited team members, we had to multitask and divide responsibilities between wiring, coding, testing, and documentation. This required prioritization and limited our ability to polish features.

2. **Learning Curve:**
   Much of the initial time was spent learning how to read datasheets, wire components, debug connections, and structure the control code properly. This slowed overall progress, especially in the first phase.

3. **Code Synchronization:**
   Synchronizing ESP32 with Arduino, handling flow sensor readings, and managing real-time logic (filling, heating, cleaning) was challenging. Debugging errors across different microcontrollers required careful planning.

❖ **User-related Limitations:**

1. **No User Customization:**
   The system doesn't allow users to fully customize their drink (e.g., sugar level, size). Adding such options would require a more advanced control system and interface.

2. **Simple Display and Interaction:**
   The use of a basic LCD and keypad limited how much information we could show. Complex menus or interactive feedback were out of scope for this version.

Despite all these constraints, we successfully implemented a working system that delivers a fully functional beverage preparation process. These challenges, while demanding, helped us gain critical hands-on experience and a deeper understanding of embedded systems, automation, and real-world design limitations.

# Chapter Two: Constraints and Standards

## 2.2: Standards / Codes.

While the project was not subject to industrial certification, we followed certain academic and practical standards to ensure system stability, safety, and expandability:

1. **IEEE Standards for Microcontroller Communication:**
   Applied particularly in wireless control and data exchange between ESP32, Arduino, and sensors using standard protocols like UART and I2C.

2. **Arduino Community Guidelines:**
   Structured coding practices, consistent pin assignments, modular sketches, and meaningful variable names for code readability.

3. **ESP32 Best Practices:**
   Proper use of Wi-Fi libraries, memory management, and non-blocking code to ensure stability during real-time communication.

4. **Modular System Design:**
   Functional separation between cold drink unit, hot drink unit, and cleaning system — allowing easy maintenance and upgrades.

5. **User Interaction Standards:**
   The LCD screen provides intuitive feedback about selected drink, drink type, and system status. Application interaction is structured based on mobile UI design guidelines (clean layout, button-based triggers, and feedback after each action).

6. **Data Integrity and Safety:**
   Although no formal encryption is implemented, care was taken to avoid command collisions and ensure that simultaneous app and keypad inputs do not disrupt the system.

# Chapter Two: Constraints and Standards

## 2.3: Earlier coursework

Several university courses provided essential knowledge and hands-on experience that directly contributed to the development of the iDrink Station. These courses helped us understand how to organize the system, write efficient code, manage hardware, and think logically about real-world problems.

The combination of programming, hardware interfacing, and system design courses gave us a strong foundation in embedded systems and automation. In particular, learning how to break down a complex idea into manageable components and follow a structured development process was a major advantage throughout this project.

The project benefited from knowledge and skills developed in earlier university courses:

1. **Embedded Systems:**
   Provided the foundation for interfacing Arduino and ESP32 with sensors, relays, and actuators.

2. **Microprocessors and Interfacing:**
   Helped us understand serial communication, pin configurations, and timing control needed for managing multiple hardware components.

3. **Programming with C/C++:**
   Allowed us to write clear, functional Arduino code with loops, conditionals, and modular structure.

4. **Computer Networks and IoT:**
   Familiarized us with Wi-Fi-based control systems and helped us configure ESP32 as a web client/server.

5. **Human-Computer Interaction (HCI):**
   Inspired us to prioritize user-friendly design for both the LCD interface and mobile app interactions.

# Chapter Three

# Literature Review

# Chapter Three: Literature Review

**Chapter Contents:**

# Chapter Three: Literature Review

## 3.1: Introduction.

In recent years, automation and smart vending machines have become increasingly common across different sectors, including food service, hospitality, and retail. These machines are designed to improve efficiency, minimize human intervention, and deliver consistent service to users. Beverage vending machines, in particular, have evolved from simple can dispensers into complex systems capable of preparing a variety of hot and cold drinks on demand.

The integration of microcontrollers such as Arduino and ESP32, along with IoT technologies, has enabled the development of smarter, more interactive systems. These systems can manage multiple inputs, control fluid delivery mechanisms, and respond to user preferences through digital interfaces. By studying existing vending systems and automated dispensers, we identified key gaps and opportunities to innovate — especially in combining low-cost hardware with smart functionality.

This literature review focuses on understanding the evolution of beverage machines, analyzing similar systems, and highlighting the unique aspects of our solution in comparison.

# Chapter Three: Literature Review

## 3.2: Project History

The concept of automated drink machines is not new. Traditional vending machines have been available for decades, offering canned or bottled drinks through coin-based systems. Over time, newer machines began offering programmable drink options like coffee, tea, and soft drinks, especially in office settings and public spaces.

In recent years, companies have developed fully automatic drink systems integrated with cashless payment options, touchscreens, and even cloud-connected inventory management. However, most of these solutions are commercially expensive and built with proprietary hardware.

Some open-source projects and academic prototypes have attempted to create DIY coffee or juice machines using Arduino or Raspberry Pi. These systems often focus on limited functionality - such as dispensing only coffee or a single type of juice - and typically lack wireless features or modular design.

Our project, iDrink Station, builds on this foundation by combining both hot and cold drink capabilities into one system, offering real-time control via keypad or mobile app, and supporting modular customization using widely available hardware. This approach creates a flexible, scalable, and student-accessible model that bridges the gap between academic prototypes and real-world smart machines.

# Chapter Four

# Methodology

# Chapter Four: Methodology

The methodology chapter outlines the structured approach used in developing the iDrink Station system. The project was built using a combination of hardware prototyping, embedded systems programming, and mobile application development. It follows a step-by-step process, starting from identifying user needs and system requirements, to designing, implementing, and testing each component. By combining Arduino, ESP32, various sensors, and a custom mobile app, the methodology ensured practical integration between electronics and user interaction, resulting in a functional and efficient automated drink dispenser.

## Chapter Contents:

# Chapter Four: Methodology

## 4.1: Development Tools and Languages

The development of iDrink Station required both embedded system programming and web development tools to build a fully functional automated beverage dispenser. The system relies on hardware components controlled by an Arduino Mega and ESP32, while the web-based interface was created using modern front-end and back-end technologies.

1. **Visual Studio Code**
   VS Code was the main code editor used for writing both the embedded firmware and the web application. Its extension support, syntax highlighting, and Git integration made it ideal for managing multiple parts of the system.

2. **Arduino IDE**
   The Arduino IDE was used to develop and upload the firmware for the Arduino Mega. Code was written in C/C++ to handle interactions with flow sensors, pumps, LCD display, keypad, and RFID reader.

3. **ESP32 + Arduino Core**
   The ESP32 was programmed using Arduino-style C++ libraries. It handled Wi-Fi connectivity, received HTTP requests from the web app, and communicated with the Arduino via serial or digital signaling.

4. **React (Front-End Web Application)**
   The control interface was developed using React. This allowed us to build a responsive, component-based single-page application (SPA) where users can control the system remotely, select drink types, start cleaning, and monitor the number of cups served.

5. **Node.js (Back-End Server)**
   Node.js was used to manage the server-side logic and route the communication between the user interface and the ESP32 module. This ensured smooth API requests and structured command delivery to the machine.

6. **Proteus (for Circuit Design)**
   These tools were used for drawing and visualizing the system's circuit diagrams, planning the connections between components before physically wiring the system.

**Programming Languages Used:**

1. **C/C++**
   Used for programming the Arduino Mega and ESP32 to handle sensor data, motor control, and communication logic.

2. **JavaScript / JSX**
   Used in React for building the front-end interface. JSX allowed for modular UI components and interactive controls.

3. **Node.js (JavaScript)**
   Used in the back-end to manage server logic, API endpoints, and real-time interactions between the user and the system.

4. **HTML/CSS**
   Used for structuring and styling the web application, ensuring responsiveness and a user-friendly experience.

This blend of embedded programming and modern web development allowed us to create an efficient, connected, and interactive vending system that bridges the gap between physical hardware and remote control.

# Chapter Four: Methodology

## 4.2: Project Hardware Components

The iDrink Station system integrates multiple hardware components to deliver an automated, smart beverage vending experience. The machine was built using off-the-shelf electronic modules, sensors, and actuators arranged across two levels to separate input and mixing functions. The following components were used:

*Figure 2: Complete Hardware Layout of iDrink Station Prototype*

1. **External Casing (Wooden Frame)**
   The machine's components were housed in a custom-made wooden frame that provides structural support and protection. The frame separates different hardware layers (liquid storage and electronics), and helps organize wiring.

*Figure 3: External Casing (Wooden Frame)*

2. **Arduino Mega 2560**

   Acts as the main controller for handling local operations. It manages pumps, mixers, flow sensors, the LCD display, keypad, and RFID reader. Its large number of I/O pins made it suitable for this multi-component system.


*Figure 4: Arduino Mega 2560*

3. **ESP32 Module**

   Responsible for handling Wi-Fi communication. It connects to a Node.js-powered backend and receives commands from the mobile or web application. The ESP32 forwards these commands to the Arduino Mega via serial communication.

*Figure 5: ESP32 Module*

4. **HC-05 Bluetooth Module**

   Used to establish wireless communication with external mobile applications (alternative to ESP32-based control). It enables local Bluetooth-based control of the machine without requiring Wi-Fi connectivity. This is especially useful for offline access or debugging during development.


*Figure 6: HC-05 Bluetooth Module*

5. **RFID Reader (MFRC522)**
   Used for two functions: powering the system ON/OFF with an authorized tag and confirming payments using a separate card. This adds a layer of access control and simulates a cashless payment system.

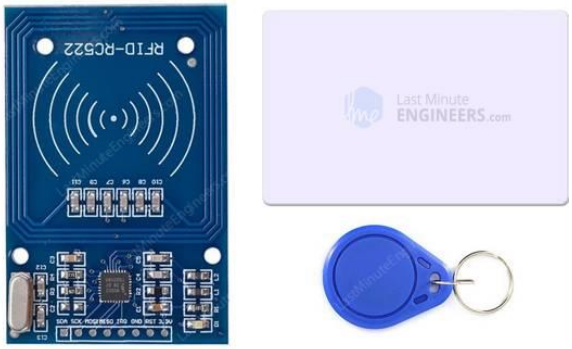


*Figure 7: RFID Reader (MFRC522)*

6. **4x4 Keypad**
   Allows users to manually control the machine without using the web application. The keypad is used to select the drink type and number of cups.

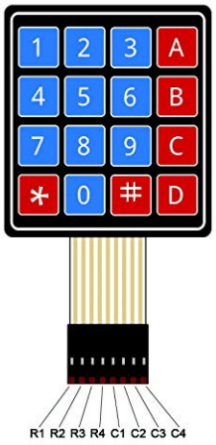


*Figure 8: 4x4 Keypad*

7. **I2C LCD Display (20x4)**
   Displays system information including startup messages, drink selection, filling status, and payment prompts. It improves the overall user experience by providing real-time visual feedback.


*Figure 9: I2C LCD Display (20x4)*

8. **Power Supply & Drivers**

All motors and modules are powered through two regulated sources (12V, 1A) and controlled using 6 driver modules (L298N), providing enough current and voltage stability.

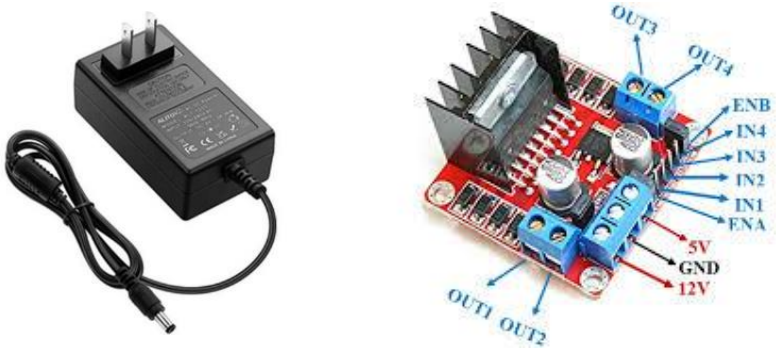


*Figure 10: Power Supply & Drivers*

9. **L298N Driver Module**

Used to control the direction and speed of DC motors and pumps. It allows the Arduino to manage multiple components like mixers and liquid pumps efficiently, enabling bidirectional control with stable power output.

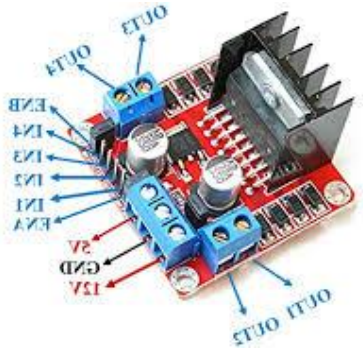

*Figure 11: L298N Driver Module*

10. **Voltage regulators and resistors**

4 Channel I2C Logic Level ShifterConverter Bi-Directional Module 5V To 3.3V. Voltage regulators were required to ensure stable 5V and 3.3V power supply for different modules like the ESP32, Arduino, RFID, and sensors. Resistors were also used in series or pull-up configurations where needed for logic level matching and sensor stability.

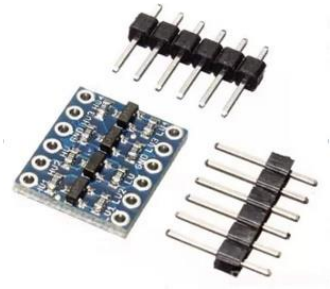

*Figure 12: ShifterConverter Bi-Directional Module 5V To 3.3V*

**11. Breadboards and jumper wires for prototyping**

Used in early prototyping to connect components without soldering. These provided a flexible platform to test and debug individual modules such as flow sensors, keypad, LCD, and pumps before final assembly. Three types of jumper wires are used for prototyping: Male to Male, Male to Female, and Female to Female

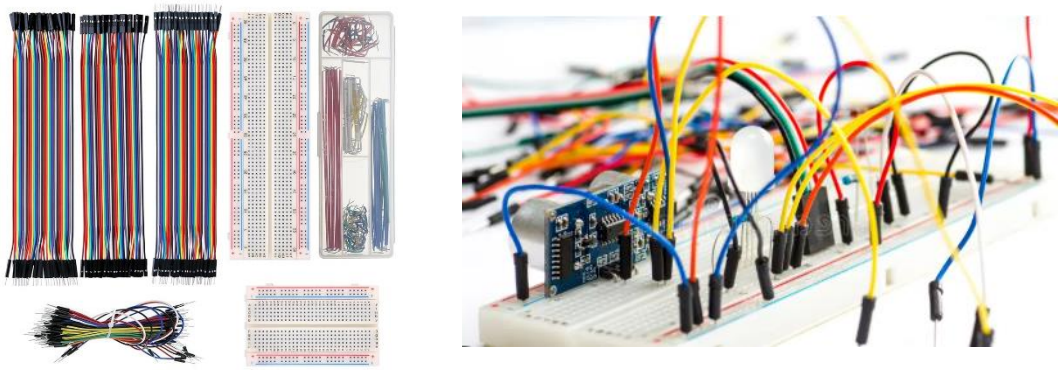

*Figure 13: Breadboards and jumper wires for prototyping*

**12. Screw Terminal Block / Power Distribution Block**

A simple connector used to group all power and ground wires from the sensors and actuators. It reduces wiring clutter and connects multiple components to the Arduino using only two main lines (5V and GND).

*Figure 14: Screw Terminal Block / Power Distribution Block*

**13. Buzzer**

A passive buzzer is integrated to provide audible alerts for different events such as system startup, payment confirmation, and error states. This helps enhance the user experience and immediate feedback without needing to look at the screen.

*Figure 15: Screw Terminal Block / Power Distribution Block*

**14. LED Indicators**

Multiple LEDs are used to indicate system states such as power on/off, drink processing, or cleaning mode. They provide a visual cue for users or developers monitoring the machine from a distance.

*Figure 16: Screw Terminal Block / Power Distribution Block*

**15. Heater Unit (Hot Water Line)**

A small heating element connected to the hot water pump to provide warm drinks like coffee or milk-based beverages.

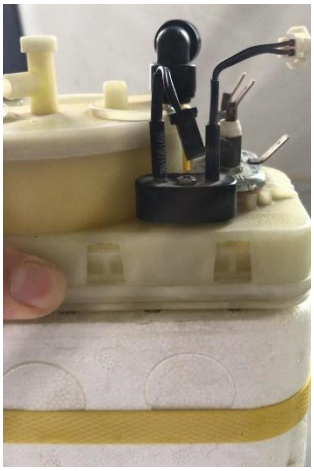

*Figure 17: Heater Unit (Hot Water Line)*

**16. Bimetallic Thermostat (Mechanical Type)**

A small mechanical device used to control the heater's temperature. It automatically cuts off power when the water reaches a specific temperature (e.g., 70°C), preventing overheating. It operates without code or sensors, making it ideal for basic heating control in the hot water line.

*Figure 18: Bimetallic Thermostat (Mechanical Type)*

**17. Relay modules (optional, for heater/safety cutoffs)**

Considered in cases where high-power components such as the heater needed to be safely switched or isolated. Relays provide an extra safety layer to cut off power in fault or overheat conditions.

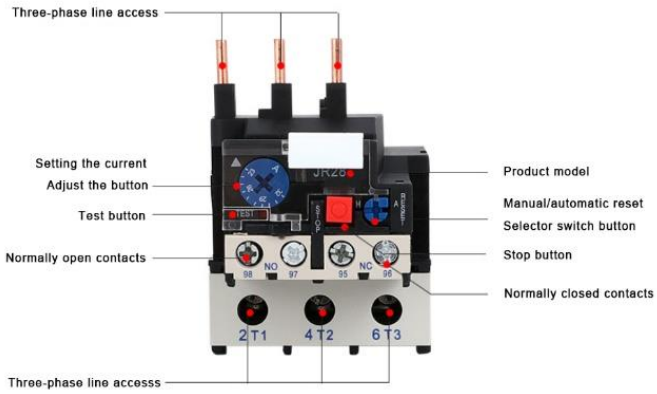


*Figure 19: Relay modules*

**18. Flow Sensors (5 Units)**

Installed to monitor the volume of liquid being dispensed. These sensors send pulses to the Arduino, which are counted to calculate flow rate and ensure consistent output per cup.

*Figure 20: Flow Sensors*

**19. Pumps and Motors**

Each pump and motor are connected via H-bridge motor drivers, enabling directional control and proper timing for sequential operations. The machine includes:

- 3 Juice Pumps (for lemon, blueberry, orange concentrates)
- 1 Cold Water Pump
- 1 Milk Pump
- 1 Hot Water Pump
- 2 Final Pumps (one for cold drinks, and another one for hot)
- 2 Mixers (DC motors, for blending ingredients before dispensing)

*Figure 21: Pumps and Motors*

**20. Tubing for liquid flow and connectors**

Flexible food-grade tubes with consideration for pressure and leakage, were used to route liquids (juice, water, and milk) between tanks, pumps, mixers, and cups. Proper pressure balancing and tight fittings were critical to avoid leaks and backflow.

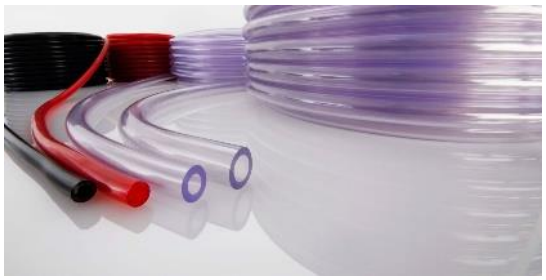

*Figure 22: Tubing for liquid flow*

**21. Plastic Check Valves / One-Way Valve Gas Air Liquid Water PVC Connectors**

Installed on each tube line to prevent reverse flow and dripping after dispensing. This ensures liquid only flows in the intended direction, maintaining hygiene and reducing waste due to backpressure issues.

*Figure 23: Check Valve Gas Air Liquid Water PVC Connectors*

Each hardware unit was carefully tested for compatibility, timing, and current load. The dual-controller architecture (Arduino + ESP32) ensured separation between hardware control and network logic, improving both safety and performance.

# Chapter Four: Methodology

## 4.3: Project Schematic and Design

The iDrink Station is designed with a dual-level structure. The first rack holds the tanks for juice, water, and milk, along with their dedicated pumps and flow sensors. The second rack contains mixers, a hot water heater, and cleaning units. A power distribution block consolidates all 5V and GND lines for stable performance and easier wiring. Safety elements like a bimetallic thermostat disconnect the heater once the water reaches a safe temperature.

All electronic components are wired to an Arduino Mega, which serves as the main controller. The ESP32 microcontroller connects to it via UART (TX/RX), and handles all wireless communication with the mobile/web application through HTTP requests. The user can also interact locally through an RFID scanner and a 4x4 keypad, while an LCD screen displays the system status, selected drink, and operational feedback.

The full system was prototyped on a breadboard inside the wooden frame and tested in Proteus before being assembled in a wooden enclosure. Careful separation was maintained between control logic and high-current elements to ensure safety and clarity.

**Internal Electrical Wiring**

The internal electrical wiring of the iDrink Station was organized to ensure both safety and functionality. All input and output components, including pumps, mixers, heaters, flow sensors, the LCD, RFID module, and keypad, were connected to the Arduino Mega using male-to-female jumper wires and screw terminals. Each motor was connected through a dedicated motor driver to allow for directional control and voltage regulation.

The system is powered using an external power supply, with a power distribution block used to unify and stabilize the 5V and GND connections across all modules. The ESP32 and Arduino share a common ground, while the 5V supply powers the logic-level components such as the keypad, RFID, LCD, and sensors. For safety, high-current components like pumps and the heater are isolated using relays and drivers, with the bimetallic thermostat acting as a physical cutoff for overheating protection.

# Chapter Four: Methodology

## 4.4: Software Implementation

The software is split between the Arduino Mega and ESP32. The Arduino handles low-level operations: reading from sensors, controlling pumps/mixers, and updating the LCD. The ESP32 connects to Wi-Fi, retrieves commands from a remote server every 3 seconds, and sends them over serial to the Arduino. These include: POWER_ON, POWER_OFF, CLEAN, or DRINK_X.

User commands can be sent either through the keypad or via the mobile/web app. Upon receiving a valid command, the Arduino activates the correct motor sequence and displays updates on the LCD. Some drinks require payment via a second RFID card; if the wrong card is scanned, the user is prompted again.

**Control Units**

The control logic is distributed between two microcontrollers. This separation ensures that wireless functions and hardware logic operate independently but stay synchronized.

1. Arduino Mega: Handles all hardware operations — inputs from keypad and RFID, outputs to LCD, pumps, heaters, and sensors.
2. ESP32: Manages all network communication, polls the server, and relays commands to the Arduino.

**Arduino Mega Logic and Control**

The Arduino Mega is the central brain of the machine, responsible for managing user input, drink preparation logic, motor control, and LCD output. Below is the logic structure that the system follows:

1. **System Activation (RFID / ESP32 Command)**
   - The system remains idle until an authorized RFID card is scanned or a "POWER_ON" command is received from ESP32.
   - Once activated, the LCD displays the main menu and allows the user to select a drink.
   - The system is turned off via "POWER_OFF" command or scanning the RFID card again.

2. **Drink Selection (Keypad / ESP32)**
   The user can press keys 1–6 to select a drink or "5" for cleaning.
   1: Lemon
   2: Blueberry

3: Orange
4: Coffee
5: Milk Drink
6: Clean

Alternatively, a mobile app connected to the ESP32 can send "DRINK_1" to "DRINK_6" or "CLEAN".

3. **Payment Verification (RFID)**
   - If the drink requires payment (e.g., 3 ILS), the system waits for a second RFID card (designated as a payment card).
   - If the wrong card is scanned, the system shows “Wrong Card” and returns to payment screen.

**Drink Processing Algorithm**

Each pump’s duration is carefully timed or controlled via flow sensors to achieve precise volume control. The LCD shows each step in real-time (e.g., “Filling…”, “Finished”). Each drink follows a predefined motor and pump sequence:

1. **Juices (1–3):**
   Pump syrup using corresponding motor
   Pump cold water
   Mix using cold mixer
   Dispense into cup using final pump

2. **Coffee (4):**
   Pump hot water (heater line)
   Mix using hot mixer
   Dispense using final hot pump

3. **Milk Drink (5):**
   Pump milk
   Pump hot water
   Mix and dispense via hot line

4. **Cleaning Mode**
   Runs water into mixers
   Activates mixers and outflow pumps
   Flushes the system for ~10 seconds

**ESP32 Logic and Communication**

The ESP32 acts as a communication bridge between the web application and the Arduino. This makes the system fully operable either locally (RFID + Keypad) or remotely (Mobile/Web App) without disrupting its logic. Its role includes:

1. Connecting to Wi-Fi.
2. Polling a server every 3 seconds for new commands.
3. Sending commands (e.g., "POWER_ON", "CLEAN", "DRINK_2") via serial to the Arduino.
4. Receiving feedback (e.g., "DONE_DRINK_1", "STATUS_OFF") and updating the web interface accordingly.

**System Communication Between ESP32 and Arduino**

The ESP32 and Arduino Mega communicate via a simple UART serial connection. The ESP32 connects to Wi-Fi, sends/receives HTTP commands, and forwards them to the Arduino. The Arduino responds by executing the command (e.g., making a drink), and sends back status messages like DONE_DRINK_1 or STATUS_OFF. This allows the app to stay updated in real-time while still enabling standalone functionality through local controls.

The figure below shows the communication flow between the ESP32 and Arduino Mega:

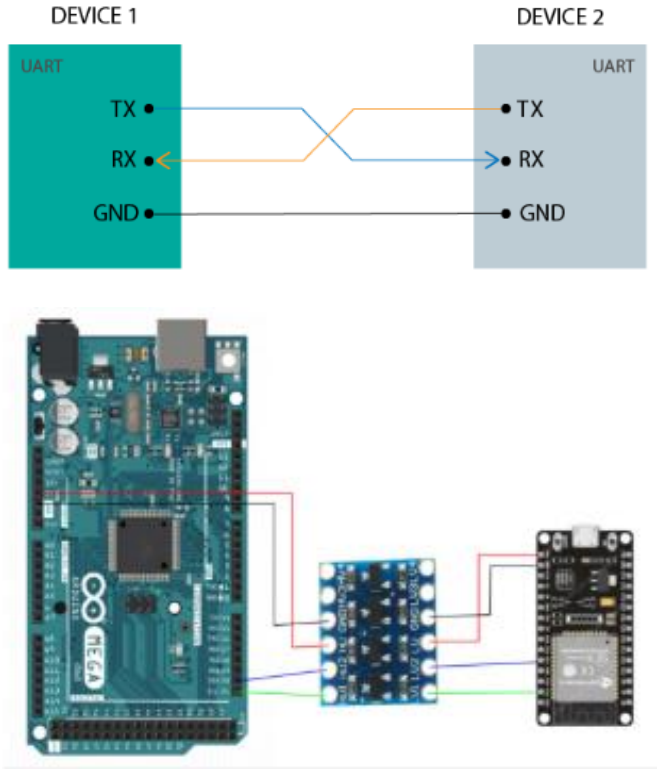


*Figure 24: Serial Communication Between ESP32 and Arduino Mega*

**Software Integration**

The iDrink Station's software is designed to connect embedded hardware with a remote mobile/web interface in a synchronized and scalable way. The system operates through two microcontrollers — Arduino Mega and ESP32 — along with a React/Node.js app. Each plays a specific role while remaining tightly integrated.

The Arduino Mega manages hardware tasks such as reading sensors, controlling motors, responding to keypad inputs, and displaying messages on the LCD. It executes commands either from local input or via the ESP32, which acts as a bridge to the app.

The ESP32 connects to Wi-Fi, communicates with the cloud server via HTTP, and relays commands to the Arduino through serial UART. It also receives feedback from the Arduino to keep the app updated.

The web/mobile app allows users to control the machine remotely by sending instructions (like starting a drink or cleaning cycle) to the ESP32, effectively turning the device into a partially IoT-enabled system.

This layered integration allows real-time interaction, local autonomy, and future expandability, while maintaining clear separation between hardware control and user interface.

**System Flow Summary:**

This hybrid system allows full operation either locally or remotely, offering flexibility and fault tolerance.

1. System activated (RFID or app command).
2. User selects a drink (keypad or app).
3. If required, payment verified via RFID.
4. Arduino executes drink logic (pumps, mixers, heaters).
5. LCD shows process: "Filling", "Finished", etc.
6. Cleaning cycle or shutdown always available.

The figure below illustrates the full system flowchart used in the implementation logic.

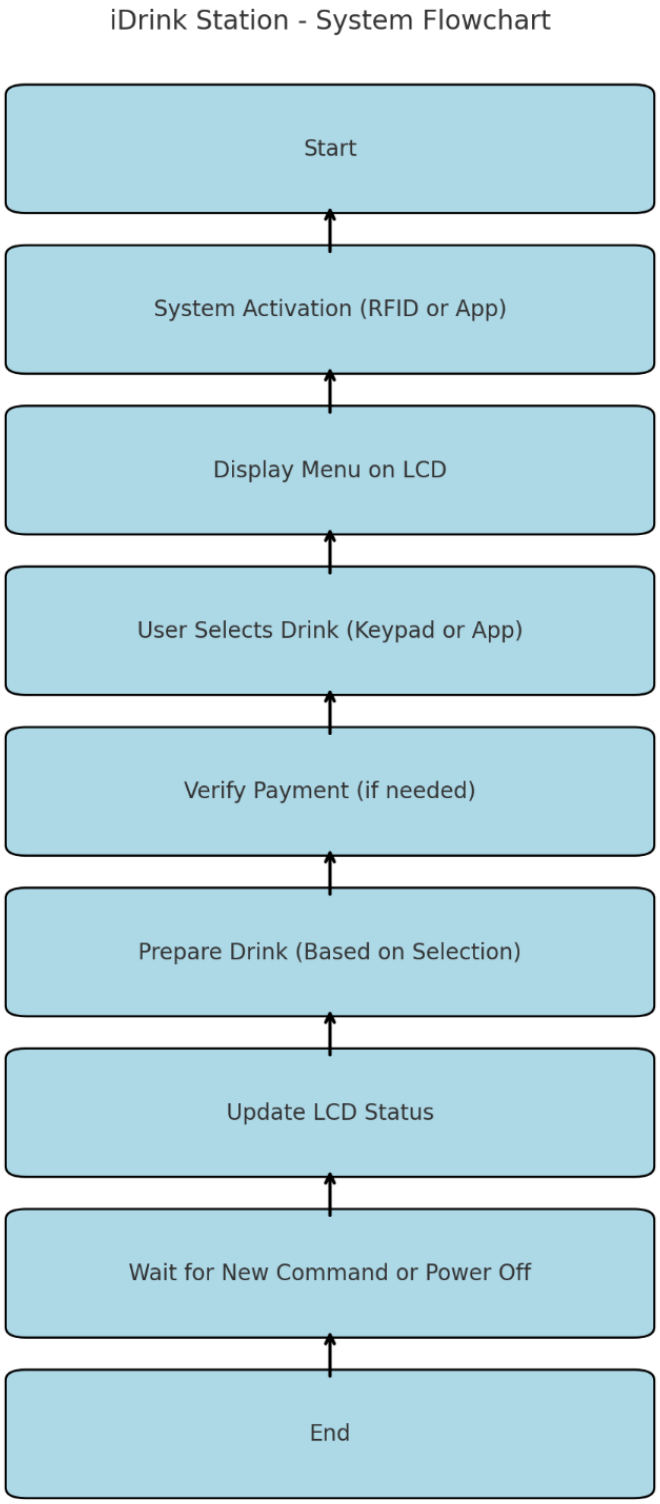


*Figure 25: iDrink Station - System Flowchart*

# Chapter Four: Methodology

## 4.5: Web-Mobile Application

The web-mobile application in the iDrink Station project serves as the digital interface that allows users to remotely interact with the beverage vending system. It was developed using React for the front-end and Node.js for backend logic, creating a responsive and intuitive user experience accessible via browser or mobile device.

Through the application, users can power the system on or off, select the type of drink, initiate the cleaning cycle, and monitor operation status — all in real time. This is made possible through communication with the ESP32 module, which receives commands from the app via HTTP requests and forwards them to the Arduino Mega for execution.

The application provides a clear and simple UI, showing information such as system status, selected drink, and feedback messages (e.g., “Drink Ready” or “Cleaning Complete”). The ESP32 checks the backend API periodically for updates and reflects the results of system operations on the app instantly.

While the machine supports manual control via keypad and RFID, the app offers an added layer of flexibility, allowing users to operate the machine remotely — a feature particularly useful for businesses, small cafés, or high-traffic areas.

This application transforms the iDrink Station from a simple vending machine into a semi-IoT-enabled smart device, combining embedded control with remote digital management. Future improvements may include user login, purchase history, and notifications to enhance the user experience further.

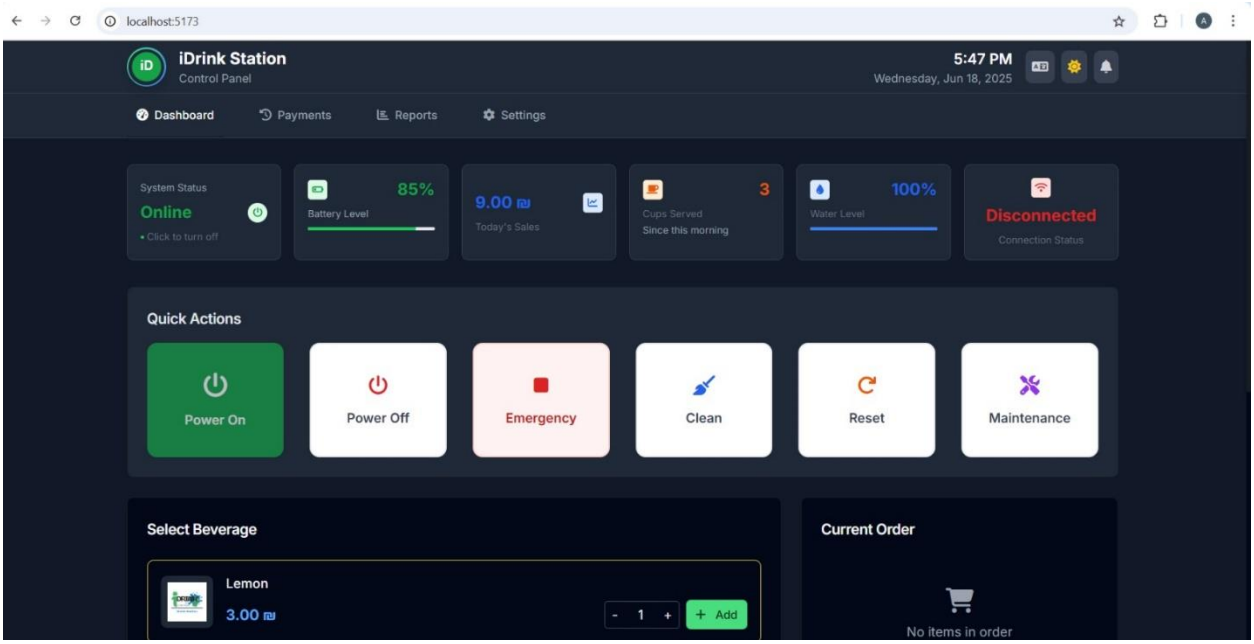


*Figure 26: iDrink Station Web-Mobile Application Dashboard in Dark Theam*

The iDrink Station dashboard offers a simple and responsive interface that allows users to control and monitor the system remotely. It displays key metrics such as system status, battery level, water level, daily sales, and connection status. Through quick action buttons, users can power the machine on or off, start cleaning, reset the system, or trigger emergency and maintenance modes. The drink selection section enables users to choose beverages, adjust quantities, and complete orders with real-time feedback. The dashboard supports both light and dark modes, enhancing usability across different environments.

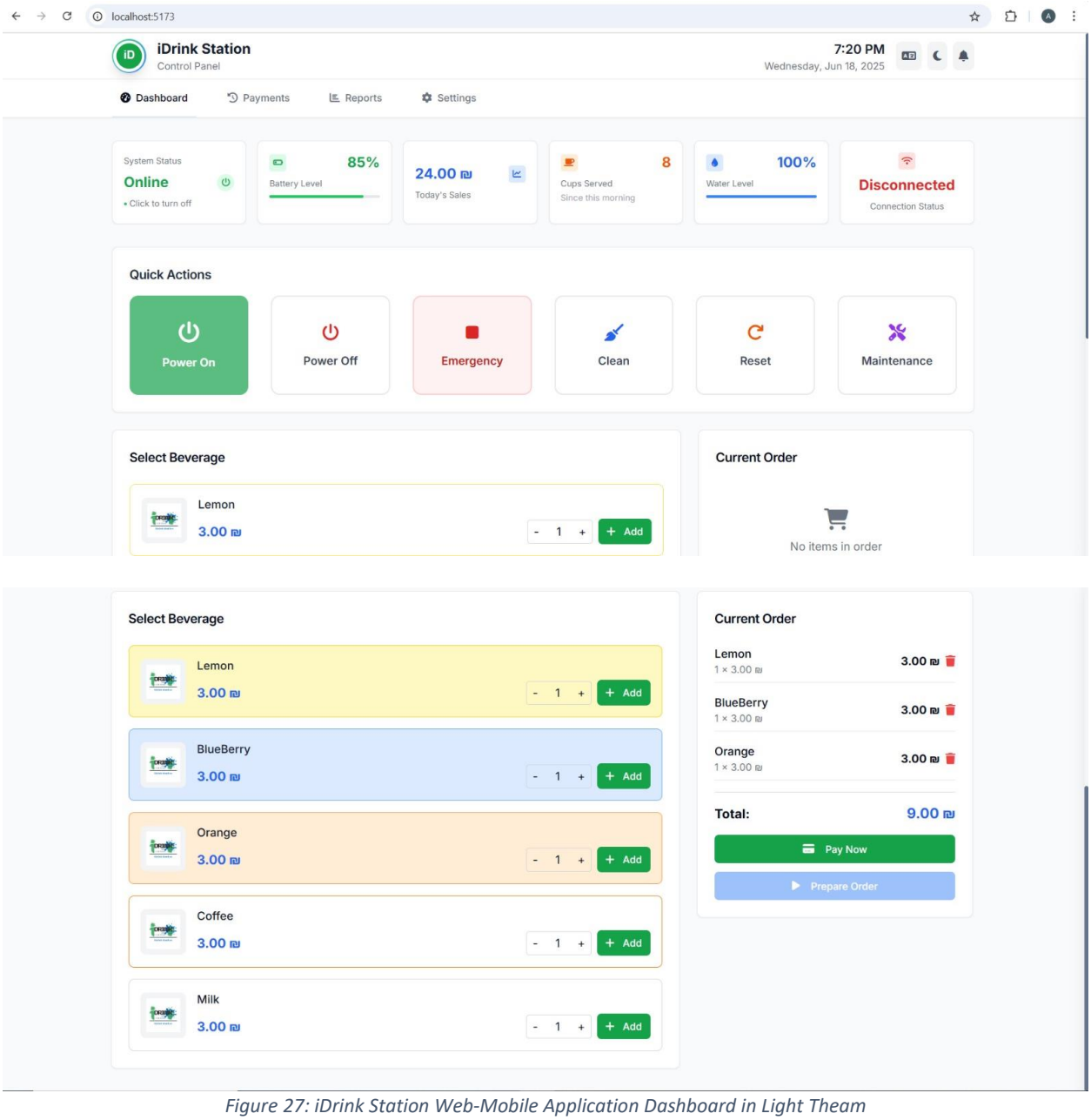


*Figure 27: iDrink Station Web-Mobile Application Dashboard in Light Theam*

Beyond the main dashboard, the iDrink Station system includes two additional sections that enhance administrative control and operational transparency:

Payments and Reports. The Payments page provides a complete transaction history, displaying all purchases made using RFID cards, along with timestamps, drink details, and payment amounts. It allows data export in CSV and PDF formats for record keeping. Similarly, the Reports page offers a summarized view of the system's operations, including total transactions, sales revenue, cleaning cycles, and daily performance. Both sections are designed to support clear data tracking and assist in analyzing the system's usage over time.

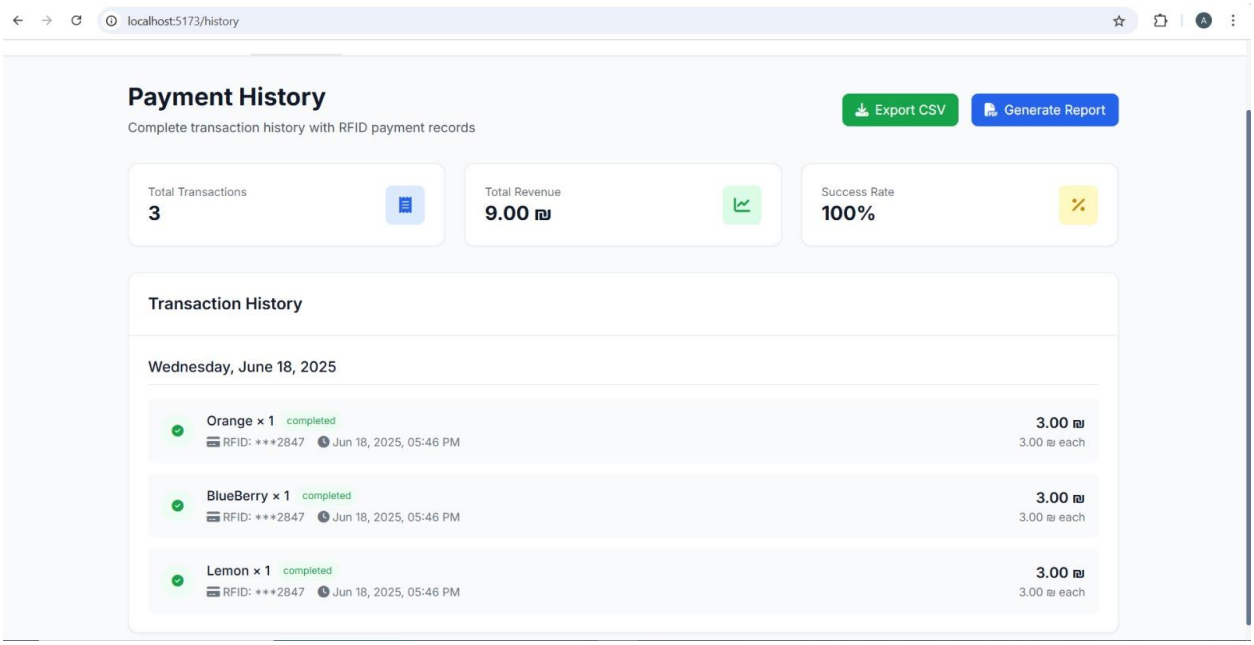


*Figure 28: iDrink Station Web-Mobile Application - Payment Page*

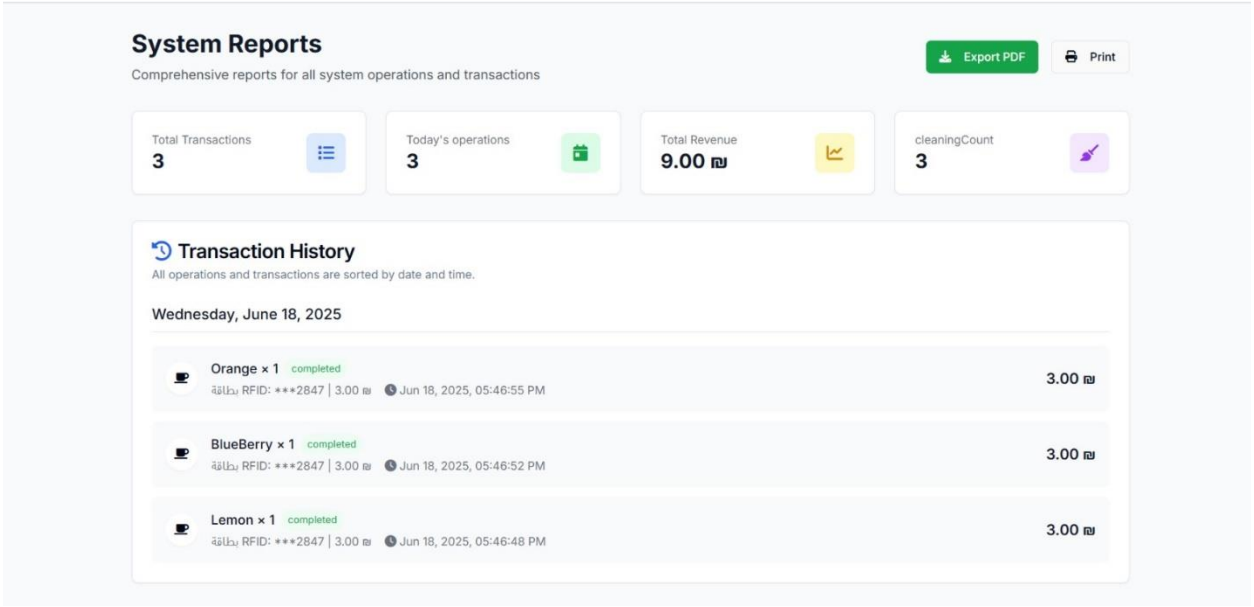


*Figure 29: iDrink Station Web-Mobile Application - Reports Page*

# Chapter Five

# Results and Analysis

# Chapter Five: Results and Analysis

## Functional Testing Results

The iDrink Station system was tested for its ability to carry out essential operations, including powering on/off, drink selection, cleaning, and emergency shutdown. Both manual input via keypad and remote control via the mobile application were successfully executed. Each selected drink was dispensed correctly according to its recipe, with synchronized operation of pumps, mixers, heaters, and sensors. The LCD display accurately reflected system states and instructions during all phases, confirming smooth hardware-software integration.

## Response Time and Efficiency

The system demonstrated fast response times across both local and remote-control inputs. Commands sent via the ESP32 were received and executed by the Arduino Mega within milliseconds. Drink preparation time ranged between 25–40 seconds depending on drink type, which is within acceptable limits for real-time vending scenarios. The cleaning function took under 15 seconds and ensured proper flushing of the fluid lines, confirming efficiency and reliability.

## Application and Interface Performance

The mobile/web application maintained stable communication with the ESP32 through a local IP network. All user commands, including drink orders, power control, and report generation, were successfully transmitted and executed. The interface dynamically updated based on feedback from the machine, displaying real-time system status, drink queue, and order completion. Features like dark/light mode and multi-drink selection contributed to enhanced usability and accessibility.

## Data Tracking and Reporting Accuracy

The system's built-in payment history and report modules accurately recorded transactions and system activities. All RFID-triggered orders were timestamped and stored in the backend, with export options available in PDF and CSV formats. The reports reflected correct drink counts, revenue totals, and cleaning cycles, offering administrators a clear overview of system usage. These results demonstrate the system's reliability in handling operational data and its potential for business-oriented deployment.

# Chapter Six

# Discussion

# Chapter Six: Discussion

The development of the iDrink Station presented a unique opportunity to apply embedded systems, electronics, and software engineering knowledge in a real-world project. Throughout the process, we successfully integrated multiple components—including Arduino Mega, ESP32, RFID, sensors, pumps, heaters, and a web/mobile application—into one cohesive and functional system. This level of integration validated our design logic and confirmed the feasibility of creating an automated drink vending solution using limited resources.

One of the key challenges faced was the synchronization between software commands and physical hardware actions. Achieving accurate drink dispensing required careful calibration of flow sensors and motor durations. The system also had to manage concurrent tasks, such as heating water while running mixing cycles. These requirements pushed us to optimize our control logic, reduce latency in communication, and ensure fail-safe operations through features like emergency stop and cleaning routines.

The project further highlighted the importance of modularity and scalability. By dividing system functions between the Arduino and ESP32, we created a flexible architecture that supports both manual control and remote access. This structure not only simplifies future upgrades—such as cloud integration or touch displays—but also allows the system to operate under various connectivity conditions without losing functionality.

Overall, the iDrink Station performed as expected during testing, meeting its main objectives. However, additional refinements can be made, such as improving the waterproofing of the tubing system, enhancing error detection, or integrating database storage for better transaction tracking. Despite the limitations in time, budget, and experience, the project demonstrated the potential of low-cost smart vending solutions in real-world environments.

# Chapter Seven

# Conclusions and Recommendations

# Chapter Seven: Conclusions and Recommendations

## 7.1: Conclusions

The iDrink Station project successfully achieved its primary goal of creating an automated beverage vending machine capable of preparing both cold and hot drinks using a combination of microcontrollers, sensors, pumps, and a web-based interface. The integration between Arduino Mega and ESP32 allowed for seamless control between local hardware operations and remote user commands through a mobile application. Despite challenges such as limited funding, time constraints, and hardware calibration issues, the system performed reliably during testing, offering users an interactive and efficient experience. The results demonstrate that even with limited resources, a smart, scalable, and partially IoT-enabled vending solution is feasible for academic, commercial, or public use.

## 7.2: Recommendations

To enhance the system further, we recommend adding safety and monitoring features such as flow error detection, water-level sensors, and real-time feedback for system malfunctions. Future versions could include a touchscreen interface, cloud integration for remote analytics, and user account features for personalized ordering. On the hardware side, using more compact PCBs, waterproof tubing, and optimized wiring would improve the system's durability and serviceability. Finally, integrating a full database backend could improve transaction history tracking and analytics for business deployment.

# References

## References


Arduino Documentation – Official Site.
https://www.arduino.cc/reference/en/

Espressif Systems. ESP32 Technical Reference Manual.
https://www.espressif.com/en/products/socs/esp32/resources

MFRC522 RFID Reader Module Guide – How to Use with Arduino.
https://randomnerdtutorials.com/security-access-using-rfid-reader/

Keypad Input with Arduino – Guide and Library.
https://playground.arduino.cc/Main/KeypadTutorial/

LCD I2C 20x4 with Arduino – Setup and Display Control.
https://www.instructables.com/I2C-LCD-How-to-Connect/

Flow Sensor with Arduino – Accurate Water Flow Measurement.
https://lastminuteengineers.com/yf-s201-water-flow-meter-arduino-tutorial/

Node.js and Express – Official Documentation.
https://expressjs.com/

React – A JavaScript library for building user interfaces.
https://react.dev/

HTTP Client Library for ESP32 (Arduino Core).
https://github.com/espressif/arduino-esp32/tree/master/libraries/HTTPClient

Bimetallic Thermostat Working Principle.
https://www.electronics-tutorials.ws/blog/bimetallic-thermostat.html

Proteus Design Suite – Circuit Simulation Software.
https://www.labcenter.com/