Informe Final del Proyecto Fast - Tap

Andrés Eduardo Villamarin Olmos, Danish Sharok Alam Rojas, Paula Andrea Castiblanco Arevalo, Brayan Ferney González Velasco, Daniel Esteban Muñoz Espinosa, José Nicolas Olarte Cortes, Laura Catalina Rozo García, Luis Pablo Valle García.

Comunicaciones Digitales. Ingeniería en Telecomunicaciones, Universidad Militar Nueva Granada.Bogotá D.C. Colombia.

mrvillagamer@gmail.com

***Resumen*—En este proyecto se integraron diversos conceptos aprendidos a lo largo de la carrera, con la finalidad de fomentar el trabajo colaborativo entre todos los integrantes del curso. Se asignaron funciones específicas a cada miembro del equipo, permitiendo coordinar y ensamblar de manera eficiente las tareas asignadas. Esto resultó en el diseño de un juego de preguntas y respuestas que integran componentes de hardware y software. Se desarrolló una interfaz atractiva que ofrece una experiencia diferenciada para el usuario, incentivando el uso y la interacción con el juego.**

***Abstract*-- In this project, various concepts learned throughout the course of our studies were integrated to foster collaborative work among all course participants. Specific functions were assigned to each team member, allowing for the efficient coordination and assembly of assigned tasks. This resulted in the design of a question-and-answer game that integrates hardware and software components. An attractive interface was developed, offering a unique user experience that encourages the use and interaction with the game.**

I. Introducción

Este informe presenta un desafío interesante y enriquecedor, donde como estudiantes se cuenta con la oportunidad de integrar los conocimientos en diversas áreas, tales como electrónica, programación, comunicaciones inalámbricas y diseño de interfaz de usuario. A través del trabajo colaborativo en equipos, se buscará alcanzar un resultado innovador y funcional que no solo consolide los conocimientos teóricos, sino que también estimule la creatividad y el trabajo en equipo.

En este informe, se expondrán detalladamente los procesos de diseño, desarrollo e implementación del juego, así como los desafíos enfrentados y las estrategias adoptadas para superarlos, haciendo uso de tecnologías como Python, Arduino UNO, Docker.

II. Objetivos

A. Objetivo General

Diseñar un juego de preguntas y respuestas que funcione a través de una red inalámbrica, integrando tanto componentes de hardware como de software de manera efectiva.

B. Objetivos Específicos

- Desarrollar una interfaz de usuario intuitiva y atractiva que facilite la experiencia de juego y promueva la participación del usuario.
- Fomentar la creatividad y la innovación en el diseño 3D
- Fomentar el trabajo en equipo y la colaboración entre los estudiantes, asignando roles y responsabilidades claramente definidos para cada miembro del grupo.

III. Marco Teórico

A. FLASK

Es un framework facilitador para el desarrollo de aplicaciones web, destacándose por su sencillez y adaptabilidad. Esta característica permite a los desarrolladores edificar aplicaciones web sólidas de manera rápida y con un código más conciso. Ofrece las herramientas y tecnologías esenciales para la creación de una aplicación web, incluyendo el redireccionamiento de URL, el manejo de solicitudes y respuestas, y la integración con bases de datos [1].

Fig. 1. Flask.

B. JSON

JSON (JavaScript Object Notation) es un formato de texto que se utiliza ampliamente para el intercambio de datos entre el backend y el frontend en el desarrollo web. JSON es útil porque los navegadores web pueden interpretar fácilmente el formato JSON. Cuando el frontend recibe datos en formato JSON del backend, puede utilizar estos datos para actualizar dinámicamente el contenido de la página web sin necesidad de recargar toda la página. Esto es especialmente útil en las

aplicaciones web modernas que requieren una comunicación constante y en tiempo real entre el servidor (backend) y el cliente (frontend) [2].

Fig. 2. JSON.

C. JQUERY

Es una biblioteca de JavaScript que acelera y simplifica el desarrollo de aplicaciones y páginas web. A pesar de permitir a los desarrolladores escribir un código JavaScript más reducido, no compromete su funcionalidad completa. jQuery hace más sencillo el manejo del Modelo de Objetos del Documento (DOM), la gestión de eventos, la generación de animaciones y la ejecución de llamadas AJAX [3].

Fig. 3. jQuery.

D. AJAX

Es un grupo de técnicas de desarrollo web que habilitan a las aplicaciones web para operar de forma asíncrona. Esto implica que AJAX tiene la capacidad de transmitir y obtener datos del servidor en segundo plano, eliminando la necesidad de recargar la página completa. AJAX es esencial para el desarrollo de aplicaciones web modernas y dinámicas, ya que permite interacciones en tiempo real entre el usuario y el servidor [4].

Fig. 4. Ajax.

E. Protocolo I2C

El bus de comunicaciones I2C (nombrado a veces como I cuadrado C, I^2) es un protocolo que se efectúa por medio de dos hilos. A través de estos dos hilos pueden conectar diferentes dispositivos donde algunos de ellos serán maestros en cuanto muchos otros dispositivos serán esclavos.

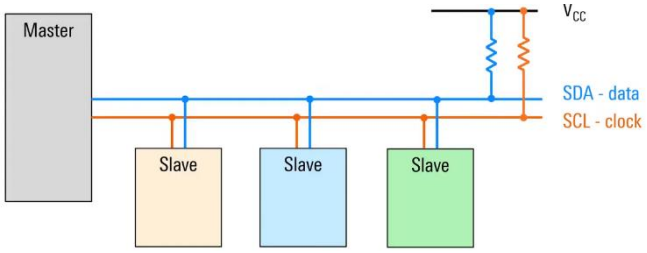

Fig. 5. Distribución bus I2C.

En la figura 5, se puede observar la distribución de los dispositivos en el bus I2C, se conectan los mismos pines entre los esclavos y el maestro.

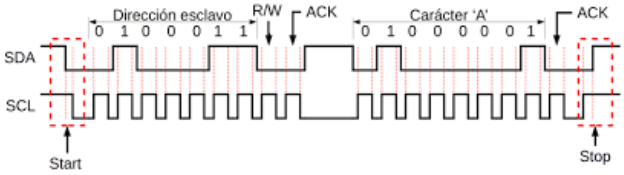

Fig. 6. Protocolo I2C.

La figura 6, muestra la manera en que opera el protocolo, al inicio el esclavo reclama el canal al enviar una señal de reloj, esto hace que se pueda enviar la información, al inicio se envía la dirección de destino y la operación (si escribir o leer), después se envía los datos y por último cada instancia de mensaje termina con un ACK para determinar si la comunicación fue exitosa al determinar una respuesta.

En este proyecto el protocolo I2C se emplea para comunicar el Arduino con la pantalla, esta se conecta los pines SCL al pin A5 del Arduino y el pin SDA se conecta al pin A4 del mismo, en este caso se maneja la dirección 0x3C para la pantalla.

F. Protocolo SPI

SPI es un protocolo de comunicación síncrono de tipo maestro-esclavo, donde los datos del maestro o del esclavo se sincronizan en el flanco de reloj ascendente o descendente, dependiendo el modo de operación. La interfaz SPI puede ser de 3 o 4 hilos, permitiendo lograr una comunicación bidireccional simultánea llamada full-duplex (4 hilos).

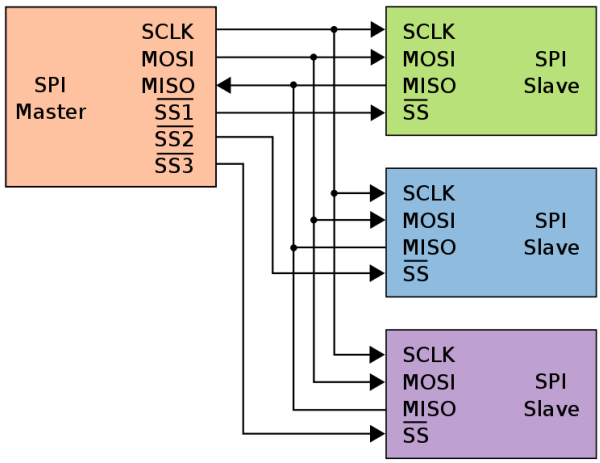

Fig. 7. Conexión protocolo SPI.

En la figura 7, se puede observar un ejemplo de conexión SPI, tanto como el reloj y los pines MOSI y MISO se conectan en modo bus y el pin de selección se conecta a cada esclavo de manera independiente, aunque también el protocolo puede emplear un solo pin de selección y emplear direcciones.

En este proyecto el protocolo SPI se emplea para comunicar el Arduino con el módulo de comunicación NRF24L01, este se conecta los pines CE al pin 8 del Arduino, el pin CSN se conecta al pin 10, el pin MOSI se conecta al pin 11, el pin MISO se conecta al pin 12 y el reloj se conecta en el pin 13.

IV. Procedimiento

A. *Metodología de juego*

1) Problema:

El problema que se debe abordar en este proyecto es la creación de un juego de preguntas y respuestas que funcione de manera fluida y confiable a través de una red inalámbrica, combinando hardware y software de manera efectiva con una cantidad de 4 módulos de transmisión (jugadores) y 1 módulo de recepción (supervisor) como lo ilustra la siguiente topología en la figura 1, por otro lado el módulo de recepción realizará una comunicación serial con un computador que será el servidor para las preguntas, en él se dará el nombre a cada uno de los transmisores y subirán las preguntas que se guardaran en una base de datos conectada a una página web.

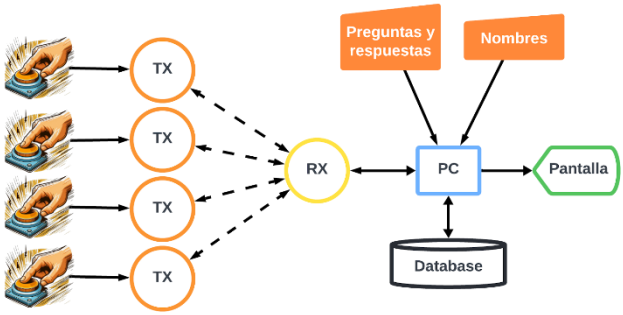

Fig. 8. Topología del juego.

2) Nombre del juego:

El nombre del juego elegido es "FashTap", con un enfoque especial en la agilidad y la atención requerida por cada equipo al momento de presionar el botón, ya que esto determina su turno en la cola de respuestas.

3) Reglas del juego:

Mantener el orden y ubicación asignada para cada equipo, evitando el ruido en la sala de juego solo tendrá la oportunidad de interrumpir si tiene el turno de respuesta. Aquel jugador que no respete esta regla será sacado de la sala de juego.

Si un equipo presiona el botón "Fast-Tap" pero proporciona una respuesta incorrecta, no tendrá más oportunidad de seguir dando más respuestas, solo tiene una única vez para responder por cada pregunta.

Cada equipo tendrá un turno para responder a una pregunta. El orden de los turnos se determinará por quién presiona primero el botón.

Los equipos acumularán puntos por cada respuesta correcta. El equipo con la mayor cantidad de puntos al final del juego será declarado ganador.

V. Pruebas

El apartado de pruebas está enfocado en la elección de los parámetros de la mejor comunicación. Se realizaron dos pruebas, una enfocada en la medición del parámetro S11 para seleccionar la frecuencia más adecuada de transmisión en los canales disponibles del NRF23L01 y la otra enfocada en envió de un ACK por defecto con la medición del tiempo de dicho envío.

A. *Prueba de Antena Omnidireccional SMA - M*

Como primer hito de este desarrollo se procede a una búsqueda de la información de la antena. La antena es de característica Omnidireccional con un conector SMA macho y su funcionalidad se encuentra en las comunicaciones inalámbricas a unas distancias experimentales de 70m máximo en pruebas realizadas anteriormente en el transcurso de la materia, aunque esa distancia máxima no va a ser usada en el proyecto y no cuenta como una prueba relevante. La impedancia de entrada de la antena es de 50Ω manejando una radiación omnidireccional como su nombre indica. La característica más importante que deseamos medir es la frecuencia de operación de la antena ya que esta opera en el rango de 2400 – 2500 Mhz. Lo deseado de la primera prueba es realizar la medición del parámetro S11 para el barrido de las frecuencias del analizador de espectro y así lograr la elección de la frecuencia donde más se aproveche la frecuencia.

Los equipos usados son los siguientes:

- Antena omnidireccional SMA – M.
- Analizador de espectro.
- Puente VSWR.

El primer paso es la realización de las conexiones de la siguiente manera:

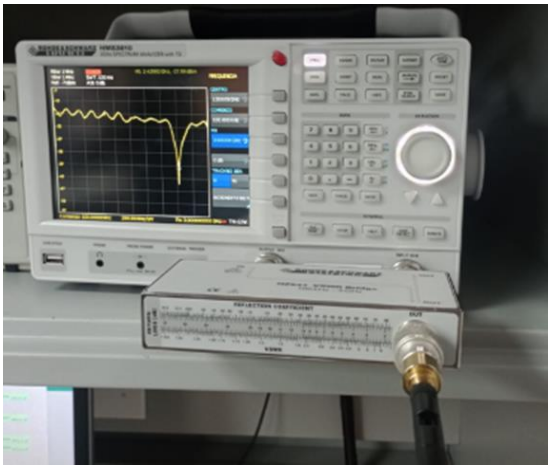

Fig. 9. Montaje de Primera Prueba.

Realizado el montaje se procede a buscar la frecuencia dando un barrido de frecuencias hasta encontrar la indicada, obteniendo lo siguiente:

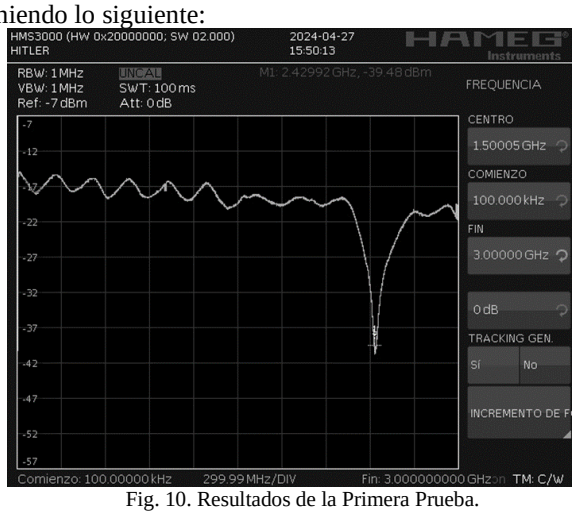

Fig. 10. Resultados de la Primera Prueba.

Lo siguiente es ubicar un marcador donde se nos entrega la gráfica comenzando en 2.4GHz donde es el comienzo de la frecuencia de operación de la antena para así ir subiendo el cursor con el fin de posicionar el menor return loss entregado obteniendo así que en la frecuencia 2.42992GHz se encuentra un valor de potencia de -39.48dBm. Pero para ser más precisos aún se buscó aumentar el tamaño de la visualización en el analizador de espectro configurando una frecuencia central de 2.4GHz, un span de 700MHz y realizando el mismo procedimiento del cursor que se realizó anteriormente obteniendo una medición de return loss de -40.56dBm en una frecuencia de 2.42760GHz como se puede evidenciar en la siguiente figura:

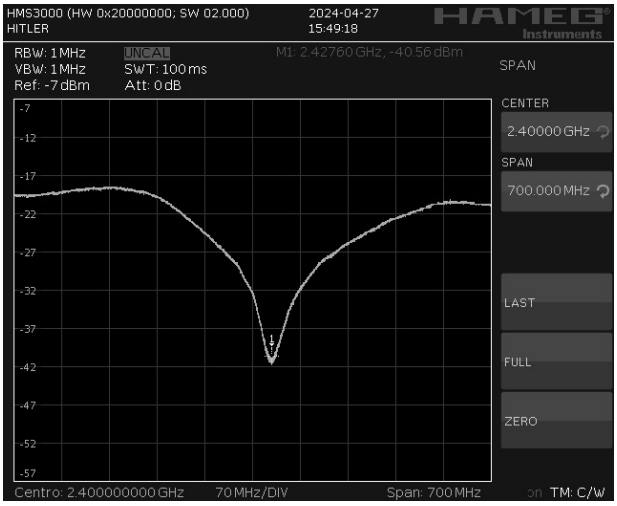

Fig. 11. Resultados Preciso de la Primera Prueba.

Al realizar la prueba se procedió a manejar el canal 27 (2.427GHz) del NRF24L01 con el fin de acercar la configuración a la frecuencia más optima de operación de la antena.

B. Prueba de Medición del Tiempo de Envió del Transceptor

Esta prueba busca realizar la medición de envió en el transceptor con el fin de reducir los tiempos y optimizar el código final del microcontrolador dependiendo de un envío ACK o no. Esta prueba se divide en 6 sub – pruebas que gracias a los resultados guiaran a los diferentes equipos para la elección de un envío ACK en la transmisión de los datos.

1) Sub – Prueba #1: Envió de 1 Byte con ACK por Defecto con Receptor.

Para la realización de la primera parte de estas pruebas fue necesario el uso del Arduino y el montaje de la shield preparados anteriormente con el fin de realizar una pequeña comunicación entre dos Arduinos, capturar el tiempo de envío de 10 muestras y calcular su promedio.

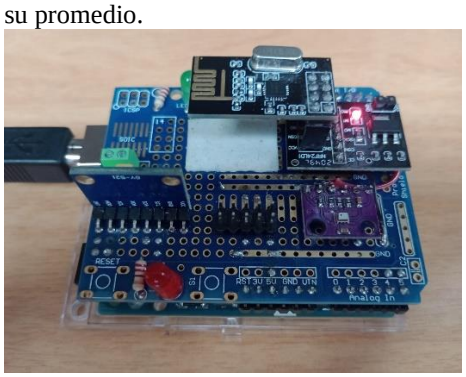

Fig. 12. Montaje para la Realización de las Sub - Pruebas.

Se realizaron dos códigos, uno para la transmisión y otro para la recepción de los datos ya que no es necesario un transceptor por el momento. El código para el envió de datos es el siguiente donde se explica cada línea de código en la figura mostrada a continuación.

```ino
// Importar librerias.
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(8, 10);  // Configuración de los puertos CE y CSN.
const byte address[6] = "00001"; // Configuración de la Pipe o dirección.

void setup () {
  Serial.begin (115200); // Configuración de los baudios.
  Serial.print ("Inicializar: "); // Imprime el mensaje de inicialización.
  Serial.println (radio.begin()); // Imprime el valor para indicar la inicialización.
  radio.setChannel(27); // Configuración del canal.
  radio.setPALevel (RF24_PA_MAX); // Potencia maxima de transmisión.
  radio.setDataRate (RF24_250KBPS); // Configuración de la transmisión de los datos.
  radio.openWritingPipe (address); // Declaración de la dirección donde se envian los datos.
  radio.stopListening (); // Quita la lectura del NRF24L01.
}

void loop () {
  unsigned long startTime = micros (); // Inicia la medición del tiempo.
  radio.write ("A", 1); // Envió de la señal a 1 Byte.
  unsigned long endTime = micros (); // Pausa la medición.
  unsigned long elapsedTime = endTime - startTime; // Calcula el tiempo de envió.
  Serial.print ("Tiempo de envìo (us): "); // Imprime el mensaje del tiempo de envió.
  Serial.println (elapsedTime); // Imprime el valor del tiempo medido.
  delay (1000); // Pausa 1 segundo antes de realizar la medición de nuevo.
}
```

Fig. 13. Código de Envió de Datos para la Sub - Prueba #1.

En la recepción de los datos el código es el siguiente:

```ino
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(8, 10);  // CE, CSN
const byte address[6] = "00001";

void setup () {
 Serial.begin (115200); // Configuración de los baudios.
  Serial.print ("Inicializar: "); // Imprime el mensaje de inicialización.
  Serial.println (radio.begin()); // Imprime el valor para indicar la inicialización.
  radio.setChannel (27); // Configuración del canal.
  radio.setPALevel (RF24_PA_MAX); // Potencia maxima de transmisión.
  radio.setDataRate (RF24_250KBPS); // Configuración de la transmisión de los datos.
  radio.openReadingPipe (0, address); // Declaración de la dirección donde se envian los datos.
  radio.startListening (); // Inicia la recepción.
}

void loop () {
  // Función para indicar si se estan recibiendo los datos.
  if (radio.available ()) {
    char receivedData; // Declaración de variable donde se guardaran los datos.
    radio.read (&receivedData, sizeof (receivedData)); // Lee el dato y lo guarda en la variable.
    Serial.print ("Dato recibido: "); // Imprime el mensaje del dato recibido.
    Serial.println (receivedData); // Imprime el valor del dato recibido.
    }
}
```

Fig. 14. Código de Envió de Datos para la Sub - Prueba #1.

El código de transmisión esta configurado para medir el tiempo de envió del mensaje con un ACK por defecto y con el receptor respectivamente. Al correr el código se obtienen los siguientes resultados.

```
COM13 (Arduino/Genuino Uno)

Tiempo de envìo (us): 5064
Tiempo de envìo (us): 2096
Tiempo de envìo (us): 2096
Tiempo de envìo (us): 2096
Tiempo de envìo (us): 5056
Tiempo de envìo (us): 5060
Tiempo de envìo (us): 8032
Tiempo de envìo (us): 5064
Tiempo de envìo (us): 5064
Tiempo de envìo (us): 2096
```

Fig. 15. Resultados para la Sub - Prueba #1.

Lo último es la realización del cálculo promedio de los datos entregados en la medición con la función micros () con la siguiente ecuación.

$$\bar{X} = \frac{5064+2096+2096+2096+5056+5060+8032+5064+5064+2096}{10} = 4172{,}4\mu s \qquad (1)$$

Este valor se comparará con los resultados de las sub – pruebas siguientes.

2) Sub – Prueba #2: Envió de 32 Bytes con ACK por Defecto con Receptor

En esta sub – prueba se realizará el mismo procedimiento que en la anterior solo que para esta ocasión el código de transmisión enviará 32 bytes para revisar si hay algún cambio en el tamaño del mensaje a enviar.

```ino
// Importar librerias.
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(8, 10);  // Configuración de los puertos CE y CSN.
const byte address[6] = "00001"; // Configuración de la Pipe o dirección.

void setup () {
  Serial.begin (115200); // Configuración de los baudios.
  Serial.print ("Inicializar: "); // Imprime el mensaje de inicialización.
  Serial.println (radio.begin()); // Imprime el valor para indicar la inicialización.
  radio.setChannel (27); // Configuración del canal.
  radio.setPALevel (RF24_PA_MAX); // Potencia maxima de transmisión.
  radio.setDataRate (RF24_250KBPS); // Configuración de la transmisión de los datos.
  radio.openWritingPipe (address); // Declaración de la dirección donde se envian los datos.
  radio.stopListening (); // Quita la lectura del NRF24L01.
}

void loop () {
  unsigned long startTime = micros (); // Comienza la medición.
  char dataToSend [32]; // Variable con la cantidad de datos a enviar.
  // Condicional para enviar 32 Bytes de la letra A.
  for (int i = 0; i < 32; i++) {
    dataToSend [i] = 'A';
    }
  radio.write (dataToSend, sizeof (dataToSend)); // Se escriben los datos a enviar.
  unsigned long endTime = micros (); // Se termina la medición.
  unsigned long elapsedTime = endTime - startTime; // Se calcula el valor de la medición.
  Serial.print ("Tiempo de envio (us): "); // Se imprime el mensaje.
  Serial.println (elapsedTime); // Se imprime el valor calculado.
  delay (1000); // Se espera un 1 segundo para realizar el proceso de nuevo.
}
```

Fig. 16. Código de Transmisión para la Sub – Prueba #2.

Los resultados de la medición son los siguientes.

```
COM13 (Arduino/Genuino Uno)

Tiempo de envìo (us): 16888
Tiempo de envìo (us): 5080
Tiempo de envìo (us): 8016
Tiempo de envìo (us): 2112
Tiempo de envìo (us): 8020
Tiempo de envìo (us): 19848
Tiempo de envìo (us): 5080
Tiempo de envìo (us): 5080
Tiempo de envìo (us): 2112
Tiempo de envìo (us): 5080
```

Fig. 17. Resultados de la Sub – Prueba #2.

Como siguiente paso se procede a calcular el promedio de los resultados arrojados por las 10 ultimas muestras de esta sub – prueba.

$$\bar{X} = \frac{16888+5080+8016+2112+8020+19848+5080+5080+2112+5080}{10} = 7731{,}6\mu s \qquad (2)$$

3) Sub – Prueba #3: Envió de 1 Byte con ACK por Defecto sin Receptor

En esta sub – prueba se realizará la medición de tiempo de envió para el envío de 1 byte, pero sin el receptor. Es decir, se usará el código de la (Fig. 5) en el transmisor y se procederá a enviar los datos respectivamente. Los resultados obtenidos para esta medición son los siguientes.

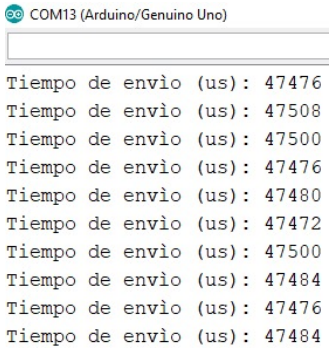

Fig. 18. Resultados de la Sub – Prueba #3.

Nuevamente se realiza el calculo promedio para estos datos dando como resultado el siguiente valor.

$$\bar{X} = \frac{47476+47508+47500+47476+47480+47472+47500+47484+47476+47484}{10} = 47485{,}6\mu s \qquad (3)$$

4) Sub – Prueba #4: Envió de 32 Bytes con ACK por Defecto sin Receptor

Con siguiente a la sub – prueba #3 se realiza la misma medición, el cambio que presenta la prueba es con respecto al tamaño de los datos que se envían ya que para esta ocasión son 32 bytes los que se van a enviar, pero sin el receptor como en el caso anterior. El código para usar es el mostrado anteriormente en la (Fig. 8) obteniendo los siguientes resultados.

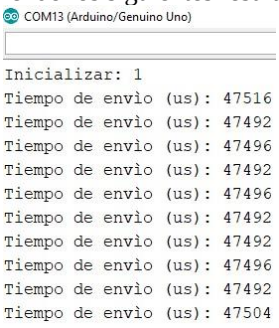

Fig. 19. Resultados de la Sub – Prueba #4.

Como en las anteriores sub – pruebas se desarrolla el promedio de los resultados de la siguiente forma.

$$\bar{X} = \frac{47516+47492+47496+47492+47496+47492+47492+47496+47492+47504}{10} = 47506{,}8\mu s \qquad (4)$$

5) Sub – Prueba #5: Envió de 1 Byte Sin ACK

Esta sub -prueba se trata de enviar un 1 byte sin ACK es decir que no es necesario el uso de un receptor. El código para implementar se mostrará más adelante donde se usó el comando "radio.setAutoAck (false);" para poder deshabilitar el ACK por defecto y tomar los siguientes resultados.

```
// Importar librerias.
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(8, 10);  // Configuración de los puertos CE y CSN.
const byte address[6] = "00001"; // Configuración de la Pipe o dirección.

void setup () {
  Serial.begin (115200); // Configuración de los baudios.
  Serial.print ("Inicializar: "); // Imprime el mensaje de inicialización.
  Serial.println (radio.begin()); // Imprime el valor para indicar la inicialización.
  radio.setChannel (27); // Configuración del canal.
  radio.setPALevel (RF24_PA_MAX); // Potencia maxima de transmisión.
  radio.setDataRate (RF24_250KBPS); // Configuración de la transmisión de los datos.
  radio.openWritingPipe (address); // Declaración de la dirección donde se envian los datos.
  radio.stopListening (); // Quita la lectura del NRF24L01.
  radio.setAutoAck (false); // Deshabilita el ACK por defecto.
}

void loop () {
  unsigned long startTime = micros (); // Inicia la medición.
  radio.write ("A", 1); // Envia 1 byte.
  unsigned long endTime = micros (); // Para la medición.
  unsigned long elapsedTime = endTime - startTime; // Calcula el tiempo de envió.
  Serial.print ("Tiempo de envio (us): "); // Imprime el mensaje de tiempo.
  Serial.println (elapsedTime); // Imprime el valor de tiempo.
  delay (1000); // Espera 1 segundo antes de inciar el proceso de nuevo.
}
```
Fig. 20. Código de Transmisión de la Sub – Prueba #5.

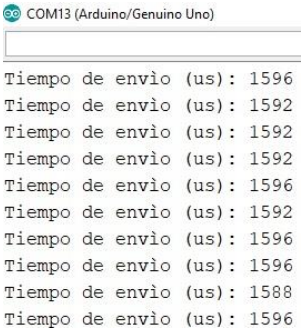

Fig. 21. Resultados de la Sub – Prueba #5.

Se realiza el calculo del promedio de los tiempos medidos. Obteniendo así el siguiente valor.

$$\bar{X} = \frac{1596+1592+1592+1592+1596+1592+1596+1596+1588+1596}{10} = 1593{,}6\mu s \qquad (5)$$

6) Sub – Prueba #6: Envió de 32 Bytes con ACK Sin ACK

Para la última sub – prueba se realizó el envío sin el ACK por defecto como se mostró en la prueba anterior con un tamaño de mensaje de 32 bytes agregándole al código de la (Fig. 8.) la línea de código para quitar el ACK como se mencionó anteriormente. Los resultados obtenidos son los siguientes.

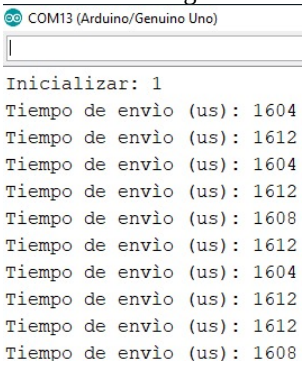

Fig. 22. Resultados de la Sub – Prueba #6.

Por último, se calcula el tiempo promedio de esta ultima prueba para dar paso a la conclusión de esta prueba.

$$\bar{X} = \frac{1604+1612+1604+1612+1608+1612+1604+1612+1612+1608}{10} = 1608{,}8\mu s \qquad (6)$$

Esta prueba evidencio tres factores. El primero se relaciona con el tiempo de envío cuando se transmite 1 byte o 32 bytes, por ejemplo, en las dos primeras sub – pruebas se evidencia que al enviar 32 bytes el tiempo de envió aumenta evidentemente por lo que se puede demostrar que entre más caracteres se agreguen al mensaje mas se demora el mensaje a enviarse.

El segundo factor se relaciona al uso o no de un receptor ya que para esta conclusión se deben revisar las sub – pruebas #1 y #3 que corresponden al envío de 1 solo byte con y sin receptor donde este ultimo presentara un mayor tiempo de envió ya que al tener un ACK por defecto configurado este aumentara el tiempo de espera para recibir dicho ACK por mucha diferencia a cuando si hay un receptor evidentemente.

El ultimo factor a identificar y por el cual se guiará el proyecto es el quitar el ACK por defecto ya que las sub – pruebas #5 y #6 muestran un tiempo de envío mucho menor que las demás sub – pruebas anteriores. Esto se debe a que el transmisor ya no se encuentra esperando a un ACK para volver a enviar, su código esta diseñado para enviar el dato cada que se cumple el bucle. La única diferencia en estas dos últimas sub – pruebas es el tiempo de envío que varia por la cantidad de datos transmitida ya que como se mencionó con anterioridad al enviar 32 bytes el tiempo aumenta, pero ya no están considerablemente alto como en las anteriores sub – pruebas.

VI. Hardware

Para el desarrollo del diseño esquemático y modelado 3D del hardware, se utilizaron diferentes aplicaciones:

A. Tinkercad

Esta herramienta se eligió por su accesibilidad y facilidad de uso, ideal para crear prototipos rápidos de modelos 3D. Tinkercad permite a los diseñadores visualizar y ajustar sus ideas de manera interactiva, lo cual es perfecto para etapas iniciales del diseño.

B. Solid Edge

Para trabajos más complejos de modelado 3D, se utilizó Solid Edge. Esta aplicación es conocida por su capacidad para manejar proyectos de ingeniería detallados y precisos, lo que la hace adecuada para desarrollar componentes de hardware con alta exactitud y funcionalidad.

C. KiCad

En el diseño esquemático y la creación de circuitos impresos (PCBs), se recurrió a KiCad. Esta herramienta es robusta y ofrece completas funcionalidades para el diseño electrónico, lo que permite la elaboración de esquemas detallados y eficientes de circuitos.

Fig. 23. Aplicaciones de diseño.

La combinación de estas plataformas permitió un desarrollo integral y detallado del hardware, abarcando desde la conceptualización inicial hasta la implementación precisa de los diseños.

D. Diagrama y esquema de conexión

1) 2.1.1 Digrama Técnico - Transmisor

El esquema de conexiones permite tener una representación visual de cómo se encuentran interconectados cada uno de los elementos dentro del sistema del transmisor.

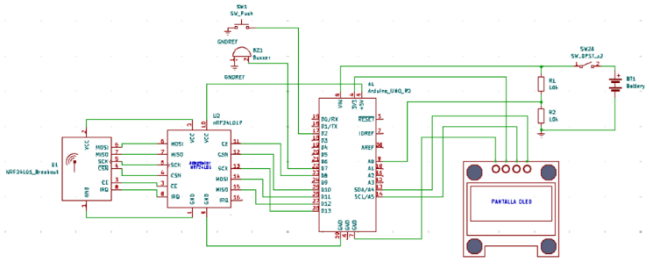

Fig. 24. Esquema de conexiones.

Se tienen las siguientes conexiones para cada elemento:

Elemento:	Pin Arduino:
NRF24LO1+Adaptador	8,10,11,13,12
Buzzer	7, GND
Pulsador	2, GND
Pantalla	A4,A5,3.3V,GND
Baterías	Vin,GND

Tabla 1. Conexiones para cada elemento.

2) 2.1.2 Diagrama Técnico - Receptor

Diagrama Técnico de conexiones del receptor:

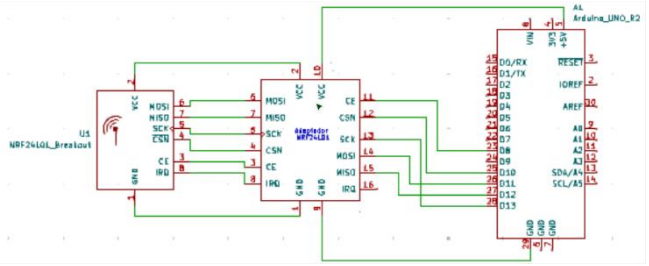

Fig. 24. Diagrama conexiones receptor.

Se tienen las siguientes conexiones para cada elemento:

Elemento:	Pin Arduino:
NRF24LO1+Adaptador	8,10,11,13,12

Tabla 2. Conexiones para elementos.

Teniendo en cuenta que el módulo físico tiene un escudo que encierra cada conexión al Arduino “Shield”. La función principal de este escudo no es defensiva, sino ofensiva:

proporciona funcionalidades adicionales y amplía las capacidades del microcontrolador al que está conectado. En otras palabras, actúa como un ángel guardián colocado sobre la placa base del microcontrolador, y trae como obsequio características específicas del proyecto: conectividad inalámbrica, más sensores, un actuador (como la pantalla OLED).

E. Modelo 3D

1) Modelo Transmisor

Se digitaliza todos los elementos con mediciones reales, de tal manera que se pudiera tener una manipulación más fácil para tener una lluvia de idea a la hora de tomar una decisión basada en la ubicación de cada uno de los elementos en la placa, teniendo como resultado la siguiente imagen diseñada en 3D esta será la estructura del hardware de cada uno del módulo la cual estará contenida en una caja diseñada posteriormente.

Fig. 25. Imagen recuperada de TinkerCad.

Así mismo, permite tomar las medidas aproximadas de los cables que permiten tener esta ubicación para cada uno de los elementos y sobre ellos se procede a realizar la caja.

2) Caja

La caja es un elemento diseñado en un material sólido pero fácil de manipular, por lo que se seleccionó cartón. A este material se le aplicaron las mediciones necesarias para cubrir los elementos requeridos. Además, se consideró que debía permitir el acceso a las baterías y al Arduino para facilitar el ensamblaje y desensamblaje de los componentes en caso de necesitar reparaciones o ajustes.

Por otra parte, se abordó el tema de la pantalla, para la cual se diseñó una superficie inclinada que facilita el rango de visión del usuario. Asimismo, el botón se ubicó en un lugar accesible, ya que es una parte fundamental para el propósito del juego.

Fig. 26. Imagen recuperada del laboratorio.

F. Modelado 3D TinkerCad

Para el modelo 3D, empleando TinkerCad, se tomaron las mediciones y se realizaron observaciones sobre los aspectos que quedaron mal o que necesitaban ajustes en el modelo de la caja. Estos ajustes se pueden realizar de manera más sencilla en el software. En este caso, se ajustó el tamaño del botón, modificándolo por un elemento más grande. Además, se evaluó cómo se unieron los elementos y se consideraron aspectos de espacio dentro del modelo para su manipulación, garantizando que sea sencillo retirar y gestionar los componentes internos.

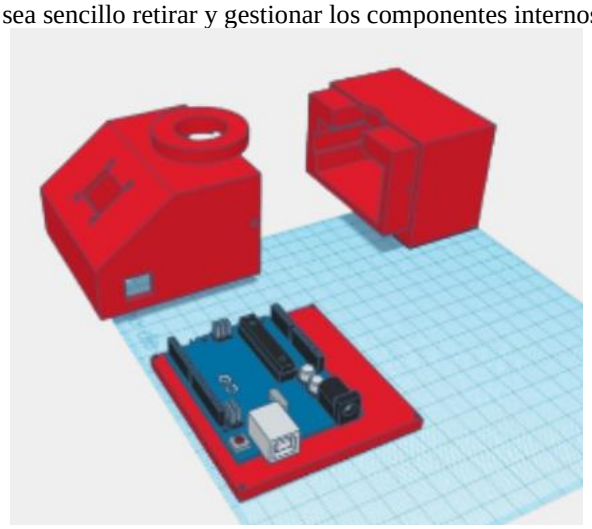

Fig. 27. Imagen recuperada del software TinkerCad.

G. Modelado 3D Solid Edge

El uso de este software se dio en el momento en que, al tener el modelo ya en una fase cercana al final, se buscó darle detalles y otros elementos que mediante TinkerCad no era posible o resultaba un proceso más complejo. Por lo tanto, se evaluó el uso de esta herramienta debido a temas de licencia y porque permite tener más opciones de edición del modelo 3D. Este software se especializa en este campo, por lo que se trasladó el diseño base, en cuanto a mediciones y esquema general, a Solid Edge.

En Solid Edge se comenzaron a realizar detalles de ajustes en las tapas para interactuar con los elementos, así como elementos externos como el botón y el switch de encendido. Además, se agregaron opciones de fijación de los materiales, como las baterías, la pantalla, el Arduino y su shield. Se ajustaron los tamaños de la estructura en general y se añadieron detalles como el nombre del juego y salidas de sonido.

a. 

Fig. 28. Imagen vista alzado 3D.

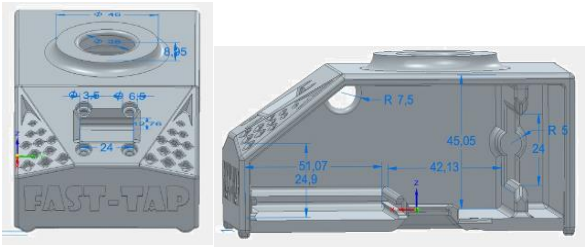

Fig. 29. Imagen vista planta y perfil derecho.

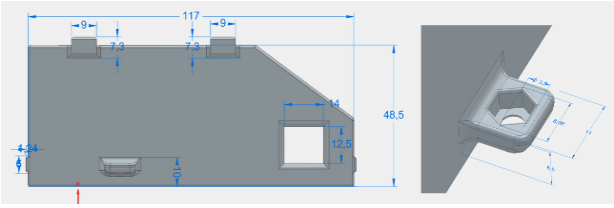

Figura 30. Imagen vista tapa y soporte de tornillo.

H. Modelado 3D Solid Edge Receptor

Con toda la implementación de los transmisores, se tomó el modelo base para realizar el receptor de manera más sencilla, debido a que no cuenta con la misma cantidad de elementos comparado con el transmisor. Se tomó parte de la estructura original y se eliminaron las partes de fijación y los espacios de elementos que no se requerían, como las baterías, la pantalla, el pulsador y el switch de encendido.

El diseño se planteó en forma de cubo, lo cual permite que, al momento de imprimir, no se requieran tantos soportes. Además, se cambió la ubicación de los tornillos de fijación y se dejaron los soportes internos para el Arduino. También se añadieron detalles como el nombre del juego y se ajustó la tapa con la que se cierra el modelo.

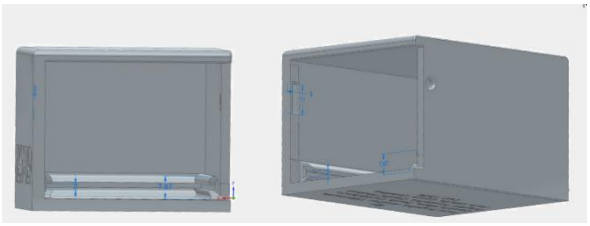

Fig. 31. Imagen del receptor vista lateral derecha y alzada 3D.

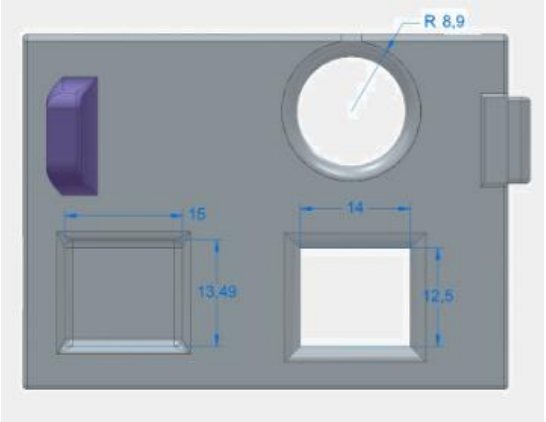

Fig. 32. Imagen tapa cierre del receptor.

I. Proceso de Impresión

Se ilustra el flujo desde el diseño digital hasta la materialización física de una pieza, subrayando la importancia de cada etapa en la fabricación mediante impresión 3D.

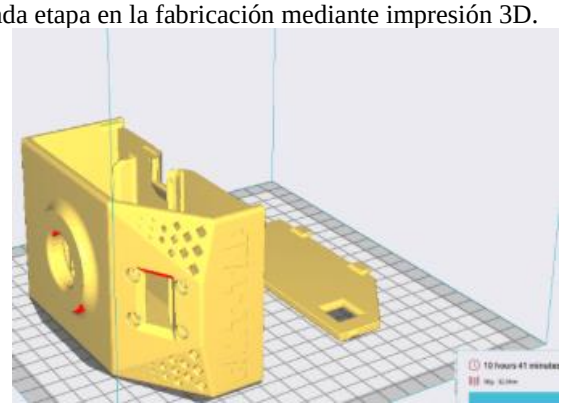

Fig. 33. Vista prototipo.

La imagen proporciona una visión clara del diseño y la configuración previa a la impresión de una pieza en 3D. Esta etapa es fundamental para asegurar que el proceso de impresión sea eficiente y que la pieza final cumpla con las especificaciones requeridas. Esta imagen resalta la importancia de la planificación detallada y la capacidad de la impresión 3D para producir piezas complejas y funcionales.

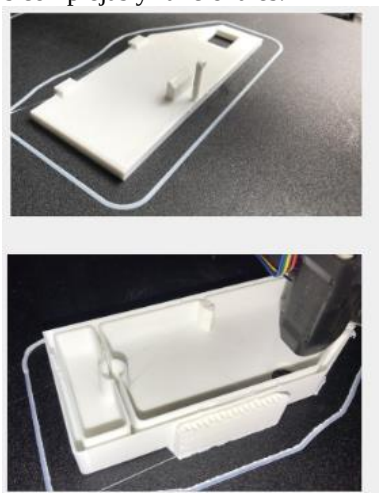

Fig. 34. Diseño en proceso de impresión.

La impresión en 3D requiere de soportes, ya que se va realizando capa por capa, como se muestra en la imagen anterior. Estos diseños se acomodaron de manera que los soportes fueran los mínimos posibles para ahorrar tiempo y material. Al momento de imprimir, los elementos salen con los soportes y otros componentes adicionales que posteriormente deben ser retirados. Una vez eliminados estos soportes, se deja la estructura con los orificios diseñados.

J. Resultados Diseño 3D

La finalización de los diseños en 3D concluyó con los cuatro transmisores y el receptor, eliminando los elementos sobrantes de los diseños, como soportes y otros componentes añadidos para poder imprimir. Posteriormente, se agregaron los elementos necesarios, como el Arduino con la shield, que encajan en los soportes mencionados en el apartado 2.2.4-5, así como las

baterías. Otros componentes, como el botón, se ajustaron mediante una rosca preexistente, y la pantalla se aseguró mediante tornillos, que se fijaron con tuercas en la parte interna.

Fig. 35. Resultado impresión 3D.

La antena se acopló en el módulo a través de un orificio, de la misma manera que el receptor. Finalmente, se operaron los diferentes botones y conectores para verificar que todo cumpliera con lo requerido.

VII. Detalles técnicos

A. Pipes del módulo NRF24L01

Para establecer comunicación entre los transceptores, se investigó el funcionamiento de las pipes con el objetivo de implementarlas en el proyecto. Los pipes son un mecanismo que facilita la transferencia de datos entre diferentes procesos, sistemas o componentes de software, actuando como canales de comunicación unidireccionales.

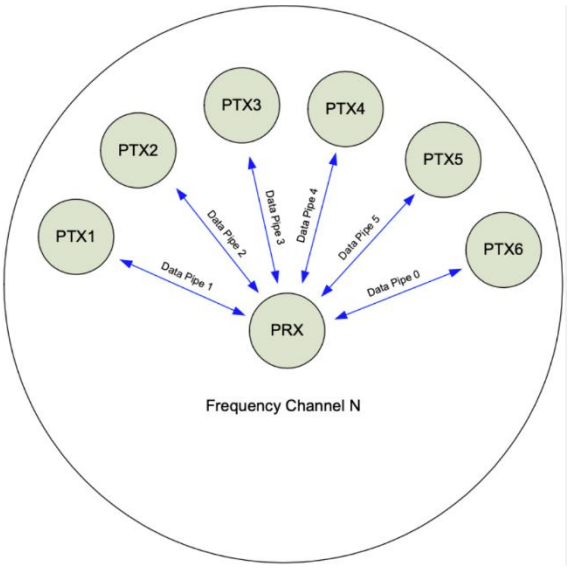

Fig. 36. Canal de Frecuencia de las Pipes.

En la imagen, se observa un conjunto de seis canales de datos paralelos (pipes), cada uno con una dirección única que los identifica. El transceptor configurado como PRx (primary Receiver) es capaz de recibir datos de estos seis pipes en un mismo canal de frecuencia. Es importante destacar que las pipes soportan un máximo de hasta seis transceptores NRF24L01 configurados en modo PTx (Primary Transmitter), los cuales pueden comunicarse con un único transceptor NRF24L01 configurado en modo PRx.

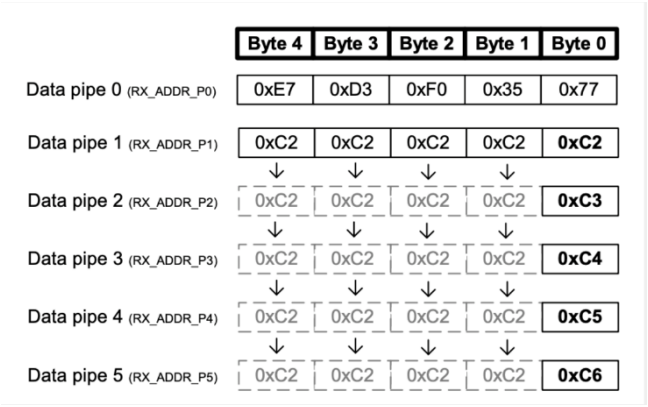

Fig. 37. Direccionamiento de las Pipes.

En la imagen, se observa el funcionamiento de las direcciones de las pipes. Se puede evidenciar que las pipes 0 y 1 almacenan una dirección completa de 5 bytes. Las pipes 2 a 5, en cambio, almacenan únicamente un byte, tomando prestados hasta 4 bytes adicionales de la pipe 1, según el ancho asignado. Las pipes 2 a 5 deben compartir la misma dirección, excepto por el primer byte, el cual debe ser único en la matriz. Es importante destacar que las direcciones de las pipes se buscan simultáneamente.

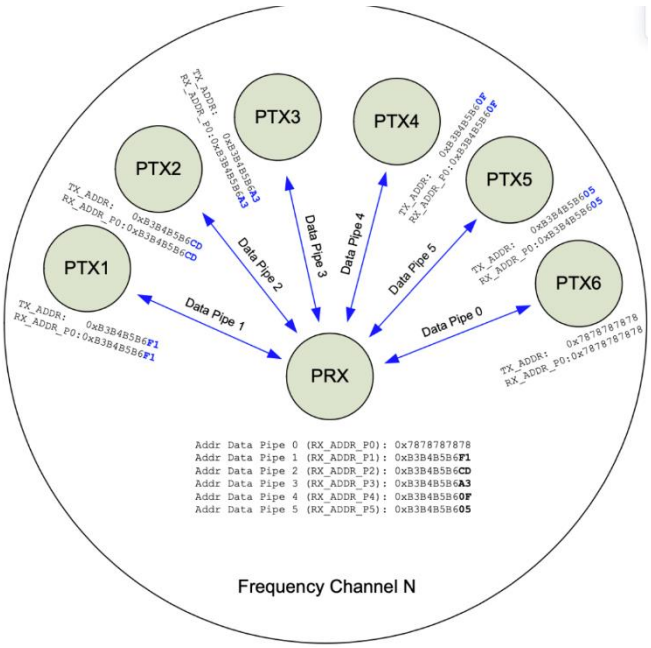

Fig. 38. Configuración de Direcciones de las Pipes.

En la imagen, se observan las direcciones asignadas a cada PTx, y se puede apreciar que el PRx tiene configuradas las direcciones de cada PTx. Dado que el PRx recibe paquetes de múltiples PTx, para garantizar que el paquete ACK se transmita

al PTx correcto, el PRx utiliza la dirección del pipe del cual recibió el paquete como dirección de transmisión al enviar el paquete ACK. Adicionalmente, cuando un pipe recibe un paquete completo, otros pipes pueden comenzar a recibir datos. Es importante destacar que, cuando varios PTx transmiten a un PRx, se puede utilizar el parámetro ARD (Automatic Retransmission Delay) para sesgar la retransmisión automática, de modo que los PTx solo se bloqueen entre sí una vez.

B. Trama de operación del módulo NRF24L01

El módulo NRF24L01 al enviar mensajes, utiliza la trama descrita en la hoja de datos, esta trama tiene el preámbulo, dirección, control de paquetes, carga útil (datos) y CRC (verificación de redundancia cíclica).

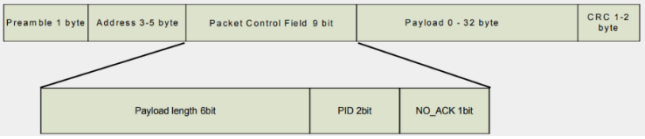

Fig. 39. Paquetes de la Trama.

El primer campo es el preámbulo y es un bit que obtiene su valor en base a la dirección, si el primer bit de la dirección es 1 el preámbulo es 1 y si es 0 el preámbulo es 0, después viene el campo de dirección que contiene la dirección de destino de la trama, la trama de control de paquetes se divide en 3 subcampos, la longitud de la carga útil (datos), el identificador (PID) y una bandera que define si el modo de auto reconocimiento está activado o no, después viene el campo de datos, donde está la información que queremos enviar, y por último el control de error (CRC), este se encarga de descartar las tramas si tienen un error haciendo que se envíen de nuevo, también trabaja con el identificador para descartar tramas duplicadas.

VIII. SOFTWARE

A continuación, se explicará todo lo relacionado con el software del proyecto, esto va desde la programación necesaria para el servidor, página web y Arduino, así como también las reglas de comunicación que todos los dispositivos deben de seguir para una correcta comunicación entre estos. Para información detallada relacionada con el software favor consultar en FastTap (github.com).

A. Reglas de Comunicación

Para la comunicación entre los dispositivos del proyecto se emplearon pautas que especifican cómo se envían y reciben cada byte de los mensajes intercambiados.

Para la comunicación del receptor al transmisor se emplean las siguientes reglas.

Identificador	Byte1	Byte2	Byte3	Byte4
Asigna usuario	0	Usuario	N/A	N/A
Iniciar nueva pregunta	1	Numero Pregunta	Puntaje Obtenido	Puntaje a Obtener
Envio posición	2	Posición	N/A	N/A
Envio turno	3	Turno	N/A	N/A
Puesto final	4	PuestoFinal	Puntaje Obtenido	N/A
Contesto correctamente	5	N/A	N/A	N/A

Fig. 40. Reglas de Comunicación #1.

Ahora para la comunicación del transmisor al receptor se emplean las siguientes reglas.

Identificador	Byte1	Byte2
Solicitar usuario	0	N/A
Envia pulso botón	1	N/A
Envia nivel batería	2	Batería

Fig. 41. Reglas de Comunicación #2.

El transmisor maneja unos datos simples como la pulsación del botón y la batería, mientras que el receptor le da información sobre el juego (puntaje, posición de respuesta y resultado de respuesta).

Para la comunicación del receptor al computador se emplean las siguientes reglas.

Identificador	Byte1	Byte2	Byte3
Solicitar lista de usuarios conectados	0	N/A	N/A
Usuario conectado	1	Usuario	N/A
Envía nivel batería	2	Usuario	Batería
Envia pulso de botón	3	Usuario	N/A

Fig. 42. Reglas de Comunicación #3.

Ahora para la comunicación del computador al receptor se emplean las siguientes reglas.

Identificador	Byte1	Byte2	Byte3	Byte4
Envia lista de usuarios conectados	0	Usuarios Conectados	N/A	N/A
Enviar puntaje jugador	1	Usuario	Puntaje Obtenido	N/A
Iniciar nueva pregunta	2	Numero Pregunta	Puntaje a Obtener	N/A
Envio posición jugador	3	Usuario	Posición	N/A
Envio turno actual	4	Turno	N/A	N/A
Contesto correctamente	5	Usuario	N/A	N/A
Puesto final	6	Usuario	PuestoFinal	PuntajeObtenido

Fig. 43. Reglas de Comunicación #4.

El receptor actúa como un puente entre el transmisor (botón) y el computador (página web), este envía los pulsos de botón y el nivel de batería, mientras que el computador le envía el puntaje y la posición de respuesta en el concurso, por último, también envía los resultados de la respuesta y también el final del concurso.

B. Arduino

Para garantizar una comunicación efectiva entre todos los dispositivos involucrados, se desarrolló un código único de Arduino siguiendo las reglas de comunicación previamente establecidas. Este código único facilita la integración y programación de los Arduinos, permitiendo controlar tanto a los múltiples transmisores como al receptor.

Como se puede observar en el diagrama de flujo en la (Fig. 44), el dispositivo identifica si debe iniciar como un transmisor o como un receptor a través de su comunicación serial por el

protocolo UART con el servidor. Esto es posible ya que los transmisores funcionan de manera portátil y, por lo tanto, no recibirán ninguna comunicación serial, mientras que el receptor sí lo hará.

En la fase inicial del programa, los dispositivos transmisores se configuran junto con el receptor a través de una pipe por defecto llamada “dpipe”, que tiene una longitud de 5 bytes. Este pipe temporal se utiliza para realizar la solicitud inicial de asignación del usuario al jugador en el transmisor. Esta asignación es crucial para identificar a los jugadores en la pantalla y también se utiliza para reconfigurar la pipe del transmisor, creando así un canal digital exclusivo para la comunicación con el receptor.

Finalmente, una vez configurada la pipe exclusivo, podemos recibir y enviar datos siguiendo las reglas de comunicación. Estos datos se procesan para mostrar información en la pantalla OLED del Arduino, acompañada de indicaciones sonoras del buzzer. De esta manera, se logra una comprensión simplificada del funcionamiento del Arduino.

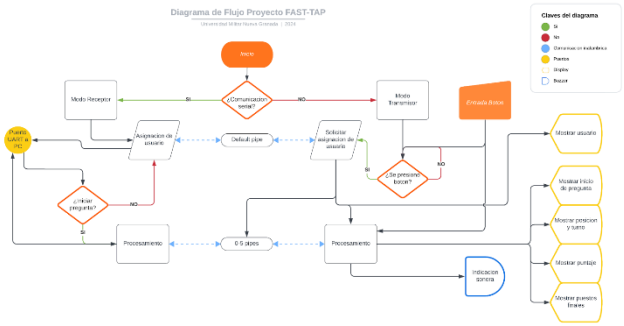

Fig. 44. Diagrama de flujo Transmisor y Receptor.

C. *Página Web*

En la construcción de la página web, se llevó a cabo un proceso meticuloso que implicó la integración de diversas tecnologías y herramientas. Entre estas, se destacan la implementación de bases de datos, el uso de Flask como framework de desarrollo web, la creación de páginas utilizando HTML para la estructura, CSS para el diseño y JavaScript para la interactividad.

Para comprender mejor la interacción entre cada uno de estos componentes y su relación visual, se elaboró un detallado diagrama de arquitectura. Este gráfico no solo ofrece una representación visual de cómo se entrelazan los distintos elementos, sino que también proporciona una visión clara de la estructura subyacente de la página web.

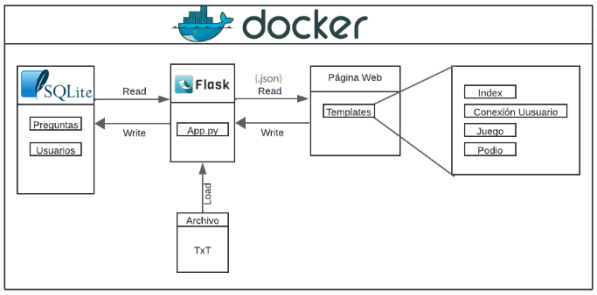

Fig. 45. Diagrama de Arquitectura de la página Web.

La integración de bases de datos desempeña un papel crucial en la funcionalidad de la página, permitiendo el almacenamiento y la gestión eficiente de datos clave. Flask, por su parte, actúa como el marco de trabajo que facilita el desarrollo rápido y la creación de rutas para la manipulación de datos y la presentación de contenido dinámico.

En cuanto a la parte visual y la experiencia del usuario, HTML se emplea para definir la estructura y el contenido de las páginas, mientras que CSS se encarga de dar estilo y formato, asegurando una apariencia atractiva y coherente en todo el sitio. Por último, JavaScript añade interactividad y dinamismo, permitiendo la creación de efectos visuales, validaciones de formularios y la implementación de funcionalidades que mejoran la experiencia del usuario.

D. *Desarrollo del Frontend*

Para la creación del frontend, se llevó a cabo un exhaustivo proceso de investigación y análisis. Inicialmente, se realizó una búsqueda minuciosa de juegos de preguntas existentes en el mercado, con el objetivo de obtener inspiración y orientación para el diseño de estos. En este contexto, se prestó especial atención a plataformas destacadas como Kahoot, reconocida por su enfoque interactivo en la formulación de preguntas y su eficaz diseño.

Tras esta fase inicial de investigación, se procedió a realizar pruebas y experimentaciones con herramientas innovadoras como ChatGPT y Copilot. Estas pruebas se centraron en la exploración de la creación de diversos templates, los cuales servirían como base para cada una de las partes fundamentales de la página. Durante este proceso, se brindaron instrucciones detalladas a estas herramientas, delineando con precisión las características y funcionalidades deseadas para cada componente.

Como resultado de este enfoque meticuloso, se desarrolló en primer lugar la página de inicio del juego. Esta página fue concebida para presentar de manera impactante el título del juego, acompañado de un llamativo botón de inicio. En cuanto al desarrollo de la interactividad utilizando JavaScript, se solicitó específicamente la implementación de una animación cautivadora. Se buscaba lograr un efecto visual de carga gradual, mediante el cual el contenido de la página se desvaneciera suavemente para luego ir apareciendo progresivamente, generando así una sensación de anticipación y expectativa en el usuario.

Fig. 46. Primer prototipo página de inicio del juego.

Como segundo template, se llevó a cabo la creación de la pestaña destinada al registro de los participantes. En este proceso, se tomó en cuenta la necesidad de incorporar cuatro botones distintos para facilitar el registro individual de cada participante. Este enfoque se alineó con la premisa de ofrecer una experiencia fluida y eficiente para los usuarios, permitiéndoles realizar el registro de manera rápida y sin complicaciones.

En la implementación de esta sección de la página web, se hizo uso de JavaScript para añadir una funcionalidad adicional y mejorar la interactividad. Específicamente, se aplicó una función que permitía cambiar el color de los botones en pantalla al ser pulsados por los usuarios. Esta característica no solo añadió un elemento visual dinámico a la página, sino que también sirvió como retroalimentación visual para los participantes, indicando de manera clara y perceptible que su acción de registro había sido reconocida por el sistema.

Fig. 47. Primer prototipo página de registro participantes.

Fig. 48. Primer prototipo página de registro participantes.

En la culminación de esta sección, se tomó la decisión estratégica de incorporar una función adicional que enriquecería la experiencia del usuario: la opción de subir un archivo de texto (.txt) para cargar preguntas personalizadas. Esta característica se diseñó con el objetivo de brindar a los usuarios una flexibilidad aún mayor al utilizar la plataforma, permitiéndoles crear y compartir cuestionarios específicos según sus necesidades y preferencias.

Al implementar esta funcionalidad, se abrió un abanico de posibilidades para los usuarios, quienes podrían ahora aprovechar una amplia variedad de contenidos de preguntas, adaptados a diferentes temas, niveles de dificultad o áreas de interés. Desde cuestionarios educativos hasta pruebas de entretenimiento, la inclusión de esta opción de carga de archivos .txt amplió significativamente el alcance y la utilidad de la plataforma.

Además, esta función no solo contribuyó a la versatilidad de la plataforma, sino que también simplificó el proceso de creación y gestión de cuestionarios por parte de los administradores. Al permitir la carga masiva de preguntas a través de un archivo de texto, se agilizó considerablemente el flujo de trabajo, reduciendo la necesidad de introducir manualmente cada pregunta individualmente. Esto no solo ahorró tiempo y esfuerzo, sino que también minimizó la posibilidad de errores humanos en el proceso.

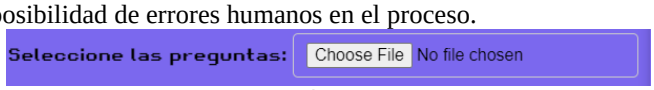

Fig. 49. Primer Prototipo Página de Registro Participantes.

Posteriormente, se procedió con la creación de la página destinada a la visualización de las preguntas. Con el objetivo de optimizar la presentación de la información y mejorar la experiencia del usuario, se implementó una estructura dividida en dos frames distintos.

En el primer frame, se dispuso una sección dedicada a mostrar a los jugadores participantes, junto con sus respectivos nombres y puntajes acumulados. Esta disposición permitió a los usuarios tener una visión rápida y clara del estado actual de la competición, facilitando así el seguimiento del desempeño de cada jugador a lo largo del juego.

En el segundo frame, se diseñó una sección específica para la presentación detallada de la pregunta en curso. Aquí, se incluyó el número de la pregunta, el enunciado de esta, el puntaje asignado a dicha pregunta y las cuatro opciones de respuesta disponibles. Esta disposición estructurada garantizó una experiencia de visualización ordenada y fácil de seguir, proporcionando a los jugadores la información necesaria de manera clara y concisa.

Además de ofrecer una presentación visualmente atractiva, la división en dos frames también contribuyó a mejorar la navegabilidad de la página, al permitir que los usuarios accedieran simultáneamente a diferentes aspectos relevantes del juego sin perder la continuidad ni la coherencia visual.

Fig. 50. Página de visualización de preguntas.

Continuando con el desarrollo, se implementó una funcionalidad clave que garantizaba la adaptabilidad del sistema a la cantidad variable de preguntas presentes en el archivo .txt cargado. Esta característica permitía que el segundo frame, destinado a la visualización detallada de las preguntas, se duplicara dinámicamente según la cantidad de preguntas

disponibles, asegurando así una presentación coherente y completa de todo el contenido.

Este enfoque flexible y escalable fue diseñado para satisfacer las necesidades cambiantes de los usuarios, independientemente de la cantidad de preguntas incluidas en el archivo de texto. Con cada duplicación del frame, se presentaba una nueva pregunta junto con su número correspondiente, el enunciado, el puntaje asignado y las opciones de respuesta, manteniendo así una disposición ordenada y coherente de toda la información.

Para completar la experiencia del usuario, se procedió con la implementación del podio, una funcionalidad crucial para visualizar a los tres participantes con mayor puntaje en el juego. Esta adición final no solo agregaba un elemento competitivo emocionante, sino que también reconocía y destacaba el rendimiento destacado de los jugadores más hábiles.

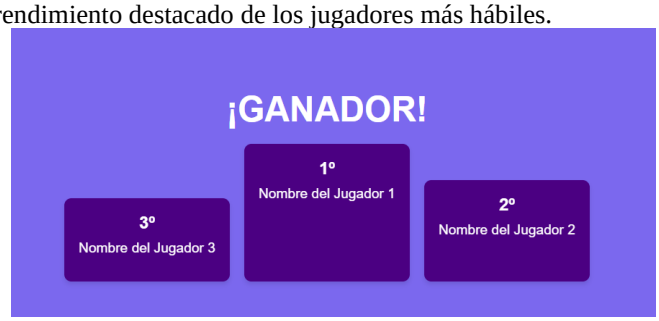

Fig. 51. Página para el Marcador final y el podio de puntuación.

E. Backend Con Python Utilizando el Framework Flask

Ya hemos obtenido el frontend, ahora debemos establecer la conexión entre la página, el Arduino y la base de datos. Sin embargo, antes de esto, tuvimos que definir las reglas de comunicación entre el Arduino RX y la computadora. Una vez establecidas estas reglas de comunicación, podemos comenzar con el registro de las páginas en el backend.

```python
@app.route("/", methods=["GET", "POST"])
def home():
    if request.method == "POST":
        dato = request.form.get("dato")
        if dato == "EMPEZAR":
            HiloRecepcionSerial.start()
            return redirect(url_for("PaginaConexionUsuarios"))
    return render_template("index.html", IniciarComSer=IniciarComSer)
```

Fig. 52. Código #1.

@app.route("/", methods=["GET", "POST"]): Esta es una ruta en Flask que se activa cuando se accede a la URL raíz (es decir, “/”) de la aplicación. La ruta acepta solicitudes de los métodos HTTP “GET” y “POST”.

```python
@app.route("/IniciarComSer")
def IniciarComSer():
    global Conectado
    if Conectado == 0:
        Conectado = IniciarComunicacionSerial()
    return jsonify({"Conectado": Conectado})
```

Fig. 53. Código #2.

Define una ruta en una aplicación Flask que se activa cuando se accede a la URL “/IniciarComSer”. La función IniciarComSer() se ejecuta cuando se accede a esta ruta.

Ahora cómo llamo esto desde el frontend.

```javascript
$.get('/IniciarComSer', function (response) {
                var Conectado = response.Conectado;
                console.log(Conectado);
                var boton = document.querySelector('.boton-estilo');
                if (Conectado == 0) {
                    boton.style.backgroundColor = 'gray';
                    boton.disabled = true;
                } else {
                    boton.style.backgroundColor = '#9c27b0';
                    boton.disabled = false;
                }
            });
```

Fig. 53. Código #3.

Como se puede observar en este código de javascript se llama a la función /InciarComSer que esta función es del backend ahi se ve que esta función nos retorna un hacia la función Comunicación Serial esta nos da todos los datos de comunicación nos indica si el arduino está activo inactivo cuantos dispositivos hay conectados.

Fig. 53. Código #2.

Al presionar Empezar se hace una solicitud POST y como se ve en la figura# ya el backend detectado que se hizo la solicitud POST y se observa que el dato es igual al ID del botón ya hecho esto nos redirecciona a la página 2 donde esta conexión usuarios.

```python
@app.route("/ConexionUsuarios.html", methods=["GET", "POST"])
def PaginaConexionUsuarios():
    if request.method == "POST":
        nombre = [None] * 5
        nombre[0] = request.form.get("input1")
        nombre[1] = request.form.get("input2")
        nombre[2] = request.form.get("input3")
        nombre[3] = request.form.get("input4")
        for Usuario in range(5):
            cursor_database.execute(
                """SELECT NumeroJugador FROM Estadisticas_Jugadores
                                WHERE NumeroJugador=?""",
                (Usuario,),
            )
            if cursor_database.fetchone() is not None:
                cursor_database.execute(
                    """UPDATE Estadisticas_Jugadores SET Nombre = ?
                                        WHERE NumeroJugador = ?""",
                    (nombre[Usuario], Usuario),
                )
        conn_database.commit()
        archivo = request.files["miArchivo"]
        if archivo.filename != "":
            CargarPreguntasRespuestas(archivo)

        return "/Juego.html"  # Pagina a donde redirigir

    return render_template("ConexionUsuarios.html", VectConUsu=VectConUsu)
```

Fig. 54. Código #4.

Esta función maneja la lógica de conexión de los usuarios, actualizar sus nombres en la base de datos y carga un archivo si

se proporciona uno. Luego, redirige al usuario a la página del juego estos datos son tomados del frontend este también está relacionado con los hilos de comunicación serial.

Fig. 55. Página de Registro.

Este se coloca activo al detectar que se conectó un usuario también cuenta para ingresar un .txt con un formato establecido.

F. página 3 Juego.html

```python
@app.route("/Juego.html", methods=["GET", "POST"])
def PaginaJuego():
    return render_template("Juego.html", EstJugadores=EstJugadores)
```

Fig. 56. Código #5.

En esta figura se puede observar cómo se carga la página y tiene sus correspondientes métodos GET y POST.

```javascript
$(document).ready(function () {
            EsposarBotones();
            $('#correct-btn').prop('disabled', true);
            $.ajax({
                url: '/ControlPregunta',
                method: 'POST',
                data: { id: 'SIGUIENTE' },
```

Fig. 57. Código #6.

En la página está el código de JavaScript donde apenas inicie la página va a hacer una solicitud post para que se detecte que se va a iniciar la nueva pregunta.

```javascript
var array = data.slice(1, -1).split(',');
var numbers = array.map(function (item) {
    item = item.replace(/['"]+/g, '');
    return isNaN(Number(item)) ? item : Number(item);
```

Fig. 58. Código #7.

Con esta parte del código se tomaron los datos que llegan de la base de datos y empieza a organizar las preguntas y los datos.

```javascript
document.getElementById('question').textContent = numbers[3];
document.getElementById('answer1').textContent = numbers[4];
document.getElementById('answer2').textContent = numbers[5];
document.getElementById('answer3').textContent = numbers[6];
document.getElementById('answer4').textContent = numbers[7];
```

Fig. 59. Código #8.

Numbers es una matriz que llega del backend donde se mira cuáles son las respuestas las preguntas la cantidad de puntos etc esto lo va a tomar del backend de la función /ControlPregunta del Backend.

```python
@app.route("/ControlPregunta", methods=["GET", "POST"])
def ControlPregunta():
    global PreguntaActual
    if request.method == "POST":
        id = request.form.get("id")
        if id == "SIGUIENTE":
            PreguntaActual += 1
            Consulta = ConsultarPreguntasRespuestas()
```

Fig. 60. Código #9.

Este al detectar una solicitud post va a pasar a pregunta actual y va a hacer y va hacer la consulta de las preguntas en la función ConsultarPreguntasRespuestas().

```python
def ConsultarPreguntasRespuestas():
    global PreguntaActual
    cursor_database.execute(
        "SELECT * FROM Preguntas_Respuestas WHERE NumeroPregunta=?", (PreguntaActual,)
    )
    var2 = cursor_database.fetchone()
    if var2 is not None:
        print(f"ConsultarPreguntasRespuestas: {str(var2)}")
        return str(var2)
    else:  # Cuando no hay otra pregunta disponible
        print(f"No hay mas preguntas: {str(var2)}")
        return None
```

Fig. 61. Código #10.

Al pasar a esta función lo que hace esta función es hacer la consulta en la base de datos y enviar preguntas y respuestas.

G. Página 4 Podio.html

```python
@app.route("/Podio.html", methods=["GET", "POST"])
def Podio():
    data = ConsultarPodio()
    return render_template("Podio.html", data=data)
```

Fig. 62. Código #11.

En la primera imagen se observa cómo se carga la página ya con sus estilos cargados y se le va a enviar la información de los que quedaron con un puntaje más alto o bajo esto se ve en la función consultar podio.

```python
def ConsultarPodio():
    matriz = [None] * 5
    for i in range(5):
        cursor_database.execute(
            """SELECT NumeroJugador, Nombre, Puntaje FROM
                        Estadisticas_Jugadores WHERE NumeroJugador=?""",
            (i,),
        )
        fetch_result = cursor_database.fetchone()
        if fetch_result is not None:
            matriz[i] = fetch_result
    return sorted(
        matriz, key=lambda x: x[2] if x is not None else float("-inf"), reverse=True
    )
```

Fig. 63. Código #12.

En esta función hace la solicitud a la base de datos se organizan y se envían en orden.

```javascript
document.addEventListener('DOMContentLoaded', (event) => {
    fetch('/Podio')
        .then(response => response.json())
        .then(data => {
            console.log(data);
            if (data.data[0] != null) {
                document.getElementById('primero').textContent = data.data[0][1];
                document.getElementById('puntaje1').textContent = data.data[0][2];
```

Fig. 64. Código #13.

Ya en el codigo del frontend se puede ver que se le hace la solicitud al backend a través de /Podio donde se le va a signar a toda la información que llega la variable data ya visto esto ya se puede empezar a organizar en cada ID de cada casilla de los ganadores.

IX. CONCLUSIONES

- En resumen, se llevó a cabo un proyecto de preguntas y respuestas con el objetivo de implementar diversas formas de comunicación digital. Para lograr esto, se seleccionaron los componentes y recursos necesarios para el procesamiento de datos, la comunicación entre dispositivos y la visualización de información. Un aspecto crucial fue la identificación de las características del transceptor NRF24L01 y la realización de pruebas para entender su comportamiento en diferentes escenarios.
- Para albergar todos estos componentes electrónicos, se diseñaron e imprimieron varios prototipos en 3D, lo que permitió identificar la distribución óptima de los componentes y mejorar la apariencia visual del dispositivo.
- Se establecieron reglas de comunicación claras y precisas para garantizar un orden en el envío y recepción de datos entre los dispositivos. Esto resultó ser un elemento esencial para la comunicación efectiva entre los dispositivos.
- Es fundamental que los jugadores puedan visualizar las preguntas y respuestas del juego en tiempo real para poder responder de acuerdo con su criterio. Para lograr esto, se creó un servidor FLASK que actúa como el backend de una página web. Desde el frontend, los usuarios pueden visualizar a los jugadores conectados y sus nombres, así como cargar cualquier tópico de preguntas y respuestas a través de un archivo. Este archivo se almacena en una base de datos junto con las estadísticas de los jugadores.
- La implementación exitosa de todas estas tecnologías permitió la realización del juego tal como se había concebido inicialmente. Gracias a este proyecto, se obtuvo una comprensión más profunda de las comunicaciones digitales, la integración de hardware y software, y la importancia del trabajo en equipo en este tipo de proyectos.

X. BIBLIOGRAFÍA

[1] "Facilitando el desarrollo de aplicaciones web: Un estudio sobre la sencillez y adaptabilidad de los frameworks"*, vol. 1, no. 1, pp. 1-10, 2024. tomado de FLASK · PYPI

[2] "JSON: Un formato esencial para el intercambio de datos en el desarrollo web", en *Revista de Desarrollo Web*, vol. 2, no. 1, pp. 11-20, 2024.tomado de GUÍA COMPLETA SOBRE JSON: FORMATO LIGERO PARA INTERCAMBIO DE DATOS - MAURICIO DEVELOPER

[3] "Técnicas modernas en el desarrollo web: , tomado de JQUERY

[4] J. Gehtland, B. Galbraith, D. Almaer, “Pragmatic Ajax: A Web 2.0 Primer,” Pragmatic Bookshelf, 2006. tomado de CAPÍTULO 13. BIBLIOGRAFÍA (INTRODUCCIÓN A AJAX) (UNIWEBSIDAD.COM)