Automation of Beer Dispensers: A Cyber-Physical System Solution

Lorenzo Piarulli[1,4], Luca Di Pietro[1,3], José A. Gaspar[1], Miguel A. Teixeira[1], Jaume Sansano[1,5], Lucas Sousa[1], Luís Paiva[1], Emanuel Gestosa[1], Alexandre Correia[1], Rui Pinto[1,2], Gil Gonçalves[1,2]
Dept. of Informatics Engineering Faculty of Engineering, University of Porto, Porto, Portugal[1]
SYSTEC, ARISE, Faculty of Engineering, University of Porto, Porto, Portugal[2]
Dept. of Computer Science Engineering Polytechnic University of Milan, Milan, Italy[3]
Dept. of Computer Science Engineering Sapienza University of Rome, Rome, Italy[4]
Dept. of Informatics Engineering Polytechnic University of Valencia, Valencia, Spain[5]
Email: {up202401433, up202401432, up202008561, up202005208, up202402362, up202004682, up202006094, up202005485, up202007042}@up.pt
{rpinto, gil}@fe.up.pt

***Abstract*—This study explores the implementation of Internet of Things technology to improve drink service and management in local establishments. Current alternatives were analyzed but failed to satisfy all the problem's requirements, leading us to propose a new solution. Our system uses a mobile application that allows users to purchase beer tokens, which can be uploaded to establishment cups using Near Field Communication (NFC) technology. Once the cup contains a token, it can interact with a prototype self-serving tank that dispenses the drink upon validation by the client. The app also provides real-time updates on tank temperature and level for the owner. A key consideration during the design phase was ensuring the architecture's scalability to handle multiple interactions between devices. The successful development of our prototype demonstrates the feasibility of implementing this technology in real-world establishments, highlighting its market potential. Finally, we analyzed the product's limitations and proposed directions for future research and development.**

Keywords-Internet of Things; Cyber-Physical Systems; self-service; service tank; automation.

I. INTRODUCTION

In recent years, there has been growing interest in the Internet of Things (IoT) [1] in fields such as personal goods, transportation, and industry due to the advances in technology and the promising results of its implementation [2]. IoT and Cyber-Physical Systems (CPS) [3] are expected to take a major role in future production value chains by using networks of sensors and actuators to gather data.

A critical challenge in the service sector is the inefficiency caused by human limitations. Nowadays, it is common for establishments to face a simultaneous high demand when dealing with traditional staff systems. The growing acceptance of self-serving devices [4] encourages big companies to implement similar technologies in areas, such as gas stations or supermarkets, removing the human factor of the cashiers. This change of mentality could be extended through other consumer sectors such as bar service. This study aims to address these challenges by developing a self-serving beer dispenser powered by IoT and CPS. The primary objectives of this work are to reduce waiting times, improve customer experience, and provide real-time data insights to establishment owners for better resource management.

Our solution consists of using IoT to create a self-serving tank that allows the user through their phone to purchase, validate, and receive a drink without the need for a member of staff. The user uses NFC tags attached to the cups to load tokens purchased via our phone app, then they can walk to the tank, validate that they want to use the token and receive a drink. This removes the necessity of a staff member and optimizes the service process and makes it more dynamic. To provide insights to the owner we can also measure in real-time values of the tank's level and temperature to ensure uninterrupted service. The resulting product could help an establishment to have a better understanding of its resources and promote a more direct service to the customers focusing on reducing waiting time and cost.

The remaining sections are arranged as follows: Section II reviews the most relevant related works in the field of automated beverage dispensing systems, highlighting their limitations. Section III describes the proposed solution, including its design, architecture, and main components. Section IV presents the results obtained and evaluates the performance of the prototype. Section V discusses the advantages, limitations, and potential improvements of the system. Finally, Section VI concludes the paper by summarizing the main contributions and possible future changes.

II. RELATED WORK

Efforts to enhance efficiency, automation, and user experience in beverage dispensing have spurred the development of various smart beer dispensing systems, each with distinct strengths and limitations [5]. The Pubinno Smart Tap [6] exemplifies the use of artificial intelligence to optimize beer quality by fine-tuning temperature and pressure levels. Despite these innovations, its reliance on proprietary kegs and the absence of self-service features may hinder its accessibility. A staff-centric application accompanies the system, offering tools for monitoring performance.

PourMyBeer [7], by contrast, empowers patrons with self-service capabilities using Radio Frequency Identification (RFID) technology, simplifying transactions and providing valuable sales insights for operators. While compatible with standard kegs, the system lacks real-time authentication and quality assurance features. Similarly, Boxbar Tech [8] provides automated self-service kiosks designed to minimize wait times and boost service speed. Although the kiosks support standard kegs, they do not specialize in beer-specific features, such as temperature control, but include an app for inventory and sales oversight.

Another self-service solution, the iPourIt system [9], grants users access to a variety of beverages while enabling real-time data monitoring. However, its limitations lie in security and centralized control integration, although it offers operators tools for live system configuration and data oversight. Heineken SmartDispense [10], on the other hand, prioritizes beer quality through advanced cooling and line management techniques, promoting sustainability. Yet, it lacks self-service functionality and real-time monitoring, while relying on proprietary kegs. Its accompanying application focuses on staff support, tracking temperature and cleaning schedules.

These systems highlight diverse approaches, emphasizing either quality assurance, operational efficiency, or self-service functionality. However, they often fall short of addressing comprehensive needs, such as robust security, authentication, and user-centric innovation. This analysis underscores the potential for future systems to bridge these gaps and adapt to evolving demands in the industry.

III. Proposed Solution

In the following sections, we present the implemented architecture, which follows the prototype illustrated in Figure 1. Our solution aims to enhance transaction security for commercial purposes. As shown in the scheme, when the user initiates liquid pouring from the tank, several steps must be completed. First, the user generates a token in the application, which is then transmitted to the cup and subsequently to the tank. The tank verifies the token with the cloud, which sends a confirmation request to the user. The pouring system starts only after the cloud confirms the token and the user accepts the request.

A. Hardware

The hardware aspect of this project incorporates two types of microcontrollers: the Arduino Nano WiFi [11] and the Raspberry Pi 4 Model B (2GB) [12], as represented in Figure 2.

The Raspberry Pi serves as an Edge server, managing local communication between the automated tanks. The system is designed so that each shop equipped with automated beer dispensing tanks has an Edge server (in this case, the Raspberry Pi implementation) to oversee all associated tanks. Each tank automation system employs an Arduino for its operations. Thus, every shop has multiple Arduino-based systems connected to the shop's Raspberry Pi, which centralizes and handles communications.

The Arduino system is responsible for monitoring the system, sending data, detecting RFID cup tokens, and pouring the correct amount of beer. It serves as the central controller for the beer dispensing system. The initial prototype was

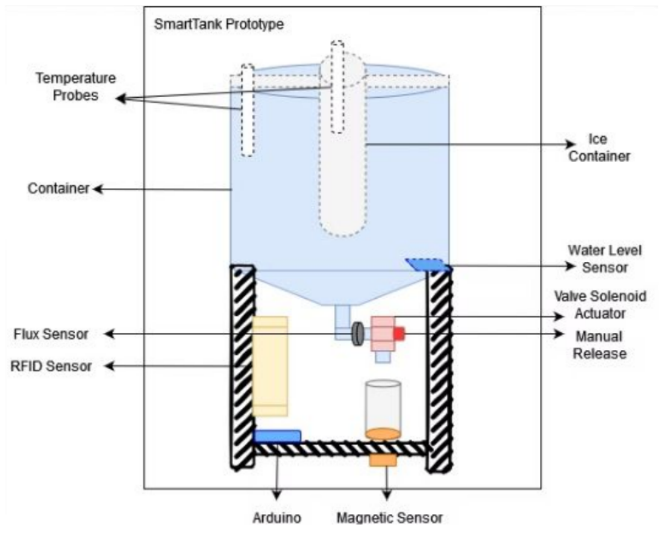


Figure 1. Overview of the system's proof of concept.

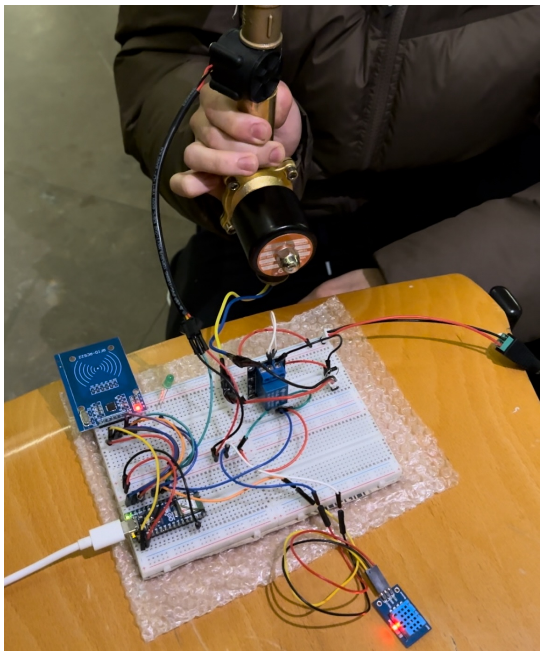


Figure 2. Hardware system with all the components.

built using an Arduino Uno, which lacks WiFi capabilities. After verifying the electronic circuit and achieving a working prototype, we transitioned to the Arduino Nano WiFi to enable seamless communication for data transmission and the inclusion of token validation features. Regarding the electronics, the circuit integrates the following sensors and actuators: I) RFID Reader (RFID-RC522) with associated tags; II) Humidity and Temperature Sensor (DHT11); III) Relay Module(JQC3F-5VDC-C); IV) Flux Sensor (Youmile YF-S201); V) Solenoid Valve (BACOENG B06XFD1CB4); VI) 12 V Charger; VII) Diode; VIII) Resistors and cables.

The system starts by identifying a user's RFID-equipped cup, using the RFID Token Detection. The token, preloaded via a mobile application, is transmitted from the cup's RFID tag to the Arduino through the RFID reader. Once the token is read, the Arduino requests validation from the central system and awaits confirmation. After validation, the Arduino signals the relay to activate the solenoid valve. The relay connects the valve to the 12V power supply, allowing the valve to open. With the valve open, the beer dispensing starts, i.e., the liquid begins to flow. The flux sensor (Youmile YF-S201 [13]) measures the volume of fluid dispensed in millilitres. Once the desired amount of beer is dispensed, the Arduino closes the valve and resets the flow status.

The system employs a solenoid valve activated by a relay, which requires a higher voltage than the Arduino's output. The relay is controlled by a 3.3V signal from the Arduino, allowing the solenoid to open and regulate liquid flow, with pressure maintenance ensuring proper functionality. Liquid flow is monitored by a sensor that generates pulses, which the Arduino processes to calculate volume after calibration for accuracy.

Both the solenoid valve and flow sensor feature commercial connectors for easy replacement and adaptation, supporting future upgrades. The system provides staff with key metrics like liquid volume and temperature, with a DHT11 sensor measuring temperature and humidity at adjustable intervals. Data is sent to a Raspberry Pi, which forwards it to the Cloud for monitoring.

Finally, the Firmware, developed using the Arduino IDE in C/C++, integrates RFID, sensor, and actuator control. Despite the Arduino's single-process nature, interrupt-based programming ensures efficient task management and continuous data collection without disruption. The modular code structure supports easy updates and the addition of new features.

B. Communication

The communication part of the project aims to define a scalable architecture for the network and the best communication protocols to go along with it. The chosen architecture is detailed in Figure 3. The overall topology of the network follows a star topology, with a Message Queuing Telemetry Transport (MQTT) [14] broker in the central hub. The communication protocols used are Hyper Text Transfer Protocol Secure (HTTPS) and MQTT.

Transitioning to Edge Computing or Decentralized Communication presents several challenges, including high infrastructure costs, complex management, and security risks due to increased attack surfaces. Ensuring data synchronization and consistency across distributed tanks is difficult, especially with limited computing power and connectivity issues. Software updates, interoperability, and failure handling require robust solutions, while compliance with regulatory standards can be complex without centralized control. Despite these challenges, edge computing offers lower latency, enhanced privacy, and reduced cloud dependency, making it a promising but demanding shift. Based on these facts, we have decided to build an architecture that leverages the benefits of both edge computing and the cloud.

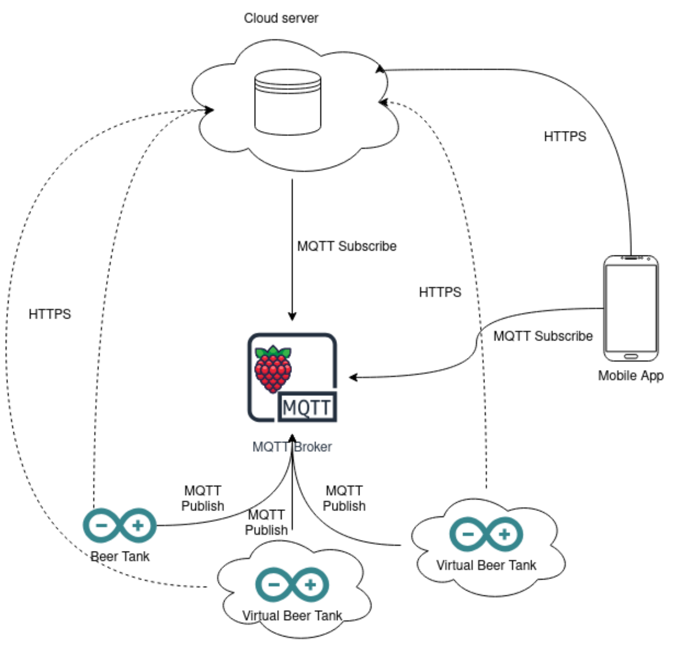


Figure 3. Overview of the system's components and their communication.

The Beer Tanks are the Edge devices, and the MQTT broker is in the same network as them, acting as a Fog layer, bridging the Edge to the Cloud layer. The MQTT protocol is used to publish sensor data from the Edge devices and to subscribe to it from the Cloud and the mobile application, where the raw data is processed and visualized.

The HTTPS protocol is used for mobile application to Cloud communication, and also for some sporadic communication between the Edge devices and the Cloud. This means that the Edge devices may bypass the Fog layer and communicate directly to the Cloud, but this is only done for purposes of validating beer tokens. This type of validation is done by sending a request for validation and waiting for a response, therefore using MQTT, which follows a publish-subscriber paradigm, for this type of communication made no sense, hence why we allow the use of HTTPS, which follows a request-response paradigm, in this specific case.

While in Figure 3 we presented an architecture with the MQTT broker as the central hub, we still accounted for the possibility of scaling the system for having multiple brokers with just minimal adjustment. Since the Edge devices and the MQTT broker are situated on the same network, they are meant to all be in the same establishment, i.e., the place that sells the beer to the public. Therefore, to scale for multiple establishments, each one of them would have their own Edge devices and their MQTT broker, and all of them would communicate with the Cloud.

C. Cloud

The objective of the API is to provide logic to the business. It supports secure client login, real-time tracking of tank parameters, data offload, and logic processing to provide better implementation of the app. Provides features such as the

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org

management of organizations, tanks, and human resources, as well as token management and key performance indicator systems.

The structure of the API's database consists of the models for establishments, users, tokens and tanks. Establishment models contain details about the bar which are used in the user model for login and access to other functionalities. Tokens are a multipurpose means that allows the user to exchange them for a drink from a particular machine, they link a user to a machine to an organization. There are other related models attached to tanks, which determine the amount of beer, its temperature and its history of consumption. The database is structured with normalized relational models to efficiently manage data integrity across entities. Advanced relationships between models, such as the many-to-many mapping of users and establishments via the *EstablishmentStaff* model, enable precise role-based access control and granular permission management. Additionally, the system supports dynamic endpoint handling, enabling contextual validation for complex workflows like token verification.

The API's endpoints handle a variety of tasks. User authentication endpoints enable registration, login, and auth token refreshing. The establishment management endpoints allow users to create establishments and retrieve related tanks or tokens. The token endpoints support the creation, updates, and verification of tokens. The staff endpoints retrieve the personnel details for the establishments. Statistics endpoints provide real-time data on tank performance, such as beer served, temperature, and levels, as well as aggregated data for establishments. Admin-specific endpoints allow access to all users and establishments for high-level management.

The API is deployed on Render.com with an automated system that pushes changes to production after every code commit. Security is ensured by requiring JavaScript Object Notation (JSON) Web Tokens (JWT) for all secure endpoints, with role-based access control distinguishing between admin and regular user permissions. This API serves as a scalable backend solution for managing self-service beer pouring systems. It combines secure user management, comprehensive data tracking, and analytics into a single service, ensuring ease of use.

D. Security

This project addressed several non-functional security requirements to ensure the system's robustness, such as Token Protection; Communication Flow; Secure Communication Protocols; Server Security; Internal Network Security; and Physical Security.

To prevent misuse of a stolen cup loaded with a beer token, a confirmation feature was implemented. When the user places the cup near the NFC sensor, a request is sent to the application for confirmation. The user must approve the request via the application's confirmation page to activate the machine and spend the token. This two-step process ensures only the rightful token owner can use the machine.

The hardware sends a confirmation request to the Cloud, and the application polls the Cloud to retrieve pending confirmations. Upon user approval, the information is sent back through the Cloud to the machine. While direct communication between the application and hardware would be more efficient, the Cloud was used as an intermediary because it verifies token validity and manages user confirmations.

Ensuring secure communication between system components is critical. All communication uses cryptographic protocols, such as HTTPS instead of HTTP, and MQTT over Transport Layer Security (TLS) [15]. Additionally, since MQTT does not enforce authentication by default, the system was configured to require strong, randomly generated passwords for authentication.

Remote management of servers introduces additional security considerations. For Cloud servers, security largely depends on the provider, but standard practices, such as using strong passwords, are essential. For Fog servers, owned and managed by the system operator, secure deployment and regular updates are critical to minimizing vulnerabilities.

In the beer establishment, where Edge and Fog devices operate, network security is crucial. The Fog device, serving as the MQTT broker, must be accessible from the Internet. To secure this setup, the Fog device should be placed in a demilitarized zone (DMZ) [16] with a firewall separating it from the Edge devices. This configuration mitigates potential risks to the internal network.

Edge devices are also vulnerable to physical attacks, whether by malicious users attempting to obtain free beer or accidental damage. Implementing robust hardware security measures is as important as addressing cyber threats.

In terms of non-functional requirements, this project took into account a few security concerns. The first one is the possibility of a stolen cup, with a loaded beer token in it. To avoid someone else, other than the token owner, to activate the machine, a confirmation feature was implemented. The idea is that, upon activation, this is, when the user passes the cup near the NFC sensor of the machine, a request is sent to the application in the form of a confirmation. The user must go to the confirmation page, in the application, and confirm that the user wants to use the machine with that token. Only if the user accepts the confirmation, the machine is activated and the token is spent. In terms of implementation, the hardware sends the request to the Cloud, and on the application side, a polling system is done to get the latest confirmations. Upon user confirmation, the information goes through the Cloud again and arrives at the machine. This solution is not optimal, it would be more efficient if the communication was directly made between the application and hardware, however, the Cloud was the only element that checked for token validity and it was also responsible for getting all confirmations for a given user.

Providing secure communications between the components of the system is also a critical requirement. Communication protocols should always use cryptographic enhancements to achieve security properties. Therefore, HTTPS is always used instead of HTTP, and MQTT is also used on top of TLS [15]. Also, MQTT by default doesn't require any authentication, so we had to configure it to require authentication using a strong, randomly generated password.

We also consider that it is desirable to have the servers, both on the Fog and the Cloud layers, be remotely managed. With that comes more security implications. For the Cloud, it will depend mostly on the Cloud provider, but standard security practices, such as choosing strong passwords, still apply. For the Fog layer, since it is controlled by the owner of the system, one has to be very careful to deploy it without security flaws and keep it properly updated.

E. Application

A mobile app was chosen over a web app for its convenience and user-friendly interface. The app allows users at establishments equipped with dispensing machines to view, use, and purchase tokens easily. For staff, it provides real-time and historical insights into beer tank metrics, such as temperature and beer levels, which are crucial for optimizing operations and boosting profitability. Development and Security

The application is developed using Flutter [17], ensuring cross-platform compatibility. It includes user-centred features such as login, registration, and session management. A session extension mechanism minimizes the need for frequent logins. Upon login, users receive two tokens: I) Bearer Token: Used for secure requests to the Cloud and automatically refreshed when invalid; II) Refresh Token: Enables the generation of new bearer and refresh tokens. If the refresh token expires (less frequently), the user must log in again.

Users can generate beer tokens for specific establishments, initialized with the status *"Phone"*. Tokens are listed in the app and, when selected, can be loaded into an NFC-equipped cup. The app uses the phone's NFC module to write the token, changing its status to *"Cup"* and syncing this update with the Cloud. The NFC writing process uses Flutter's NFC Manager package [18], encoding the token as a 4-digit integer in the NFC Data Exchange Format (NDEF) [19]. Also, the app displays each establishment's tanks and total beer servings. Detailed metrics, such as beer level and temperature, are available for individual tanks. Historical data is visualized through graphs created using the FL Chart [20] package, enabling better decision-making and operational efficiency.

Regarding real-time data, Beer tank metrics (temperature and level) are collected via an MQTT channel from the establishment's Raspberry Pi using the Pub/Sub protocol. This ensures immediate updates for the app. There are also historical data, pre-processed in the Cloud and retrieved via an API. This is formatted for visualization on the app, reducing client-side processing overhead.

Finally, an API wrapper was developed to streamline communication with the Cloud. Although not a core feature, it significantly simplifies app development. All communication uses secure POST and GET HTTPS requests, protected by bearer tokens.

IV. Validation and Result Analysis

In this section, we present the methods used to test our automated beer dispensing system, discuss the experimental setup and analyze the outcomes to demonstrate the system's effectiveness and feasibility.

A. Testing Methodology

To verify the functionality, performance, and reliability of our automated beer dispenser, we established the following testing methodology:

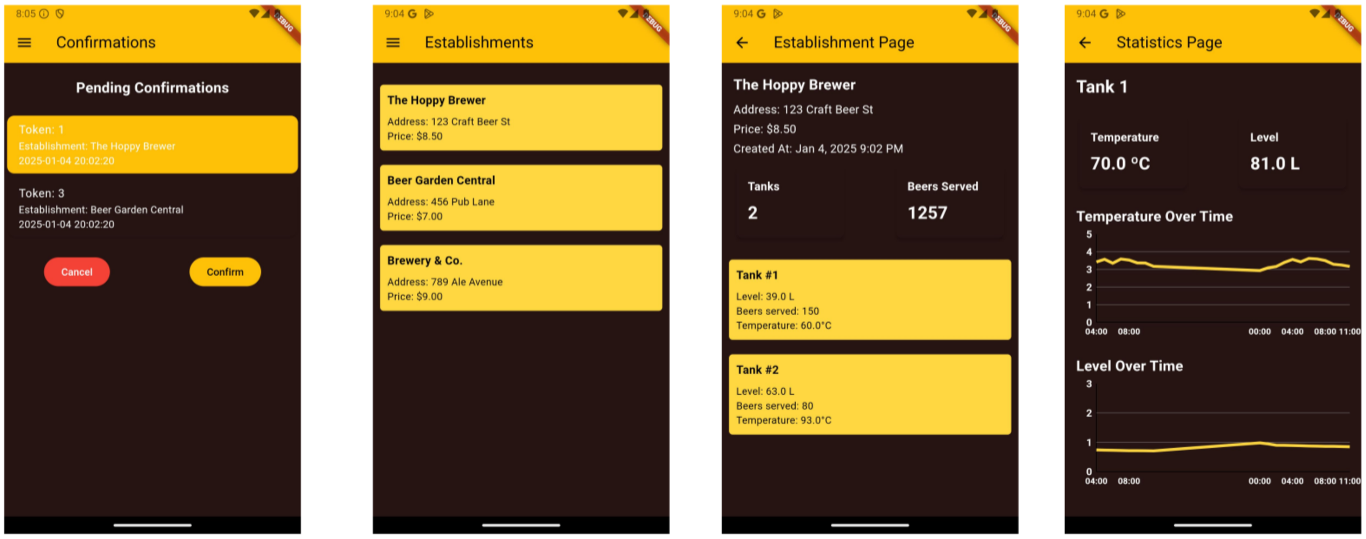
- **Functional Tests:** We examined the basic features (e.g., token validation, solenoid valve control, flow sensor readings) in a controlled environment. Each feature was tested independently, and success/failure logs were recorded.
- **Integration Tests:** Once the individual modules were verified, we tested the entire workflow from token creation on the mobile application to the final dispensation of beer. This aimed to spot any synchronization or communication issues among the Arduino, Raspberry Pi, mobile application, and the Cloud.

B. Result Analysis

Through our tests, we confirmed that the prototype is both functional and robust, successfully meeting the specified performance objectives in terms of reliability, scalability and security. Figure 4 shows the final results on the application side. Note that the data displayed in these figures are mocked values rather than actual sensor readings. Nonetheless, the sensors were fully tested and confirmed to be working correctly during firmware development.

V. Discussion

The development and implementation of the automated beer dispensing system aim to address periods of high demand and labour shortages by providing an efficient, self-service solution. The system offers several advantages that enhance its overall effectiveness and adaptability. By streamlining the beverage-dispensing process, it reduces waiting times and improves the customer experience. Operational efficiency is significantly increased as the system automates repetitive tasks, allowing staff to focus on enhancing service quality. The integration of cloud infrastructure enables real-time monitoring and analysis of operational data, providing valuable insights into customer behaviour and resource management. Security is strengthened through NFC tokens and confirmation systems, which restrict access to authorized users only. Moreover, the system's modularity and scalability allow for easy upgrades and adaptations to evolving operational needs, while proper customization can help minimize waste, optimize resource utilization, and promote sustainability. However, despite these advantages, the system faces several limitations that impact its performance and usability. Communication latency, due to reliance on cloud hosts for token validation and confirmation, may lead to delays that affect the user experience in high-demand scenarios. Hardware constraints, particularly the single-threaded nature of the Arduino, limit its ability to



Confirmations

Pending Confirmations

Token: 1
Establishment: The Hoppy Brewer
2025-01-04 20:02:20

Token: 3
Establishment: Beer Garden Central
2025-01-04 20:02:20

Establishments

The Hoppy Brewer
Address: 123 Craft Beer St
Price: $8.50

Beer Garden Central
Address: 456 Pub Lane
Price: $7.00

Brewery & Co.
Address: 789 Ale Avenue
Price: $9.00

Establishment Page

The Hoppy Brewer
Address: 123 Craft Beer St
Price: $8.50
Created At: Jan 4, 2025 9:02 PM

Tanks 2
Beers Served 1257

Tank #1
Level: 39.0 L
Beers served: 150
Temperature: 60.0°C

Tank #2
Level: 63.0 L
Beers served: 80
Temperature: 93.0°C

Statistics Page

Tank 1

Temperature 70.0 °C
Level 81.0 L

Figure 4. Confirmations, establishments and statistics screens.

handle multiple concurrent operations, potentially affecting real-time performance. The solenoid valve's dependency on stable liquid pressure makes it less reliable under varying conditions. Additionally, users unfamiliar with NFC technology or app-based interactions may initially find the system less intuitive, affecting adoption rates. While digital security is robust, the potential for physical theft of RFID cups remains a concern. Despite these challenges, the system presents a promising solution with opportunities for further improvement and optimization.

Addressing some of those limitations would not compensate the additional cost required to solve them. Examples of those improvements would be the usage of a more robust valve in terms of pressure, which will make the prototype suitable for most liquids, usage of hardware devices that support multi-threading and new hardware devices used in Edge computing to avoid reliance on the Cloud. The solution has been built for easy actuator and sensor replacement, allowing the system to adapt to new solutions that require automated tanks, like for agricultural or industrial applications.

VI. Conclusion & Future Work

Using IoT, mobile app and other automation hardware, the automated beer dispensing system built in this project shows that we can overcome some of the operational bottlenecks in beverage service in general. The system shows great promise in increasing customer satisfaction and operational efficiency by enabling self-service capabilities, reducing wait time and providing real-time data insights. Due to the modular nature of the design, it can be scaled and adapted to various use cases, making it a versatile solution.

In the future, we will continue to improve the system architecture by trying Edge computing or decentralized communication models to avoid reliance on the Cloud. To improve real-time responsiveness and multitasking capabilities, hardware upgrades, including the use of multithreaded microcontrollers, will also be explored. Machine learning algorithms could also be integrated for predictive maintenance and dynamic adaptation based on user behaviour. In addition, features such as the implementation of different cup sizes and the consideration of the beer supplier as an actor in the system will be introduced. This will allow the system to automatically order beer from the supplier when the bar runs out, ensuring seamless operation. Lastly, usability improvements will be made based on user feedback, keeping the system intuitive and accessible to a wider audience. These improvements could make the automated beer dispensing system a groundbreaking innovation in the service sector.

Acknowledgment

This work is financially supported by national funds through the FCT/MCTES (PIDDAC), under the Associate Laboratory Advanced Production and Intelligent Systems – ARISE LA/P/0112/2020 (DOI 10.54499/LA/P/0112/2020) and the Base Funding (UIDB/00147/2020) and Programmatic Funding (UIDP/00147/2020) of the R&D Unit Center for Systems and Technologies – SYSTEC.

References

[1] D. Raggett, "W3c plans for developing standards for open markets of services for the iot: The internet of things (ubiquity symposium)," *Ubiquity*, vol. 2015, no. October, pp. 1–8, 2015.

[2] E. Borgia, "The internet of things vision: Key features, applications and open issues," *Computer Communications*, vol. 54, pp. 1–31, 2014, ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2014.09.008.

[3] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: The next computing revolution," in *Design Automation Conference*, 2010, pp. 731–736. DOI: 10.1145/1837274.1837461.

[4] M. L. Meuter, A. L. Ostrom, R. I. Roundtree, and M. J. Bitner, "Self-service technologies: Understanding customer satisfaction with technology-based service encounters," *Journal of marketing*, vol. 64, no. 3, pp. 50–64, 2000.

[5] S. Violino, S. Figorilli, C. Costa, and F. Pallottino, "Internet of beer: A review on smart technologies from mash to pint," *Foods*, vol. 9, no. 7, p. 950, 2020.

[6] Pubinno, *Smart Tap: AI-powered Beer Dispensing System*, https://pubinno.com/smart-tap/, [Accessed: January 9, 2025], 2025.

[7] PourMyBeer, *Self-Pour Tap System*, https://pourmybeer.com/, [Accessed: January 9, 2025], 2025.

[8] Boxbar Tech, *Automated Self-Serve Bar Terminals*, https://www.boxbar.live/, [Accessed: January 9, 2025], 2025.

[9] iPourIt, *Self-Pour Beverage System*, https://ipouritinc.com/, [Accessed: January 9, 2025], 2025.

[10] HEINEKEN, *SmartDispense: Draught Beer Dispense System*, https://smartdispense.heineken.co.uk/, [Accessed: January 9, 2025], 2025.

[11] H. Al-Mimi, A. Al-Dahoud, M. Fezari, and M. S. Daoud, "A study on new arduino nano board for wsn and iot applications," *International Journal of Advanced Science and Technology*, vol. 29, no. 4, pp. 10 223–10 230, 2020.

[12] H. D. Ghael, L. Solanki, and G. Sahu, "A review paper on raspberry pi and its applications," *International Journal of Advances in Engineering and Management (IJAEM)*, vol. 2, no. 12, p. 4, 2020.

[13] Mauser, *Youmile YF-S201*, https://mauser.pt/catalog/product_info.php?products_id=095-0376, [Accessed: January 13, 2025], 2025.

[14] M. Bender, E. Kirdan, M.-O. Pahl, and G. Carle, "Open-source mqtt evaluation," in *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, IEEE, 2021, pp. 1–4.

[15] P. V. A. Alexandre Urzhumtsev and P. D. Adams, "Tls from fundamentals to practice," *Crystallography Reviews*, vol. 19, no. 4, pp. 230–270, 2013. DOI: 10.1080/0889311X.2013.835806. eprint: https://doi.org/10.1080/0889311X.2013.835806.

[16] J. Crichigno, E. Bou-Harb, and N. Ghani, "A comprehensive tutorial on science dmz," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 2041–2078, 2019. DOI: 10.1109/COMST.2018.2876086.

[17] M. L. Napoli, *Beginning flutter: a hands on guide to app development*. John Wiley & Sons, 2019.

[18] S. Ghosh, J. Goswami, A. Kumar, and A. Majumder, "Issues in nfc as a form of contactless communication: A comprehensive survey," in *2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*, IEEE, 2015, pp. 245–252.

[19] S. Hameed, U. M. Jamali, and A. Samad, "Integrity protection of ndef message with flexible and enhanced nfc signature records," in *2015 IEEE Trustcom/BigDataSE/ISPA*, IEEE, vol. 1, 2015, pp. 368–375.

[20] J. Nanavati, S. Patel, U. Patel, and A. Patel, "Critical review and fine-tuning performance of flutter applications," in *2024 5th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI)*, IEEE, 2024, pp. 838–841.