Redes Inalámbricas - Proyecto Corto 1

1st Sthefany Carabajo Avila, 2nd David Castillo, 3rd Franklin Gomez, 4th Micaela Tobar
Facultad de Ingeniería
Universidad de Cuenca
Cuenca, Ecuador
sthefany.carabajo1911@ucuenca.edu.ec,franklin.gomez@ucuenca.edu.ec,
micaela.tobar@ucuenca.edu.ec, nestor.castillo@ucuenca.edu.ec

***Resumen*—En el presente trabajo se realiza un acercamiento al modulo RFID/NFC PNS32 conjuntamente con el microcontrolador ESP32. El controlador ESP32 se encarga de realizar la configuración de pines para la lectura de datos del modulo RFID. A continuación, dentro de la misma red local se establece conexión con un servidor MQTT para la transmisión de datos. Finalmente, se crea un entorno de control utilizando la herramienta Node-RED que nos permite recibir los datos transmitidos por el servidor mosquitto y con un protocolo de comunicación MQTT y los visualiza dentro de una interfaz local; de la misma forma, también se puede crear dentro de Node-RED un entorno de comunicación donde intervienen los usuarios y el microcontrolador.**

I. Introducción

El rápido avance de la tecnología en los últimos años ha impulsado la evolución de las redes inalámbricas, dentro de estas tecnologías se encuentra la de identificación por radiofrecuencia (RFID). Las redes inalámbricas se han convertido en una solución eficaz para el acceso a Internet y la conectividad de dispositivos, mientras que la tecnología RFID se utiliza cada vez más en una amplia gama de aplicaciones, desde el seguimiento de inventario, monitoreo por radiofrecuencia, seguimiento y localización de objetos, etc. Además, el protocolo MQTT (Message Queuing Telemetry Transport) se ha vuelto cada vez más popular como una forma eficiente de transmitir datos a través de redes inalámbricas. En este informe, se explorarán en profundidad estos tres temas y se analizarán sus aplicaciones, ventajas y desventajas, así como su impacto en la industria y la sociedad en general.

El presente documento se divide en 4 secciones, empezando por la actual sección I en donde se da a conocer un poco sobre la evolución de RFID y de MQTT. A continuación, en la sección II se describe todos los pasos implementados para la preparación del hardware, la comunicación con el servidor mosquitto y el desarrollo del entorno gráfico de control para el usuario final. En la sección III se tiene información acerca de los logros obtenidos en cuanto a la implementación descrita en la sección anterior. Previamente antes de finalizar el documento, se agrega la sección IV que contiene información acerca de algunos obstáculos que se encontraron al momento de realizar el proyecto. Para finalizar, la sección V incluye comentarios finales con respecto a las aportaciones realizadas en el documento.

Facultad de Ingeniería.

II. Metodología

II-A. Parte 1: Preparación del hardware

En esta primera parte se procederá a describir los materiales utilizados, conjuntamente con sus respectivas librerías, herramientas de programación y herramientas de control de datos. A continuación, se describe todos los materiales y herramientas utilizados:

- ESP32
- Módulo Lector RFID/NFC PN532
- *Tags* RFID
- 2 LED
- Entorno de control Node-RED
- Mosquitto como *bróker* de MQTT
- Resistencias, conectores, y *protoboard*

En la Figura 1, se observa la el circuito implementado para la conexión entre el ESP32 y el módulo RFID.

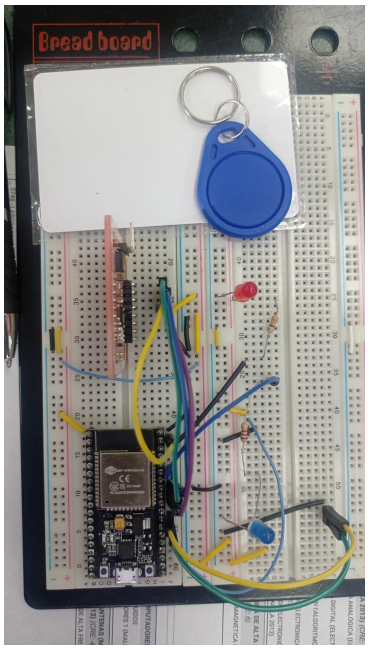

Figura 1: Circuito Implementado

A continuación en la tabla I se detalla la conexión de pines realizada

Cuadro I: *Configuración de Pines para el SP32*

Pin SP32	Conexión
G23	PN532_MOSI
G22	LED rojo
G21	LED azul
G19	PN532_MISO
G18	PN532_SSK
G5	PN532_SS
Alimentación 3.3v	Alimentación general

II-B. Parte 2: Comunicación entre el Módulo RFID y el ESP32

Para esta práctica se empleó el módulo PN532, el cual cuenta con diferentes protocolos para comunicarse con el IDE de Arduino: UART, I2C o SPI. Estos diferentes protocolos utilizan pines y bibliotecas específicos del microcontrolador y para seleccionar el modo de comunicación, el PN532 debe configurarse utilizando los interruptores DIP.
En este caso se empleó la comunicación SPI, por lo que se usó la librería *SPI.h* y se configuró el módulo como se muestra en la figura 2.

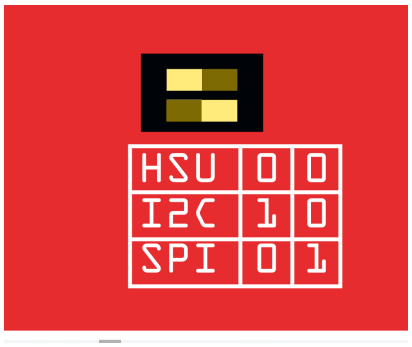

Figura 2: Configuración del modulo para emplear SPI

Para la comunicación con el módulo se empleó la librería *Adafruit_PN532.h* con las conexiones que se mostraron en la tabla I.

II-C. Parte 3: Servidor MQTT

Como se mencionó anteriormente, para esta práctica se emplea el protocolo MQTT para comunicar el ESP32 con Node-RED.
MQTT es un protocolo de mensajería que utiliza un enfoque cliente-servidor para la transmisión de mensajes y se basa en el modelo de publicación/suscripción, donde los dispositivos pueden publicar mensajes en un tema o *topic* específico y los suscriptores pueden recibir los mensajes publicados en ese tema y debido a que es un protocolo de comunicación eficiente y confiable es ideal para su uso en sistemas de IoT con recursos limitados [1].

Como servidor se empleó *Mosquitto*, el cual es un software de servidor de código abierto que implementa el protocolo MQTT para permitir la comunicación entre dispositivos IoT de manera eficiente y confiable [2].

Y para poder interactuar con el microcontrolador mediante MQTT de forma sencilla se usó Node-RED, en la figura 3 se muestra un diagrama que ilustra el proceso de publicación y suscripción entre el ESP32, Mosquitto y Node-RED.

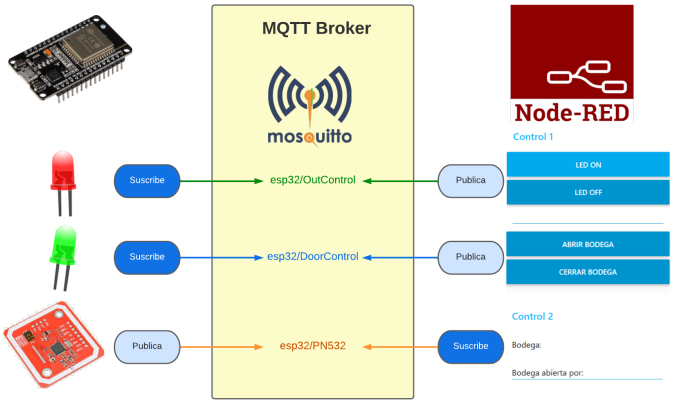

Figura 3: Diagrama de bloques de la conexión entre el ESP32, Mosquitto y Node-RED

II-C1. Mosquitto Broker: Como servidor MQTT se usó el servicio de Mosquitto *broker*, en Windows tanto su instalación como uso es muy sencillo, basta con descargar el instalador desde el sitio oficial https://mosquitto.org/download/ y seguir todos los pasos.

Se puede ejecutar el servidor como si fuera un servicio de Windows, sin embargo de esta forma las peticiones anónimas están desactivadas, lo que impide que el ESP32 se pueda conectar al servidor de Mosquitto, la solución a este problema se detalla más adelante, en la sección de conflictos.

Para comprobar que el servidor se está ejecutando, se puede usar el siguiente comando, el cual desplegará una lista de todos los servicios que se estan ejecutando y el puerto que usan.

```
>netstat -an
```

La figura 4 muestra que la escucha se lleva a cabo a través del puerto TCP 1883.

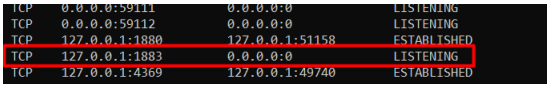

Figura 4: Listado de servicios activos y sus puertos de destino

Mosquitto incorpora dos ejecutables para realizar publicaciones y subscripciones: ***mosquito_sub*** y

mosquito_sub, estos servicios se usaron para comprobar el funcionamiento del servidor, en la sección de resultados se muestran dichas pruebas, en la figura 10.

II-C2. MQTT con ESP32: Para usar el protocolo MQTT con ESP32 se empleó la librería *Async MQTT Client Library* de Marvin Roger, esta librería no se encuentra disponible en el administrador de bibliotecas de Arduino y puede ser descargada desde el repositorio de esta práctica.

Esta librería permite crear un objeto *AsyncMqttClient*, el cual permite manejar todos los parámetros necesarios para establecer conexión con un servidor MQTT, como la configuración del servidor y la suscripción y publicación en los *topics*, tal como se muestra a continuación.

```
AsyncMqttClient mqttClient;

// funcion para establecer conexion MQTT
void connectToMqtt() {
  Serial.println("Connecting to MQTT...");
  mqttClient.connect();
}

mqttClient.onConnect(onMqttConnect);
mqttClient.onDisconnect(onMqttDisconnect);
mqttClient.onPublish(onMqttPublish);
mqttClient.onSubscribe(onMqttSubscribe);
mqttClient.onUnsubscribe(onMqttUnsubscribe);
mqttClient.onMessage(onMqttMessage);
mqttClient.setServer(MQTT_HOST, MQTT_PORT);
```

En la figura 3 se mostró que para esta práctica se usó tres *topics* MQTT, los mismos deben ser declarados dentro del código para poder publicar y recibir mensajes en los mismos, a continuación se muestra la declaración de los *topics* empleados.

```
//__________ MQTT Topics _____
#define MQTT_PUB_UID  "esp32/PN532"
#define MQTT_SUB_OUT_TEMP "esp32/OutputControl"
#define MQTT_SUB_DOOR "esp32/DoorControl"
```

Para publicar un mensaje en un *topic* se emplea la función *.publis()*, en donde se indica el nombre del *topic*, la calidad del servicio y el mensaje que se va a enviar, también es importante establecer un ID para cada paquete que contendrá un mensaje.
A continuación se muestran los mensajes que se publican cuando se acerca una tarjeta RFID al lector. Al reconocer una tarjeta válida, se publica un mensaje en el *topic esp32/DoorControl* con el mensaje *OPEN* y al mismo tiempo, se envía la identidad a la que pertenece el UID de la tarjeta leída por el *topic esp32/PN532*, estos mensajes luego son leídos por los nodos suscriptores de Node-RED y pueden presentar la información en el *Dashboard*.

```
// Publica un mensaje MQTT en el topic  "esp32/
    DoorControl"
uint16_t packetIdPub3 = mqttClient.publish(
    MQTT_SUB_DOOR, 0, true, String("OPEN").c_str()
    );
Serial.printf("Publishing on topic %s at QoS 1,
    packetId: %i", MQTT_SUB_DOOR, packetIdPub3);
Serial.println("");

  // Publica un mensaje MQTT en el topic  "esp32/
     PN532"
uint16_t packetIdPub4 = mqttClient.publish(
    MQTT_PUB_UID, 0, true, String("Franklin").
    c_str());
Serial.printf("Publishing on topic %s at QoS 1,
    packetId: %i", MQTT_PUB_UID, packetIdPub4);
Serial.println("");
```

Para suscribir el ESP32 a un *topic* se emplea la función *suscribe(topic, QoS)*, en la cual se especifica el nombre del tema y la calidad del servicio, con ello el microcontrolador será capaz de recibir todos los mensajes enviados por el nodo publicador de Node-RED y, según el mensaje recibido el ESP32 activará o desactivará el pin 21, tal como se muestra a continuación.

```
  //suscripcion para control de la puerta
  uint16_t packetIdSub2 = mqttClient.subscribe(
      MQTT_SUB_DOOR, 2);
  Serial.print("Subscribing  at QoS 2, packetId: "
      );
  Serial.println(packetIdSub2);

   if (messageTemp == "OPEN"){
  digitalWrite(DoorPin, HIGH);
  Serial.println("Puerta abierta");
  }
  if (messageTemp == "CLOSE"){
  digitalWrite(DoorPin, LOW);
  Serial.println("Puerta cerrada");
  }
```

El código completo puede encontrarse en el repositorio de esta práctica.

II-D. Entorno de control Node-RED

La herramienta Node-RED funciona como una plataforma de programación visual para conectar con servidores, microcontroladores, bases de datos, servicios en la nube, etc; también permite crear nodos personalizados y entornos de trabajo para el control y automatización de tareas. En la práctica se ha previsto dar uso a Node-RED para generar una interfaz de usuario amigable que permita visualizar indicadores como el nombre del usuario que accede con su *tag* o la manipulación del LED que se configuro en las secciones anteriores. Para lograr lo propuesto se necesita realizar una comunicación con el servidor Mosquitto, para ello en la paleta de red existen nodos que permite enviar y recibir datos. Los nodos mencionados se observan en la figura 5

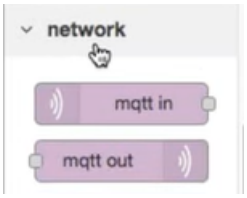

Figura 5: Nodos MQTT para recepción y transmisón de datos.

Dentro de los nodos MQTT se configura la información del servidor como se observa en la Figura 6, donde se incluye la dirección IP y el puerto al que se conectará el servidor.

Después de configurar el servidor MQTT, se debe especificar el *topic* al que se suscribirá o al que publicará el nodo, en la figura 7 se muestra un nodo de salida (publicador), que publicará los mensajes en el topic *esp32/DoorControl*.

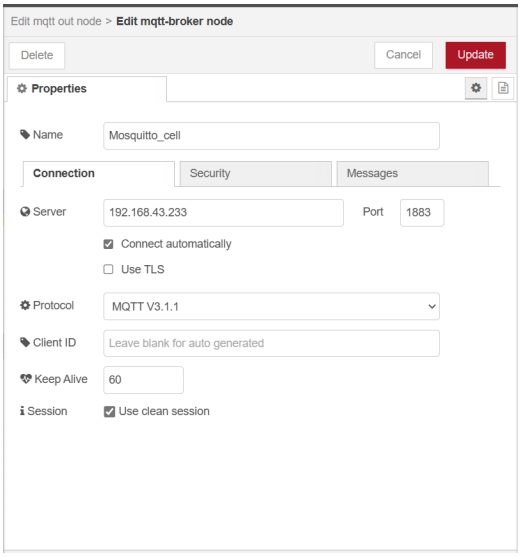

Figura 6: Configuración del nodo: servidor MQTT

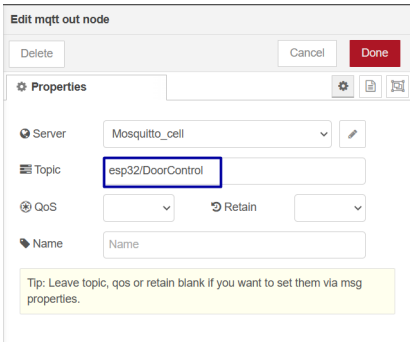

Figura 7: Configuración del nodo: *topic*

A continuación se procede a colocar nodos que pertenecen a la paleta de *dashboard* como botones y visualizadores de texto. En la Figura 8 se tiene todos los flujos implementados.

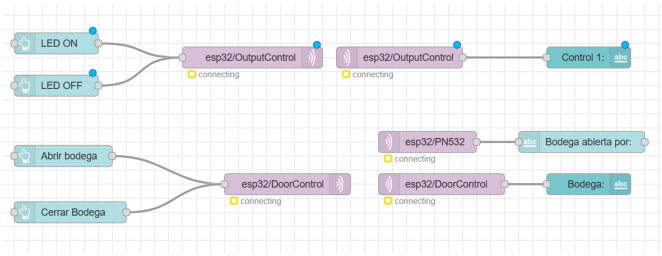

Figura 8: Diagrama de Flujo en Node-RED

Finalmente, la interfaz de trabajo se observa en la Figura 9.

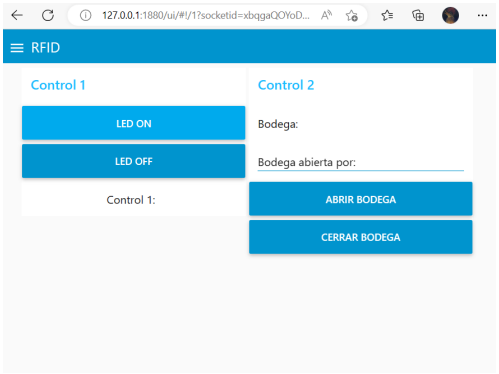

Figura 9: Interfaz de usuario en Node-RED

III. Análisis de Resultados

Para probar los resultados, en primer lugar se debe levantar el servidor Mosquitto; para ello, en una consola de comandos en windows se cambiara el directorio a la carpeta en donde se encuentre instalado Mosquitto. Como método de prueba, se utiliza el comando *mosquitto_sub* para poder suscribirse a al *topic* "prueba" e indicamos mediante el comando *-h localhost* que el servidor se esta ejecutando en la misma máquina. Ahora el cliente está escuchando los mensajes que se publican en el mismo tema pero en otro cliente Mosquitto. En la Figura 10 se visualiza la validez del cliente.

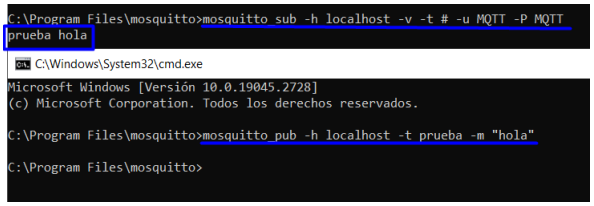

Figura 10: Prueba del cliente creado en Mosquitto

Como se menciono anteriormente, hay tres *topic* con los que se trabaja. Dentro del código del microcontrolador se elije la información que se desea transmitir por cada tópico para finalmente en Node-RED tener un conjunto de información descentralizada que pueda ser publicada y utilizada en sus respectivos flujos de programación. Un ejemplo de esto se observa en la Figura 11, en donde se tiene que el microcontrolador publica la información de una tarjeta RFID que se ha

acercado al lector. En el *topic esp32/DoorControl* se transmite el mensaje "OPEN" y a su vez en el *topic esp32/PN532* se transmite el nombre del usuario portador de la tarjeta RFID.

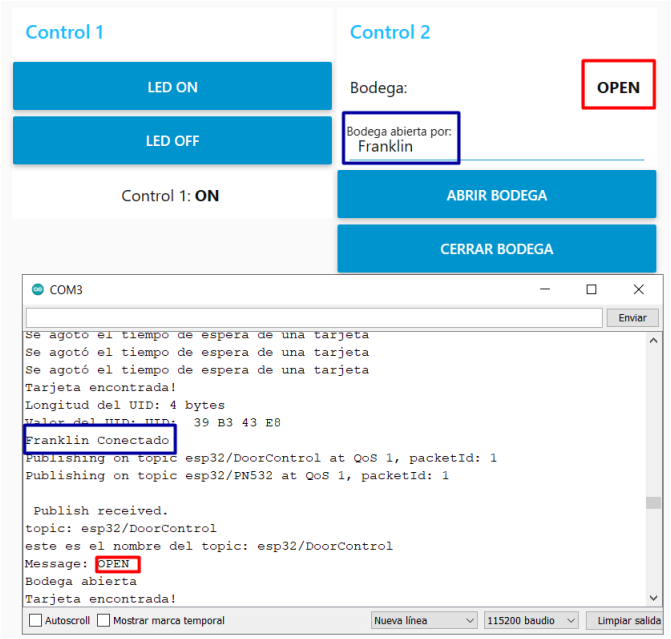

Figura 11: Envió de información al servidor Mosquitto. Recepción de información en Node-RED

Por otro lado, en la Figura 12 se puede visualizar la interacción y el flujo de mensajes que arriban al cliente con el comando:

```
>>mosquitto_sub -d -h 192.168.0.53 -p 1883
-t "esp32/DoorControl"
```

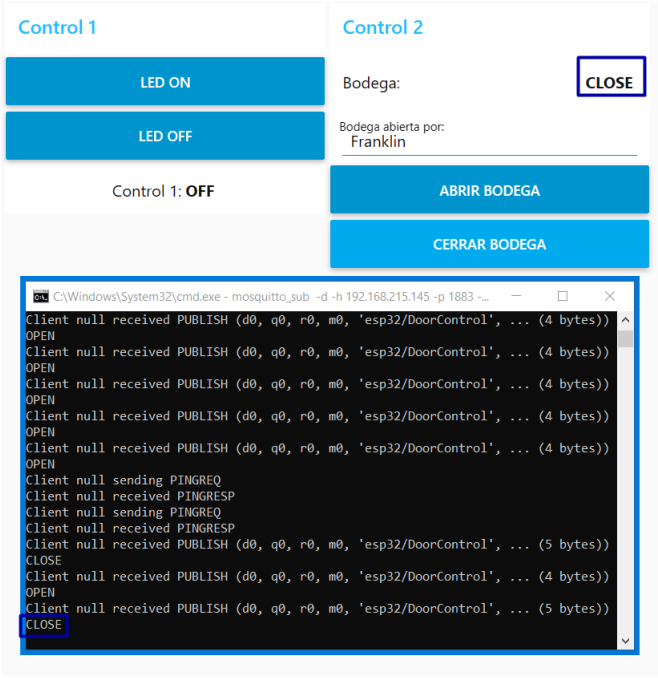

Figura 12: Mensajes que circulan por el *topic esp32/DoorControl*

IV. Conflictos

- Con la librería empleada, el comportamiento predeterminado del PN532 es el de aguardar permanentemente por un *tag* RFID, este comportamiento impide que se envíen los mensajes al servidor MQTT hasta que una tarjeta sea leída por el módulo. Para solucionar este problema se colocó un tiempo de desconexión de cinco segundos al final de la función *nfc.readPassiveTargetID*, tal como se muestra a continuación:

```
nfc.readPassiveTargetID(, , , 5000);
```

Con este cambio, el módulo solo esperará cinco segundos por un *tag* y después se ejecutan las demás instrucciones del código, lo cual produce que constantemente se este realizando una retroalimentación de información a los servidores de transmisión y control. En la figura 13 se observa un mensaje que se envía al serial en donde se indica que el tiempo de espera a culminado.

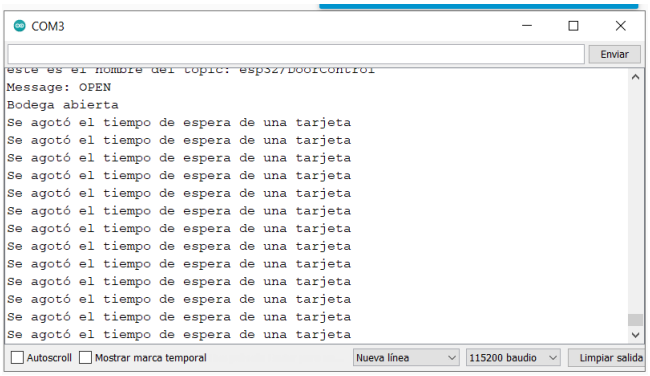

Figura 13: Mensaje del puerto serial que indica el fin del tiempo de espera

- Anteriormente se mencionó que el funcionamiento predeterminado de Mosquitto es el de rechazar las peticiones de direcciones IP externas, lo cual generó muchos inconvenientes en la realización de esta práctica, ya que no era posible la conexión del ESP32 con el *broker*, para solucionar este inconveniente, se modifico el archivo de configuración de mosquitto, agregando las líneas que se muestran en la figura 14.
  Para poder modificar el archivo de configuración, se debe ejecutar como administrador el editor de texto empleado, para poder guardar las modificaciones.

Figura 14: Edición del archivo de configuración del *broker*

Con las modificaciones realizadas ya es posible conectar cualquier dispositivo externo con el servidor MQTT.

V. Conclusiones

Es importante destacar el uso de tecnologías RFID ya que ofrece numerosas ventajas de aplicación como el registro de datos, monitoreo de inventario, etc y así lograr un aumento en la efectividad y productividad. A su vez, para poder utilizar un modulo RFID se tiene una gran variedad de librerías que se encuentran a libre alcance para diferentes plataformas como Python, Arduino, C++, etc. El uso de las librerías proporcionan funciones y métodos preescritos que facilitan la comunicación con el módulo RFID y la lectura de datos.

Por otro lado, el uso del controlador ESP32 ofrece una fácil integración con el modulo lector RFID y a su vez ofrece la posibilidad de conectar con el servidor mosquitto MQTT. Además, la conexión con Mosquitto MQTT es una excelente opción para la comunicación entre dispositivos, ya que MQTT es un protocolo ligero y eficiente diseñado específicamente para la comunicación máquina a máquina

Por último, Node-RED es una herramienta visual muy útil para la creación de flujos de trabajo y la integración de diferentes dispositivos y servicios en una aplicación. Al utilizar Node-RED para visualizar los datos de RFID, se pueden crear interfaces intuitivas y personalizadas para la gestión y el control de los procesos de la tecnología RFID.

En conclusión, en la practica realizada se ha generado una incorporación de diferentes tecnologías y se ha logrado que el desarrollo de aplicaciones basadas en RFID sea simple y efectivo. Y además, al ser de libre alcance, estas herramientas son accesibles y fáciles de usar para cualquier desarrollador.

Referencias

[1] Muhammad Saqib Nawaz, Muhammad Adnan, Syed Ahmed Safi, and Nida Shamim. Mqtt protocol: A technical overview. *International Journal of Advanced Computer Science and Applications*, 9(12):127–132, 2018.

[2] Roger Light Hill. Mosquitto: server and client implementation of the mqtt protocol. *Journal of Open Source Software*, 3(25):655, 2018.

[3] Microcontrollers Lab. Esp32 mqtt publish multiple sensor readings to node-red, Aug 2022.

[4] Atefeh Zare and M Tariq Iqbal. Low-cost esp32, raspberry pi, node-red, and mqtt protocol based scada system. In *2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, pages 1–5. IEEE, 2020.