ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

UNIDAD DE TITULACIÓN

DISEÑO E IMPLEMENTACIÓN DE UN DISPOSITIVO IOT Y PLATAFORMA WEB PARA DISPENSO Y GESTIÓN EN MÁQUINA DE BEBIDAS CALIENTES EN ESTABLECIMIENTOS COMERCIALES.

DISPOSITIVO IOT PARA MEDIDA DEL FLUJO DE BEBIDAS DISPENSADAS

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN CIENCIAS DE LA COMPUTACIÓN

EDUARDO JAVIER AVILÉS FUERTES

eduardo.aviles@epn.edu.ec

DIRECTORA: DIANA CECILIA YACCHIREMA VARGAS. PhD.

diana.yacchirema@epn.edu.ec

DMQ, julio de 2024

CERTIFICACIONES

Yo, EDUARDO JAVIER AVILÉS FUERTES declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

_____

EDUARDO JAVIER AVILÉS FUERTES

Certifico que el presente trabajo de integración curricular fue desarrollado por EDUARDO JAVIER AVILÉS FUERTES, bajo mi supervisión.

_____

DIANA CECILIA YACCHIREMA VARGAS. PhD.

DIRECTORA

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Eduardo Javier Avilés Fuertes

Diana Cecilia Yacchirema Vargas

DEDICATORIA

A mi padre Holguer, quien que con su incansable dedicación y apoyo ha sido un pilar fundamental para mi éxito. Gracias por siempre confiar y creer en mí, por recordarme siempre lo importante de la educación en la vida, incluso cuando se atraviesa momentos difíciles. Sin tu guía y ejemplo no habría llegado hasta aquí.

A mi madre Marisol, quien aún que ya no esté físicamente presente, su memoria, su amor y enseñanzas continúan iluminando mis pasos en este camino. Este logro también es tuyo, pues tu espíritu ha sido fuerza e inspiración en mi vida.

A mi abuelita Mariana, un ser especial que sé que desde el cielo cuida de mí, su ejemplo de mujer trabajadora y su amor han sido de total inspiración.

A mi hija Nathaly, mi más grande motivación y la luz de mi camino. Eres y serás la razón por la que sigo adelante. Cada paso que he recorrido ha sido con la buena esperanza de brindarte un mejor futuro y también demostrarte que con esfuerzo y dedicación es posible alcanzar el éxito.

Con todo mi amor y total gratitud,

Eduardo Javier.

AGRADECIMIENTO

Quiero expresar mi total gratitud a Dios, mi familia y amigos, por su constante apoyo durante estos años, a mi tía Samia Fuertes, quien siempre me ha brindado su apoyo incondicional.

A la tutora de la Tesis Dra. Diana Cecilia Yacchirema Vargas, PhD., por su invaluable asesoría a lo largo de este proyecto. Su conocimiento, orientación y sobre todo su paciencia han sido cruciales para la culminación de este trabajo de investigación.

A los docentes de la Escuela Politécnica Nacional, quienes en las aulas han compartido conmigo su conocimiento y experiencia, y han sido fuente de inspiración para seguir por el camino del aprendizaje y la investigación.

Con sincero agradecimiento,

Eduardo Javier.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El sector comercial de servicios de alimentación y bebidas ha venido experimentando un notable crecimiento y una transformación digital relevante en los últimos años, reflejado en la creciente demanda del consumo de bebidas calientes en locales comerciales; la medición de consumo de estas bebidas se realiza manualmente, dispensando cantidades establecidas por los envases y cobrando según el volumen dispensado. En este contexto, este proyecto presenta una solución innovadora para optimizar el proceso de dispensado y el registro de la cantidad de bebida que el cliente consume.

La solución se la realiza mediante la implementación de un dispositivo IoT destinado a registrar las medidas de consumo de bebidas calientes populares como café, té y leche en autoservicios comerciales. El desarrollo del proyecto siguió una metodología de diseño centrado en el usuario, que incluyó análisis contextual, definición de requisitos, diseño e implementación del dispositivo IoT, y la evaluación de este en un ambiente simulado.

El diseño e implementación del dispositivo IoT incluye la selección de la arquitectura IoT, la configuración hardware (sensor de flujo, sensor RFID y placas NodeMCU ESP8266), el desarrollo de código necesario para su integración, la configuración de una red inalámbrica Wi-Fi y la configuración de un bróker MQTT para la gestión de datos.

Para la constatación de resultados de la medición de bebidas calientes efectuadas por el dispositivo IoT, se lo realizó mediante la simulación de un autoservicio de bebidas en un entorno controlado.

PALABRAS CLAVE: IoT, dispositivo IoT, medición de flujo, RFID, autoservicio de bebidas, NodeMCU, ESP8266, bróker MQTT.

ABSTRACT

The commercial food and beverage services sector has been experiencing notable growth and a relevant digital transformation in recent years, reflected in the growing demand for the consumption of hot drinks in commercial premises; The measurement of the consumption of these drinks is carried out manually, dispensing quantities established by the containers and charging based on the volume dispensed. In this context, this project presents an innovative solution to optimize the dispensing process and the recording of the amount of drink consumed by the customer.

The solution is carried out by implementing an IoT device intended to record consumption measurements of popular hot beverages such as coffee, tea and milk in commercial self-services. The development of the project followed a user-centered design methodology, which included contextual analysis, requirements definition, design and implementation of the IoT device and its evaluation in a simulated environment.

The design and implementation of the IoT device includes the selection of the IoT architecture, the hardware configuration (flow sensor, RFID sensor and ESP8266 NodeMCU boards), the development of the code necessary for its integration, the configuration of a Wi-Fi wireless network . and configure an MQTT broker for data management.

To verify the results of the measurement of hot drinks carried out by the IoT device, it was carried out by simulating a self-service of drinks in a controlled environment.

KEYWORDS: IoT, IoT device, Flow Measurement, RFID, Self-Service Beverages, NodeMCU, ESP8266, MQTT Broker.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

A raíz de la salida de la pandemia sanitaria, que años atrás enfrentamos, los diferentes negocios dentro del ámbito nacional han experimentado un alza en las ventas de productos y servicios que ofrecen. Uno de los sectores que más prosperidad ha mostrado es el negocio referente al consumo de bebidas, tanto alcohólicas, energizantes, de hidratación, así como el de bebidas comerciales calientes. En consecuencia de esto y gracias a los avances de la tecnología en sistemas de automatización, control e informáticos, los negocios han optado por incorporar tecnología con el objetivo de facilitar sus procesos y ser más productivos y competitivos dentro del mercado.

Este proyecto está diseñado para mejorar la gestión de la dispensación de bebidas calientes en autoservicios de establecimientos comerciales. Con esto se busca automatizar las funciones de dispenso de líquidos contenidos en los autoservicios y simplificar el proceso de la venta de las bebidas comerciales.

Con la finalidad de aportar en la mejora al proceso, este componente del proyecto consiste en el diseño y la implementación de un dispositivo de Internet de las Cosas (IoT, *Internet of Things* por sus siglas en inglés) que permita medir el flujo de bebidas calientes en los autoservicios de los establecimientos comerciales. La configuración del dispositivo consta de dos placas base, una placa está conectada a un sensor de flujo, mientras que la otra placa está conectada al un módulo RFID. El sensor de flujo permitirá tomar las medidas de flujo de las bebidas que pasan a través del sensor, mientras que el módulo lector RFID se empleará para la identificación de la persona que está realizando la compra de la bebida por medio de un TAG identificador. Los mencionados sensores de flujo y RFID estarán conectados a dos placas ESP8266 NodeMCU V1.0 ESP-12E de manera independiente, tal que, gestionarán la adquisición de datos y la comunicación inalámbrica por medio de una tecnología de red Wi-Fi, para que posteriormente mediante un bróker se realice el respectivo tratamiento de datos ya recopilados.

Finalmente, después del proceso descrito anteriormente, los datos recopilados serán utilizados para gestionar y presentar información que facilite el proceso de venta de diversas bebidas comerciales. Esto se realizará en el módulo complementario del proyecto de tesis.

1.1 Objetivo general

Diseñar e implementar un dispositivo IoT que permita sensar el flujo dispensado por los consumidores en un autoservicio de bebidas calientes comerciales (tales como, café, té, leche, capuchino, etc.).

1.2 Objetivos específicos

1. Adaptar y aplicar la metodología de diseño centrado en el usuario para el diseño e implementación del dispositivo IoT.

2. Seleccionar la arquitectura IoT, tomando en consideración la capa de percepción y capa de red para el diseño del dispositivo IoT.

3. Diseñar el dispositivo IoT en función de las capas consideradas dentro la arquitectura seleccionada y considerando la metodología de diseño centrado en el usuario.

4. Seleccionar y configurar los sensores adecuados para medir el flujo de las bebidas calientes en el autoservicio, asegurando su precisión y fiabilidad en diferentes condiciones de uso.

5. Ensamblar el hardware, según el diseño del dispositivo IoT, incluyendo la integración de los sensores, la electrónica de control y la conectividad necesaria entre los componentes.

6. Desarrollar e implementar el código correspondiente destinado al ensamble a nivel de software (código fuente) de las placas y sensores.

7. Configurar un bróker para recibir y almacenar los datos de flujo y los identificadores de los TAGs correspondientes a los sensores y placas, asegurando la disponibilidad de estos.

8. Establecer la conexión necesaria entre las placas, sensores y el bróker para el correcto funcionamiento del dispositivo IoT.

9. Realizar pruebas de funcionamiento del dispositivo IoT, simulando el uso de un autoservicio de bebidas calientes.

1.3 Alcance

Al ser considerado como producto el diseño e implementación del dispositivo IoT, el alcance se basa en la metodología de Diseño Centrado en el Usuario (UCD). Para el desarrollo de este componente el alcance lo adaptamos en cuatro etapas descritas en la Figura 1.

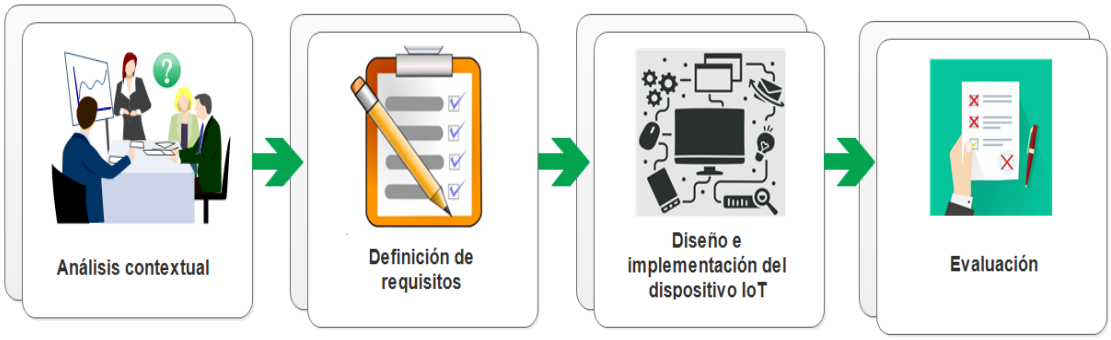

Figura 1. Etapas de la metodología UCD

1. Análisis contextual:
    - Definición de los objetivos del componente: Los objetivos generales y específicos del componente se los definió en la sección 1.1 y 1.2 de este documento.
    - Identificación de los usuarios finales y sus necesidades: Identificar tanto el tipo de usuario como sus necesidades en el proceso de dispenso de bebidas calientes.
2. Definición de los requisitos:
    - Identificar los requisitos que demanda el modelo de negocio dentro del contexto en que se desenvuelve.
    - Definir los parámetros necesarios correspondientes a factores relacionados a la medición de flujo, sensores necesarios para implementar el dispositivo IoT y el equipo necesario para sensado de datos.
    - Planificar las tareas que son necesarias para diseñar e implementar el dispositivo IoT.

3. Diseño e implementación del dispositivo IoT:

    - Selección de la arquitectura IoT: Seleccionar la arquitectura IoT que satisfaga las necesidades transmitidas por los usuarios.
    - Diseño del dispositivo IoT: Diseñar el Dispositivo IoT bajo la arquitectura de 3 capas de IoT. (percepción, red y aplicación). Tomando como referencia las capas de percepción y de red.
    - Implementación a nivel de hardware del dispositivo IoT: Realizar las conexiones necesarias a nivel de hardware entre las placas y los sensores de tal modo que garanticen la toma de datos.
    - Implementación a nivel de software del dispositivo: Implementar y compilar el código fuente necesario para el funcionamiento de los dispositivos en las diferentes placas.
    - Configuración del bróker: Dentro de la capa de red es necesario realizar la configuración del bróker tal que se puedan realizar las publicaciones y subscripciones de varios clientes dentro de un tópico destinado para el tratamiento de datos.
    - Conexión entre el dispositivo IoT y el bróker: Constatar que el proceso de conexión entre el dispositivo IoT y el bróker garantice la captación de datos sensados.
    - Adaptar el dispositivo de manera que pueda evaluar la medición el flujo de diferentes tipos de bebidas calientes tales como café, té, leche y sus diferentes mezclas comerciales (capuchino, mocaccino, etc.).

4. Evaluación:

    - Realizar las pruebas de funcionamiento del dispositivo IoT, mediante el despliegue de este en un entorno simulado, para verificar que los datos guarden integridad y que se garantice que estos sean transmitidos de manera precisa y completa por los sensores.

1.4 Marco teórico

1.4.1 *User Centered Design*

El diseño centrado en el usuario (UCD) es un enfoque para el diseño y el desarrollo de productos que se orienta en las necesidades, expectativas y limitantes de los usuarios finales del producto. Para lograr concluir con éxito el proceso de desarrollo de un producto, este enfoque comúnmente recomienda hacer uso de siete etapas que son: Investigación del usuario (análisis contextual), definición de requisitos, diseño, evaluación, implementación, lanzamiento y mantenimiento e iteración [1].

Para el desarrollo de este trabajo se optó por usar cuatro etapas descritas a continuación:

1. Análisis contextual: Se define al usuario, sus necesidades, objetivos y el contexto en que hará uso del producto.
2. Definición de requisitos: Se definen los requisitos para el desarrollo del producto identificando las características y funcionamiento necesario, con el fin de satisfacer los requerimientos del usuario.
3. Diseño: Aquí se desarrollan prototipos iniciales que son la base del diseño final del producto, enfocado en la usabilidad y experiencia del usuario.
4. Evaluación: Los resultados de los prototipos de la etapa de diseño son expuestos a pruebas de usabilidad bajo condiciones reales de usabilidad.

En el UCD hay cuatro principios clave para el desarrollo y diseño: Enfoque en los usuarios, diseño iterativo, pruebas de usabilidad y trabajo multidisciplinario.

El propósito del diseño centrado en el usuario es el de crear productos enfocados en los usuarios, que garanticen la decisión de diseño y desarrollo bajo la perspectiva del usuario, esto en busca de lograr productos que sean funcionales, intuitivos y que rindan satisfacción al usarlo.

Dentro del enfoque de este proyecto el UCD se lo usa debido a la adaptación flexible a productos y su orientación a que el producto final sea útil, intuitiva, fácil de usar y sobre todo que satisface las necesidades del usuario.

1.4.2 *Internet of Things*

La definición de Internet de las cosas no es única, ya que esta se ajusta al punto de vista de diferentes grupos, sean estos académicos, investigadores, empresarios y en general

grupos afines a las tecnologías de información y comunicación. Una de las definiciones más concretas y que se ajusta a la actualidad sería la siguiente:

El internet de las cosas es un conjunto de objetos inteligentes, que son capaces de compartir información, datos y recursos, con capacidad autoorganizativa. Dichos objetos interactúan bajo una red que permite su conexión y comunicación, pudiendo así reaccionar y actuar ante diversas situaciones y en variaciones en el entorno [2].

1.4.3 Arquitecturas IoT

De acuerdo con lo detallado anteriormente dentro de la definición de IoT y la complejidad que conlleva por el punto de vista de diferentes grupos, y que el funcionamiento de IoT se basa en una variedad de sensores interconectados a redes por tecnologías de comunicación [3], no existe el concepto de una arquitectura definida de manera única y oficial. Variedad de profesionales e investigadores han propuesto algunas arquitecturas y según los requerimientos del proyecto y factores de calidad de servicio, integridad de datos, tráfico de datos, etc. estas pueden variar. En vista de esto se describe las arquitecturas IoT más utilizadas para el desarrollo de dispositivos IoT y proyectos afines, comenzando desde el modelo más referenciados de tres capas hasta llegar a un de cinco capas.

1.4.3.1 Arquitectura de tres capas

Según [4], tradicionalmente la arquitectura IoT más utilizada se forma por 3 capas, las cuales son:

- Capa de percepción: Es la capa de más bajo nivel, en la cual se reconoce y se recopila la información (data) de los objetos.
- Capa de transporte: También conocida como capa de red, es una capa de nivel intermedio en la cual se direccionan y transmiten los datos que fueron recopilados en la capa de percepción, por medio de una red. Ejemplos de tecnologías de comunicación inalámbrica más utilizadas por esta capa son Wi-Fi y Bluetooth.
- Capa de aplicación: Esta capa es dedicada a la presentación visual de los datos obtenidos y transmitidos por las capas anteriores, hacia los usuarios finales (clientes).

Las capas de percepción y transporte de datos (llamada también de red), son transversales en las arquitecturas que constan de más capas, motivo por el cual se las repite en su estructura.

En la Figura 2 se muestra la arquitectura de tres capas y elementos que la conforman.

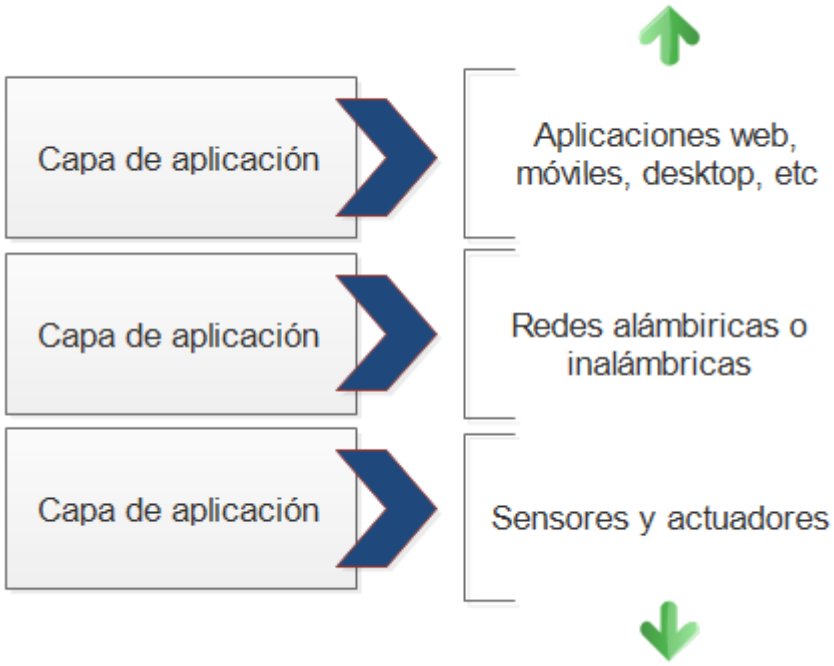

Figura 2. Arquitectura de tres capas [5]

1.4.3.2 Arquitectura de cinco capas

En el modelamiento de la arquitectura de cinco capas se mantiene las capas transversales de aplicación, de red y la capa de aplicación, se incluye únicamente una capa de *middleware* y una de negocio. El funcionamiento de estas dos nuevas capas dentro de la arquitectura se las detalla a continuación:

- Capa de *middleware*: Su papel en la arquitectura es el de ser un intermediario entre la capa de red y la capa de aplicación, facilitando la interoperabilidad, gestión de datos y puede también asignar un almacenamiento temporal de datos en caso de ser requerido [6].
- Se la conoce también como capa de servicios y es encargada de brindar servicio a través de aplicaciones con el objetivo de transformar datos obtenidos y procesados por las capas anteriores en un valor agregado de interés para las organizaciones que usan estas aplicaciones [7].

En la Figura 3 se muestra la disposición de la arquitectura de cinco capas.

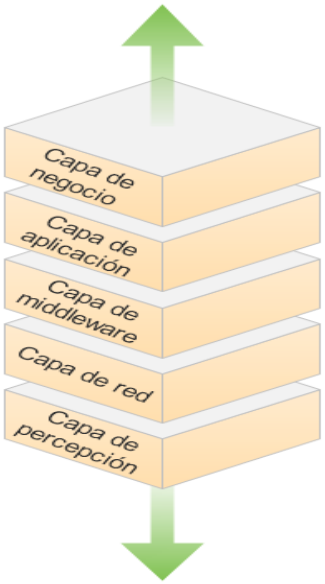

Figura 3. Arquitectura de cinco capas [8]

1.4.4 Protocolos usados en IoT

Dentro del ecosistema del IoT existen varios protocolos, estos protocolos son usados para permitir la comunicación entre diferentes dispositivos y aplicaciones de IoT, también se usan para gestionar la transferencia de datos junto con sus características de integridad, disponibilidad y confidencialidad. Los protocolos pueden variar dependiendo los requerimientos y necesidades específicos de cada dispositivo o aplicación de IoT [9].

1.4.5 *Message Queuing Telemetry Transport* (MQTT)

De acuerdo con [10], MQTT es un protocolo específicamente diseñado para la comunicación entre máquinas, este protocolo tiene como característica el consumo de energía muy bajo, esto debido a que los paquetes de datos tienen un tamaño que cuentan con sobrecarga mínima, produciendo así una mejor vida útil de los dispositivos. Al ser un protocolo independiente de datos, los puede transmitir bajo diversas formas como binario, texto, XLM o JSON.

MQTT utiliza un esquema de publicación/suscripción para la transmisión de mensajes y es compatible con microcontroladores y dispositivos IoT dominantes en el mercado como Arduino, Raspberry Pi, Wemos, entre otros. Es necesario implementar un bróker MQTT para la administración de datos

Componentes principales:

Cliente MQTT: Se puede considerar al cliente a cualquier software o aplicación instalada en un dispositivo IoT conectado a un bróker MQTT para poder intercambiar mensajes, usualmente utilizan tecnologías como JavaScript y la librería *Client* PAHO licenciado por Eclipse.

Bróker MQTT: Es un núcleo central de mensajería que maneja la publicación y suscripción entre clientes MQTT. Normalmente el bróker se lo desarrolla bajo una plataforma Linux y se puede también hacer uso de bróker gratuitos como Mosquitto, HimeMQ, EMQX, etc.

Conexión de clientes

Los clientes exclusivos que hacen uso del servicio son nombrados clientes MQTT, pueden ser comprendidos por sensores, placas, dispositivos IoT, teléfonos inteligentes o cualquier dispositivo que utilice el protocolo MQTT.

Estos clientes pueden actuar como suscriptores o publicadores dentro de tópicos definidos en el bróker. Los tópicos actúan como canales, en los que se categorizan los mensajes.

Publicación

Un cliente MQTT es un publicador cuando envía (publica) datos (mensajes) en un tópico definido en el bróker.

Suscripción

Un cliente MQTT es considerado como suscriptor cuando éste se suscribe a un determinado tópico para recibir los datos publicados. Dentro de la arquitectura del proyecto el suscriptor es una aplicación web.

Tópico

Es un canal virtual por el cual se organizan los diferentes mensajes provenientes de los clientes MQTT.

En la Figura 4 se presenta la arquitectura del protocolo MQTT.

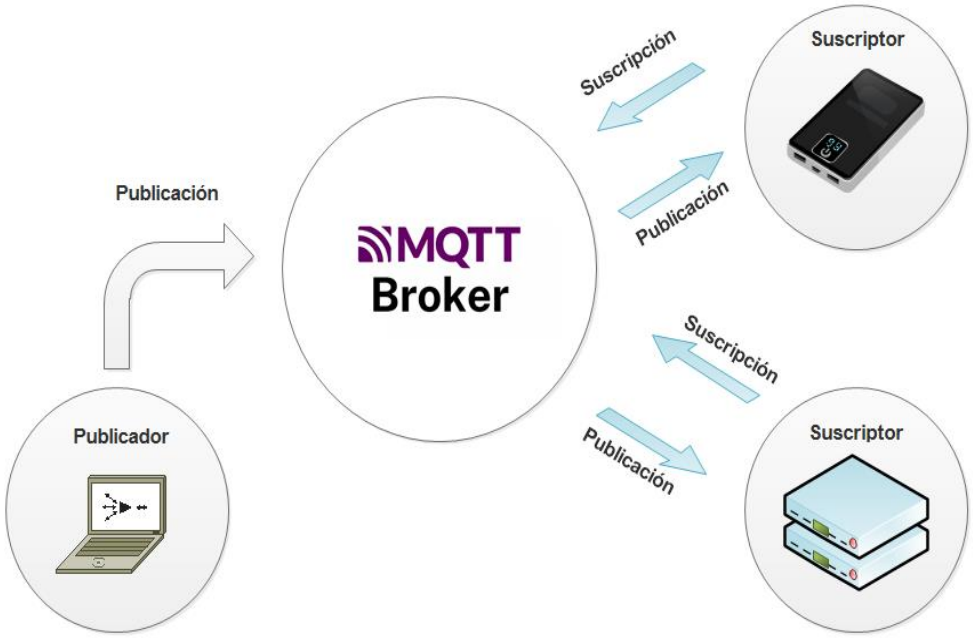

Figura 4. Arquitectura de publicación y suscripción de clientes del protocolo MQTT

Ventajas de uso

- Tanto el publicador como el suscriptor no necesitan conocerse ya que están conectados al bróker.
- Al no estar conectados simultáneamente, los clientes pueden recibir los datos publicados con un retraso.
- Utilización de *Topics*, que sirven como filtros para que el bróker envíe mensajes a los clientes conectados y suscritos al mismo.
- Utiliza tres niveles de calidad de servicio, los cuales garantizan la fiabilidad en la entrega de mensajes y se los detalla a continuación:
    - QoS 0: Entrega el mensaje como máximo una vez y no se garantiza la entrega a todos los suscriptores y no hay reintentos en caso de una falla.
    - QoS 1: Entrega el menaje garantizándolo al menos una vez a un suscriptor, mediante un intento de reenvío si lo reconoce.
    - QoS 2: Se garantiza que el mensaje se lo entregue exactamente una sola vez a un suscritor, de manera que no se duplique este mensaje.

1.4.6 *Constrained Application Protocol* (CoAP)

Bajo el contexto del internet de las cosas, el protocolo CoAP es específicamente hecho para la interoperabilidad de dispositivos con recursos limitados, este protocolo es muy parecido al protocolo HTTP debido a que se basa en envío de solicitudes y respuestas, lo que hace ligero a este protocolo es el uso de UDP (*User Datagram Protocol,* por sus siglas en inglés) con mensajes mucho más compactos.

Gracias a su configuración, CoAP es fácilmente traducible a HTTP y reduce la sobrecarga en el proceso de comunicación, mejorando así la eficiencia [11].

1.4.7 *Advanced Message Queuing Protocol* (AMQP)

Abreviado con las siglas AMQP, este protocolo es diseñado para la facilitar la comunicación entre diferentes dispositivos y aplicaciones, al ser un protocolo abierto y destinado a trabajar en la capa de *middleware* garantiza la fiabilidad en la entrega de información. Una de las características más sobresalientes de AMQP es la seguridad, debido a soporta autenticación y cifrado con el fin de proteger la información, esto hace que cumpla con integridad y confidencialidad en el proceso de transmisión de datos en diferentes entornos de IoT [12].

1.4.8 NodeMCU

Es una plataforma de código abierto, la cual está basada en una placa cuyo diseño es dispuesto para la libre edición, modificación y construcción.

Una de las configuraciones de esta placa es la conformada por el microcontrolador ESP8266, el cual tiene la capacidad de conectarse por el protocolo de comunicación Wi-Fi 802.11 b/g/n con pila TCP/IP completa. Usa un sistema de archivos SPIFFS desarrollado por la empresa *Espressif Systems*, entre sus características principales se tiene:

- Conexión y alimentación mediante un puerto micro USB.
- De bajo costo.
- Compatibilidad con varios entornos de programación.
- Voltaje variable de operación, oscilante entre 2.5V a 3.6V.
- Memoria: 32Kb en RAM, 80 Kb en DRAM y 200 Kb en *Flash*.
- Alineación de pines: 17 de GPIO, un canal ADC de 10 bits, cuatro pines PWM, interfaces SPI e I2C, y una interfaz I2S.

- Botones de *Flash* y *Reset* [13].

Debido a que la placa NodeMCU en configuración con el microcontrolador ESP8266 es de compatibilidad múltiple, se puede emplear Arduino IDE para su programación [14]. En la Figura 5 se puede apreciar la disposición física de pines y componentes de la placa NodeMCU.

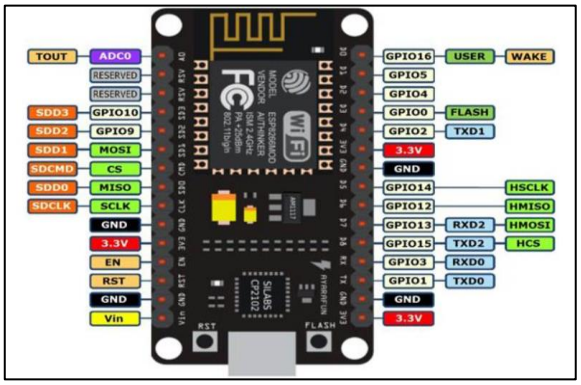

Figura 5. Asignación de pines de la placa NodeMCU-ESP8266 [15]

1.4.9 Sensor de flujo modelo yf-s401

El sensor de flujo de modelo yf-s401, es un sensor destinado a medir el flujo de líquidos (generalmente agua) que fluye por él, su composición es de un cuerpo de cloruro de polivinilo (PVC) y en su interior tiene un rotor y un transductor de efecto *Hall*.

Su funcionamiento se basa en la detección de la rotación del rotor cuando un flujo de agua pasa por el sensor, lo cual produce una cantidad de pulsos digitales que son proporcionales a la cantidad de agua que fluye en el sensor [16].

Para la conexión con dispositivos consta de tres cables conectores (Rojo: *IN positive*, Amarillo: *OUT signal output* y Negro: GND *negative*), su funcionamiento se limita a un voltaje mínimo de 4.5 voltios y 15 miliamperios (en corriente continua), con un flujo máximo soportado de 1 a 5 litros por minuto.

Este tipo de sensor es adecuado para medir flujo de agua en cafeteras [17], la Figura 6 representa la disposición física con cablería del sensor.

Figura 6. Sensor de flujo yf-s401 [17]

1.4.10 Sensor de Radio frecuencia RFID

Este sensor es un circuito integrado con funcionalidad de lectura y escritura para la comunicación sin tener contacto físico entre dispositivos, su nombre de modelo se debe al transmisor interno que lo compone nombrado como MFRC522, este controlador maneja a una antena de lectura y escritura con la capacidad de comunicarse con tags compatibles con los estándares ISO/IEC 14443 A/MIFARE y NTAG [18].

En la versión del sensor utilizada (V2.0) se destaca las siguientes características y disposición de pines mostradas en la Figura 7.

Características generales:

- Distancia máxima de funcionamiento (en lectura y escritura): 50 mm.
- Cifrado admitido (en lectura y escritura): MF1xxS20, MF1xxS70 y MF1xxS50.
- Transferencia máxima de datos: 10 Mbit/s.
- Modos de interrupción flexibles: Disponible.
- Apagado mediante software: Disponible.
- Voltaje de funcionamiento: de 2.5 a 3.3 V en DC.

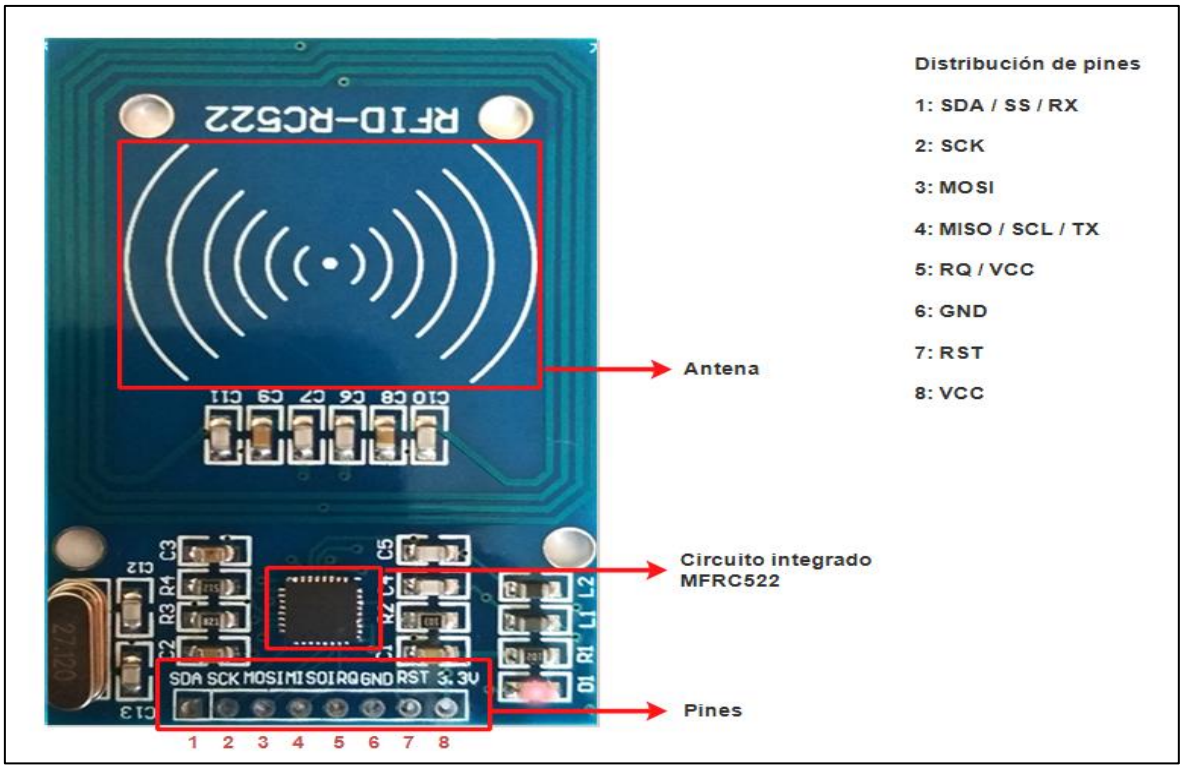

Figura 7. Módulo RFID MFRC522 con su distribución de pines

2 METODOLOGÍA

El desarrollo del presente proyecto se enmarcó bajo la metodología de diseño centrado en el usuario, debido a que este enfoque se basa en el diseño y el desarrollo de productos. Este enfoque es muy popular e itera en su ciclo de vida conformado por siete etapas, pero para el propósito del proyecto se considera al dispositivo IoT como producto y las siete etapas del proyecto las sintetizamos en cuatro etapas, dentro de estas etapas, específicamente en la etapa de diseño e implementación del dispositivo IoT se considera la aplicación de la arquitectura de tres capas.

A continuación, se puntualiza las etapas con sus respectivas actividades desarrolladas para alcanzar los objetivos presentados en la elaboración de esta investigación.

2.1 Análisis contextual

Se identifica a los usuarios con sus características y roles dentro del proceso de dispenso de bebidas calientes en los autoservicios. Se considera las necesidades que exponen los usuarios ante el modelo de negocio, para transformarlas en requisitos tanto funcionales como no funcionales y de esta manera poder diseñar el dispositivo IoT.

2.1.1 Identificación de los usuarios finales y sus necesidades:

El objetivo de esta etapa es el de la identificación de los usuarios que harán uso del autoservicio de bebidas calientes en conjunto funcionamiento con el dispositivo IoT, para lograrlo se necesita definir las personas que interactuarán con el autoservicio.

Estas personas consideradas como usuarios finales del autoservicio son tres y sus roles se los presenta a continuación en la Tabla 1:

Tabla 1. Usuarios y roles que usan el autoservicio

Nombres	Rol	Descripción
Diana Yacchirema	Administradora	Persona encargada de realizar el cobro por utilizar el autoservicio, maneja el sistema para monitoreo y administración del negocio.
Samir Zurita	Cliente	Persona que usa el autoservicio para servirse la bebida caliente
Eduardo Avilés	Empleado operador	Persona encargada de mantener el autoservicio funcional, lleno de la bebida caliente que corresponde y realiza mantenimientos.

2.1.1.1 Necesidades del cliente, administrador y empleados

En [19] se describe a la experiencia de consumo como una satisfacción que el consumidor obtiene a partir del valor de la oferta de un servicio que resulta ser atractivo para él. La evaluación de esta experiencia se la realiza mediante factores influyentes, los más importantes comprendidos por: necesidades y deseos, nivel de satisfacción, hábitos de consumo, beneficios y estímulos.

El proceso que implica el autoservicio de las bebidas a elección del cliente, la identificación del cliente con sus respectivos datos personales y el proceso de pago en ocasiones puede ser un proceso un poco engorroso por diversas situaciones del contexto del negocio, un ejemplo de esto es, el proceso que implica en servirse cierta cantidad de bebida y su posterior pago, para lograrlo implica invertir un lapso de no menos de diez minutos.

En este panorama la principal necesidad de los clientes implicados es el de obtener una experiencia de consumo satisfactoria que de valor al servicio brindado por el establecimiento comercial, lo que implica un servicio eficiente, simplificado y con productos calidad. En ese sentido el cliente necesita de un punto de dispenso (autoservicio) en el cual se pueda lograr el consumo de bebida a su elección y que al momento de cobrar por esto, se lo haga de manera fácil sin ningún trámite que dificulte el proceso.

Por el lado de los dueños de los establecimientos comerciales, la principal necesidad incurre en brindar un servicio eficiente, novedoso y óptimo, que garantice una mejora en la experiencia del cliente, que de valor agregado a sus productos y servicios y que todo el proceso de dispenso de la bebida y cobro por el servicio y producto sea automatizado.

2.1.1.2 Beneficio de uso del dispositivo IoT

El proceso para comprar una bebida caliente dentro de un establecimiento comercial que dispensa bebidas en un autoservicio se lo realiza mediante un proceso largo y de cierto modo restringido para el cliente. El proceso inicia con la compra de una bebida a elección en una cantidad definida, seguidamente una vez cancelado el valor de esta bebida, el empleado del establecimiento toma los datos del cliente para llenar en la factura correspondiente, finalmente el empleado debe ir a operar el autoservicio para servir la cantidad de bebida que el cliente compró, en caso de que el usuario requiera de otra bebida debe realizar el mismo procedimiento.

En la actualidad la automatización del proceso de autoservicio de bebidas calientes no es implementada por empresas en el ámbito nacional, razón por la cual se plantea esta solución. Al usar los autoservicios en conjunto con el dispositivo IoT se busca brindar un servicio eficiente y que simplifique los pasos en el proceso de dispenso y recarga de bebida caliente, también facilita el registro del dispenso de bebidas al identificar un cliente mediante un TAG, para el posterior registro de los datos del cliente y cobro de la bebida consumida. Con esto el establecimiento comercial logra brindar una mejor experiencia de consumo al usuario y aumenta la propuesta de valor del establecimiento comercial que brinda el servicio.

2.2 Definición de requisitos

Para que el dispositivo IoT satisfaga las necesidades de los usuarios hay que identificar claramente los requisitos, este proceso puede resultar un poco complejo ya que no existe una técnica o lineamientos establecidos específicos para garantizar la calidad del resultado final. Sin embargo, para realizar este proceso se hace uso de diferentes técnicas que se propone en las metodologías de desarrollo [20].

En la Figura 8 se muestra el proceso de definición de requisitos.

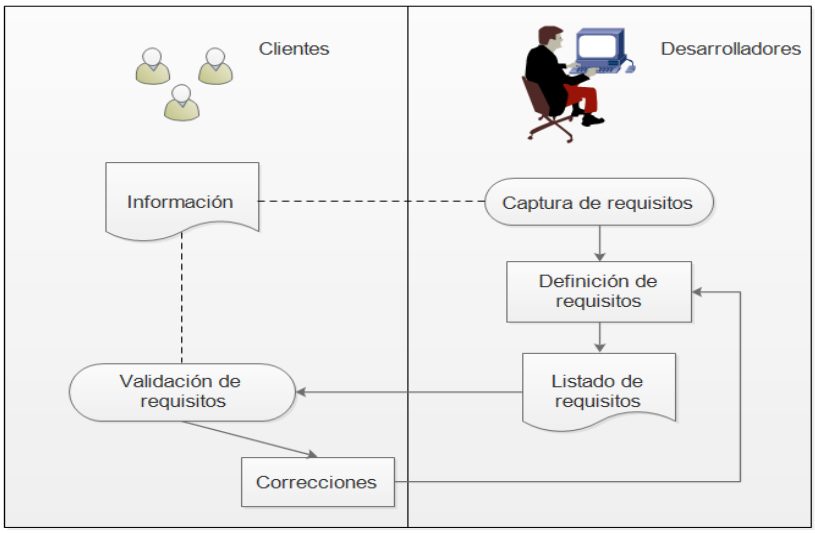

Figura 8. Proceso de definición de requisitos

2.2.1 Captura de requisitos

Para esta actividad se hace uso de la técnica de casos de uso, decido a la efectividad de esta técnica, el enfoque al usuario y la facilidad de detección de requisitos completos. Estos casos de uso nos proporcionan un panorama mediante el cual podemos documentar el proceso del uso de los autoservicios y los implicados en ello. Bajo la observación y manipulación de los autoservicios en proceso de dispenso de bebidas en establecimientos comerciales ubicados en sectores comerciales de la ciudad de Cuenca y Quito, se evidenció los siguientes casos de uso.

Tabla 2. Casos de uso de utilización de los autoservicios

Caso de uso	Actor	Descripción	Consideraciones
Selección y dispenso bebida	Cliente	El cliente elije la bebida caliente de su preferencia y se dispone a utilizar el autoservicio para obtener la cantidad de bebida que el disponga	El autoservicio debe activarse cuando el cliente sea identificado, esto para poder ligarlo al consumo y de esa manera cobrar por la bebida que consumió
Recarga de bebidas	Empleado operador	El empleado recarga de bebida caliente cuando el líquido en el autoservicio esté agotado	El autoservicio debe desconectarse de los elementos electrónicos y mecánicos para poder recargar de líquido
Realizar inspección del encendido y apagado del autoservicio	Empleado operador	El empleado debe encender el autoservicio al inicio de la jornada laborar	El autoservicio no necesita desconectarse físicamente de los componentes

		y constatar que funcione correctamente, así mismo el empleado apaga el autoservicio al finalizar la jornada	electrónicos y mecánicos que lo componen
Realizar mantenimiento	Empleado operador	El empleado realiza un mantenimiento rutinario que consiste en limpieza y un chequeo rápido en el que el autoservicio funcione con normalidad	El autoservicio debe estar apagado para realizar la limpieza
Realizar cobro por el servicio y la bebida consumida por el cliente	Administrador	El administrador hace uso de una aplicación web para realizar el cobro del servicio y de la o las bebidas que el cliente consumió	El cliente debe estar ligado el consumo de las diferentes bebidas por algún medio o método que garantice su consumo y utilización de autoservicio. El administrador debe registrar los datos del cliente y poder guardarlos y modificarlos desde la aplicación

2.2.2 Definición de requisitos

Con base en los casos de uso se procede a definir los requisitos funcionales y no funcionales del producto. Cabe recalcar que cuando se nombra al autoservicio se hace referencia al autoservicio simulado de bebidas calientes funcionando en conjunto con el dispositivo IoT.

Requisitos funcionales:

- El autoservicio debe permitir a los clientes el dispenso y servicio de la bebida de su elección y la cantidad que ellos estimen necesaria.
- El autoservicio debe estar ligado a un tipo de bebida con identificación única.
- El autoservicio debe permitir el dispenso siempre y cuando el usuario sea identificado por un TAG que se le proporciona.
- El autoservicio debe permitir a los clientes dispensar las bebidas calientes las veces que ellos requieran hasta su posterior pago para finalizar el servicio.

- Mediante el TAG y los sensores del autoservicio en conjunto con la aplicación web (otro componente del plan de diseño de integración) se debe permitir ligar el consumo de bebidas a un cliente para poder realizar el cobro del servicio y la bebida.
- El autoservicio permitirá reconocer todos los TAGs previamente registrados.
- El autoservicio debe ser rápido y eficiente, minimizando el tiempo de espera para recibir la bebida.
- El autoservicio debe ser compatible con los módulos electrónicos de hardware y componentes mecánicos existentes para la dispensación de bebidas.

Requisitos no funcionales:

- El autoservicio debe responder a las acciones del cliente en menos de 2 segundos.
- El autoservicio debe permitir una disponibilidad de al menos el 99% del tiempo cuando esté encendido.
- El autoservicio deberá permitir futuras actualizaciones de su software.
- El autoservicio deberá facilitar el acceso para el mantenimiento, sin comprometer ningún elemento mecánico ni electrónico que lo conforman.
- Los datos obtenidos por el autoservicio deben guardar integridad y confidencialidad.
- La arquitectura del autoservicio debe ser escalable para manejar un incremento en el número de clientes y nuevos módulos para administración de los datos de bebidas.

2.2.3 Validación de requisitos

Para realizar el proceso de validación de resultados existen numerosas técnicas, en este caso se decidió por realizar una matriz de trazabilidad, por medio de la cual se seleccionamos los objetivos del producto y los chequeamos con los requerimientos para así cubrir cada objetivo. Como objetivo de la implementación de esta matriz se tiene principalmente la validación de los requisitos y como un beneficio complementario, se podrán detectar objetivos no cubiertos e inconsistencias dentro del desarrollo del proyecto.

A continuación se muestra la matriz de trazabilidad:

Tabla 3. Matriz de trazabilidad de objetivos y requisitos

Objetivo específico	Requisitos funcionales	Requisitos no funcionales
Seleccionar la arquitectura IoT para utilizarse en la implementación del dispositivo	N/A	El autoservicio deberá permitir futuras actualizaciones de su software
Adaptación y aplicación de la metodología de diseño centrado en el usuario para el diseño e implementación del dispositivo	El autoservicio debe permitir una disponibilidad de al menos el 99% del tiempo cuando esté encendido	N/A
Seleccionar las capas dentro de una arquitectura IoT	N/A	La arquitectura del autoservicio debe ser escalable para manejar un incremento en el número de clientes y nuevos módulos para administración de los datos de bebidas
Diseñar el dispositivo en función de las capas de la arquitectura	El autoservicio debe ser compatible con los módulos electrónicos de hardware y componentes mecánicos existentes para la dispensación de bebidas	N/A
Seleccionar y configurar los sensores adecuados para medir el flujo de las bebidas calientes en el autoservicio, asegurando su precisión y fiabilidad en diferentes condiciones de uso	El autoservicio debe permitir a los clientes dispensar las bebidas calientes las veces que ellos requieran hasta su posterior pago para finalizar el servicio	El autoservicio deberá facilitar el acceso para el mantenimiento, sin comprometer ningún elemento mecánico ni electrónico que lo conforman
Ensamblar el hardware, según el diseño del prototipo, incluyendo la integración de los sensores, la electrónica de control y la conectividad necesaria entre los componentes	El autoservicio debe permitir el dispenso siempre y cuando el usuario sea identificado por un tag que se le proporciona	N/A
Desarrollar e implementar el código correspondiente destinado al ensamble a nivel de software de las placas y sensores	El autoservicio debe permitir a los clientes el dispenso y servicio de la bebida de su elección y la cantidad que ellos estimen necesaria	El autoservicio debe responder a las acciones del cliente en menos de 2 segundos
Configurar un bróker para recibir y almacenar los datos de flujo y los identificadores de los TAGs correspondientes a	El autoservicio permitirá reconocer todos los tags registrados	Los datos obtenidos por el autoservicio deben guardar integridad y confidencialidad

los sensores y placas, asegurando la disponibilidad de los mismos.		
Establecer la conexión necesaria entre las placas y el bróker.	El autoservicio debe permitir a los clientes el dispenso y servicio de la bebida de su elección y la cantidad que ellos estimen necesaria.	N/A
Realizar pruebas de funcionamiento del dispositivo IoT mediante el despliegue en condiciones reales de uso en el autoservicio de bebidas calientes.	El autoservicio debe ser rápido y eficiente, minimizando el tiempo de espera para recibir la bebida.	El autoservicio debe permitir una disponibilidad de al menos el 99% del tiempo cuando esté encendido. La aplicación deberá funcionar en cualquier sistema operativo y en cualquier navegador.

La matriz de trazabilidad muestra como los objetivos específicos se relacionan directamente con los requisitos funcionales y no funcionales y de esta manera asegurar que se cumplan las necesidades y expectativas del producto final.

2.3 Diseño e implementación del dispositivo IoT

En esta etapa se detalla el diseño e implementación del dispositivo IoT que cumplirá con las expectativas del producto final.

2.3.1 Selección de la arquitectura IoT

Para el diseño del dispositivo IoT se basó en la arquitectura de tres capas. Esto debido a que este enfoque es muy comúnmente utilizado en el diseño de dispositivos IoT, tanto por su simplicidad, modularidad y escalabilidad [21]. En la Tabla 4 se muestra los criterios de selección para elegir la arquitectura de tres capas.

Tabla 4. Criterios de selección de la arquitectura de tres capas [22]

Criterio	Descripción
Simplicidad	Esta arquitectura es fácil de entender y también fácil en su implementación
Modularidad	Se mantiene una clara separación de las responsabilidades de cada capa y son independientes una de otra
Escalabilidad	Ofrece una adaptación al crecimiento del proyecto sin la necesidad de rediseño
Interoperabilidad	Tiene facilidad en integrarse con otros sistemas, dispositivos IoT, asegurando la comunicación entre estos
Seguridad	Se puede implementar medidas de seguridad en cada capa
Costo-Efectividad	Con su simplicidad reduce los costos de implementación y optimiza el uso de recursos

Estos criterios satisfacen a los objetivos del componente, a las necesidades de los clientes y a los requerimientos del producto.

El modelo de tres capas está conformado por las capas de percepción, red y aplicación, pero para el desarrollo de este componente se consideró las capas de percepción y de red. En la Figura 9 se referencia la imagen de la arquitectura considerando las dos capas ya mencionadas.

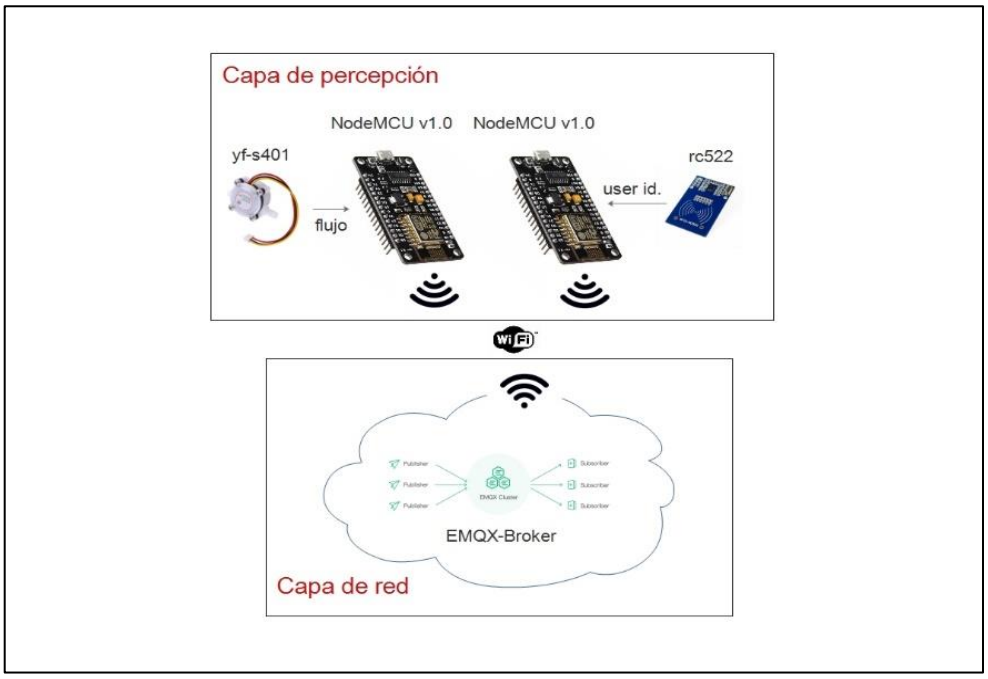

Figura 9. Arquitectura del dispositivo IoT

2.3.1.1 Capa de percepción

La capa de percepción está constituida esencialmente por el dispositivo IoT, el cual se conforma por dos placas NodeMCU ESP8266 V1.0, cada una de estas placas se conecta de manera independiente a un sensor de flujo modelo yf-s401 y a un módulo lector de Identificación por Radio Frecuencia (RFID) modelo MFRC522, como se muestra en la siguiente Figura 10:

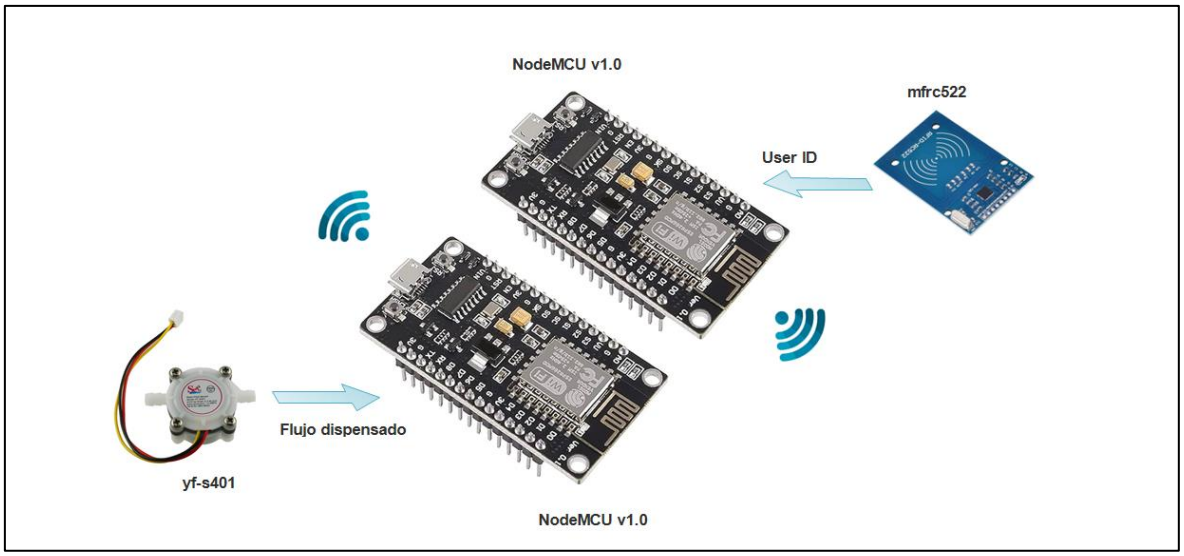

Figura 10. Capa de percepción

Esta configuración de sensores y placas correctamente conectados garantizan la recopilación de datos del entorno físico en la que los autoservicios cumplen su funcionamiento.

A continuación, se detallará el contexto de funcionamiento general de los sensores y placas utilizados en la capa de percepción.

2.3.1.1.1 NodeMCU ESP8266 V1.0

Dentro de la arquitectura en la capa de percepción, la función de cada placa NodeMCU es el de la gestión de datos de flujo y de identificación del TAG que usa el cliente, respectivamente, obtenidos mediante los sensores conectados.

2.3.1.1.2 Sensor de flujo

El sensor de flujo permitió captar el volumen que fluye por él, a través del rotor de efecto *Hall* que lleva en su interior y transmitir estos datos de volumen captados hacia el bróker por medio de la placa NodeMCU a la que está conectado.

Para la primera vez en que sean utilizados, este tipo de sensores requieren de una calibración según [23], esta calibración se la realizó mediante comparación de los datos de volumen de agua carbonatada que pasan por el sensor en cada dispenso. Los resultados obtenidos del proceso de comparación con un envase con un volumen definido se los muestra en la siguiente tabla:

Tabla 5. Medidas de calibración y errores del sensor de flujo

Volumen definido (ml)	Volumen detectado por el sensor (ml)			Error promedio (ml)	Porcentaje de error (%)
	Vol. 1	Vol. 2	Vol. 3		
100	101	105	103	3	3
200	197	203	201	2.3	1.15
300	300	302	303	1.7	0.57
400	389	401	400	0.7	0.18
500	502	500	500	0.7	0.14
Promedio del porcentaje de error					**1.008 %**

Para obtener los resultados del error promedio y el porcentaje de error, se los obtuvieron mediante las siguientes ecuaciones:

$$Error\ promedio = \frac{1}{n}\sum_{i=1}^{n}|y_0 - y_i|$$

Ecuación 1. Error promedio

Donde:

n = número de volúmenes detectadas por el sensor

y_0 = volumen definido en cada medida

y_i = volumen detectado por el sensor.

$$Porcentaje\ de\ error = \frac{Error\ promedio}{Volumen\ definido} \times 100\%$$

Ecuación 2. Porcentaje de error

Como se puede observar en la Tabla 1 el error promedio junto con el porcentaje de error disminuye, esto debido a que el sensor mientras se humecta va calibrándose y acercándose más a una medida real, obviamente siempre va a existir un error mínimo, que según [17] tiene siempre un margen de un 2%.

2.3.1.1.3 Módulo lector de Identificación por Radio Frecuencia (RFID)

La identificación de los usuarios por un medio físico se la consiguió con el uso TAGs, administrados por el sensor MFRC522. Este sensor capta el desplazamiento del TAG y reconoce el número identificativo, ligando así este número al volumen consumido de la

bebida caliente realizada por el usuario, este sensor va conectado a la segunda placa NodeMCU para lograr su cometido ya descrito.

Este módulo no necesita de una calibración previa para ser utilizado y se complementa con las tarjetas de identificación (TAGs). Estos TAGs normalmente vienen en 2 presentaciones (tarjetas y llaveros) y su configuración se presenta en la Figura 11.

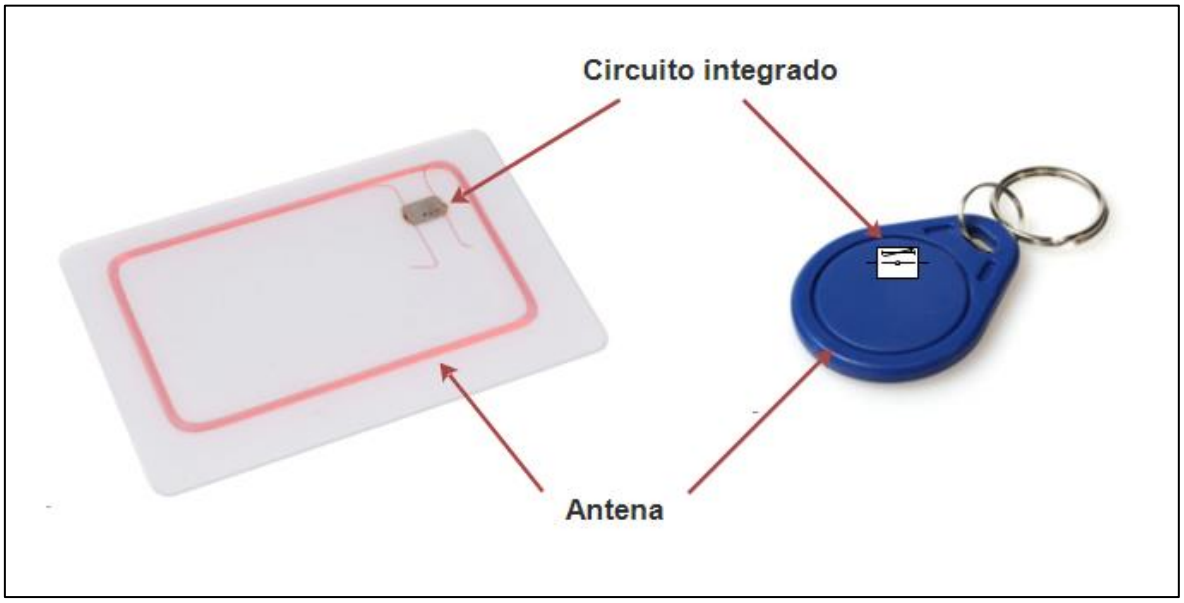

Figura 11. TAGs identificadores

Se puede evidenciar que la disposición de estos tags está conformada por una bobina NFC (antena) y su respectivo circuito integrado (chip), cada uno de estos tags viene configurado con un con una memoria de 1 KB con la intención de almacenar datos únicos, siendo el caso de este proyecto un número único para identificación de los clientes.

2.3.1.2 Capa de red

Esta capa es destinada a la administración y transportación de los datos obtenidos en la capa de percepción, para realizar el proceso de transmisión de datos captados por los sensores primeramente se implementó una red Wi-Fi debido a su capacidad de conexión rápida y sin cables, seguidamente se seleccionó un protocolo de IoT.

Para la elección del protocolo a utilizarse en este proyecto, se lo respaldó en la comparativa presentada en la Tabla 6. Esta comparativa se la realizó en base de las investigaciones realizadas y referenciadas en [24], [25] y [26], estas investigaciones.

Tabla 6. Comparativa entre protocolos de comunicación en el ámbito de IoT

Característica	MQTT	HTTP	CoAP	AMQP	LoRaWAN
Tipo de arquitectura	Publicación/ Suscripción	Solicitud/ Respuesta	Solicitud/ Respuesta	Solicitud/ Respuesta y Publicación/ Suscripción	LPWAN
Peso	Ligero	Pesado	Muy ligero	Pesado	Ligero
Utilización de ancho de banda	Eficiente	Alto	Muy eficiente	Alto	Muy eficiente
Confiabilidad	Alta (con QoS)	Variable	Media	Alta	Media
Consumo de energía	Bajo	Alto	Muy bajo	Alto	Muy bajo
Finalidad de uso	Redes IoT con recursos limitados y comunicación en tiempo real	Aplicaciones web, APIs RESTful	Dispositivos con recursos muy limitados	Aplicaciones empresariales y transacciones	Comunicación a larga distancia con baja tasa de datos

En consecuencia, de esta comparativa se concluye que el protocolo más idóneo para implementar la comunicación inalámbrica y de acuerdo con las características que lo componen, es el protocolo MQTT. En la siguiente sección se va a explicar el funcionamiento del bróker MQTT en el contexto del proyecto.

2.3.1.2.1 Bróker MQTT

En la capa de red, el bróker actúa como un administrador intermediario de mensajes, esta entidad tiene como finalidad garantizar la disponibilidad, integridad y confiabilidad de los mensajes y entregarlos a los diferentes suscriptores de su arquitectura [27].

La implementación del bróker fue mediante la herramienta EMQX y su plataforma de administración (*cluster*), este bróker se configura por la arquitectura MQTT en la cual los actores son los clientes publicadores conformadas por las placas NodeMCU, el cliente suscriptor que es la aplicación web (complementaría en el otro componente del proyecto) y el tópico nombrado como *flowmeter/data* para el filtrado de datos provenientes de los clientes publicadores.

En la Figura 12 se representa su configuración.

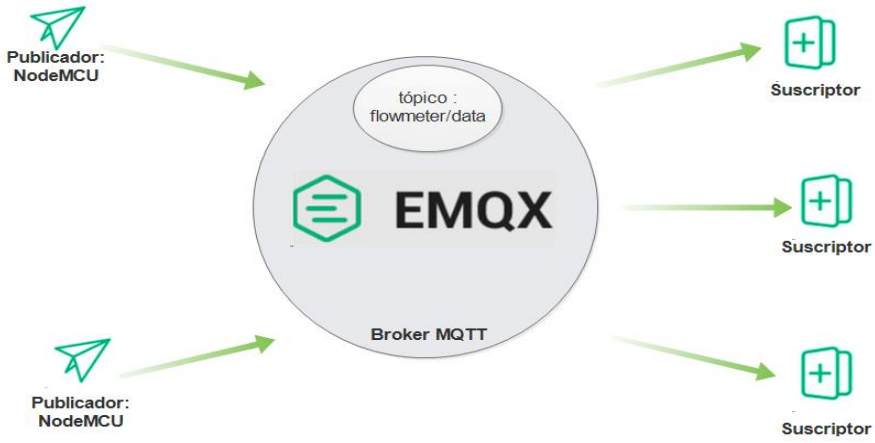

Figura 12. Arquitectura del bróker EMQX

2.3.2 Implementación del dispositivo IoT a nivel de hardware

En esta sección se detalla las conexiones entre las placas y sensores para implementar el dispositivo IoT, estas conexiones deben ser correctas para garantizar el sensado de datos y su posterior tratamiento.

Debido a la arquitectura y disposición de sensores del dispositivo IoT, para el desarrollo de este componente se utilizó dos placas (una para cada sensor), esto para poder evitar el *delay* de las placas debido a las interrupciones de los pulsos. Para identificarlas se nombró a las placas como: “Placa control de identificación” y “Placa control de flujo”. Ambas placas son suscriptoras de un solo tópico en el bróker MQTT.

2.3.2.1 Placa control de identificación

La placa de control de identificación NodeMCU se conecta al módulo de identificación por radio frecuencia MFRC522, con el objetivo de identificar el tag que es usado por el cliente. Con esto se logra ligar al cliente con el consumo de la bebida caliente que dispensó en el autoservicio. La configuración de conexión de pines entre la placa y el módulo MFRC522 se detalla en la Figura 13.

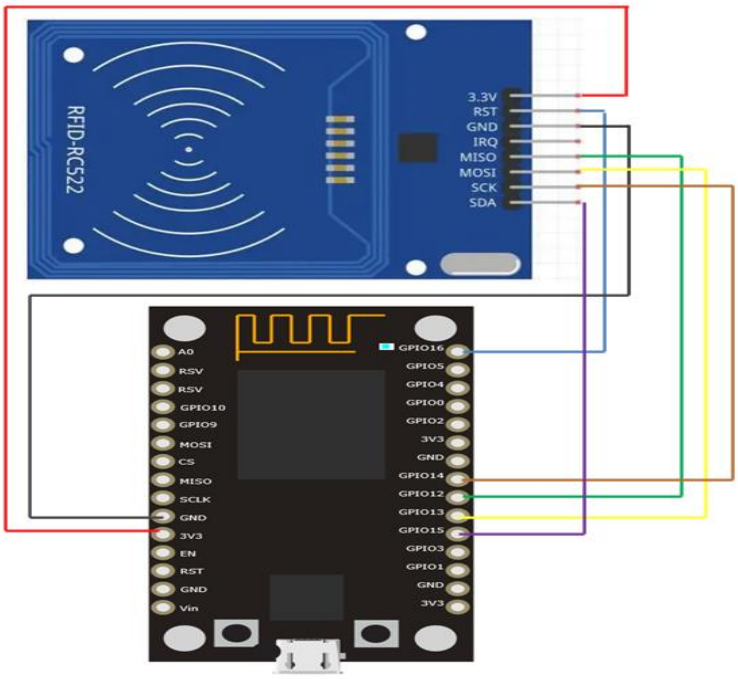

Figura 13. Diagrama de conexión “Placa control de identificación”

2.3.2.2 Placa control de flujo

En su configuración, la segunda placa NodeMCU se conecta al sensor de flujo yf-s401, esto con el objetivo de sensar el volumen de bebida caliente dispensado en el autoservicio por el cliente y su diagrama de conexión con el sensor se muestra en la Figura 14.

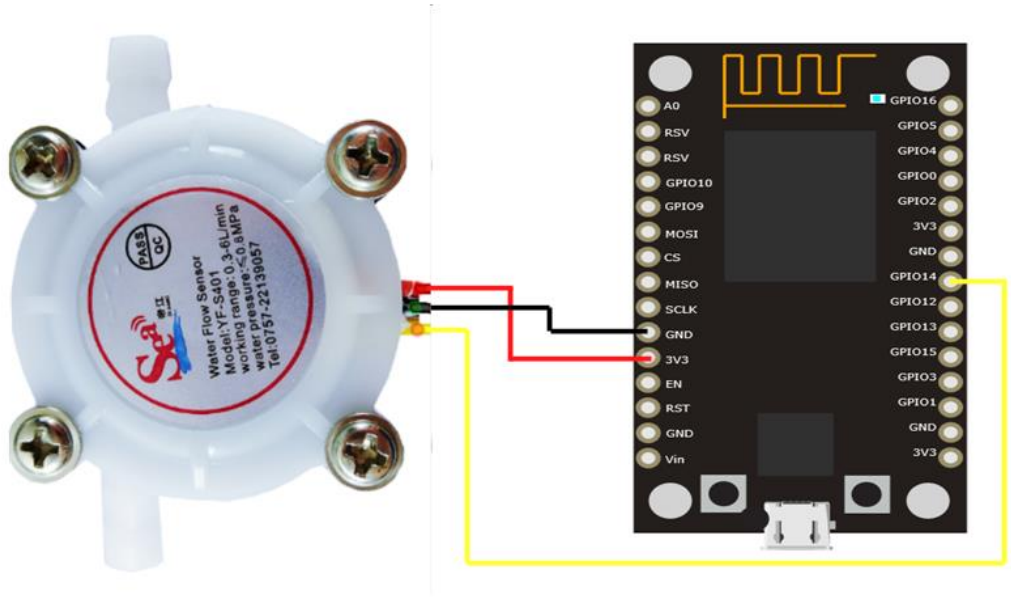

Figura 14. Diagrama de conexión “Placa control de flujo”

Finalmente en la Figura 15 se muestra al dispositivo IoT, conformado por las placas NodeMCU y sensores, soldados en una placa de prototipado con sus respectivos bornes de conexión rápida para poder conectar diferentes componentes de salida de información como *displays* o pantallas compatibles con la placa, fuentes de poder y los cables de conexión provenientes del sensor de flujo.

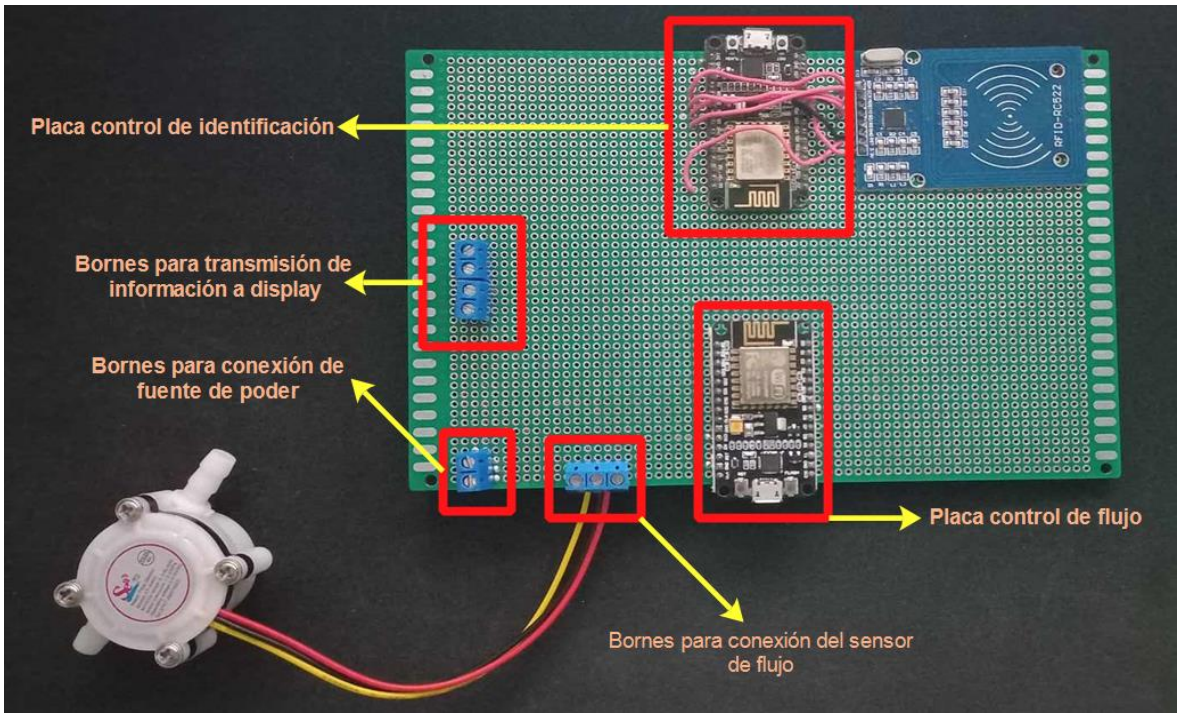

Figura 15. Conexiones del dispositivo IoT

2.3.3 Implementación del dispositivo IoT a nivel de software

En esta sección se detalla la implementación de los fragmentos de mayor relevancia de código fuente. El código fuente completo se lo encuentra en el ANEXO I y es necesario compilarlo en su totalidad para que el dispositivo IoT cumpla sus funciones de sensado de datos de identificación y flujo.

Si bien existen varios lenguajes de programación para implementar el código fuente del dispositivo, se ha elegido al lenguaje C/C++ debido a la eficiencia y rendimiento que brinda, al control sobre el hardware y sobre todo a la compatibilidad que tiene con varias plataformas IoT [28].

Al igual que en la anterior sección, se destinó dos subsecciones para describir el código implementado en cada placa.

2.3.3.1 Implementación del código fuente para la “Placa control de identificación”

La implementación de esta placa de control de identificación consta de dos procesos: Registro del código identificador y Lectura del código identificador TAG.

Es necesario mencionar que las conexiones a nivel de hardware entre la placa NodeMCU y el sensor MFRC522 no varían para ninguno de los dos procesos.

2.3.3.1.1 Registro del código identificador (TAGs)

Previo a usarse la placa como administradora de los códigos identificadores de los TAGs válidos, se necesita registrar estos códigos contenidos en cada TAG.

Para este proceso de registro primeramente se tiene que definir las librerías adecuadas para programación, mostradas en la Figura 16.

```
#include <MFRC522.h>
#include <SPI.h>
```

Figura 16. Código de librerías del registro de TAG

Este fragmento incluye las librerías para utilizar el bus SPI y el control del módulo RFID.

Para definir los pines de *reset* y *sda*, se lo realizó por medio de constantes asignadas a los pines D0 y D8 de la placa como se lo muestra en la Figura 17:

```
#define RST_PIN D0
#define SS_PIN D8
```

Figura 17. Código de definición de pines

Una vez definidas librerías y pines, se procede con la configuración inicial para la comunicación serial, inicialización del bus SPI y el comienzo de las funciones del sensor RFID, mostrando el mensaje “Desplace la tarjeta”, este se aprecia en la Figura 18.

```
void setup() {
  Serial.begin(9600);
  SPI.begin();  //inicio de BUS SPI
  RadioLectorRF.PCD_Init(); //comenzar a utilizar el rfid
  Serial.println("Desplace la tarjeta.......");
}
```

Figura 18. Código de configuración inicial

Finalmente dentro del bucle principal se comprueba si hay un nuevo TAG detectado por el sensor, y de ser así, lee su número identificador de este TAG para guardarlo en la variable tag de tipo String y así poder mostrarlo en monitor serie.

Luego de completar este proceso reinicia la variable tag para poder realizar una nueva lectura. Este proceso se lo detalla en el código de la Figura 19.

```
void loop() {
  // detección de la tarjeta mediante validación
  if(RadioLectorRF.PICC_IsNewCardPresent()){
    return;
  }
  if(RadioLectorRF.PICC_ReadCardSerial()){
    for (byte i = 0; i < 4; i++) {
      tag += RadioLectorRF.uid.uidByte[i];
  }
  Serial.println(tag);
    tag = "";
    RadioLectorRF.PICC_HaltA();
    RadioLectorRF.PCD_StopCrypto1();
  }
}
```

Figura 19. Código del bucle principal

Con este proceso obtenemos los números identificadores de los TAGs para registrarlos y usarlos en el proceso de lectura del TAG.

2.3.3.1.2 Lectura del código identificador del TAG

En esta sección se dará una explicación sobre la implementación del código de configuración de la placa y el módulo RFID, para detectar los códigos únicos identificadores de los tags. A continuación se desglosa de manera ordenada los bloques de código para la funcionalidad.

Como primer paso se incluye las librerías y se define los pines destinados a utilizarse, mostrados en la Figura 20.

```
#include <SPI.h>
#include <MFRC522.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

#define SS_PIN D8       // Pin D8 para el SS (SDA) del RC522
#define RST_PIN D0      // Pin D0 para el reset del RC522
```

Figura 20. Código para definición de librerías y pines

A diferencia del código del registro del código identificador, se ha añadido dos librerías extras:

- La librería ESP8266WiFi.h que proporciona instrumentos necesarios para el establecimiento, gestión y utilización de conexiones por medio de la red inalámbrica Wi-Fi [29].
- La librería PubSubClient.h que permite establecer un cliente MQTT, que tiene funciones de publicación y suscripción con el bróker MQTT con el que se encuentra conectado [30].

Seguidamente, para conectar la placa de control de identificación con el broker MQTT, se declaró las credenciales de la red Wi-Fi a la que se va a conectar, la información del bróker MQTT y el tópico a utilizar para transferir los datos. Además, se configura el cliente Wi-Fi para reconocimiento en el bróker MQTT. El código de esta configuración se muestra en la Figura 21.

```
const char* ssid = "AVILES_CNT";
const char* password = "javyavi.9";
const char* mqtt_server = "broker.emqx.io";
const char* mqtt_topic = "flowmeter/data";

WiFiClient rfidLector;
PubSubClient client(rfidLector);
```

Figura 21. Código para declaración de credenciales y cliente Wi-Fi

Posteriormente, para definir los tags válidos que se van a utilizar se define una lista con los códigos únicos hexadecimales registrados, esta lista de códigos se muestran en la Figura 22.

```
// Lista de IDs de tarjetas registradas
const String UserReg_1 = "82D0AC29";
const String UserReg_2 = "23368459";
const String UserReg_3 = "07763D3B";
```

Figura 22. Código para registro de los códigos de los TAGs válidos

Como siguiente paso, para identificar los TAGs se conecta al bróker MQTT en el puerto definido (1883), se comprueba si hay una tarjeta nueva sobre el lector del módulo, y de ser el caso, la identifica; lee su información, verifica si el código de la tarjeta está presente en el listado registrado, y de ser cierto que está, publica el código único de identificación en formato hexadecimal en el tópico del bróker. Esta configuración se la muestra en la Figura 23.

```
// Conectar al servidor MQTT
client.setServer(mqtt_server, 1883);

if (!rfid.PICC_IsNewCardPresent()) {
  return;
}

if (!rfid.PICC_ReadCardSerial()) {
  return;
}

Serial.print(F("PICC type: "));
MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid.sak);
Serial.println(rfid.PICC_GetTypeName(piccType));

if (piccType != MFRC522::PICC_TYPE_MIFARE_MINI &&
    piccType != MFRC522::PICC_TYPE_MIFARE_1K &&
    piccType != MFRC522::PICC_TYPE_MIFARE_4K) {
  Serial.println("Su Tarjeta no es del tipo MIFARE Classic.");
  return;
}
```

```
if (rfid.uid.uidByte[0] != nuidPICC[0] ||
    rfid.uid.uidByte[1] != nuidPICC[1] ||
    rfid.uid.uidByte[2] != nuidPICC[2] ||
    rfid.uid.uidByte[3] != nuidPICC[3]) {

  Serial.println("Se ha detectado una nueva tarjeta.");

  // Almacenar NUID en el arreglo nuidPICC
  for (byte i = 0; i < 4; i++) {
    nuidPICC[i] = rfid.uid.uidByte[i];
  }

  String DatoHex = printHex(rfid.uid.uidByte, rfid.uid.size);
  Serial.print("Codigo Tarjeta: ");
  Serial.println(DatoHex);
// Verificar si el ID de la tarjeta está en la lista de IDs registrados
if (UserReg_1 == DatoHex ||
    UserReg_2 == DatoHex ||
    UserReg_3 == DatoHex) {
  // Publicar el ID de la tarjeta en el tópico MQTT
  if (client.publish(mqtt_topic, DatoHex.c_str())) {
    Serial.println("ID de tarjeta publicado correctamente");
  } else {
    Serial.println("Error al publicar el ID de la tarjeta");
  }
}

delay(1000); // Esperar un segundo para permitir que la tarjeta se retire completamente

Serial.println();
```

Figura 23. Código para administración de los códigos únicos identificadores de los TAGs

2.3.3.2 Implementación del código fuente para la “Placa control de flujo”

En la implementación de código, primeramente, declaramos las librerías que serán herramientas para la funcionalidad de la placa.

```
#include <PubSubClient.h>
#include <time.h>
#include <ESP8266WiFi.h>
```

Figura 24. Código de librerías declaradas en la placa control de flujo

Como se evidencia en la Figura 24, se ha incluido una nueva librería diferente (*time*.h) a las ya detalladas en la sección anterior del código de la placa de identificación.

La librería *time*.h brinda la funcionalidad de temporizador y funciones de sincronización de fecha y hora con *Network Time Protocol* (NTP, por sus siglas en inglés).

Seguidamente se define las credenciales de conexión con el bróker, estas credenciales resultan ser las mismas que las credenciales de la placa de identificación del TAG.

Posteriormente, al igual que la placa anterior se configura un nuevo cliente Wi-Fi para identificación con el bróker MQTT, cabe recalcar que, para no tener conflictos de identidad de clientes en el bróker, a cada cliente se le debe asignar un nombre único. En este caso hemos nombrado al cliente publicador como esp8266ClientDisp y lo evidenciamos en la Figura 25.

```
WiFiClient esp8266ClientDisp;
PubSubClient client(esp8266ClientDisp);
```

Figura 25. Código para configuración de cliente placa control de flujo

A continuación, para la configuración del sensor de flujo en el código fuente se define el pin de señal que ha sido asignado al pin D5, se define un *buffer* de tamaño 256 caracteres para almacenar los mensajes que se publican en el bróker MQTT y finalmente se declaran las variables necesarias para el conteo de pulsos, conteo de tiempo, constante de calibración del sensor, tasa de flujo y la cantidad de volumen de líquido que fluye a través del sensor. Todo esto evidenciado en la Figura 26.

```
// Sensor de flujo
#define SENSOR D5  // Pin del sensor de flujo nodeMCU
#define BUFFER_LEN 256
char msg[BUFFER_LEN];

long milisegActual = 0;
long milisegPrevios = 0;
int interval = 1000;
float factorCalibracion = 18.5;
volatile byte pulseCount;
byte pulse1Sec = 0;
float tasaFlujo;
unsigned long flujoMililtrs;
unsigned int totalMililtrs;
```

Figura 26. Código para configuración del sensor de flujo

Seguidamente, para determinar el volumen que pasa por el sensor de flujo, se debe detectar el número de pulsos emitidos por el sensor, para ello, primero se debe guardar este número de pulsos en una variable y alojarlos en la memoria RAM del ESP8266, para luego transformarlo en una medida de volumen. Este proceso se logra con la función llamada *pulseCounter*, evidenciado en la Figura 27.

```
void ICACHE_RAM_ATTR pulseCounter() {
  pulseCount++;
}
```

Figura 27. Función *pulseCounter*

Para garantizar un acceso rápido a los datos de pulso se almacenó los mismos en la memoria RAM para ello se declaró junto al método la palabra reservada ICACHE_RAM_ATTR.

Posteriormente, para convertir los pulsos en medida de volumen se configuró una interrupción de conteo de pulsos en un pin digital específico. Lo detallado anteriormente junto con la configuración del cliente MQTT y la configuración de la hora desde un servidor NTP se lo evidencia en la Figura 28:

```
void setup() {
  Serial.begin(9600);
  setup_wifi();
  pinMode(SENSOR, INPUT_PULLUP);
  pulseCount = 0;
  tasaFlujo = 0.0;
  flujoMililtrs = 0;
  totalMililtrs = 0;
  milisegPrevios = 0;

  attachInterrupt(digitalPinToInterrupt(SENSOR), pulseCounter, FALLING);
  client.setServer(mqtt_server, 1883);
  configTime(0, 0, "pool.ntp.org");
}
```

Figura 28. Código para configuración de pulsos, cliente MQTT y servidor NTP

Después, para poder calcular el flujo que pasa por el sensor, se hizo uso del bucle principal del programa (*void loop*), en este bucle se almacena los intervalos de tiempo transcurridos desde el inicio del programa, el número de pulsos detectados desde el último cálculo de flujo, la tasa de flujo, y con esto, se calcula el volumen total dispensado en el intervalo de tiempo.

Finalmente, para publicar los datos en el bróker se estructuró los mismos en un mensaje de formato JSON que incluye los datos de identificación de la placa, el volumen de líquido dispensado y los datos de fecha y hora en que se realizó el dispenso, para publicarlos en el bróker, en la Figura 29 se tiene la evidencia respectiva.

```
void loop() {
  milisegActual = millis();
  if (milisegActual - milisegPrevios > interval)
  {
    pulse1Sec = pulseCount;
    pulseCount = 0;

    // Calcula el flujo
     tasaFlujo = ((1000.0 / (millis() - milisegPrevios)) * pulse1Sec) / factorCalibracion;
    milisegPrevios = millis();

    // Divide el flujo en mililitros por minuto
    flujoMililtrs = (tasaFlujo / 60) * 1000;

    // Añade los mililitros pasados en este segundo al total acumulado
    totalMililtrs += flujoMililtrs;
    // Publica el valor del flujo en el tópico MQTT
    if (!client.connected()) {
      client.connect("esp8266ClientDisp");
    }
    client.loop();
    String medidaTotal = String(totalMililtrs);
    String timestamp = getTimestamp();
    snprintf(msg, BUFFER_LEN, "{\"placa_Id\" : \"Llave 1\", \"Consumo\" : %s, \"FechaHora\" : \"%s\"}",
     medidaTotal.c_str(), timestamp.c_str());
    client.publish(mqtt_topic, msg);
    Serial.print("disp.val=");
    Serial.print(medidaTotal);
    Serial.write(0xff);
    Serial.write(0xff);
    Serial.write(0xff);
  }
}
```

Figura 29. Código del bucle principal para el cálculo del volumen dispensado y publicación de datos en el bróker

En resumen, el código implementado permitió que las placas actúen como un cliente publicador MQTT y que los datos sensados se publiquen en el bróker dentro del tópico llamado *flowmeter/data*.

2.3.4 Configuración del bróker

La elección de utilizar EMQX bróker fue debido a los resultados positivos en los criterios de evaluación presentados en [31], como característica principal se tiene la adaptabilidad a este proyecto y también debido a que en su versión web en la capa gratuita ofrece una interfaz gráfica de fácil uso, esta permite administrar y monitorear instancias del bróker, se tiene acceso desde cualquier terminar web, un aspecto relevante y destacable es que en la configuración web ofrece una administración de clientes MQTT y tópicos [32]. Para la configuración del bróker EMQX se realizaron los siguientes pasos:

2.3.4.1 Configuración de parámetros del bróker en la versión web

Para comenzar a utilizar el servicio del bróker mediante una terminal web, primeramente, fue necesario acceder a su sitio web oficial (emqx.com). En el menú principal ubicado en la parte superior de la pantalla de inicio se encuentra la sección *Plataform*, al desplazar el cursor sobre esta sección observamos el servicio llamado MQTTX, mostrado en la Figura 30.

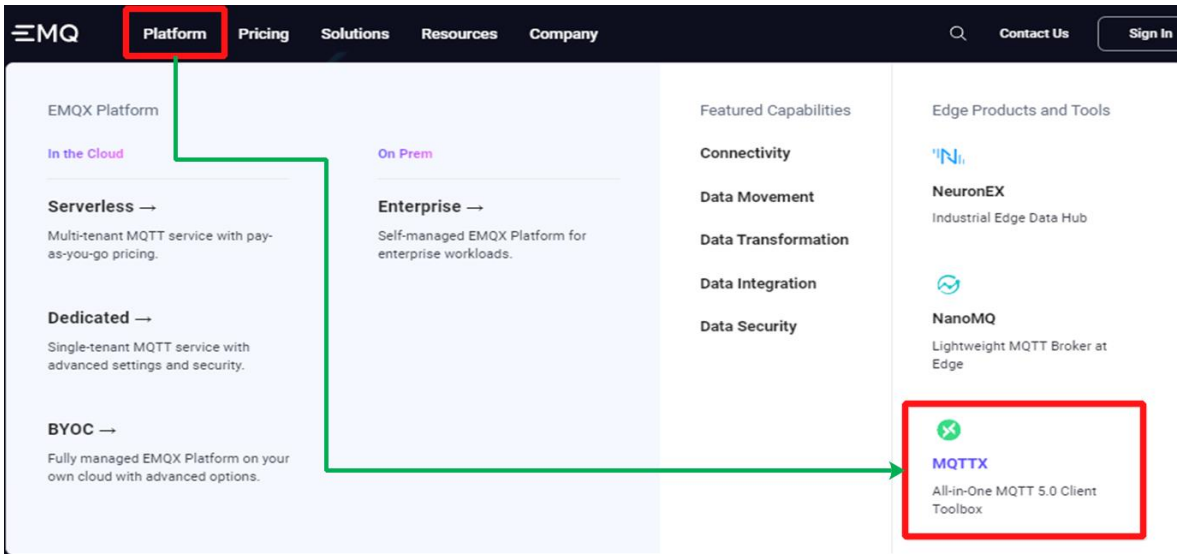

Figura 30. Menú de servicios *Plataform* de EMQX [33]

Seguidamente, al acceder al servicio de MQTTX, se nos mostró una interfaz intermedia para elegir la versión a utilizar (web o *desktop*) ubicada en el menú principal, en la cual se seleccionó la versión web. Esto se visualiza de la siguiente Figura 31.

Figura 31. Menú de servicios EMQTTX [34]

Posteriormente, se procede con el registro del usuario que utilizará los servicios, como se muestra en el formulario de la Figura 32.

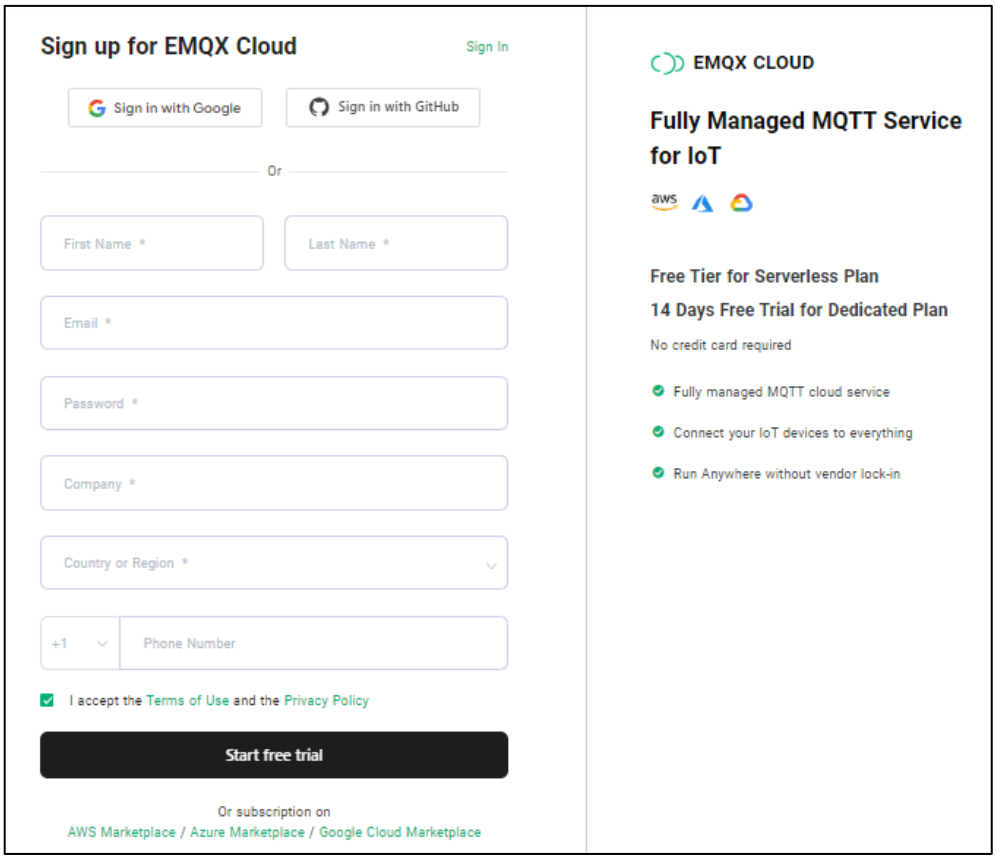

Figura 32. Formulario de registro de usuario de EMQTTX [34]

Concluido el registro y aceptados los términos y condiciones de uso, se despliega la interfaz del bróker. Esta interfaz consta de dos paneles: un panel lateral izquierdo donde se tiene la información de conexiones, íconos de acceso a conexiones, la opción para crear una nueva conexión, ajustes, documentación acerca de MQTT y documentación de la aplicación web de MQTTX; el panel lateral derecho se usa para visualizar los mensajes de los publicadores y consta de una terminal que hace las veces de publicador (en caso de requerirlo). Todo esto se lo puede evidenciar mediante la Figura 33.

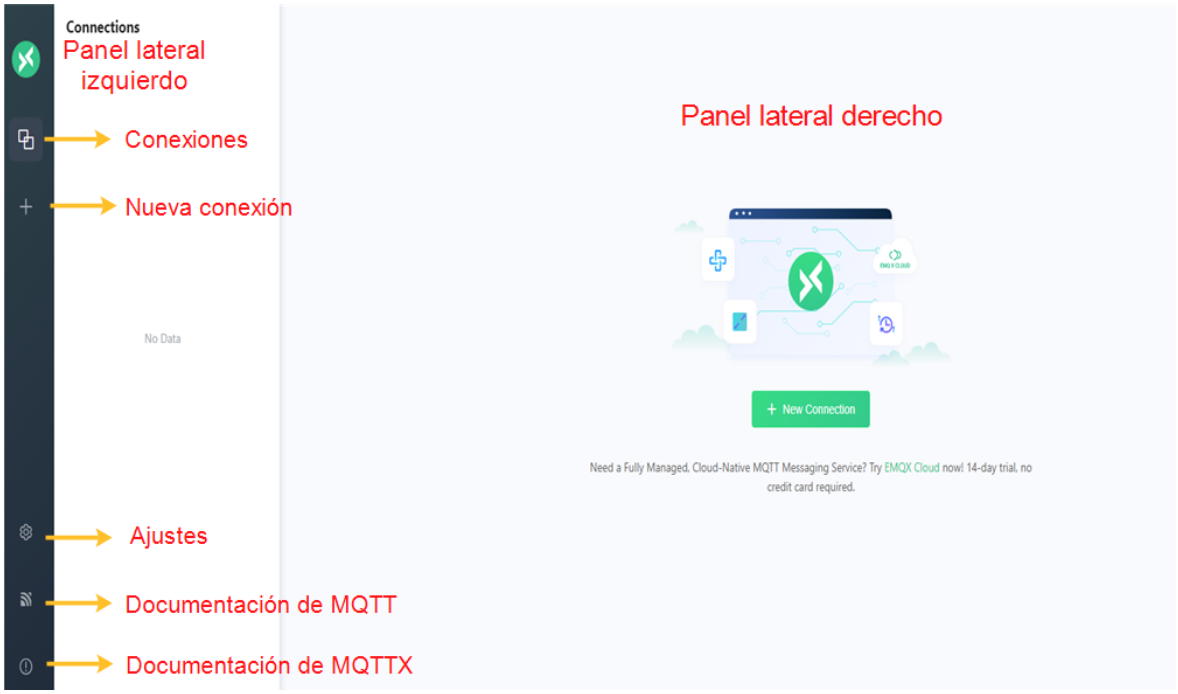

Figura 33. Interfaz de administración del bróker [34]

Posteriormente, ya en la interfaz de administración, se necesita crear una nueva conexión, para lo cual en primera instancia, el formulario exige la información mostrada en la Figura 34:

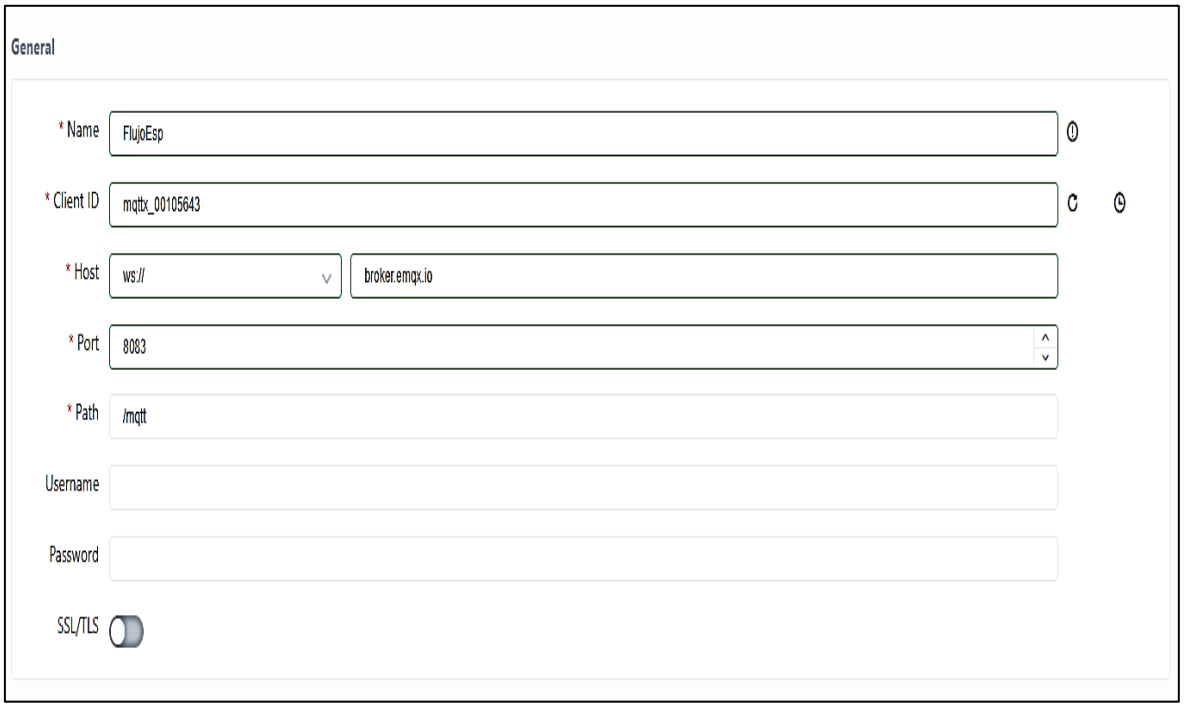

Figura 34. Formulario de información general para configuración del bróker [34]

En este formulario de información general se debe llenar de forma obligatoria el campo de nombre, este campo es dedicado al nombre de la conexión (no acepta nombres de conexiones duplicadas). Por defecto los capos de *Client* ID, *Host*, *Port*: 8083 (Al optar por la elección del puerto 8083 estamos dando paso para usar el protocolo MQTT con la apertura de *WebSockets* [35]) y *Path* se auto completan con su información correspondiente.

Por otra parte los campos de *Username* y *Password* no necesariamente se los debe llenar, al menos que se requiera una autenticación. Finalmente el botón *toggle switch* referente a SSL/TLS no necesita ser activado.

Cabe recalcar que para el componente de esta investigación no se requiere la autenticación de usuarios y tampoco el uso del protocolo criptográfico para conexión segura (SSL/TLS), ya que el ambiente en el que se desarrolló y se probó el dispositivo IoT es un ambiente controlado y no está comprometida la seguridad de la información.

Para finalizar la configuración, dentro del formulario se debe limitar a los ajustes por defecto realizados por la aplicación, esto se evidencia en la Figura 35.

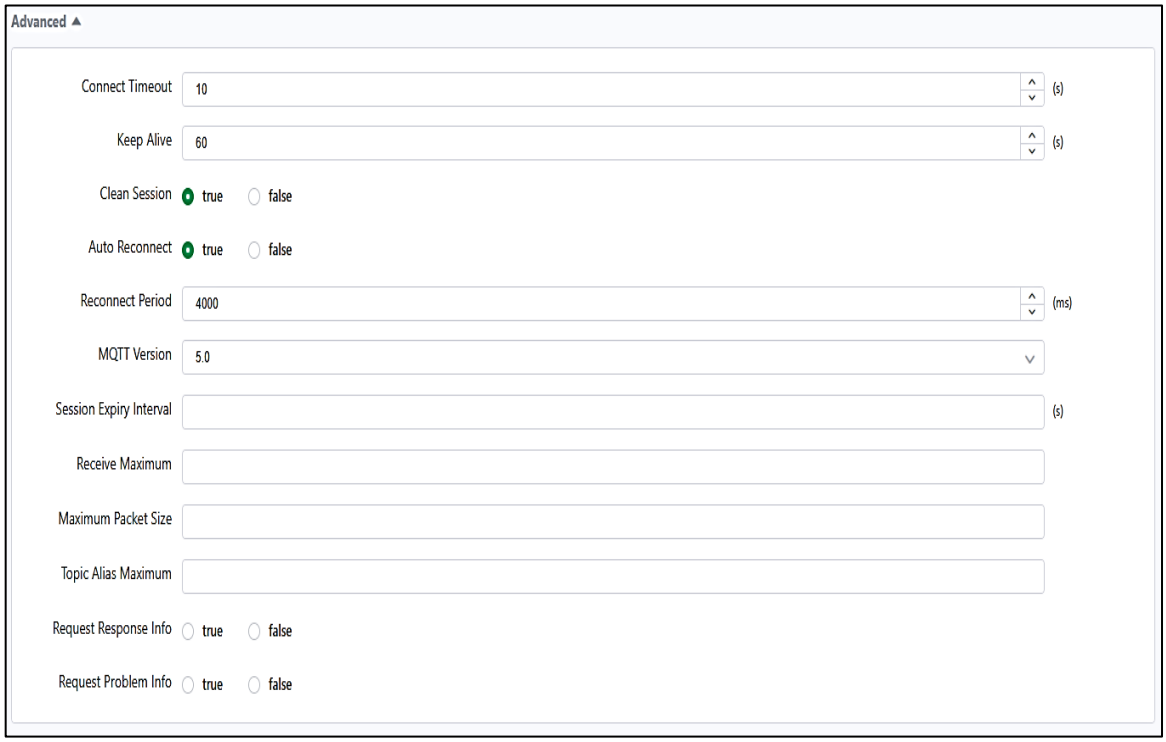

Figura 35. Formulario de información avanzada para configuración del bróker [34]

Con estas configuraciones creadas por defecto la sesión de conexión se mantiene activa y cada vez que se ingrese a la interfaz de administración del bróker se auto conectará a la conexión guardada.

De manera seguida, para configurar la calidad de servicio (QoS) en el nivel 0, que nos garantiza integridad de datos y la entrega del mensaje como máximo una vez a todos los suscriptores sin reintentos en caso de una falla, se ajustó en el formulario *Last Will* and *Testament* los parámetros, tal y como se lo muestra en la Figura 36.

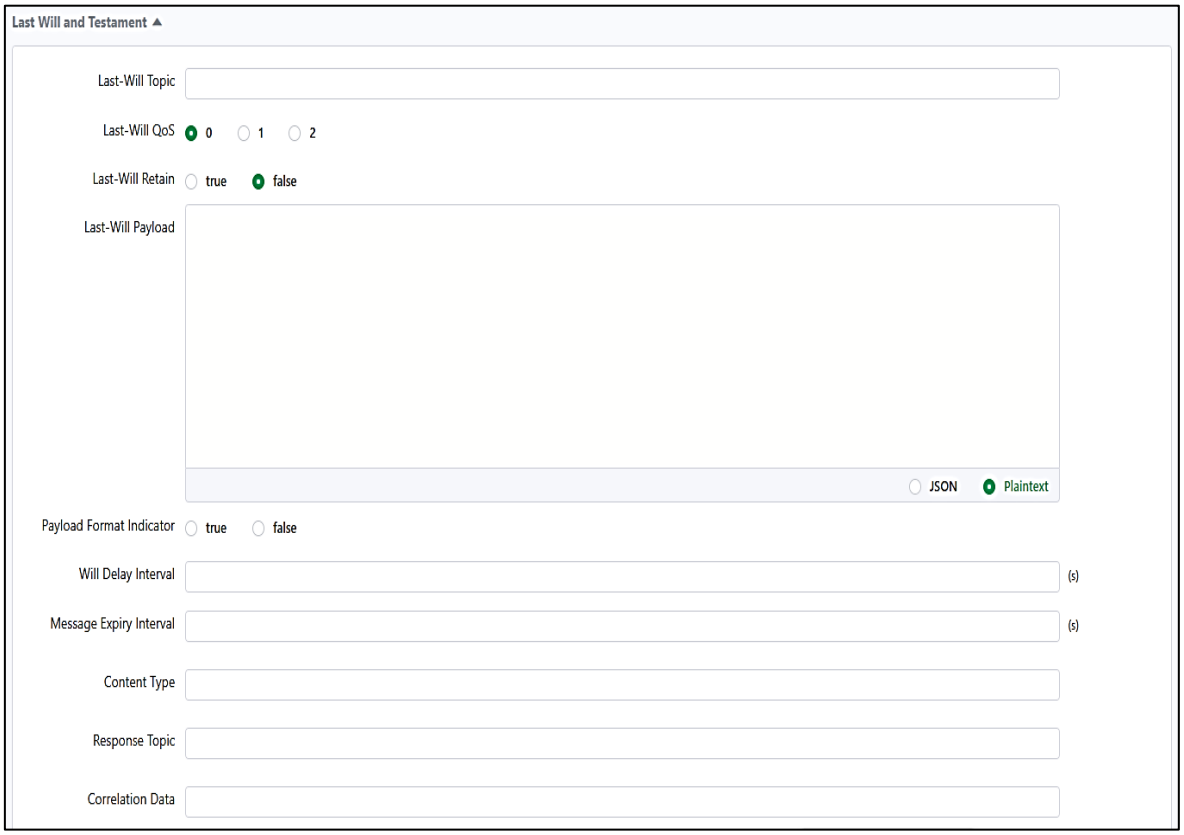

Figura 36. Formulario de *Last Will and Testament* para configuración del bróker [34]

A continuación, para concluir con la configuración del bróker se necesita añadir una nueva suscripción a un determinado tópico, para ello se llenó los campos del formulario *New Subscription,* tal y como se lo muestra en la Figura 37:

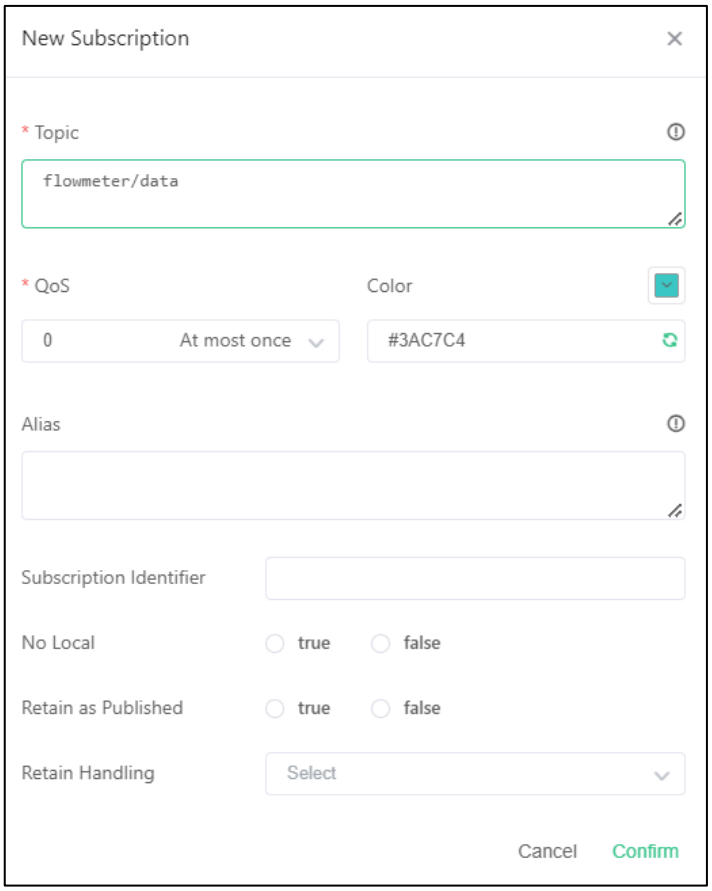

Figura 37. Formulario de nueva suscripción [34]

Para el caso de esta investigación el nombre del tópico es nombrado como *flowmeter/data* y las demás configuraciones son ajustadas por defecto.

Este proceso se lo concluyó pulsando el botón de confirmación y de esta manera el bróker quedó configurado para poder administrar publicaciones, mensajes y suscripciones.

Finalmente, para realizar la comprobación de la conexión entre el bróker y el dispositivo se compiló el código fuente en las placas NodeMCU y se realizó las conexiones correspondientes tanto en cablería en el dispositivo como en el simulador del autoservicio, es de total relevancia mencionar que el dispositivo IoT debe conectarse a una fuente de poder de capacidad mínima de cinco voltios y máxima de nueve voltios para funcionar con normalidad. Como resultado de la conexión exitosa entre este conjunto de dispositivos, tecnologías de comunicaciones y el autoservicio, se evidenció la toma de datos de prueba sin volumen dispensado mediante un mensaje de formato JSON mostrado en la consola del bróker.

Figura 38. Consola de administración de mensajes del bróker

Como se muestra en la Figura 38, el mensaje que llegó desde el dispositivo IoT como publicador hacia el tópico en el bróker (nombrado como *flowmeter/data)*, se lo mostró en la consola del bróker, este mensaje contiene los datos de la placa y el dispenso de volumen igual a cero.

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 Pruebas y evaluación del dispositivo IoT

Para verificar los resultados del sensado de flujo y datos se realizó pruebas, para esto fue necesario configurar un ambiente en el cual se pueda simular el autoservicio de bebidas calientes y su correspondiente dispenso de bebidas. Para poder simular el autoservicio se utilizó diferentes materiales a fin de captar los datos de flujo de diferentes bebidas, el ambiente donde se realizó la implementación del dispositivo IoT para captar los datos de flujo del autoservicio se describe a continuación.

3.1.1 Descripción del ambiente para simular un autoservicio de bebidas calientes

Este ambiente de simulación se lo configuró de tal manera que este sea un ambiente controlado, ya que se encuentra ubicado dentro de un departamento, específicamente en la sala. Aquí se apartó un espacio para armar el autoservicio y conectarlo al dispositivo IoT. Cercano a este espacio (a cuatro metros de distancia), se encuentra el *Router* al que el dispositivo IoT se conectará por medio de una red inalámbrica Wi-Fi. Es importante resaltar que la conexión inalámbrica entre las placas y una red Wi-Fi no debe superar los quince metros en diagonal, caso contrario la comunicación se vería afectada.

La Figura 39 describe la representación física del ambiente.

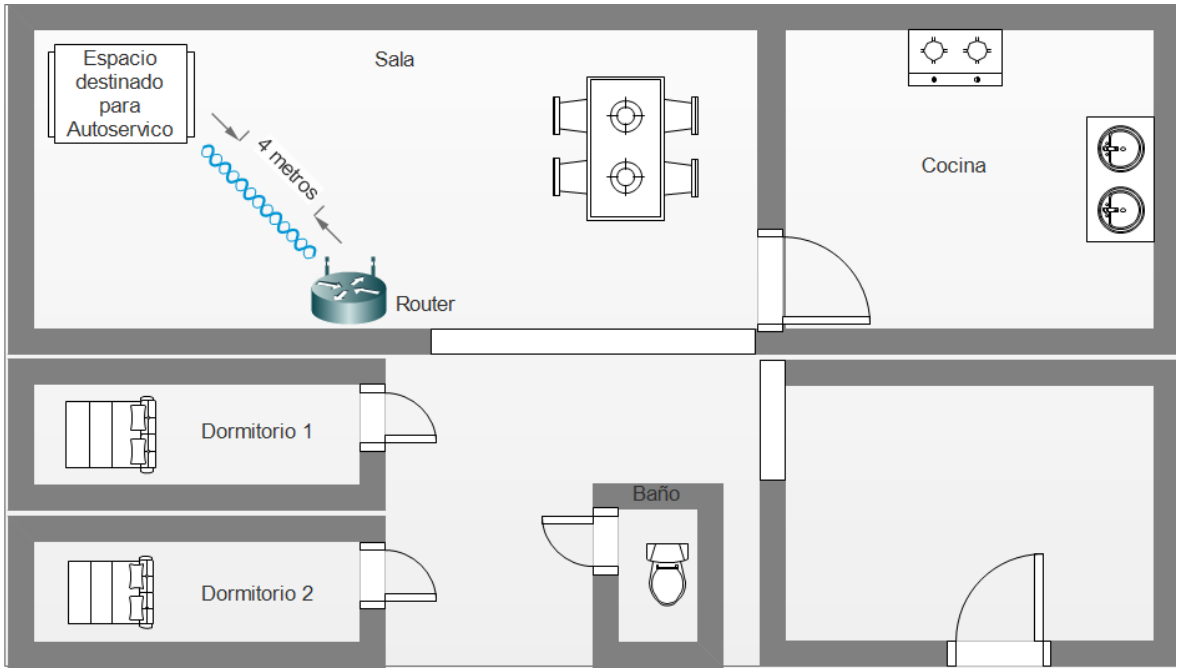

Figura 39. Disposición física del ambiente de simulación

Para simular el autoservicio de bebidas se ha usado y configurado los materiales descritos en la siguiente tabla:

Tabla 7. Materiales y configuraciones para simular un autoservicio de bebidas

Material	Función	Configuración
Tanque de CO_2	Generar presión en el tanque de líquido para expulsarlo hacia el grifo	El tanque debe mantenerse a una medida no menor a 5% de su máxima capacidad
Tanque contenedor de líquidos	Contener la bebida para dispersión	El contenedor debe mantenerse a una medida no menor a 3% de su máxima capacidad
Manguera de líneas	Llevar a su destino tanto el líquido de la bebida como el CO_2	En su configuración debe ser de grado alimenticio y de baja presión
Regulador de CO_2	Regular la presión de CO_2 para que el líquido de la bebida fluya normalmente hacia el grifo	Debe mantenerse en el umbral de 8 a 15 PSI de presión
Grifo dispensador	Dispensar el líquido proveniente del tanque contenedor de líquido	Grifo destinado para dispenso de bebidas de grado alimenticio y alcohólico
Cabezal de acople rápido	Gestionar el ingreso de CO_2 al tanque contenedor de líquidos y la salida del líquido hacia el grifo	Debe ser de tipo A debido al tanque utilizado
Dispositivo IoT	Sensar el líquido que dispensa el tanque de líquidos, para ligarlo a un cliente	Calibrado para medir agua carbonatada

En la Figura 40 se constata el autoservicio configurado y con sus respectivas conexiones dentro del ambiente destinado para la simulación.

Figura 40. Autoservicio conectado al dispositivo IoT

3.1.2 Prueba de funcionamiento del dispositivo IoT

Como se puede apreciar en la Figura 41, el dispositivo IoT al estar conectado a una fuente de poder y a los cables provenientes del sensor de flujo, el botón led del módulo RFID se enciende, pudiendo de esta manera constatar que el dispositivo IoT está apto para sensar el ingreso de líquido y dispuesto para tomar los datos de identificación de los TAGs.

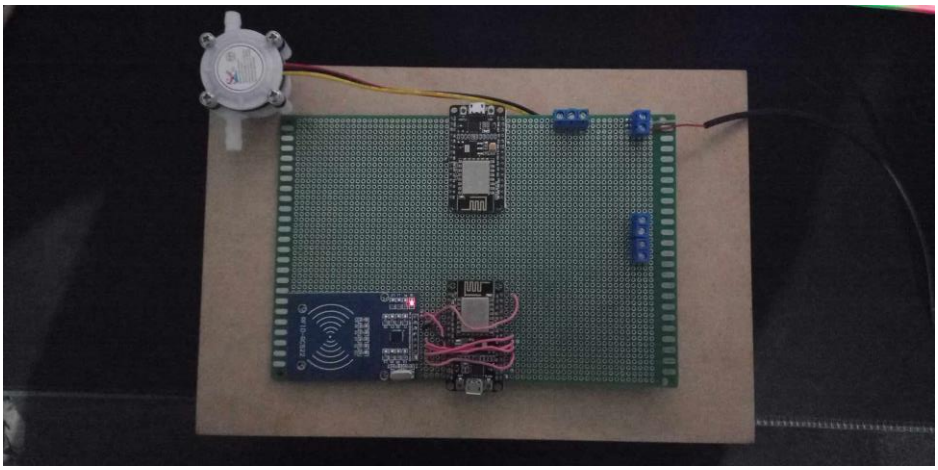

Figura 41. Dispositivo IoT en funcionamiento

3.1.3 Prueba de medición de flujo y datos

Para evidenciar el resultado de la medición de flujo de líquidos, en primera instancia se verificó la conexión de los cables de señal, tierra y voltaje a los bornes correspondientes de la placa de prototipado, seguidamente se realizó una simulación del flujo de líquido. En este caso, la simulación se efectuó utilizando aire proveniente del soplo de una persona. Este procedimiento se llevó a cabo con el objetivo de mover el rotor de efecto *Hall* del medidor de flujo, siendo necesario para poder eliminar cualquier residuo que pueda encontrarse en su interior. De esta manera, se garantizó que el sensor funcione normalmente al momento de medir el flujo real. El resultado de este proceso se evidencia en un video cuyo enlace es presentado en el ANEXO II. La Figura 42 captura evidencia del resultado.

Figura 42. Prueba de flujo de aire

Una vez confirmado el correcto funcionamiento del dispositivo IoT y asegurado un flujo de aire constante sin interrupciones, se procedió a probar el flujo de líquidos, específicamente de bebidas basadas en agua.

Esto se realizó para evaluar si el dispositivo puede cumplir con los objetivos planteados y para verificar la administración de datos a través del bróker, este proceso se evidencia mediante la Figura 43 y un enlace a un video presentado en el ANEXO III.

Figura 43. Prueba de flujo de agua

Para el registro de las pruebas se utilizó la siguiente tabla:

Tabla 8. Pruebas de dispenso del dispositivo IoT

Día	Número de horas conectado	Número de dispensos	Presenta fallos en la toma de datos	Presenta fallos en la configuración electrónica
06/07/2024	7	7	No	No
07/07/2024	6	6	No	No
08/07/2024	8	5	No	No
09/07/2024	12	5	No	No
10/07/2024	10	7	No	No
11/07/2024	10	7	No	No
16/07/2024	11	7	No	No
18/07/2024	8	7	No	No
21/07/2024	8	9	No	No
22/07/2024	7	5	No	No
23/07/2024	7	6	No	No
24/07/2024	7	8	No	No
25/07/2024	8	4	No	No
30/07/2024	10	8	No	No

Cabe recalcar que para realizar estas pruebas se empleó un barril lleno de agua carbonatada, este barril tiene una capacidad de cincuenta litros y el número total de dispensos fue de noventa y uno.

Según los datos obtenidos se puede verificar que la eficiencia fue del cien por ciento ya que no se presentó fallas en la toma de datos, mientras que la eficacia se la refleja mediante

el cociente entre el número total de dispensos y el número total de horas conectado, dando un valor de setenta y seis por ciento.

3.1.4 Prueba de medición de flujo con diferente tipo de bebida

Posterior a la prueba de sensado de flujo de líquidos basados en agua se hizo una adaptación al escenario más complicado de medición, este es un escenario en el cual la base de la bebida consta de una mayor densidad y viscosidad, Para el caso concreto se y utilizó leche y también chicha, ya que estos dos líquidos tienen medidas de densidad y viscosidad muy similares. Los valores de la composición se deben a sólidos disueltos y emulsionados, como proteínas, grasas, azúcares y minerales hallados en la leche y la chicha [36]. A continuación, se muestra una tabla comparativa entre propiedades de densidad y viscosidad entre la composición de agua y leche.

Tabla 9. Comparativa entre propiedades de leche y agua

Líquido	Densidad	Viscosidad
Agua	1.000 g/cm³	Entre 1.503 y 2.004 centipoise
Leche	Entre 1.028 y 1.034 g/cm³	1.002 centipoise

Mediante un simple cálculo matemático se puede evidenciar que la leche es alrededor de dos a tres por ciento (aproximadamente) más densa que el agua, esto para la medida de flujo significa una variación mínima, por lo cual se modifica el factor de calibración en máximo dos centésimas y se lo dispone de la misma manera en la que medimos las bebidas basadas en agua. El resultado del proceso de medición de la chicha se lo puede evidenciar mediante la Figura 44 y un enlace a un video presentado en el ANEXO IV.

Figura 44. Prueba de flujo de chicha

3.2 Conclusiones

1. En el ámbito nacional, la automatización del control y monitoreo del dispenso de bebidas calientes en autoservicios comerciales no es comúnmente abordada, este proceso se lo realiza de forma manual, comenzando con un dispenso en envases con medidas establecidas y finalizando con el cobro constatando el volumen dispensado en el envase del cliente. Esto ocasiona una experiencia de usuario ineficiente. En ese sentido, este proyecto proporciona una alternativa de solución tecnológica que permite mejorar el proceso operativo de dispenso de bebidas haciéndolo eficiente y con ello mejorando la experiencia del cliente en estos establecimientos.

2. Al adaptar el proyecto a la metodología de diseño centrado en el usuario y basarlo para su desarrollo en cuatro etapas, no solamente se aseguró que el dispositivo sea técnicamente funcional, sino también de uso fácil, simplificado y sobre todo que se adapte a las necesidades reales de los clientes, propietarios de los establecimientos comerciales y sus empleados.

3. Siguiendo las cuatro etapas de la metodología centrada en el usuario (análisis contextual, definición de los requisitos, diseño e implementación del dispositivo IoT y evaluación) se seleccionó la arquitectura de tres capas, esto fue debido a su simplicidad, modularidad, escalabilidad e interoperabilidad. Aquí se seleccionó dos capas de esta arquitectura (la capa de percepción y la capa de red) para diseñar e implementar el dispositivo IoT. Esta elección permitió una integración cohesiva de todos los componentes que conforman el dispositivo IoT y con esto se aseguró que cada capa cumpliera su función específica dentro del funcionamiento del dispositivo IoT.

4. Las capas de percepción y de red son transversales en la arquitectura de tres capas y su funcionamiento depende una de otra.

    4.1 La capa de percepción compuesta por las dos placas NodeMCU, conectadas al sensor de flujo y el módulo lector RFID respectivamente, conforman al dispositivo IoT. Este dispositivo IoT capta los datos de flujo y los números identificadores de los TAGs que se desplazan por el lector del RFID. El dispositivo IoT garantiza la recopilación de datos sensados en el dispenso de bebidas de los autoservicios, cumpliendo así su funcionamiento y consecuentemente con los objetivos del proyecto.

4.2 La capa de red está conformada por el bróker, este bróker es implementado bajo el protocolo MQTT mediante la plataforma EMQX y su comunicación con la capa de percepción se la hace a través de una red inalámbrica Wi-Fi. Su papel dentro de la arquitectura y específicamente dentro del proyecto es el del tratamiento de datos en un tópico (*flowmeter/data*) y la administración de los clientes suscriptores y publicadores a este tópico. El bróker EMQX garantizó la disponibilidad de datos para el cliente suscriptor representado por la aplicación web (desarrollada en el componente complementario de esta investigación).

5. Para asegurar un sensado de datos preciso y fiable, fue necesario verificar el buen estado y correcto funcionamiento de los sensores para posteriormente calibrarlos y consecuentemente poder obtener los datos sensados. Con este debido proceso de calibración los sensores empleados demostraron ser precisos y fiables, lo que significó una contribución significativa al rendimiento general del dispositivo evidenciado por la correcta conexión de los sensores y las placas sin presentar ningún tipo de inconveniente.

6. De la misma manera en que se verificó que los sensores estén en buen estado, también se procedió a la verificación del estado correcto de las placas y su normal funcionamiento. Como consecuencia de una buena operabilidad de las placas y los sensores ya calibrados, se conectó las placas y los sensores y se aseguró el correcto funcionamiento en conjunto de estos. Como fruto de este ensamble y conexión exitoso entre placas y sensores se obtuvo la comunicación entre componentes. Esto se verificó por medio del reconocimiento de las placas en entorno de desarrollo de Arduino.

7. Mediante el entorno de desarrollo de Arduino se desarrolló el código para implementarse a nivel de software en las placas, este desarrollo de código se lo realizó siguiendo buenas prácticas de desarrollo de software, con el objetivo de poder obtener un código bien estructurado y que presente modularidad. Con esto se pudo asegurar para el correcto funcionamiento de las placas, lo que permitió una transmisión de datos precisa, eficiente y sin interrupciones.

8. Mediante el despliegue del dispositivo IoT y un autoservicio de bebidas simulado en un entorno controlado se realizó las pruebas con bebidas calientes típicamente comercializadas. Este entorno fue controlado debido a que se garantizó la conexión a una red Wi-Fi estable y sin interrupciones en el funcionamiento del dispositivo IoT,

bajo estas condiciones se constató los resultados en los cuales, el dispositivo IoT mide con precisión el flujo de diversas bebidas calientes basadas en agua, garantizando la integridad y transmisión completa de los datos del dispenso, el dispositivo no presentó fallos en la toma de datos ni en la configuración electrónica.

9. Al seguir una metodología de diseño centrado en el usuario, se tiene mucha consideración en las necesidades del usuario. La necesidad de adaptar el dispositivo IoT para que pueda medir diferentes tipos de bebidas calientes fue una de las principales necesidades, al utilizar la arquitectura de tres capas y siguiendo las cuatro etapas de la metodología UCD se aseguró que el dispositivo IoT sea escalable y que se puede emplear las mismas configuraciones para adaptarlo a medir el dispenso de bebidas de grado alcohólico, bebidas basadas en leche y bebidas que en su composición de viscosidad y densidad no afecten a el sensor de flujo, tomando siempre en consideración los factores de calibración de flujo para su configuración.

3.3 Recomendaciones

1. Se recomienda realizar la calibración de los sensores antes de su uso, siguiendo los parámetros específicos necesarios detallados en los *datasheets*, para asegurar su precisión y funcionamiento adecuado, esto con la finalidad de evitar el sensado de medidas incorrecto.

2. Durante la conexión de los sensores con las placas se produjo varios problemas en la captación de datos debido a cables sueltos o mal conectados, para evitar estos inconvenientes es indispensable trabajar siempre con cablería nueva o a su vez que esté en buen estado.

3. El sobre voltaje puede provocar que las placas se quemen, para evitar este inconveniente, es crucial familiarizarse con la documentación proporcionada por los fabricantes mediante el *datasheet*. Una manera en la cual se puede prevenir este sobre voltaje es revisando cuidadosamente el voltaje máximo de operación especificado para las placas y asegurándonos de no exceder ese límite. También se puede emplear el uso de fusibles, disyuntores y protección contra sobre carga de voltaje para evitar el daño de los componentes electrónicos.

4. En el proceso de desarrollo del código fuente para el funcionamiento del dispositivo IoT, surgieron varios obstáculos debido a las librerías utilizadas para los métodos y funciones. Para evitar estos inconvenientes, es importante emplear librerías

certificadas por entidades confiables, que estén disponibles en repositorios públicos y sean recomendadas en el entorno de desarrollo integrado (IDE) que se está utilizando.

5. Si se está trabajando con la plataforma EMQX, se recomienda consultar la documentación proporcionada por sus desarrolladores para utilizar su servicio de bróker alojado en la nube. Es fundamental tener precaución con los datos almacenados en el bróker, especialmente teniendo en cuenta que en su versión gratuita existe un límite de tiempo después del cual los datos pueden desaparecer.
6. Es recomendable explorar la posibilidad de ampliar el uso del dispositivo IoT para la medición de flujo en otros tipos de establecimientos y con diferentes líquidos, no solo bebidas calientes. Esto podría incluir bebidas alcohólicas carbonatadas como cerveza o basadas el alcohol etílico, esto a fin de tener nuevos servicios.
7. Para adaptar el dispositivo IoT a diferentes tipos de bebidas, se recomienda estudiar la estructura y composición de estas, ya que algunas bebidas presentan residuos sólidos y estos pueden ocasionar la obstrucción y daño del sensor de flujo.

REFERENCIAS BIBLIOGRÁFICAS

[1] T. Lowdermilk, User-Centered Design: A Developer's Guide to Building User-Friendly Applications, Sebastopol: O'Reilly Media Inc. , 2013.

[2] Somayya Madakam, R. Ramaswamy, Siddharth Tripathi, «Internet of Things (IoT): A Literature,» *Journal of Computer and Communications,* vol. 12, nº 5, p. 10, 2015.

[3] Matthew Gigli, Simon Koo, «Internet of Things: Services and Applications Categorization,» *Advances in Internet of Things,* vol. 1, nº 2, p. 5, 2011.

[4] Mohd Muntjir, Mohd Rahul, Hesham A. Alhumyani, «An Analysis of Internet of Things(IoT): Novel Architectures, Modern Applications,Security Aspects and Future Scope with Latest Case Studies,» *International Journal of Engineering Research & Technology (IJERT),* vol. 6, nº 6, p. 28, 2017.

[5] Muhammad Burhan, Rana Asif Rehman, Bilal Khan y Byung-Seo Kim , «IoT Elements, Layered Architectures and Security Issues: A Comprehensive Survey,» *Sensors,* vol. 18, nº 9, p. 2796, 2018.

[6] Habib Larian, Ali Larian, Mahdi Sharifi y Homa Movahednejad, «Towards Web of Things Middleware: A Systematic Review,» *arXiv preprint arXiv,* 2022.

[7] Ayoub Benayache, Azeddine Bilami, Sami Barkat, Pascal Lorenz y Hafnaoui Taleb, «MsM: A microservice middleware for smart WSN-based IoT application,» *Journal of Network and Computer Applications,* vol. 144, pp. 138-154, 2019.

[8] Surapon Kraijak y Panwit Tuwanut, «A survey on IoT architectures, protocols, applications, security, privacy, real-world implementation and future trends,» *in Proceedings of the 11th International Conference on Wireless Communications,* pp. 1-6, 2015.

[9] Tara Salman y Raj Jain, Internet of Things and Data Analytics Handbook, Hwaiyu Geng, 2016.

[10] R A Atmoko, R Riantini, & M. K. Hasin, «IoT real time data acquisition using MQTT protocol,» *Journal of Physics: Conference,* vol. 853, nº 1, p. 012003, 2017.

[11] Fahd A. Alhaidari y Ebtesam J. Alqahtani, «Securing Communication between Fog Computing and IoT Using Constrained Application Protocol (CoAP): A Survey,» *Journal of Communications,* vol. 15, nº 1, pp. 14-30, 2020.

[12] B.B. Gupta y Megha Quamara, «An overview of Internet of Things (IoT): Architectural aspects, challenges, and protocols,» *Concurrency and Computation: Practice and Experience,* vol. 32, nº 21, p. e4946, 2020.

[13] Espressif, «Datasheet NODEMCU,» 2020.

[14] Y. S. Parihar, «Internet of Things and Nodemcu: A review of use of Nodemcu ESP8266 in IoT products,» *Journal of Emerging Technologies and Innovative Research,* vol. 6, nº 6, p. 6, 2019.

[15] Components101, «components101.com,» components101.com, 01 01 2023. [En línea]. Available: https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet. [Último acceso: 25 06 2024].

[16] N. N. &. A. Supriyono, «Rancang Bangun Sistem Pengisian Air Menggunakan Sensor YF-S401 Berbasis HMI,» *Jurnal Ilmiah Elektrokrisna,* vol. 8, nº 3, p. 9, 2020.

[17] *YF-S401 Datasheet,* 2020.

[18] N. Semiconductors, «MFRC522: Standard performance MIFARE and NTAG frontend,» NXP Semiconductors, 2016.

[19] B. I. J. C. Saurin, «Experiencia de consumo y notoriedad de marca en clientes de la empresa Pepas Bar,» *Universidad César Vallejo,* vol. 1, nº 1, p. 66, 2019.

[20] M. y. K. N. Escalona Cuaresma, «Ingeniería de Requisitos en Aplicaciones para la Web: Un estudio comparativo,» Depósito de investigación: Universidad de Sevilla, Sevilla, 2022.

[21] R. R. N. G. y. S. B. S. Madakam, «The key layers of IoT architecture,» *IEEE: 5th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTech),* vol. 1, nº 1, p. 4, 2020.

[22] Nallapaneni Manoj Kumar y Pradeep Kumar Mallick, «The Internet of Things: Insights into the building blocks, component interactions, and architecture layers,» *Procedia Computer Science,* vol. 132, pp. 109-117, 2018.

[23] D. H. &. D. H. P. Desti Fitria Rahmawati, «Development of Water Distribution and Water Volume Practicum Tools Using the YF-S401 Flow Sensor on Dynamic Fluid Materials in Senior High School,» *Jurnal Pendidikan Matematika Dan Ipa,* vol. 14, nº 2, p. 15, 2023.

[24] M. G. M. M. M. A. &. M. A. Ala Al-Fuqaha, «Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications,» *IEEE COMMUNICATION SURVEYS & TUTORIALS,* vol. 14, nº 4, pp. 2347 - 2376, 2015.

[25] K. R. K. J. B. J. K. S. S. T. V. &. C. Y. E. Al-Masri, «Investigating Messaging Protocols for the Internet of Things (IoT),» *IEEE Access,* vol. 8, pp. 94880-94911, 2020.

[26] A. M. N. S. y. S. M. S. Elhadi, «Comparative Study of IoT Protocols,» *SSRN Electronic Journal,* vol. 1, 2018.

[27] Pongnapat Jutadhamakorn, Tinnapat Pillavas, Vasaka Visoottiviseth, Ryousei Takano, Jason Haga y Dylan Kobayashi, «A scalable and low-cost MQTT broker clustering system,» *2nd International Conference on Information Technology (INCIT),* pp. 1-5, 2017.

[28] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari y Moussa Ayyash, «Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications,» *IEEE Communication Surveys & Tutorials,* vol. 17, nº 4, pp. 2347 - 2376, 2015.

[29] Arduino, «Arduino-esp8266.readthedocs.io,» 01 01 2017. [En línea]. Available: https://arduino-esp8266.readthedocs.io/en/latest/libraries.html#wifi-esp8266wifi-library. [Último acceso: 20 Junio 2024].

[30] N. O'Leary, «github.com,» 2020. [En línea]. Available: https://github.com/knolleary/pubsubclient/tree/master. [Último acceso: 20 Junio 2024].

[31] Melvin Bender, Erkin Kirdan, Marc-Oliver Pahl y Georg Carle, «Open-Source MQTT Evaluation,» de *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, Las Vegas, NV, USA, 2021.

[32] E. T. Inc, «EMQX docs,» 2013. [En línea]. Available: https://docs.emqx.com/en/emqx/latest/. [Último acceso: 26 Junio 2024].

[33] E. T. Inc., «Emqx.com,» 2013. [En línea]. Available: https://www.emqx.com/en. [Último acceso: 26 Junio 2024].

[34] E. T. Inc., «Mqttx.app,» 2013. [En línea]. Available: https://mqttx.app/. [Último acceso: 26 Junio 2024].

[35] HiveMQ, «HiveMQ.com,» [En línea]. Available: https://www.hivemq.com/blog/understanding-the-differences-between-mqtt-and-websockets-for-iot/. [Último acceso: 28 Junio 2024].

[36] Eric Montes de Oca-Flores, Angélica Espinoza-Ortega y Carlos Manuel Arriaga-Jordán, «Propiedades tecnológicas y fisicoquímicas de la leche y características fisicoquímicas del queso Oaxaca tradicional,» *Revista mexicana de ciencias pecuarias,* vol. 10, nº 2, 2019.

[37] L. Carvajal, Metodología de la Investgación Científica. Curso general y aplicado, 28 ed., Santiago de Cali: U.S.C., 2006, p. 139.

[38] F. B. Z. Omar Enrique Barra Zapata, Microcontroladores PIC con programación PBP, Madrid: Ra-Ma, 2011.

[39] A. O. e. Y. Pigneur, Generación de modelos de negocio: Un manual para visionarios, revolucionarios y retadores, Barcelona: Centro Libros PAPF, S. L. U., 2011.

[40] Redacción El Universo, «¿Cuánto café consumen los ecuatorianos y cuáles son sus favoritos?,» *El Universo,* p. 1, 30 Septiembre 2023.

[41] Arduino, «Arduino.cc,» 2024. [En línea]. Available: https://www.arduino.cc/reference/en/libraries/time/. [Último acceso: 22 Junio 2024].

[42] Bhagya Nathali Silva, Murad Khan y Kijun Han, «Internet of Things: A Comprehensive Review of Enabling Technologies, Architecture, and Challenges,» *IEEE Internet of Things Journal,* vol. 35, nº 2, pp. 205-220, 2017.

[43] Jonathan de Carvalho Silva, Joel J. P. C. Rodrigues, Antonio M. Alberti, Petar Solic y Andre L. L. Aquino, «LoRaWAN: A low power WAN protocol for Internet of Things: A review and

opportunities,» *In 2017 2nd International multidisciplinary conference on computer and energy science (SpliTech),* pp. 1-6, 2017.

4 ANEXOS

ANEXO I. CÓDIGO FUENTE DEL DISPOSITIVO IOT

1. Script de la placa para identificación de TAGs:

   Code_Registro_TAG.ino

2. Script de la placa de identificación de usuarios:

   Code_Placa_Control_ID.ino

3. Script de la placa de control de flujo:

   Code_Placa_Control_Flujo.ino

ANEXO II. PRUEBA DE MEDICIÓN DE FLUJO Y DATOS PARTE 1

Simulación del paso de líquido

Enlace:

simulacion de paso de liquido.mp4

ANEXO III. PRUEBA DE MEDICIÓN DE FLUJO Y DATOS PARTE 2

Flujo de agua y enlace con el aplicativo web

Enlace:

Integración con el componente de la capa de aplicación.mp4

ANEXO IV. ADAPTACIÓN PARA MEDIR DIFRENTE TIPO DE BEBIDA

Enlace:

Adaptación a otra bebida.mp4