Universidad Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

Curso Académico 2023/2024

CONSOLA BLUETOOTH ARDUINO

SISTEMAS EMPOTRADOS Y DE TIEMPO REAL

GRUPO 13

Javier Rubia Pérez, Aitor Martín Gómez y Pablo Martín Cobo

Tabla de contenido

INTRODUCCIÓN

Nuestra idea parte de la posibilidad de disfrutar de juegos multijugador de manera sincronizada en una consola portátil de bajo costo. Para lograr estas funcionalidades, planeamos utilizar módulos Bluetooth y diseñar la consola de modo que sea recargable por pilas.

Uno de los puntos fuertes de nuestro proyecto es la capacidad para escalar y mejorar la consola, puesto que se pueden desarrollar gran cantidad de juegos, teniendo como única limitación la memoria del Arduino.

Nuestra propuesta incluye la implementación de una interfaz desarrollada en Python que facilita la carga de juegos en cada una de las consolas. Este enfoque no solo garantiza una experiencia de usuario más fluida, sino que también abre la puerta a futuras actualizaciones y expansiones del catálogo de juegos disponibles.

COMPONENTES UTILIZADOS

Para el montaje de cada una de las consolas hemos necesitado de:

- Arduino Nano: el cual tiene el puerto en una ranura en la parte baja de la consola para poder cargar los juegos o incluso poder jugar sin la necesidad de batería externa.
- Zumbador: encargado de reproducir música del juego.
- Pantalla OLED: parte grafica donde se muestra el juego.
- 4 botones: usados para jugar.
- Joystick: uso de movimiento en los juegos.
- Modulo Bluetooth HC-05: para la conexión con otra consola.
- Interruptor: encendido y apagado de la consola si usamos la batería externa.
- Adaptador alimentación pila 9V: localizado en un compartimento lateral de la consola, el cual ha sido fabricado reciclando una caja de caramelos, utilizado para meter la batería de 9V.
- Pila 9V: encargada de la alimentación.

COSTES DEL PROYECTO

A continuación, se muestra la tabla de los componentes necesarios con sus respectivos precios, se debe tener en cuenta que varios componentes no se pueden comprar por unidades, por tanto, es un precio orientativo. Además, pueden surgir otros costes, por ejemplo, herramientas para hacer los agujeros de la caja, espray para pintarla, etc.

Componente	Precio
Arduino Nano (3)	21,60€
Pantalla OLED SH1106 (2)	17,85€
Botones (160)	7,89€
Módulo BT HC-05 (2)	13,90€
Zumbador (2)	-
Cables	-
Joystick (2)	4,44€
Pilas 9V (4)	10€
Madera	4€
Total:	75,33€

ANÁLISIS DE MERCADO

Hemos realizado un pequeño análisis de mercado para contemplar la posibilidad de una futura comercialización de la consola, puesto que, viendo el precio de los materiales creemos que tendría cabida en el mercado.

El primer paso ha sido realizar una búsqueda de videoconsolas similares a la nuestra, pero no hemos encontrado ninguna de características similares (con multijugador de juegos retro).

Nuestra consola tendría un coste de producción de alrededor de 16,21€, mientras que el coste de consolas similares va desde unos 11€ aproximadamente hasta unos 36€.

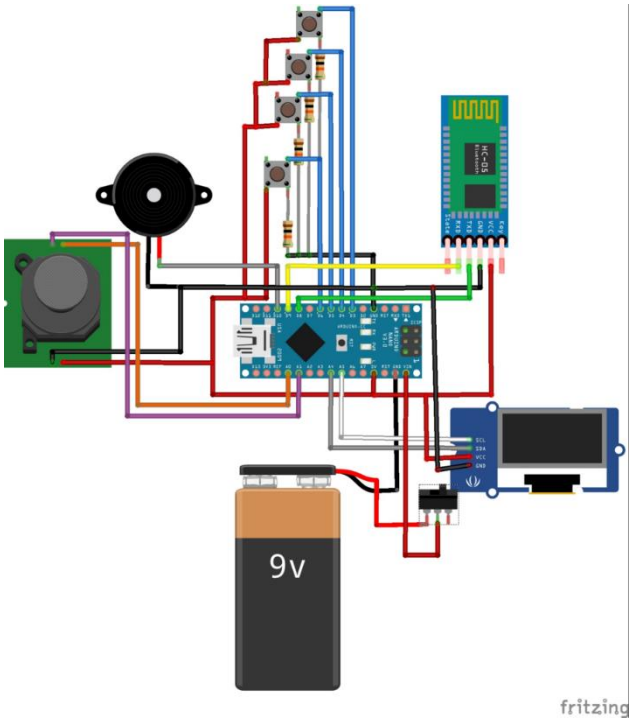
El problema de estas videoconsolas que hemos mencionado, es la falta de multijugador Bluetooth, lo cual nos hace pensar que en un futuro, con una mayor disponibilidad de recursos y tiempo, y a su vez, mejoras tanto en el hardware y software, podría llegar a ser comercializable y tener su hueco en el mercado.

HARDWARE Y MONTAJE

Para la hora del montaje de este proyecto hemos seguido los siguientes pasos:

- En primer lugar, tras recibir los materiales que habíamos pedido, comenzamos probando todos los componentes por separado, viendo así la configuración necesaria para aplicarlo en el proyecto.
- Continuamos con el montaje de todos los componentes en una protoboard, para de esta forma poder centrarnos en el código y completar todas las funcionalidades.
- Una vez terminamos todo el proceso de código pasamos al montaje final. Como principal material para la carcasa teníamos en mente realizar el diseño para después imprimirlo en una impresora 3D. No vimos viable esa opción y elegimos un material fácil de manejar como es la madera.

- Para trabajar la madera no teníamos los mejores instrumentos, pero conseguimos un final diseño limpio y sin imperfecciones tras el lijado y pintado de la madera.



SOFTWARE

Antes de comenzar a programar o realizar cualquier avance en el ámbito del software, es necesaria la configuración de los módulos Bluetooth, los pasos para llevarlo a cabo se describen a continuación:

1. Envía el comando AT desde el monitor serial del IDE de Arduino. Si recibes un OK, la comunicación es correcta.
2. Utiliza AT+ADDR? para obtener la dirección MAC de cada módulo.
3. Establece un Arduino como esclavo (AT+ROLE=0) y el otro como maestro (AT+ROLE=1).
4. Usa AT+CMODE=0 en ambos dispositivos.
5. Vincula los dispositivos usando AT+BIND=DIR_MAC_OTRO.
6. Tras reiniciar, verifica que la luz LED del módulo parpadee más despacio, indicando una conexión exitosa.

Las librerías necesarias son las siguientes:

- Adafruit SH1106
- Adafruit GFX
- SoftwareSerial

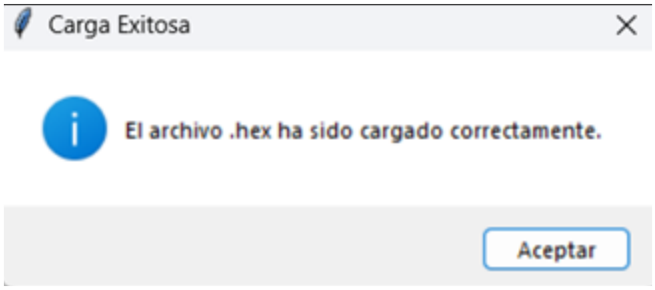
Por otra parte, como se mencionó en la introducción, desarrollamos una aplicación en Python para cargar los juegos de manera sencilla y eficiente, para que el usuario no necesite acceder a ningún entorno de desarrollo. A continuación, se encuentra una foto de la interfaz:

La aplicación tiene una funcionalidad muy simple, primero debemos seleccionar el puerto en el que se encuentra conectado el Arduino para poder cargar el juego, simplemente haciendo click nos aparecen todas las opciones disponibles.

Después haciendo click en “Seleccionar Juego”, se abre el explorador de archivos para escoger el juego que vas a jugar, cabe destacar que el juego debe estar en el formato .hex, que es el formato de Arduino(.ino) pero ya compilado.

Realizados los pasos anteriores, se habilita el botón de “Jugar”, una vez se hace click en él, el juego comienza a cargarse en nuestra videoconsola.

Además, aparece esta ventana que confirma que todos los procesos se han realizado de forma correcta:



Los videojuegos que hemos adaptado tienen las siguientes líneas de código:

```
SoftwareSerial BT(8, 9);
int receivedLevel;
int receivedScore;

void setup() {

  BT.begin(9600);
  pinMode(6, INPUT);
```

Este fragmento de código nos muestra como inicializamos el módulo Bluetooth.

```
while (!BT.available()) {
  BT.write("Ready");
}
```

Aquí vemos uno de los mecanismo de sincronización que hemos aplicado a lo largo de la aplicación, consiste en un bucle que no termina hasta que el módulo Bluetooth tiene información disponible, esto indicaría que el otro módulo Bluetooth ha enviado alguna trama y por tanto, también está esperando a ser sincronizado.

```
String winner = "";
String receivedMsg = "";
receivedLevel = -1;
receivedScore = -1;
while (receivedScore == -1) {
  while (!receivedMsg.length() || receivedMsg.charAt(0) != '$') {
    if (BT.available()) {
      receivedMsg = BT.readStringUntil('\n');
    }
    String message = "$" + String(level) + "," + String(points);
    BT.println(message);
    delay(500);
  }

  // Procesar el mensaje recibido
  int separatorIndex = receivedMsg.indexOf(',');
  if (separatorIndex != -1) {
    receivedLevel = receivedMsg.substring(1, separatorIndex).toInt();
    receivedScore = receivedMsg.substring(separatorIndex + 1).toInt();
  }
}
```

Podemos ver como hemos manejado la recepción y el envío de las tramas que contienen la puntuación y el nivel en el que se encuentra una consola y otra.

Consiste en un bucle principal de tipo “while” que se ejecuta infinitamente hasta que se recibe una trama que contenga una puntuación válida en este caso.

```
while (BT.available() > 0) {
  BT.read();  // Leer y descartar los datos disponibles en el buffer
}
```

Ya por último, observamos que existía la posibilidad de que si una consola terminase antes que otra, el buffer de la otra consola se saturase por recibir tantas tramas. Para evitar este problema añadimos las líneas de código que se observan más arriba para vaciar todos los mensajes recibidos para cuando empiece la siguiente partida.

PROBLEMAS Y SOLUCIONES

A lo largo del desarrollo del proyecto hemos encontrado diversos problemas, para los cuales hemos tenido que buscar una solución. Estos problemas se encuentran detallados a continuación.

- *Exceso de memoria ocupada:* Uno de los principales problemas que tiene este proyecto sería la memoria dinámica del Arduino Nano. Gran parte del éxito de este proyecto es gracias a la optimización de código, variables y refactorizaciones que se han llevado a cabo a lo largo de nuestro software, observamos que si la memoria reservada a las variables dinámicas alcanza un número *>90%* la ejecución del videojuego fallaba por culpa de un desbordamiento de memoria
    - *POSIBLE SOLUCIÓN* -> Ampliación de memoria mediante el uso de una tarjeta SD. Si esto no fuera suficiente, sería considerable cambiar la placa de desarrollo a una más potente (p.e. Raspberry PI)
- *Pantalla significativamente pequeña*: La pantalla apenas ocupa una pulgada, esto es bueno respecto al consumo de nuestra batería de 9V, sin embargo, para el usuario puede resultar incomodo leer u observar ciertas partes de esta.
    - *POSIBLE SOLUCIÓN* -> Sustitución de pantalla a una más grande, sin embargo, esto también llevaría a la sustitución de la placa de desarrollo por otra más potente tal y como hemos comentado más arriba.
- *Batería de poca duración*: El diseño de nuestra consola contempla la facilidad de sustitución de dicha batería, sin embargo, no es una batería que pueda durar un tiempo prolongado.
    - *POSIBLE SOLUCIÓN* -> Reemplazo de batería alcalina por una de litio o similares, esto significaría añadir un módulo de carga

adicional para recargar esta batería y evitar que haya que sustituirla cada cierto tiempo.

- *Poca retroalimentación de errores*: Esto ciertamente supone una preocupación mayor para nosotros como desarrolladores. Somos conscientes de que si, por ejemplo, una consola perdiese la conexión con la otra, el usuario no sería capaz de enterarse más que en el momento de que su partida no consigue cargar.
    - *POSIBLE SOLUCIÓN* -> Modificación del código y retroalimentación añadida, sin embargo, esto no es tan fácil como parece, pues si lo implementásemos la memoria de las variables subiría de manera considerable y es posible que nuestro videojuego dejase de funcionar directamente, por tanto, habría que valorar si es algo necesario por completo.

CONCLUSIONES

A medida que avanzamos en este proyecto, hemos notado la notable capacidad de mejora y escalabilidad que posee nuestra videoconsola. Tal y como se mencionó en el apartado anterior, con simples ajustes en el hardware, especialmente en la placa, u otros componentes como la pantalla, podríamos expandir considerablemente la capacidad de almacenamiento de juegos y evitar la saturación de la memoria del Arduino.

A pesar de los obstáculos que se nos han presentado, nos complace confirmar que hemos alcanzado exitosamente nuestro objetivo. Hemos logrado construir una videoconsola con funcionalidad multijugador, todo ello dentro de un presupuesto económico.

Esto demuestra que no es necesaria una gran inversión para realizar este tipo de proyectos, con un software adecuado y una dosis de ingenio y creatividad, se abre un amplio abanico de posibilidades, demostrando que la innovación puede superar las barreras financieras.