Implementación De Una Solución IoT

Alexander Aponte Largacha
Ingenieria de Sistemas
Pontificia Universidad Javeriana
Bogotá, Colombia
apontea@javeriana.edu.co

Sofia Guerra Jimenez
Ingenieria de Sistemas
Pontificia Universidad Javeriana
Bogotá, Colombia
sofiaguerraj@javeriana.edu.co

Viviana Gomez Leon
Ingenieria de Sistemas
Pontificia Universidad Javeriana
Bogotá, Colombia
gomezlv@javeriana.edu.co

***Abstract*—Abstract— In this paper, a breathalyzer prototype is evaluated in order to implement an Internet of Things (IoT) solution. This is given the worrying figures of alcohol-related deaths and accidents in Colombia. This device allows to monitor the alcohol level in real time. Through the MQTT protocol, the device can send alerts in case the values are above a recommended threshold. The integration of sensors and notifications aims to prevent road accidents and deaths caused by this substance.**

Keywords: *IoT, MQTT Protocol, Breathalyzer, Network.*

***Resumen*—** *En el presente documento, se evalúa un prototipo de alcoholimetro con el fin de implementar una solución del Internet de las Cosas (IoT). Esto gracias a las preocupantes cifras de muertes y accidentes relacionadas con el alcohol en Colombia. Este artefacto permite monitorear el nivel de alcoholemia en tiempo real. A través del protocolo MQTT, el dispositivo puede enviar alertas en caso de que los valores se encuentren por arriba de un umbral recomendado. La integración de sensores y notificaciones tiene como objetivo prevenir accidentes viales y muertes por esta sustancia.*

Palabras Clave: *IoT, Protocolo MQTT, Alcoholimetro, Red.*

I. Introducción

El Internet de las Cosas (IoT) ha brindado múltiples avances a lo largo de los últimos años, con usos enriquecedores. Mientras que estos, se pueden apreciar principalmente desde un punto de vista académico o científico, muy pocas veces suelen tener un impacto real en la sociedad. Es gracias a esto, que se tomo la decisión de relacionar esta innovadora tecnología con una problemática del mundo actual, y más específicamente colombiana. Con este enfoque, esperamos aportar a la sociedad un nuevo uso de IoT para avanzar a un mejor futuro.

II. Definición de la problemática

Con el objetivo principal de identificar una problemática que pueda ser solucionada a través de IoT, se tomo la decisión de abordar la problemática de muertes relacionadas con alcohol. Las muertes relacionadas a esta sustancia representaron 1.900 casos en el 2020, de las cuales aproximadamente 1000 se encontraban en un grado 3 de alcoholemia [3]. Apoyando este dato, en el año 2021, la Agencia Nacional de Seguridad Vial advirtió que conducir en estado de embriaguez es la tercera causa de muerte en las vías del país. Más específicamente, de los 375 siniestros registrados ese año, 121 personas fallecieron y otras 525 resultaron lesionadas [2]. Sin embargo este no es un problema reciente en la sociedad colombiana, de hecho es un problema que ha persistido a lo largo de la ultima década, tal como demuestra el hecho de que hay 47 mil víctimas anuales en promedio, desde el año 2001 [1].

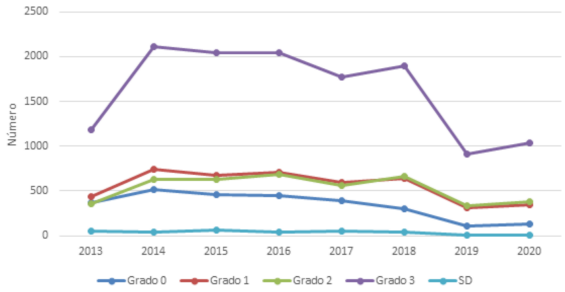

Fig. 1. Muertes asociadas al alcohol según grado de alcoholemia. [3]

Teniendo en cuenta el impacto que representa las muertes por alcohol en el país y la preocupante cifra de muertes por accidentes automovilísticos en estado de alcoholemia, se tomo la decisión de crear un prototipo de Alcoholimetro que permita tanto a consumidores habituales, policías, entidades de transito y particularmente personas con adicción a esta sustancia determinar su estado de embriaguez. Esto se realizara determinando el umbral donde el estado de alcoholemia sea alto, y si se sobrepasa este, el dispositivo emitirá un pitido.

Se espera que este dispositivo pueda brindar datos inmediatos del estado en el que se encuentra una persona bajo los efectos del alcohol y aumentar la prevención frente a las consecuencias que puede llegar a tener. Si en caso tal, se supera el umbral indicado, se pueden tomar acciones rápidas frente a su estado. De igual manera, para asegurar una implementación

más efectiva y con un mayor impacto, si un usuario llega a sobrepasar el umbral, se enviaran unas notificaciones por *telegram* notificando a sus contactos de emergencia, y en caso tal de que su grado de alcoholemia sea grave, se notifiquen a sus contactos de emergencia que es recomendable llamar a las autoridades.

III. Diagrama esquemático

A. Primer ESP8266

Conexiones del sensor al ESP8266:

- VCC del sensor a 3V3 del ESP.
- GND del sensor a GND del ESP.
- Salida analógica (A0) del sensor a A0 en el ESP.

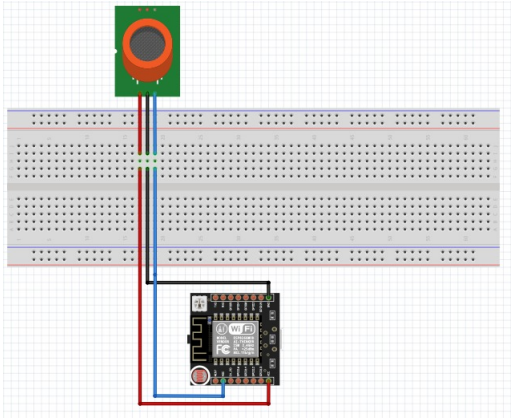

Fig. 2. Diagrama esquemático sensor.

B. Segundo ESP8266

Conexiones del buzzer al ESP8266:

- VCC del buzzer a un pin digital (D1).
- GND del buzzer a GND del ESP.

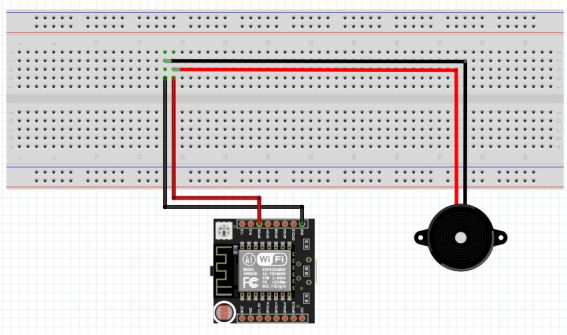

Fig. 3. Diagrama esquemático actuador.

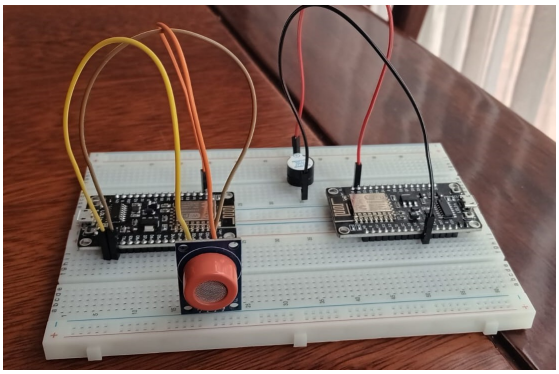

Fig. 4.1. Prototipo del proyecto implementado.

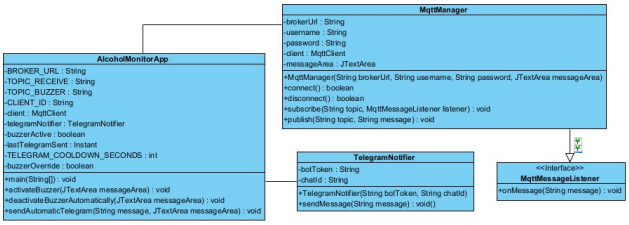

Fig. 4.2. Diagrama de clases del proyecto implementado.

IV. Topología de red

Los protocolos utilizados para la comunicación de cada uno de los componentes son:

A. Comunicación del primer ESP8266 con el broker MQTT

A través del protocolo MQTT, el primer ESP8266 se conecta al broker y envía los datos del sensor MQ3. MQTT es un protocolo de comunicación ligero ideal para dispositivos IoT con baja potencia de procesamiento y ancho de banda limitado. El ESP8266 publica los niveles de alcohol en un tópico específico al cual la aplicación Java y el segundo ESP8266 se suscribirán.

B. Comunicación de la aplicación en Java con el broker MQTT

A través del protocolo MQTT, la aplicación en Java se suscribe al mismo tópico donde el primer ESP8266 publica los datos del sensor. Esto permite que la aplicación reciba notificaciones en tiempo real cada vez que se detecta un nivel de alcohol. Luego, la aplicación Java decide si el segundo ESP8266 debe activar el buzzer.

C. Publicación de acciones desde la aplicación en Java

A través del protocolo MQTT y basándose en los datos recibidos, la aplicación Java publica instrucciones en un tópico de control al cual el segundo ESP8266 está suscrito. Esto garantiza que el segundo ESP8266 ejecute la acción (activación del buzzer) en función de los valores recibidos.

D. Comunicación del segundo ESP8266 con el broker MQTT

A través del protocolo MQTT, el segundo ESP8266 se suscribe al tópico de control que gestiona la aplicación Java. Una vez que recibe el mensaje correspondiente, activa o desactiva el buzzer dependiendo de la orden recibida. Esta comunicación es esencial para que el segundo ESP8266 actúe sin necesidad de conexión física directa con el sensor.

E. Notificaciones de estado al usuario a través de Telegram

A través del protocolo HTTPS, la aplicación en Java se conecta a la API de Telegram para enviar notificaciones al usuario. HTTPS se utiliza para asegurar la conexión y la transmisión de datos entre la aplicación y los servidores de Telegram, garantizando la privacidad de la información sobre el estado de alcohol detectado.

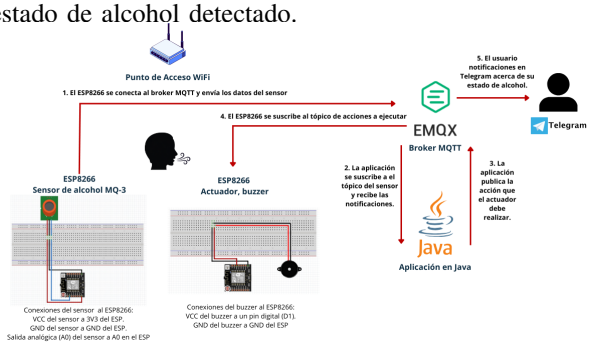

Fig. 5. Topología de red.

V. Protocolo MQTT

MQTT o Message Queuing Telemetry Transport es un protocolo de mensajería para microcontroladores pequeños que están conectados a redes no fiables, poseen un ancho de banda bajo o tienen recursos hardware limitados. Este se utiliza principalmente para comunicaciones de máquina a máquina o conexiones IoT.

A. Historia

Este protocolo fue inventado por Andy Stanford-Clark y Arlen Nipper en 1999. Fue creado con el propósito de comunicar los dispositivos de monitorio en la industria petrolera y de gas para que sus datos fueran enviados a servidores remotos. Usualmente para conectar dispositivos de medición y recopilar sus datos, se usaba cableado o comunicaciones por satélite, sin embargo dentro de este contexto resultaba imposible o altamente costoso. Así, se vio la necesidad de crear una forma de comunicación para sensores que pudieran proveer datos de manera barata, eficiente y que empleara un ancho de banda mínimo [4].

En 2012, OASIS (Organización para el Avance de Estándares de Información Estructurada) estandarizo MQTT como código abierto. A día de hoy, OASIS aun gestiona MQTT [6].

B. Como funciona

Lo más importante para comprender este protocolo, es entender su modelo de publicación-suscripción. En este existen dos tipos de sistemas: clientes y brókeres. Un bróker es el servidor donde los clientes interactúan: recibe las comunicaciones de algunos y las transmite a otros. Los clientes no establecen una comunicación directa, sino que establecen una conexión con el bróker. Cabe destacar que cada cliente puede tomar el rol de publicador o suscriptor. Los sensores suelen ser los publicadores, y los host que reciben los datos emitidos por los sensores suele ser los suscriptores. [4]

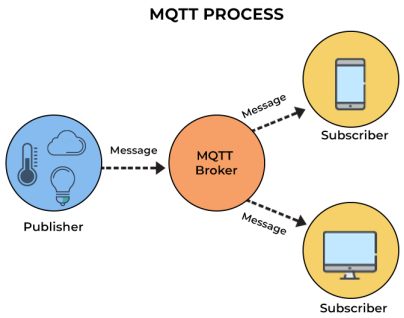

Fig. 6. Proceso MQTT.

Este protocolo es controlado por eventos, es decir, un cliente sólo publica cuando hay información para enviar, y un bróker sólo envía información a los suscriptores cuando llegan nuevos datos.

En esta topología se pueden realizar 6 acciones principalmente:

1) Conectar: La primera acción que realiza un cliente para establecer una conexión con el broker.
2) Publicar: Acción mediante la cual el cliente envía un mensaje a un tema específico.
3) Suscribirse: Acción que permite a un cliente recibir mensajes sobre un tema específico.
4) De suscribirse: Acción mediante la cual un cliente elimina su suscripción a uno o varios temas.
5) Ping: Acción para verificar que la conexión sigue activa y en buen estado.
6) Desconectar: Acción mediante la cual un cliente cierra su conexión con el broker de manera deliberada.

C. Temas

Los mensajes enviados por los editores al broker son publicados como ”temas”. Estos son palabras claves que utiliza el broker con el fin de filtrar los mensajes para los clientes de MQTT. Así, el broker gestiona todo los mensaje que llegan, y reenvía a los suscriptores según el tema como se ve en la **Figura—**. De esta manera, los suscriptores solo recibirán mensajes relacionados al tema al que estén suscritos. Los tema están organizados jerárquicamente, con una estructura similar a un directorio de carpetas, usando el carácter barra (/) como delimitador. [7]

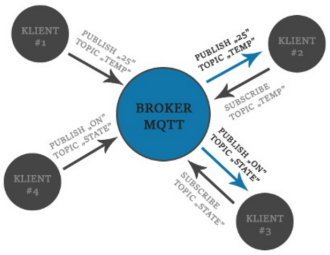

Fig. 7. Sistema de envio de mensajes según temas.

Si el broker recibe datos publicados sobre un tema que actualmente no existe, este se encarga de crear el tema para que los suscriptores pueda suscribirse a este. Teniendo en cuenta que los suscriptores pueden suscribirse a temas especifios o a una estructura de temas, se pueden llegar a usar los siguientes comodines:

- **+:** Representa un nivel de jerarquía.
  Por ejemplo, para la estructura de temas *"casa/sala/temperatura"*, para suscribir a todos los sensores de temperatura en diferentes habitaciones se usaria *"casa/+/temperatura"*.

- **#**: Representa múltiples niveles de jerarquía.
  Por ejemplo, para la estructura de temas *"vehiculo/motor/rpm"*, para suscribir a todos los datos del vehículo se usaría *"vehiculo/"*.

D. Calidad de Servicio (QoS)

Existen tres niveles diferentes de calidad de servicios (QoS) con respecto a garantizar la entrega de mensajes entre el servidor y los clientes. Dependiendo de cual se use, se puede minimizar la transmisión de datos o maximizar la fiabilidad. Estos son los tres niveles:

- **QoS 0**: En este nivel, cada mensaje (PUBLISH) se envía una sola vez, y al no haber confirmación, no se garantiza que llegue. Este es el más rápido, pero menos confiable.

Fig. 8. Metodo de envio QoS 0.

- **QoS 1**: En este nivel, el publicador intenta entregar el mensaje, posteriormente espera una respuesta de confirmación (PUBACK). Si no se recibe una confirmación dentro de un rango de tiempo especifico, el mensaje se envía de nuevo. Así, el broker puede recibir el mensaje más de una vez si el publicador no recibe el PUBACK a tiempo.

Fig. 9. Metodo de envio QoS 1.

- **QoS 2**: Este nivel, utiliza un proceso de 4 pasos en el cual asegura que el mensaje se entregue exactamente una vez. Inicialmente, se envia el mensaje (PUBLISH) y se recibe la confirmación de que fue entregado con un PUBREC (Publish Recieved), luego el publicador confirma que puede liberar el mensaje con un PUBREL (Publish Release) y el borker responde finalmente con un PUBCOMP (Publish Complete) para confirmación final y que el broker lo ha procesado. Este nivel da la máxima fiabilidad, pero más latencia.

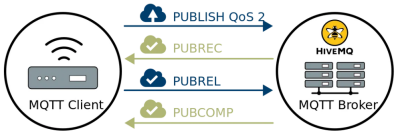

Fig. 10. Metodo de envio QoS 2.

Cabe destacar que estos niveles se pueden dar entre publicador-broker y broker-suscriptor. Así, cada cliente (suscriptor o publicador) pueden configurar su nivel QoS de manera independiente [7].

E. Mensajes Retenidos / Última voluntad y testamento

Cuando un publicador envía un mensaje al broker y este lo reenviar a sus suscriptores, el broker no suele guardar este a menos que tenga una bandera de retención. Esto es un mensaje retenido. Sin embargo, para asegurarse que un nuevo suscriptor reciba los mensajes de su tema asociado, los brokers pueden conservar el ultimo mensaje enviado de cada tema. Así, cada vez que un nuevo cliente se suscriba a un tema o cuando un cliente existente vuelva a conectarse después de un tiempo, el mensaje retenido se envía a los suscriptores para garantizar que la suscripción se encuentre activa y tenga la información más reciente. [5]

Por otro lado, en caso de que un publicador se desconecte de la red de manera imprevista, el puede registrar un mensaje para enviarlo a sus suscriptores. Este se denomina Última voluntad y testamento o Will Message. Este mensaje se almacena en el caché del broker e incluye información que permite identificar al publicador desconectado para que se puedan tomar acciones.

F. Seguridad

Teniendo en cuenta el objetivo principal del protocolo, y la razón por la cual fue inventado, la seguridad no fue un factor fundamental a la hora de su creación. Sin embargo, para mantener la seguridad en su entorno se consideran 3 factores principales:

1) SSL/TLS: Teniendo en cuenta que se monta sobre TCP/IP, se puede implementar SSL/TLS. Asín cifrando la conexión entre los clientes y el broker, manteniendo la privacidad. Sin embargo, esto añade una sobrecarga a las comunicaciones.
2) Autenticación: Se puede establecer usuarios y contraseña para establecer una conexión con el broker. Sin embargo, al no estar cifrado, esto se suele usar para evitar conexiones involuntarias.
3) Autorización: Los brokers pueden configurar permisos específicos de acceso a temas para cada usuario.

G. Brokers actuales

Hoy en día se tienen múltiples brokers tales como RabbitMQ, HBMQTT, Aedes y ActiveMQ. Pero ninguna tan conocida como Mosquitto, la cual es open Source y fue desarrollado por la fundación Eclipse y distribuido bajo licencia EPL/EDL. Este es un broker liviano y adecuado para uso en servidores de baja potencia.**b8**

VI. Protocolos de prueba

A. Prueba de Conexión WiFi

Propósito: Asegurarse de que ambos ESP8266 se conectan correctamente a la red WiFi.
Procedimiento:

- Conectar ambos ESP8266 a la red WiFi y verificar en el monitor serial si se muestra el mensaje de conexión exitosa.

Criterio de Éxito: Ambos dispositivos deben conectarse a la red sin errores.

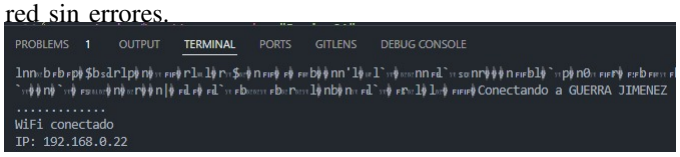

Fig. 11. Conexión a WiFi del primer ESP8266.

Fig. 12. Conexión a WiFi del segundo ESP8266.

B. Prueba de Publicación y Suscripción MQTT del lado del sensor

Propósito: Verificar que el primer ESP8266 publique los datos correctamente y que la aplicación en Java los reciba.
Procedimiento:

- Configurar el primer ESP8266 para publicar los valores de alcohol recibidos en el sensor MQ-3.
- Observar la interfaz gráfica creada en la aplicación de Java para verificar que el mensaje publicado se reciba correctamente.

Criterio de Éxito: La interfaz de la aplicación muestra el valor publicado en el monitor serial sin pérdidas de mensajes.

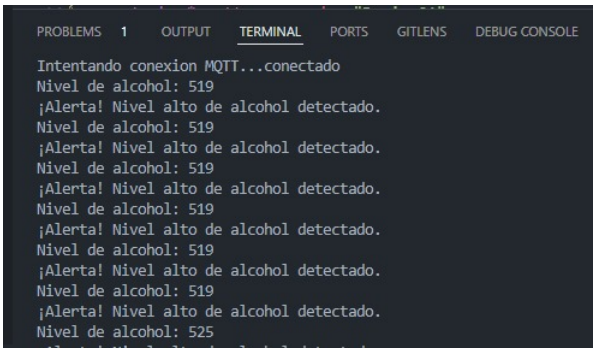

Fig. 13. Conexión y publicación al topico "Sensor" del primer ESP8266.

Fig. 14. Suscripción al topico "Sensor" de la aplicación.

C. Prueba de Publicación y Suscripción MQTT del lado del actuador

Propósito: Verificar que la aplicación en Java publique la instrucción correctamente y que el segundo ESP8266 la reciba.
Procedimiento:

- Configurar la aplicación para que, pasado el umbral, envíe una señal de activación (del buzzer) al tópico MQTT.
- Observar el monitor serial del segundo ESP8266 para verificar que el mensaje publicado se reciba correctamente.
- El buzzer conectado al segundo ESP8266 se enciende hasta que baje nuevamente el nivel de alcohol por debajo del umbral, o hasta que se desactive manualmente (a través de la aplicación).

Criterio de Éxito: El segundo ESP8266 muestra el valor publicado en el monitor serial sin pérdidas de mensajes. Adicionalmente, se enciende el buzzer cuando el valor de alcohol supera el umbral.

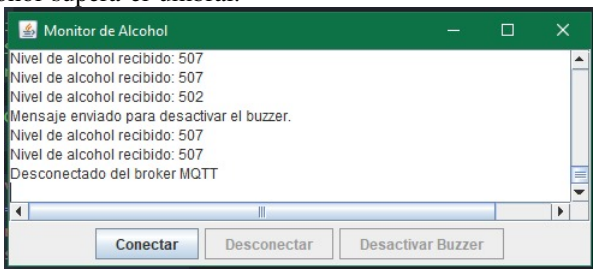

Fig. 15. Instrucción de desactivar el buzzer.

D. Prueba de Sensor MQ3

Propósito: Confirmar que el sensor MQ3 detecta cambios en el nivel de alcohol y envía el valor adecuado al primer

ESP8266.

Procedimiento:

- Conectar el sensor MQ3 y medir su salida en condiciones normales y luego al exponerlo a alcohol.
- Observar los valores leídos en el monitor serial del primer ESP8266.

Criterio de Éxito: El sensor detecta un cambio en la salida (en el valor analógico de A0) al estar expuesto a alcohol.

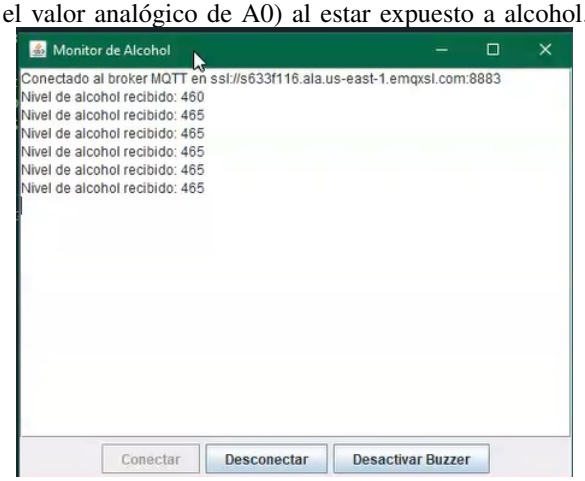

Fig. 16. Valores antes de acercar el alcohol.

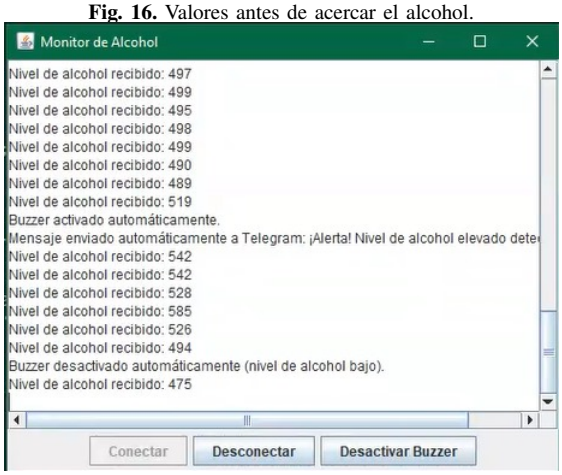

Fig. 17. Valores despues de acercar el alcohol.

E. Prueba de Umbrales de Alerta en el Segundo ESP8266

Propósito: Verificar que el segundo ESP8266 activa el buzzer cuando el nivel de alcohol supera un umbral predefinido.

Procedimiento:

- Configurar un valor de umbral en el código del segundo ESP8266.
- Exponer el sensor a un nivel de alcohol que sobrepase el umbral establecido.

Criterio de Éxito: El buzzer se activa cuando el nivel de alcohol supera el umbral predefinido.

Prueba en el link: https://youtu.be/1chsSwvc5q8.

F. Prueba de Notificación en Telegram

Propósito: Confirmar que el segundo ESP8266 envíe notificaciones en Telegram cuando el nivel de alcohol es alto.

Procedimiento:

- Configurar el segundo ESP8266 para enviar una notificación en Telegram cuando el nivel de alcohol supere el umbral crítico.
- Exponer el sensor a un nivel alto de alcohol para activar la notificación.

Criterio de Éxito: La notificación es enviada en Telegram a los contactos de emergencia cuando se supera el umbral crítico.

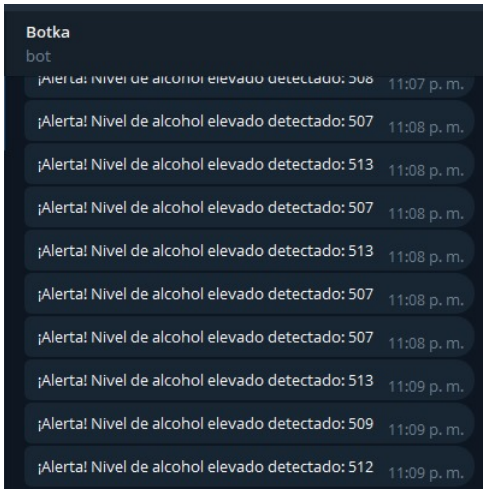

Fig. 18. Mensajes de alerta del bot de Telegram.

VII. PRESUPUESTO

Teniendo en cuenta la problemática planteada y los requisitos que deben cumplir los dispositivos para realizar la implementación, se usaron los siguientes dispositivos. Cabe tener en cuenta que los precios fueron obtenidos en tiendas dedicadas a dispositivos electrónicos. A partir de estos, se obtuvo un presupuesto final de $97,500. A continuación se hace un listado de los dispositivos:

1. Protoboard

Se requieren Protoboards, la cual tiene un costo individual de $11,000. Se usaran 2, por lo tanto su costo total es de $22,000.

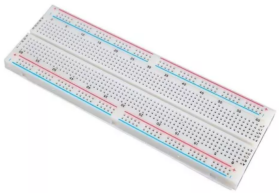

Fig. 19. Protoboard

2. Sensor

Se requiere un sensor de alcohol y etanol MQ-3, el cual tiene un costo de $10,000.

Fig. 20. Sensor de alcohol y etanol MQ-3.

3. Buzzer

Se requiere un buzzer, el cual tiene un costo de $2,000.

Fig. 21. Buzzer.

5. Cables

Se requieren cables para realizar las conexiones. Se usaran cables jumper que tienen un costo de $2,500 y un cable v8 que tiene un costo de $10,000.

Fig. 22. Cables jumper.

Fig. 23. Cable v8.

6. Placa de desarrollo para IoT

Se requieren 2 tarjetas de desarrollo para IoT NodeMCU ESP8266, la cual tiene un costo individual de $25,000. Se usaran 2, por lo tanto su costo total es de $50,000.

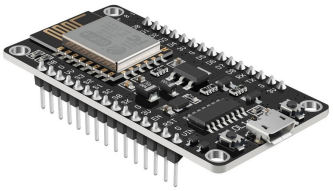

Fig. 24. NodeMCU ESP8266.

REFERENCES

[1] "COLOMBIA: Conducción y alcohol... atracción fatal". Todos somos parte de la solución. Accedido el 5 de noviembre de 2024. [En línea]. Disponible: https://contralaviolenciavial.org/actualidad/43-conduccion-y-alcohol-atraccion-fatal/gmx-niv44-con114.htm

[2] "Conducir en estado de embriaguez, la tercera causa de muerte en accidentes viales en Colombia". infobae. Accedido el 5 de noviembre de 2024. [En línea]. Disponible: https://www.infobae.com/america/colombia/2021/09/21/conducir-en-estado-de-embriaguez-la-tercera-causa-de-muerte-en-accidentes-viales-en-colombia/

[3] "Ministerio de Justicia y del Derecho". Ministerio de Justicia y del Derecho. Accedido el 5 de noviembre de 2024. [En línea]. Disponible: https://www.minjusticia.gov.co

[4] C. BasuMallick. "MQTT Working, Types, Applications". Spiceworks Inc. Accedido el 5 de noviembre de 2024. [En línea]. Disponible: https://www.spiceworks.com/tech/iot/articles/what-is-mqtt/

[5] "¿Qué es MQTT? Definición y detalles". Paessler - The Monitoring Experts. Accedido el 5 de noviembre de 2024. [En línea]. Disponible: https://www.paessler.com/es/it-explained/mqtt

[6] "¿Qué es el MQTT? - Explicación del protocolo MQTT - AWS". Amazon Web Services, Inc. Accedido el 5 de noviembre de 2024. [En línea]. Disponible: https://aws.amazon.com/es/what-is/mqtt/

[7] "What is MQTT Quality of Service (QoS) 0,1, 2? – MQTT Essentials: Part 6". HiveMQ – The Most Trusted MQTT platform to Transform Your Business. Accedido el 5 de noviembre de 2024. [En línea]. Disponible: https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/

[8] "Principales broker MQTT Open Source para proyectos IoT". Luis Llamas. Accedido el 5 de noviembre de 2024. [En línea]. Disponible: https://www.luisllamas.es/principales-broker-mqtt-open-source-para-proyectos-iot/