ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE UN PROTOTIPO PARA REPRODUCCIÓN DE CANCIONES MEDIANTE TARJETAS RFID Y APLICACIÓN MÓVIL

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN REDES Y TELECOMUNICACIONES

CRISTHIAN WILLIAM VERDEZOTO ARAUZ

cristhian.verdezoto@epn.edu.ec

DIRECTOR: ANDRES FERNANDO REYES CASTRO

andres.reyes@epn.edu.ec

DMQ, AGOSTO 2023

CERTIFICACIONES

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

CRISTHIAN WILLIAM VERDEZOTO ARAUZ

cristhian.verdezoto4@gmail.com

cristhian.verdezoto@epn.edu.ec

Certifico que el presente trabajo de integración curricular fue desarrollado por CRISTHIAN WILLIAM VERDEZOTO ARAUZ, bajo mi supervisión.

ANDRES FERNANDO REYES CASTRO
DIRECTOR

andres.reyes@epn.edu.ec

DECLARACIÓN DE AUTORÍA

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

CRISTHIAN WILLIAM VERDEZOTO ARAUZ

DEDICATORIA

Este documento va dedicado a mi familia, gracias al apoyo y esfuerzo que me han brindado he podido alcanzar una de mis metas, los consejos y enseñanzas que ellos me han aportado a lo largo de mi vida han sido instrumento para seguir adelante pese a las dificultades y obstáculos que se han presentado.

A mi madre, porque siempre ha estado junto a mi durante este proceso y en todos los momentos de mi vida pese a cualquier dificultad que se ha presentado, A mi hermana que ha sido un pilar fundamenta en mi vida.

AGRADECIMIENTO

Le agradezco a Dios por darme salud durante toda esta etapa y que mi familia está a mi lado para verme cumplir esta meta. Le Agradezco a mi madre por darme las herramientas para tomar los desafíos y no rendirme hasta lograr mi objetivo.

A mi hermana por darme su consejo cada vez que tomo decisiones y estar siempre conmigo en los momentos buenos y malos que se presentan en la vida.

Les agradezco a mis profesores por compartir sus conocimientos, tener paciencia y ser una guía durante este proceso.

ÍNDICE DE CONTENIDOS

RESUMEN

En el presente Trabajo de Integración Curricular, se desarrolla la implementación de un prototipo enfocado en un reproductor de música utilizando tecnología RFID y una aplicación móvil, para mejorar la interacción con el usuario. Para cumplir con el objetivo se realizó un estudio de los elementos electrónicos necesarios para el funcionamiento óptimo del prototipo, además se hizo la comparación entre diferentes plataformas que permitan la conexión del prototipo con Internet, con el fin de encontrar la solución más viable para el uso de una aplicación móvil vinculada al dispositivo.

La primera parte consistió en realizar el análisis de los componentes utilizados, para ello se compararon de forma técnica diferentes elementos, con el fin de escoger los elementos óptimos para el dispositivo.

Los elementos principales que se eligieron son, el microcontrolador ESP32 este dispositivo fue escogido debido a su precio accesible, dado que Arduino tiene dispositivos similares, pero a precios más altos, el módulo RFID RC522 y el módulo *DFPlayer Mini.* Por otra parte, se añadió un amplificador PAM8403 para mejorar el audio del dispositivo y también se escogió la plataforma *Arduino Cloud* para el desarrollo de la aplicación móvil.

Luego se realizó el diseño del circuito y la programación necesaria para las funciones del prototipo. Se tomo en cuenta aspectos técnicos para que la conexión entre módulos sea correcta, con el fin de evitar daños en los dispositivos y que el prototipo pueda ser escalable, además en esta parte se realizaron simulaciones con *protoboard* hasta que el dispositivo obtuvo un funcionamiento adecuado, para así ensamblar el prototipo y realizar el análisis de resultados finales.

Con base en los resultados obtenidos se concluyó que, el diseño y los elementos electrónicos utilizados en la implementación del prototipo fueron apropiados para la implementación del reproductor de música utilizando tecnología RFID y una aplicación móvil.

PALABRAS CLAVE: Aplicación móvil, Wi-Fi, RFID, ESP32, prototipo, reproductor de música.

ABSTRACT

In this Curricular Integration Project, we develop the implementation of a prototype focused on a music player using RFID technology and a mobile application, to improve the interaction with the user. To meet the objective, a study of the electronic elements necessary for the optimal functioning of the prototype was carried out, and a comparison was made between different platforms that allow the connection of the prototype with the Internet, to find the most viable solution for the use of a mobile application linked to the device.

The first part consisted of analyzing the components used, for which different elements were technically compared to choose the optimal elements for the device.

The main elements that were chosen are the ESP32 microcontroller, this device was chosen due to its price since Arduino has similar devices but at higher prices, the RC522 RFID module and the DFPlayer Mini module. On the other hand, a PAM8403 amplifier was added to improve the audio of the device and the Arduino Cloud platform was also chosen for the development of the mobile application.

Then the circuit design and programming necessary for the functions of the prototype was performed. Technical aspects were considered so that the connection between modules is correct, in order to avoid damage to the devices and that the prototype can be scalable, also in this part simulations were performed with breadboard until the device obtained proper operation, in order to assemble the prototype and perform the analysis of final results.

Based on the results obtained, it was concluded that the design and the electronic elements used in the implementation of the prototype were appropriate for the implementation of the music player using RFID technology and a mobile application.

KEYWORDS: *mobile app, Wi-Fi, RFID, ESP32, prototype, music player.*

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

La tecnología en la actualidad ha permitido innovar en distintas áreas con el objetivo de mejorar la calidad de vida de las personas, dando mayor comodidad en acciones cotidianas como escuchar música. En la actualidad, la forma de manejar los dispositivos digitales y de acceder a sus recursos, es un problema para los adultos mayores y niños debido a varios factores como la perdida de la visión, interfaz compleja desde el punto de vista de los adultos mayores. Por otro lado, los niños tienen una comprensión limitada con respecto a la tecnología.

Para resolver está problemática, en el presente proyecto se desarrolló un reproductor de música interactivo que permita al usuario disfrutar de la música de forma clásica sin interfaces difíciles de comprender, de forma que sea agradable disfrutar de su música favorita, utilizando un microcontrolador ESP32 conectado a un lector de Radiofrecuencia y a un módulo MP3. Mediante la programación de los dispositivos mencionados, el prototipo reproduce música al contacto de una tarjeta de proximidad RFID (*radiofrequency Identification*) por comunicación de radiofrecuencia. El uso de las tarjetas RFID permite que el dispositivo sea interactivo de esta forma se resuelven las dificultades que pueden tener tanto niños como adultos mayores al utilizar un reproductor de música.

Adicionalmente, se utiliza una plataforma IOT (*Internet of Things)*, para visualizar y seleccionar las canciones mediate un dispositivo móvil conectado a la nube, con el objetivo de tener una alternativa para reproducir música desde lugares remotos por medio de Internet. Para que esta función sea implementada correctamente el dispositivo debe tener conexión a Internet de forma inalámbrica.

1.1 Objetivo general

Implementar un prototipo que reproduzca música utilizando tecnología RFID y una aplicación móvil para mejorar la interacción con el usuario.

1.2 Objetivos específicos

- Establecer los requerimientos del prototipo.
- Definir los componentes de hardware y software.
- Diseñar el prototipo.
- Implementar el prototipo.

- Realizar pruebas de funcionamiento.

1.3 Alcance

El presente proyecto, tiene la finalidad de implementar un prototipo de reproductor de canciones que permita reproducir música por medio de un identificador de radiofrecuencia o una aplicación móvil conectada a la nube con la finalidad de que el uso del dispositivo sea utilizado de forma remota, por lo tanto, la placa principal que controla la comunicación entre el dispositivo y la red debe tener comunicación inalámbrica. Para determinar los componentes que se utilizaron en el prototipo se realizó una comparativa y elección de los componentes a utilizar.

El prototipo incluye el diseño de la placa electrónica, un diseño 3D del prototipo, se realizó un diagrama de flujo que representa el funcionamiento del código implementado en el controlador del prototipo.

El prototipo reproduce música a través de dos parlantes integrados, el prototipo tiene funcionalidades básicas como control de volumen, reproducir/parar la canción y reproducción aleatoria. Para la selección de canciones/álbumes se utilizarán tarjetas RFID o mediante la aplicación móvil.

1.4 Marco Teórico

Comunicación por identificación de radio frecuencia RFID

Los primeros usos para la comunicación por identificación de radiofrecuencia sucedieron en el año 1939 con el transpondedor IFF (*Identification Friend or Foe*) para identificar aeroplanos británicos en el canal de la mancha y evitar que reciban fuego amigo; en la actualidad la tecnología es utilizada en varios sectores para diferentes funciones como la identificación, supervisión y autenticación de objetos.

RFID necesita de dos componentes para su funcionamiento: etiquetas y lectores. una etiqueta cumple la función de almacenar y transmitir los datos, por otro lado, un lector recibe la información por media de ondas de radio para que un servidor procese esta información [1], [2].

El sistema RFID está compuesto por los siguientes componentes.

- **Etiqueta o transponder:** este componente contiene el identificador este es adherido a el objeto al cual se debe identificar.
- **Lector:** es un módulo digital que provee de la suficiente energía a la etiqueta para que pueda transmitir al identificador, también captura y digitaliza la señal

de radiofrecuencias emitida por la etiqueta, se encarga también de la transmisión de la información recibida al servidor para que procese la información.

- **Host, aplicación:** es el dispositivo que recibe la información tomada de la conexión entre el *transponder* y el lector RFID.

En la Figura 1.1 se observa un esquema de un sistema RFID.

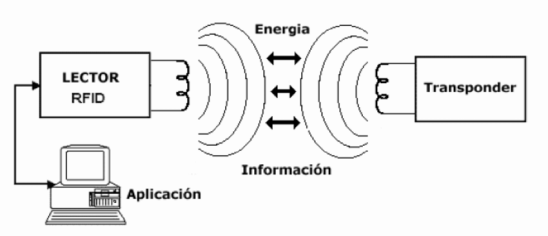

Figura 1.1 Sistema RFID [3]

Microcontrolador

Es un elemento electrónico que tiene la capacidad de controlar y supervisar sistemas electrónicos, está compuesto por un solo chip que tiene una formación lógica de varios circuitos integrados esto permiten realizar control de diversos tipos como procesamiento de datos, gestión de entrada y salida de datos de sensores, control de energía, etc. El control es posible porque el microcontrolador tiene la capacidad de convertir diferentes tipos de señales en señales digitales [4], [5].

La programación de los microprocesadores en sus tiempos de inicio eran un lenguaje ensamblador dado que cada fabricante vendía su propio microprocesador. Sin embargo, en la actualidad se utiliza lenguajes de programación de alto nivel para programar los microcontroladores [4].

El microprocesador está formado por bloques con funciones específicas como:

- **Memoria** es un elemento que funciona como bloque digital, almacena el programa interno que se encarga de proveer las instrucciones y datos para el funcionamiento del microcontrolador [6].
- **Unidad lógica aritmética** es el bloque encargado de ejecutar las operaciones lógicas y aritméticas necesarias para cumplir las tareas específicas del microprocesador, entre las operaciones que realiza este bloque están, la suma lógica de números binarios; la operación lógica AND, OR, OR exclusivo; suma de números decimales de codificación binaria; producto entre números decimales codificados en binario [6].

- **Registro** es la parte de la memoria que se encarga de almacenar datos de lectura y escritura, de acuerdo con el tipo de dato, existen cuatro tipos de registros que son registro acumulador, registro de instrucciones, el registro acumulador y el registro de propósitos generales estos van a ser utilizados de acuerdo con los tipos de datos que reciba el microprocesador [6].
- **Barra ómnibus** son las líneas de conexión entre bloques, pueden estar divididas en varios grupos de líneas, para unir bloques con propósitos especiales al microprocesador como una conexión a memoria externa o para lectura de resultados [6].
- **Unidad de control** es la parte fundamental del microcontrolador dado que, este bloque permite automatizar funciones, sus funciones son sincronización con otras unidades, codificar instrucciones, controlar los circuitos, distribuir la memoria para las microprogramas del microcontrolador, maneja los programas que se ejecutan de forma instantánea [6].

En la Figura 1.2 se muestra la estructura típica de un microcontrolador.

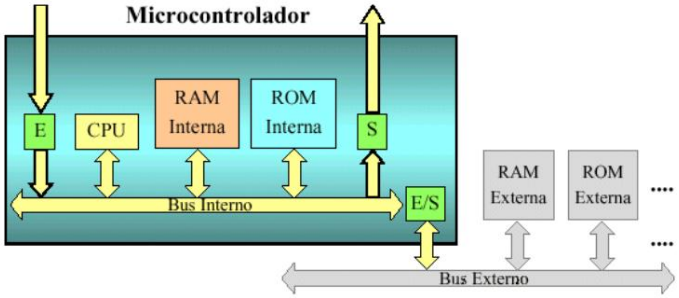

Figura 1.2 Esquema de un microcontrolador [7]

Modulo ESP32

El microcontrolador ESP32 fue lanzado en 2016, por la empresa multinacional *Espressif Systems*, para reemplazar a su versión anterior el ESP8266, con el objetivo de mejorar el desarrollo de dispositivos relacionados con IoT. El ESP32 es un módulo compacto con potentes características como Wi-Fi, Bluetooth, programable en diversos lenguajes y utiliza OTA (*Over the Air*) para actualizar el producto de forma segura [8], [9]. Es importante la capacidad de establecer comunicación con otro dispositivo dado que esta característica es esencial en la evolución de la tecnología IoT. Por otro lado, ESP32 también tiene aplicaciones enfocadas en el ámbito de la robótica y la medicina [8].

A continuación, en la Figura 1.3 se muestra la placa ESP32 WROOM 32.

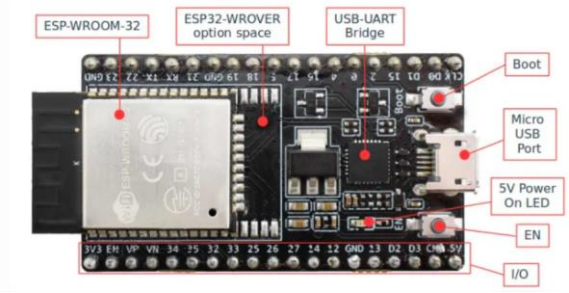

Figura 1.3 Placa de desarrollo ESP32 [10]

Arduino IDE (*Integrated Development Environment*)

Es una plataforma de código abierto que brinda servicios de *hardware* y *software* para el desarrollo de proyectos electrónicos. El objetivo de Arduino es darle al usuario una plataforma simple y económica, de esta forma es accesible para cualquier tipo de usuario. Por otro lado, las placas de Arduino permiten modificaciones según las implementaciones que el usuario quiera añadir. En la actualidad cuenta con una plataforma web que permite a las placas de Arduino y de terceros que se conecten a la nube con el objetivo de desarrollar dispositivos IoT [11].

El *software* de Arduino es compatible con varios sistemas operativos, el desarrollo del *software* es de código abierto y puede ser almacenado en *GitHub* [12], [13].

DFPlayer Mini MP3 Player

Este módulo es compatible con placas de desarrollo como ESP32, Arduino, Raspberry u otros microcontroladores con interfaz UART (*Universal Asynchronous Receiver/transmitter*). El módulo mp3 (Figura 1.4) tiene la capacidad de reproducir sonidos sin necesidad de conectarse a un microcontrolador externo. Sus características más importantes son: compatibilidad con sistemas de archivos como FAT (*File Allocation Table*)16/32, soportar una memoria máxima de 32 (Gb) en su ranura para microSD, funcionar en modo de control serie y modo de control E/S, volumen ajustable de 0 a 30 niveles y puede ser utilizado en varias aplicaciones como transmisión de voz GPS para automóviles, avisos de voz, indicadores con voz automatizados [14].

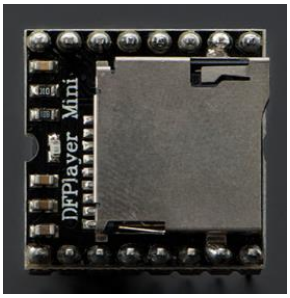

Figura 1.4 Modulo MP3 - *DFplayer Mini* [15]

Plataforma IoT

Una plataforma IoT es muy versátil ya que pueden ser utilizadas de forma básica, para almacenar datos y ofrecer interfaces estándares, o también pueden ser utilizadas en sistemas más complejos. El objetivo de una plataforma IoT es recoger los datos de los dispositivos conectados, también se encarga de proporcionar la interfaz en donde se visualizan y controlan los dispositivos conectados [16].

Existen varias plataformas IoT en el mercado como *Amazon Web Services IoT*, Arduino IoT, Azure IoT *Hub*, *Oracle* IoT *Cloud*, etc. Una plataforma IoT permite que los dispositivos se conecten mediante HTTPS, Websockets o MQTT (*Message Queing Telemetry Transport*). Por otro lado, las plataformas IoT tienen una arquitectura la cual debe tener la capacidad de recoger la información almacenarla, analizarla, exponer la información para que el usuario interactúe con esta y garantizar la seguridad de los datos del sistema [16].

En la Figura 1.5 se muestra la arquitectura que debe tener una plataforma IoT.

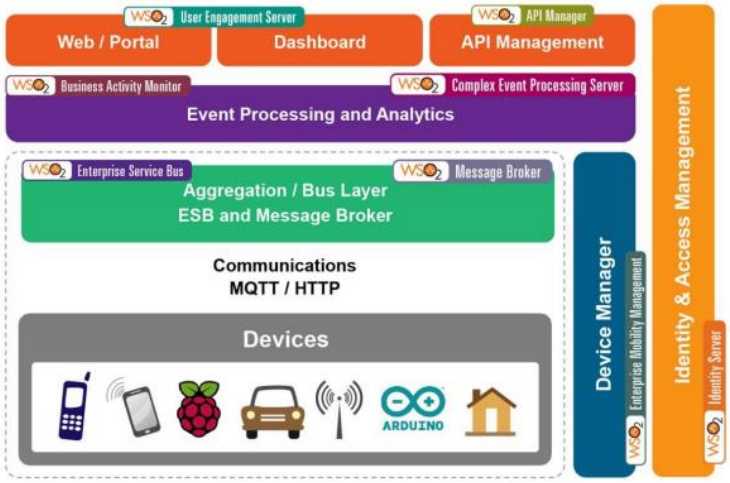

Figura 1.5 Arquitectura de una plataforma IoT [17]

MQTT

MQTT (Message Queuing Telemetry Transport) es un protocolo de comunicación ligero y eficiente diseñado para la transmisión de datos entre dispositivos en entornos con restricciones de ancho de banda y conectividad inestable. Este protocolo se basa en el modelo de publicación/suscripción, donde los dispositivos pueden publicar información en "temas" y suscribirse a temas para recibir actualizaciones. MQTT utiliza un enfoque de mensajería asincrónica, lo que permite una comunicación bidireccional eficiente y minimiza la sobrecarga de red. Además, MQTT proporciona características como la retención de mensajes y la calidad de servicio (QoS) para garantizar la entrega confiable de los mensajes, lo que lo convierte en una opción popular para aplicaciones de IoT, monitoreo remoto y sistemas de control. [18],

2 METODOLOGÍA

Para la implementación del prototipo se realizó un análisis para identificar los requerimientos necesarios para el óptimo funcionamiento del prototipo. Se definieron los componentes electrónicos necesarios para la implementación del circuito, también se analizó el tipo de comunicación más adecuado para cumplir las necesidades definidas en el alcance del proyecto.

Se realizó un análisis de las características del hardware y software para definir los componentes de mayor relevancia que se utilizaron en el prototipo como son: el microcontrolador, el módulo MP3, el lector RFID, la plataforma IoT. Para el análisis de las características se tomaron al menos tres dispositivos de cada componente para la comparativa, además se justificó el motivo por el cual se seleccionó el componente.

Para el diseño 3D de la placa y sus elementos se utilizó el *software Proteus.* Por otro lado, con el mismo *software* se realizó el diagrama esquemático de prototipo con todos sus elementos, también se realizó un diagrama de flujo del código de programación utilizado para el funcionamiento del prototipo.

Para el diseño exterior que contiene el circuito electrónico se utilizó madera en donde se acoplo la placa y sus componentes de forma en que se vea estéticamente bien y sea óptimo para el usuario. Adicionalmente se realizó un presupuesto de los costos involucrados en la implementación del prototipo.

En la fase final se realizaron diferentes pruebas para verificar el funcionamiento del prototipo, estas pruebas tuvieron la finalidad de comprobar el correcto funcionamiento

del lector RFID, la comunicación con la plataforma IoT y el funcionamiento de los botones reproducir/parar y aleatorio.

3 RESULTADOS

El prototipo para reproducción de canciones mediante tarjetas RFID y aplicación móvil, permite al usuario tener un reproductor de canciones interactivo, fácil de usar e ideal para personas mayores y niños. El uso de tarjetas RFID facilita la reproducción de música, dado que es similar a utilizar un CD o un disco de vinilo, también tiene botones que permiten reproducir en orden aleatorio las canciones almacenadas. El uso de la plataforma IoT permite que se pueda acceder al contenido de la memoria SD de forma remota, de esta forma llega a usuarios que estén más familiarizados con la tecnología.

3.1 Estudio de requerimientos para la implementación del prototipo

El prototipo debe permitir que los usuarios finales puedan reproducir contenido de audio mediante una tarjeta RFID la cual estará asociada mediante una etiqueta a la canción correspondiente, el dispositivo tiene como objetivo menorar la brecha tecnología de los usuarios que no tengan mucha habilidad en el uso de dispositivos y aplicaciones de reproducción de música o contenido auditivo. Las funciones principales del prototipo son la reproducción de contenido MP3 por medio de tarjetas RFID y mediante una aplicación. Para la implementación del prototipo se necesita que se cumplan los siguientes requerimientos:

- Se necesita un microcontrolador con conexión inalámbrica para facilitar la comunicación entre dispositivos.
- Se utilizará un lector RFID, el cual se comunica con señales de radiofrecuencia con el objetivo de transmitir la información de la tarjeta RFID e identificada el valor asociado a la canción para iniciar la reproducción de esta.
- El prototipo debe utilizar una plataforma IoT donde el usuario por medio de una aplicación móvil va a poder seleccionar y reproducir la canción deseada.
- Se debe utilizar un módulo MP3 para leer el archivo de tipo mp3 y transmitir el audio por medio de altavoces.
- Se debe utilizar un altavoz para que el sonido sea escuchado.

En la Figura 3.1 se muestra un esquema de los requerimientos.

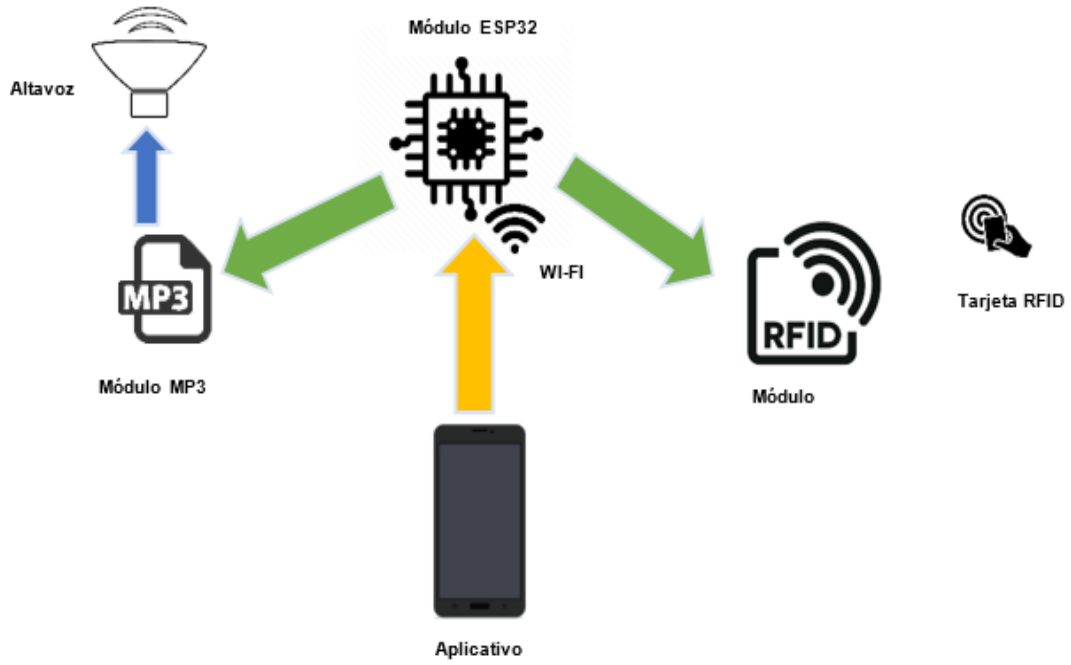

Figura 3.1 Esquema de los requerimientos

La integración de los componentes detallados anteriormente busca el funcionamiento óptimo para que la interacción del dispositivo sea eficiente para el usuario, la comunicación con el aplicativo brinda una mejora para que se pueda añadir más funciones al prototipo y de esta forma el producto sea escalable.

3.2 Definición de los componentes de hardware y software

Con los requerimientos ya especificados en la sección previa se define cada componente, tomando en cuenta las características generales para solventar las necesidades del prototipo.

Microcontrolador con comunicación inalámbrica

Para la implementación del prototipo sé analizaron varios microcontroladores que permitan comunicación inalámbrica como: Arduino Mkr Wi-Fi 1010, Netduino, ESP8266 y ESP32; en la Tabla 3.1 se realiza una comparación de los elementos mencionados.

Tabla 3.1 Comparativa de módulos controladores con tarjeta inalámbrica[19], [20], [21].

Especificación	Arduino Mkr Wi-Fi 1010	Netduino	ESP8266	ESP32 WROOM
Procesador CPU	ARM Cortex-M0+ SAMD21 32 (bits)	ARM Cortex-M4 32 (bits)	RISC CPU Xtensa LX106 32 (bits)	Dual Core LX6 32(bits)
Voltaje de alimentación	3.6 - 6 (V)	3.3 - 5 (V)	5 (V)	5 (V)
Pines de entrada y salida digital	28	12	16	24
Pines de entrada analógicos	9	8	1	18
Puertos	Micro USB	Micro USB y USB	Micro USB	Puerto Micro USB
Dimensiones	63x18 (mm)	25x62 (mm)	49x26 (mm)	52x28 (mm)
Estándar de comunicación inalámbrica	802.15	802.11b/g/n	802.11 b/g/n	802.11 b/g/n
Temperatura de operación	0 – 70 (°C)	-40 a 125 (°C)	-40 a 125 (°C)	-40 a 80 (°C)
Protocolos de red	Ethernet	IPv4 /TCP/IP	IPv4 /TCP/IP	IPv4/TCP/UDP/HTTP
Consumo de corriente	97.5 (mA)	150 (mA)	70 (mA)	70 a 250 (mA)

El Microcontrolador ESP32 WROOM tiene un módulo WI-FI incorporado en la placa de desarrollo, esta característica es necesaria en la implementación del prototipo ya que el dispositivo tiene que conectarse a una plataforma IoT. Adicionalmente, tiene las condiciones ideales para la distribución de energía para los módulos que conforman el prototipo.

El microcontrolador ESP8266 tiene características similares al microcontrolador ESP32, sin embargo, existen características que hacen que el ESP8266 sea inferior. La placa ESP32 es superior porque posee pines DAC los cuales sirven para mejorar el sonido, estas entradas realizan la conversión de analógico a digital, esto es útil para el prototipo porque permite una salida de audio adicional. Otra característica es la velocidad de transmisión a través de una red WLAN en el caso de la placa ESP8266 es de 72.2 (Mbps), en comparación con la velocidad de 150 (Mbps) que alcanza el módulo ESP32.

Por lo tanto, se llegó a la conclusión que la placa de desarrollo ESP8226 fuese descartada.

Aunque cumple con todos los requerimientos la placa Netduino-3 resulta económicamente no viable dado que en el mercado cuesta 69.95 (USD) en comparación a el costo del módulo ESP32 que es de 13.65 (USD). Por otra parte, la capacidad de procesamiento de Netduino es menor que el de la placa ESP32 analizando los argumentos presentados se llegó a la conclusión de no utilizar la placa Netduino para el desarrollo del prototipo.

La placa Arduino MKR Wi-Fi 1010 resulta económicamente no viable dado que en el mercado cuesta 48.5 (USD) en comparación a el costo del módulo ESP32 que es de 13.65 (USD). Por otra parte, el dispositivo no se vende en ecuador, el Arduino MKR Wi-Fi 1010 tiene el mismo modulo Wi-Fi del microcontrolador ESP32 por lo que es más viable el módulo ESP32 que es más económico y se encuentra en el mercado ecuatoriano.

En conclusión, el módulo ESP32 es el dispositivo que cumple con las especificaciones de alimentación correctas para repartir energía a los otros módulos electrónicos que se utilizan en la implementación del prototipo después de haber realizado el análisis con respecto a otros dispositivos se decido utilizar el módulo ESP32 para la implementación del prototipo.

Módulo MP3

Para la elección de un módulo MP3 se hizo la comparativa entre tres diferentes tipos de módulos MP3. De acuerdo con las características y la compatibilidad que tenga, se eligió uno para llevar a cabo la implementación del prototipo.

A continuación, en la Tabla 3.2 se muestran las características técnicas de los módulos MP3 analizados.

Tabla 3.2 Comparación de Módulos MP3[15], [22], [23] .

Especificaciones	Módulo MP3 *DFPlayer Mini*	Módulo MP3 JQ6500	Módulo MP3 VS1053
Voltaje de operación	Entre 3.3 a 5 (V) (DC)	Entre 3.3 a 5 (V) (DC)	Entre 3.3 a 5 (V) (DC)
Interfaz de comunicación	UART (Serial) a 9600 (bps)	UART (Serial) a 9600 (bps)	SPI, UART

Almacenamiento	microSD 32 (GB)	Módulo microSD 32 (GB) externo	microSD 32 (GB)
Formatos de audio compatibles	MP3, WAV	MP3, WAV	MP3, WMA, MIDI, Ogg Vorbis, AAC, AAC+, FLAC, WAV,
Tamaño	27 (mm) x 21 (mm) x 4 (mm)	27 (mm) x 30 (mm)	25 (mm) x 30 (mm)
Control de volumen:	Ajuste de avance/retroceso	Ajuste de avance/retroceso	Ajuste de avance/retroceso
Consumo de corriente	>20 (mA)	20 (mA)	40 (mA)

El módulo MP3 JQ6500 se descartó, debido a que no incorpora el lector de tarjeta microSD necesario para la reproducción de los archivos MP3 que se utilizan en el funcionamiento del prototipo.

Dado que los módulos VS1053 y DFPlayer mini tienen características similares, entonces los parámetros que se toman en cuenta para elegir el módulo MP3 son su tamaño, corriente y costo. Tomando en cuenta estos parámetros se descartó el módulo MP3 VS1053 porque la corriente que consume es de 40 (mA), que en comparación al módulo MP3 *DFPlayer Mini* que es de 20 (mA), el consumo de corriente es muy alto. Por otro lado, el costo del módulo MP3 VS1053 se encuentra entre 12 y 15 (USD) dado que este módulo reproduce archivos WMA, MIDI, *Ogg Vorbis*, AAC, AAC+, FLAC, MP3 y WAV y su valor es mayor que el costo del módulo MP3 *DFPlayer.*

Por lo tanto, el módulo *MP3 DFPlayer mini* cumple con los requerimientos necesarios, para la implementación del prototipo, lo que garantiza el funcionamiento óptimo del circuito.

Lector RFID

Para la elección del lector RFID se hizo la comparación de tres modelos de lectores que cumplan con la necesidad del prototipo y que posean las características de compatibilidad necesarias para que funcione el circuito de forma óptima; en la Tabla 3.3 se muestra la información respecto a los lectores RFID comparados.

Tabla 3.3 Comparación de Módulos RFID [24], [25], [26].

Especificaciones	RDM6300	PN532	RC522
Frecuencia de operación	125 (KHz)	13.56 (MHz)	13.56 (MHz)
Alimentación	5 (V) (DC)	2.7 a 5.4(V) (DC)	2.5 (V) (DC) a 3.3 (V) (DC)
Protocolo de comunicación	Wiegand 26/34 bits	ISO/IEC 14443A/B, MIFARE, FeliCa	SPI (Serial Peripheral Interface)
Dimensiones	38 (mm) x 18 (mm)	45 (mm) x 35 (mm)	39 (mm) x 59.5 (mm)
Interfaces de comunicación	Wiegand	UART, SPI, e I2C	SPI(Serial)
Consumo de corriente	20 a 50 (mA)	100 a 120 (mA)	13 a 26 (mA)
Modos de funcionamiento	Lectura/ escritura	Lectura/escritura, emulación de tarjeta (P2P) emulación de etiqueta (HCE)	Lectura/escritura
Distancia de lectura	Entre 20 y 50 (mm)	Entre 5 y 20 (mm)	50 (mm)

El módulo RFID RDM6300 opera en una frecuencia de 125 (kHz). Por lo tanto, es necesario utilizar tarjetas de lectura que estén dentro de este rango de frecuencia. Sin embargo, este tipo de tarjetas no son tan comunes en el mercado de dispositivos y componentes electrónicos. Por otro lado, es importante mencionar que el módulo RDM6300 no es compatible con la tecnología MIFARE.

Al analizar el lector RFID PN532 satisface todos los requisitos establecidos, además que es compatible con tecnologías como NFC. Sin embargo, es importante considerar que, debido a las características extras que no se alinean con las necesidades del prototipo, su costo aumenta. Este resulta en la decisión de descartar este dispositivo en la selección final.

El lector RFID RC522 se ajusta a los requisitos necesarios para la implementación del circuito. Esto se debe a que su consumo de energía es apropiado para operar de manera

armoniosa junto con los demás componentes electrónicos que integran el prototipo; en la Figura 3.2 se muestra el lector RFID RC522.

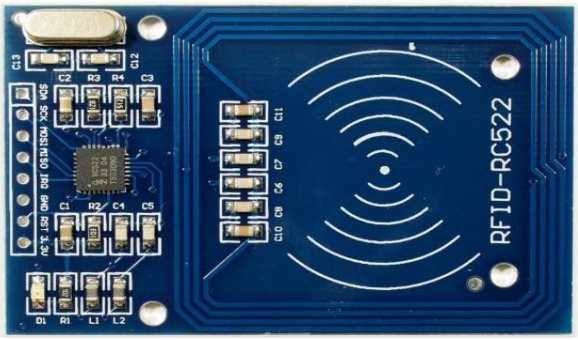

Figura 3.2 Modulo RFID [27]

Plataforma IoT para interactuar con el prototipo

Para elegir la plataforma IoT se recabo información de tres plataformas disponibles en Internet:

Tabla 3.4 Comparación Plataformas IoT [28], [29], [30]

Especificaciones	Arduino Cloud	Blynk	AWS IoT Core
Conectividad	MQTT, HTTP	MQTT, HTTP	MQTT, HTTP, *WebSockets*
Tipo de plan	Gratuito, y de pago	Gratuito y de pago	Gratuito por 12 meses
Duración de datos en la nube	1 día de retención de datos	1 semana de retención de datos	Pago por uso
Widgets e indicadores	básicos	Básicos	De acuerdo con el plan

Después de analizar los datos de la Tabla 3.4 se concluye escoger la plataforma *Arduino Cloud*, tomando en cuenta su compatibilidad con el módulo ESP32. Por otra parte, la plataforma permite crear objetos que se vinculen a dispositivo y controlarlos por medio de una aplicación móvil utilizado MQTT o HTTP, de esta forma se tiene acceso al prototipo de forma remota desde un dispositivo móvil.

3.3 Diseño del prototipo

Comunicación entre microcontrolador y módulo RFID

El microcontrolador ESP32 maneja diferentes formas para comunicarse con módulos complementarios como: UART, I2C, SPI y CAN. Para comunicar el microcontrolador y el módulo RFID se utiliza comunicación SPI (*Serial Peripheral Interface*); este protocolo permite operar a los dispositivos como maestro o esclavo y es utilizado para comunicar sensores, módulos entre otros dispositivos.

El módulo RFID es compatible con el protocolo SPI de esta forma la comunicación entre el microcontrolador y el módulo es síncrona y full dúplex. Para optimizar la transmisión y recepción de la información entre los dispositivos, el enlace entre el microntrolador ESP32 y el módulo RFID necesita de 4 líneas de conexión que son:

- **SDA:** (*Serial Data Input / Output*) transmite y recibe datos entre microcontrolador y módulo RFID
- **SCK:** (*Serial Clock*) sincroniza la transmisión de datos entre los dos dispositivos
- **MOSI:** (*Master Output Slave Input*) lleva los datos desde el microcontrolador al módulo RFID
- **MISO:** (*Master Input Slave Output*) lleva los datos desde el módulo RFID al microcontrolador

La conexión con el microcontrolador ESP32 se debe hacer de acuerdo con el *PinOut* del microcontrolador, en este diagrama se encuentra los pines que corresponden a las líneas de salida del lector RFID de esta manera los dos dispositivos tendrán una comunicación correcta. A continuación, se muestra en la Figura 3.3 el *PinOut* del microcontrolador ESP32 y en la Figura 3.4 el *PinOut* del módulo RFID.

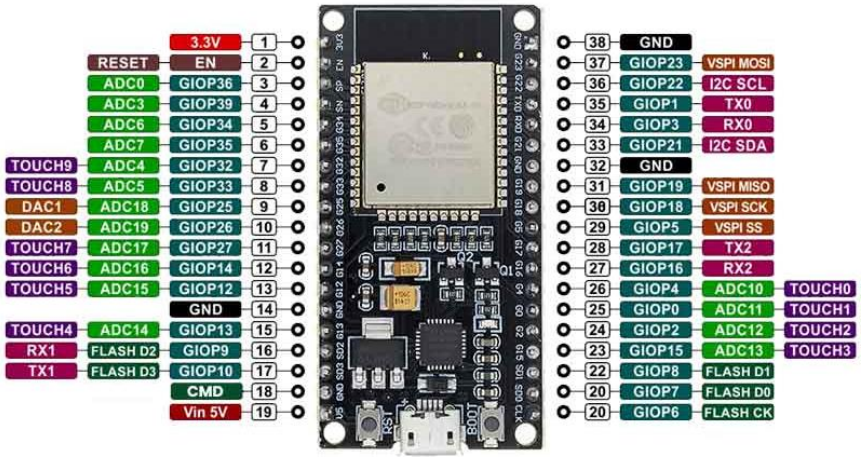

Figura 3.3 *PinOut ESP32* [31]

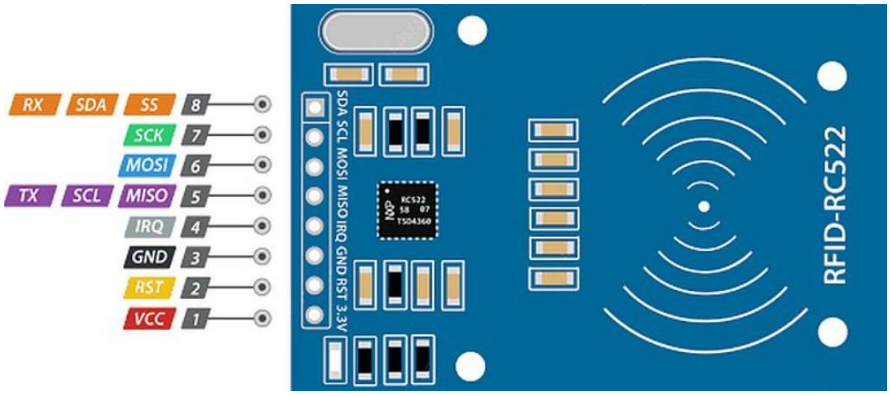

Figura 3.4 *PinOut RFID RC 522* [32]

Comunicación entre el microcontrolador y el módulo MP3

El módulo MP3 *DFPlayer mini* se comunica con el microcontrolador ESP32 sólo a través de la interfaz UART. Esta interfaz (UART) permite la transmisión de datos mediante una conexión serie, donde el microcontrolador envía comandos específicos a través de la línea de transmisión (TX) para controlar las funciones del módulo MP3. A su vez, el módulo MP3 responderá y enviará datos al microcontrolador a través de la línea de recepción (RX), que estará conectada a las líneas de recepción asignadas en el microcontrolador ESP32. De esta manera, el microcontrolador y el módulo MP3 se comunican de manera bidireccional, permitiendo el control y la interacción con las funciones del reproductor de audio MP3 [33], [34].

Adicionalmente, se debe tomar en cuenta que la comunicación entre el microcontrolador y el módulo MP3 necesita una resistencia para reducir el voltaje que se transmite del módulo ESP32 al módulo *DF Player mini* con la finalidad de evitar un sobrevoltaje y daños en el módulo MP3. Por otro lado, el fabricante sugiere utilizar una resistencia de 1 kΩ entre los pines TX y RX del módulo ESP32 y el módulo *DFPlayer Mini*. Esta recomendación se basa en la configuración PULL-UP que establece un voltaje constante entre los dos canales. Debido a que estos canales no están directamente conectados a la fuente de alimentación, mantener el nivel alto garantiza una comunicación activa y reduce el ruido en la transmisión de información. La Figura 3.5 se muestra el *PinOut* del módulo MP3[15].

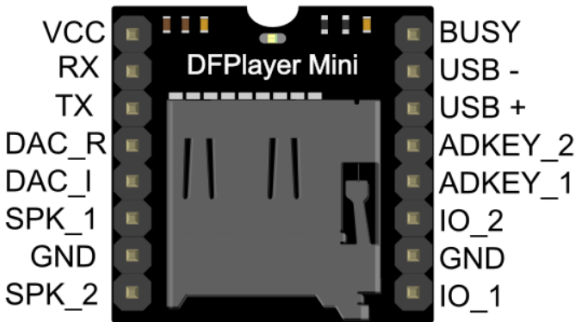

Figura 3.5 *PinOut DFPLAYER mini*.

Adicional para que el audio mejore se conectó el módulo MP3 a un módulo amplificador Pam 8403, esto se realizó debido a que el audio que emitían los parlantes conectados directamente al módulo MP3 era muy bajo implementando esta solución se obtuvo un mejor resultado en el producto final en la Figura 3.6 se muestra el módulo amplificador utilizado.

Figura 3.6 Amplificador PAM8403

Placa electrónica

Antes de implementar el circuito, se creó un diagrama esquemático que representa las conexiones del prototipo, con el propósito de establecer un bosquejo del circuito a implementar. Para llevar a cabo esta tarea, se empleó el software Proteus, el cual permite tanto el diseño como la simulación de circuitos electrónicos. A continuación, en la Figura 3.7 se muestra el diagrama de conexión entre microntrolador ESP32 y el módulo MP3 y el amplificador PAM8403, añadido para mejorar la calidad de audio que emite el módulo MP3.

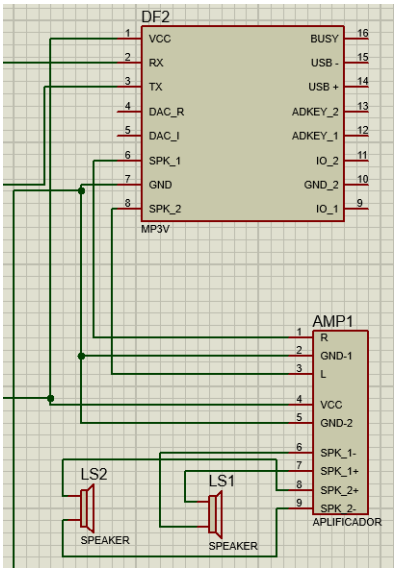

Figura 3.7 Conexión *DFPlayer mini* con amplificador.

De la misma manera se realizó un diagrama esquemático en el *software* "*Proteus",* para establecer como se conecta el microcontrolador con el módulo RFID (Figura 3.8).

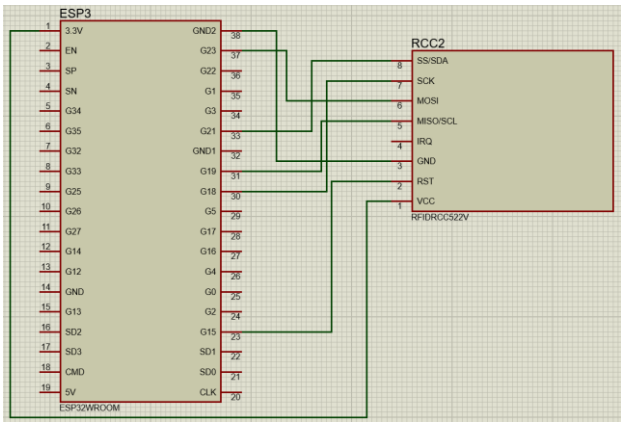

Figura 3.8 Conexión entre el módulo ESP32 y el módulo RFID

Para finalizar la ilustración del diagrama esquemático, en la Figura 3.9 se presenta el diagrama general del prototipo, en este punto se añaden los botones que tienen la funcionalidad de reproducir y parar las canciones y reproducir de forma aleatoria. En este punto se añade la configuración *PULL DOWN* para mantener el estado alto del botón; también se añade las conexiones con el parlante. La alimentación del prototipo se lleva a cabo mediante el puerto micro-USB del módulo ESP32 y dado que este dispositivo no requiere movilidad, no es necesario el uso de una batería para su funcionamiento.

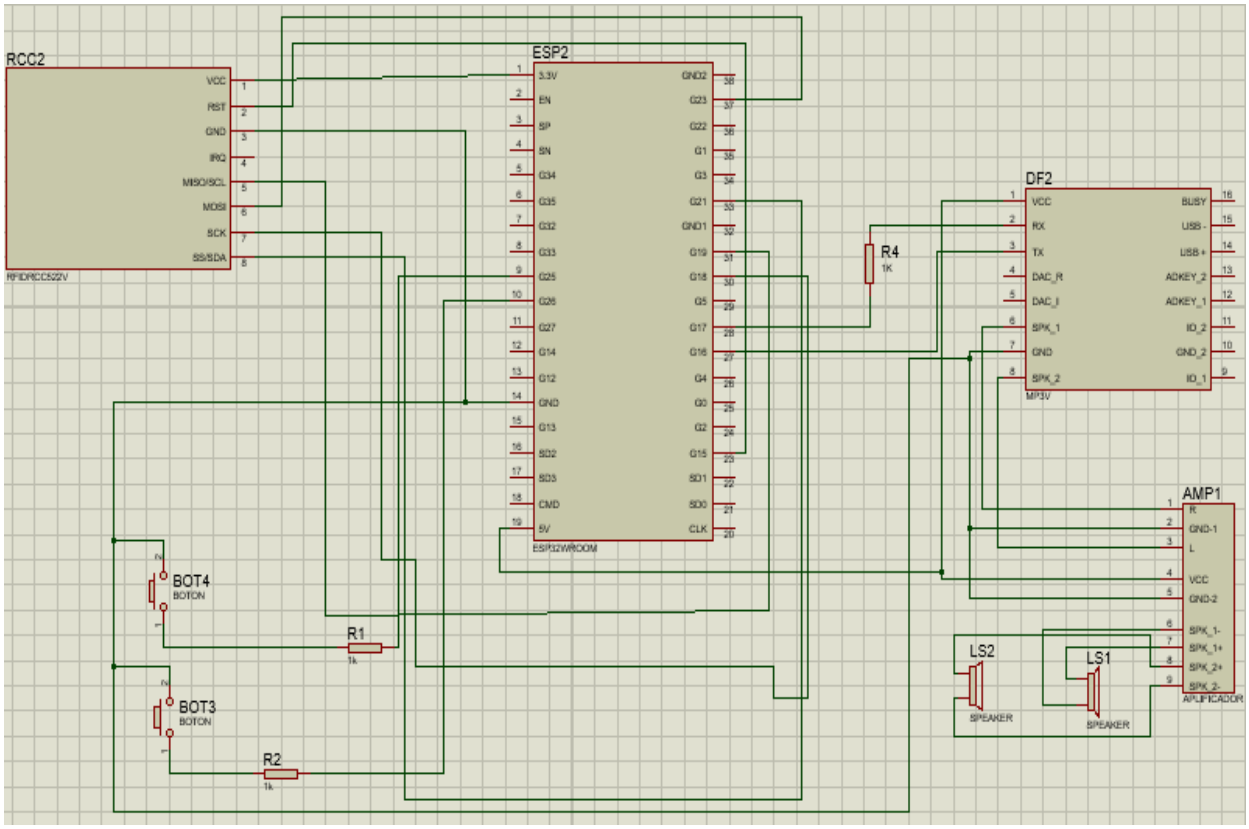

Figura 3.9 Diagrama general del prototipo

Basado en el diagrama general se realizó la modelación 3D del circuito y se crearon las pistas para interconectar los elementos que conforman el circuito en una PCB (placa de circuito impreso).

El proceso para crear las pistas en el PCB con “proteus” empieza con ingresar a la opción PCB *layout*, donde se utiliza la herramienta de "Pista" para conectar pines de componentes trazando rutas y ajustando el ancho de las pistas para finalizar se accede a la opción *autorute,* luego abre una ventana donde se debe elegir la opción de iniciar ruteo automático para generar automáticamente las pistas del circuito. Para agregar modelos 3D, se deben crear archivos de modelos 3D compatibles de formato (STL), luego en el modo de diseño de PCB, selecciona el componente, habilita la "Visualización 3D" en las propiedades del componente y carga el archivo del modelo 3D se genera la imagen 3D y se ajusta de acuerdo a el espacio de la placa esto se muestra en la Figura *3.10* y en la Figura *3.14*.

El diagrama permite visualizar la interconexión de los componentes, de esta forma se facilita la comprensión de como interactúan los elementos con el microcontrolador.

Diseño 3D del prototipo

Para crear el modelo 3D del circuito se utilizó *Proteus,* el software no tiene por defecto los diseños de los módulos que se necesitan para el prototipo por lo que es necesario crearlos desde cero, para esto se siguen los siguientes pasos:

- Se crea el componente en la sección *"Schematic Capture"*. Mediante herramientas de dibujo, se asignan y etiquetan los pines del módulo siguiendo su disposición en el *PinOut.*
- Luego, en la sección *"PCB Layout"*, se diseña el encapsulado. Aquí, se asocian los pines de acuerdo con el diseño previo.
- Finalmente, el componente se añade a la librería *"devices-user"*, completando el proceso.

Este proceso se repite para cada módulo, es importante tener en cuenta las medidas de los componentes, dado que a partir del diseño se crearán las pistas de interconexión de los componentes del circuito, al finalizar el proceso el resultado del modelo 3D se presenta en la Figura 3.10.

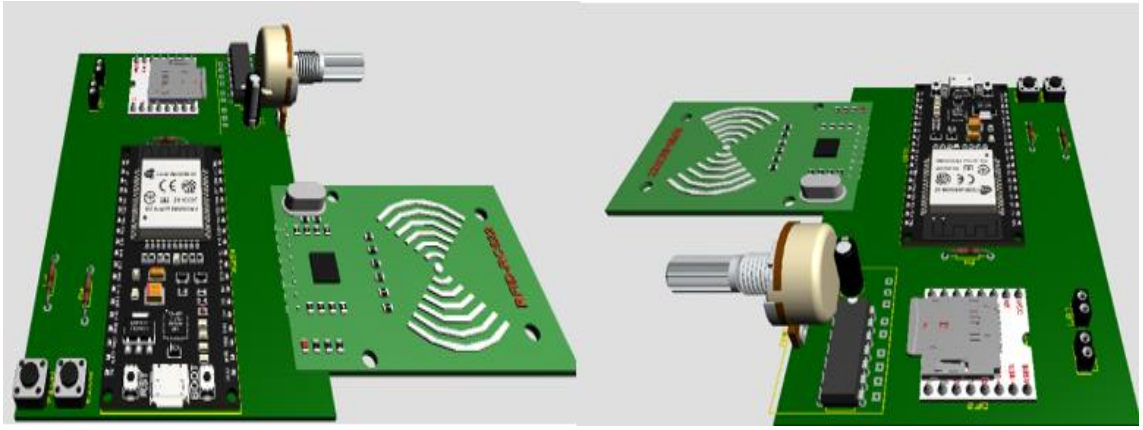

Figura 3.10 Diseño 3D

Por otro lado, se incluye el diseño del protector del prototipo (Figura 3.11), para realizar este proceso se utilizó la herramienta de diseño 3D que ofrece la plataforma *TinkerCAD.*

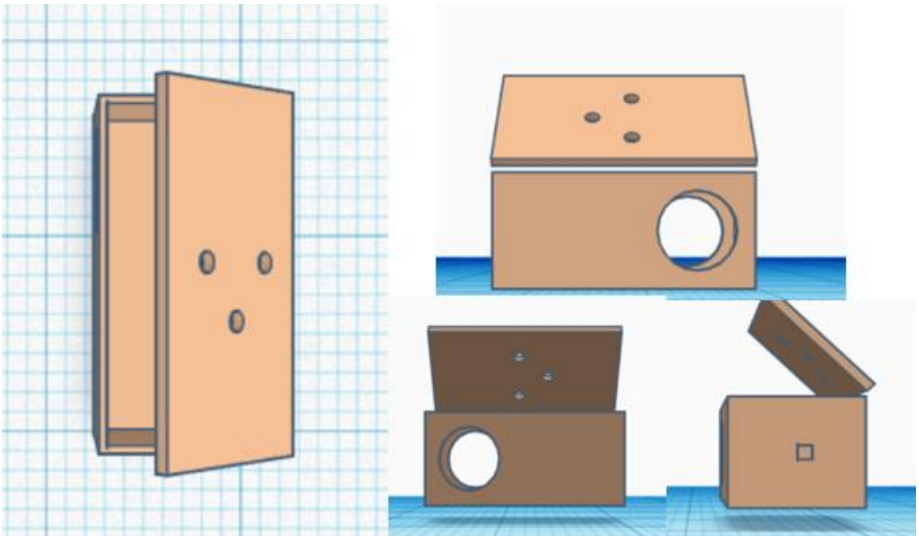

Figura 3.11 Diseño de la caja protectora del prototipo

Diagrama de flujo

Con el diagrama de flujo de la Figura 3.12 se explica de manera sencilla el desarrollo del código que se utiliza para controlar el módulo *DFPlayer Mini* y el módulo MFRC522 usando un microcontrolador ESP32. Para el desarrollo se siguieron los siguientes pasos:

- Primero es necesario implementar las bibliotecas dentro del gestor de código Arduino IDE, las bibliotecas necesarias para el desarrollo de la primera parte del proyecto necesita las siguientes bibliotecas: SPI, MFRC522, *SoftwareSerial* y *DFRobotDFPlayerMini.* Estas bibliotecas proporcionan funciones para comunicarse con los módulos y realizar diversas operaciones.
- A continuación, se debe definir los pines que serán utilizados para la comunicación con los módulos. En este caso, se establecen los pines RST_PIN y SS_PIN para el módulo MFRC522 y los pines 16 y 17 para la comunicación de *SoftwareSerial* con el módulo *DFPlayer Mini.*
- Para que las funciones del código se puedan ejecutar en es necesario crear instancias de las clases MFRC522 y *SoftwareSerial* para el lector RFID y el reproductor *DFPlayer Mini*, respectivamente. También se define una instancia de *DFRobotDFPlayerMini,* también se declaran lo pines para los botones de reproducción y pausa, reproducción aleatoria y se definen variables para el control del volumen y el estado de reproducción.
- En la función *setup* se inicializan las comunicaciones, pines y dispositivos. También se establece la conexión con el reproductor *DFPlayer Mini.*
- Por otra parte, en la función *loop* comienza el bucle principal del programa, que se ejecuta repetidamente, se controlan los botones de reproducción de forma

que, si se presiona el botón *playPause*, el reproductor se pausa o reanuda y si se presiona el botón reproducción automática, se activa la reproducción aleatoria.

- También se verifica por medio de una función que forma parte de la biblioteca MFRC552 si hay una tarjeta RFID nueva presente en el lector MFRC522. Si es así, se autentica y lee el número de la tarjeta luego se reproduce la canción correspondiente al número de tarjeta leído en el reproductor *DFPlayer Mini.*
- Al finalizar la lectura de la tarjeta RFID y realizar la acción correspondiente, se imprime un mensaje de finalización y se realizan las operaciones necesarias para detener la comunicación con la tarjeta, este ciclo se repite mientras el dispositivo este activo.

El código busca que la funcionalidad del lector RFID y el reproductor MP3 sea desarrollado en un solo programa, de esta manera se puede controlar la reproducción de canciones utilizando tarjetas RFID específicas Figura 3.12.

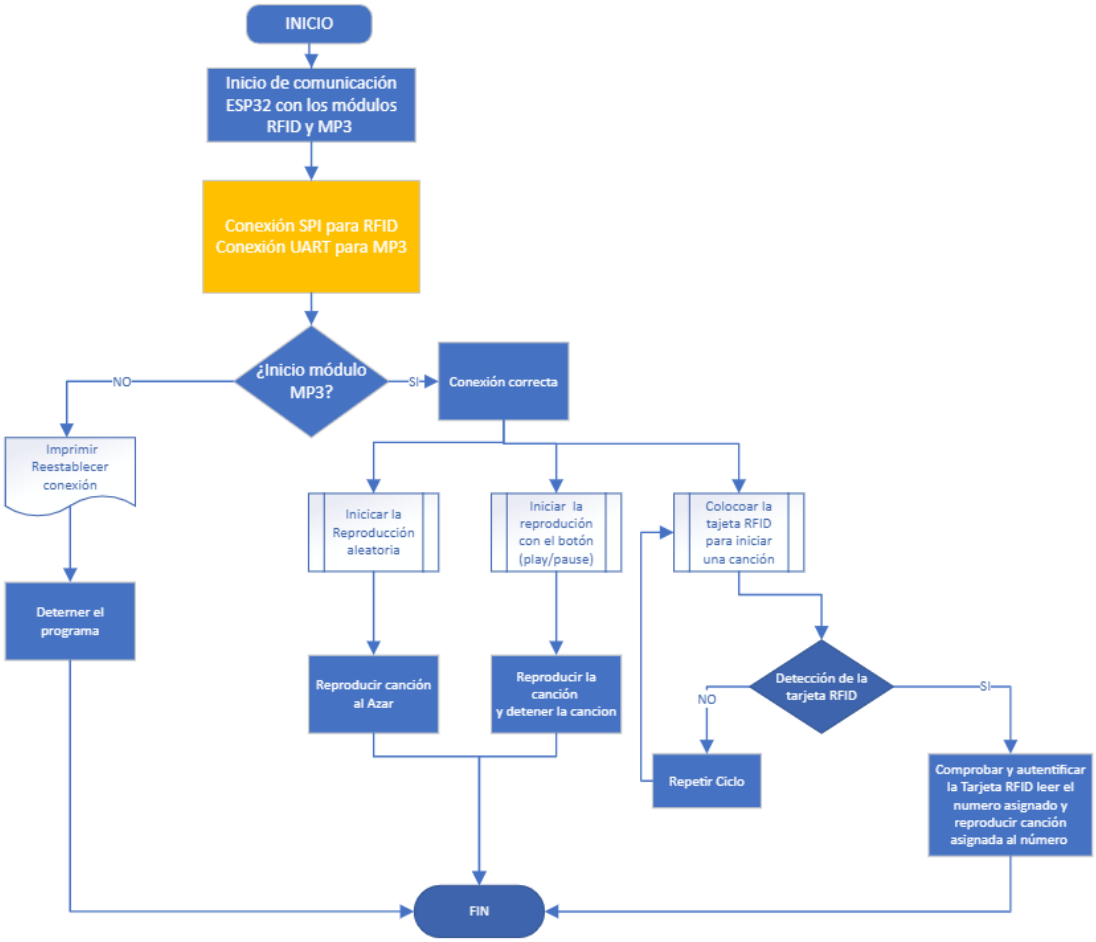

Figura 3.12 Diagrama de flujo conexión y funcionamiento entre módulos RFID, MP3 y ESP32

De manera similar, se elaboró el diagrama de flujo donde se describe la conexión del dispositivo con la plataforma de *Arduino Cloud*; el diagrama se presenta en la Figura 3.13. Además, se suministra una descripción del procedimiento, estructurado en los siguientes pasos consecutivos:

- Para iniciar el proceso, es necesario crear una cuenta en la plataforma *Arduino Cloud*. Una vez dentro de la plataforma, se procede a vincular el dispositivo ESP32, estableciendo la conexión con la red *Wi-Fi*. En la plataforma se configuran las variables esenciales para el funcionamiento de los *widgets*, los cuales permitirán el control del dispositivo tanto a través de una aplicación móvil como de la interfaz web. Asimismo, se crean los *widgets* que serán responsables de gestionar las diversas acciones ejecutadas por el prototipo.
- El siguiente paso es crear las funciones de acuerdo con las variables creadas en el paso anterior. Para el prototipo, se crearon dos variables distintas: la primera, es utilizada para iniciar o detener la reproducción del sonido, y la segunda, habilitando la selección y reproducción de una canción almacenada en el dispositivo.
- Las líneas de código que se ejecutan con la nube necesariamente dependen de la biblioteca *"thingProperties.h";* en esta biblioteca se ejecuta el código que permite la conexión con la red y con la plataforma también almacena las variables que se crean en la plataforma y las enlaza al código que se ejecuta en el prototipo.
- Para finalizar se ejecuta el código y el dispositivo se conecta a la red y puede seguir las instrucciones que da el usuario desde la plataforma web o la aplicación web.

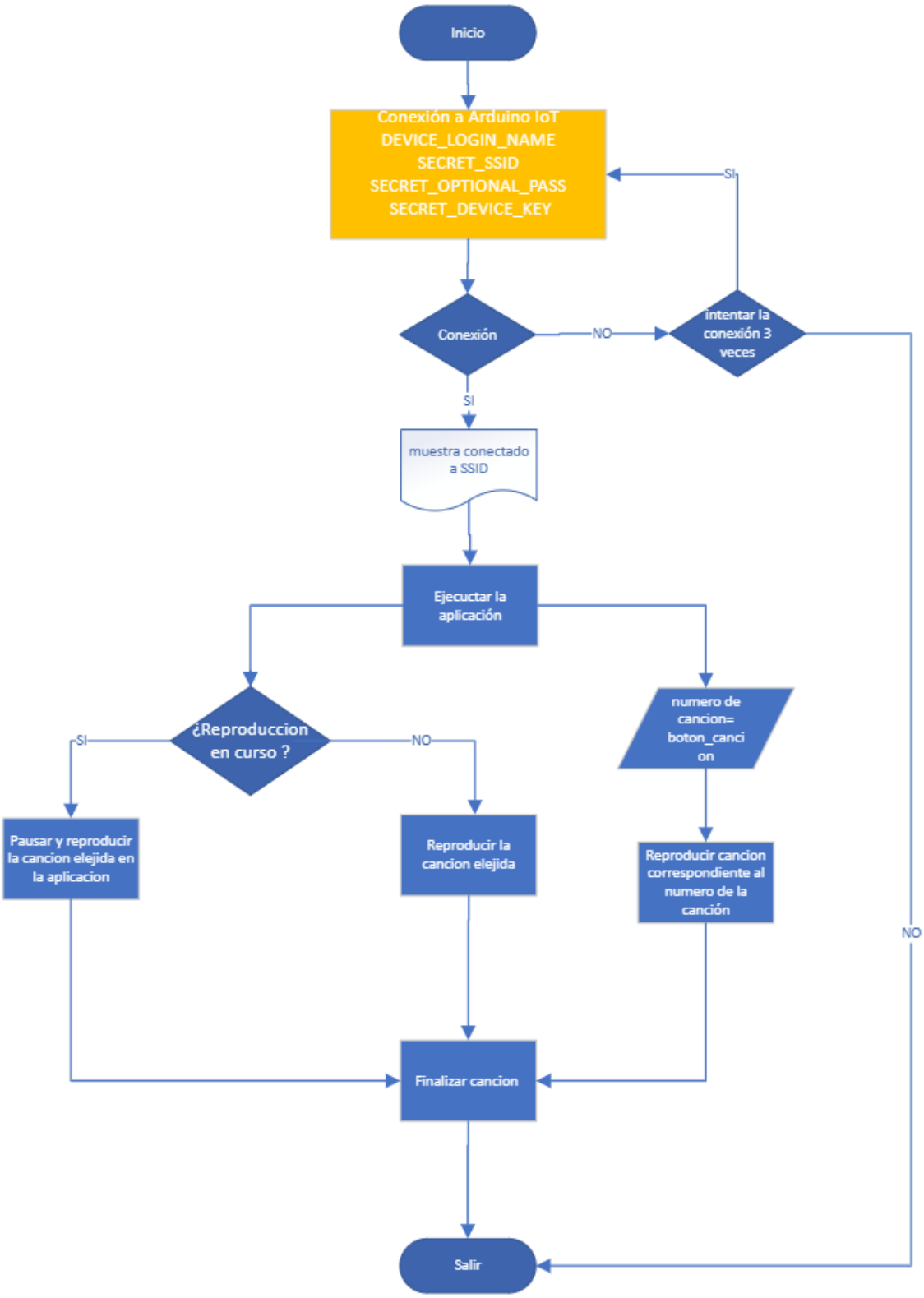

Figura 3.13 Diagrama de flujo conexión a la nube

3.4 Implementación del prototipo

Circuito del prototipo

Para crear la placa del prototipo se realizó el diseño de las pistas utilizando el software *Proteus*; en la Figura 3.14 se puede observar las pistas trazadas y los módulos empleados para el funcionamiento del prototipo.

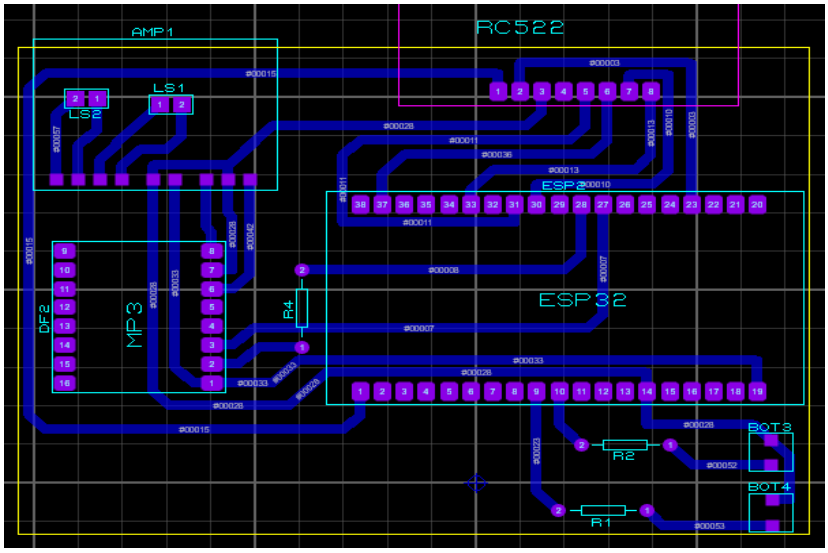

Figura 3.14 Diagrama de pistas.

Durante la fabricación del circuito, se empleó el método de planchado, que implica la impresión del diseño de pistas en papel fotográfico utilizando tecnología láser. Este proceso resulta fundamental, ya que la tinta de la impresión actúa como una capa protectora sobre el cobre de la baquelita para formar las pistas. Después de imprimir, se procedió a calentar el papel sobre la baquelita utilizando una plancha durante unos 40 minutos. Al retirar el papel, el diseño se transfiere a la baquelita. A continuación, se emplea ácido cloruro férrico para corroer el exceso de cobre de la baquelita, este proceso dura aproximadamente 40 minutos. Al finalizar este paso, solo quedaba cobre en el lugar de la impresión de las pistas. El siguiente paso consistió en eliminar la tinta protectora utilizando alcohol isopropílico. Luego, se perforaron agujeros en la baquelita en las ubicaciones donde se colocan los módulos. Finalmente, para asegurar un acoplamiento efectivo de los módulos con la placa, se soldaron regletas en los orificios de los pines, mientras que los componentes extras se soldaron directamente a la placa. A continuación, en la Figura 3.15 se muestra la placa PBC finalizada

Figura 3.15 Placa del circuito del Prototipo

Construcción de la caja protectora del prototipo

La construcción del prototipo se basó en el diseño 3D ilustrado en la Figura 3.11, utilizando madera MDF como material principal. Para lograr un acabado profesional del protector, se realizaron los cortes y orificios a través de corte láser. Luego se unieron las partes utilizando goma blanca.

Una vez completo el protector, se instaló el circuito en el interior de la caja. En esta etapa se ajustaron los módulos y componentes adicionales, de forma en que la usabilidad del dispositivo sea optima. Para finalizar se muestra en la Figura 3.16 el resultado final de la construcción del protector del prototipo.

Figura 3.16 Caja protectora del Prototipo

Configuración para el funcionamiento de la plataforma IoT

Para empezar la configuración dentro de la plataforma, es necesario tener una cuenta registrada en la plataforma *Arduino Cloud*, la plataforma permite ingresar con una cuenta Gmail, GitHub, Facebook y Apple, facilitando la creación de la cuenta.

Dentro de la plataforma en la pantalla inicial se debe elegir la opción *Things*, para asociarla con el dispositivo (microcontrolador ESP32). En la Figura 3.17 se muestra la pantalla en donde la plataforma presenta la opción de asociar un dispositivo. Sin embargo, para escoger el dispositivo ESP32 es necesario elegir dispositivos de terceros.

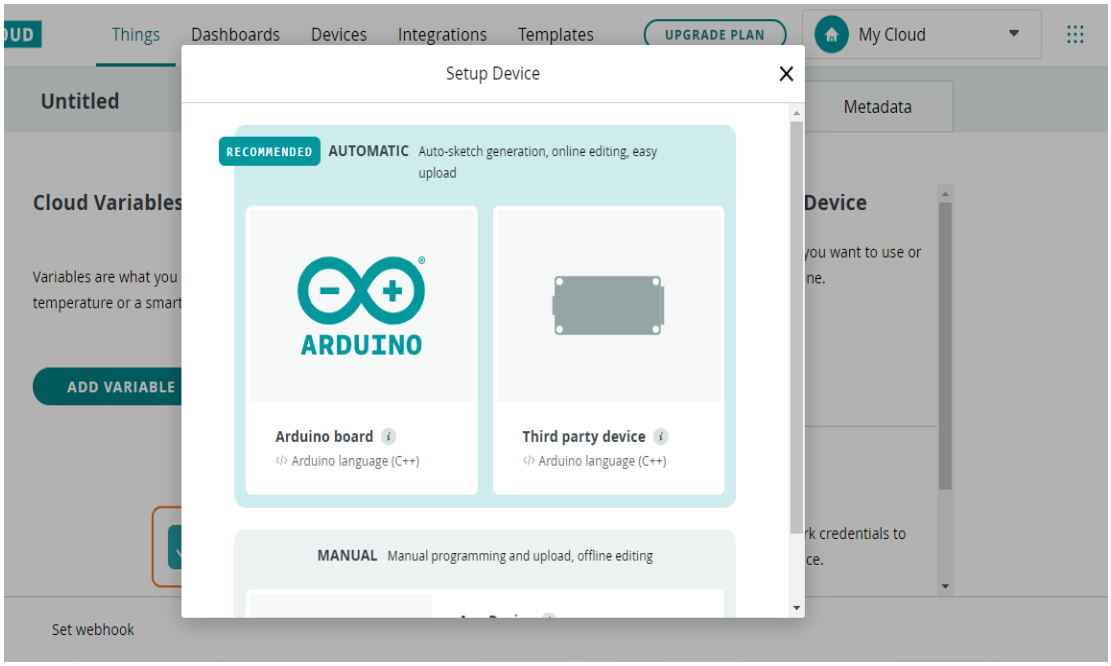

Figura 3.17 Crear un dispositivo dentro de la plataforma IoT

A continuación, se muestra el apartado para seleccionar el dispositivo, en esta ventana la plataforma presenta varias opciones para elegir, en este caso se elige el dispositivo que se utiliza en el prototipo y el modelo. En la Figura 3.18 se muestra la selección de acuerdo con las necesidades del prototipo.

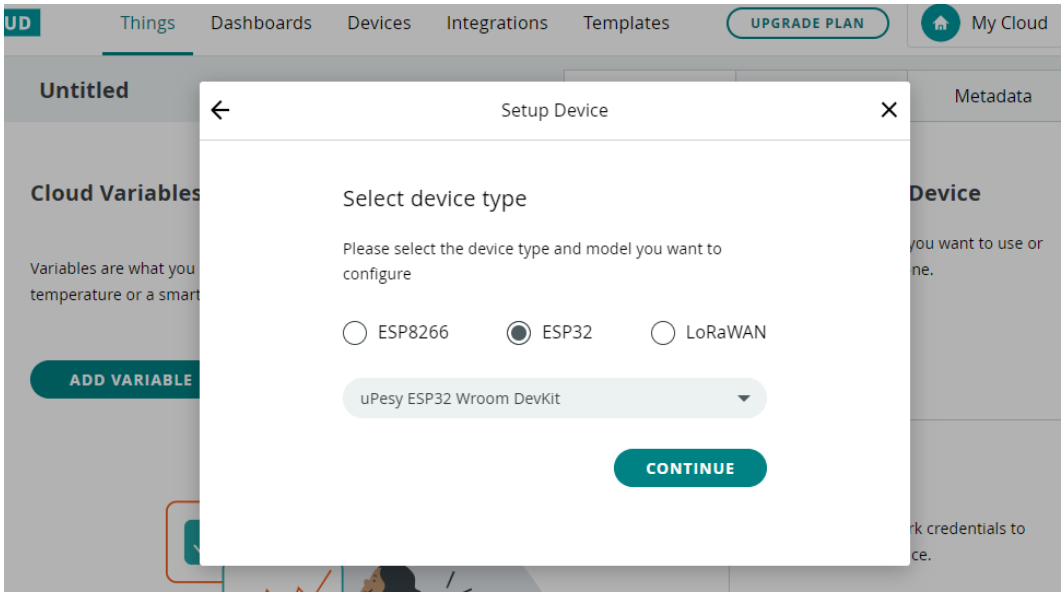

Figura 3.18 Selección del tipo de dispositivo.

Para terminar con la creación de dispositivo, la plataforma solicita darle un nombre al dispositivo. Después la plataforma asigna un *ID* y una *Secret* key: estos dos valores son importantes para la conexión del dispositivo a la plataforma, por lo que es importante descargar el documento que proporciona la plataforma en este paso (Figura 3.19).

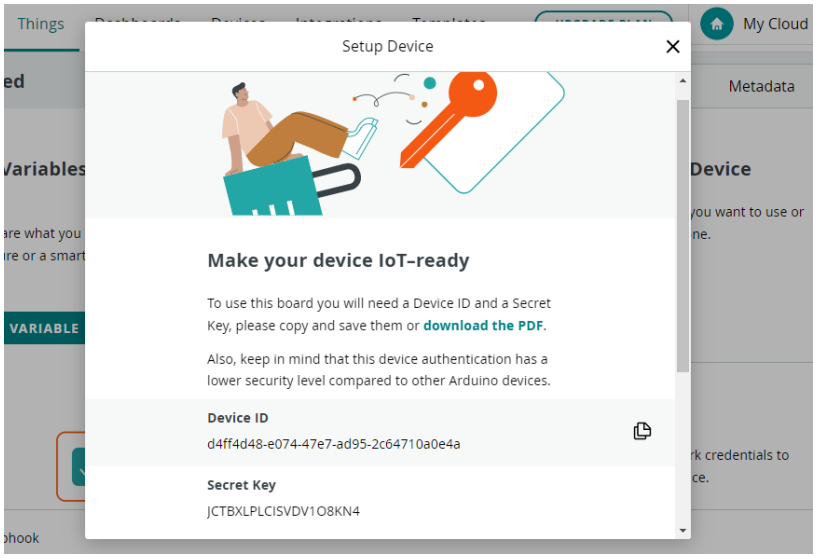

Figura 3.19 Identificador y llave secreta del dispositivo

Para conectar el dispositivo con la plataforma de *Arduino Cloud*, se selecciona en la pantalla de la opción *things, configure network*. En este paso se despliega una ventana en donde se solicita al usuario que ingrese el SSID de la red, el PASSWORD y el *Secret key* que proporcionó la plataforma en el paso anterior como se puede observar en la Figura 3.20 se muestra la ventana en donde se configuran los datos de la red.

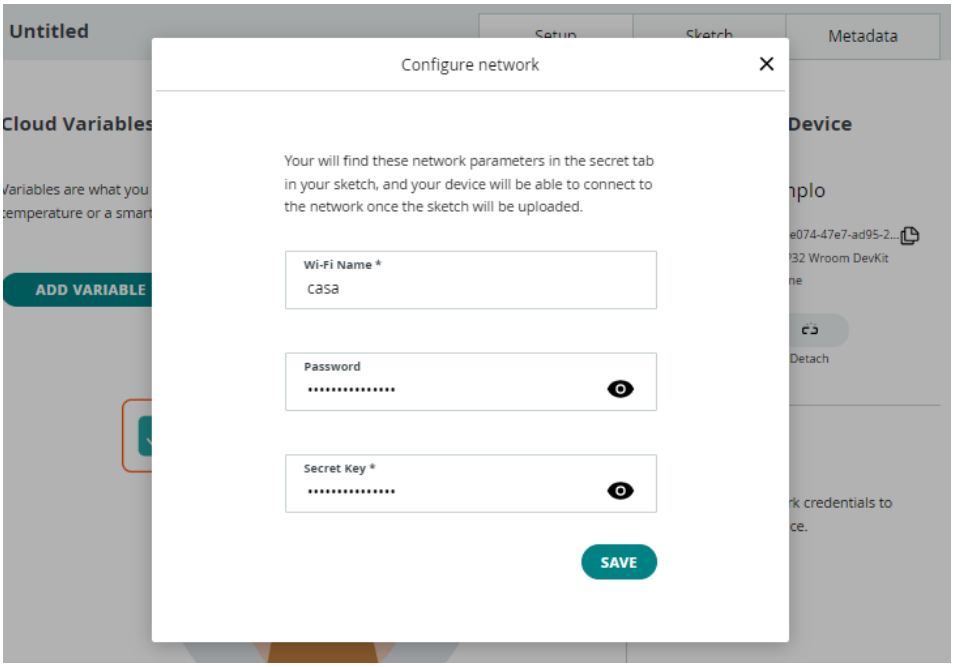

Figura 3.20 Conexión del dispositivo a la red

En este paso la plataforma IoT proporciona las líneas de código que al ejecutarse permite al dispositivo conectarse a la red; también a estas líneas de código se le agrega las funciones con las que funciona el dispositivo y la plataforma en conjunto. A continuación, en la Figura 3.21 se muestra el código generado por la plataforma para la conexión con el prototipo.

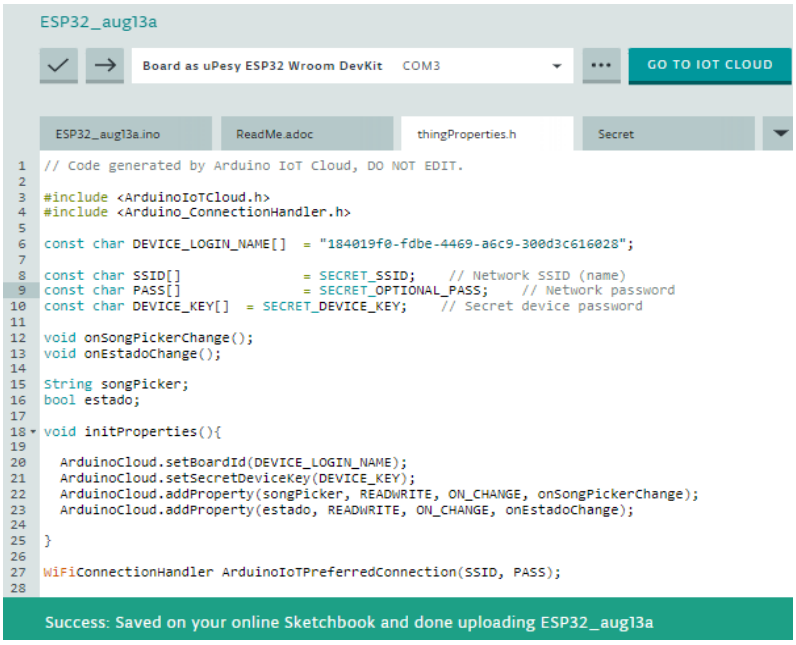

Figura 3.21 Código con el que se crea la conexión entre el prototipo y la nube.

Para que el código funcione en conjunto con la plataforma es necesario crear las variables que se van a ser utilizadas para manejar el prototipo desde la plataforma IoT o la aplicación móvil, para esto es necesario conectar el dispositivo a la red. En la plataforma se muestra la opción de añadir variable y en la misma pantalla se muestra si el dispositivo está activado, como se observa en la Figura 3.22

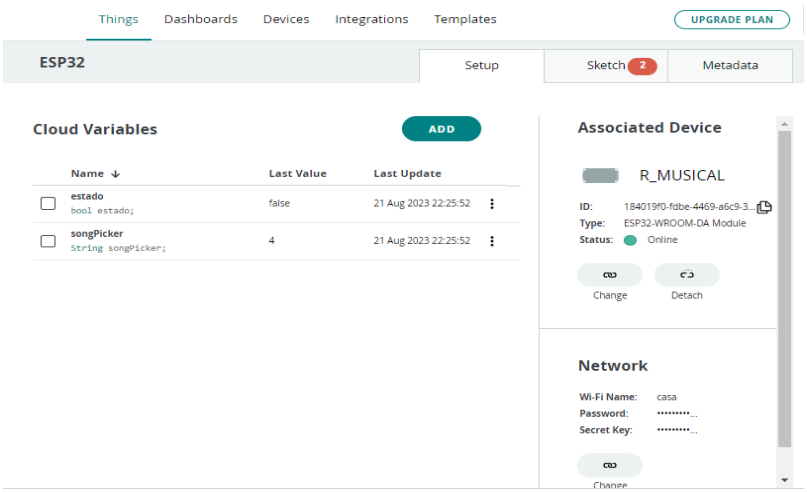

Figura 3.22 Sección para crear variables

Al momento de seleccionar añadir variable se despliega una pantalla en donde los campos permiten asignar un nombre a la variable, el tipo de variable y si la variable será solo de lectura o de lectura/escritura. Esta variable se crea automáticamente en las líneas de código. A continuación, en la Figura 3.23 se muestra la pantalla en donde se asigna las variables.

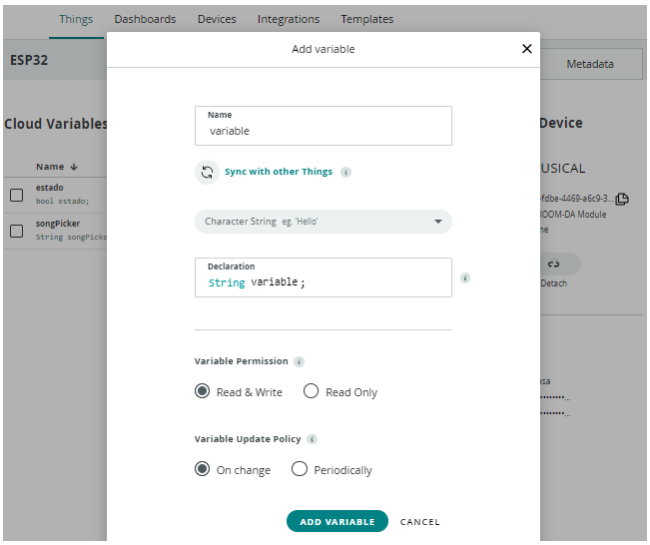

Figura 3.23 Campos para crear una variable.

Luego se crea la interfaz gráfica para que el usuario acceda a las canciones, para esto se escoge la opción *dashboards,* en este apartado se crea la interfaz o tablero para el usuario, después de darle nombre al tablero se selecciona ingresar al tablero donde se puede crear los diferentes objetos que se enlazaran al prototipo. Se escogió el *widget value selector* ya que, este elemento permite escoger y reproducir la canción; también se añadió el widget *push button* para reproducir y pausar la canción seleccionada. En la Figura 3.24 se muestra la interfaz creada dentro de la plataforma IoT.

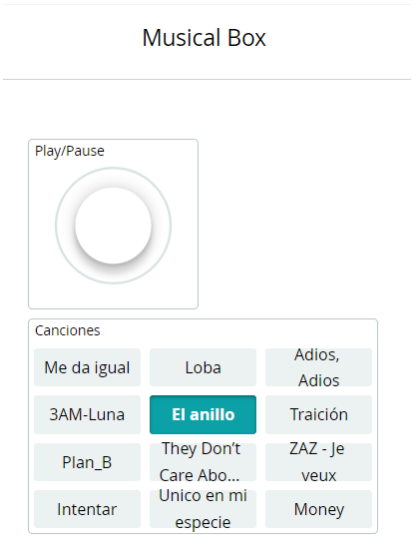

Figura 3.24 interfaz gráfica de la aplicación web para controlar el prototipo de forma remota.

Para lograr la vinculación exitosa entre el prototipo y la aplicación móvil, es necesario seguir los siguientes pasos:

- Descargar la aplicación *"Arduino IoT Remote"* de Arduino desde la tienda Google Play Store o Apple App Store.
- Luego, iniciar sesión en la aplicación utilizando la misma dirección de correo electrónico que fue utilizado para crear la cuenta en Arduino Cloud en la plataforma web.
- Una vez iniciada la sesión, se muestra la pantalla donde se selecciona los tableros que creados.
- Luego se muestra la interfaz en donde se encuentran los widgets para controlar el dispositivo desde la aplicación móvil.

A continuación, en la Figura 3.25 se muestra las pantallas correspondientes a la aplicación terminada.

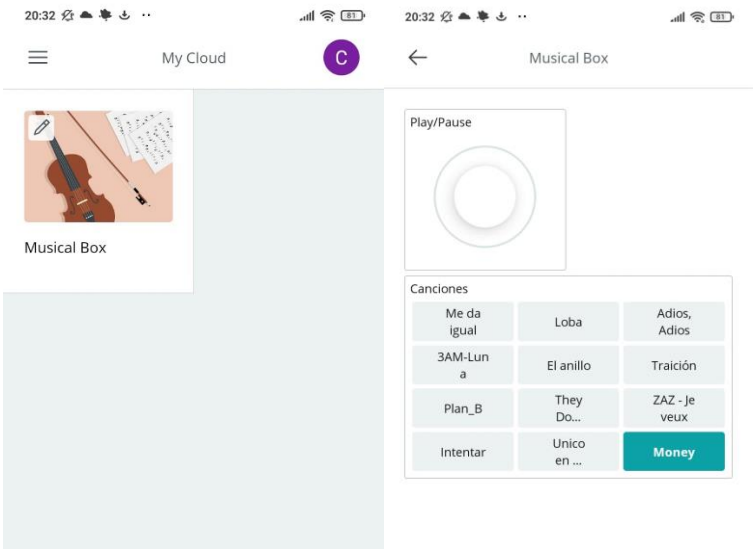

Figura 3.25. Aplicación móvil

Programación del prototipo

Para programar el microcontrolador ESP32 se utilizó el software Arduino IDE y el editor de la plataforma de *Arduino Cloud*.

Es necesario que el código contenga las librerías necesarias para que el microcontrolador se pueda comunicar con los demás elementos que conforman el prototipo para esto se utilizaron las siguientes librerías.

- *#include <SPI.h>* permite la comunicación SPI que se da entre el microcontrolador maestro y los dispositivos periféricos que actúen como esclavos.
- *#include <MFRC522.h>* Permite al microcontrolador ESP32 interactuar con módulos RFID de lectura y escritura basados en el modelo NFRC522 que trabaja a una frecuencia de 13.56 MHz.
- *#include "Arduino.h"* es utilizada para evitar problemas de compilación dado que esta biblioteca contiene los archivos con las definiciones y declaraciones necesarias para compilar de forma correcta los programas.
- *#include "SoftwareSerial.h"* es utilizada para habilitar los puertos seria para la comunicación de los pines digitales que no son dedicados para esa función es decir se utiliza para simular más puertos series.
- *#include "DFRobotDFPlayerMini.h"* Esta librería permite controlar el módulo *DFPlayer Mini* permitiendo que se reproduzcan archivos de audio de tipo MP3 alojados en una memoria microSD o en una memoria USB y reproducirlas a través de una bocina o audífonos.

- *#include "thingProperties.h"* no es considerada una librería, pero es el archivo generado automáticamente por la plataforma de *Arduino Cloud* que contiene las líneas de código necesarias para la conexión entre el prototipo y la plataforma IoT.

Luego se definen las variables y constantes que se utilizan en el código para crear procesos y funciones que servirán posteriormente para el funcionamiento del prototipo.

- *MFRC522 mfrc522(SS_PIN, RST_PIN)* define los pines de comunicación con el módulo RFID.
- *SoftwareSerial mySoftwareSerial(16, 17);* define los pines que se comunicación con el módulo MP3.
- *const int playPause = 25;* define el pin que se utilizara para controlar el boton de *play/pause.*
- *const int aleatorio = 26:* define el pin que se utilizara para controlar el botón de reproducción aleatoria.
- *bool inicioC= false;* se utiliza para controlar el estado del dispositivo, es decir si está reproduciendo un sonido o no.
- *bool inicioOnline = false;* es utilizado para simular el estado del botón play/pause en la nube.
- *String numero_cancion = "";* Permite leer el numero de la canción que se encuentra en la memoria microSD del módulo MP3 desde la plataforma IoT.
- *int contador_boton = 0;* esta variable se utiliza para la función de selector de canciones que se utiliza para seleccionar la canción en el widget que se creó en la plataforma IoT.

A continuación, en la Figura 3.26 se muestra las líneas en donde se define las librerías y variables para el funcionamiento del prototipo.

```
//Autor: Cristhian Verdezoto Implmenteacion de un Repductor de canciones con RFID y Aplicacio movil
////////**************-----Sección Variables u bbibliotecas-----*********////////
#include <SPI.h>
#include <MFRC522.h>
#include "Arduino.h"
#include "SoftwareSerial.h"
#include "DFRobotDFPlayerMini.h"
#include "thingProperties.h"
#define RST         15
#define SDA          21
MFRC522 mfrc522(SDA, RST);
SoftwareSerial mySoftwareSerial(16, 17);
DFRobotDFPlayerMini myDFPlayer;
const int playPause = 25; //pines elegidos para los botones solicitados
const int aleatorio = 26;
bool inicioC = false;  //Estado del inicio de cancion en el modulo fisico, empiesa apagado.
bool inicioOnline = false;//Estado del inicio de cancion en el modulo en linea
String numero_cancion = ""; // variable para extraer el valor int  de la cancion guardada en la memoria microSD
int contador_boton = 0;// iniciar el valor del widjet selectro de valores de la nube
```

Figura 3.26 Librerías y variables para el funcionamiento del prototipo

El siguiente paso para la configuración es definir el *void setup()* del programa, en esta parte se inicia y configura el estado de ejecución de las funciones variables que necesite el programa para funcionar correctamente. A continuación, la Figura 3.27 muestra el el código del *void setup()* del programa que ejecuta el prototipo.

```
void setup() {
  Serial.begin(9600);//inicio serial
  delay(500);
  initProperties();//inicio de las funciones del archivo thingProperties.h
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  mySoftwareSerial.begin(9600);// inicio del modulo MP3
  SPI.begin();                 // inicio de la comunicacíon del modulo RFID
  mfrc522.PCD_Init();          // inicio de la biblioteca RFID

  pinMode(playPause, INPUT_PULLUP);// botones en configuracion pull UP con R 1kohmios
  pinMode(aleatorio, INPUT_PULLUP);


  Serial.println(F("Iniciar Modulo MP3 ..."); //aviso de inicio

  if (!myDFPlayer.begin(mySoftwareSerial)) {      //Use softwareSerial to communicate with mp3.
    Serial.println(F("Revisar que anda mal:"));// aviso si no se inicia el modulo

  }
  Serial.println(F("DFPlayer Mini online. Place card on reader to play a specific song"));
  myDFPlayer.EQ(DFPLAYER_EQ_NORMAL);//aviso si esta en linea el reproductor y ya se puede leer una tarjeta RFID
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();

}
```

Figura 3.27 Código *void setup()* del prototipo

En el desarrollo del *void loop()* del programa se configuran los parámetros para que se detecte y lea la tarjeta RFID, para esto se utilizan las funciones *IsNewCardPresent(),* y *ReadCardSerial()* definidas en la biblioteca MFRC522. Además, se crea un *array* para guardar los valores que genera la tarjeta RFID con la finalidad de convertir el valor hexadecimal que arroja la tarjeta RFID en un numero decimal que pueda entender el módulo MP3. Luego, a partir de este valor el módulo MP3 reproduce la canción correspondiente al valor que esta almacenado en la tarjeta RFID, por esta razón en la

memoria microSD los archivos deben tener el formato *000X-Nombre de la canción* donde *X* es un valor entero positivo,

Adionalmente, se utilizan las funciones *myDFPlayer.start(), myDFPlayer.pause()* y *myDFPlayer.randomAll()*, definidas en la biblioteca *"DFRobotDFPlayerMini.h" para* darle la función de Play/ pause y reproducción aleatoria a los botones. Con el código que se muestra en la Figura 3.28 y Figura 3.29, el prototipo realiza estas operaciones.

```
void loop() {
    ArduinoCloud.update();// actualizacion del estado de la coneccion con la nube
      myDFPlayer.volume(30);// Valor Limite del volumen del modulo MP3
      //*******todas las claves están configuradas en FFFFFFFFFFFFh de la fábrica.**
     MFRC522::MIFARE_Key key;// autenticar la Tarjeta RFID
  for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;
  byte bloque;//variable para limitar el valor del numero hexadecimal
  byte longitud;// variable que define el tamaño del bloque
  MFRC522::StatusCode stado;//dar acceso al estado del la variable de estado del autentificador de la tarjeta RFID
//-----------Asignacion de funciones a los botones-------------
      if (digitalRead(playPause) == LOW)//inica en 0
  {
    if (inicioC) // decisión para iniciar o parar la reproduccion de la canción
    {
      myDFPlayer.pause();
      inicioC = false;
      Serial.println("Para");
    }
    else
    {
      inicioC = true;
      myDFPlayer.start();
      Serial.println("Inicia");
    }
    delay(500);
  }
```

Figura 3.28 Código *void loop()* primera parte

```
  stado = mfrc522.MIFARE_Read(bloque, array, &longitud);
  if (stado != MFRC522::STATUS_OK)
  {
    Serial.print(F("Fallo la lectura: "));
    Serial.println(mfrc522.GetStatusCodeName(stado));
    return;
  }
  String valorT = "";
  for (uint8_t i = 0; i < 16; i++) //
  {
    valorT += (char)array[i];
  }
  valorT.trim();//cambia el valor leido de la tarjeta MP3 por el valor aasignado a la cancion en el Modulo MP3
  Serial.print(valorT);//Muestra el valor
int numeroCancion = valorT.toInt(); // Extrae el valor entero de la memoria microSD
Serial.print("\nSe leyó la tarjeta con número: ");
Serial.print(numeroCancion);
Serial.print("\n");


 myDFPlayer.play(valorT.toInt());//Deacuerdo al valor final reproduce la cancion
    inicioC = true;


    Serial.println(F("\n End Reading"));
  delay(500);
  mfrc522.PICC_HaltA();//detiene la instancia
  mfrc522.PCD_StopCrypto1();//detiene la operacion de autenticacion
}
```

Figura 3.29 Código *void loop()* segunda parte

Para finalizar se crearon funciones para ejecutar acciones desde la plataforma IoT o la aplicación móvil en el prototipo, para esto se crearon las funciones *void onSongPickerChange(), void reproducirCancionOnline(Stringnombre) y void onEstadoChange()*, las cuales permiten reproducir la canción seleccionada en el aplicativo utilizando la comunicación MQTT entre la plataforma IoT y el dispositivo. Para seleccionar la canción se utiliza la función *onSongPickerChange* el cual se le vincula con la variable que designa el número de canción que se reproducirá, por otro lado, con

la función *reproducirCancionOnline (String nombre)* se inicia o se pausa la canción de acuerdo a la variable contador que controla el valor que tiene el botón del *widget* selector de canciones. Mediante la Figura 3.30 y la Figura 3.31, se muestra el código desarrollado.

```
void onEstadoChange()  {
  contador_boton++;
    Serial.print("\nContador: ");
    Serial.print(contador_boton);
    Serial.print("\n");
    if(contador_boton==2){
    if(isPlayingOnline){
        Serial.print("\n Pausando la música...\n");
        //paramos la musica
        myDFPlayer.pause();
        isPlayingOnline = false;
        isPlaying = false;
        Serial.println("Paused ok..\n");
      }else{
        Serial.print("Ejecutando la música...\n");
        reproducirCancionOnline(numero_cancion);
      }
        contador_boton=0;
    }
  }
```

Figura 3.30 Código funciones vinculadas a la plataforma IoT parte 1

```
void reproducirCancionOnline(String nombre){
  Serial.print("\nReproduciendo cancion online\n");
  myDFPlayer.pause();
  delay(300);
  myDFPlayer.start();
  isPlayingOnline = true;
  String cancion_a_reproducir="";
    if(nombre==""){
      cancion_a_reproducir = "1"; // por defecto si no se ha selecci
    }else{
      cancion_a_reproducir = nombre;
    }
  Serial.print("Se va a reproducir la siguiente canción:\n");
  Serial.print(cancion_a_reproducir);
  Serial.print("\n");
    // reproducir cancion
    int numero_de_pista = cancion_a_reproducir.toInt();
  Serial.print(numero_de_pista);
  Serial.print("\nDFP Player play...\n");
    myDFPlayer.play(numero_de_pista);
    contador_boton=0;
}
void onSongPickerChange()  {
    Serial.print("\nModificando la canción actual con el selector...\n")
    Serial.print(songPicker);
    Serial.print("\n");
    numero_cancion = songPicker;
  reproducirCancionOnline(numero_cancion);
}
```

Figura 3.31 Código funciones vinculadas a la plataforma IoT parte 2

Programador de cartas RFID

Dado que es necesario asignar un número específico a las tarjetas RFID, el cual se vincula con el nombre de la canción almacenada en la tarjeta microSD del módulo MP3. Para esto se realizó un programa que escriba en la tarjeta RFID dicho número. Este proceso necesita del uso del lector RFID RC522, dado que cuenta con la capacidad de

escribir en las tarjetas RFID, también se utiliza el módulo ESP32 que es el elemento que manda la orden a ejecutar a el módulo RFID.

La implementación de este programa proporciona soluciona la interacción entre las tarjetas RFID y el contenido musical.

En la primera parte del código se definen los pines utilizados, las bibliotecas para comunicar los módulos y darles funcionalidades como se observa en la Figura 3.32.

```
1  //Cristhian Verdezoto Codigo para designar numero a las tarjetas RFID
2  #include <SPI.h>
3  #include <MFRC522.h>
4  #define RST         15           // Asignacion de pines
5  #define SDA         21
6  String mode;
7  MFRC522 mfrc522(SDA, RST);   // Create MFRC522 instance
```

Figura 3.32 Código para el programador de tarjetas

Luego se inicia las funciones de comunicación serial entre el módulo ESP32 y el módulo RFID y se inicia el módulo RFID, luego para empezar la lectura y escritura de las tarjetas RFID se solicita al usuario ingresar al modo manual del dispositivo dado que la biblioteca tiene este modo de uso y es necesario llamar esta función en el código (Figura 3.33).

```
9  void setup() {
10   Serial.begin(115200);
11   SPI.begin();
12   mfrc522.PCD_Init();
13   Serial.println(F("escribe manual para empesar a designar el valor a la tarjeta "));
14 }
15 void loop() {
16   if (Serial.available()) {
17     mode = Serial.readStringUntil('\n');
18     if (mode == "manual") {
19       Serial.println(" ");
20       Serial.println("El dispositivo esta en modo  " + mode + );
21       Serial.println(F("Coloque la tajeta en el lector y escriba el numero que quiera asigna a la tarjeta desde 001-255"));
22
23     }
24
25     while (mode == "manual") {
26       manualModus();
27     }
```

Figura 3.33 Escritura manual de la tarjeta RFID

Para continuar con el desarrollo del programa se da las instrucciones para que las tarjetas sean autenticadas por el lector. Este proceso es necesario porque las tarjetas están configuradas por defecto con una clave en hexadecimal desde fábrica, luego se ordena al lector detectar la tarjeta. Por medio de una decisión *IF* se evalúa si existe o no una tarjeta en el lector, en el caso de que si se solicita al usuario que escriba en el monitor serial el valor de la tarjeta que se vincula con la canción deseada, y en caso de que no se encuentre tarjeta en el lector se solicita al usuario que coloque una tarjeta

RFID, luego se repite este proceso hasta que se configuren las tarjetas que se utilizaran en el prototipo. En la Figura 3.34 que se muestra a continuación se observan las líneas de código utilizadas.

```
// todas las claves están configuradas en FFFFFFFFFFFFh en el momento de la entrega del chip desde la fábrica
MFRC522::MIFARE_Key key;
for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;
// Restablece el bucle si no hay ninguna tarjeta nueva en el sensor/lector
if ( ! mfrc522.PICC_IsNewCardPresent()) {
  return;
}
// Seleccionar una carta
if ( ! mfrc522.PICC_ReadCardSerial()) {
  return;
}
Serial.print(F("identificador de tarjeta:"));    //Cambio de indentificador
for (byte i = 0; i < mfrc522.uid.size; i++) {
  Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
  Serial.print(mfrc522.uid.uidByte[i], HEX);
}
Serial.print(F(" Tipo de tarjeta de proximidad : "));   // elección de  tarjeta de proximidad
MFRC522::PICC_Type piccTipo = mfrc522.PICC_GetType(mfrc522.uid.sak);
Serial.println(mfrc522.PICC_GetTypeName(piccTipo));
byte buffer[34];
byte block;
MFRC522::StatusCode status;
byte len;
Serial.setTimeout(20000L) ;     // espera para la entrada al serial
// numero de la cancion
Serial.println(F("escribe el numero de la cancion para asociar"));
len = Serial.readBytesUntil('\n', (char *) buffer, 30) ; // ller el numero del serial
for (byte i = len; i < 30; i++) buffer[i] = ' ';     // espacio donde se guarda
block = 1;
  estado = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, &key, &(mfrc522.uid));
if (estado != MFRC522::STATUS_OK) {
  Serial.print(F("autenticado fallido: "));
  Serial.println(mfrc522.GetStatusCodeName(estado));
  return;

 }
 else Serial.println(F("exito: "));
 Serial.println("coloque una nueva tarjeta en el lector");
 Serial.println(" ");
 mfrc522.PICC_HaltA(); // detener deteccion del lector RFID
 mfrc522.PCD_StopCrypto1();  // parar el cifrado de autenticacion
```

Figura 3.34 Instrucciones para escribir la tarjeta RFID

Ensamblaje del circuito dentro del protector del prototipo

Después de comprobar las conexiones del circuito, el funcionamiento de la nube y el funcionamiento de los botones del prototipo se realizó el ensamblaje dentro de la caja protectora del dispositivo como se observa en la Figura 3.35.

Figura 3.35 Prototipo ensamblado

El presupuesto que se utiliza para el armado del prototipo se detalla en la Tabla 3.5 y la Tabla *3.6* los materiales y componentes utilizados con sus respectivos costos.

Tabla 3.5 Detalle de precios de los componentes del prototipo [35], [37]

Componentes	Cantidad	Precio
Modulo ESP32	1	12.00$
Modulo RFID	1	4.00$
Modulo MP3	1	5.00$
Amplificador	1	3.00$
Baquelita	1	1.25$
Botones	2	0.50$
Parlantes	1	2.00$

Tabla 3.6 Detalle de precios de los materiales utilizados

Materiales	Cantidad	Precio
Estaño	1 (m)	1.25 $
Cautín	1	2.50 $
Cloruro férrico	1	1.70 $
Broca 1 (mm)	1	1.25 $
Protoboard	1	3.00 $
Cortes a laser	1	1.00 $
Madera MDF	1	2.00 $
TOTAL		40.45

Ejecución de las pruebas de funcionamiento

La primera prueba que se realiza es mostrar que se enciende el módulo ESP32, el módulo MP3 y el módulo RFID para mostrar esto en la Figura 3.36 se muestra el dispositivo encendido, una vez energizado el prototipo.

Figura 3.36 Encendido del prototipo

La verificación del funcionamiento de la reproducción de canciones se la realiza mediante mensajes en el monitor serial de la plataforma *Arduino Cloud*.

Para el funcionamiento de los botones *play/pause* y reproducción aleatoria se designó en el código que se muestre las palabras “iniciar” y “parar” para las funciones *play* y *pause*, mientras que para el botón aleatorio se muestra el mensaje “Reproducción aleatoria. A continuación, se muestra en la Figura 3.37 los mensajes mencionados.

```
DFP Player play...
Inicia
Parar
Inicia
Parar
Inicia
Parar
Inicia
Parar
Inicia
Parar
Inicia
Parar
Inicia
Parar
Inicia
Parar
Reproduccion aleatoria
Reproduccion aleatoria
```

Figura 3.37 Funcionamiento de los botones

Luego para mostrar el funcionamiento del lector RFID, en la Figura 3.38 se puede observar la información de la tarjeta RFID, el número de canción que se va a reproducir en el prototipo y el mensaje que termina la reproducción cuando se cambia de canción.

```
Se encontro tarjeta*
Card UID: 23 38 1A 1C
Card SAK: 08
PICC type: MIFARE 1KB
valorT: 0004
Se leyó la tarjeta con número: 4

**Fin de la reproduccion

Se encontro tarjeta*
Card UID: A3 82 E6 95
Card SAK: 08
PICC type: MIFARE 1KB
valorT: 0002
Se leyó la tarjeta con número: 2

**Fin de la reproduccion

Se encontro tarjeta*
Card UID: 83 F6 78 A3
Card SAK: 08
PICC type: MIFARE 1KB
valorT: 0005
Se leyó la tarjeta con número: 5

**Fin de la reproduccion
```

Figura 3.38 Funcionamiento del Lector RFID

A continuación, se muestra en la Figura 3.39 los mensajes desplegados en el monitor serie cuando se ejecutan los widgets de la aplicación móvil.

```
Modificando la canción actual con el selector...
8

Reproduciendo cancion online
Se va a reproducir la siguiente canción:
8
8
DFP Player play...

Modificando la canción actual con el selector...
11

Reproduciendo cancion online
Se va a reproducir la siguiente canción:
11
11
DFP Player play...
```

Figura 3.39 Funcionamiento de la aplicación móvil

Para finalizar se muestra en la Figura 3.40, el estado del dispositivo en la plataforma de Arduino Cloud con el fin de mostrar que el dispositivo se encuentra conectado a Internet.

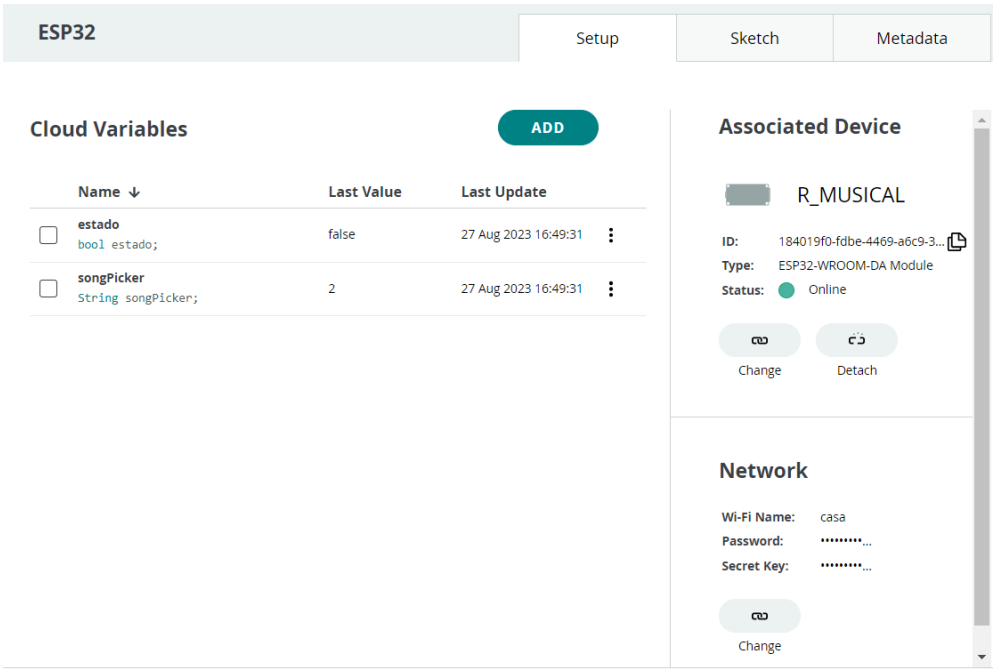

Figura 3.40 Dispositivo en línea conectado a *Arduino Cloud*

4 CONCLUSIONES

- Después de utilizar el microcontrolador ESP32 en conjunto con los módulos *DFPlayer mini* y RFID RC522 y conectarse a la plataforma de Arduino Cloud, el prototipo permite reproducir una canción mediante el uso de una tarjeta RFID y desde la aplicación móvil, además los botones *play/pause* realizan su función de manera correcta.
- El uso de tarjetas RFID permite al usuario cambiar fácilmente de canción reproducida, por lo cual el prototipo puede ser utilizado por personas con conocimientos limitados en tecnología.
- En el momento de la experimentación se presentó un problema con el volumen del módulo MP3, cuando se subía el volumen al máximo el dispositivo se apagaba, para solucionar este problema se agregó un amplificador para controlar el volumen y mantener el voltaje en el módulo MP3. Al ser implementada esta solución el dispositivo ya no presentó ese problema.
- La decisión de utilizar Arduino IDE como software de programación para el prototipo, fue un acierto dado que existe bastante documentación sobre cómo realizar la programación y además es compatible con el microcontrolador; también, facilitó la implementación de la aplicación móvil dado, que el código y

las bibliotecas utilizadas en el software de Arduino IDE son totalmente compatibles en el editor de código de *Arduino Cloud*.

- La plataforma *Arduino Cloud* facilitó la implementación de la aplicación móvil, las funcionalidades que utiliza el prototipo no requieren utilizar un plan de pago de la plataforma, lo que volvió la mejor opción para la implementación del prototipo.
- El diseño del prototipo emplea diversas áreas de conocimiento juntas como: la programación que se necesita para dar el funcionamiento al prototipo, el conocimiento para entender cómo se comunican los módulos empleados, la creatividad para el diseño 3D del protector del prototipo y el diseño del circuito fueron claves para materializar la idea propuesta para este Trabajo de Integración Curricular.

5 RECOMENDACIONES

- El dispositivo es escalable, por lo cual se pueden añadir más funcionalidades como la implementación de más widgets que proporcionen más funciones al dispositivo como volumen, tiempo de reproducción, entre otras.
- Implementar un diseño más atractivo al aspecto exterior del dispositivo para que sea más llamativo para los niños, o adultos que les llame la atención interactuar con música o sonidos.
- El dispositivo puede ser utilizado para aprender idiomas o escuchar audiolibros de una forma más interactiva, debido a que la tarjeta RFID es un puente entre lo digital y lo físico, esto permite que el usuario pueda asociar fácilmente la tarjeta con el contenido especifico.
- Es posible que el dispositivo se vuelva portátil agregando una batería, esta característica mejoraría la eficiencia y la comodidad de uso del dispositivo para el usuario.
- El dispositivo también podría interactuar con aplicativos como Spotify o YouTube, debido a la característica de comunicación por medio de Wi-Fi que posee el módulo ESP32 o con las URLs grabadas en las tarjetas RFID, esto puede hacer que el dispositivo proporcione al usuario una experiencia más actualizada a las tendencias tecnológicas.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] S. I. Ao and International Association of Engineers., *World Congress on Engineering and Computer Science : WCECS 2009 :* Newswood Ltd., International Association of Engineers, 2009.

[2] E. Y. Telecomunicaciones Puma Maldonado Galo Javier and T. Castro César David, “Implementación de un sistema RFID para el control y seguridad de los equipos del laboratorio de informática de la facultad de Ingeniería eléctrica y electrónica de la Escuela Politécnica Nacional,” 2014.

[3] E. Pacheco Franklin, “2. Funcionamiento de un sistema RFID | Descargar Diagrama Científico.” Accessed: Jul. 31, 2023. [Online]. Available: https://www.researchgate.net/figure/Figura-22-Funcionamiento-de-un-sistema-RFID_fig1_277204150

[4] D. Ibrahim, “Front Matter,” in *Microcontroller Based Applied Digital Control*, John Wiley & Sons, Ltd, 2006, pp. i–xii. doi: 10.1002/0470863374.fmatter.

[5] Corrales Santiago, “Electrónica Práctica Con Microcontroladores Pic (Santiago Corrales V.)”.

[6] D’Inca Celeste O., “Microprocesadores,” pp. 81–94, Sep. 2012.

[7] Electrotec, “Electrotec | Microcontrolador VS PLC.” Accessed: Aug. 21, 2023. [Online]. Available: https://electrotec.pe/blog/microcontroladorvsplc

[8] R. Picking, Glyndŵr University. ARCLab, Institute of Electrical and Electronics Engineers. United Kingdom and Republic of Ireland Section, and Institute of Electrical and Electronics Engineers, *2017 Internet Technologies and Applications (ITA)*. 2017.

[9] Espressif Systems, “ESP32WROOM32 Datasheet,” 2023. [Online]. Available: https://www.espressif.com/en/support/download/documents.

[10] Anonimo, “ESP32-DevKitC Development Boards - Espressif Systems | Mouser.” Accessed: Aug. 21, 2023. [Online]. Available: https://www.mouser.ec/new/espressif/espressif-esp32-devkitc-boards/

[11] Arduino.cc, “¿Qué es Arduino?,” https://www.arduino.cc/en/Guide/Introduction.

[12] R. Enríquez Herrador, “Guía de Usuario de Arduino,” 2009.

[13] R. Enríquez Herrador, “Guía de Usuario de Arduino,” 2009.

[14] DFRobot, “ModuloMP3.” Accessed: Aug. 26, 2023. [Online]. Available: https://www.dfrobot.com/blog

[15] DF Robot, “DFPlayer Mini Mp3 Player - DFRobot Wiki.” Accessed: Aug. 05, 2023. [Online]. Available: https://wiki.dfrobot.com/DFPlayer_Mini_SKU_DFR0299

[16] Martinez Jacobson Rodrigo, “Comparativa y estudio de plataformas IoT,” Catalunya, 2017.

[17] Fremantle Paul, “A Reference Architecture for the Internet of Things.” Accessed: Aug. 21, 2023. [Online]. Available: https://wso2.com/whitepapers/a-reference-architecture-for-the-internet-of-things/

[18] IBM, “Mensajería de MQTT - Documentación de IBM.” Accessed: Sep. 25, 2023. [Online]. Available: https://www.ibm.com/docs/es/maximo-monitor/continuous-delivery?topic=security-mqtt-messaging

[19] Anonimo, “ESP32 vs ESP8266 ¿Cuales son las diferencias entre ambos módulos?” Accessed: Aug. 05, 2023. [Online]. Available: https://descubrearduino.com/esp32-vs-esp8266/

[20] Atmel, “Hoja de datos ARM,” *DATASHHET*, pp. 70–100, 2012.

[21] Anonimo, “Teensy Microcontrolador.” Accessed: Aug. 05, 2023. [Online]. Available: https://www.pjrc.com/store/teensy36.html

[22] Anonimo, “JQ6500 Mp3 Player Module.” Accessed: Aug. 05, 2023. [Online]. Available: https://sparks.gogo.co.nz/jq6500/index.html

[23] Fraunhofer IIS and Thomson, “Datasheet Mp3VS1053b,” pp. 25–50, Dec. 2012.

[24] Nuñes Jose, “RFID Usando RDM6300 y Arduino 101, TinyTILE o ARDUINO UNO – COSTARICAMAKERS.com.” Accessed: Aug. 05, 2023. [Online]. Available: https://costaricamakers.com/rfid-usando-rdm6300-y-arduino-101-tinytile/

[25] NXP semiconductors, “General description PN532/C1 Near Field Communication (NFC) controller,” pp. 20–50, 2017.

[26] Sabogal Alejandro, “Módulo Lector RFID-RC522 RF con Arduino UNO R3 SPI.” Accessed: Aug. 05, 2023. [Online]. Available: https://hetpro-store.com/TUTORIALES/modulo-lector-rfid-rc522-rf-con-arduino/

[27] calero. P, “Lector de tarjetas RFID con PIC18F2455 y lector RC522.” Accessed: Aug. 01, 2023. [Online]. Available: https://www.pcalero.com/Blog/lector-de-tarjetas-rfid-con-pic18f2455-y-lector-rc522/

[28] Arduino IoT Cloud, “API en la nube de Arduino IoT | Documentación de Arduino.” Accessed: Aug. 05, 2023. [Online]. Available: https://docs.arduino.cc/arduino-cloud/getting-started/api-overview

[29] Blync, “Pricing | Blynk IoT Platform.” Accessed: Aug. 05, 2023. [Online]. Available: https://blynk.io/pricing

[30] Amazon Web Services, “Conexión segura de dispositivos IoT – Precios de AWS IoT Core – Amazon Web Services.” Accessed: Aug. 05, 2023. [Online]. Available: https://aws.amazon.com/es/iot-core/pricing/?did=ap_card&trk=ap_card

[31] Anonimo, “Un relee me vuelve Loco - Español / Microcontroladores - Arduino Forum.” Accessed: Aug. 22, 2023. [Online]. Available: https://forum.arduino.cc/t/un-relee-me-vuelve-loco/698769/3

[32] Anonimo, “Aumento distancia rfid - Español / Hardware - Arduino Forum,” Arduino Forum. Accessed: Aug. 22, 2023. [Online]. Available: https://forum.arduino.cc/t/aumento-distancia-rfid/519073

[33] Crespo.J, “Bus SPI | Aprendiendo Arduino,” Arduino, Bus SPI, MISO, MOSI, Programado ISP . Accessed: Aug. 22, 2023. [Online]. Available: https://aprendiendoarduino.wordpress.com/2016/11/13/bus-spi/

[34] Mallari Jan, “Cómo configurar la comunicación UART en Arduino | Arduino.cl - Compra tu Arduino en Línea.” Accessed: Aug. 22, 2023. [Online]. Available: https://arduino.cl/como-configurar-la-comunicacion-uart-en-arduino/

[35] mercado libre, “Módulo Rfid Rc522 Tarjeta Y Llavero | MercadoLibre,” Modulo RC22. Accessed: Sep. 25, 2023. [Online]. Available: https://articulo.mercadolibre.com.ec/MEC-521144758-be-modulo-rfid-rc522-tarjeta-y-llavero-_JM#position=4&search_layout=stack&type=item&tracking_id=a2187811-9d73-4e13-887c-5636405fbcc3

[36] mercado libre, “Módulo Dfplayer Mini Mp3 Player | MercadoLibre.” Accessed: Sep. 25, 2023. [Online]. Available: https://articulo.mercadolibre.com.ec/MEC-547464818-modulo-dfplayer-mini-mp3-player-

_JM#position=2&search_layout=stack&type=item&tracking_id=14dd1a7b-23e0-4998-ac2f-b50996cb2ae2

[37] mercado libre, “Modulo Esp32 Wi-Fi Bluetooth Arduino 38 Pines Innovatech | MercadoLibre.” Accessed: Sep. 25, 2023. [Online]. Available: https://articulo.mercadolibre.com.ec/MEC-517915266-modulo-esp32-Wi-Fi-bluetooth-arduino-38-pines-innovatech-_JM#position=3&search_layout=stack&type=item&tracking_id=48d685c6-3098-41fa-a989-9e74261667fd

7 ANEXOS

La lista de los **Anexos** se muestra a continuación:

ANEXO I. Certificado de originalidad

ANEXO II. Enlaces

ANEXO III. Conjunto de datos extensos

ANEXO I: Certificado de Originalidad

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 28 de agosto de 2023

De mi consideración:

Yo, ANDRES FERNANDO REYES CASTRO, en calidad de Director del Trabajo de Integración Curricular titulado IMPLEMENTACIÓN DE UN PROTOTIPO PARA REPRODUCCIÓN DE CANCIONES MEDIANTE TARJETAS RFID Y APLICACIÓN MÓVIL elaborado por el estudiante CRISTHIAN WILLIAM VERDEZOTO ARAUZ de la carrera en TECNÓLOGÍA SUPERIOR EN REDES Y TELECOMUNICACIONES, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito completo, producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 11%.

Es todo cuanto puedo certificar en honor a la verdad, pudiendo el interesado hacer uso del presente documento para los trámites de titulación.

NOTA: Se adjunta el link del informe generado por la herramienta Turnitin.

LINK

Atentamente,

ANDRES FERNANDO REYES CASTRO

Profesor Ocasional a Tiempo Completo

Escuela de Formación de Tecnólogos

ANEXO II: Enlaces

Anexo II.I Código QR de la implementación y pruebas de funcionamiento

ANEXO III: Códigos Fuente

Código Parte 1

```
//Autor: Cristhian Verdezoto Implementación de un Reproductor de canciones con RFID y Aplicación móvil

////////**************-----Sección Variables u bibliotecas-----*********////////

#include <SPI.h>

#include <MFRC522.h>

#include "Arduino.h"

#include "SoftwareSerial.h"

#include "DFRobotDFPlayerMini.h"

#include "thingProperties.h"

#define RST        15

#define SDA         21

MFRC522 mfrc522(SDA, RST);

SoftwareSerial mySoftwareSerial(16, 17);

DFRobotDFPlayerMini myDFPlayer;

const int playPause = 25; //pines elegidos para los botones solicitados

const int aleatorio = 26;

bool inicioC = false;  //Estado del inicio de canción en el módulo físico, empieza apagado.

bool inicioOnline = false;//Estado del inicio de canción en el módulo en linea

String numero_cancion = ""; // variable para extraer el valor int de la canción guardada en la memoria microSD

int contador_boton = 0;// iniciar el valor del widget selector de valores de la nube

//-------- Inicio de Comunicacion SPI,UART y Plataforma Arduino Cloud----------

void setup() {

  Serial.begin(9600);//inicio serial

   delay(500);
```

```
  initProperties();//inicio de las funciones del archivo thingProperties.h

  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  mySoftwareSerial.begin(9600);// inicio del módulo MP3

  SPI.begin();               // inicio de la comunicación del módulo RFID

  mfrc522.PCD_Init();        // inicio de la biblioteca RFID

  pinMode(playPause, INPUT_PULLUP);// botones en configuración pull UP con R
1kohmios

  pinMode(aleatorio, INPUT_PULLUP);

  Serial.println(F("Iniciar Modulo MP3 ...")); //aviso de inicio

  if (!myDFPlayer.begin(mySoftwareSerial)) {      //Use softwareSerial to communicate
with mp3.

    Serial.println(F("Revisar que anda mal:"));// aviso si no se inicia el modulo

  }

  Serial.println(F("DFPlayer Mini en linea"));

  myDFPlayer.EQ(DFPLAYER_EQ_NORMAL);//aviso si está en linea el reproductor y
ya se puede leer una tarjeta RFID

  setDebugMessageLevel(2);

  ArduinoCloud.printDebugInfo();

}

//------- Ejecución de las Funciones para el funcionamiento------------

void loop() {

    ArduinoCloud.update();// actualización del estado de la conexión con la nube

      myDFPlayer.volume(30);// Valor Limite del volumen del módulo MP3

      //*todas las claves están configuradas en FFFFFFFFFFFFh de la fábrica.

     MFRC522::MIFARE_Key key;// autenticar la Tarjeta RFID

  for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;

  byte bloque;//variable para limitar el valor del número hexadecimal

  byte longitud;// variable que define el tamaño del bloque
```

```
  MFRC522::StatusCode stado;//dar acceso al estado de la variable de estado del autentificador de la tarjeta RFID

//-----------Asignacion de funciones a los botones----------

      if (digitalRead(playPause) == LOW)//inicia en 0

 {

   if (inicioC) // decisión para iniciar o parar la reproducción de la canción

   {

     myDFPlayer.pause();

     inicioC = false;

     Serial.println("Para");

   }

   else

   {

     inicioC = true;

     myDFPlayer.start();

     Serial.println("Inicia");

   }

   delay(500);

 }

  if (digitalRead(aleatorio) == LOW) {//decisión para iniciar la reproduciioon aleatoria

   myDFPlayer.randomAll();// funcion de la biblioteca del módulo MP3

   Serial.println("reproduccion aleatoria");

   inicioC = true;

   delay(500);

 }

  if ( mfrc522.PICC_IsNewCardPresent())// decisión para saber si se presenta una nueva carta RFID

 {
```

```
    if ( ! mfrc522.PICC_ReadCardSerial())// decisión para leer una nueva carta RFID

    {

      return;

    }

    Serial.println(F("Se encontro tarjeta"));//en caso de que exista una carta en el leector

     mfrc522.PICC_DumpDetailsToSerial(&(mfrc522.uid)); //función para mostrar la
información de la tarjeta RFID

    Serial.print(F("valorT: ")); //imprime el valor numérico

    byte array[18]; //reglas de almacenamiento de los valores arrojados por la tarjeta
puesta en el lector

    bloque = 1;

    longitud = 18;

   stado = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, 1,
&key, &(mfrc522.uid));//designación de la variable de estado de la Tarjeta RFID

    if (stado != MFRC522::STATUS_OK)//decisión de estado del lector RFID respeto a
la Tarjeta RFID

    {

      Serial.print(F("No se autentico: "));

      Serial.println(mfrc522.GetStatusCodeName(stado));

      return;

    }

    stado = mfrc522.MIFARE_Read(bloque, array, &longitud);

    if (stado != MFRC522::STATUS_OK)

    {

      Serial.print(F("Fallo la lectura: "));

      Serial.println(mfrc522.GetStatusCodeName(stado));

      return;

    }

    String valorT = "";
```

```
    for (uint8_t i = 0; i < 16; i++) //

    {

      valorT += (char)array[i];

    }

    valorT.trim();//cambia el valor leído de la tarjeta MP3 por el valor asignado a la
canción en el Módulo MP3

    Serial.print(valorT);//Muestra el valor

          int numeroCancion = valorT.toInt(); // Extrae el valor entero de la memoria
microSD

          Serial.print("\nSe leyó la tarjeta con número: ");

          Serial.print(numeroCancion);

          Serial.print("\n");

   myDFPlayer.play(valorT.toInt());//De acuerdo al valor final reproduce la canción

      inicioC = true;

      Serial.println(F("\n End Reading"));

    delay(500);

    mfrc522.PICC_HaltA();//detiene la instancia

    mfrc522.PCD_StopCrypto1();//detiene la operación de autenticación

  }

}

// Funciones para el uso de los widgets de la nube

void onEstadoChange()  {

  // función para parao iniciar la canción que se encuentre en la cola.

          contador_boton++;

    Serial.print("\nContador: ");

    Serial.print(contador_boton);

    Serial.print("\n");

          if(contador_boton==2){//para permitir estado encendido o apagado
```

```
                if(inicioOnline){// se decide entre iniciar o parar la canción ya sea que se
haya dado la orden de reproducir desde el módulo físico o desde la nube

                        Serial.print("\NPausando la música...\n");

                        //paramos la música

                        myDFPlayer.pause();

                        inicioOnline = false;

                        inicioC = false;

                        Serial.println("Pausado..\n");

                }else{

                        Serial.print("Reanudando la música...\n");

                        myDFPlayer.start();

                }

        contador_boton=0;

          }

}

void reproducirCancionOnline(String nombre){//de acuerdo al botón seleccionado se
reproducirá la canción

          Serial.print("\nReproduciendo cancion online\n");

          myDFPlayer.pause();

          delay(300);

          myDFPlayer.start();

          inicioOnline = true;

          String cancion_a_reproducir="";

          if(nombre==""){

                  cancion_a_reproducir = "1";

          }else{

                  cancion_a_reproducir = nombre;

          }
```

```
        Serial.print("Se va a reproducir la siguiente canción:\n");
        Serial.print(cancion_a_reproducir);
        Serial.print("\n");
    int numero_de_pista = cancion_a_reproducir.toInt();
        Serial.print(numero_de_pista);
        Serial.print("\nDFPlayer play...\n");
    myDFPlayer.play(numero_de_pista);
    contador_boton=0;
}
void onSongPickerChange()  {//se define a los botones del selector como numeros de cancion de acuerdo a la lista de reproducción del módulo MP3
        Serial.print("\nModificando la canción actual con el selector...\n");
        Serial.print(songPicker);
        Serial.print("\n");
        numero_cancion = songPicker;
        reproducirCancionOnline(numero_cancion);
}
```

Codigo Parte 2 (Programador de Tarjetas RFID)

```
//Cristhian Verdezoto Código para designar número a las tarjetas RFID
#include <SPI.h>
#include <MFRC522.h>
#define RST         15          // Asignación de pines
#define SDA         21
String mode;
MFRC522 mfrc522(SDA, RST);
//--------------Inicio de la comunicación ESP32 RFID-----------------
void setup() {
  Serial.begin(115200);
```

```
  SPI.begin();

  mfrc522.PCD_Init();

  Serial.println(F("escribe manual para empezar a asignar el valor a la tarjeta "));

}

void loop() {

  if (Serial.available()) {

    mode = Serial.readStringUntil('\n');

    if (mode == "manual") {

      Serial.println(" ");

      Serial.println("El dispositivo está en modo  " + mode + );

      Serial.println(F("Coloque la tarjeta en el lector y escriba el numero que quiera
asigna a la tarjeta desde 001-255"));



    }



    while (mode == "manual") {

      manualModus();

    }



    if (mode !== "manual") {

      Serial.println("vuelva a escribir un valor");

    }

  }

}

void manualModus() {



  // todas las claves están configuradas en FFFFFFFFFFFFh en el momento de la
entrega del chip desde la fábrica
```

```
MFRC522::MIFARE_Key key;
for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;


if ( ! mfrc522.PICC_IsNewCardPresent()) {
  return;
}
// Seleccionar una carta
if ( ! mfrc522.PICC_ReadCardSerial()) {
  return;
}
Serial.print(F("identificador de tarjeta:"));    //Cambio de identificador
for (byte i = 0; i < mfrc522.uid.size; i++) {
  Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
  Serial.print(mfrc522.uid.uidByte[i], HEX);
}
Serial.print(F(" Tipo de tarjeta de proximidad : "));   // elección de  tarjeta de
proximidad
MFRC522::PICC_Type piccTipo = mfrc522.PICC_GetType(mfrc522.uid.sak);
Serial.println(mfrc522.PICC_GetTypeName(piccTipo));
byte array[34];
byte bloque;
MFRC522::StatusCode status;
byte longitud;
Serial.setTimeout(20000L) ;     // espera para la entrada al serial
// numero de la canción
Serial.println(F("escribe el numero de la canción para asociar"));
len = Serial.readBytesUntil('\n', (char *) array, 30) ; // leer el numero del serial
for (byte i = longitud; i < 30; i++) buffer[i] = ' ';     // espacio donde se guarda
```

```
  bloque = 1;

    estado = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A,
bloque, &key, &(mfrc522.uid));

  if (estado != MFRC522::STATUS_OK) {

    Serial.print(F("autennticado fallido: "));

    Serial.println(mfrc522.GetStatusCodeName(estado));

    return;

  }

  else Serial.println(F("autenticado con exito: "));

  // Escribir bloque

  status = mfrc522.MIFARE_Write(bloque, array, 16);

  if (estado != MFRC522::STATUS_OK) {

    Serial.print(F("failo: "));

    Serial.println(mfrc522.GetStatusCodeName(status));

    return;}

  else Serial.println(F("exito: "));

  Serial.println("coloque una nueva tarjeta en el lector");

  Serial.println(" ");

  mfrc522.PICC_HaltA(); // detener detección del lector RFID

  mfrc522.PCD_StopCrypto1();  // parar el cifrado de autenticación}
```