A Low-cost IoT-based Solution to Monitor Customer Behavior in Storefront Displays

Bruna Henning Pereira
University of Vale do Itajaí
Itajaí, Brazil
bruna.henning@edu.univali.br

Ernani Alexandre Wippel Neto
University of Vale do Itajaí
Itajaí, Brazil
ernani.neto@edu.univali.br

Cesar Albenes Zeferino
University of Vale do Itajaí
Itajaí, Brazil
zeferino@univali.br

Abstract

This works presents the development and evaluation of an IoT-based system designed for monitoring customer engagement through ultrasonic sensing and real-time data transmission. The proposed solution employs an ESP8266 microcontroller to measure the presence and duration of individuals in front of a store display using a filtered ultrasonic distance-measurement algorithm. Timestamp synchronization is achieved via NTP, and the collected data, total visitors, arrival time, and observation duration, is transmitted to the cloud using the MQTT protocol. A Node-RED dashboard visualizes the information, enabling intuitive monitoring of customer behavior in physical retail environments. The results demonstrate that the system performs reliably, with consistent detection, stable wireless communication, and low-latency data visualization. The work shows the feasibility of combining low-cost hardware with lightweight IoT protocols to support decision-making in marketing and point-of-sale analytics.

Keywords

Internet of Things, MQTT, Ultrassonic Sensor, Display Monitoring

1 Introduction

The current consumer market is highly competitive, requiring companies to adopt effective strategies to attract customers to their points of sale. In this context, storefront displays play a strategic role, functioning not only as product exhibition elements but also as marketing tools capable of capturing attention and influencing consumer purchase interest [1]. Storefront monitoring in retail environments involves tracking customer behavior and evaluating the effectiveness of displays in increasing foot traffic and sales. This analysis is typically performed using technologies such as cameras that provide real-time data, heat maps of popular areas, and video analytics of shoppers' paths. By integrating these data with sales information, retailers can assess display effectiveness and make more informed decisions for future exhibitions [2, 3, 4].

Heat maps and other camera-based monitoring methods are effective for analyzing customer behavior and space occupancy; however, they involve high financial, computational, and energy costs, making them difficult to manage without specialized infrastructure. Cameras are expensive, require maintenance, and demand real-time image processing, resulting in large data storage requirements [5]. These costs can be reduced through more accessible technologies and by optimizing analysis methods using applications such as the Internet of Things (IoT) [6, 7]. For example, accelerometers can be used to detect object movement, or infrared sensors can identify the presence or absence of customers in front of a storefront, among other alternatives [8].

In the literature, there are already examples of using Internet of Things (IoT) sensors and alternative methods to video-based approaches for people counting and flow monitoring. For instance, in [9], the authors implemented an IoT project using a Passive Infrared (PIR) sensor for pedestrian counting and environmental monitoring, demonstrating technical feasibility. In [10], the authors present a review of detection and counting methods for indoor environments and discuss the advantages of non-invasive and low-cost approaches. Other studies, such as the use of IR-UWB (Impulse Radio Ultra-Wideband) sensors and binary PIR sensors [11, 12], demonstrate that it is possible to obtain accurate information about flow and presence without relying on cameras. These works reinforce the feasibility of IoT-based solutions for monitoring and counting people.

Within this context, the present work aims to contribute by proposing an IoT–based alternative using an ultrasonic sensor and an ESP8266 board to monitor customer behavior in storefront displays. The system detects individuals, records observation time (the duration of stay within the sensor coverage area), and logs arrival time, performing this monitoring in real time. The collected data are transmitted via the MQTT (Message Queuing Telemetry Transport) protocol to a Mosquitto broker and processed in the Node-RED environment, enabling visualization through an interactive dashboard with updated information, as well as access to the complete historical dataset of customers who observed the storefront, available for consultation and download.

The remainder of this article is organized as follows. Section 2 presents background concepts related to the context of this work. Section 3 introduces the architectural design of the proposed solution, while Section 4 describes the materials and methods employed. Finally, Section 5 examines the obtained results, and Section 6 presents the concluding remarks.

2 Background

This section presents fundamental topics and concepts that supported the development of the project.

2.1 Storefront Monitoring as a Marketing Strategy

According to [1], the retail sector is undergoing constant transformation, marked by the daily emergence of new companies seeking growth and market consolidation. To stand out in this competitive environment, organizations must adopt strategies that expand sales and strengthen their presence before the public. In this context, marketing plays an essential role, as it is responsible for creating and maintaining profitable relationships between brands and consumers. Within the marketing mix, the point of sale—also known as

"place"—becomes especially relevant, as it represents the physical space where products are presented and distributed.

Still, according to [1], storefront displays are among the most important resources at the point of sale, as they function as the first point of contact between the store and the customer. By attractively presenting products, they influence perceptions, arouse interest, and can directly impact the purchase decision-making process. Since social, cultural, psychological, and economic factors shape consumer behavior, understanding this impact becomes essential. Therefore, monitoring storefront displays by identifying peak observation periods and average attention time is a valuable strategy for improving marketing actions and optimizing commercial results.

2.2 Ultrasonic Sensor

An ultrasonic sensor is a device that uses a transducer to transmit and receive ultrasonic pulses that convey information about the proximity of an object. High-frequency sound waves reflect off surfaces, producing distinct echo patterns. Ultrasonic sensors operate by emitting sound waves at frequencies above the range of human hearing. The sensor transducer acts as both a transmitter and a receiver—similar to a microphone—sending and capturing ultrasonic signals [13]. Figure 1 illustrates the operating principle of an ultrasonic sensor.

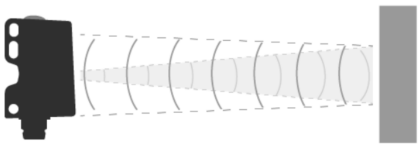

Figure 1: Ultrasonic sensor operation [14].

Ultrasonic sensing is reliable under any lighting conditions and can be used in both indoor and outdoor environments. Ultrasonic sensors can handle collision avoidance for robots and frequent movements, provided the motion is not excessively fast. Ultrasonic technology is so widely adopted that it can be safely implemented in sensing applications such as grain silos, water level sensors, distance measurement, people detection, drones, and vehicle detection [13].

An ultrasonic sensor calculates the distance to an object using the echo principle and the fundamental equation of motion. It emits a high-frequency sound wave and measures the time required for the wave to reach the object and return to the sensor after reflection. From this time interval, the device determines the distance traveled by the wave and, consequently, the position of the object [15]. The fundamental equation is described below in (1).

$$\mathrm{d} = \frac{\mathrm{v} \times \mathrm{t}}{2} \tag{1}$$

where d is the distance traveled, v is the speed of sound in the medium in which the sensor operates, typically air, and t is the total time taken by the sound pulse to travel from the sensor to the object and back after reflection. The division by two is required because the measured time corresponds to the round-trip travel of the pulse. To obtain the distance to the object, half of the traveled path must therefore be considered. Taking into account that the speed of sound in air is a constant equal to 343 m/s, (1) can be rewritten as (2).

$$d = 171.5 \cdot t \quad [m] \tag{2}$$

The sensor selected for this work was the HC-SR04 ultrasonic sensor. This sensor is shown in Figure 2 and is described in detail later in Section 3.

Figure 2: HC-SR04 ultrasonic sensor board [16].

2.3 MQTT Protocol

In traditional network communication, clients request resources or data from a server, which processes the request and sends a response, enabling direct communication between the entities. In contrast, MQTT uses a publish/subscribe pattern to decouple the message sender, or publisher, from the message recipient, known as the subscriber [17]. A third component, called a message broker, manages communication between publishers and subscribers. The broker's role is to filter all messages received from publishers and distribute them appropriately to the interested subscribers.

Currently, MQTT is used across several sectors, including automotive, telecommunications, and oil and gas. Its widespread adoption in multiple domains, including IoT applications [18], is due to its lightweight and efficient nature, bidirectional communication capabilities, scalability to connect millions of devices, and security features. MQTT supports message encryption via TLS (Transport Layer Security) and client authentication via modern authentication protocols [17].

2.4 Mosquitto

Mosquitto is a broker, that is, an intermediary that mediates communication between devices using network protocols. It is used within the MQTT protocol to enable devices to communicate with each other and operate in an automated manner [19]. Mosquitto manages communication between publisher and subscriber devices via topics: it receives messages from publishers, filters them, and delivers them to subscribers interested in specific topics. This approach enables efficient and scalable communication, even in networks with low bandwidth [20, 21].

2.5 Node-RED

This work also employs Node-RED, a low-code visual programming tool designed for integrating hardware devices, APIs, and online services. The platform enables the intuitive creation of workflows by interconnecting nodes (functional blocks) within a web browser–based environment [22, 23]. Node-RED operates with the MQTT protocol as follows: devices such as ESP32 and Raspberry Pi

publish data as messages to an MQTT broker, which acts as a central server (e.g., Mosquitto), using specific topics. The broker functions as a hub, receiving and distributing messages to subscribing devices. Finally, Node-RED uses MQTT input and output nodes to connect to the broker, allowing it to both subscribe to topics and publish messages [22].

- *Devices and Sensors*: Devices such as ESP32 and Raspberry Pi send data (e.g., temperature, status) as messages to an MQTT broker (central server, such as Mosquitto) using specific topics.
- *MQTT Broker*: Acts as a hub, receiving and distributing messages to interested subscribers.
- *Node-RED*: Uses MQTT input and output nodes to connect to the broker, either listening to (subscribing) topics or publishing messages.
- *Workflows*: In Node-RED, visual workflows are created in which an MQTT node receives data, function nodes or other nodes process, transform, or filter the data, and additional nodes can send commands back or trigger actions (e.g., turning devices on/off or sending alerts).

3 System Architecture

This work describes a storefront monitoring system capable of detecting customer presence, recording the duration of their display observation, and transmitting this information in real time to a dashboard via MQTT. The system integrates an ultrasonic sensor, an ESP8266 microcontroller, wireless connectivity, and the MQTT protocol, enabling both real-time monitoring and historical data storage through Mosquitto and Node-RED. Power is supplied by a personal computer (Galaxy Book4 360). An overview of the project is shown in Figure 3.

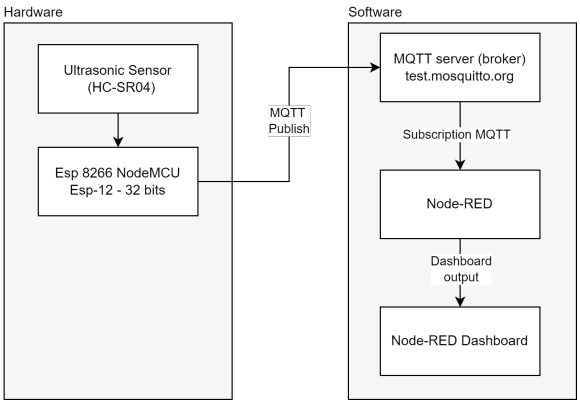

Figure 3: System overview.

3.1 Hardware Platform

The proposed system was developed using the ESP8266 as the main microcontroller, coupled with an ultrasonic sensor (HC-SR04) to detect the presence of individuals and measure their observation time. The selection of these components is based on the need for a low-cost solution, easy integration with wireless networks, and sufficient processing capability to perform real-time readings and transmit data via IoT protocols. The role of each component of the hardware platform used is detailed below.

3.1.1 ESP8266 Microcontroller. The ESP8266 NodeMCU ESP-12 board (Figure 4) is a low-cost microcontroller with an integrated Wi-Fi chip, manufactured by Espressif Systems, which enables the development of IoT devices. It features a 32-bit CPU, support for the IEEE 802.11 b/g/n wireless networking standard, and multiple communication interfaces, including GPIO, SPI, and I^2C, enabling connectivity with sensors, actuators, and other peripherals. The board provides 11 digital I/O pins and one analog input, operates at 3.3 VDC, and accepts power supply voltages of 6–12 VDC. It includes 4 MB of flash memory and 64 kB of SRAM. The ESP8266 is widely adopted due to the ease of direct programming via the Arduino IDE and can also function as a complementary module to another microcontroller, offering flexibility in project architecture [24, 25].

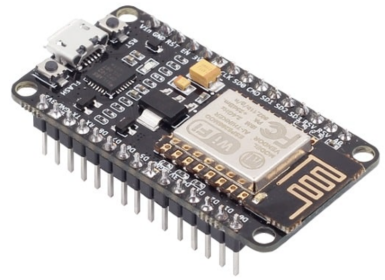

Figure 4: ESP8266 NodeMCU ESP-12 board [25].

In this work, the ESP8266 board performs three main functions: data acquisition from the ultrasonic sensor, processing of the readings to identify presence as well as the start and end of the observation period, and finally, data transmission via the MQTT protocol to Mosquitto and Node-RED, enabling visualization through dashboards and access to historical customer data.

3.1.2 Ultrasonic Sensor. In the proposed project, the HC-SR04 sensor (previously illustrated in Figure 2) is responsible for detecting the presence of individuals in front of the storefront. Its operation is based on the emission of high-frequency ultrasonic sound pulses and the measurement of the time required for the echo to return after reflecting off an object. The HC-SR04 sensor provides distance measurements ranging from 2 cm to 4 m with adequate accuracy and has a low cost (ranging from USD 0.10 to USD 0.85, FOB—Free On Board—price per unit). This sensor module integrates both a transmitter and a receiver and features four pins (VCC, TRIG, ECHO, and GND) for operation, two of which are used for control: the TRIG (*trigger*) pin initiates the pulse, and the ECHO pin signals the return of the pulse (i.e., the arrival of the reflected ultrasonic wave) [16]. Its operating voltage is 5 VDC.

In this work, the sensor is positioned so that its detection range covers the area in which customers observe the storefront, with a reliable minimum distance of approximately 2 cm and a maximum configured distance of 200 cm. It provides periodic readings to the ESP8266, which are processed to determine the entry of an

individual, the total observation time, and the exit of the individual from the monitored area.

3.1.3 Connectivity Resources. The platform leverages the ESP8266's integrated Wi-Fi capability (IEEE 802.11 b/g/n in the 2.4 GHz band) to establish communication with an MQTT broker, in this case Mosquitto, which acts as an intermediary between the microcontroller and the visualization interface implemented in Node-RED. The MQTT protocol is lightweight and efficient, making it particularly suitable for IoT applications that require real-time data exchange. It enables asynchronous message publishing while ensuring low bandwidth and processing consumption, which is essential for resource-constrained microcontrollers, the focus of this work.

The MQTT architecture allows different types of data to be transmitted using separate topics. In the proposed system, distinct topics are published for the number of detected individuals, arrival time (start of detection), and observation interval, enabling granular monitoring and organized data handling for subsequent analysis [26].

The ESP8266 also performs periodic queries to Network Time Protocol (NTP) servers to keep its internal clock synchronized. This approach ensures that customer arrival and departure records remain accurate, regardless of system restarts or minor timing drifts of the microcontroller.

3.1.4 Hardware Design. The hardware connection diagram for the platform was developed using Tinkercad and diagrams.net (a.k.a., draw.io) and is illustrated in Figure 5.

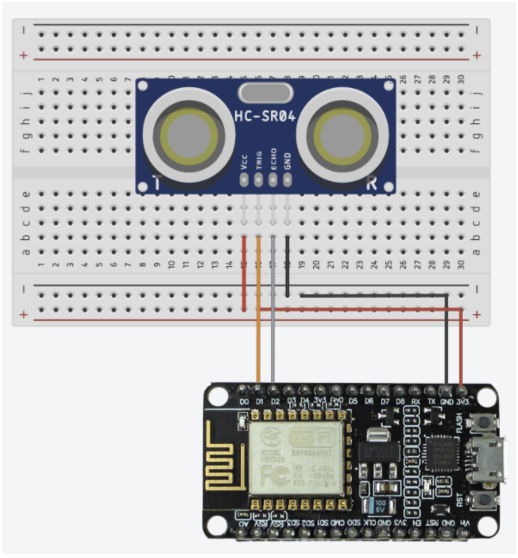

Figure 5: Hardware design.

The ultrasonic sensor is powered by the ESP8266's 3.3 V pin (red wire) and grounded (black wire). The TRIG pin (orange wire) is connected to GPIO D1 (digital pin 5), while the ECHO pin (gray wire) is connected to GPIO D2 (digital pin 4). These GPIOs were selected because they are available for digital use, allow fast sensor readings, and are compatible with the `NewPing` library, ensuring that the transmitted and received pulse timings are correctly interpreted by the microcontroller.

The sensor operates at 3.3 V instead of 5 V to ensure compatibility with the ESP8266 logic levels, thereby preventing potential damage to the microcontroller, as its digital pins do not tolerate voltages above 3.6 V. This configuration enables the detection of individuals in front of the storefront and provides more accurate distance measurements, enabling reliable data for real-time monitoring.

3.2 Software Development

The system software was developed using the Arduino IDE, targeting the NodeMCU ESP8266 1.0 board. The program performs the main functions illustrated in the flowchart shown in Figure 6.

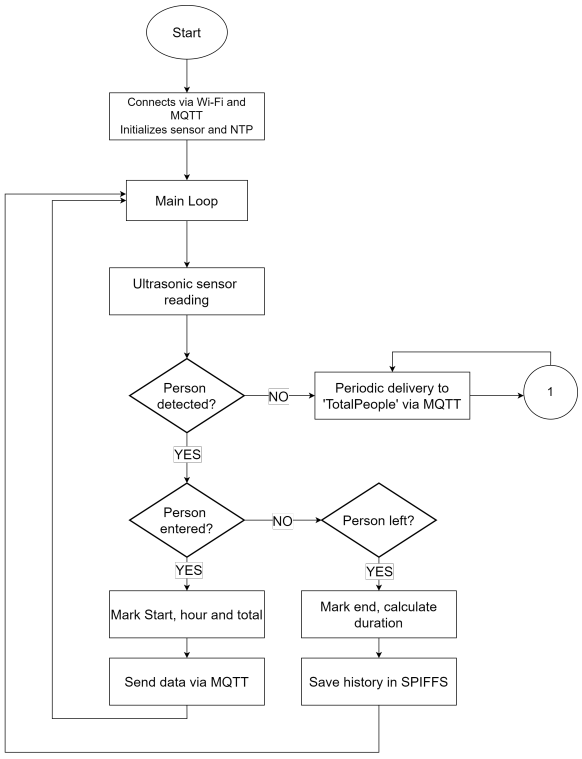

Figure 6: Embedded software design.

The system software was developed to manage the entire operation of the smart storefront automatically. Initially, the ESP8266 board configures its Wi-Fi interface and connects to a previously configured mobile hotspot, ensuring stable and continuous wireless communication. The microcontroller authenticates with the network, obtains an IP address via DHCP, and continuously monitors the connection status to ensure that the remainder of the system is initialized only after the network has been successfully established.

Subsequently, the ESP8266 establishes a connection with the MQTT broker using the `PubSubClient` library, enabling efficient data publishing to specific topics. During each cycle of the main loop, the HC-SR04 sensor periodically measures the distance to individuals in front of the storefront by emitting ultrasonic pulses and calculating the echo return time to determine proximity. Based on these measurements, the system automatically identifies customer entry and exit events, recording the exact duration during which

any individual remains in view of the storefront. It is important to note that the system does not count the number of individuals simultaneously in front of the storefront, but rather whether at least one person was present during a given time interval.

The collected data, including the total number of detections, arrival time, and dwell time, are published to distinct MQTT topics, allowing the dashboard implemented in Node-RED to consume this information in real time and display dynamic visualizations for immediate analysis. In parallel, all information is stored locally in the ESP8266 SPIFFS file system, ensuring a complete historical record that can be accessed later for behavioral analysis and strategic decision-making. The circle labeled with number 1 in Figure 6 represents a custom algorithm that maintains the detection state to ensure accurate timestamps and the creation of a reliable historical dataset. The corresponding pseudocode is presented below.

The MQTT configuration on the device enables real-time transmission of data collected by the ultrasonic sensor via the Mosquitto broker, as illustrated in Figure 7 (the value remains unchanged until a new detection occurs). For this project, the public broker "https://test.mosquitto.org" was used on port 1883, ensuring compatibility with a wide range of IoT data visualization and analysis tools.

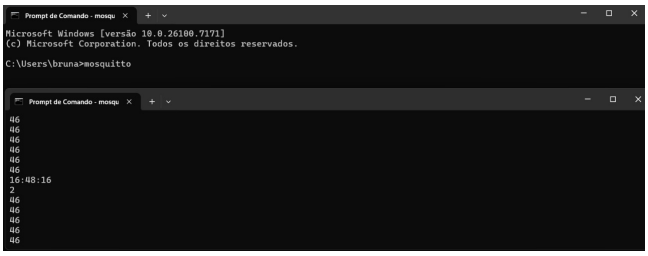

Figure 7: Mosquitto terminal (*MS Windows v. 10.0.26100.7171 – Portuguese version*).

The MQTT topics were standardized and organized using the following structure:

(1) `display/totalPeople`
(2) `display/hour`
(3) `display/duration`

The `totalPeople` topic publishes the total number of people detected in front of the storefront (i.e., the total number of detection events). Each time a new detection occurs, this value is updated and transmitted via MQTT. The `hour` topic publishes the exact time at which a detection begins (i.e., when a person is first detected). This approach enables recording when an individual first observes the storefront and can be used for temporal analyses of visitor flow. Finally, the `duration` topic publishes the dwell time, in seconds, that a person remains in front of the storefront during a detection period. This value is automatically calculated from the difference between the entry and exit timestamps, allowing the analysis of customer engagement with the storefront. In scenarios involving the monitoring of multiple storefronts, individual topics can be defined for each one (e.g., `diplay01/...`, `display02/...`, etc.).

Algorithm 1 Main Flow of the Monitoring System

START
Configure Serial and SPIFFS
Connect to Wi-Fi network
Synchronize time via NTP
Configure MQTT broker
    while *System is running* **do**
    end
    MAIN LOOP **if** *MQTT not connected* **then**
    end
    Connect to the MQTT broker

Read distance from ultrasonic sensor (filtered)
    if *distance indicates presence of a person* ***and*** *person was not previously present* **then**
    end
    Person entered
Record entry time
Increment total number of people
Publish total and arrival time via MQTT
person was previously present
Person left
Calculate dwell time
Save history to SPIFFS
Publish total, arrival time, and duration via MQTT

Periodically publish total number of people (e.g., every 0.1 s)
Wait for next cycle

The MQTT publishing string was configured to transmit only raw data (numerical values), allowing the Node-RED dashboard to interpret and display the information accurately. To ensure continuous and responsive updates, data transmission occurs at a minimum interval of 0.1 seconds, providing near real-time integration of presence information, arrival time, and observation duration for individuals in front of the storefront.

3.3 Integration with Node-RED for Monitoring

The Node-RED flow (Figure 8) operates as a complete monitoring system connected to the MQTT broker. Each message received from the subscribed topics is processed by a function node that stores the corresponding value in the flow context (`flow.set`). These data are then forwarded to dashboard components, where they are displayed with customized formatting, including font size, color, and visual alignment. The duration value also feeds a historical logging mechanism: for each new entry, the flow appends an item to an observation array, limiting the array to 20 records. This historical data is then converted into a format compatible with the `ui_chart` node, which displays a bar chart illustrating the evolution of observed dwell time.

In addition to data presentation, the flow provides tools for interaction and historical data management within the dashboard (Figure 9). A `Clear History` button allows the user to completely reset the stored data, updating the chart to an empty state. Another button, `Export CSV`, triggers a function node that iterates over the stored

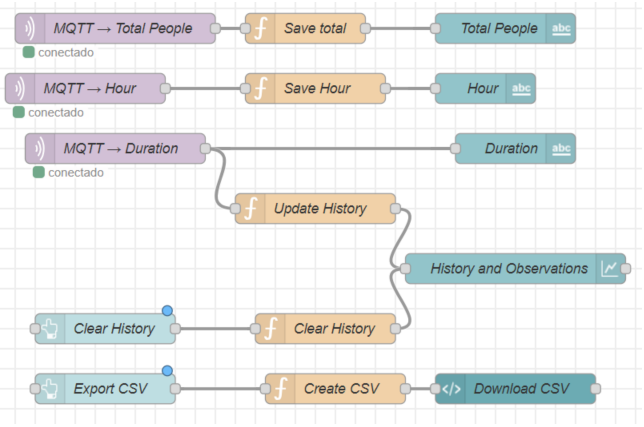

Figure 8: Node-RED flow.

history and writes the data to a CSV file containing the observed arrival times and dwell durations.

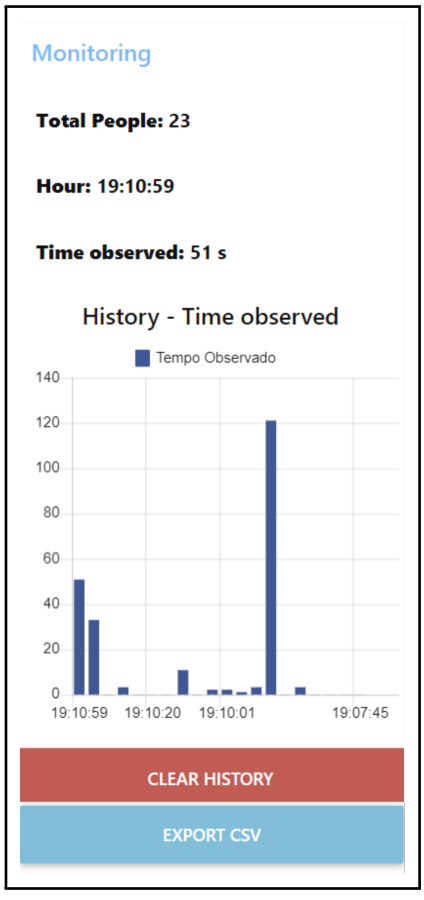

Figure 9: Node-RED dashboard.

Then, this file is automatically made available for download via a `ui_template` component, which generates a dynamic link and initiates the download. In this way, the flow not only monitors the values received via MQTT but also stores, visualizes, organizes, and exports information in an integrated manner, resulting in a functional and interactive IoT dashboard.

4 Materials and Methods

4.1 Materials

In this work, we used the following hardware components and software tools:

(1) Hardware:
- Galaxy Book 4 360 PC.
- ESP8266 Microcontroller.
- HC-SR04 Ultrasonic Sensor.
- Micro USB power cable.

(2) Software:
- Tinkercad: free web application for 3D design, electronics, and programming.
- Broker Mosquitto v2.0.22.
- Node-RED v4.1.0.
- ARDUINO IDE v2.3.6.

4.2 Methods

The system development was carried out using Arduino IDE version 2.3.6, configured for the NodeMCU ESP-12E Module (ESP8266) microcontroller, connected via COM5. The software was developed in C, with support from libraries for Wi-Fi communication, ultrasonic sensor data acquisition, and MQTT data transmission. All libraries were installed using the Arduino IDE Library Manager. The project configuration details are presented below.

(1) Tools and environment components:
- ESP8266 Boards platform (added via the Boards Manager using the URL: http://arduino.esp8266.com/stable/package_esp8266com_index.json).
- Selected board: NodeMCU 1.0 (ESP-12E Module).
- Communication port: COM5.

(2) Libraries used:
- `FS.h`: file handling in the SPIFFS file system.
- `NewPing.h`: optimized reading of the HC-SR04 ultrasonic sensor.
- `TimeLib.h`: time manipulation and timestamp formatting.
- `ESP8266WiFi.h`: ESP8266 connection to the Wi-Fi network.
- `WiFiUdp.h`: UDP communication, used for NTP synchronization.
- `PubSubClient.h`: implementation of the MQTT protocol for data transmission.

(3) Wi-Fi credentials: the microcontroller is configured to connect to the defined Wi-Fi network automatically.

Algorithm 2 Wi-Fi Configuration

```
1: const char* WIFI_SSID = "Bruna's iPhone"
2: const char* WIFI_PASS = "******"
```

(4) MQTT configuration: the system publishes data to the public Mosquitto broker.

Algorithm 3 MQTT Configuration

```
MQTT Configuration
const char* MQTT_SERVER = "test.mosquitto.org";
const int MQTT_PORT = 1883;
```

(5) The HC-SR04 sensor was configured on the following pins.

Algorithm 4 Ultrasonic Sensor Configuration

```
Define ultrasonic sensor pins and limits
#define TRIGGER D1
#define ECHO D2
#define MAX_DIST 200          ▷ maximum distance in cm
```

The reading utilizes the NewPing library, which provides faster and more stable measurements.

Algorithm 5 Ultrasonic Sensor Configuration

```
Define ultrasonic sensor pins and limits
#define TRIGGER D1
#define ECHO D2
#define MAX_DIST 200          ▷ maximum distance in cm
```

(6) NTP Synchronization (Time Server): UDP communication is used to obtain the actual time via NTP.

Algorithm 6 NTP Synchronization

```
Define NTP server and time zone
const char* NTP_SERVER = "pool.ntp.org";
const unsigned long TZ_OFFSET = -3L * 3600L;   ▷ UTC-3 time zone
```

This allows for proper registration:

- arrival time.
- departure time.
- duration of stay.

(7) General Code Structure:

- Wi-Fi initialization.
- NTP synchronization.
- filtered reading of the ultrasonic sensor.
- identification of people entering and exiting.
- duration calculation.
- data transmission via MQTT.
- local logging (SPIFFS – SPI Flash File System).

The beginning of the code contains all essential environment configurations:

Algorithm 7 Required Libraries Inclusion

```
Include required libraries
#include <FS.h>                    ▷ SPIFFS file system
#include <NewPing.h>               ▷ Ultrasonic sensor handling
#include <TimeLib.h>               ▷ Time manipulation
#include <ESP8266WiFi.h>           ▷ Wi-Fi connectivity
#include <WiFiUdp.h>               ▷ UDP communication for NTP
#include <PubSubClient.h>          ▷ MQTT protocol
```

5 Experimental Results

The experimental setup (prototype) is illustrated in Figure 10.

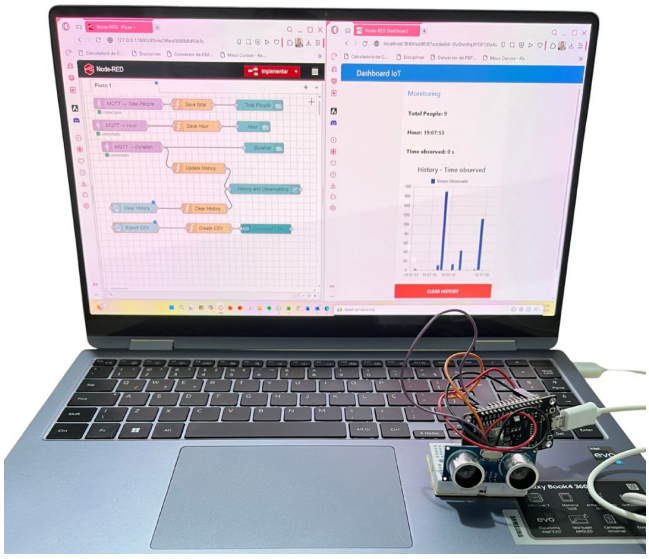

Figure 10: Experimental setup.

5.1 Presence Detection and Data Collection

In the experiments, the implemented algorithm detected individuals based on variations in distance measurements from the ultrasonic sensor. The applied filtering techniques and the 200 cm distance threshold proved adequate for identifying actual approaches, thereby reducing false readings caused by environmental noise. For each detection event, the system automatically recorded the arrival time, the dwell duration in seconds, and the cumulative total number of detected individuals. Communication with the NTP server ensured temporal consistency in the records, eliminating time discrepancies.

5.2 MQTT Transmission

Data transmission to the MQTT broker remained stable throughout the tests. All three topics were updated correctly, and the periodic update rate (0.1 s) operated without significant packet loss, demonstrating that the ESP8266 can maintain a continuous connection to the public broker.

5.3 Dashboard Visualization

The transmitted data were displayed in real time on the Node-RED dashboard. The interface allowed monitoring the increase in the total number of individuals detection, visualizing individual arrival times, and analyzing the average dwell duration per person. This visualization facilitated the interpretation of flow behavior for a hypothetical monitored storefront, enabling the identification of periods with higher activity levels.

6 Conclusion

The developed project achieved its objective of creating a functional presence-monitoring system using an ultrasonic sensor integrated with an ESP8266, with data transmitted via MQTT and visualization in Node-RED. The solution proved effective in detecting individuals, recording arrival times, and estimating dwell durations, providing relevant metrics for behavioral analysis in retail environments.

The use of the ESP8266 board enabled the implementation of a low-cost, network-connected device capable of autonomous operation. Integration with NTP services ensured the temporal accuracy required for report generation, while MQTT ensured lightweight, efficient communication with the dashboard. Based on the results, the system is suitable for storefront monitoring, customer flow analysis, and consumer behavior studies.

It is important to note that the proposed solution cannot simultaneously identify the number of individuals present in front of a storefront (as detected by the sensor), but only the presence of one or more individuals and the duration of their presence. For more accurate counting, a video-based camera system would be required to identify individuals in the scene; however, such a solution entails significantly higher costs.

As future work, a field evaluation is planned by installing the sensor in an actual storefront to collect real-world data. For validation purposes, video recordings of the same testing period will be conducted to compare results. In addition, future improvements may include cloud-based historical data analysis, integration with multiple sensors, or the application of machine learning techniques to enhance classification and understanding of dwell-time patterns.

Acknowledgments

This work was supported by the Brazilian National Coordination of Superior Level Staff Improvement – CAPES (Financial Code 001), the Foundation for Support of Research and Innovation, Santa Catarina – FAPESC (Grant 2025TR001570), and the Brazilian National Council for Scientific and Technological Development – CNPq (Grants 408641/2023-1 and 306478/2025-0). During the preparation of this work, the authors used Grammarly and ChatGPT to revise grammar and enhance clarity and conciseness in specific sections of the text. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the article's content. All conceptual, methodological, analytical, and interpretative content remains entirely the authors' own work.

References

[1] Lucas Gabriel de Souza Galvão. "As vitrines como agentes influenciadores no comportamento do consumidor e processo decisório de compra". In: *Revista de Gestão e Secretariado* 14.4 (2023), pp. 4749–4761.

[2] J Lilly. "The Effects of Store Window Display on Customers Perception and Attitude in Retail Clothing Outlets". In: *Journal of Global Economy* 6.1 (2010), pp. 16–20.

[3] Retail Sensing. *Footfall: Testing Shop Window Displays*. https://www.retailsensing.com/people-counting/testing-shop-window-displays/. Accessed on: 25 abr. 2025. Feb. 2019.

[4] Alexandra Sheehan. *Window Displays: How to Make Your Storefront Stand Out*. https://www.shopify.com/uk/blog/window-displays. Accessed on: 26 nov. 2025. Sept. 2025.

[5] DINO. *Sensores e inteligência artificial redefinem a experiência nos supermercados*. https://www.em.com.br/mundo-corporativo/2025/09/7251815-sensores-e-inteligencia-artificial-redefinem-a-experiencia-nos-supermercados.html. Accessed on: 26 nov. 2025. Sept. 2025.

[6] Antonio Carlos Cob-Parro et al. "Smart video surveillance system based on edge computing". In: *Sensors* 21.9 (2021), p. 2958.

[7] Felipe Viel et al. "Internet of Things: Concepts, architectures and technologies". In: *2018 13th IEEE International Conference on Industry Applications (INDUSCON)*. IEEE. 2018, pp. 909–916.

[8] Hyunwoo Hwangbo, Yang Sok Kim, and Kyung Jin Cha. "Use of the smart store for persuasive marketing and immersive customer experiences: A case study of Korean apparel enterprise". In: *Mobile Information Systems* 2017.1 (2017), p. 4738340.

[9] Fowzia Akhter et al. "IoT enabled intelligent sensor node for smart city: pedestrian counting and ambient monitoring". In: *Sensors* 19.15 (2019), p. 3374.

[10] Sara Comai et al. "Review of methods and technologies to detect, count and identify people in indoor environments". In: *Internet of Things* 29 (2025), p. 101466.

[11] Azad Shokrollahi et al. "Non-Invasive People Counting in Smart Buildings: Employing Machine Learning with Binary PIR Sensors". In: *17th International Conference on Agents and Artificial Intelligence, Porto, Portugal, February 23-25, 2025*. INSTICC. 2025, pp. 394–405.

[12] Ange Joel Nounga Njanda et al. "People counting using IR-UWB radar sensors and machine learning techniques". In: *Systems and Soft Computing* 6 (2024), p. 200095.

[13] MaxBotix Inc. *How Ultrasonic Sensors Work*. https://maxbotix.com/blogs/blog/how-ultrasonic-sensors-work. Accessed on: 30 nov. 2025.

[14] *Guida alla Scelta Sensori Automazione*. Mef. 2025.

[15] Saddam. *Arduino Ultrasonic Sensor Based Distance Measurement*. https://circuitdigest.com/microcontroller-projects/arduino-ultrasonic-sensor-based-distance-measurement. Accessed on: 30 nov. 2025. 2015.

[16] MakerHero. *Sensor de Distância Ultrassônico HC-SR04*. https://www.makerhero.com/produto/sensor-de-distancia-ultrassonico-hc-sr04/. Accessed on: 26 nov. 2025.

[17] AWS. *What is MQTT?* https://aws.amazon.com/what-is/mqtt/. [Online; accessed 24-Dec-2025]. 2025.

[18] Vedat Ozan Oner. *Developing IoT Projects with ESP32: Unlock the full Potential of ESP32 in IoT development to create production-grade smart devices*. Packt Publishing Ltd, 2023.

[19] EngProcess. *Mosquitto: broker MQTT*. https://engprocess.com.br/mosquitto/. Accessed on: 2025. 2018. URL: https://engprocess.com.br/mosquitto/.

[20] IFPR – Instituto Federal do Paraná. *Mosquitto*. https://wiki.foz.ifpr.edu.br/wiki/index.php/Mosquitto. Accessed on: 2025. 2020. URL: https://wiki.foz.ifpr.edu.br/wiki/index.php/Mosquitto.

[21] Energia Automação. *Protocolo MQTT: o que é, como funciona e suas vantagens*. https://energiaautomacao.com.br/artigo/protocolo-mqtt-o-que-e-como-funciona-e-suas-vantagens. Accessed on: 2025. 2022. URL: https://energiaautomacao.com.br/artigo/protocolo-mqtt-o-que-e-como-funciona-e-suas-vantagens.

[22] Node-RED. *Node-RED: Low-code programming for event-driven applications*. https://nodered.org. Accessed on: 2025. 2025. URL: https://nodered.org.

[23] FlowFuse. *Node-RED: programação low-code para aplicações orientadas a eventos*. https://flowfuse.com/node-red/. Accessed on: 2025. 2025. URL: https://flowfuse.com/node-red/.

[24] Marcio Mello. *Como programar ESP8266? Veja dicas de como usar o microcontrolador*. https://victorvision.com.br/blog/como-programar-esp8266/. Accessed on: 26 nov. 2025. Jan. 2024.

[25] MakerHero. *Módulo WiFi ESP8266 NodeMCU ESP-12*. https://www.makerhero.com/produto/modulo-wifi-esp8266-nodemcu-esp-12/. Accessed on: 26 nov. 2025.

[26] HiveMQ. *MQTT Essentials – All Core Concepts Explained*. https://www.hivemq.com/mqtt/. Accessed on: 26 nov. 2025.