Escuela Superior Politécnica del Litoral

Facultad de Ingeniería en Electricidad y Computación

Detección de asistencia a eventos con tecnología de Identificación por Radio Frecuencia.

Proyecto Integrador

Previo la obtención del Título de:

Ingeniero en Telecomunicaciones

Presentado por:
Bryan Andrés Yagual Abata
Aaron Andrés Soria Triviño

Guayaquil - Ecuador
Año: 2023

Dedicatoria

Este proyecto es dedicado en primer lugar a Dios, guía constante en mi camino académico. A mis queridos padres Roxana Abata y Felipe Yagual cuyos sacrificios y amor incondicional han sido mi inspiración. A mi querida compañera de vida Yuditt Ayala por su amor y apoyo inquebrantable. A mis abuelos que han sido fundamental en mi vida. A mis hermanos, cuyas risas y complicidad han enriquecido mi vida. A mis respetados profesores gracias por su orientación y enseñanzas transformadoras. Este logro es el resultado del apoyo y amor de cada uno de ustedes.

Bryan Yagual

Dedicatoria

Esta tesis la dedico a Dios por haberme brindado de salud y sabiduría para poder llegar a este punto de mi vida. A mi madre Alexandra Triviño que siempre me brindó su apoyo, amor y puso toda su confianza en mí para ver este sueño hecho realidad. A mi padre William Soria por haber formado las bases de mi educación, además de haber forjado mi carácter para saber actuar ante cualquier circunstancia que se me presentó. A mis hermanos, tíos y abuelos que siempre estuvieron al pendiente de mí, me apoyaron dándome motivación en los momentos más difíciles de mi carrera universitaria.

A la Escuela Superior Politécnica del Litoral por permitirme educarme en sus aulas y poder cumplir una meta más en mi vida, por formar en estos años un profesional competitivo que aspira aportar a la sociedad. También agradezco a los maestros docentes que brindaron de su conocimiento y experiencia para darme una formación profesional, muchas gracias a todos.

Aaron Soria

Agradecimientos

Deseo expresar mi profundo agradecimiento a nuestros mentores, Juan Carlos Avilés y María Antonieta Álvarez, cuya excepcional guía ha sido fundamental para la finalización exitosa de este proyecto. Igualmente, extiendo mi gratitud a mi futuro colega, Aaron Soria, quien desde el comienzo ha sido un pilar de apoyo y un amigo inspirador en este camino. Finalmente, y no por ello menos esencial, deseo expresar mi agradecimiento. sinceramente a mis amigos y familiares por sus constantes buenos deseos y apoyo incondicional desde el principio; sin su aliento, este logro no habría sido posible.

Bryan Yagual

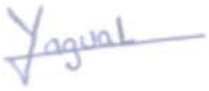
Agradecimientos

Quisiera extender mi más profunda gratitud a nuestros estimados profesores, Juan Carlos Avilés y María Antonieta Álvarez, por su inagotable paciencia y constante contribución fundamental en la realización de este proyecto. Asimismo, mi reconocimiento a mi compañero de estudios, Bryan Yagual, cuya amistad y respaldo han sido indispensables a lo largo de mi carrera. Por último, pero no por ello menos importante, estoy profundamente agradecido con mis amigos y familiares, quienes me han proporcionado invaluable asesoramiento, sabios consejos y confianza. Sin su presencia y apoyo, este logro no hubiera sido posible.

Aaron Soria

Declaración Expresa

"Los derechos de titularidad y explotación, nos corresponde conforme al reglamento de propiedad intelectual de la institución; Bryan Andrés Yagual Abata y Aaron Andrés Soria Triviño y damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"



Bryan Andrés Yagual Abata

Aaron Andrés Soria Triviño

Evaluadores

Firmado electrónicamente por:
MARIA ANTONIETA
ALVAREZ VILLANUEVA

PhD. María Álvarez Villanueva

Profesor de Materia

PhD. Juan Avilés Castillo

Tutor de proyecto

Resumen

El presente proyecto de tesis desarrolla un sistema de detección de asistencia para eventos empleando la tecnología de Identificación por Radiofrecuencia (RFID). El objetivo es proporcionar un método más eficiente y preciso para el registro de asistentes, superando las limitaciones de los sistemas manuales. Se realiza un estudio exhaustivo de las etiquetas RFID, tanto pasivas como activas, evaluando su alcance, coste y aplicabilidad en diferentes tipos de eventos. La selección y configuración de microcontroladores, particularmente el ESP32, juega un papel crucial en el sistema, permitiendo una gestión efectiva y en tiempo real de los datos recogidos. Los resultados obtenidos demuestran que el uso de la tecnología RFID pasiva, combinada con microcontroladores avanzados, ofrece una solución práctica y rentable para la gestión automatizada de la asistencia en eventos. El sistema no solo mejora la precisión del registro, sino que también proporciona datos valiosos para la organización y seguimiento. Además, presenta posibilidades de escalabilidad y adaptación a diferentes tipos y tamaños de eventos.

Palabras clave: Detección de Asistencia, Tecnología RFID, Microcontroladores ESP32, Gestión de Eventos, Automatización de Registro, FireBase.

Abstract

This thesis project develops an attendance detection system for events using Radio Frequency Identification (RFID) technology. The goal is to provide a more efficient and accurate method for attendee registration, overcoming the limitations of manual systems. An exhaustive study of RFID tags, both passive and active, evaluating their range, cost, and applicability in different types of events, is conducted. The selection and configuration of microcontrollers, particularly the ESP32, play a crucial role in the system, allowing for effective and real-time management of the collected data. The results obtained demonstrate that the use of passive RFID technology, combined with advanced microcontrollers, offers a practical and cost-effective solution for automated event attendance management. The system not only improves the accuracy of the registration but also provides valuable data for organization and tracking. Furthermore, it presents possibilities for scalability and adaptation to different types and sizes of events.

Keywords: Attendance Detection, RFID Technology, ESP32 Microcontrollers, Event Management, Registration Automation, Firebase.

Índice general

Abreviaturas

ESPOL	Escuela Superior Politécnica del Litoral
RFID	Identificación por radio frecuencia
MOSI	Master Output Slave Input
IoT	Internet of Things
RST	Reset
Wi-Fi	Wireless Fidelity
TX	Transmisión
SCK	Serial Clock
RX	Recepción
GND	Ground
IRQ	Interrupt Request
IDE	Integrated Development Environment
MISO	Master Input Slave Output
PIR	Passive Infrared
GPIO	General Purpose Input/Output
AC	Corriente Alterna
DC	Corriente Continua
ID	Identificador
SDA	Serial Data Signal

Simbología

GHz	Gigahercios
MHz	Megahercios
Mbps	Megabits por Segundo
V	Voltio
m	Metro
km	Kilometro
dBm	Decibelio-Milivatio
h	Hora
EHF	Extremely High Frequency
LF	Low Frequency
SLF	Super Low Frequency
HF	High Frequency
UHF	Ultra High Frequency
MF	Medium Frequency
ULF	Ultra Low Frequency
VLF	Very Low Frequency

Índice de figuras

Índice de tablas

Capítulo 1

1. INTRODUCCIÓN

En un contexto cada vez más globalizado y enfocado en la interconexión hacia eventos como, conferencias, congresos y clases educativas, uno de los desafíos logísticos y de seguridad más significativos radica en la gestión eficaz de la asistencia. La supervisión precisa de la entrada y salida de asistentes no es sólo una cuestión de orden, sino más bien un factor esencial para asegurar la seguridad, el confort y la eficiencia en las operaciones. Tradicionalmente, los sistemas de control de acceso se han basado en métodos manuales o semiautomatizados, incluyendo listas de papel, códigos de barras (BR) y lectores biométricos. Sin embargo, estos sistemas presentan vulnerabilidades y limitaciones evidentes: son susceptibles a errores humanos, fraude, y pueden resultar ineficientes en eventos con gran afluencia de público, provocando largas filas y demoras que afectan negativamente la experiencia del asistente.

La utilización de un sistema automatizado, tal como el basado en la Identificación por Radiofrecuencia (RFID), en la gestión de eventos resuelve alguna de las dificultades y limitaciones mencionadas. Sin esta automatización, el proceso de registro y control de acceso depende en gran medida de intervenciones manuales, lo que no solo ralentiza el flujo de asistentes, sino que también aumenta la probabilidad de errores, de inconsistencias en la recolección y gestión de datos. Estos errores pueden derivar en situaciones problemáticas, como el acceso no autorizado o el conteo inexacto de asistentes, afectando tanto la seguridad como la capacidad de toma de decisiones basadas en datos confiables. Además, en un entorno no automatizado, se pierden oportunidades valiosas para analizar y comprender el comportamiento y las preferencias de los asistentes, aspectos cruciales para la mejora continua.

1.1 Descripción del Problema

La gestión de asistencia en diversos eventos, tales como reuniones de trabajo o clases académicas, ha sido tradicionalmente un proceso manual, sujeto a errores humanos y a menudo ineficiente. La falta de un sistema automatizado y confiable de registro de asistencia puede llevar

a inexactitudes, pérdida de tiempo y posibles conflictos derivados de registros incorrectos. Estos problemas se magnifican cuando el número de asistentes es elevado o cuando se requiere un control constante y detallado de la asistencia.

La legibilidad de la escritura, las distracciones momentáneas o la simple omisión de un nombre pueden conducir a registros inexactos. En un evento grande, estos errores se acumulan rápidamente. Además, en eventos de cualquier magnitud, el registro manual puede ser un proceso tedioso y consumir un tiempo valioso que podría destinarse a actividades más productivas o al contenido central del evento. Por otro lado, preocupaciones adicionales como la seguridad y confidencialidad se hacen presentes; las listas físicas pueden ser fácilmente extraviadas o vistas por personas no autorizadas, lo que genera potenciales problemas de privacidad. Estas inexactitudes y problemas de privacidad, a su vez, pueden dar lugar a disputas sobre la asistencia, poniendo en riesgo las relaciones laborales o académicas. Aunque la gestión manual podría parecer económico a primera vista, esta gestión puede incurrir en costos ocultos considerables cuando se toma en cuenta el tiempo, los materiales y la rectificación de errores.

A su vez, en el contexto actual donde la tecnología se integra cada vez más en todas las facetas de la vida diaria y laboral, se hace evidente la necesidad de optar por sistemas automatizados que se adapten a los nuevos desafíos y demandas. Estos sistemas especializados no solo optimizarían la detección y registro de asistencia, sino que también permitiría una integración con bases de datos y aplicaciones informáticas, brindando una visión más completa y precisa de los patrones de asistencia.

1.2 Justificación de la solución

La rápida evolución de las tecnologías y su integración en las diversas áreas de la vida diaria y profesional enfatizan la importancia de contar con sistemas eficientes y confiables, especialmente para tareas esenciales como el registro de asistencia. Las dificultades asociadas con los sistemas manuales y basados en la web, tales como errores e interrupciones debido a problemas

de conectividad, subrayan la urgencia de encontrar soluciones más robustas. A pesar de los esfuerzos realizados mediante el uso de páginas web académicas y aplicaciones de control de asistencia, persisten desafíos, en particular la lentitud de ciertos aplicativos al registrar asistencia.

Se ha explorado la implementación de un sistema automatizado para el registro de asistencia, aprovechando tecnologías como el Bluetooth, sistemas de visión artificial y tecnología LED. Sin embargo, a pesar de las ventajas potenciales de esta aproximación, existen desafíos significativos que afectan su precisión. Entre las limitaciones identificadas se encuentran la incompatibilidad del sistema con diversos sistemas operativos en dispositivos móviles y una alta sensibilidad en la decodificación vinculada al rendimiento de los indicadores usados. [1]

Por otro lado, la implementación de un sistema biométrico de huellas dactilares para controlar la asistencia de profesores puede ofrecer precisión y seguridad, pero de igual forma íntegra un nuevo proceso de toma de datos previo, en el cual es necesario que estén presentes los posibles asistentes a eventos con personal apto para manejar esta tecnología, lo cual generaría costos indirectos y un proceso de logística adecuado. Alternativamente, la tecnología RFID ofrece una serie de ventajas que pueden ser más adecuadas para el control de asistencia a eventos, tales como; velocidad de registro, coste, escalabilidad y expansión. [1]

Es crucial, por tanto, investigar y desarrollar un prototipo que no solo automatice el proceso de registro de asistencia, sino que también sea confiable y fácil de integrar con plataformas existentes. Se busca diseñar un sistema de registro automatizado respaldado por una base de datos que valide con rapidez y precisión la asistencia a eventos mediante la tecnología RFID. Una solución de esta naturaleza no solo mejoraría la experiencia de los asistentes y organizadores, sino que también sentaría un precedente para futuras innovaciones en la gestión de eventos. Dado el creciente énfasis en la eficiencia y la toma de decisiones basada en información, llenar este vacío con una solución tecnológica adecuada se presenta como una oportunidad esencial para el sector productivo. [2]

1.3 Objetivos

1.3.1 Objetivo general

Implementar un sistema basado en tecnología RFID para la detección y registro automatizado de asistencia a eventos.

1.3.2 Objetivos específicos

- Investigar las capacidades y limitaciones actuales de la tecnología RFID en la detección y registro de asistencia, para realizar un sistema eficaz y robusto.
- Desarrollar un sistema de registro de asistencia a eventos basado en tecnología RFID, considerando las buenas prácticas y estándares actuales.
- Analizar los resultados del sistema propuesto, para la evaluación del desempeño en el registro de asistencia.

1.4 Marco teórico

1.4.1 Tecnología RFID

En el contexto actual, donde la eficiencia y rapidez son fundamentales, la implementación de un sistema de detección de asistencia para eventos se ha vuelto imprescindible. La tecnología de Identificación por Radiofrecuencia (RFID) emerge como una solución óptima, destacando por su eficacia y automatización. A diferencia de los sistemas tradicionales, como los códigos de barras que se basan en la lectura óptica de patrones impresos, RFID utiliza microchips que le permiten almacenar códigos alfanuméricos. Estos microchips transmiten la información mediante ondas de radiofrecuencia, lo que permite una captura de datos más rápida. [3]

Los sistemas RFID se caracterizan por su capacidad para el almacenamiento y recuperación remota de datos. Operan mediante el uso de etiquetas o tags, que pueden contener desde un único bit hasta varios kilobytes de información. Estas etiquetas, al ser activadas por las señales de radiofrecuencia emitidas por los lectores, responden transmitiendo los datos almacenados. Este

método de transmisión inalámbrica ofrece ventajas significativas sobre los métodos basados en señales ópticas, tales como una mayor flexibilidad en el posicionamiento de las etiquetas. [3]

Los sistemas RFID pueden operar en diversas bandas de frecuencia, incluyendo 125 KHz, 13.56 MHz, y bandas UHF como 860-950 MHz y 2.45 GHz, cada una con características específicas que se adaptan a diferentes aplicaciones y entornos. Esta versatilidad hace de la tecnología RFID un recurso poderoso y adaptable para una variedad de aplicaciones, incluyendo la gestión de asistencia en eventos.

Figura 0.1.1.- Elementos de un Sistema RFID

Nota: La imagen ilustra los componentes clave de un sistema RFID, destacando el flujo de información inalámbrica desde la etiqueta hasta la plataforma de gestión de datos. Este diagrama es esencial para entender cómo cada parte del sistema colabora para lograr una identificación y captura de datos efectiva y eficiente. [3]

1.4.2 Frecuencias en la tecnología RFID

Las ondas de radiofrecuencia (RF), esenciales en la tecnología RFID, se originan por el flujo de corriente alterna a través de un conductor y se distinguen por dos parámetros: su frecuencia medida en hercios (Hz) y su longitud de onda expresada en metros (m). Estas ondas se desplazan a través del espacio a la velocidad de la luz, y la relación entre la velocidad de la luz (c), la frecuencia y la longitud de onda constituye un principio fundamental en el campo de las RF. [4]

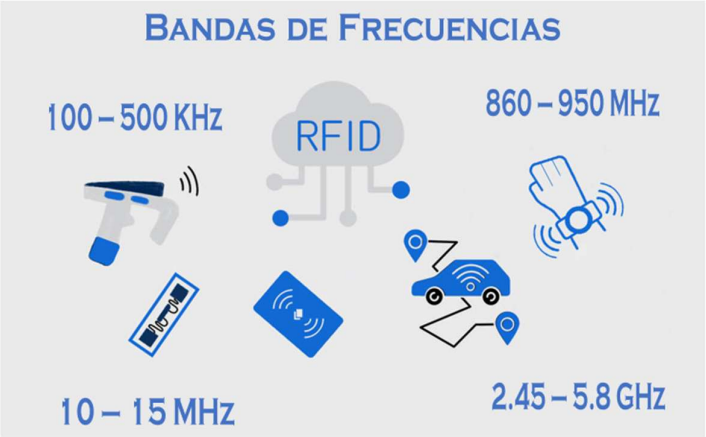


Figura 1.0.2.- Frecuencias RFID.

Nota: La imagen proporciona una representación visual de las cuatro principales categorías de frecuencias utilizadas en la tecnología RFID, destacando la relación entre la frecuencia operativa y el rango de detección. [5]

La tecnología RFID aprovecha específicamente distintas bandas de frecuencias: baja (LF), alta (HF), ultra alta (UHF) y microondas (MW), cada una abarcando un rango que va desde los kilohercios (kHz) a los gigahercios (GHz).

El control sobre la asignación de frecuencias es crucial para el funcionamiento de esta tecnología, ya que existen variados tipos diferenciadas principalmente por su rango de frecuencia de operación. Los dispositivos RFID operan en distintas bandas de frecuencia: la banda de baja frecuencia, que va de 100 a 500 KHz y permite una distancia de lectura de hasta un metro y medio; la banda de alta frecuencia, en el rango de 10 a 15 MHz, con un alcance de lectura de menos de dos metros; la banda de frecuencia más alta, de 860 a 950 MHz, incluyendo dispositivos de microondas de 2.45 GHz – 5.8 GHz.

Banda de Frecuencia	Características	Aplicaciones Comunes
Baja **100 – 500 KHz**	Velocidad de Lectura Baja. Sistemas con Tags económicos. Lectura para corta y mediana distancia.	Control de acceso, seguimiento de animales.
Intermedia **10 – 15 MHz**	Lectura para corta y mediana distancia. Velocidad media de lectura.	Gestión de bibliotecas, control de acceso, tarjetas inteligentes.
Alta **860 – 950 MHz** **2.4 – 5.8 GHz**	Línea de vista requerida. Tecnología costosa. Lectura para corta y mediana distancia. Velocidad de lectura alta.	Monitoreo en sistemas ferroviarios y automovilísticos, gestión de acceso y control en sistemas de peaje.

Tabla 1.- Bandas de Frecuencia y usos en etiquetas RFID.

Nota: La tabla proporciona una visión clara y concisa de las distintas Banda de frecuencias RFID y su utilidad práctica. [5]

En la Tabla 1 se puede observar las características distintivas y aplicaciones específicas de la tecnología RFID. Sistemas operando en Frecuencia Baja (LF) se basan en el acoplamiento inductivo mostrando menor susceptibilidad a la interferencia de líquidos y metales, lo que la hace ideal para entornos desafiantes. Por su parte, aquellos sistemas que operan en Alta Frecuencia (HF) son convenientes para la integración en objetos pequeños como tarjetas de crédito y pasaportes debido a su capacidad para trabajar con tamaños de etiqueta reducidos. [4]

En el caso de la Ultra Alta Frecuencia (UHF), estos sistemas se destacan por permitir la lectura simultánea de múltiples etiquetas y ofrecer un mayor alcance de detección, ampliando así sus posibilidades de uso en la gestión de inventarios y seguimiento de activos. Finalmente, las etiquetas de Microondas (MW) son reconocidas por sus altas velocidades de transferencia de datos, aunque esto las hace más propensas a las interferencias. [5]

La interacción entre el lector RFID y las etiquetas se produce cuando el lector emite señales de RF, alterando los campos eléctricos y magnéticos circundantes. Los tags responden generando una corriente en su antena que les permite comunicarse con el lector. Este fenómeno, conocido como acoplamiento inductivo, es característico de las etiquetas pasivas LF y HF. Por otro lado, las etiquetas UHF y MW emplean una técnica diferente denominada modulación backscatter, donde modulan y reflejan la señal del lector para transmitir información. [4]

En términos de potencia de la señal RF, esta se mide en milivatios (mW) y se expresa en decibelios-milivatios (dBm). El dBm es la unidad de referencia utilizada para denotar estos niveles de potencia en términos de milivatios.

1.4.3 Lectores RFID

Los lectores RFID son dispositivos esenciales en el ámbito de la tecnología de identificación automática, funcionando como puentes entre las etiquetas RFID y los sistemas de procesamiento de datos. Estos dispositivos se categorizan en: lectores fijos, diseñados para aplicaciones industriales o comerciales, donde su capacidad para manejar volúmenes altos de lectura es crucial; lectores portátiles, que proporcionan la comodidad y la eficiencia para el personal en movimiento mediante el uso de antenas integradas u opciones de conectividad inalámbrica como Bluetooth y lectores USB, que son idóneos para aplicaciones de escritorio o puntos de venta, donde el alcance y la complejidad del sistema son más limitados. [5]

Estos lectores están equipados para trabajar en diversas bandas de frecuencia, lo que les permite leer etiquetas desde baja frecuencia (LF) hasta la banda de microondas (MW), cada una con su respectivo alcance y aplicaciones. Esta flexibilidad asegura que puedan adaptarse a los requisitos específicos de cada aplicación, desde control de inventario hasta seguimiento de activos a gran escala.

En términos de funcionalidad, los lectores RFID operan en dos fases principales: emisión y recepción. Inicialmente, el lector genera una señal analógica ajustada a una frecuencia y potencia

determinada, que es transmitida a través de su antena. Luego, procesa la señal de retorno emitida por la etiqueta, que contiene información específica del objeto al que está adherida. Este intercambio de señales no solo establece la identidad de la etiqueta, sino que también convierte los datos en información procesable, que puede ser almacenada, analizada o utilizada para mejorar la toma de decisiones en tiempo real. Además, algunos lectores avanzados están dotados de capacidades adicionales como el filtrado de datos y la ejecución de comandos remotos, lo que aumenta su utilidad en entornos complejos.

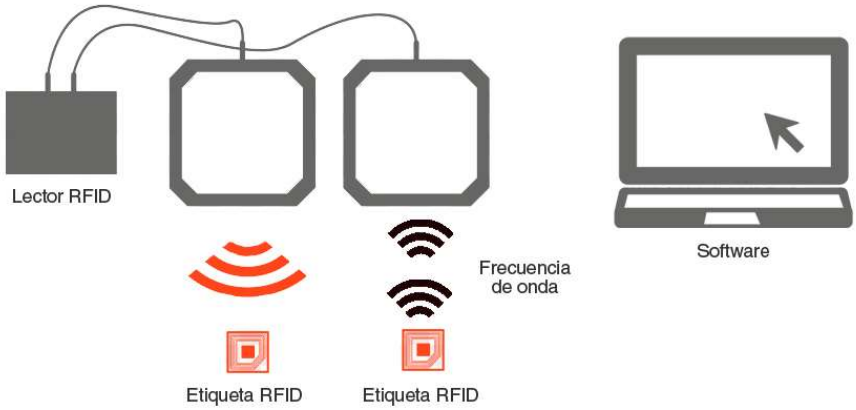


Figura 1. 0.3.- Sistema Básico de RFID.

Nota: La imagen facilita la comprensión del flujo de datos en un sistema RFID, desde la generación de la señal por parte del lector hasta el procesamiento final de la información por el software correspondiente. Este tipo de visualización es crucial para entender cómo las etiquetas RFID interactúan con su entorno y cómo la información es capturada y utilizada en aplicaciones prácticas. [5]

1.4.3.1 Etiquetas o TAGS RFID

Las etiquetas o TAGS RFID constituyen uno de los componentes más destacados en un sistema RFID, desempeñando un papel crítico a pesar de su diminuto tamaño. Estos dispositivos son capaces de almacenar y transmitir información mediante señales de radiofrecuencia, lo que resulta fundamental para la identificación precisa y el seguimiento efectivo. La funcionalidad de las etiquetas RFID es clave para optimizar la gestión de inventarios desempeñando un papel vital

en los sistemas de seguimiento y control. La selección de la etiqueta RFID más adecuada es determinante para asegurar la eficacia operativa del sistema.

Cada etiqueta RFID está compuesta por tres elementos esenciales: un sustrato que ofrece soporte físico y garantiza la durabilidad ante diferentes condiciones ambientales, una antena diseñada para recibir y transmitir señales en las frecuencias específicas del sistema, y un chip que integra memoria y microprocesador, siendo este último el encargado de procesar la información. Mientras que las etiquetas pasivas obtienen la energía necesaria del campo electromagnético generado por el lector RFID, las etiquetas activas se valen de una batería interna para su funcionamiento autónomo. La selección de estos componentes se debe realizar considerando factores como las condiciones ambientales del entorno de aplicación, los requerimientos de los procesos de lectura dentro de la cadena de suministro, y las necesidades específicas de capacidad de memoria y seguridad. Para la mayoría de las aplicaciones, los chips con capacidad de memoria reducida son suficientes para cumplir con los requerimientos de identificación y seguridad. [6]

Estas etiquetas se dividen en tres tipos:

- Las etiquetas pasivas RFID no requieren alimentación interna y son ampliamente utilizadas debido a su bajo costo. Funcionan activándose cuando se acercan a un lector que les suministra la energía necesaria para la comunicación, con un rango de funcionamiento que varía de 10 cm a 30 cm de acuerdo con la Tabla 2. Por otro lado, en la figura 1.4 se puede apreciar el tamaño y característica de esta etiqueta, de igual forma se puede evidenciar que no existe ninguna fuente de energía integrada.

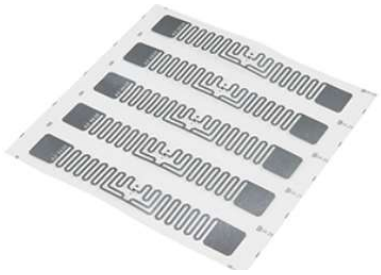


Figura 1. 0.4.-Etiqueta Pasiva RFID.

Nota: La imagen muestra un conjunto de etiquetas pasivas RFID, que son comúnmente utilizadas en una amplia gama de aplicaciones debido a su costo-efectividad y facilidad de producción en masa. [6]

- Las etiquetas activas, a diferencia de las pasivas, tienen una fuente de energía autónoma que les permite emitir señales al lector. Sin embargo, presentan diferencias significativas en términos de costos, tamaño y vida útil. Las etiquetas activas son mucho más costosas debido a la incorporación de una batería interna y tienen un tamaño mayor. Además, su vida útil es finita, en contraste con las etiquetas pasivas que podríamos categorizar como prácticamente ilimitadas en su duración, en la figura 1.5 podemos apreciar una etiqueta activa RFID y como tiene un tamaño diferente, puesto que necesita una fuente de alimentación interna para su correcto funcionamiento, ya que puede tener un alcance de 20 hasta 100 metros.

Figura 1.0.5.- Etiqueta Activa RFID.

Nota: Las etiquetas activas como esta son fundamentales para la gestión de activos en grandes áreas, como en la logística de almacenes o seguimiento de maquinaria, donde se requiere un rango de lectura extenso. [6]

- Las etiquetas semipasivas o semiactivas son dispositivos que funcionan utilizando una batería que proporciona la energía requerida para emitir y recibir señales de manera independiente, como se ilustra en la figura 1.6. Una de sus principales ventajas en comparación con las etiquetas pasivas es su alcance de lectura en larga distancia y su capacidad para funcionar en condiciones extremas. Aunque son similares a las etiquetas pasivas en el sentido de que no emiten señales por sí mismas, son más eficientes y tienen una velocidad de respuesta más rápida que las etiquetas pasivas, al mismo tiempo que consumen menos energía que las etiquetas activas.

Figura 1. 0.6.- Etiqueta Semipasiva RFID.

Nota: Esta etiqueta Semipasiva RFID es un dispositivo avanzado de seguimiento y monitorización que combina la eficiencia energética de las etiquetas pasivas con la funcionalidad mejorada de las activas, ideal para aplicaciones como el seguimiento de activos y la logística de cadena de frío. [6]

Tipo de Etiqueta	Fuente de Energía	Alcance	Ventajas	Desventajas
Pasivas RFID	Campo electromagnético del lector RFID	De 10 cm a 30 cm	Costo bajo, no requiere batería, vida útil larga	Menor alcance, dependiente de la potencia del lector
Activas RFID	Batería interna	De 20 hasta 100 metros	Alcance largo, señal más fuerte, capacidad de memoria mayor	Costo más alto, mantenimiento de batería, tamaño más grande
Semipasivas RFID	Batería interna para el chip, pero reciben energía del lector para la transmisión	Hasta 50 metros	Mejor alcance que las pasivas, más económicas que las activas	Requieren cambio de batería, aunque menos frecuente que las activas

Tabla 2.- Tabla Comparativa de Etiquetas RFID.

Nota: Esta tabla proporciona una comparación clara entre las etiquetas pasivas, activas y semipasivas RFID. La elección entre estas etiquetas dependerá del balance deseado entre alcance, duración de la batería, tamaño, memoria y costo. [6]

1.4.4 Microcontroladores

Los microcontroladores juegan un papel indispensable en los sistemas RFID, actuando como el cerebro de los lectores y a menudo de las propias etiquetas. Estos componentes electrónicos se encargan de ejecutar el software que permite la comunicación y el procesamiento de la información recogida por las antenas RFID.

En un sistema RFID, el microcontrolador interpreta las señales electromagnéticas recibidas, ejecuta los algoritmos de decodificación, y gestiona la transmisión de datos hacia sistemas de almacenamiento o procesamiento más complejos. Por ejemplo, en un lector RFID, el microcontrolador puede filtrar y analizar los datos recogidos de múltiples etiquetas antes de enviar solo la información relevante al sistema central.

La elección del microcontrolador adecuado para un sistema RFID depende de varios factores, como el rango de operación requerido, la cantidad de datos a procesar, y la complejidad del software. Microcontroladores populares como el Arduino y el ESP32 ofrecen diferentes ventajas. Arduino, con su amplia comunidad y facilidad de uso, es ideal para prototipos y proyectos educativos. Por otro lado, el ESP32 ofrece capacidades superiores de conectividad inalámbrica y procesamiento.

Comparando distintos microcontroladores, las consideraciones incluyen la velocidad de procesamiento, número y tipo de puertos de entrada/salida, capacidades de comunicación inalámbrica y la memoria disponible. Por ejemplo, un microcontrolador con un módulo Wi-Fi integrado como el ESP32 podría ser más adecuado para aplicaciones que requieren conectividad remota, mientras que un microcontrolador con una gran cantidad de pines de entrada/salida como el Arduino Mega puede ser preferible para sistemas que necesitan conectarse a múltiples sensores o actuadores.

La elección correcta del microcontrolador puede mejorar significativamente el rendimiento del sistema y expandir sus capacidades, permitiendo su aplicación en una gama aún más amplia de escenarios industriales y comerciales.

1.4.4.1 Microcontrolador ESP32

Espressif Systems, compañía innovadora con sede en China, es reconocida por la creación del ESP32, un chip avanzado que sucede y mejora considerablemente al ESP8266. Este System on a Chip (SoC) integra capacidades de conectividad Wi-Fi y Bluetooth, haciéndolo extraordinariamente apto para la interconexión y comunicación con una variedad de dispositivos en la era del Internet de las Cosas (IoT). Con su diseño ingeniosamente compacto, el ESP32 se adapta con facilidad en placas de circuito impreso (PCB).

Dotado de un procesador de doble núcleo, el Xtensa LX6, este SoC es capaz de manejar múltiples procesos en paralelo, mejorando el rendimiento y ofreciendo una flexibilidad excepcional para desarrolladores y entusiastas de la tecnología, facilitando así la creación de soluciones complejas y multifacéticas en el campo de la automatización y los sistemas inteligentes. [7] A continuación, se puede observar en la figura 1.7 el microcontrolador ESP32:

Figura 1.0.7.- Microcontrolador ESP32.

Nota: La imagen muestra el microcontrolador ESP32, un componente esencial en el desarrollo de proyectos de IoT, robótica y automatización, gracias a su versatilidad y poderosas funciones integradas. [7]

Los microcontroladores ESP32 destacan por su impresionante gestión del consumo energético, especialmente a través del modo "deep sleep", que reduce significativamente el uso de energía al desactivar la CPU, la mayoría de la RAM y todos los periféricos digitales, manteniendo en funcionamiento solo partes críticas como las memorias RTC.

Estos SoC también están diseñados para optimizar la conectividad inalámbrica, incorporando amplificadores de potencia y de bajo ruido, junto con filtros y módulos de gestión de energía, para maximizar la eficiencia. Con una gama de modelos como el ESP32-WROOM y ESP32-CAM, cada uno con periféricos específicos, los ESP32 son versátiles para diversas aplicaciones y compatibles con el entorno de desarrollo Arduino IDE, lo que facilita su implementación en múltiples proyectos. [7]

Pines del ESP32: Fundamentales y Versátiles

Los pines del ESP32 desempeñan un papel crítico en la estructura de este microcontrolador, cumpliendo el rol de conexiones vitales que posibilitan la interacción con el entorno, la comunicación con otros dispositivos y el control de componentes electrónicos. Estos pines se utilizan en una variedad de tareas, que abarcan desde la lectura de sensores hasta la supervisión de actuadores, así como la comunicación con periféricos externos. La versatilidad de los pines radica en su habilidad para adaptarse a diferentes funciones según las necesidades particulares de cada proyecto. Esta adaptación se logra configurando los pines en modos específicos, como entrada, salida, PWM, I2C, SPI, entre otros. Además, estos pines pueden colaborar en conjunto para habilitar operaciones complejas en proyectos que involucran tanto hardware como software.

Una comprensión completa de las funciones y capacidades de los pines del ESP32 resulta esencial para maximizar el potencial de este microcontrolador en una amplia gama de aplicaciones. Tanto en proyectos de Internet de las Cosas (IoT) destinados a la recopilación de datos, como en aplicaciones robóticas que requieren el control preciso de motores y sensores, o en sistemas de

automatización diseñados para gestionar diversos dispositivos, los pines del ESP32 desempeñan un rol central. Su versatilidad permite adaptarse a diversos contextos y requisitos, convirtiendo este microcontrolador en un componente esencial para la implementación exitosa de una variedad de proyectos.

Capítulo 2

2. AUTOMATIZACIÓN DE ASISTENCIA A EVENTOS

Al abordar la implementación de este sistema de detección de asistencia, se detallan los métodos y tecnologías empleadas para garantizar el registro de la asistencia a eventos. El diseño integral del sistema se presenta a través de diagramas metodológicos y de flujo, proporcionando una visión detallada del funcionamiento y del proceso de implementación.

La sección de implementación aborda aspectos fundamentales, como la elección de sistemas RFID pasivos o activos, microcontroladores, las configuraciones de hardware y software necesarias, así como los códigos de programación específicos y las interconexiones esenciales para lograr con éxito la implementación del sistema de detección a eventos. Se enfatiza la importancia de la interoperabilidad del sistema, permitiendo una integración fluida con otras soluciones tecnológicas, lenguajes de programación y plataformas existentes.

2.1 Elección estratégica de sistemas RFID

Este capítulo se enfoca en el análisis de dos sistemas principales de Identificación por Radio Frecuencia (RFID) destinados a la detección de asistencia en eventos: el RFID con etiquetas pasivas y el RFID con etiquetas activas. Además, se llevó a cabo un examen detallado del tipo de microcontroladores que se emplearán, considerando que todos estos parámetros contribuirán significativamente al rendimiento óptimo del sistema. Los dos microcontroladores bajo análisis son el ESP-32 y el ARDUINO-MEGA.

Por otro lado, se analizaron diversas variables como el alcance de la señal, la durabilidad, la capacidad de almacenamiento de datos, la velocidad de procesamiento y los costos asociados. El análisis de estos elementos proporciona una visión integral de las capacidades y limitaciones de cada sistema y microcontrolador, permitiendo así la toma de decisiones informadas para la implementación efectiva del sistema de detección de asistencia en eventos.

2.2 Análisis TAGS RFID Pasivo y Activo

En esta sección, se aborda el análisis de los tags RFID, tanto pasivos como activos, elementos cruciales en el ámbito de la identificación por radiofrecuencia. El RFID (Identificación por Radiofrecuencia) es una tecnología que emplea ondas de radio para la identificación automática de objetos, individuos o animales, involucrando componentes como una etiqueta o "tag," un lector y un sistema de procesamiento de datos.

2.2.1 Etiquetas RFID Pasivo

En el contexto de este estudio, se realizó un análisis exhaustivo sobre la implementación de la tecnología RFID pasiva en la administración de asistencia para eventos de pequeña y mediana escala. Las etiquetas RFID pasivas, que se caracterizan por su falta de fuente de energía interna y se activan mediante la energía emitida por los lectores RFID, exhiben un conjunto de propiedades distintivas que las hacen particularmente apropiadas para su aplicación en ciertos tipos de eventos. El análisis se enfoca en detallar estas características, en el contexto específico de la gestión de asistencia, con el fin de proporcionar una perspectiva integral y fundamentada sobre su utilidad y aplicabilidad en dicho ámbito. [8]

Características Principales

- **Frecuencia y Alcance**: La banda de frecuencias de LF cubre desde 30 kHz hasta 300 kHz. Los sistemas típicos de identificación por radiofrecuencia de baja frecuencia (RFID LF) funcionan a 125 kHz o 134 kHz. El alcance de lectura es aproximadamente de 10 cm, pero destacan por su robusta capacidad para resistir a las interferencias externas.
- **Costo:** Al no contener baterías, estas etiquetas son más económicas y duraderas en comparación con las activas, lo que las hace ideales para la implementación a gran escala y para eventos recurrentes. El costo de los Tags RFID para frecuencias de 125 kHz esta alrededor de $ 0.08 a $0.53 centavos por etiqueta. Además, existen varios formatos: reloj, llavero o tarjeta. [8]
- **Sencillez y Facilidad de Uso:** La etiqueta pasiva se distingue por su diseño minimalista, lo cual no solo contribuye a su tamaño compacto, sino que también influye positivamente

en su facilidad de uso. Esta característica, respaldada por una estructura simple, facilita la operación del sistema en el que se implementa, proporcionando una solución eficaz y accesible para diversas necesidades.

- **Ámbitos de aplicación:** Debido a que el alcance de lectura es limitado dado a la frecuencia (LF RFID), se tendrá diversas aplicaciones tales como: Controles de acceso y rutas, inmovilizadoras, lavandería, lectura de gas, identificación de animales, gestión de inventario.
- **Vida útil:** Las etiquetas pasivas tienen una vida útil más larga y son menos propensas a fallas, ya que no dependen de una fuente de energía que pueda agotarse. La durabilidad de las etiquetas RFID pasivas está determinada por los materiales de construcción y las condiciones ambientales en las que se utilicen. En entornos poco hostiles, estas etiquetas tienen la capacidad de mantenerse operativas durante un período que puede extenderse hasta dos décadas.

2.2.2 Etiquetas RFID Activo

En este estudio se realizó un análisis exhaustivo sobre el uso de la tecnología RFID activa en el monitoreo de asistencia en eventos de pequeña y mediana escala. Las etiquetas RFID activas, a diferencia de sus homólogas pasivas, están equipadas con su propia fuente de energía, permitiéndoles emitir señales de manera independiente y con un alcance más amplio. Esta capacidad esencial otorga a las etiquetas activas una serie de beneficios destacados, en particular en lo que respecta al alcance de detección y a la confiabilidad de la señal, haciéndolas particularmente útiles en determinadas situaciones de eventos.

Características Principales

- **Frecuencia y Alcance:** Las etiquetas RFID activas operan en las bandas de frecuencia de UHF (433 MHz y 868 MHz a 2.4 GHz), lo que les permite un alcance de lectura mucho mayor, que puede extenderse hasta varios cientos de metros. Esta capacidad de largo alcance las hace ideales para aplicaciones de seguimiento en tiempo real y para entornos donde la distancia de lectura es un factor crítico.

- **Costo:** Las etiquetas activas, al incorporar baterías y circuitos más complejos, son generalmente más costosas que las pasivas, con precios que pueden variar significativamente en función de sus capacidades y características. A pesar de su mayor costo inicial, su mayor alcance y funcionalidades avanzadas justifican su inversión en aplicaciones específicas. El costo de los Tags RFID activas están alrededor de $30.00 a $ 100 dólares por etiqueta (valor de referencia local). [8]

- **Sencillez y Facilidad de Uso:** A pesar de su mayor complejidad interna, las etiquetas RFID activas son diseñadas para ser igualmente sencillas de usar. A menudo vienen con características adicionales como sensores de temperatura, movimiento o luz, y pueden configurarse para comunicarse con otros dispositivos o redes, lo que facilita su integración en sistemas más amplios de seguimiento y gestión.

- **Ámbitos de aplicación:** Las etiquetas RFID activas son ideales para aplicaciones que requieren un seguimiento preciso y en tiempo real, como la gestión de activos a gran escala, seguimiento de vehículos, logística y cadena de suministros, y aplicaciones de seguridad. Su capacidad para transmitir señales a grandes distancias las hace también adecuadas para el seguimiento de activos en grandes instalaciones industriales o comerciales.

- **Vida útil:** La vida útil de las etiquetas RFID activas está generalmente limitada por la duración de su batería. Sin embargo, las tecnologías avanzadas de gestión de energía y los modos de bajo consumo han aumentado significativamente su longevidad. Además, la posibilidad de reemplazar o recargar las baterías puede extender aún más su vida útil, haciéndolas una opción viable a largo plazo en muchas aplicaciones. En cuanto a durabilidad, dependen de la vida útil de la batería,

que generalmente dura entre 3 y 5 años, aunque este período puede variar según el uso y el tipo de batería.

2.2.3 Selección de alternativa

Antes de tomar una decisión en esta área, es crucial evaluar ciertas características adicionales específicas. La tecnología RFID activa se distingue por utilizar una batería interna en la etiqueta, manteniendo activos de forma continua tanto la etiqueta como su circuito de comunicación por radiofrecuencia. Por otro lado, la RFID pasiva depende de la energía generada por el lector y transmitida a su antena para su funcionamiento.

La RFID pasiva funciona reflejando la energía proveniente del lector y utilizando una pequeña cantidad de esta energía para formar su respuesta. Necesita señales intensas del lector, y la potencia de la señal que regresa a la etiqueta depende de la energía que pueda captar. En contraste, la RFID activa puede captar señales de muy baja intensidad, dado que no requiere activación por parte del lector, y la etiqueta es capaz de emitir señales potentes gracias a su propia fuente de energía interna, las cuales son enviadas de vuelta al lector.

La tabla 3 y 4 presenta comparativas que facilitan la toma de decisiones al seleccionar la alternativa más adecuada.

	RFID Activa	***RFID Pasiva***
Fuente de alimentación de las etiquetas	*Incorporada en la Tarjeta.*	Proviene del lector a través de radiofrecuencia.
Batería de las etiquetas	*Presente*	*Ausente*
Suministro de Energía a las Etiquetas	*Constante*	Solo cuando está dentro del alcance del lector.
Intensidad de la señal necesaria del lector a la etiqueta	*Menor*	*Mayor*
Intensidad de la señal emitida por la etiqueta al lector	*Mayor*	*Menor*

Tabla 3.- Diferencias técnicas RFID Activas vs Pasiva.

Tabla 3

Nota: Distinciones técnicas entre las tecnologías RFID activa y pasiva. [8]

	RFID Activa	*RFID Pasiva*
Frecuencia y Alcance	Trabaja en las frecuencias de UHF, específicamente en 433 MHz, 868 MHz a 2.4 GHz. Ofrece una capacidad de lectura considerablemente ampliada, alcanzando distancias que pueden extenderse hasta varios cientos de metros.	Trabaja en frecuencias de la banda LF abarca desde 30 kHz hasta 300 kHz. Ofrece un alcance de lectura limitado, aproximadamente 10 cm.
Tamaño y peso	Voluminoso y pesado	Reducido y Liviano
Funcionalidad del Sensor	Capaz de monitorear de manera constante y registrar datos del sensor.	Solo puede leer y transmitir datos del sensor cuando es activada por el lector.
Capacidad de Almacenamiento de datos	Gran capacidad de almacenamiento (128kb) con funciones avanzadas de búsqueda y acceso a datos.	Limitada capacidad de almacenamiento para leer y escribir datos (en bytes).
Aplicaciones	Controles de acceso y rutas, inmovilizadoras, lavandería, lectura de gas, identificación de animales, gestión de inventario.	La gestión de activos a gran escala, seguimiento de vehículos, logística y cadena de suministros, y aplicaciones de seguridad

Tabla 4.- Capacidades funcionales RFID Pasivos vs Activo.

Tabla 4

Nota: Funcionalidades de las tecnologías RFID activa y pasiva. [8]

Una vez analizado cada uno de los apartados y las correspondientes tablas 3 y 4, se eligió la tecnología RFID pasiva como alternativa inicial de prueba para la gestión de asistencia en eventos de pequeña y mediana escala. Algunas de las razones que fundamentan esta decisión, a pesar de su corto alcance, son las siguientes:

- **Eficiencia Costo-Beneficio:** Las etiquetas RFID pasivas presentan una ventaja económica significativa sobre las activas. Mientras que el costo de las etiquetas pasivas oscila entre $0.08 y $0.53 por unidad, las activas pueden costar entre $30.00 y $100.00 por etiqueta. Esta diferencia de precio es crucial, especialmente en la implementación a gran escala y en eventos recurrentes, donde la eficiencia en términos de costos es un factor decisivo. [8]
- **Durabilidad y Vida Útil:** Las etiquetas pasivas no dependen de una fuente de energía interna, lo que elimina la necesidad de mantenimiento relacionado con el reemplazo o la

recarga de baterías. Esto no solo simplifica la logística, sino que también aumenta su vida útil, la cual puede extenderse hasta dos décadas en condiciones favorables. En comparación, las etiquetas activas tienen una vida útil limitada por la duración de su batería, generalmente de 3 a 5 años.

- **Sencillez y Facilidad de Uso:** El diseño minimalista y la estructura simple de las etiquetas pasivas contribuyen a su facilidad de uso y a su adaptabilidad a diferentes contextos. A diferencia de las activas, que pueden requerir configuraciones más complejas debido a sus características adicionales, las etiquetas pasivas ofrecen una solución efectiva y accesible para una variedad de necesidades, sin la necesidad de conocimientos técnicos avanzados.

- **Aplicabilidad en Diversos Ámbitos:** Aunque las etiquetas pasivas tienen un alcance limitado, son idóneas para aplicaciones como controles de acceso, gestión de inventario y seguimiento en entornos cerrados o de tamaño reducido, que son comunes en eventos de pequeña y mediana escala. Esta limitación de alcance, en realidad, se convierte en una ventaja en escenarios donde se requiere control y precisión en áreas definidas.

- **Resistencia a Interferencias Externas:** Las etiquetas RFID pasivas operan en la banda LF, demostrando una notable resistencia a las interferencias externas. Esta característica es particularmente valiosa en entornos de eventos donde pueden existir múltiples fuentes de interferencia.

2.3 Análisis de microcontroladores

En esta sección se realiza un análisis detallado de la tecnología RFID pasiva para la gestión de asistencia en eventos de pequeña y mediana escala. Este análisis tiene como objetivo explorar las sinergias y posibles desafíos que surgen al combinar la eficiente tecnología RFID pasiva con la versatilidad y capacidades de estos dos microcontroladores.

En la siguiente sección, las ventajas y consideraciones prácticas de utilizar la tecnología RFID pasiva en conjunción con el ESP32, destacando su potencial para lograr una gestión de asistencia más eficiente y conectada. Posteriormente, se abordará la integración con el Arduino Uno, explorando cómo este microcontrolador puede contribuir a la implementación exitosa de sistemas de seguimiento de asistencia basados en RFID pasivo.

A través de este análisis comparativo, se busca proporcionar una visión completa de las capacidades y limitaciones de cada plataforma en combinación con la tecnología RFID pasiva, permitiendo una toma de decisiones informada para la implementación práctica en entornos específicos de eventos.

2.3.1 Microcontrolador ESP32 y RFID Pasivo

En el ámbito de la tecnología RFID pasiva, la implementación de sistemas eficientes y confiables es crucial. El Microcontrolador ESP32 emerge como una opción prominente debido a sus características avanzadas y su compatibilidad con RFID. Al integrar el ESP32 en sistemas con RFID pasivo, se abren posibilidades para crear soluciones inteligentes y conectadas, ideales para el seguimiento y control en diversos escenarios, incluyendo la asistencia a eventos. La figura 2.1 muestra las conexiones del sistema de prueba.

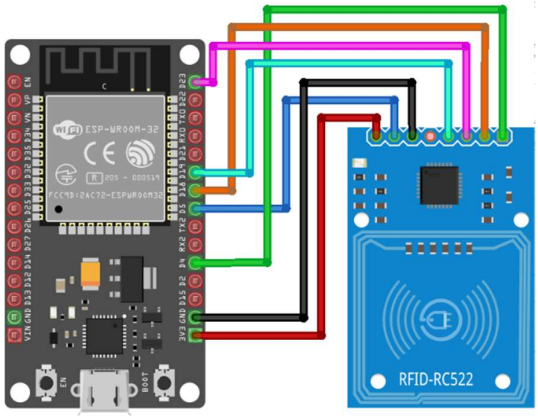


Figura 2.1.- Microcontrolador ESP32 Y RFID RC522.

Nota: La imagen muestra un diagrama de conexiones entre un Microcontrolador ESP32 y un módulo lector RFID RC522. Las conexiones están indicadas con líneas de colores que representan los cables que unen los pines específicos del ESP32 con los del módulo RFID. [9]

La tabla 5 que se presenta a continuación proporciona una visión detallada de las conexiones entre el microcontrolador ESP32 y los módulos RFID, especificando los pines utilizados en el ESP32, su respectiva función y el color del cable correspondiente para cada conexión. Esta tabla es esencial para entender cómo se realiza el cableado y la asignación de pines para facilitar la comunicación entre el microcontrolador y los módulos RFID, lo cual es fundamental para la configuración y el correcto funcionamiento de sistemas de detección de asistencia a eventos.

Definición en Código	Pin en ESP32	Función	Color de cable
#define Vcc	3V3 o Vin	Alimentación (3.3V o 5V)	Rojo
#define RST	D22	Reset	Azul
#define GND	GND	Tierra (Ground)	Negro
#define MISO	19	Master Input Slave Output	Turquesa
#define MOSI	23	Master Output Slave Input	Púrpura
#define SCK	18	Serial Clock	Naranja
#define SS / SDA	5	Slave Select / Serial Data	Verde
#define RST_PIN	22	Reset	Azul
#define SS_PIN2	21	Segundo Slave Select	No especificado

Tabla 5.- Conexiones ESP32 y RFID.

Características principales

- **Conectividad Avanzada:** El ESP32 no solo ofrece Wi-Fi y Bluetooth, sino también soporta Bluetooth Low Energy (BLE) y tiene capacidades de modo dual, lo cual es esencial para sistemas que requieren intercambio de datos en tiempo real y conexión con dispositivos móviles.

- **Capacidad de Procesamiento y Multitarea:** El procesador de doble núcleo del ESP32, junto con su capacidad para ejecutar tareas en paralelo, lo hace ideal para manejar simultáneamente la comunicación RFID y procesos de fondo como la conectividad de red y la gestión de datos.

- **Interfaces de Entrada/Salida Versátiles:** Además de los pines digitales y analógicos, el ESP32 soporta interfaces como SPI, I2C y UART, lo que permite una amplia gama de conexiones con sensores, lectores RFID y otros dispositivos periféricos.

- **Pequeño Tamaño y Flexibilidad de Diseño:** Su tamaño compacto permite integrarlo en sistemas donde el espacio es un factor crítico. Además, su diseño modular facilita la implementación en diferentes tipos de aplicaciones RFID.

- **Eficiencia Energética:** El ESP32 es notable por su rendimiento en aplicaciones de bajo consumo, una característica esencial para sistemas RFID pasivos que operan con baterías o fuentes de alimentación limitadas.

- **Seguridad Mejorada:** La capacidad del ESP32 para implementar algoritmos de cifrado avanzados y medidas de seguridad en la comunicación inalámbrica lo convierte en una opción segura para sistemas donde la protección de datos es primordial.

- **Amplia Comunidad y Ecosistema de Desarrollo:** La popularidad del ESP32 ha generado una extensa comunidad de desarrolladores y una rica biblioteca de recursos y herramientas

de desarrollo, facilitando la experimentación, el aprendizaje y la implementación rápida de soluciones RFID.

La tabla 6 proporciona un resumen técnico del microcontrolador ESP32 fabricado por Espressif Systems. Este microcontrolador es un dispositivo altamente integrado y de bajo consumo, adecuado para aplicaciones en la detección de asistencia a eventos y otros proyectos de IoT.

Especificación	Detalle
Fabricante	Espressif Systems
Tipo de Dispositivo	Microcontrolador
Arquitectura	Xtensa® Dual-Core 32-bit LX6 microprocessor
Frecuencia de CPU	Hasta 240 MHz
Wi-Fi	802.11 b/g/n/e/i
Bluetooth	v4.2 BR/EDR y BLE
GPIOs	Hasta 34 pines
Memoria Flash	Hasta 16 MB (dependiendo del modelo)
Memoria SRAM	520 KB
Interfaces Periféricas	UART, SPI, I2C, I2S, PWM, ADC, DAC
Seguridad	WEP, WPA/WPA2 PSK/Enterprise, RSA, AES, SHA
Tensión de Alimentación	2.2V a 3.6V
Temperatura de Operación	-40°C a 125°C
Dimensiones	Varía según el módulo (ej. 18mm x 25.5mm)

Tabla 6.- Especificaciones Técnicas del ESP32.

Tabla 6

Nota: *Esta tabla proporciona un resumen de las especificaciones técnicas más importantes del ESP32. Es importante señalar que el ESP32 es conocido por su versatilidad, eficiencia energética y capacidades de conectividad, lo que lo hace una opción atractiva para proyectos de IoT, especialmente aquellos que integran la tecnología RFID.*

2.3.2 Microcontrolador Arduino MEGA 2560 y RFID Pasivo

El Arduino Mega, una plataforma de microcontrolador fundamentada en el ATmega2560, proporciona un mayor número de pines de entrada y salida, así como una memoria ampliada. Esto lo hace idóneo para proyectos más complejos, incluyendo sistemas RFID pasivos de alta gama. Este estudio nos ayuda a comprender sus propiedades y potencial, encaminándonos hacia una solución eficiente y escalable. La figura 2.2 muestra las conexiones realizadas entre el Microcontrolador ATmega2560 y el RFID RC522.

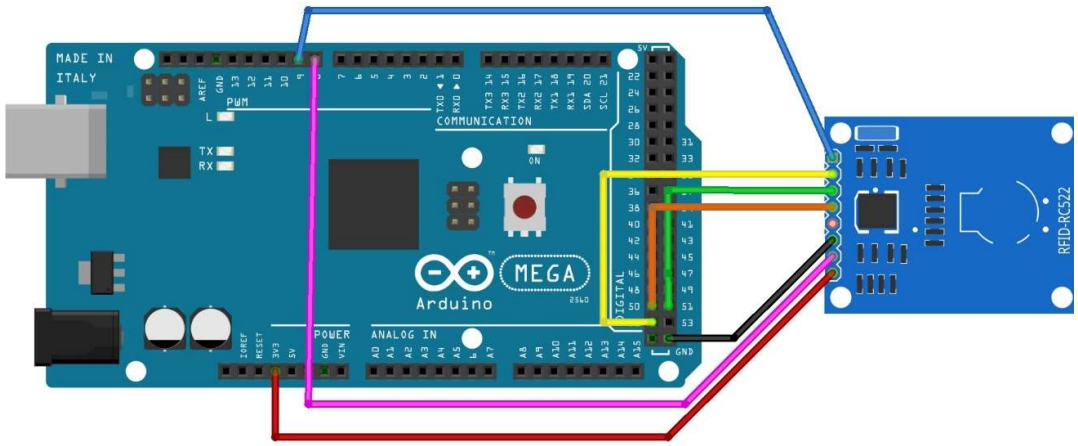


Figura 2.2.- Microcontrolador Arduino MEGA Y RFID RC522.

Nota: La imagen muestra un diagrama de conexiones entre un Microcontrolador Arduino MEGA y un módulo lector RFID RC522. Las conexiones están indicadas con líneas de colores que representan los cables que unen los pines específicos del ESP32 con los del módulo RFID. [10]

Características principales

- **Pines de E/S Abundantes:** El Arduino Mega cuenta con 54 pines digitales de entrada/salida y 16 entradas analógicas, superando las capacidades del Arduino Uno. Esto permite conectar un mayor número de dispositivos y sensores, facilitando la creación de sistemas RFID de mayor complejidad.

- **Potencia de Procesamiento Mejorada:** Equipado con un ATmega2560, el Arduino Mega cuenta con 256 KB de memoria Flash, 8 KB de SRAM y 4 KB de EEPROM. Estas

características le otorgan una capacidad de almacenamiento y procesamiento superior, ideal para gestionar diversas tareas relacionadas con RFID y el procesamiento de datos en tiempo real.

- **Interfaces de Comunicación Diversas:** Al igual que el Arduino Uno, el Mega puede conectarse a través de USB y soporta comunicación serial, I2C y SPI, pero su mayor número de puertos seriales permite una comunicación más compleja y diversa con otros dispositivos.

- **Dimensiones y Formato:** El Arduino Mega es ligeramente más grande que el Uno, lo que facilita la integración de componentes adicionales directamente en la placa sin necesidad de shields externos.

Ventajas en la Aplicación de Sistemas RFID Pasivos

- **Capacidad de Expansión Superior:** La amplia gama de pines y la mayor memoria del Arduino Mega soportan la expansión y la complejidad del sistema sin sacrificar el rendimiento, lo que es ideal para aplicaciones RFID que requieren procesamiento de datos extenso o la integración de múltiples módulos.

- **Entorno de Desarrollo Consistente:** Al mantener el uso del IDE de Arduino, el Arduino Mega beneficia de la misma facilidad de programación y la extensa biblioteca de recursos que hacen accesible incluso las tareas de programación más complejas.

- **Comunidad Robusta y Soporte:** Beneficiándose de la misma comunidad activa que el Arduino Uno, el Arduino Mega disfruta de una gran cantidad de documentación, tutoriales y ejemplos que son particularmente valiosos para proyectos avanzados.

2.3.3 Selección mejor Alternativa

La decisión entre el ESP32 y el Arduino MEGA para implementaciones RFID Pasivo se basa en factores clave como conectividad, capacidad de procesamiento, eficiencia energética,

comunidad de soporte y coste. Estos criterios son esenciales para determinar la solución más eficiente y rentable para un proyecto determinado.

	ESP32	Arduino MEGA 2560
Conectividad	Wi-Fi y Bluetooth integrado.	Requiere módulos adicionales para conectividad inalámbrica.
Procesamiento y Memoria	CPU de doble núcleo con hasta 240 MHz, 520 KB SRAM.	16 MHz, 8 KB SRAM, pero mayor cantidad de pines de E/S.
Eficiencia Energética	Diseñado para aplicaciones de bajo consumo.	Mayor consumo, adecuado para aplicaciones donde la alimentación no es una limitación.
Soporte y Comunidad	Amplio soporte para aplicaciones IoT.	Fuerte soporte para aplicaciones generales de microcontroladores.
Costo	$9.49 a $15	$25 a $35

Tabla 7.- Especificaciones Técnicas del ESP32 vs Arduino Mega.

Tabla 7

Nota: *Esta tabla proporciona una comparativa de las especificaciones técnicas más importantes del ESP32 vs Arduino Mega 2560.*

La elección del Microcontrolador ESP32 y RFID Pasivo es la mejor alternativa, fundamentada en:

- **Costo-Efectividad:** El ESP32 ofrece una solución más económica en términos de coste por unidad y por las características integradas que reduce la necesidad de componentes adicionales.

- **Capacidades Integradas de Conectividad:** La presencia de Wi-Fi y Bluetooth ofrece una ventaja inmediata en proyectos IoT frente al Arduino MEGA que requiere módulos externos para tales capacidades.
- **Rendimiento Superior:** El procesamiento de alta velocidad y la eficiencia energética del ESP32 son más adecuados para aplicaciones exigentes y eficientes en energía.
- **Soporte de la Comunidad IoT:** La comunidad en torno al ESP32 está orientada hacia aplicaciones IoT, lo que brinda una ventaja para el soporte de desarrollo y la disponibilidad de recursos.

Teniendo en cuenta los criterios de conectividad, rendimiento, eficiencia energética y coste, el microcontrolador ESP32 junto con RFID Pasivo emerge como la solución más adecuada. Su accesibilidad económica y su adecuación para aplicaciones de IoT modernas le otorgan la preferencia sobre el Arduino MEGA para la implementación de sistemas RFID en este contexto.

2.4 Diagrama de Conexiones

La interacción entre el microcontrolador y los sistemas de identificación por radiofrecuencia (RFID) es fundamental en el desarrollo de aplicaciones inteligentes y conectadas. La Figura 2.3 ofrece una representación visual de cómo el Microcontrolador ESP32 puede ser utilizado para gestionar múltiples módulos RFID RC522, lo cual es esencial para sistemas de monitoreo y control de acceso que requieren la lectura simultánea de varias etiquetas RFID. Además, en la tabla 5 “Conexiones ESP32 y RFID” se encuentra compilada la información esencial correspondiente a las conexiones y los pines que se emplean en la interfaz entre el microcontrolador ESP32 y los módulos RFID.

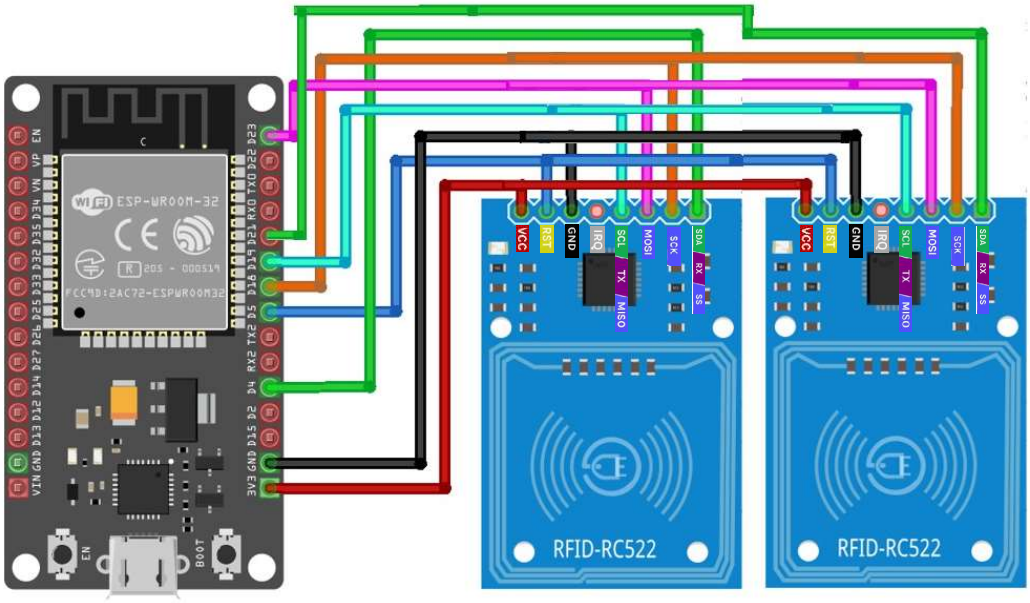


Figura 2.3.- Microcontrolador ESP32 Y RFID RC522 simulado

Nota: La imagen muestra un diagrama de conexiones simulado entre un Microcontrolador ESP32 y dos módulos lectores RFID RC522. Las conexiones están indicadas con líneas de colores que representan los cables que unen los pines específicos del ESP32 con los del módulo RFID.

Las líneas de colores en el diagrama no solo ilustran la conexión física entre los dispositivos, sino que también reflejan la sinergia entre la comunicación digital y la funcionalidad inalámbrica, aspectos cruciales para la implementación de soluciones RFID eficientes. Este diagrama simulado sirve como guía para el ensamblaje y la configuración de un sistema RFID integrado, destacando la flexibilidad y capacidad del ESP32 para complejas tareas de comunicación con precisión y eficiencia.

2.5 Funcionalidad y diagrama de flujo del sistema

El sistema de detección de asistencia optimiza el proceso de detección y registro mediante la tecnología RFID pasiva. Su diseño está orientado a facilitar la recolección y el procesamiento de datos de asistencia en eventos, proporcionando una solución automatizada y eficiente. La estructura del sistema se detalla en la figura 2.4

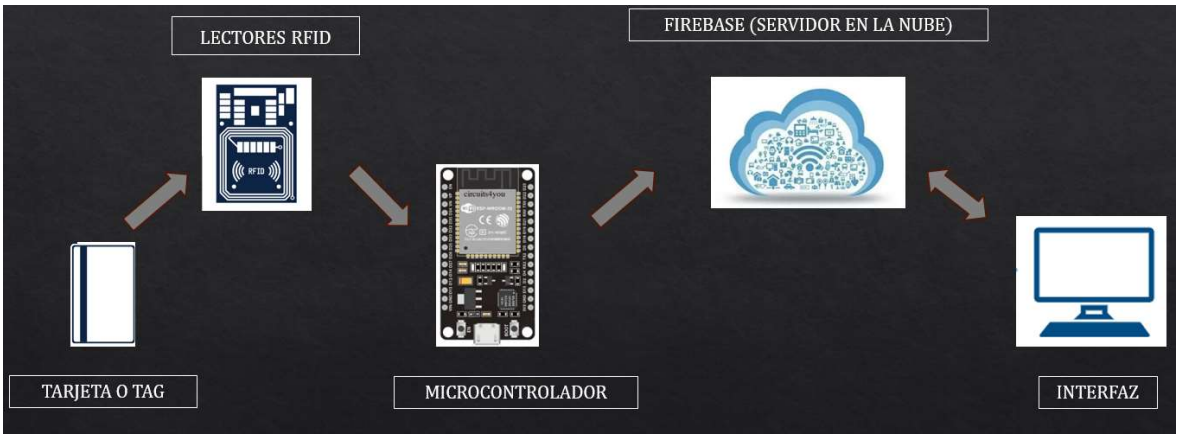


Figura 2.4.- Elementos RFID, Esquema Básico de funcionamiento.

Nota: *La imagen muestra un esquema simplificado de un sistema de detección de asistencia usando tecnología RFID y un microcontrolador, con conexión a un servidor en la nube y una interfaz de usuario.*

- **Etiquetas RFID Pasivas:** Estos dispositivos, carentes de fuente de energía propia, almacenan información específica del portador. Se distribuyen a los asistentes al iniciar el evento, funcionando como identificadores únicos para cada uno de ellos.

- **Lectores RFID (Integración de ESP32 con módulo RC522):** Ubicados estratégicamente en puntos de control clave del evento, estos lectores tienen la función de detectar las etiquetas RFID pasivas. Cada lector incluye un microcontrolador ESP32, responsable del procesamiento de los datos obtenidos, y un módulo RC522, dedicado a la detección de señales provenientes de las etiquetas RFID.

- **Procesador Central (Servidor):** Una vez que el ESP32 captura los datos, los envía a un servidor central o a una base de datos. Este proceso es fundamental para la recolección y almacenamiento de información de asistencia.

- **Base de Datos/Servidor de Aplicaciones:** Este componente actúa como el repositorio central de toda la información recabada por los lectores RFID. Aquí se procesan los datos, permitiendo registrar aspectos importantes como la hora de entrada y salida de los asistentes, la duración de su estancia y la generación de informes detallados de asistencia.

- **Interfaz de Usuario/Administración del Evento:** A través de una interfaz gráfica intuitiva, los organizadores del evento tienen acceso a la información procesada. Esta interfaz permite la visualización de datos en tiempo real, la generación de informes de asistencia y la realización de análisis detallados tras la conclusión del evento.

El flujo de datos en este sistema es directo y eficiente. Los lectores RFID capturan la información de las etiquetas a medida que los asistentes pasan por los puntos de control. Esta información se envía inmediatamente al servidor, donde se procesa de manera centralizada, optimizando la gestión y el análisis de la asistencia en el evento.

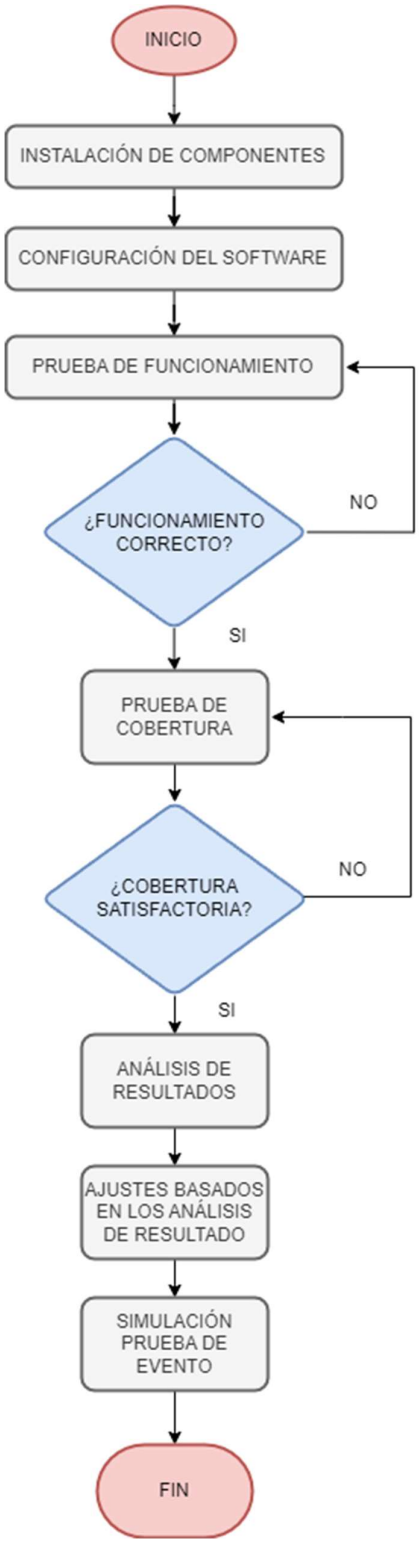


Figura 2.5.- Diagrama de Flujo para implementación de RFID

Nota: La imagen muestra un diagrama de flujo detallado para la implementación y prueba de un sistema, de detección de asistencia utilizando tecnología RFID.

- **Iniciación del Proceso**

  Se inicia con la configuración inicial del sistema de control de asistencia RFID.

- **Instalación de Componentes Hardware**

  Colocación estratégica de lectores RFID, utilizando el ESP32 integrado con el módulo RC522, en los puntos de acceso relevantes del evento.

- **Configuración y Programación del Software**

  Programación y calibración del ESP32 para la lectura eficiente de las etiquetas RFID y la transmisión segura de los datos hacia el servidor de la base de datos.

  Preparación y estructuración de la base de datos para recibir, almacenar y procesar adecuadamente los datos de asistencia.

- **Evaluación de Funcionalidad del Sistema**

  Realización de pruebas exhaustivas para confirmar que los lectores RFID identifican correctamente las etiquetas pasivas y comunican los datos de forma precisa al servidor.

- **Verificación del Funcionamiento Adecuado**

  En caso de detectar anomalías, se procede a la recalibración de los lectores RFID, así como a la reconfiguración del software, seguido de la repetición de la evaluación de funcionalidad.

  Si el sistema funciona adecuadamente, se procede a la fase siguiente.

- **Test de Cobertura de Lectores**

  Inspección para asegurarse de que la cobertura de los lectores RFID abarca todos los puntos de acceso designados sin zonas muertas o de señal débil.

- **Confirmación de Cobertura Óptima**

  Si la cobertura es insuficiente, se realiza la reubicación y ajuste de los lectores RFID, seguido de una nueva prueba de cobertura.

  Si la cobertura es la adecuada, se avanza a la etapa de análisis de resultados.

- **Análisis de Resultados Preliminares**

  Revisión de los datos recolectados para comprobar su precisión y la integridad del sistema.

- **Optimización Basada en el Análisis**

  Implementación de mejoras en la configuración del sistema, fundamentadas en las conclusiones del análisis previo.

- **Simulación en Evento Piloto**

  Despliegue del sistema en un entorno controlado que simule un evento real para validar su funcionalidad integral y la experiencia del usuario.

- **Conclusión y Puesta a Punto Final**

  Con la validación exitosa del sistema en la simulación, se concluye el proceso de implementación, quedando el sistema listo para su aplicación en eventos.

2.6 Implementación del Sistema

La Figura 2.6 muestra un esquemático de prototipo para un sistema de detección de asistencia, implementado con un microcontrolador ESP32 y dos módulos RFID RC522. Esta configuración está diseñada para registrar la entrada y salida de eventos, aprovechando la capacidad del ESP32 para manejar múltiples tareas y comunicarse con sistemas de almacenamiento de datos.

Los módulos RFID RC522, conectados a través de un arreglo de cables multicolores, son los encargados de leer las señales de las etiquetas RFID pasivas, que representan a los participantes del evento. El conjunto se monta sobre una placa de pruebas (breadboard), lo que facilita la modificación y el ajuste del circuito durante la fase de desarrollo y prueba.

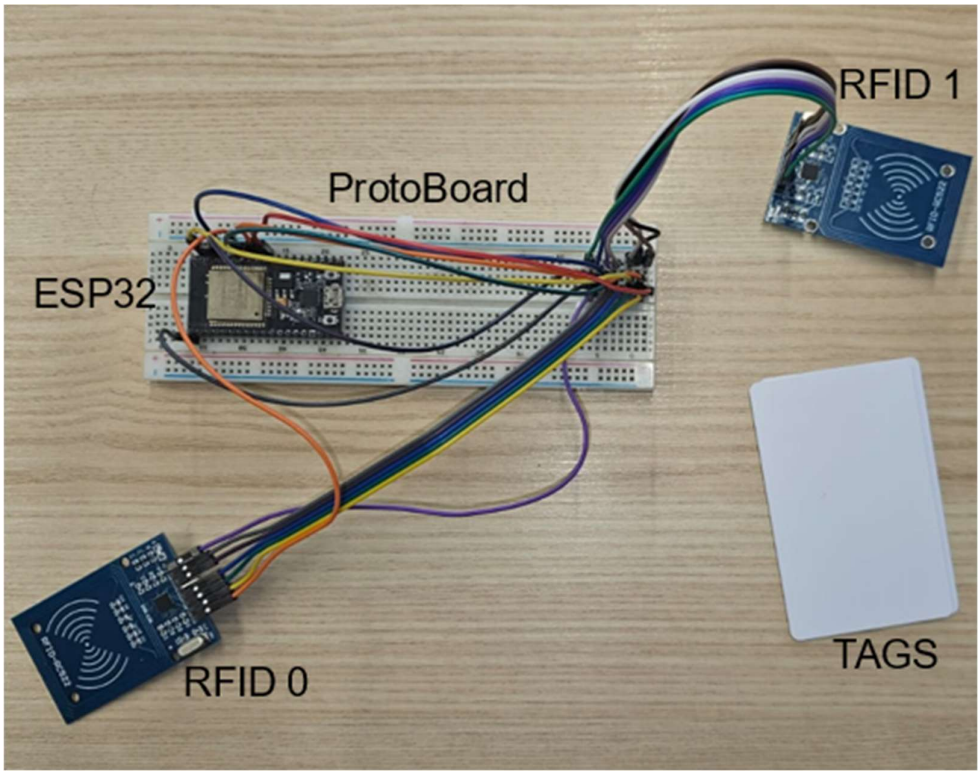


Figura 2.6.- Microcontrolador ESP32 Y dos RFID RC522 implementado.

Nota: Nota: La imagen muestra un diagrama de conexiones implementado entre un Microcontrolador ESP32 y dos módulos lectores RFID RC522. Las conexiones están indicadas con líneas de colores que representan los cables que unen los pines específicos del ESP32 con los del módulo RFID.

2.7 Diagrama de flujo de validaciones y codificación.

El diagrama de flujo que veremos a continuación muestra las distintas validaciones que se realizaron a través de codificación de en el lenguaje Arduino y Python.

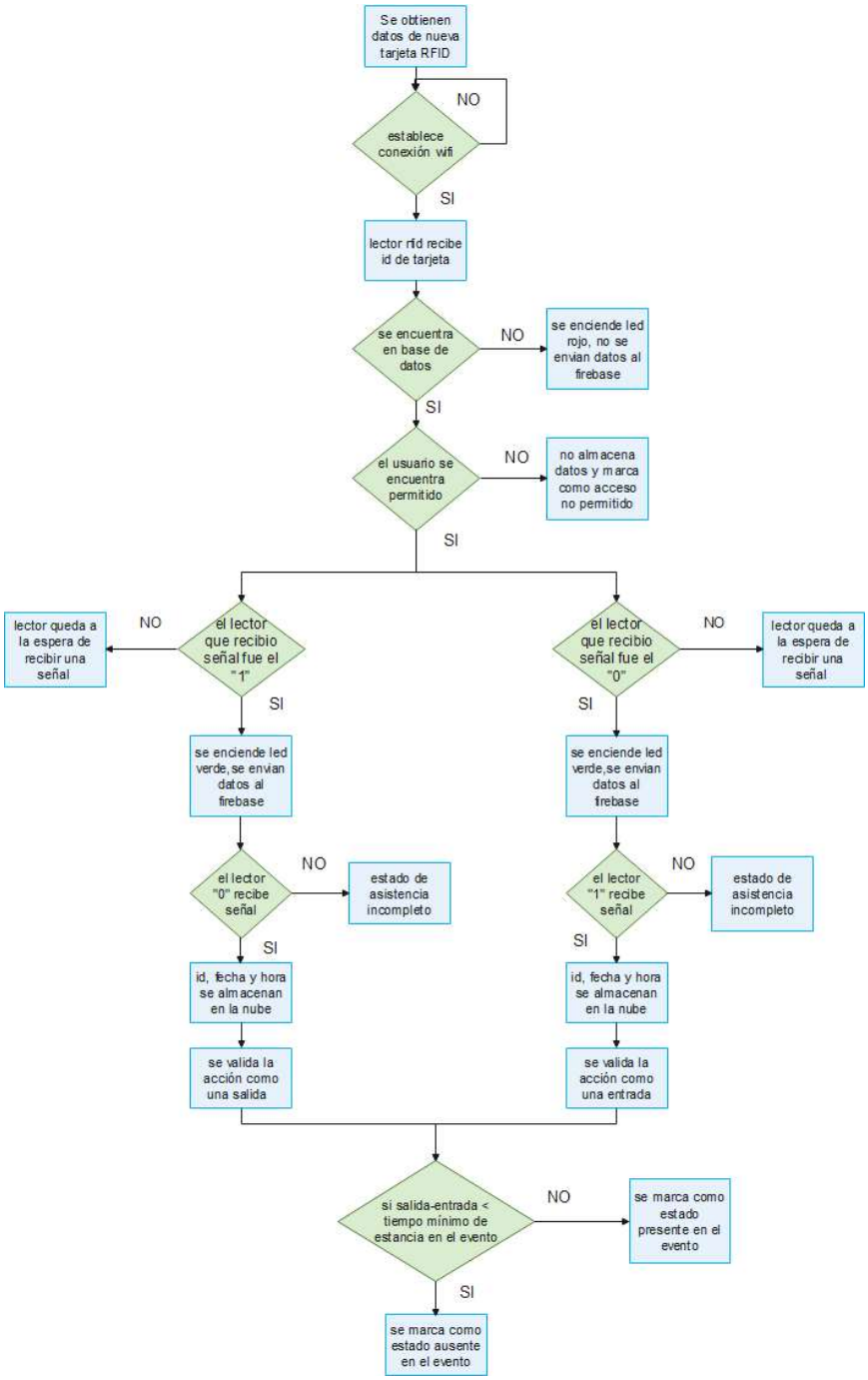


Figura 2.7.- Diagrama de flujo de codificación.

- **Iniciación del sistema**

  El ESP32, valida la conexión a una red WIFI, la cual se encuentra preconfigurada y fijada en el ESP32 a través del lenguaje ARDUINO; esta red es configurable dependiendo del lugar en donde se encuentre el dispositivo.

- **Lector RFID recibe la señal**

  La primera validación de la data obtenida por el lector es que si esos datos se encuentran en el FIREBASE (base de datos en la nube), se permite que la data sea enviada a la base de datos y de igual forma a la interfaz final del cliente.

- **Validación de usuario permitido**

  La segunda validación depende del tipo de acceso que tiene el usuario en la base de datos. Puede ocurrir que el usuario se encuentre en la base de datos pero con un acceso restringido al evento.

- **Validación de tipo de marcación**

  La segunda validación es el tipo de dato obtenido, es decir si la señal pasa primero por el lector 0 y luego por el lector 1 el sistema guarda esos datos como una entrada, caso contrario marca como una salida.

- **Validación de tiempo de estancia**

  Por último, el sistema establece si la estancia de la persona es lo suficiente para que sea marcada como presente o ausente en el evento. El valor mínimo de estancia es configurado a consideración del cliente.

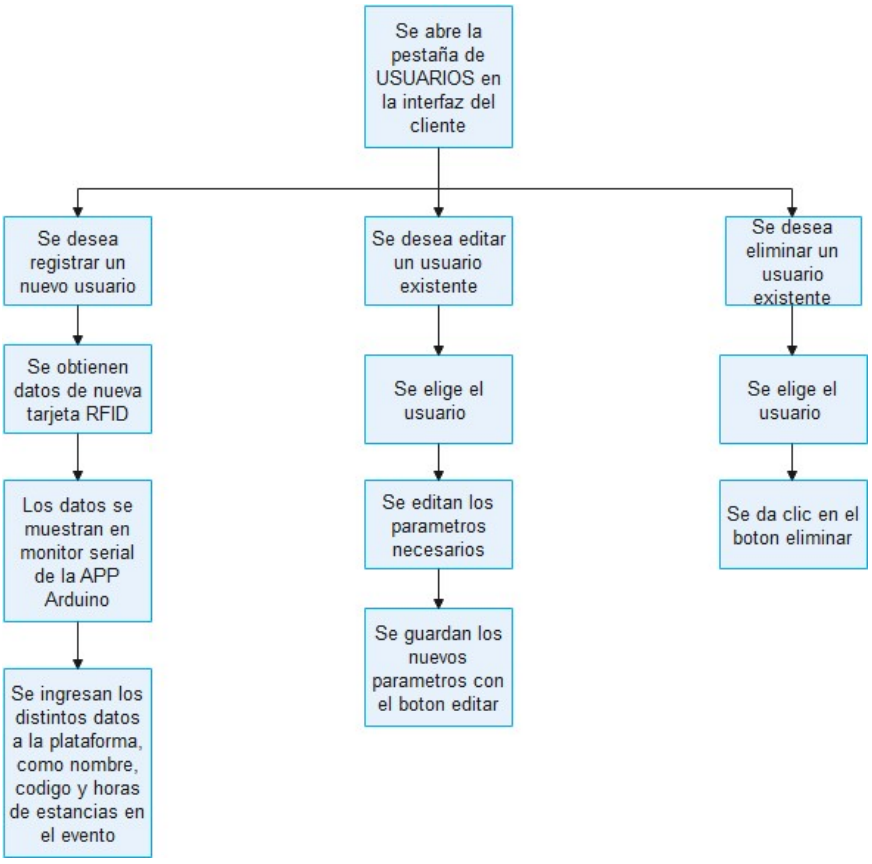


Figura 2.8.- Diagrama de flujo, sección "Usuarios"

- **Validación de viñeta.**

  Se realiza la respectiva validación de la viñeta en la que se encuentra el cliente que debe de ser la de USUARIOS.

- **Opciones**

  Opción 1: Agregación de nuevo usuario, para realizar esta acción es necesario obtener el ID de la nueva tarjeta a través de programa ARDUINO y colocarlo junto a los nuevos datos de usuario como lo son nombre y horas de estancias durante el evento.

  Opción 2: Edición de usuario. Con esta acción se elige el usuario que se desea editar y cambiar los distintos parámetros como sea necesario; para que la información quede guardada se da clic en botón editas

Opción 3: Por último, existe la opción de eliminar un usuario elegido por el cliente con un clic en eliminar, con lo cual se borra de la base de datos.

En la figura 2.9 se puede observar las tres secciones o viñetas que contiene la interfaz del sistema RFID detección de asistencias, cada una de estas respectivas secciones se las analizará en el siguiente capítulo, realizando pruebas de funcionamiento.

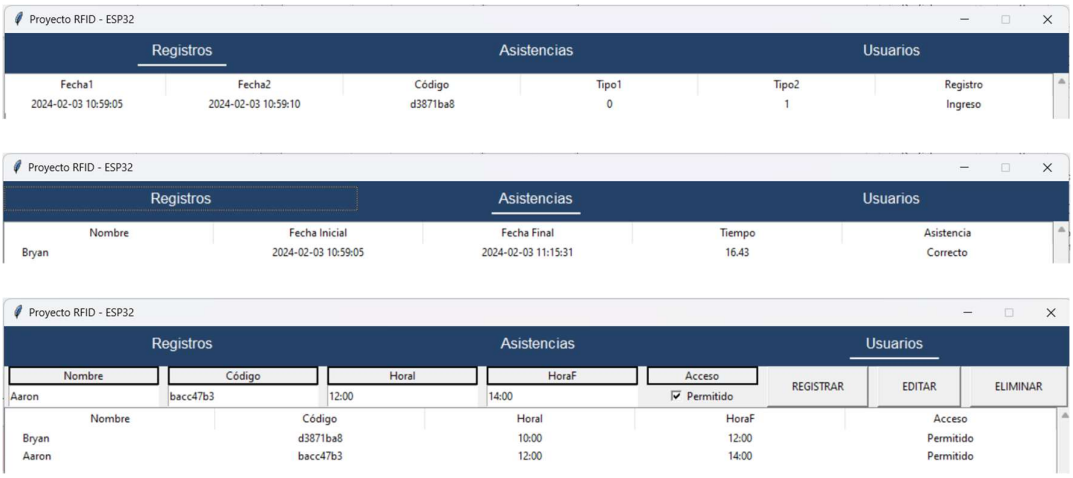


Figura 2.9.- Interfaces del sistema.

Capítulo 3

3. RESULTADOS Y ANÁLISIS

Este capítulo se analizará los resultados obtenidos tras la implementación del sistema de detección de asistencia basado en RFID pasiva. Los datos recopilados durante las diversas fases de prueba proporcionan una visión crítica sobre el sistema propuesto.

La importancia de este análisis radica en su habilidad para confirmar el correcto funcionamiento y examinar si se han cumplido las metas del proyecto. Lo que sigue es una descripción de un proceso iterativo de ensayos y modificaciones, llevado a cabo para lograr un rendimiento óptimo.

En las siguientes secciones, se desglosarán los componentes del análisis, presentando los resultados obtenidos y su correspondiente interpretación, para finalmente ofrecer una evaluación objetiva del sistema y su impacto potencial en la industria de la gestión de eventos.

3.1 Análisis de Interfaces

En esta sección, se llevará a cabo un análisis exhaustivo de cada una de las interfaces del sistema de detección de asistencia a eventos, explorando en profundidad las características, funcionalidades y el diseño de la interfaz de usuario. Se examinarán detalladamente aspectos como la navegabilidad, las opciones de menú y la eficacia de los sistemas de registro y detección de asistencia. Este análisis detallado tiene como objetivo no solo destacar las capacidades actuales del sistema, sino también identificar oportunidades de mejora y potenciales innovaciones que puedan ser integradas en futuras versiones del software, buscando siempre la excelencia en la gestión de eventos y una experiencia de usuario superior.

3.1.1 Análisis Sección Usuario

La figura 3.1 ilustra la interfaz principal del software, diseñada para administrar los usuarios existentes dentro de un evento. Cada pestaña cumple una funcionalidad distinta del sistema, proporcionando una experiencia de usuario cohesiva y organizada, esencial para la

detección de asistencia y la gestión de los participantes en los eventos. A continuación, se presentan más detalles sobre la interfaz, pestaña “Usuarios”.

La interfaz presenta un diseño sencillo y minimalista, con un esquema de colores neutros. Los elementos están claramente distribuidos, con separación entre las listas de usuarios y los botones de acción, lo cual facilita la navegación. Por otro lado, la interfaz está segmentada en tres secciones principales: Registros, Asistencias y Usuarios, indicando diferentes funcionalidades del sistema. Se muestra una tabla con las columnas "Nombre", "Código" y "Acceso".

El campo "Nombre" está destinado a identificar al usuario o participante del evento, permitiendo un registro manual directamente en la interfaz de la aplicación sin requerir intervención directa en la base de datos de Firebase ni en el código fuente dentro del IDE de Arduino, simplificando así el proceso de entrada de datos y manteniendo la gestión centrada en el usuario.

El campo "Código" se asigna al identificador único asociado con la etiqueta RFID de cada usuario. Este código se puede adquirir previamente al pasar la tarjeta por el lector RFID y es visible directamente desde el entorno de desarrollo integrado (IDE) de Arduino, facilitando así su registro y seguimiento en el sistema.

La columna "Acceso" indica si al usuario se le ha otorgado permiso de acceso, lo cual puede ser útil para controlar y restringir la entrada o salida. Además, en la interfaz se puede observar que se tiene los campos HoraI (hora inicial) y HoraF (hora final), los cuales permites controlar el intervalo de tiempo en la cual es permitido el ingreso del usuario.

Los botones "REGISTRAR", "EDITAR" y "ELIMINAR" ofrece funcionalidades claras y directas para la gestión de usuarios. REGISTRAR se usa para añadir nuevos usuarios al sistema. La opción "EDITAR" permite modificar la información de usuarios existentes y por último la opción, "ELIMINAR" sería para remover usuarios del sistema.

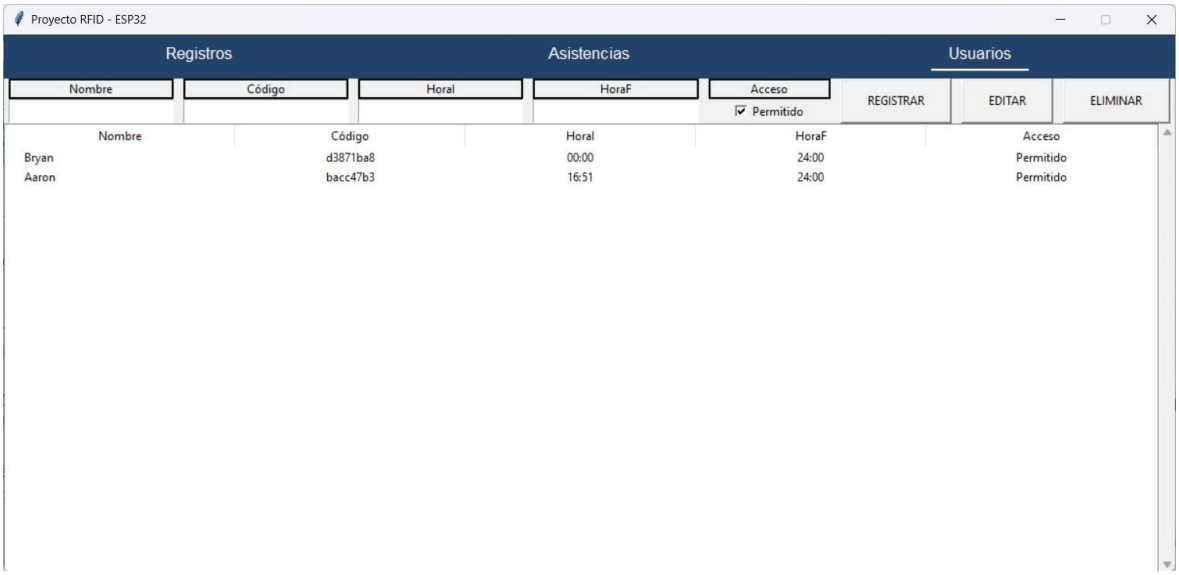


Figura 3.1.- Interfaz del proyecto, Sección “Usuarios.”

3.1.2 Análisis Sección Registro

La figura 3.2 muestra la sección "Registros" de la interfaz del software de gestión de asistencia RFID. Esta pantalla es fundamental para el seguimiento de la actividad de los usuarios dentro de un evento. Aquí se detallan las entradas y salidas de los participantes, registradas automáticamente por el sistema. El análisis de los elementos en la interfaz se presenta a continuación:

El campo “Fecha” muestra la fecha y la hora exacta de cada evento registrado, ofreciendo un registro del tiempo detallado de las acciones del usuario.

El campo “Nombre” corresponde al nombre del usuario o participante del evento, lo cual sugiere que el sistema permite rastrear la actividad individual.

El campo “Tipo” indica la naturaleza del registro, ya sea "Ingreso" al evento o "Salida" del mismo, proporcionando una información para validar la asistencia.

La sección "Registros" de la aplicación proporciona una visión completa y en tiempo real de la actividad de los usuarios en un evento, facilitando la gestión y supervisión de la asistencia y contribuyendo a la seguridad y al análisis operativo del evento. Los registros de la interfaz se generan de manera automática y confiable a través del sistema cada vez que una etiqueta RFID es detectada por los lectores del evento. Este proceso automatizado garantiza una colección de datos precisa y eficiente, eliminando las inexactitudes que suelen acompañar la entrada manual de datos. Al escanear la etiqueta, el sistema captura de inmediato la fecha y hora específicas del evento, registrando meticulosamente cada instancia en la que un usuario cruza un punto de control. Esta precisión temporal es fundamental para analizar el flujo de asistencia y para monitorear la duración de las visitas de cada participante.

Además, la integración de la tecnología RFID con el sistema permite una identificación sin errores del código único de cada usuario, asociado directamente con su etiqueta personal. La capacidad de correlacionar sin fisuras cada interacción con un perfil de usuario específico refuerza tanto la seguridad como la funcionalidad operativa del evento. Los organizadores pueden así revisar el comportamiento de las asistencias en tiempo real o realizar análisis retrospectivos, lo que es esencial para optimizar la gestión de eventos futuros y para responder de manera proactiva ante cualquier incidencia de seguridad o asistencia.

Este sistema de registro automático también proporciona una base de datos robusta para informes detallados después del evento. Puede utilizarse para validar la conformidad con las normativas de seguridad, evaluar la efectividad de la logística del evento y proporcionar observaciones valiosas para la planificación estratégica y la toma de decisiones basadas en datos. En definitiva, la automatización del registro de asistencias a través de la tecnología RFID aporta un valor incalculable al ofrecer un método de seguimiento exhaustivo, confiable y libre de errores, crucial para la administración avanzada de cualquier evento.

Figura 3.2.- Interfaz del proyecto, Sección “Registros”

3.1.3 Análisis Sección Asistencia

La figura 3.3 y 3.4 proporciona una vista de la sección "Asistencias" de un sistema de gestión basado en RFID, orientado a capturar y analizar la participación de los usuarios en un evento. El campo “Nombre” se muestra el nombre del participante, esto sugiere que el sistema es capaz de rastrear a los usuarios de manera individual a lo largo de diferentes puntos en el tiempo. Además, el campo “Fecha Inicial” y "Fecha Final" registra el momento exacto en donde el participante ha ingresado al evento o salido respectivamente, proporcionando una marca de tiempo precisa que es vital para la gestión de la asistencia.

El campo “Tiempo” indica la cantidad de tiempo que el participante permaneció en el evento. También se tiene el campo "Asistencia", muestra si la asistencia fue "Correcta" o "Incorrecta". Este estado es configurable y puede variar dependiendo de las reglas o parámetros establecidos para el evento. Por ejemplo, puede referirse a si el usuario cumplió con un tiempo mínimo requerido de permanencia en el evento o si completó todas las etapas necesarias en un proceso de Entrada/Salida.

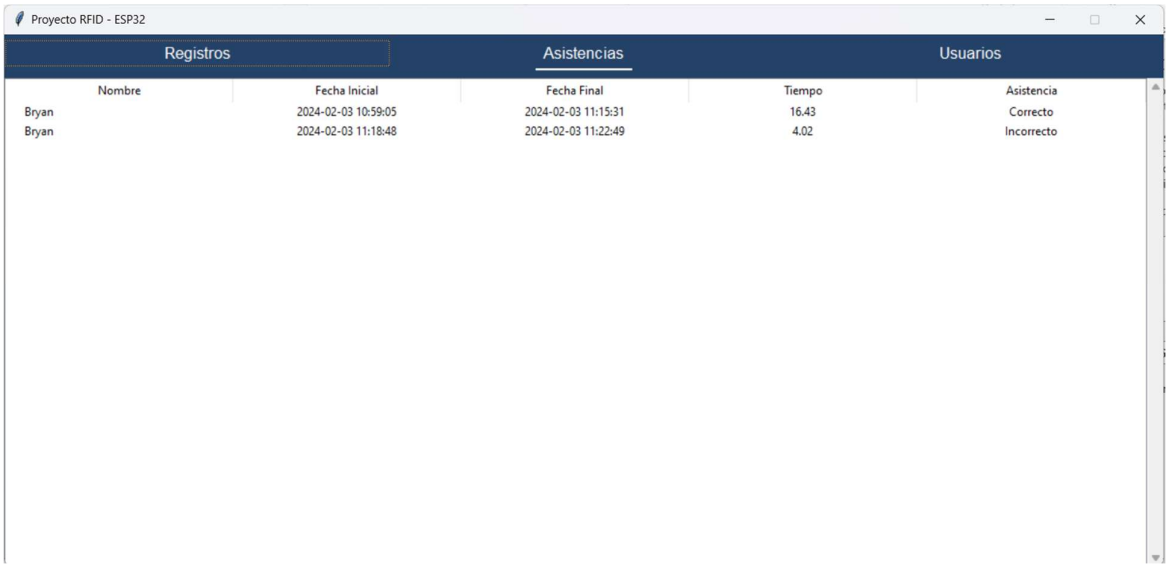


Figura 3.3.- Interfaz del proyecto, Sección "Asistencias"

3.2 Análisis de FireBase y Arduino

La implementación de Firebase Realtime Database ha demostrado ser una herramienta esencial en la gestión y almacenamiento en tiempo real de los datos de asistencia recopilados a través de la tecnología RFID. Al utilizar esta base de datos, hemos podido observar una actualización instantánea y sincronizada de la información cada vez que un participante ingresa o sale del evento, lo cual se refleja en la interfaz de usuario diseñada para la administración de registros. Cada etiqueta RFID asociada a los participantes tiene un código único que, al ser detectado por el lector RFID conectado al ESP32, transmite los datos pertinentes a la base de datos. Este sistema ha permitido un monitoreo constante de las asistencias, así como la capacidad de realizar seguimientos precisos del flujo de personas, proporcionando una interfaz intuitiva y eficiente para la administración de los eventos. En la figura 3.5 se puede observar la eficacia del sistema en la captura y visualización de datos en tiempo real, destacando su fiabilidad y su potencial para adaptarse a diferentes escenarios y tamaños de eventos. La estructura de la base de datos muestra un diseño coherente y normalizado, lo que facilita la escalabilidad del proyecto y la integración con otras herramientas y servicios en el futuro.

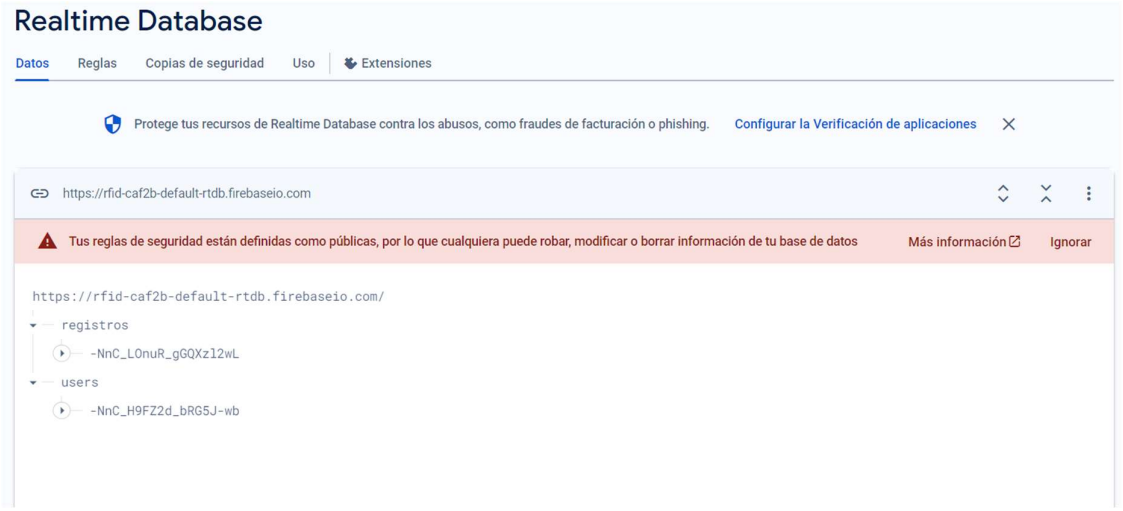


Figura 3.4.- Base de Datos Registros y Usuarios.

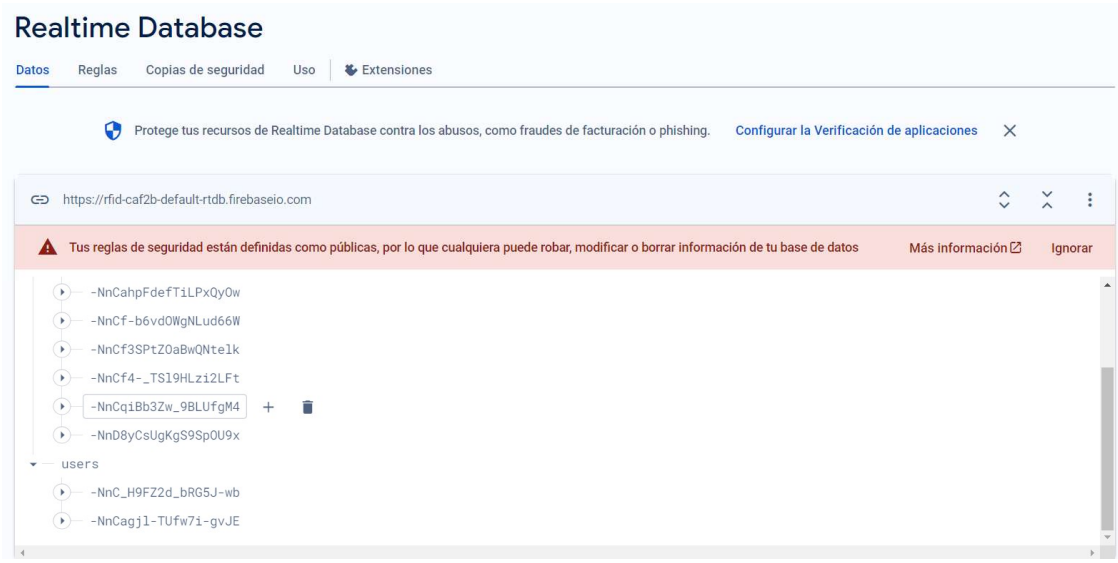


Figura 3.5.- Base de datos de FireBase.

La salida del monitor serial del IDE de Arduino nos proporciona una visión detallada del rendimiento del sistema al interactuar con la tecnología RFID. Al presentar una tarjeta RFID, identificada por un código único, el sistema inicia una consulta en Firebase Realtime Database bajo la ruta especificada "/users/". En esta instancia, el código "bacc47b3" fue reconocido exitosamente como el usuario "Aaron", quien cuenta con el acceso autorizado, como lo indica la propiedad "isAllowed": true en la base de datos. Esta interacción desencadena una actualización

en tiempo real en la ruta "/registros/", donde se documenta la asistencia del usuario. La confirmación "ACCESO PERMITIDOPASSED" es un claro testimonio de la precisión y eficiencia del sistema, asegurando que solo los usuarios autorizados sean registrados, lo cual es crítico para la gestión segura y efectiva de la asistencia a eventos. Estos resultados demuestran la robustez del sistema y su capacidad para manejar la autenticación y el registro de asistencia en tiempo real. En la figura 3.6 se muestra el monitor serial de la interfaz de Arduino al pasar el Tags.

```
Código de la tarjeta (string): leyendo rfid 1
bacc47b3
OBTENIENDO DATOS
Data from Firebase: /users/
{"-NnC_H9FZ2d_bRG5J-wb":{"codigo":"bacc47b3","isAllowed":true,"nombre":"Aaron"}}
Datos del JSON:
ACCESO PERMITIDOPASSED
PATH: /registros/
TYPE:
SE ENCONTRARON COINCIDENCIAS
```

Figura 3.6.- Monitor Serial del IDE de Arduino.

3.3 Análisis y pruebas de funcionamiento

Esta sección se enfoca en el análisis y las pruebas de funcionamiento de la solución propuesta para la "Detección de Asistencia a Eventos mediante Tecnología RFID".

En la figura 3.7, se puede observar el prototipo final el cual está compuesto por dos módulos RFID RC522, un ESP32 como unidad de control principal, Tags, LEDs de indicadores para el acceso, resistencias, un protoboard para facilitar la conexión de componentes y Firebase como plataforma de almacenamiento de datos en la nube. La elección de Firebase como plataforma de almacenamiento en la nube garantiza la accesibilidad y la disponibilidad de los datos recopilados.

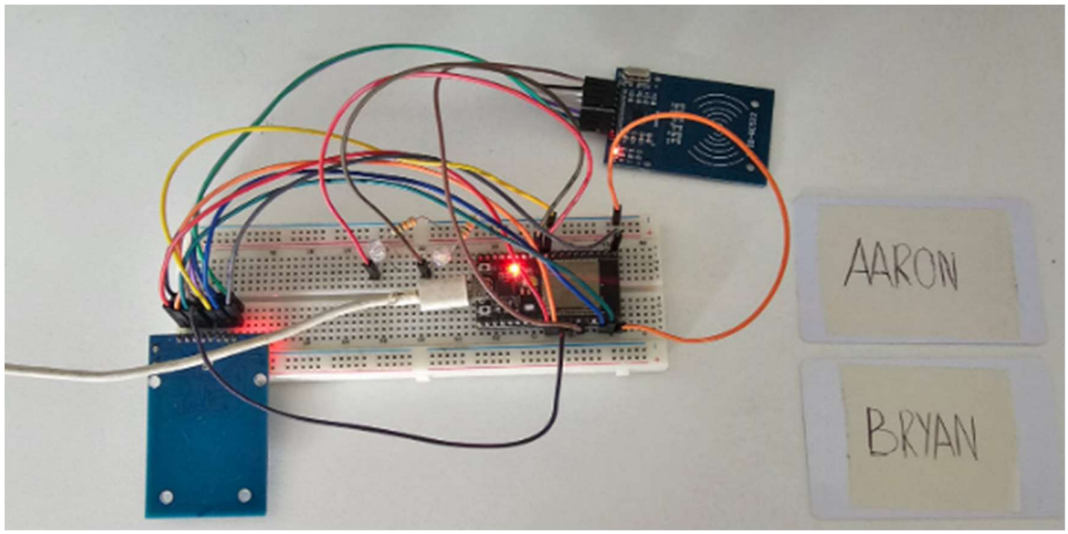


Figura 3.7.- Prototipo Final “Detección de asistencia a eventos”

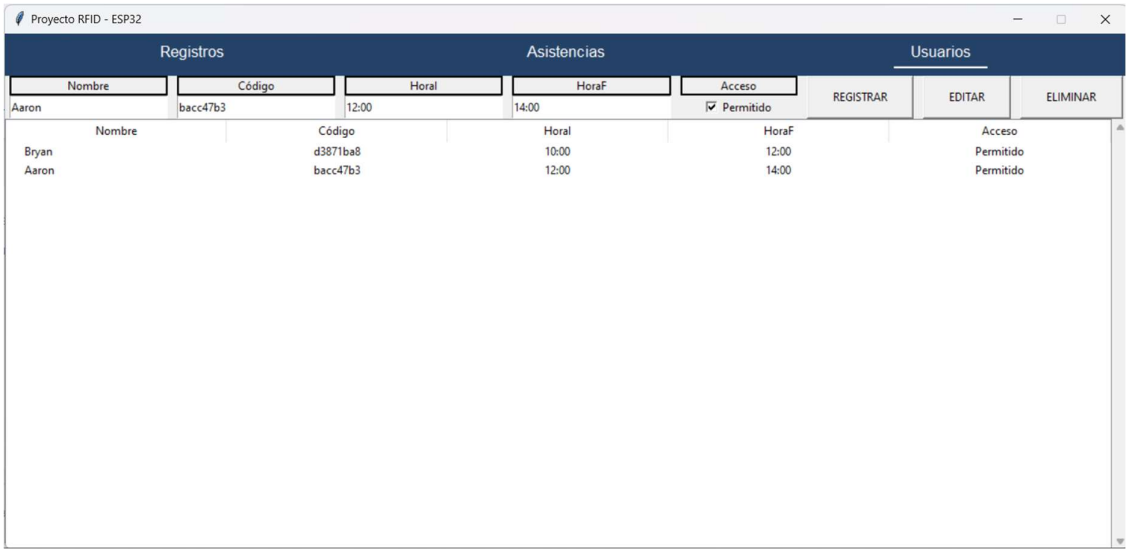


Figura 3.8.- Interfaz sección “Usuarios”

En la Figura 3.8, se presenta la sección "Usuarios" de la interfaz, la cual exhibe dos registros vigentes. El primer registro corresponde al usuario de nombre "Bryan", identificado con su respectivo Código de Identificación extraído del tag mediante el entorno de desarrollo Arduino. Este usuario tiene un intervalo horario autorizado para ingresar al evento que abarca desde las 10:00 hasta las 12:00. Por otro lado, se destaca la presencia del usuario denominado "Aaron", cuyo código único del tag también se encuentra registrado. No obstante, la ventana de acceso para este

usuario se ha establecido desde las 12:00 hasta las 14:00. Es importante subrayar que tanto la hora de inicio como la de finalización permitidas pueden ajustarse fácilmente dentro de la misma interfaz, mediante los campos "HoraI" (hora inicial) y "HoraF" (hora final). Estas configuraciones son adaptables a las exigencias específicas del evento o a los horarios que se requieran validar para la entrada.

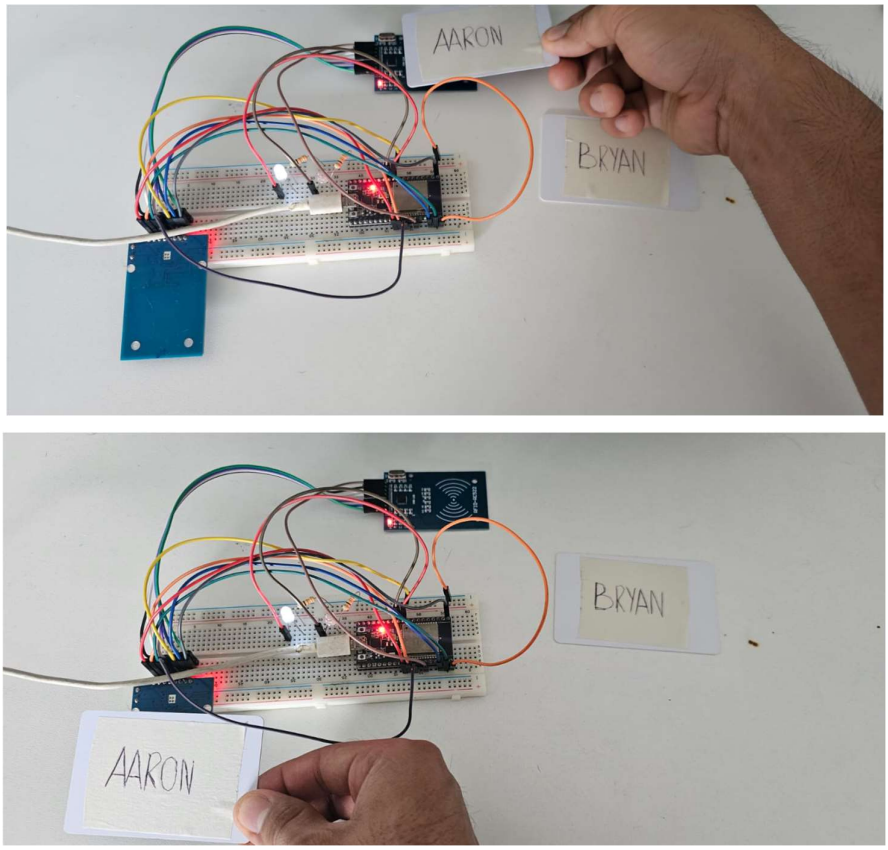


Figura 3.9.- Validación del Usuario “Aaron”.

Figura 3.10.- Serial Monitor, Validación del Usuario “Aaron”

En la Figura 3.9, se evidencia el intento de registro por parte del usuario "Aaron". No obstante, dicho intento no es permitido ya que el acceso se le ha denegado en virtud de que el horario autorizado para ingresar comprende desde las 12:00 hasta las 14:00. La Figura 3.10 presenta el Serial Monitor donde las líneas de código señalan que el acceso para el usuario "Aaron" no está permitido. Este feedback del sistema mediante el Serial Monitor proporciona una herramienta efectiva para la identificación y comunicación inmediata de situaciones en las que la entrada no cumple con los parámetros establecidos.

Se puede observar en las pruebas la efectividad del mecanismo de validación de horarios, ya que se evidencia con claridad que el usuario "Aaron" ha sido restringido de manera exitosa durante el intento de registro. La implementación de las restricciones horarias ha demostrado su funcionalidad al denegar el acceso al usuario en un período no permitido, según las configuraciones establecidas en el sistema. Este resultado respalda la eficacia del sistema en la gestión y aplicación de políticas horarias para la validación de la entrada, confirmando así el correcto funcionamiento de la validación de horarios en el proyecto.

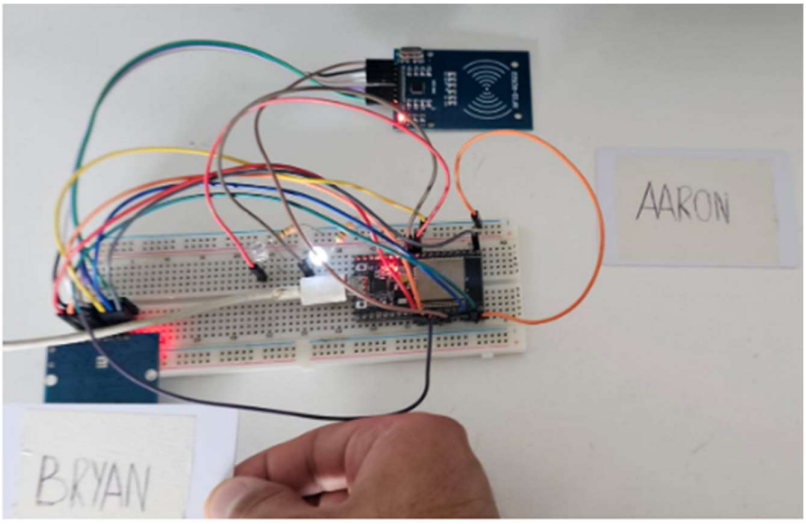


Figura 3.11.- Validación de usuario “Bryan”. Lector RFID 0

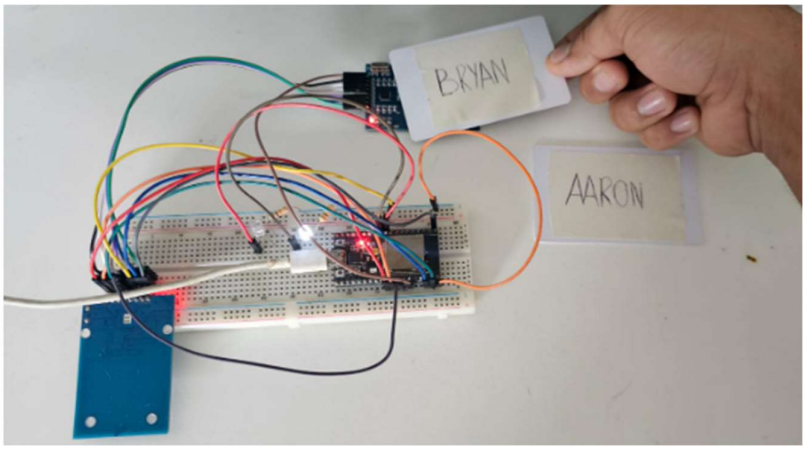


Figura 3.12.-Validación de usuario “Bryan”. Lector RFID 1

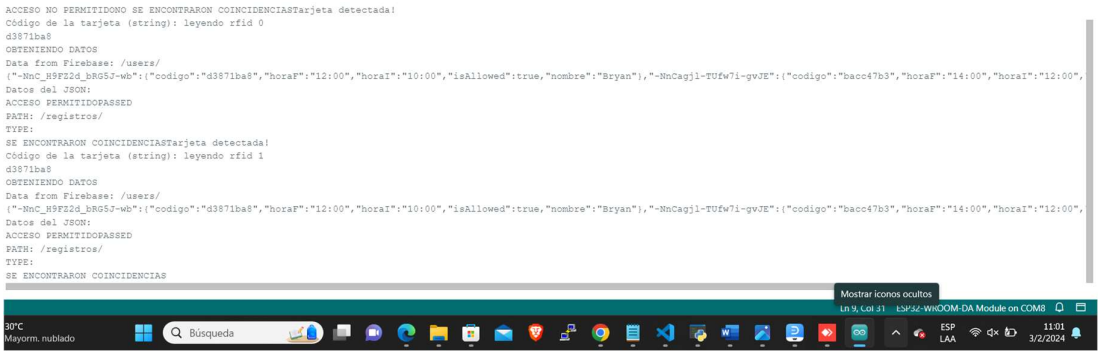


Figura 3.13.- Serial Monitor, Validación del Usuario “Aaron”

En las Figuras 3.11 y 3.12, se observa el proceso en el que el usuario "Bryan" pasa primero por el RFID 0 y posteriormente por el RFID 1, indicando claramente que ha realizado un ingreso

al evento. La confirmación de este acceso se refleja en la Figura 3.13, donde el Serial Monitor registra que el usuario atraviesa inicialmente el RFID 0 y luego el RFID 1, permitiendo así su acceso al evento. Este flujo de datos corroborado en el monitor serial confirma que el sistema ha validado exitosamente el ingreso de "Bryan" conforme al horario configurado, que en este caso fue establecido entre las 10:00 y las 12:00.

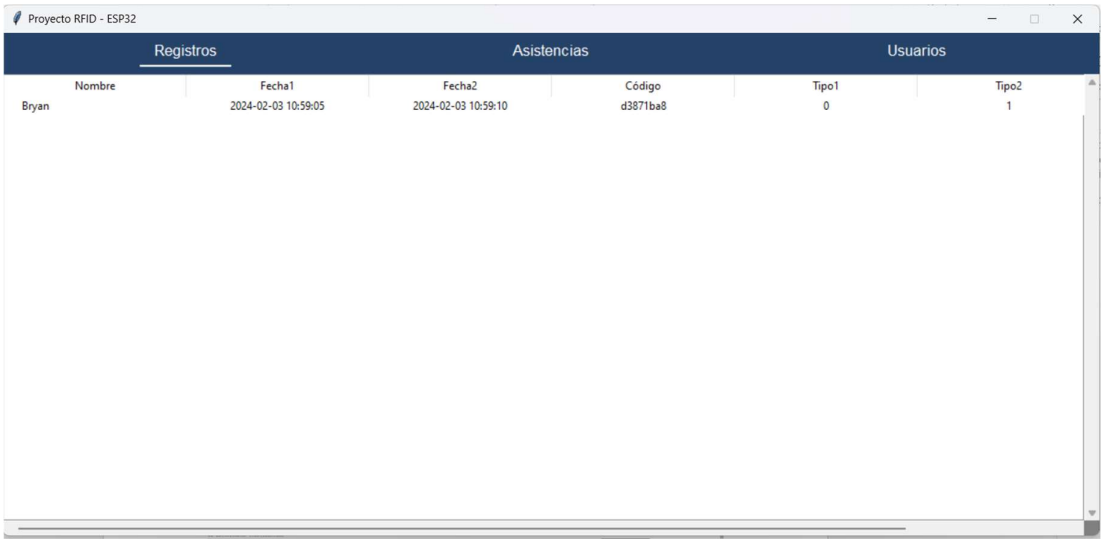


Figura 3.14.- Interfaz Sección “Registros”

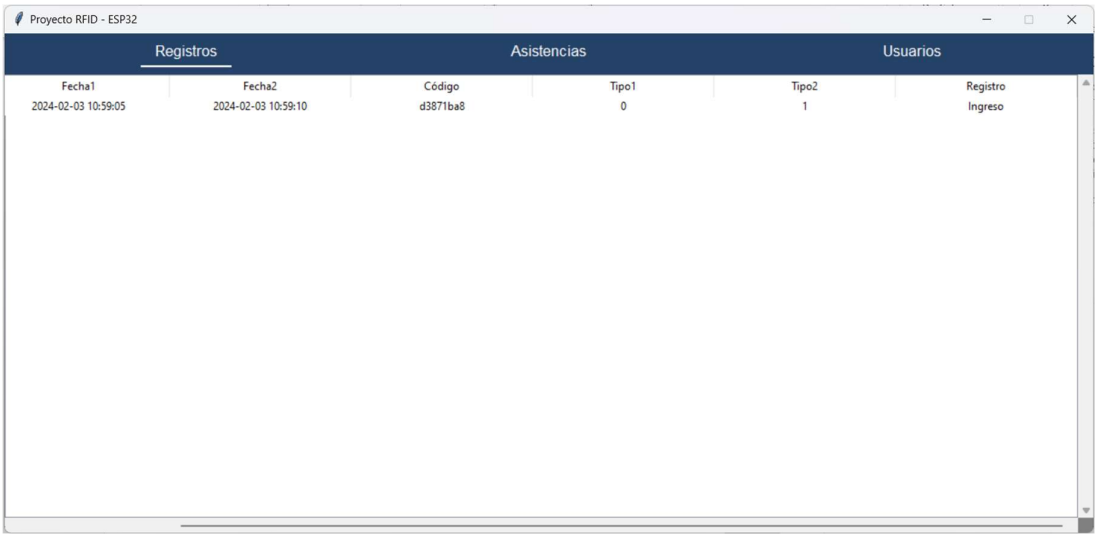


Figura 3.15.- Interfaz Sección “Registros”

En las Figuras 3.14 y 3.15, se valida que únicamente el usuario "Bryan" se encuentra registrado en la base de datos de la sección "Registros". El usuario "Aaron", al no haber tenido acceso permitido, no ha sido considerado ni registrado en dicha sección, demostrando así la efectividad del control de acceso dentro de los horarios establecidos.

La Figura 3.14 muestra los campos "tipo 1" y "tipo 2", correspondientes al RFID 0 y RFID 1, respectivamente. El hecho de que el usuario primero haya pasado por el RFID 0 y luego por el RFID 1 indica claramente que se trata de un ingreso como se pude observar en la figura 3.15. Por el contrario, en el caso de una salida, el orden sería inverso. Esta estructura de registro y clasificación de eventos confirma la capacidad del sistema para distinguir entre ingresos y salidas, proporcionando así una gestión precisa de la asistencia a eventos de acuerdo con los parámetros establecidos.

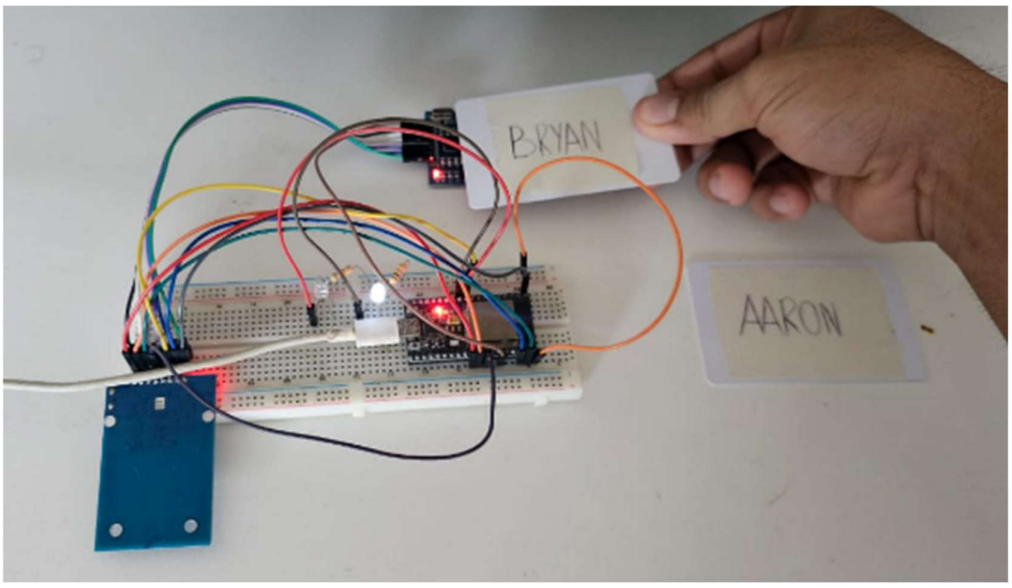


Figura 3.16.- Validación de usuario “Bryan”. Lector RFID 1

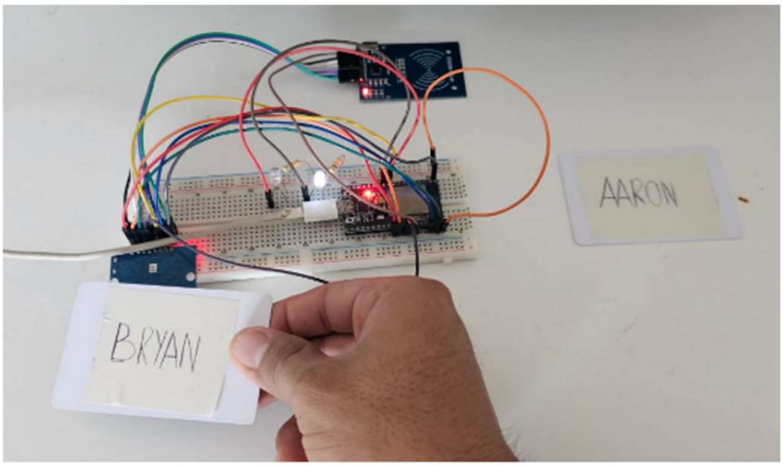


Figura 3.17.- Validación de usuario “Bryan”. Lector RFID 0

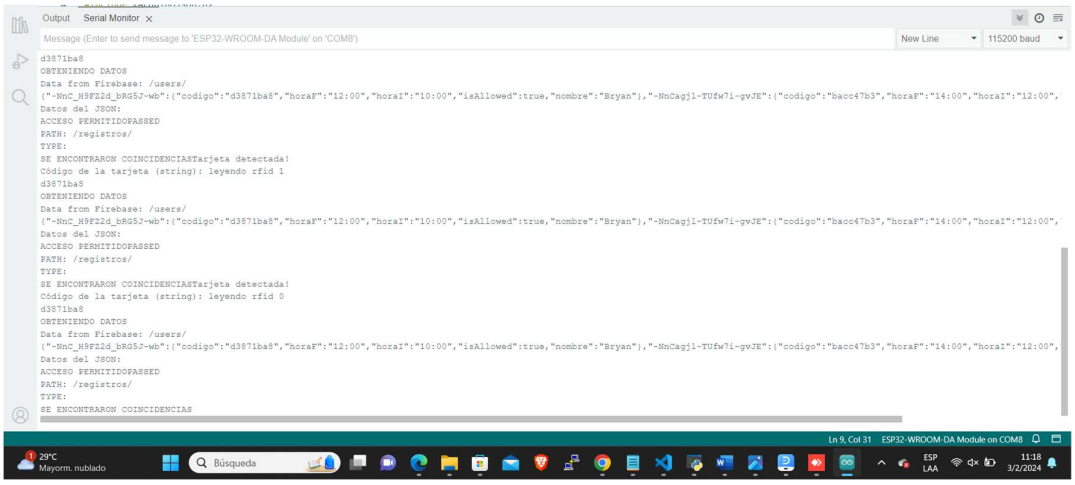


Figura 3.18.- Serial Monitor, Validación del Usuario “Bryan”

Figura 3.19.- Validación de tipo de registro y Asistencia.

En las Figuras 3.16 y 3.17, se observa que el usuario "Bryan" atraviesa primero el RFID 1 y luego el RFID 0 como se confirma en el monitor serial de la Figura 3.18, donde se indica que el usuario "Bryan" tiene acceso permitido y se registra en la base de datos. La Figura 3.19 muestra los datos correspondientes a los registros, esto es, un valor de 1 para el tipo 1 y 0 para el tipo 2. Estos valores indican que el usuario pasó primero por el RFID 1 y luego por el RFID 0, lo cual se interpreta como una salida.

Este proceso confirma el funcionamiento del sistema para registrar tanto ingresos como salidas, permitiendo una gestión precisa de la asistencia a eventos. La consistencia en la interpretación de los datos y la correcta asignación de tipos de registros demuestran la efectividad del sistema de detección de asistencia basado en la tecnología RFID implementado en el proyecto.

Finalmente, al cambiar de sección y hacer clic en la pestaña de "Asistencias", se puede apreciar el registro correspondiente al usuario "Bryan". En este registro se presenta el nombre del usuario, la fecha y hora de entrada al evento (10:59:05) y la fecha y hora de salida al evento (11:15:13). Se realiza el cálculo del tiempo total que el usuario permaneció en el evento, siendo este de 16.43 minutos que se clasifica como una asistencia correcta.

Es relevante destacar que el tiempo requerido para que una asistencia sea considerada como correcta está sujeto a las configuraciones realizadas por el usuario. En este escenario de pruebas prácticas, se ha establecido que una asistencia será válida únicamente si supera los 15 minutos de estancia dentro del evento; de lo contrario, se marcará como asistencia incorrecta. Este enfoque flexible permite ajustar el sistema según las necesidades específicas del evento y garantiza una gestión precisa y personalizada de las asistencias.

En las figuras siguientes, se documenta el proceso llevado a cabo durante una entrada correspondiente desde RFID 0 hasta RFID 1, seguida por una salida desde RFID 1 hasta RFID 0. El propósito fundamental de esta prueba es validar el comportamiento del sistema cuando el tiempo de estancia es inferior a 15 minutos.

En este escenario específico, se busca verificar la capacidad del sistema para identificar y registrar asistencias incorrectas, dado que el tiempo de permanencia no alcanza el umbral establecido. Este tipo de pruebas contribuye a evaluar la precisión del sistema en situaciones donde los participantes ingresan y salen del evento en lapsos de tiempo más cortos, proporcionando así una visión completa de su desempeño en diversas circunstancias.

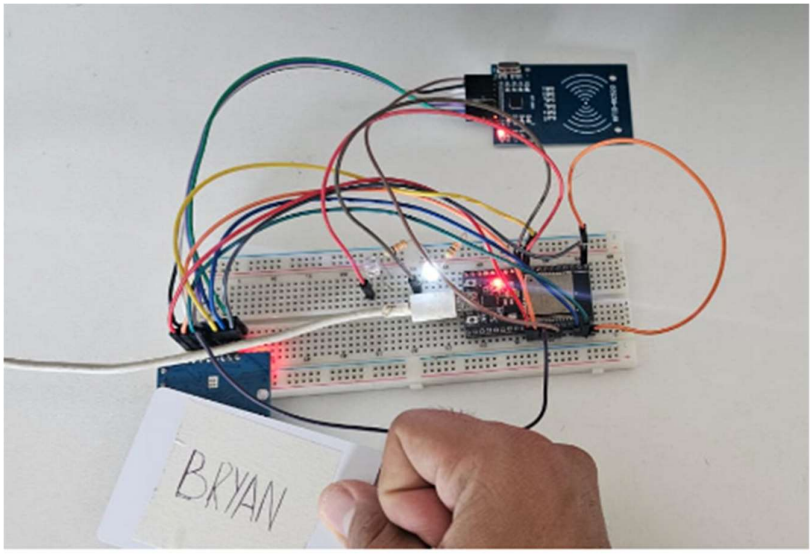


Figura 3.20.- Validación de usuario “Bryan”. Lector RFID 0

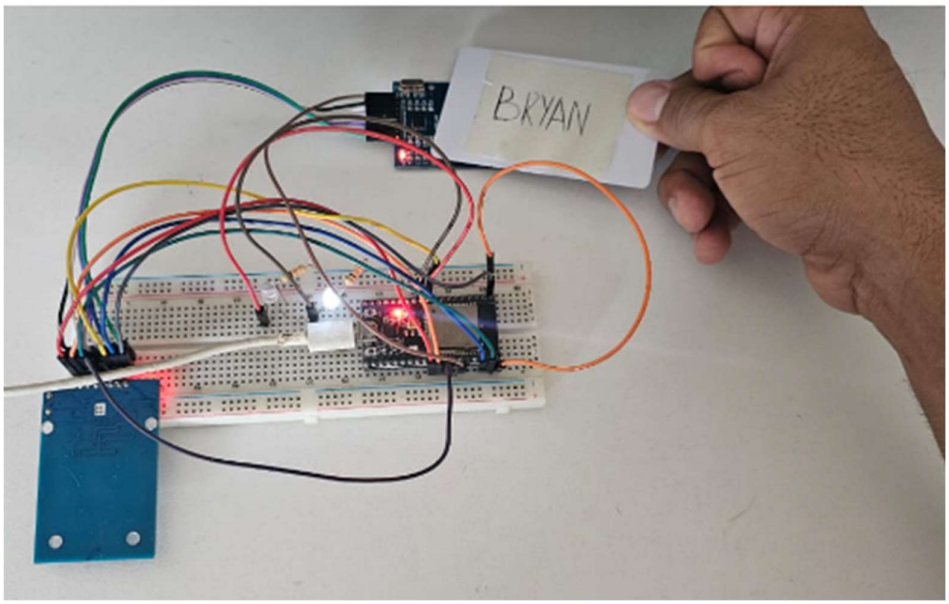


Figura 3.21.- Validación de usuario “Bryan”. Lector RFID 1

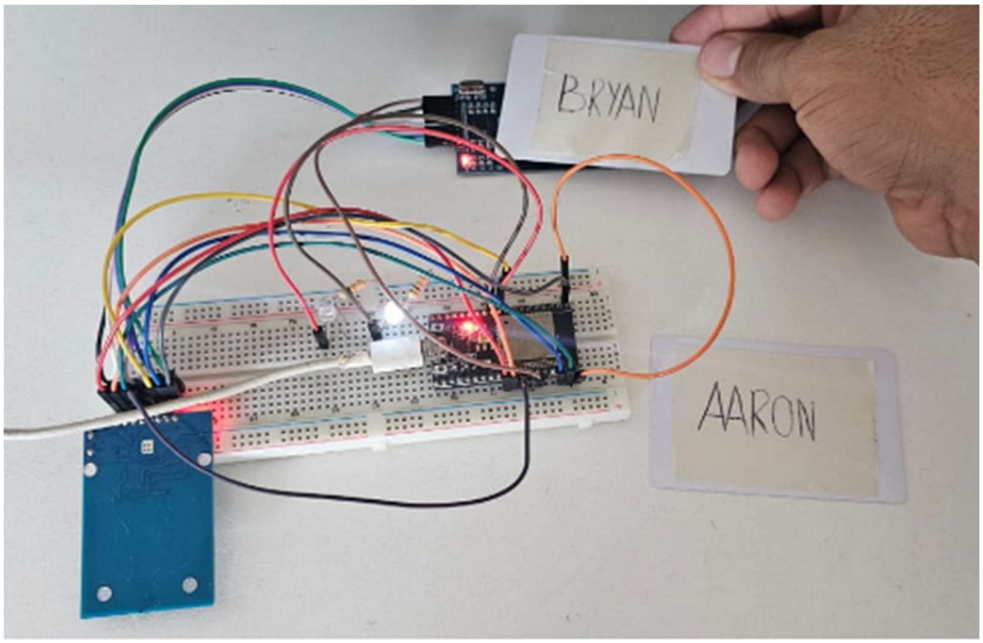


Figura 3.22.- Validación de usuario “Bryan”. Lector RFID 1

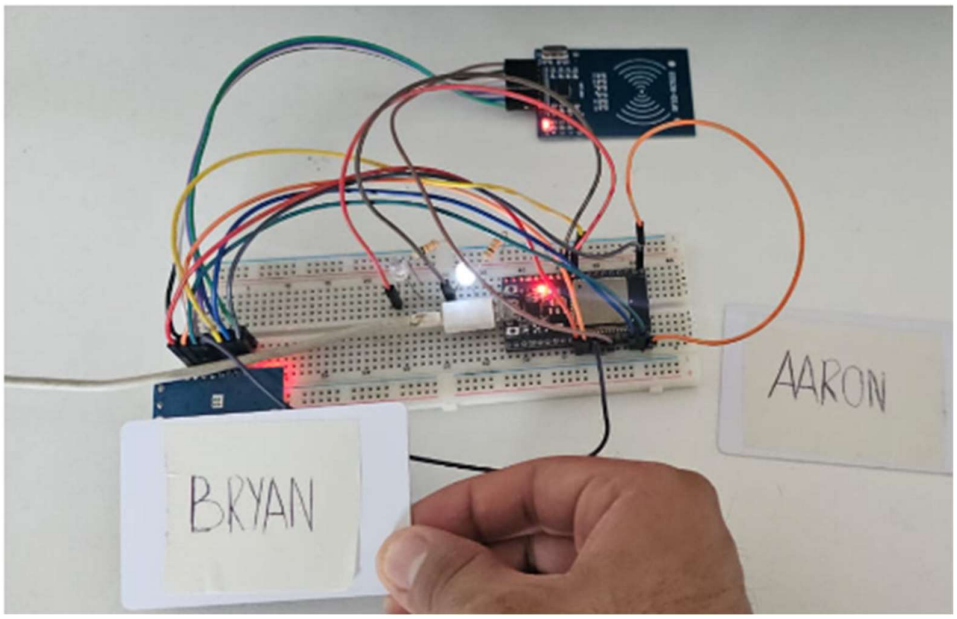


Figura 3.23.- Validación de usuario “Bryan”. Lector RFID 0

Finalmente, en la Figura 3.24, se confirma que el tiempo de estancia registrado fue de 4.02 minutos, siendo este tiempo inferior al umbral establecido por lo que la asistencia correspondiente se marcará como incorrecta. La capacidad de marcar asistencias incorrectas según las configuraciones establecidas refleja la robustez y la precisión del sistema de detección de asistencia implementado.

Figura 3.24.- Validación de tipo de registro y Asistencia.

Capítulo 4

4. CONCLUSIONES Y RECOMENDACIONES

En esta sección final, se presentan las conclusiones derivadas del estudio y desarrollo del sistema de detección de asistencia a eventos, basado en la tecnología RFID y los microcontroladores ESP32 y Arduino Mega. A lo largo de este trabajo, se ha realizado un exhaustivo análisis de las capacidades, limitaciones y aplicaciones prácticas de estas tecnologías en el contexto de la gestión y monitoreo de asistencia. Las conclusiones se fundamentan en los resultados obtenidos de la implementación, proporcionando una visión integral de la efectividad del sistema propuesto.

4.1 Conclusiones

- Los resultados obtenidos, en la implementación práctica, proporciona una comprensión integral de la efectividad del sistema de detección y registro de asistencia basado en RFID. Este sistema demuestra su efectividad en la gestión y monitoreo de asistentes a eventos, destacándose por su adaptabilidad y eficacia.

- La implementación de la tecnología RFID, en conjunto con una base de datos alojada en FireBase, ha permitido la detección en tiempo real de los asistentes, asegurando una base de datos confiable y robusta que resulta crucial para una administración eficiente del evento. Además, la codificación realizada tanto en Python como en Arduino ha sido fundamental para esta integración, asegurando una funcionalidad y estabilidad del sistema RFID.

- El análisis de los resultados del sistema RFID propuesto revela un rendimiento destacado en la gestión de asistencia a eventos. Las pruebas confirman su precisión y eficiencia, mostrando potencial para mejorar la seguridad y la experiencia de los asistentes, cumpliendo con los estándares modernos de registro y monitoreo.

4.2 Recomendaciones

- Se recomienda fomentar la innovación en el diseño y desarrollo de sistemas de asistencia más avanzados y adaptados a las necesidades cambiantes del sector, integrando nuevas tecnologías y metodologías.

- Se sugiere llevar a cabo investigaciones adicionales para mejorar el diseño del sistema RFID actual, con el fin de optimizar aún más su eficiencia y efectividad.

- Para garantizar una adaptabilidad futura, se recomienda la actualización del sistema de registro de asistencia RFID, incorporando avances tecnológicos y retroalimentación de usuarios para mantener la relevancia y eficiencia del sistema frente a las cambiantes demandas del mercado.

- Se sugiere explorar la integración de capacidades de análisis de datos en tiempo real, lo cual permitiría a los organizadores de eventos obtener conocimientos valiosos sobre el comportamiento de los asistentes, optimizando así la planificación y la asignación de recursos para futuros eventos.

REFERENCIAS

[1] R. A. Ramiréz Urquijo y F. A. Cruz Laguna, Artists, *Sistema para el control de asistencia de docentes y estudiantes en el encuentro académicos.* [Art]. ESPOL, 2012.

[2] J. E. Rocafuerte Villón y C. A. Jaramillo Espinoza, Artists, *Diseño e implementación de sistemas para tomar automáticamente asistencia a clase usando visión artificial y técnología móbiles.* [Art]. ESPOL, 2018.

[3] D. Chang Falconpi y A. Lozano Solís, Artists, *Desarrollo e implementación de un sistema de control e inventario continuo, utilizando tecnolo´gia RFID, para la biblioteca de la UPS sede guayaquil..* [Art]. Universidad Politécnica Salesiana, 2013.

[4] J. P. Palacios Andrade y P. A. Rodas Lema, Artists, *Desarrollo de interfaces para equipos de identificación por radio frecuencia, aplicado al control de ingreso de estudiantes a los laboratorios..* [Art]. Universidad de Azuay, 2012.

[5] BAROIG, «Sistema de Etiquetado RFID para identificación de Activos.,» 2021. [En línea]. Available: https://baroig.com/impresion-etiquetado/sistema-rfid-etiquetas-identificacion-activos/.

[6] DIPOLE/RFID, «Etiquetas RFID,» 2018. [En línea]. Available: https://www.dipolerfid.es/blog-rfid/etiquetas-rfid-y-aplicaciones.

[7] ESPRESSIF, «ESP32,» 2024. [En línea]. Available: https://www.espressif.com/en/products/socs/esp32.

[8] NextPoints, «Precios, tarifas y costos del RFID,» [En línea]. Available: https://nextpoints.com/costos-rfid/#:~:text=Etiquetas%20RFID%20pasivas,0%2C15%E2%82%AC%20por%20unidad..

[9] ElectronicWing, «RFID RC522 Interfacing with ESP32,» [En línea]. Available: https://www.electronicwings.com/esp32/rfid-rc522-interfacing-with-esp32.

[10] ARDUINOFORUM, «Arduino Mega 2560 y RFID,» [En línea]. Available: https://forum.arduino.cc/t/arduino-mega-2560-rfid/479440.

APENDICE

ARDUINO IDE

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

#include <WiFi.h>
#include <Firebase_ESP_Client.h>
#include "addons/TokenHelper.h"
#include "addons/RTDBHelper.h"
#include "time.h"
#include "sntp.h"

// Configuración del RFID de entrada
#define Vcc <->3V3(o Vin(5V) según la versión del módulo)
#define RST (Reset) <->D22
#define GND (Masse) <->GND
#define MISO (Master Input Slave Output) <->19
#define MOSI (Master Output Slave Input) <->23
#define SCK (Serial Clock) <->18
#define SS / SDA(Slave select) <->5
#define SS_PIN 5
#define RST_PIN 22
#define SS_PIN2 21

MFRC522 rfid(SS_PIN, RST_PIN);   // Instance of the class
MFRC522 rfid2(SS_PIN2, RST_PIN); // Instance of the class
MFRC522::MIFARE_Key key;

// Configuración del WIFI
#define WIFI_SSID "Reda"
#define WIFI_PASSWORD "a2st2000"

// Configuración de Firebase
#define API_KEY "AIzaSyD0AEe-4gB6pU5XvVqtMVw90qIL1EEBJ7k"
#define DATABASE_URL "https://rfid-caf2b-default-rtdb.firebaseio.com/"
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;
FirebaseJson   json;   //   or   constructor   with   contents   e.g.   FirebaseJson
json("{\"a\":true}");
unsigned long sendDataPrevMillis = 0;
```

```
bool signupOK = false;

// Configuracion del servidor NTP
const char *ntpServer = "pool.ntp.org"; // Servidor NTP
const long gmtOffset_sec = -5 * 3600;   // Desplazamiento GMT
const int daylightOffset_sec = 0;

// Configuracion del LED

#define RED_LED_PIN 26
#define GREEN_LED_PIN 27

void setup()
{
  Serial.begin(115200);

  pinMode(GREEN_LED_PIN, OUTPUT);
  pinMode(RED_LED_PIN, OUTPUT);

  SPI.begin();      // Init SPI bus
  rfid.PCD_Init();  // Init MFRC522
  rfid2.PCD_Init(); // Init MFRC522
  Serial.println();

  Serial.println();
  Serial.println();
  Serial.println("Iniciando el Programa");

  connectToWifi();

  authFirebase();

  // Init and get the time
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
}

void loop()
{

  if (rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial())
  {
    Serial.println("Tarjeta detectada!");

    Serial.print("Código de la tarjeta (string): ");
```

```
    String codigo = "";
    for (byte i = 0; i < rfid.uid.size; i++)
    {
      codigo += String(rfid.uid.uidByte[i], HEX);
    }
    Serial.println("leyendo rfid 0");
    Serial.println(codigo);

    isValidCode(codigo, 0);

    rfid.PICC_HaltA();
    rfid.PCD_StopCrypto1();
  }

  if (rfid2.PICC_IsNewCardPresent() && rfid2.PICC_ReadCardSerial())
  {
    Serial.println("Tarjeta detectada!");

    Serial.print("Código de la tarjeta (string): ");
    String codigo = "";
    for (byte i = 0; i < rfid2.uid.size; i++)
    {
      codigo += String(rfid2.uid.uidByte[i], HEX);
    }
    Serial.println("leyendo rfid 1");
    Serial.println(codigo);

    isValidCode(codigo, 1);

    rfid2.PICC_HaltA();
    rfid2.PCD_StopCrypto1();
  }
}

// Para conectar por WiFi
void connectToWifi()
{

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Conectando a WiFi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
```

```
  }
  Serial.println();

  Serial.print("WiFi conectado");
}

// Para conectarse a Firebase
void authFirebase()
{
  config.api_key = API_KEY;
  config.database_url = DATABASE_URL;
  if (Firebase.signUp(&config, &auth, "", ""))
  {
    Serial.println("Ingreso exitoso");
    signupOK = true;
  }

  else
  {
    Serial.printf("%s\n", config.signer.signupError.message.c_str());
  }

  /* Assign the callback function for the long running token generation task */
  config.token_status_callback = tokenStatusCallback; // see addons/TokenHelper.h

  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);
}

void isValidCode(String codigo, int tipo)
{
  if (Firebase.ready() && signupOK)
  {
    sendDataPrevMillis = millis();

    Serial.println("OBTENIENDO DATOS");

    if (Firebase.RTDB.getJSON(&fbdo, "/users/"))
    {

      Serial.print("Data from Firebase: ");
      Serial.println(fbdo.dataPath());
      // Obtén el JSON como String
      String jsonString = fbdo.to<String>().c_str();
```

```
      Serial.println(jsonString);
      processJsonData(jsonString, codigo, tipo);
    }
    else
    {
      Serial.println(fbdo.errorReason());
    }
  }
  else
  {
    Serial.println("Error conectando con Firebase");
  }
}

void ledAlert(int LED)
{
  digitalWrite(LED, HIGH);
  delay(1000);
  digitalWrite(LED, LOW);
  delay(1000);
}

void processJsonData(const String &jsonString, const String codigo, int tipo)
{
   DynamicJsonDocument doc(10240); // Ajusta el tamaño del documento según sea
necesario
  DeserializationError error = deserializeJson(doc, jsonString);

  bool isAllow = false;

  if (error)
  {
    Serial.print("Error al deserializar JSON: ");
    Serial.println(error.c_str());
  }

  Serial.println("Datos del JSON:");

  // Recorre los elementos del JSON
  for (JsonPair kv : doc.as<JsonObject>())
  {
    const char *key = kv.key().c_str();
    JsonObject value = kv.value();
```

```
  if (value["codigo"].as<String>() == codigo)
  {

    if (!value.containsKey("horaI"))
    {
      Serial.println("No se encontró horaI en Db");
      break;
    }

    if (!value.containsKey("horaF"))
    {
      Serial.println("No se encontró horaF en Db");
      break;
    }

    String horaI = value["horaI"].as<String>();
    String horaF = value["horaF"].as<String>();

    String currentDate = getLocalTime();

    if (currentDate == "")
    {
      Serial.println("Error verificando usuario");
      break;
    }

    bool isInRange = isHourValid(horaI, horaF, currentDate);

    if (value["isAllowed"].as<bool>() && isInRange)
    {
      Serial.print("ACCESO PERMITIDO");
      isAllow = true;
      sendRecord(key, tipo, currentDate);
      break;
    }
    else
    {
      Serial.print("ACCESO NO PERMITIDO");
      break;
    }
  }
}
```

```
   Serial.print(isAllow ? "SE ENCONTRARON COINCIDENCIAS" : "NO SE ENCONTRARON
COINCIDENCIAS");

  isAllow ? ledAlert(GREEN_LED_PIN) : ledAlert(RED_LED_PIN);
}

// Enviar registro a firebase
void sendRecord(String idUser, int tipo, String currentDate)
{

  // String dateStr = getLocalTime();
  FirebaseJson json2;
  json2.set("fecha", currentDate);
  json2.set("idUser", idUser);
  json2.set("tipo", tipo);

  if (Firebase.ready() && signupOK)
  {

    // TODO Aqui se deben obtener los datos del RFID, en el momento en que
    if (Firebase.RTDB.pushJSON(&fbdo, "/registros/", &json2))
    {
      Serial.println("PASSED");
      Serial.println("PATH: " + fbdo.dataPath());
      Serial.println("TYPE: " + fbdo.dataType());
    }
    else
    {
      Serial.println("FAILED");
      Serial.println("REASON: " + fbdo.errorReason());
    }
  }

  else
  {

    Serial.println("Error conectando con Firebase");
  }
}

String getLocalTime()
{
  struct tm timeinfo;
  if (!getLocalTime(&timeinfo))
```

```
  {
    Serial.println("Error obteniendo fecha actual");
    return "";
  }

  // Serial.println(&timeinfo, "%Y-%m-%d %H:%M:%S");

  char buffer[20]; // Ajusta el tamaño del búfer según sea necesario
  strftime(buffer, sizeof(buffer), "%Y-%m-%d %H:%M:%S", &timeinfo);

  return String(buffer);
}

bool isHourValid(String horaInicial, String horaFinal, String currentDate)
{

  // Divide las horas en horas y minutos
  int hC = currentDate.substring(11, 13).toInt();
  int mC = currentDate.substring(14, 16).toInt();

  int hI = horaInicial.substring(0, 2).toInt();
  int mI = horaInicial.substring(3, 5).toInt();

  int hF = horaFinal.substring(0, 2).toInt();
  int mF = horaFinal.substring(3, 5).toInt();

  // Calcula los minutos totales
  int currentMinutes = hC * 60 + mC;
  int initMinutes = hI * 60 + mI;
  int finalMinutes = hF * 60 + mF;

  // Comprueba si los minutos generados están dentro del rango
  return (currentMinutes >= initMinutes) && (currentMinutes <= finalMinutes);
}
```

PYTHON

SERVICES

```
from firebase import firebase
from datetime import datetime, timedelta
class FireServer:
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

        self.fb = firebase.FirebaseApplication(
            "https://rfid-caf2b-default-rtdb.firebaseio.com/"
        )

    def getRecords(self):
        data = self.fb.get("/registros/", "")
        users = self.fb.get("/users/", "")
        records = data.values()

        # Crear un nuevo diccionario organizado por 'idUser'
        diccionario_organizado = {}

        # Recorremos cada record para modificar su nombre y codigo
        for record in records:
            user = users[record["idUser"]]

            record["nombre"] = user["nombre"]
            record["codigo"] = user["codigo"]

            id_usuario = record["idUser"]
            if id_usuario not in diccionario_organizado:
                # Si es registro nuevo
```

```
            diccionario_organizado[id_usuario] = []

            # se crea una plantilla de record
            newRecord = self.getNewRecord(record)

            # se modifica la plantilla con el record
            self.setRecord(newRecord, record)

            diccionario_organizado[id_usuario].append(newRecord)
        else:
            lastRecord = diccionario_organizado[id_usuario][-1]
            if lastRecord["isCompleted"]:
                newRecord = self.getNewRecord(record)
                self.setRecord(newRecord, record)
                diccionario_organizado[id_usuario].append(newRecord)
            else:
                self.setRecord(lastRecord, record)

# Imprimir el nuevo diccionario
for id_usuario, lista_usuarios in diccionario_organizado.items():
    print(f"ID de Usuario: {id_usuario}")
    for usuario in lista_usuarios:
        print(usuario)
    print("------")

print("DICCIONAARIO")
print(diccionario_organizado)

# Obtener todos los valores en una sola lista
listValues = sum(diccionario_organizado.values(), [])
```

```
    print("VALORES")
    print(listValues)

    return listValues

def getNewRecord(self, record):
    return {
        "idUser": record["idUser"],
        "nombre": record["nombre"],
        "date1": "",
        "date2": "",
        "codigo": record["codigo"],
        "tipo1": "",
        "tipo2": "",
        "registro": "",
        "isCompleted": False,
    }

def setRecord(self, template, record):
        # Caso inicial
        if template["tipo1"] == "":
            template["date1"] = record["fecha"]
            template["tipo1"] = record["tipo"]
        # Caso final
        else:
            template["date2"] = record["fecha"]
            template["tipo2"] = record["tipo"]

            try:
```

```
            if template["tipo1"] == 0 and template["tipo2"] == 1:
                template["registro"] = "Ingreso"
            elif template["tipo1"] == 1 and template["tipo2"] == 0:
                template["registro"] = "Salida"

            else:
                template["registro"] = "Incompleto"

            template["isCompleted"] = True
        except ValueError as e:
            # Manejar el error de valor incorrecto
            print("Error:", e)

def getAssistance(self):
    records = self.getRecords()

    print("lista de records")
    print(records)

    # Crear un nuevo diccionario organizado por 'idUser'
    diccionario_organizado = {}

    for record in records:
        id_usuario = record["idUser"]
        if id_usuario not in diccionario_organizado:
            diccionario_organizado[id_usuario] = []

            newAssistance = self.getNewAssistance(record["nombre"])
            self.setAssistance(newAssistance, record)
            diccionario_organizado[id_usuario].append(newAssistance)
```

```
            else:
                lastAssistance = diccionario_organizado[id_usuario][-1]
                if lastAssistance["isCompleted"]:
                    newAssistance = self.getNewAssistance(record["nombre"])
                    self.setAssistance(newAssistance, record)
                    diccionario_organizado[id_usuario].append(newAssistance)
                else:
                    self.setAssistance(lastAssistance, record)

        # Imprimir el nuevo diccionario
        for id_usuario, lista_usuarios in diccionario_organizado.items():
            print(f"ID de Usuario: {id_usuario}")
            for usuario in lista_usuarios:
                print(usuario)
            print("------")

        print("DICCIONAARIO")
        print(diccionario_organizado)

        # Obtener todos los valores en una sola lista
        listValues = sum(diccionario_organizado.values(), [])

        print("VALORES")
        print(listValues)
        return listValues

    def getNewAssistance(self, nombre):
        return {
            "nombre": nombre,
            "startdate": "",
```

```
        "enddate": "",
        "tiempo": "",
        "asistencia": False,
        "isCompleted": False,
    }

def setAssistance(self, assistance, record):
    if record["registro"] == "Ingreso" and assistance["startdate"] == "":
        assistance["startdate"] = record["date1"]
    elif record["registro"] == "Salida" and assistance["enddate"] == "":
        assistance["enddate"] = record["date2"]

        try:
            # Convertir cadenas a objetos datetime
            fecha_inicio = datetime.fromisoformat(assistance["startdate"])
            fecha_fin = datetime.fromisoformat(assistance["enddate"])

            # Calcular la diferencia
            diferencia = fecha_fin - fecha_inicio
            diferencia = diferencia.total_seconds() / 60
            diferencia = round(diferencia, 2)

            assistance["asistencia"] = diferencia > 15

            assistance["tiempo"] = f"{diferencia}"

            assistance["isCompleted"] = True
        except ValueError as e:
            # Manejar el error de valor incorrecto
            print("Error:", e)
```

```
            print(
                "Por favor, ingrese una fecha y hora válida en el formato especificado."
            )

    def getUsers(self):
        return self.fb.get("/users/", "")

    def newUser(self, name, code, horaI, horaF, isAllowed):
        datos = {
            "nombre": name,
            "codigo": code,
            "horaI": horaI,
            "horaF": horaF,
            "isAllowed": isAllowed,
        }
        resultado = self.fb.post("/users/", datos)
        print(resultado)

    def editUser(self, id, name, code, horaI, horaF, isAllowed):
        datos = {
            "nombre": name,
            "codigo": code,
            "horaI": horaI,
            "horaF": horaF,
            "isAllowed": isAllowed,
        }
        resultado = self.fb.put("/users/", id, datos)
        print(resultado)

    def delUser(self, id):
```

```
    resultado = self.fb.delete("/users/", id)
    print(resultado)

def addListRecords(self, names):
    for name in names:
        now = datetime.now()

        self.datos = {
            "fecha": now,
            "tipo": 0,
            "idUser": name["name"],
        }

        # Para enviar datos
        self.fb.post("/registros/", self.datos)

    for name in names:
        now = datetime.now()
        now = now + timedelta(hours=1)

        self.datos = {
            "fecha": now,
            "tipo": 1,
            "idUser": name["name"],
        }

        resultado = self.fb.post("/registros/", self.datos)

def addListUsers(self):
    listNames = []
```

```
        for i in range(2):
            index = i + 1
            self.datos = {
                "nombre": "usuario" + str(index),
                "codigo": str(index),
                "isAllowed": True,
            }
            # Para enviar datos
            resultado = self.fb.post("/users/", self.datos)

            listNames.append(resultado)

        self.addListRecords(listNames)
```

REGISTRO

```
from tkinter import messagebox
from tkinter import ttk
from firebase import firebase
from datetime import datetime

class RegistroFrame(tk.Frame):
    def __init__(self, _, fb):
        super().__init__()
        self.fb = fb

        # Crear el Frame
        self.records_page_fm = tk.Frame(_, bg="gray")

        self.tree = ttk.Treeview(
            self.records_page_fm,
```

```
    columns=("Fecha1", "Fecha2", "Código", "Tipo1", "Tipo2", "Registro"),
)

self.tree.heading("#0", text="Nombre")
self.tree.heading("Fecha1", text="Fecha1")
self.tree.heading("Fecha2", text="Fecha2")
self.tree.heading("Código", text="Código")
self.tree.heading("Tipo1", text="Tipo1")
self.tree.heading("Tipo1", text="Tipo2")
self.tree.heading("Registro", text="Registro")

self.fillData()

# Crear y configurar la barra de desplazamiento
scrollbar = ttk.Scrollbar(
    self.records_page_fm,
    orient="vertical",
    command=self.actualizar_desplazamiento,
)
# self.tree.configure(yscrollcommand=scrollbar.set)

# Create a horizontal Scrollbar
h_scroll = ttk.Scrollbar(
    self.records_page_fm, orient="horizontal", command=self.tree.xview
)
# h_scroll.pack(side=tk.BOTTOM, fill=tk.X)

# Configure the Treeview to use the horizontal scrollbar
self.tree.configure(yscrollcommand=scrollbar.set, xscrollcommand=h_scroll.set)
```

```
    # Ubicar el TreeView y la barra de desplazamiento en la ventana
    self.tree.grid(row=0, column=0, sticky="nsew")
    scrollbar.grid(row=0, column=1, sticky="ns")
    h_scroll.grid(row=1, column=0, sticky="ew")

    # Hacer que la ventana redimensione el TreeView cuando cambie de tamaño
    self.records_page_fm.grid_rowconfigure(0, weight=1)
    self.records_page_fm.grid_columnconfigure(0, weight=1)

    # Centrar los datos en el TreeView
    self.centrar_datos(self.tree)

    self.records_page_fm.pack(fill=tk.BOTH, expand=True)

def fillData(self):
    try:
        # TODO se debe ordenar los campos por fecha

        # Para obtener datos de firebase

        self.records = self.fb.getRecords()

        print("Se obtuvieron los registros")

        # Para iterar los datos
        for record in self.records:
            self.tree.insert(
                "",
                tk.END,
                text=record["nombre"],
```

```
                values=(
                    record["date1"],
                    record["date2"],
                    record["codigo"],
                    record["tipo1"],
                    record["tipo2"],
                    record["registro"],
                ),
            )
    except ValueError as e:
        # Manejar el error de valor incorrecto
        print("Error:", e)

def formatDate(self, fecha):
    try:
        # Convertir el string a un objeto datetime
        fecha_hora_obj = datetime.strptime(fecha, "%Y-%m-%d %H:%M:%S")

        # Formatear la fecha y hora en un nuevo formato
        nuevo_formato = fecha_hora_obj.strftime("%Y-%m-%d %H:%M:%S")

        return nuevo_formato
    except ValueError as e:
        return ""

def centrar_datos(self, tree):
    for column in tree["columns"]:
        tree.column(column, anchor="center")  # This will center text in rows
        tree.heading(column, text=column)
```

```
    # Función para actualizar la posición de la vista del TreeView cuando se desplaza la barra de desplazamiento
    def actualizar_desplazamiento(self, *args):
        self.tree.yview(*args)

    def get_type(self, tipo):
        if tipo == 0:
            return "Sensor 1"
        else:
            return "Sensor 2"
```

ASISTENCIA

```
import tkinter as tk
import tkinter as tk
from tkinter import messagebox

from tkinter import ttk

from firebase import firebase
from datetime import datetime



class AsistenciaFrame(tk.Frame):
    def __init__(self, _, fb):
        super().__init__()
        self.fb = fb

        # Crear el Frame
        self.assistance_page_fm = tk.Frame(_, bg="gray")
```

```
self.tree = ttk.Treeview(
    self.assistance_page_fm,
    columns=("Fecha Inicial", "Fecha Final", "Tiempo", "Asistencia"),
)

self.tree.heading("#0", text="Nombre")
self.tree.heading("Fecha Inicial", text="Fecha Inicial")
self.tree.heading("Fecha Final", text="Fecha Final")
self.tree.heading("Tiempo", text="Duración")
self.tree.heading("Asistencia", text="Asistencia")

self.fillData()
# Crear y configurar la barra de desplazamiento
scrollbar = ttk.Scrollbar(
    self.assistance_page_fm,
    orient="vertical",
    command=self.actualizar_desplazamiento,
)
self.tree.configure(yscrollcommand=scrollbar.set)

# Ubicar el TreeView y la barra de desplazamiento en la ventana
self.tree.grid(row=0, column=0, sticky="nsew")
scrollbar.grid(row=0, column=1, sticky="ns")

# Hacer que la ventana redimensione el TreeView cuando cambie de tamaño
self.assistance_page_fm.grid_rowconfigure(0, weight=1)
self.assistance_page_fm.grid_columnconfigure(0, weight=1)

# Centrar los datos en el TreeView
self.centrar_datos(self.tree)
```

```
    self.assistance_page_fm.pack(fill=tk.BOTH, expand=True)

def fillData(self):
    try:
        # TODO se debe ordenar los campos por fecha

        # Para obtener datos de firebase

        print("Se obtuvo el registro de asistencias")

        self.records = self.fb.getAssistance()

        print(self.records)
        # Para iterar los datos
        for record in self.records:
            # print(record)
            self.tree.insert(
                "",
                tk.END,
                text=record["nombre"],
                values=(
                    self.formatDate(record["startdate"]),
                    self.formatDate(record["enddate"]),
                    record["tiempo"],
                    self.formatAssistance(record["asistencia"]),
                ),
            )
    except ValueError as e:
        # Manejar el error de valor incorrecto
```

```
            print("Error:", e)

    def centrar_datos(self, tree):
        for column in tree["columns"]:
            tree.column(column, anchor="center")  # This will center text in rows
            tree.heading(column, text=column)

    def formatDate(self, fecha):
        try:
            # Convertir el string a un objeto datetime
            fecha_hora_obj = datetime.strptime(fecha, "%Y-%m-%d %H:%M:%S")

            # Formatear la fecha y hora en un nuevo formato
            nuevo_formato = fecha_hora_obj.strftime("%Y-%m-%d %H:%M:%S")

            return nuevo_formato
        except ValueError as e:
            return ""

    def formatAssistance(self, isAssistance):
        if isAssistance:
            return "Correcto"
        else:
            return "Incorrecto"

    # Función para actualizar la posición de la vista del TreeView cuando se desplaza la
barra de desplazamiento
    def actualizar_desplazamiento(self, *args):
        self.tree.yview(*args)
```

USERS

```
from tkinter import messagebox
from tkinter import ttk
from firebase import firebase

class UserFrame(tk.Frame):
    def __init__(self, _, fb):
        super().__init__()
        self.fb = fb

        # Crear el Frame

        self.users_page_fm = tk.Frame(_, bg="gray")

        self.create_variables()
        self.create_form(self.users_page_fm)

        self.tree = ttk.Treeview(
            self.users_page_fm, columns=("Código", "HoraI", "HoraF", "Acceso")
        )

        self.tree.heading("#0", text="Nombre")
        self.tree.heading("Código", text="Código")
        self.tree.heading("HoraI", text="HoraI")
        self.tree.heading("HoraF", text="HoraF")
        self.tree.heading("Acceso", text="Acceso")

        self.fillData()

        # Enlazar la función al evento de selección
```

```
        self.tree.bind("<ButtonRelease-1>", self.on_treeview_select)

        # Crear y configurar la barra de desplazamiento
        scrollbar = ttk.Scrollbar(
            self.users_page_fm,
            orient="vertical",
            command=self.actualizar_desplazamiento,
        )
        self.tree.configure(yscrollcommand=scrollbar.set)

        # Ubicar el TreeView y la barra de desplazamiento en la ventana
        self.tree.grid(row=1, column=0, sticky="nsew")
        scrollbar.grid(row=1, column=1, sticky="ns")

        # Hacer que la ventana redimensione el TreeView cuando cambie de tamaño
        self.users_page_fm.grid_rowconfigure(1, weight=1)
        self.users_page_fm.grid_columnconfigure(0, weight=1)

        # self.tree.pack(fill=tk.BOTH, expand=True)

        # Centrar los datos en el TreeView
        self.centrar_datos(self.tree)

        self.users_page_fm.pack(fill=tk.BOTH, expand=True)

    def create_variables(self):
        # Crear una variable de tipo StringVar
        self.id = ""
        self.name = tk.StringVar(value="")
        self.code = tk.StringVar(value="")
```

```
    self.horaI = tk.StringVar(value="")
    self.horaF = tk.StringVar(value="")
    self.allowed = tk.BooleanVar(value=True)

def create_form(self, frame):
    users_page_container = tk.Frame(frame)
    users_page_container.grid(row=0, column=0, columnspan=6, sticky="nsew")

    nombre_lb = tk.Label(users_page_container, text="Nombre", relief="solid")
    nombre_lb.grid(row=0, column=0, padx=5, sticky="nsew")

    self.nombre_entry = tk.Entry(
        users_page_container,
        textvariable=self.name,
    )
    self.nombre_entry.grid(row=1, column=0, padx=5, sticky="nsew")

    codigo_lb = tk.Label(users_page_container, text="Código", relief="solid")
    codigo_lb.grid(row=0, column=1, padx=5, sticky="nsew")

    self.codigo_entry = tk.Entry(users_page_container, textvariable=self.code)
    self.codigo_entry.grid(row=1, column=1, padx=5, sticky="nsew")

    horaI_lb = tk.Label(users_page_container, text="HoraI", relief="solid")
    horaI_lb.grid(row=0, column=2, padx=5, sticky="nsew")

    self.horaI_entry = tk.Entry(users_page_container, textvariable=self.horaI)
    self.horaI_entry.grid(row=1, column=2, padx=5, sticky="nsew")

    horaF_lb = tk.Label(users_page_container, text="HoraF", relief="solid")
```

```
        horaF_lb.grid(row=0, column=3, padx=5, sticky="nsew")

        self.horaF_entry = tk.Entry(users_page_container, textvariable=self.horaF)
        self.horaF_entry.grid(row=1, column=3, padx=5, sticky="nsew")

        allowed_lb = tk.Label(users_page_container, text="Acceso", relief="solid")
        allowed_lb.grid(row=0, column=4, padx=5, sticky="nsew")

        self.allowed_checkbutton = tk.Checkbutton(
            users_page_container,
            text="Permitido",
            variable=self.allowed,
            onvalue=True,
            offvalue=False,
        )
        self.allowed_checkbutton.grid(row=1, column=4, padx=5, sticky="nsew")

        button = tk.Button(users_page_container, text="REGISTRAR",
command=self.newUser)
        button.grid(row=0, column=5, rowspan=2, padx=5, sticky="nsew")

        button = tk.Button(users_page_container, text="EDITAR", command=self.editUser)
        button.grid(row=0, column=6, rowspan=2, padx=5, sticky="nsew")

        button = tk.Button(users_page_container, text="ELIMINAR",
command=self.delUser)
        button.grid(row=0, column=7, rowspan=2, padx=5, sticky="nsew")

        users_page_container.grid_columnconfigure(0, weight=1)
        users_page_container.grid_columnconfigure(1, weight=1)
```

```
    users_page_container.grid_columnconfigure(2, weight=1)
    users_page_container.grid_columnconfigure(3, weight=1)
    users_page_container.grid_columnconfigure(4, weight=1)
    users_page_container.grid_columnconfigure(5, weight=1)
    users_page_container.grid_columnconfigure(6, weight=1)
    users_page_container.grid_columnconfigure(7, weight=1)

def fillData(self):
    try:
        # Para obtener datos de firebase

        # Para recibir datos
        print("Se obtuvo el registro de usuarios")

        self.users = self.fb.getUsers()
        print(self.users)

        # Para iterar los datos
        for key, values in self.users.items():
            self.tree.insert(
                "",
                tk.END,
                key,
                text=values["nombre"],
                values=(
                    values["codigo"],
                    values["horaI"],
                    values["horaF"],
                    self.formatAllowed(values["isAllowed"]),
                ),
```

```
            )
        except:
            print("An exception ocurred")

    def newUser(self):
        self.fb.newUser(
            self.name.get(),
            self.code.get(),
            self.horaI.get(),
            self.horaF.get(),
            self.allowed.get(),
        )
        self.tree.delete(*self.tree.get_children())
        self.fillData()

    def editUser(self):
        seleccion = self.tree.selection()
        if seleccion:
            # Obtener el ID del ítem seleccionado
            id_seleccionado = seleccion[0]
            print("ID seleccionado:", id_seleccionado)

            self.fb.editUser(
                id_seleccionado,
                self.name.get(),
                self.code.get(),
                self.horaI.get(),
                self.horaF.get(),
                self.allowed.get(),
            )
```

```
            self.tree.delete(*self.tree.get_children())
            self.fillData()
        else:
            print("Ningún ítem seleccionado.")

    def delUser(self):
        seleccion = self.tree.selection()
        if seleccion:
            # Obtener el ID del ítem seleccionado
            id_seleccionado = seleccion[0]
            print("ID seleccionado:", id_seleccionado)

            self.fb.delUser(id_seleccionado)

            self.tree.delete(*self.tree.get_children())
            self.fillData()
        else:
            print("Ningún ítem seleccionado.")

    def centrar_datos(self, tree):
        for column in tree["columns"]:
            tree.column(column, anchor="center")  # This will center text in rows
            tree.heading(column, text=column)

    # Función para actualizar la posición de la vista del TreeView cuando se desplaza la barra de desplazamiento
    def actualizar_desplazamiento(self, *args):
        self.tree.yview(*args)
```

```
    def formatAllowed(self, isAllowed):
        if isAllowed:
            return "Permitido"
        else:
            return "Denegado"

    def on_treeview_select(self, event):
        # Obtener el elemento seleccionado
        selected_item = self.tree.selection()

        # Mostrar el elemento seleccionado (en este caso, solo el primer elemento si hay varios)
        if selected_item:
            # Obtener los valores de todas las columnas del elemento seleccionado
            key = self.tree.item(selected_item)["text"]
            item_values = self.tree.item(selected_item)["values"]

            self.id = selected_item[0]
            self.name.set(key)
            self.code.set(item_values[0])
            self.horaI.set(item_values[1])
            self.horaF.set(item_values[2])

            if item_values[3] == "Permitido":
                self.allowed.set(True)
            else:
                self.allowed.set(False)

            print(
```

```
            f"id: {self.id}, name: {self.name.get()}, code: {self.code.get()}, horaI: {self.horaI.get()}, horaF: {self.horaF.get()},allowed: {str(self.allowed.get())}"
        )
```

MAINPAGE

```
import tkinter as tk
import tkinter as tk
from services import FireServer
from registro import RegistroFrame
from asistencia import AsistenciaFrame
from users import UserFrame

class MainPage(tk.Frame):
    def __init__(self, parent, *args, **kwargs):
        super().__init__(parent, *args, **kwargs)
        self.parent = parent

        self.options_fm = tk.Frame(self.parent)
        self.options_fm.pack(pady=5)
        self.options_fm.pack_propagate(False)
        self.options_fm.configure(width=1200, height=35, bg="#254368")

        self.records_indicator_lb = tk.Label(self.options_fm, bg="white")
        self.records_indicator_lb.place(x=150, y=30, width=100, height=2)

        self.records_btn = self.getButton(caption="Registros")

        self.records_btn.place(x=0, y=0, width=400)

        self.attendance_indicator_lb = tk.Label(self.options_fm, bg="#254368")
```

```
        self.attendance_indicator_lb.place(x=550, y=30, width=100, height=2)
        self.attendance_btn = self.getButton(caption="Asistencias")

        self.attendance_btn.place(x=400, y=0, width=400)

        self.users_indicator_lb = tk.Label(self.options_fm, bg="#254368")
        self.users_indicator_lb.place(x=950, y=30, width=100, height=2)
        self.users_btn = self.getButton(caption="Usuarios")

        self.users_btn.place(x=800, y=0, width=400)

        self.centrar_pantalla()

        self.main_fm = tk.Frame(self.parent)
        self.main_fm.pack(fill=tk.BOTH, expand=True)

        self.firebase = FireServer()

        # self.firebase.addListUsers()
        # self.firebase.addListUsers()

        RegistroFrame(self.main_fm, self.firebase)

    def getButton(self, caption):
        return tk.Button(
            self.options_fm,
            text=caption,
            font=("Arial", 13),
            bd=0,
            fg="white",
```

```
            bg="#254368",
            activeforeground="#254368",
            command=lambda: self.switch(caption=caption),
        )
    def switch(self, caption):
        for child in self.options_fm.winfo_children():
            if isinstance(child, tk.Label):
                child["bg"] = "#254368"

        for fm in self.main_fm.winfo_children():
            fm.destroy()
            self.parent.update()

        if caption == "Registros":
            self.records_indicator_lb["bg"] = "white"
            RegistroFrame(self.main_fm, self.firebase)

        elif caption == "Asistencias":
            self.attendance_indicator_lb["bg"] = "white"
            AsistenciaFrame(self.main_fm, self.firebase)

        elif caption == "Usuarios":
            self.users_indicator_lb["bg"] = "white"
            UserFrame(self.main_fm, self.firebase)

    def centrar_pantalla(self):
        #  Obtenemos el largo y  ancho de la pantalla
        wtotal = self.parent.winfo_screenwidth()
        htotal = self.parent.winfo_screenheight()
        #  Guardamos el largo y alto de la ventana
```

```
        wventana = 1200
        hventana = 550

        #  Aplicamos la siguiente formula para calcular donde debería posicionarse
        pwidth = round(wtotal / 2 - wventana / 2)
        pheight = round(htotal / 2 - hventana / 2)

        #  Se lo aplicamos a la geometría de la ventana
        self.parent.geometry(
            str(wventana) + "x" + str(hventana) + "+" + str(pwidth) + "+" + str(pheight)
        )
```

MAIN

```
import time
import tkinter as tk
from mainPage import MainPage


def main():
    root = tk.Tk()
    root.title("Proyecto RFID - ESP32")
    root.geometry()
    root.geometry("1500x750+0+0")
    root.minsize(800, 500)
    root.resizable(height=False, width=False)
    root.configure(bg="#254368")
    MainPage(root)
    root.mainloop()
if __name__ == "__main__":
    main()
```