



GENERATOR: INFORME FINAL

TALLER DE PROYECTO I

FACULTAD DE INGENIERÍA - UNIVERSIDAD NACIONAL DE LA PLATA

13 DE FEBRERO DEL 2025

AUTORES:

GARCIA IACOVELLI, TOBIAS

MADERA, RAMIRO

PEREIRA, ULISES

ÍNDICE

INTRODUCCIÓN.....	3
OBJETIVOS DEL PROYECTO.....	4
1. OBJETIVOS PRIMARIOS.....	4
2. OBJETIVOS SECUNDARIOS.....	4
LICITACIÓN DE REQUERIMIENTOS.....	5
1. REQUERIMIENTOS DE HARDWARE.....	5
2. REQUERIMIENTOS DE SOFTWARE.....	5
3. REQUERIMIENTOS NO FUNCIONALES.....	6
4. REQUERIMIENTOS SECUNDARIOS.....	7
DISEÑO GENERAL DEL PROYECTO.....	8
DESCRIPCIÓN DE HARDWARE.....	9
1. DISPLAY TFT + ILI9341 + LÁMINA CAPACITIVA + XPT2046.....	9
2. MPU6050.....	10
3. SALIDA DE AUDIO CON LM386.....	10
4. ALIMENTACIÓN.....	12
DISEÑO ESQUEMÁTICO DE CONEXIONES.....	13
1. EDU-CIAA.....	13
2. DISPLAY + TOUCH.....	13
3. LM386 + PARLANTE.....	14
4. MPU6050.....	15
5. ESQUEMÁTICO FINAL.....	15
DISEÑO DEL PCB.....	17
1. ASIGNACIÓN DE FOOTPRINTS A LOS COMPONENTES.....	17
I. DISPLAY + TOUCH.....	17
II. LM386.....	17
III. MPU6050.....	18
IV. OTROS FOOTPRINTS.....	18
2. PROPIEDADES DE LAS CONEXIONES.....	18
3. PCB FINAL.....	19
4. FABRICACIÓN.....	20
DISEÑO DE SOFTWARE.....	22
1. ARQUITECTURA DEL FIRMWARE.....	22
2. MÁQUINA DE ESTADOS DEL SISTEMA.....	22



3. GENERACIÓN DE VALORES ALEATORIOS.....	25
REPOSITORIO DEL PROYECTO.....	25
VALIDACIÓN DE REQUERIMIENTOS.....	26
CONCLUSIONES.....	27
REFERENCIAS BIBLIOGRÁFICAS.....	28
ANEXO.....	29
A. DIVISIÓN DE TAREAS.....	29
B. CRONOGRAMA FINAL.....	31
C. CONSUMOS ELÉCTRICOS DE LOS COMPONENTES.....	32
D. SELECCIÓN DE DATOS Y BOTÓN DE INICIO.....	32
E. CÁLCULO DE JUEGOS Y PUNTAJES.....	33

INTRODUCCIÓN

El concepto del dado acompaña a la humanidad desde hace siglos, siendo un elemento recurrente en las distintas sociedades a lo largo del tiempo. Incluso desde antes que se tenga registro de la historia, elementos similares a los dados que se conciben hoy en día se encontraban presentes en los juegos y las costumbres populares.

Esto se debe gracias a su característica distintiva de la aleatoriedad: el dado permite simular la obtención de valores impredecibles, imposibilitando así la determinación previa de una secuencia de acciones de las que dependa del resultado que el mismo vaya a devolver. Gracias a esto, los dados se encuentran presentes en diversos elementos culturales, entre ellos, el ámbito lúdico. Muchos juegos utilizan al dado dentro de su reglamento para desarrollar sus dinámicas, y muchos otros construyen su funcionamiento únicamente a partir de los valores que devuelven uno o más dados al ser lanzados. Este es el caso de los distintos juegos de Cubilete, entre los que se encuentra el popular juego de la Generala.

Sin embargo, las últimas innovaciones digitales han desplazado parcialmente a los dados como forma de entretenimiento entre la gente, particularmente en las generaciones más jóvenes que no han crecido con estos juegos. De este modo, las redes sociales y los juegos en línea o de teléfono celular le han quitado el protagonismo a otras actividades que aún no han incursionado en las nuevas plataformas digitales.

Es gracias a estos hechos que se ha propuesto una digitalización de los dados que innove sobre su propósito y los presente en un formato llamativo y moderno. Así, se diseñó un primer prototipo de cubilete electrónico, el GENERATOR, que emule las tiradas que se realizan en la generala. Esta idea se inscribe en el marco de diseñar una consola que contenga múltiples juegos de dados; partiendo de una base sencilla, y relativamente conocida por el público general, este proyecto podrá expandirse e incorporar distintos juegos al cubilete, así como acceso a internet, dándole versatilidad al dispositivo y volviéndolo interesante para todos aquellos niños y jóvenes que les interesa el mundo de los videojuegos.

Este primer prototipo contará con una pantalla táctil, un sensor de movimiento y un parlante. Deberá permitir “agarrar” los dados a tirar, detectar el movimiento que recrearía la sacudida del cubilete, y emitir sonidos que emulen la tirada de los dados. El mismo permitirá realizar tiradas de generala, calcular los posibles juegos, y seleccionar dados a guardar de cara a próximas tiradas para poder, a lo largo de los turnos, llegar a armar el icónico juego de 5 números iguales, la “Generala”.

OBJETIVOS DEL PROYECTO

Para poder organizar de manera óptima el trabajo a realizar, se han delimitado los objetivos específicos a cumplir, jerarquizándolos de mayor a menor importancia en su cumplimiento.

1. OBJETIVOS PRIMARIOS

Los objetivos primarios han sido los fundamentales para considerar al proyecto como logrado. Son en los cuales recae la esencia del primer prototipo, y los cuales se ha buscado cumplir a toda costa. Dichos objetivos son los siguientes:

- Simular una tirada de juego de Generala de manera funcional;
- Desarrollar la interfaz gráfica en la pantalla LCD, mostrando los dados y puntaje obtenidos;
- Integrar un sensor de movimiento que detecte la sacudida del dispositivo que simula la mezcla y lanzamiento de los dados;
- Configurar el táctil de la pantalla para permitir la selección de dados a mantener en el siguiente lanzamiento;
- Asegurar que el dispositivo funcione de manera autónoma utilizando una batería;
- Implementar efectos de sonido que simulen los distintos momentos de una tirada de dados.

2. OBJETIVOS SECUNDARIOS

Los objetivos secundarios han sido aquellos que, si bien han sido planteados, no eran fundamentales para lograr el propósito de este primer prototipo. Han sido objetivos no necesarios pero sí deseables, que hubieran mejorado el resultado final. Hubo 2 objetivos secundarios: construir una cubierta protectora para el dispositivo que funcione de carcasa del mismo, e incorporar un sistema de puntuación histórica que registre el progreso del jugador a lo largo de múltiples partidas. Ninguno de estos objetivos mencionados ha sido alcanzado; por cuestiones de cronograma, se decidió enfocar el trabajo en que los objetivos primarios sean alcanzados completamente, y se han dejado a estos objetivos secundarios como próximos aspectos a trabajar en una posible expansión del proyecto.

LICITACIÓN DE REQUERIMIENTOS

Definidos los objetivos, se han enumerado los requerimientos que deben cumplirse para alcanzar los mismos. Dichos requerimientos se han dividido entre los requerimientos de Hardware y Software, y dentro de estas categorías, se han descompuesto en funcionales y no funcionales.

A continuación, se mencionarán los requerimientos funcionales tanto de Hardware como de Software, los cuales serán retomados en la validación de los requerimientos más adelante en el informe. Además, se listarán los requerimientos no funcionales que se han contemplado en el desarrollo, y los requerimientos listados para los objetivos secundarios que no han sido implementados.

1. REQUERIMIENTOS DE HARDWARE

1. Debe contar con una pantalla que sirva como interfaz visual;
 - 1.1. La pantalla debería ser lo suficientemente grande poder mostrar el puntaje de los dados y el juego que armen los mismos;
 - 1.2. Esta pantalla será táctil, para poder interactuar con el dispositivo;
2. El dispositivo debe contar con una batería que le brinde una autonomía que permita jugar al menos una partida;
 - 2.1. Esta batería debe poder recargarse, para luego poder volver a sacudir el dispositivo sin inconvenientes;
3. Debe poder tener un sensor de movimiento;
 - 3.1. Este sensor debe poder detectar el movimiento oscilatorio que se lleva a cabo a la hora de agitar un cubilete;
4. Debe contar con un botón que permita encender y apagar el dispositivo;
5. Debe contar con un parlante para poder emitir sonidos en relación a las acciones que se estén llevando a cabo.

2. REQUERIMIENTOS DE SOFTWARE

1. Requerimientos relativos a los datos:
 - 1.1. Debe poder generar valores aleatorios entre los números 1 y 6 para los mismos;
 - 1.2. Se deben poder seleccionar uno o más dados para conservar su valor a lo largo de las próximas tiradas;
 - 1.3. Se debe poder deshacer el estado previo, permitiendo volver a tirar dichos dados;

- 
2. Requerimientos relativos a los puntajes:
 - 2.1. Debe ser capaz de identificar el tipo de combinación especial dentro de los juegos posibles a armar en la generala, en caso de presentar alguna de estas los 5 dados en pantalla;
 - 2.2. Debe poder calcular el puntaje que tiene el conjunto de 5 dados en el momento;
 3. Requerimientos relativos a la interfaz gráfica:
 - 3.1. Debe poder mostrar los valores actuales de los dados tirados;
 - 3.2. Debe poder mostrar el juego que se arma y el puntaje;
 4. Requerimientos relacionados a la interacción con el sistema:
 - 4.1. Al tocar los dados en la pantalla, permitirá guardar su valor para que no sea vuelto a calcular en la próxima tirada;
 5. Requerimientos relacionados con el movimiento:
 - 5.1. El sistema debe ser capaz de comunicarse con el sensor de movimiento del dispositivo;
 - 5.2. Al detectar el movimiento por parte del sensor, debe aleatorizar el valor de los dados no guardados;
 6. Requerimientos relacionados con el sonido:
 - 6.1. Debe emitir efectos de sonido, según las diferentes etapas de juego, por medio del parlante:
 - 6.1.1. Debe emitir un sonido que aluda a dados mezclándose cuando se está sacudiendo el dispositivo;
 - 6.1.2. Debe emitir un sonido que aluda a la tirada de dados una vez que se deja de sacudir el dispositivo;
 - 6.1.3. Debe emitir un sonido de campanilla cuando la tirada sea una Generala.

3. REQUERIMIENTOS NO FUNCIONALES

Estos requerimientos son requerimientos que no afectan en la funcionalidad del proyecto, por lo cual no se ve necesidad de hacer particular énfasis en la validación, pero que sí han sido tenidos en cuenta en el desarrollo.

HARDWARE

1. El proyecto debe desarrollarse utilizando el Microcontrolador EDU-CIAA-NXP;
2. Todos los elementos físicos deben estar ensamblados en un único dispositivo final;
3. La pantalla debe poder manejar colores y mostrar una interfaz atractiva;
4. Los componentes deben ser poco pesados para poder agitar fácilmente el dispositivo;

SOFTWARE

- 
1. El código debe estar desarrollado en C;
 2. El código debe ser desarrollado con los entornos de trabajo “CIAA Launcher” e “IDE Eclipse”.
 3. La interfaz diseñada para la pantalla debe ser estéticamente agradable;
 - 3.1. La información de los 5 dados y el puntaje debe mostrarse de forma clara en ella;
 - 3.2. Deben estar incluidos los nombres o firmas de los desarrolladores del proyecto.

4. REQUERIMIENTOS SECUNDARIOS

Estos son los requerimientos que no han sido implementados, dado que aspiraban a cumplir objetivos secundarios que fueron dejados de lado.

En cuanto al objetivo de tener una carcasa protectora de los componentes:

- Este dispositivo debe contar con una carcasa (requerimiento secundario);
- El diseño de dicha carcasa debe ser agradable para el usuario, y ergonómico para facilitar su uso (requerimiento secundario);

Y en cuanto al objetivo de tener un historial de puntajes:

- La pantalla debe poder mostrar el registro histórico de puntajes más altos;
- Estos puntajes deben ser accesibles al tocar ciertos botones o espacios en la pantalla;
- El firmware debe poder almacenar los puntajes más altos obtenidos por el usuario históricamente, siendo este registro persistente en el tiempo; también debe ser capaz de eliminar este registro histórico de puntajes;

DISEÑO GENERAL DEL PROYECTO

A raíz de los requerimientos pensados, se planteó un esquema de diseño general en relación al Microcontrolador el cual se basa en la relación con 5 periféricos: una pantalla que funciona de interfaz gráfica, un panel táctil para dicha pantalla, un sensor de movimiento, una salida de audio, y una alimentación que abastezca todo el sistema. El firmware del proyecto interactuará con el táctil y el sensor de movimiento, los cuales proveerán la información de entrada necesaria, y emitirán las salidas gráficas y de audio correspondientes para armar el juego. El diagrama de bloques de este esquema se encuentra presente en la Figura 1.

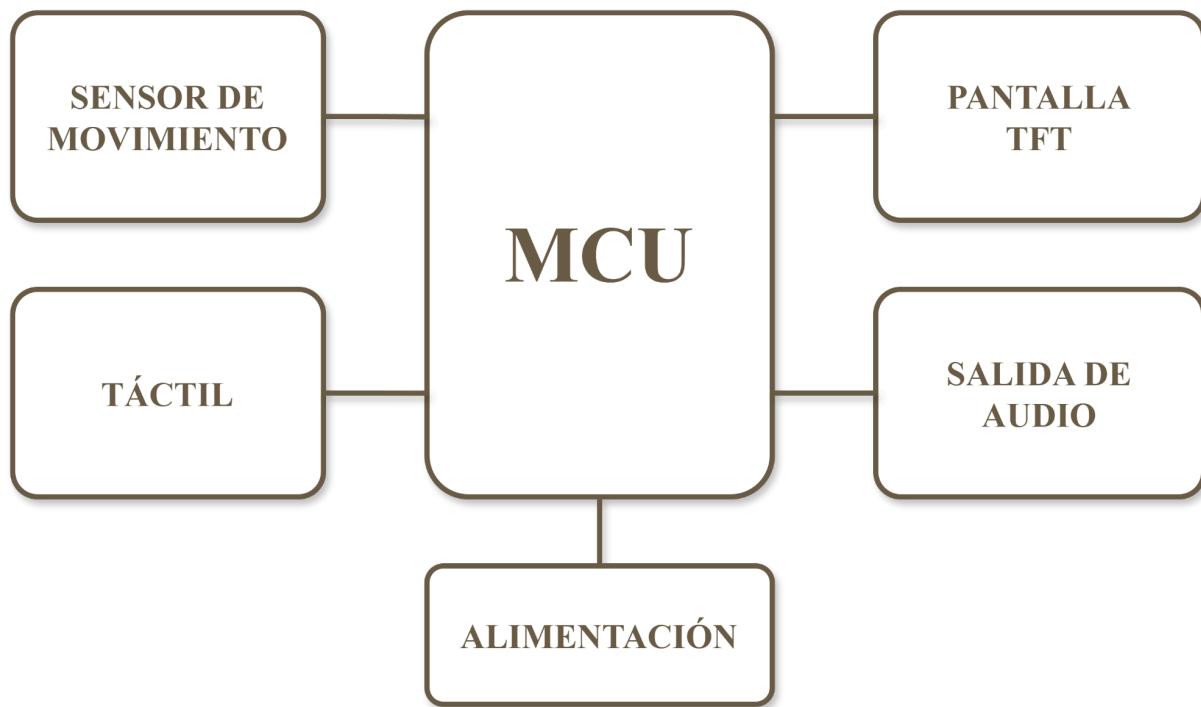


Figura 1: Diagrama de Bloques del Sistema

DESCRIPCIÓN DE HARDWARE

Para la realización del proyecto, se hizo uso de un módulo de pantalla táctil, un acelerómetro, un parlante con un amplificador, y una fuente de alimentación portátil y recargable. Para mitigar el posible ruido proveniente de la alimentación en todos los componentes, se añadieron capacitores de desacoplo de $100nF$ y $47nF$ en la línea de alimentación del amplificador. Para tener mayor información respecto a los consumos de cada módulo, revisar el anexo C.

1. DISPLAY TFT + ILI9341 + LÁMINA CAPACITIVA + XPT2046

Para la interfaz gráfica, se utilizó pantalla TFT de 2,8 pulgadas y 240x320 píxeles, con un controlador anexo (ILI9341) que se comunica con el microcontrolador utilizando el protocolo SPI; además de una lámina capacitiva presente sobre la misma, con su propio controlador (XPT2046), el cual se comunica a través del mismo protocolo con el MCU.

El módulo cuenta con 5 pines para la comunicación con el ILI9341 (4 para la comunicación por SPI, y un “d/c” que indica si la información enviada es un dato o un comando), 5 pines para la comunicación con el XPT2046 (4 para SPI y uno de interrupción), 2 para la alimentación (Vcc y Gnd), 1 para alimentar el backlight de la pantalla y, adicionalmente, 4 pines para controlar una tarjeta SD que se puede conectar; los cuales no fueron utilizados en el proyecto.

Tanto la alimentación del display como la del backlight es de 3,3V.

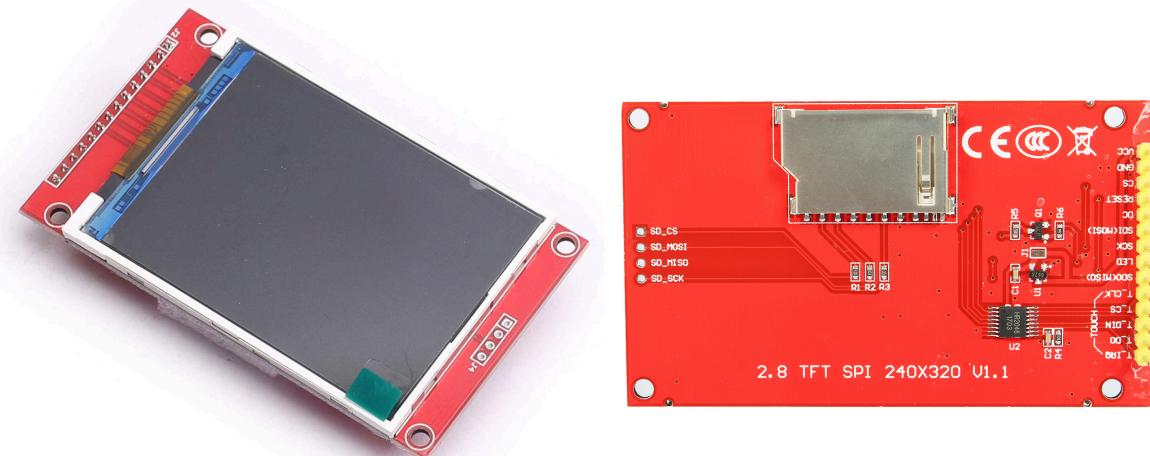


Figura 2: Display TFT utilizado

2. MPU6050

El MPU6050 es un sensor de movimiento de alta precisión que combina un acelerómetro triaxial y un giroscopio triaxial en un solo chip. El acelerómetro integrado permite medir la aceleración en los ejes X, Y y Z, lo que facilita la detección de inclinaciones y movimientos lineales. Por otro lado, el giroscopio mide la velocidad angular en los mismos tres ejes, permitiendo obtener información sobre la orientación y la rotación del dispositivo. Además, utiliza comunicación I2C para facilitar su integración. Su alimentación debe ser de 3,3V.

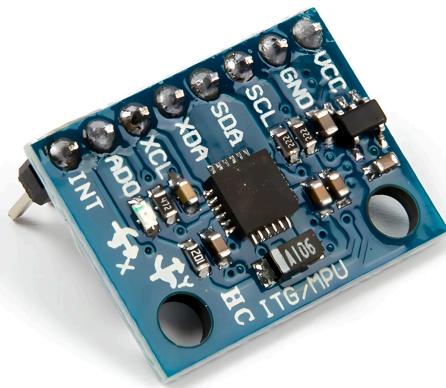


Figura 3: Módulo MPU6050

3. SALIDA DE AUDIO CON LM386

El circuito implementado utiliza un amplificador LM386 para procesar la señal de audio proveniente de un DAC del MCU y amplificarla antes de su salida a un parlante de 8Ω y 0,5W. La Figura 4 muestra el esquema del sistema diseñado, incorporando mejoras destinadas a optimizar la calidad de la señal y reducir interferencias no deseadas.

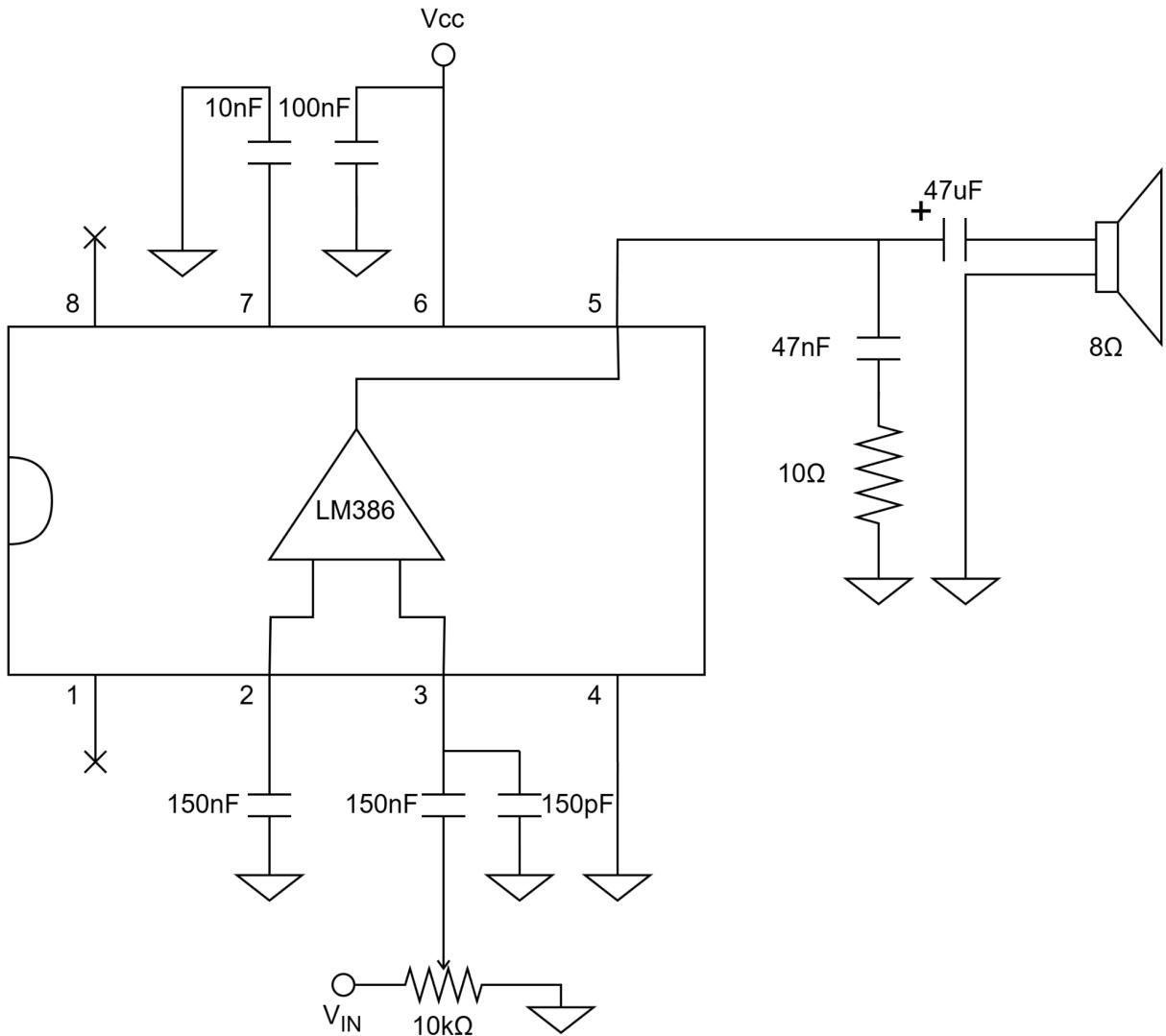


Figura 4: Esquema del circuito de amplificación de audio basado en el LM386.

Para regular el nivel de entrada al amplificador y evitar una amplificación excesiva que pueda dañar el parlante, se incluyó un resistor variable (preset) antes de la señal de entrada. Dado que el LM386 presenta una ganancia mínima de 20, este ajuste permite un control adecuado del nivel de amplificación.

A la salida del amplificador, se incorporó una red de Zobel, compuesta por un capacitor de $47nF$ y una resistencia en serie de 10Ω , con el objetivo de estabilizar la impedancia en altas frecuencias y prevenir posibles oscilaciones que podrían afectar la calidad del sonido o comprometer la integridad del parlante.

Además, se integró un capacitor de $100pF$ antes de la entrada de señal para reducir interferencias de alta frecuencia.

El diseño también contempla un capacitor de acople de $100nF$ en la entrada del amplificador para bloquear cualquier componente de corriente continua que pueda afectar la amplificación. Adicionalmente, se incorporó un capacitor de $100nF$ en la entrada inversora para mantener la simetría en las corrientes de base de los transistores internos del LM386, mejorando su desempeño y reduciendo distorsiones en la señal amplificada.

4. ALIMENTACIÓN

La alimentación utilizada para el proyecto fue el Powerbank Samsung EB-PG950, con una capacidad de $5100mA$. La misma fue conectada al microcontrolador con un switch de por medio, que permitirá habilitar e interrumpir la alimentación para prender o apagar el dispositivo.



Figura 5: Powerbank Samsung EB-PG950

DISEÑO ESQUEMÁTICO DE CONEXIONES

Para el diseño del esquemático del proyecto, se utilizó el software KiCad, un programa de código abierto que permite tanto la creación de esquemáticos como el diseño de la PCB, con varias librerías con esquemas y huellas de fabricantes, como también la posibilidad de crear los propios o modificar los existentes.

1. EDU-CIAA

En el esquemático, se representó al microcontrolador como dos conectores (P2 y P1), de 2x20 pines, ya que son los únicos dos conectores que se utilizarán de interfaz entre los componentes del proyecto y la EDU-CIAA:

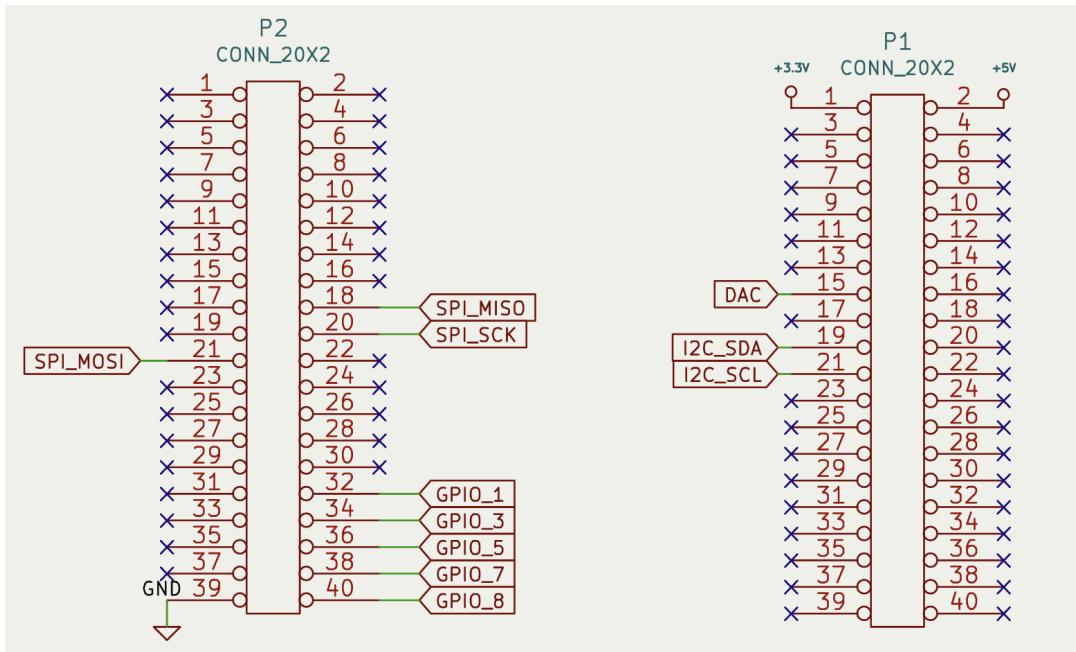


Figura 6: Esquemático de la placa EDU-CIAA

2. DISPLAY + TOUCH

En las librerías integradas al software no existía el componente deseado, por lo que se optó por modificar un esquema de una versión del display con el ILI9341 que carecía del módulo touch, agregando así este último:

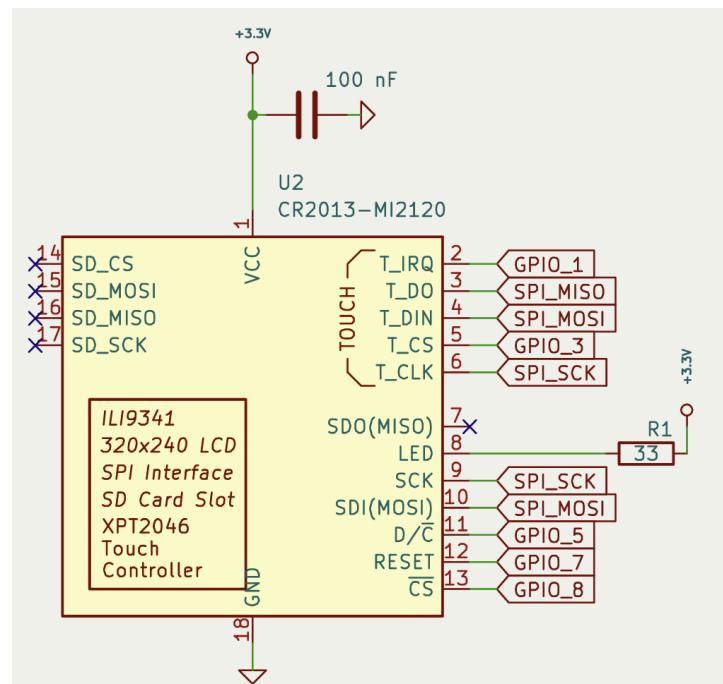


Figura 7: Esquemático de la pantalla y el táctil

3. LM386 + PARLANTE

Tanto el esquema del amplificador como el del parlante y elementos pasivos se encontraron sin problema en las librerías:

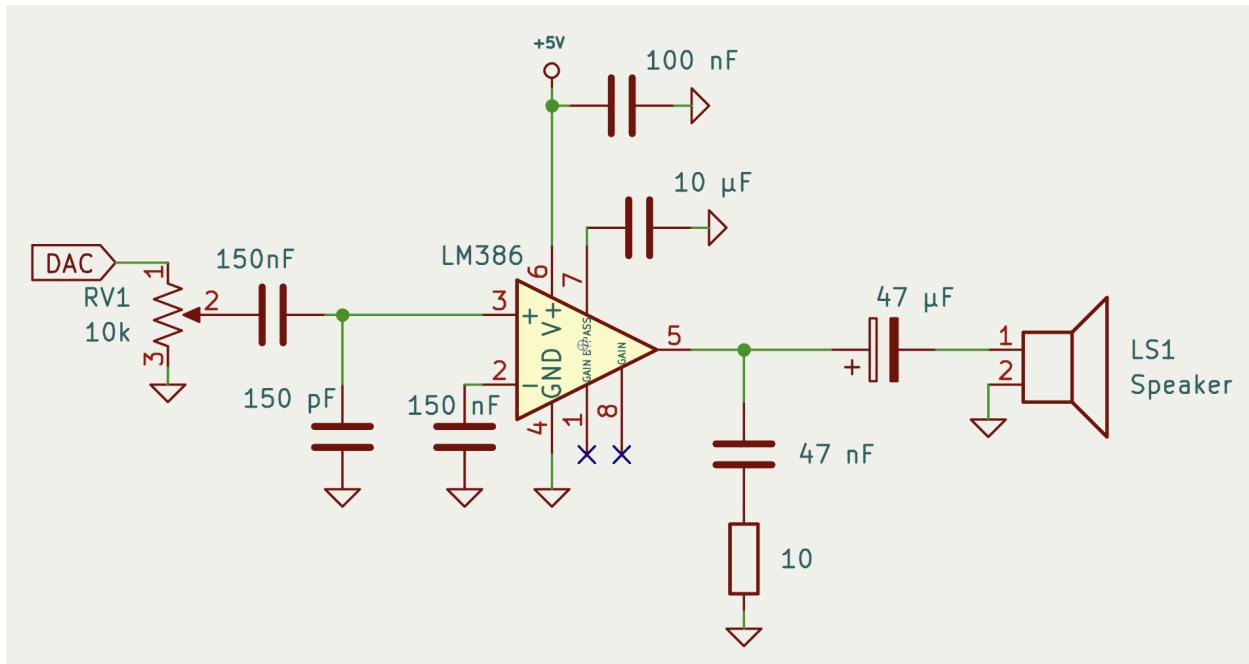


Figura 8: Esquemático de la salida de audio

4. MPU6050

El esquemático también estaba disponible en las librerías de KiCad:

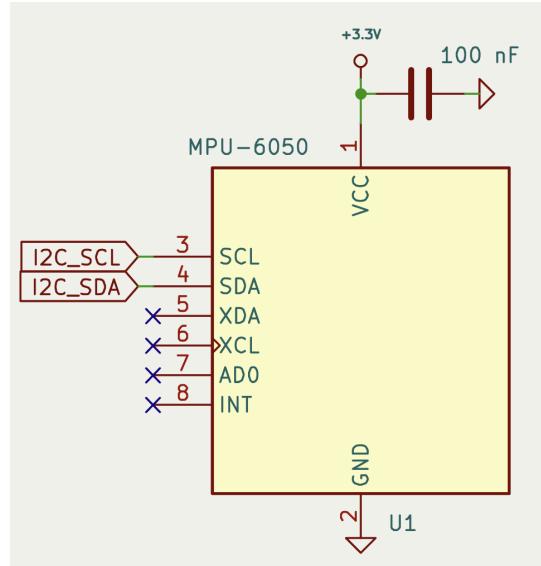


Figura 9: Esquemático del acelerómetro

5. ESQUEMÁTICO FINAL

El diseño completo del esquemático se muestra en la figura 10, y se encuentra adjunto en el repositorio de Github, en conjunto con el código del proyecto.

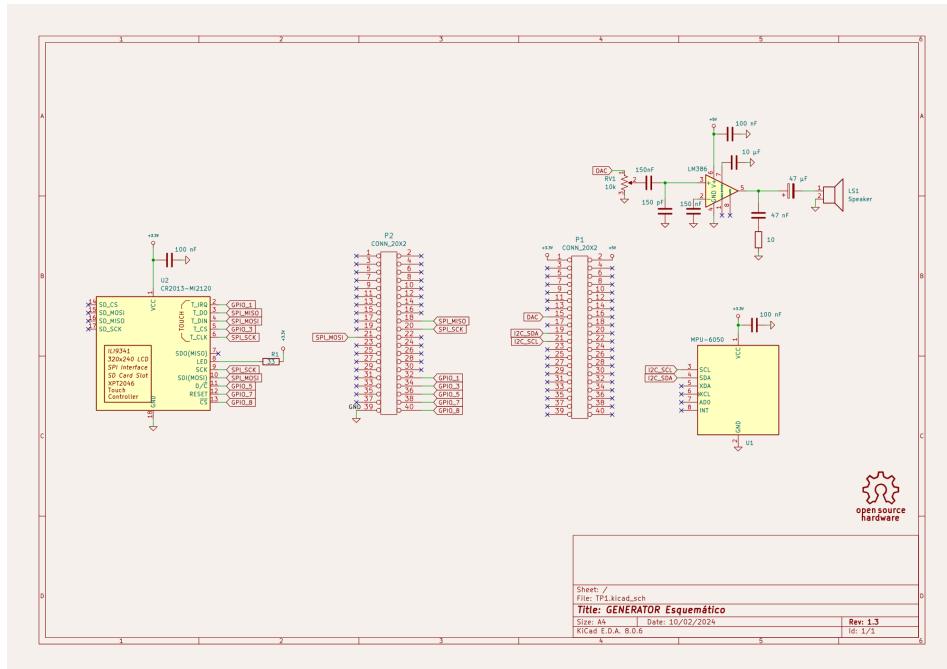


Figura 10: Diseño final del esquemático

DISEÑO DEL PCB

Para lograr la interconexión de los distintos módulos, se tuvo que diseñar una placa de circuito impreso (PCB). Para el desarrollo del mismo se continuó utilizando el antes mencionado programa KiCad.

1. ASIGNACIÓN DE FOOTPRINTS A LOS COMPONENTES

Ya habiendo diseñado el esquemático para el proyecto, el primer paso para diseñar el PCB fue asignarle una huella (footprint) a cada componente:

I. DISPLAY + TOUCH

Como las librerías del software utilizado no contaban con el módulo específico, se tuvo que diseñar agregando las conexiones del touch a una huella existente de un display con el controlador ILI9341.



Figura 11: Huella del display

II. LM386

En este caso la huella se encontraba en el software, por lo cual no presentó mayores complicaciones.

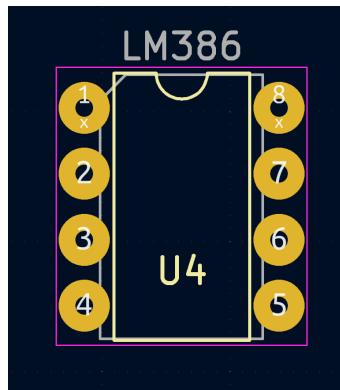


Figura 12: Huella del LM386

III. MPU6050

En este caso, tampoco existía la huella necesaria en las librerías integradas en el software, pero, por la simplicidad del dispositivo, se optó por crearla manualmente. El diseño consta de un encapsulado de $21,4\text{ms} \times 16,4\text{ms}$, con 8 pines con una separación estándar de $2,54\text{ms}$ entre sí.

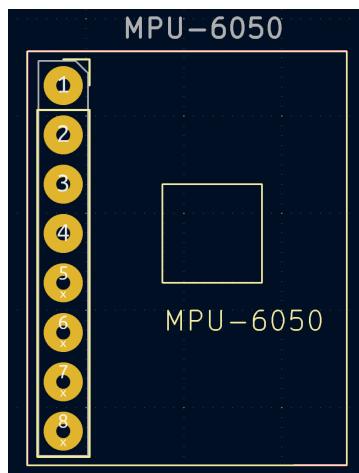


Figura 13: Huella del MPU6050

IV. OTROS FOOTPRINTS

El resto de los componentes son resistores, capacitores, el potenciómetro, y el parlante; todos elementos comunes y presentes en las librerías integradas.

El MCU se fue representado por los 2 puertos de 2×20 pines del esquemático, eliminando los pines que no servían para el conexionado del proyecto o como apoyo estructural para sostener el PCB.

2. PROPIEDADES DE LAS CONEXIONES

Luego se eligieron las propiedades de los pines y las pistas; determinadas por el proceso de fabricación a utilizar:

Pines:

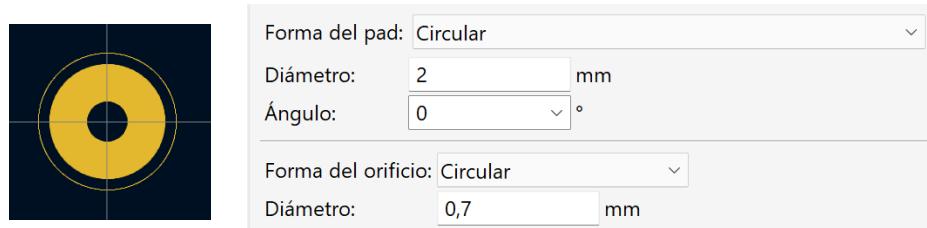


Figura 14: Propiedades de pines

Pistas:

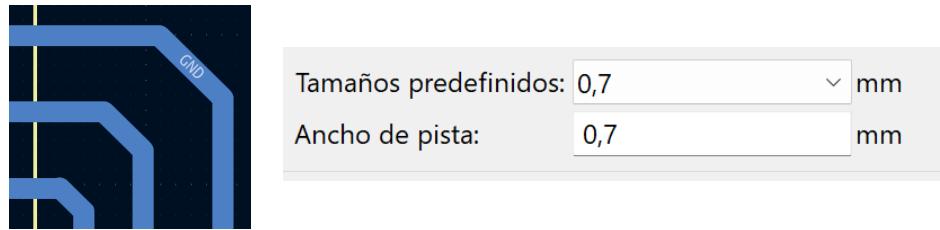


Figura 15: Propiedades de pistas

3. PCB FINAL

Por último, se aprovechó la capacidad que tiene KiCad de crear un PCB a partir del esquemático del proyecto, importando todas las huellas y mostrando una guía de las conexiones entre los componentes.

Como la idea es poseer un dispositivo portable de tamaño lo más reducido posible, se buscó que todos los componentes entren sobre el rectángulo que forma la placa de la EDU-CIAA, lo cual se logró con la ayuda de un esquema de la misma, proveniente de una librería de EDU-CIAA para KiCad.

Luego se posicionaron todos los elementos de manera que se minimice el entrecruzamiento de las vías de cobre y la complejidad de dichas vías; para esto se trabajó en simultáneo con el esquemático, intercambiando las conexiones que se podían modificar para simplificar el resultado.

El conexionado final se adjunta en la figura 16. El PDF y los archivos KiCad del mismo, se encuentran en la rama “main” del repositorio GitHub adjunto más adelante:

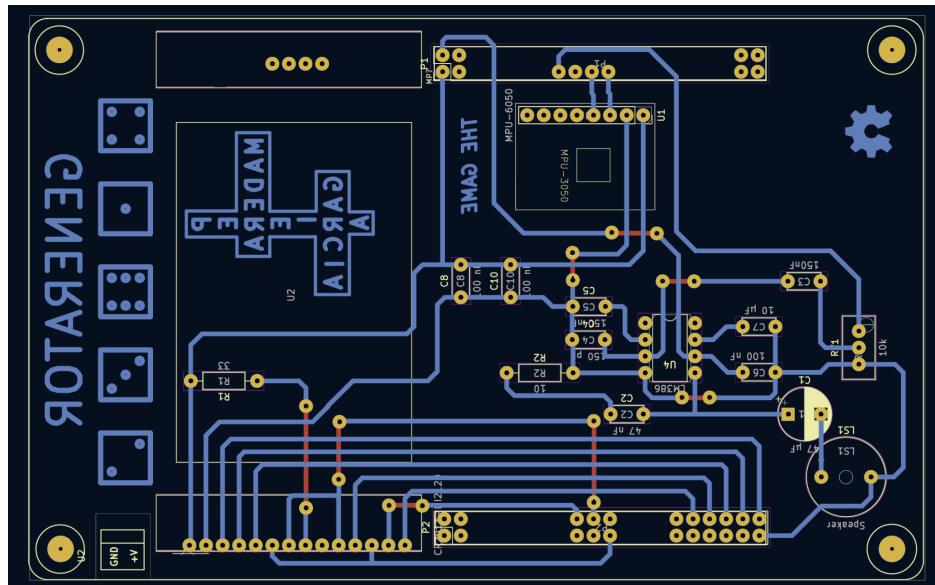


Figura 16: Diseño PCB del GENERATOR

4. FABRICACIÓN

Una vez terminado y corregido el diseño, se procedió con la etapa de fabricación del PCB. A continuación, se presentan imágenes descriptivas de dicho proceso.

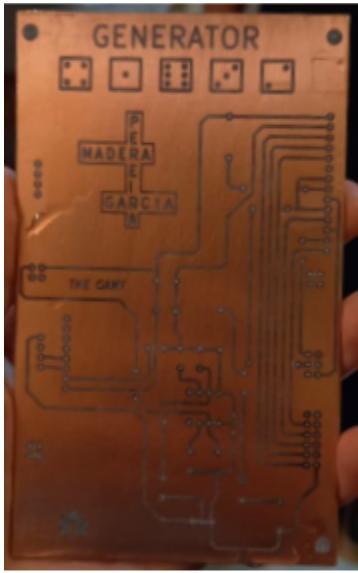


Figura 15: Placa luego de la impresión.

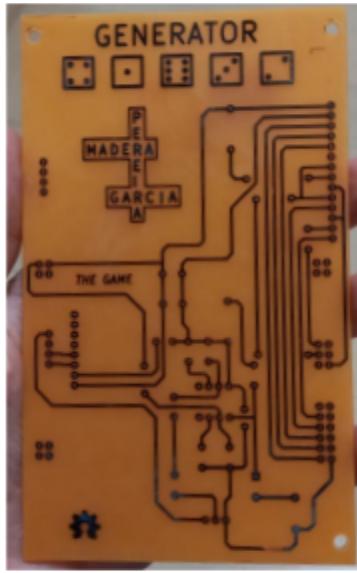


Figura 16: Placa luego del baño de ácido.

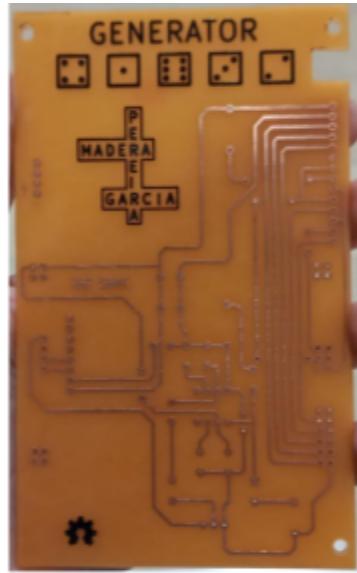
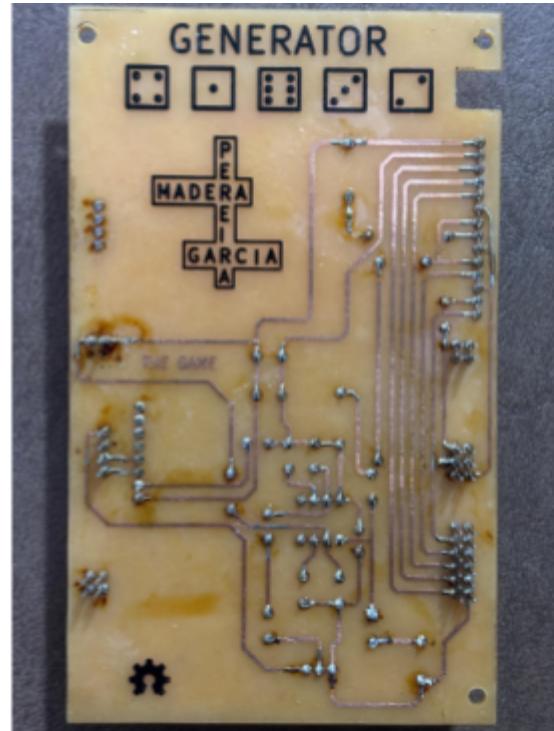


Figura 17: Placa des taladrar orificios y limpiar las pistas.



Figura 18: Placa con los puertos soldados y los componentes conectados



DISEÑO DE SOFTWARE

1. ARQUITECTURA DEL FIRMWARE

El firmware del proyecto se basa en una arquitectura de tipo *Time-Triggered*, la cual contará con una máquina de estados que se actualizará cada un tiempo determinado. En los distintos estados de esta máquina, se leerán las entradas del sistema provenientes tanto del módulo táctil como del sensor del movimiento. En relación a estas entradas, se permitirá pasar de la pantalla de inicio a la pantalla con los dados a tirar, se podrá seleccionar y deseleccionar un dado, y se podrá detectar si el dispositivo se está agitando para así poder calcular aleatoriamente los valores de los dados. A su vez, se harán las mínimas actualizaciones en la interfaz gráfica para reflejar el cambio de los estados (con la menor latencia posible) y las variables del juego, y se emitirán por el parlante los sonidos correspondientes al momento del juego. Cada iteración de la máquina de estados se ejecutará cada $25ms$, siendo este el tiempo elegido para poder dar un buen tiempo de respuesta a los toques que se hagan en la pantalla.

Cabe destacar que este no fue el primer modelo de arquitectura pensado para el proyecto. Inicialmente, se pensó en una arquitectura de firmware con aspectos de tipo *background-foreground*, donde la máquina de estados seguiría funcionando con actualizaciones periódicas para leer el sensor de movimiento, pero con las acciones relacionadas al táctil se ejecutarían solo al detectar una interrupción por flanco de bajada proveniente de este dispositivo, indicando que la pantalla estaba siendo tocada. Se decidió migrar a un modelo completamente *Time-Triggered* debido a las dificultades que presentó implementar la lectura de los valores de la pantalla y la implementación de la interrupción de la misma en conjunto; por cuestiones de cronograma, se reemplazó la lectura por interrupción por una lectura por *polling*. Para los fines del juego, en el cual la pantalla es tocada con los dedos de la mano y en la cual no se requiere de una gran resolución o un tiempo inmediato de respuesta, es suficiente con detectar una cantidad limitada de coordenadas iguales leídas consecutivamente para interpretar que se está tocando cierto punto de la pantalla, y así poder realizar las distintas acciones del juego.

2. MÁQUINA DE ESTADOS DEL SISTEMA

La máquina de estados resultante consta de 3 estados principales: *inicio*, que es donde se muestra la pantalla inicial del sistema; *reposo*, que representa cuando el sistema está en el juego pero el dispositivo no se está moviendo; y *sacudiendo*, que representa el momento del juego donde el dispositivo está en movimiento con al menos un dado seleccionado para mezclar. La máquina cuenta con 2 entradas, la del sensor de

movimiento (AC) y la del táctil (START), las cuales permiten transicionar entre estados. El táctil permite pasar de *inicio* a *reposo*, y el sensor de *reposo* a *sacudiendo* y viceversa. El táctil permite, además, cambiar el estado de los dados cuando se está en el estado *reposo* para permitir que los mismos sean aleatorizados o no. Esta descripción de la máquina de estados puede verse reflejada en la representación gráfica de la figura 19.

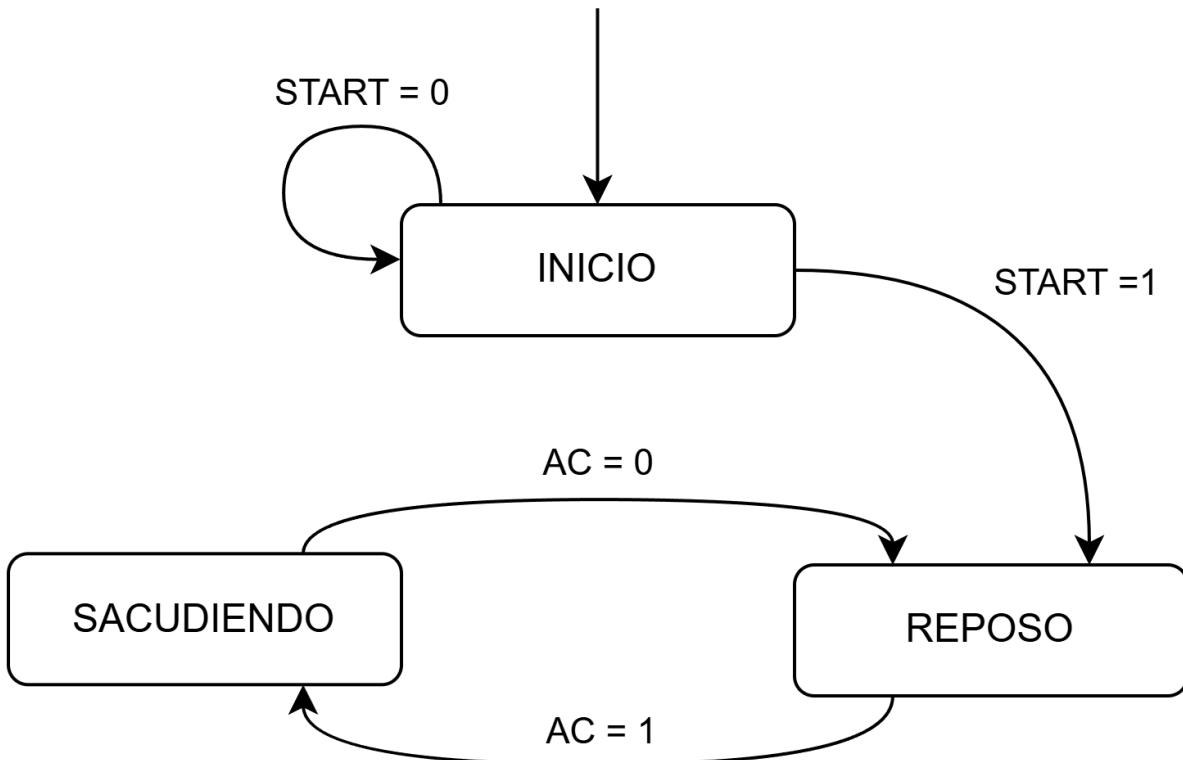


Figura 19: Diagrama de la máquina de estados

Este diagrama conceptual de máquina de estados se tradujo en el siguiente pseudocódigo de control:

```
Iniciar componentes
Bucle infinito
    En caso de que el estado sea
        "Inicio"
            Mostrar menú de inicio
            Leer táctil
            Si se presiona el botón inicio
                Cambiar a estado "Reposo"
        "Reposo"
            Si es la primera vez en "reposto" mostrar datos "?"
            Sino si vengo de "sacudiendo"
                Aleatorizar dados
                Emitir sonido de dados tirados
                Calcular juego
                Si el juego es "Generala" emitir sonido
            Leer táctil
            Si se toco un dado
                Si está seleccionado
                    Deseleccionar dado
                Si no está seleccionado
                    Seleccionar dado
                Si se está sacudiendo y hay al menos un dado deselecciona
                    Cambiar a estado "Sacudiendo"
        "Sacudiendo"
            Si se sigue sacudiendo
                Aleatorizar dados
                Emitir sonido de mezcla
            Sino
                Cambiar a estado "Reposo"
    Retardo
```

3. GENERACIÓN DE VALORES ALEATORIOS

Un aspecto fundamental del proyecto consiste en la posibilidad de generar números de forma aleatoria; esto, sin embargo, es imposible de realizar en la computadora, debido al aspecto determinístico de su funcionamiento. En el lenguaje de programación utilizado, la función `rand()` permite generar una secuencia de valores pseudo-aleatorios determinada para la "semilla" que posee el programa; para poder aleatorizar los resultados de esta secuencia, es necesario poder cambiar de la forma más impredecible posible el valor de la semilla.

Para ello, se determinó un valor de semilla relativo al tiempo transcurrido en el juego. Se contabilizaron 2 valores relacionados al tiempo transcurrido: la cantidad de actualizaciones de la máquina de estados mientras se encuentra en la pantalla de inicio del juego, y la cantidad de actualizaciones de la máquina de estados mientras se encuentra por primera vez en el estado de reposo, previamente a realizar la primera tirada. Al momento de detectar movimiento del dispositivo por primera vez, se multiplican ambos valores y se toma la semilla en relación al producto devuelto por ese valor. Al realizarse las actualizaciones cada $25ms$, la resolución de ambos valores resulta adecuada para que un usuario común no logre reproducir dichos períodos de manera consistente para "trucar" el juego.

Además, dentro de la misma secuencia de una semilla, el tiempo que se pase agitando el dispositivo permitirá "desplazarse" en la obtención de valores de esta secuencia única. Esto se debe a que, al estar en el estado *sacudiendo*, se generan valores aleatorios para imprimir en los dados. En consecuencia, según el tiempo que se pase agitando la consola y la cantidad de dados que se estén mezclando, se consumirán más o menos valores de la secuencia, modificando el resultado de los dados que se estén tirando.

Si bien esta solución es apropiada para los fines del proyecto, se podría haber apelado a la generación de una semilla en función a una señal de ruido presente en el microcontrolador, la cual se acerca a una señal aleatoria. Si bien esta solución brinda más confianza a la aleatoriedad de la semilla, la baja probabilidad de repetición de semilla con el método implementado hace que no sea necesario utilizar este proceso, simplificando así el diseño del Hardware del dispositivo.

REPOSITORIO DEL PROYECTO

En el siguiente [vínculo](#) se encuentra el repositorio que contiene el firmware completo del proyecto, así como las imágenes y archivos del diseño esquemático y el diseño PCB.

VALIDACIÓN DE REQUERIMIENTOS

En la siguiente [Playlist](#) se encuentran subidos los ensayos de validación de cada uno de los requerimientos funcionales que han sido solicitados. De todos modos, se enumeran a continuación cada uno de los videos, referenciando los requerimientos que validan:

- Juego completo: muestra el sistema andando en su totalidad, corroborando los objetivos principales del proyecto;
- Encendido y Apagado del dispositivo: corroboran los requerimientos relacionados a la alimentación del equipo;
- Estado inicial + Tirada normal: Corrobora los requerimientos referentes a generar valores aleatorios para los dados, así como todos los requerimientos relacionados al sensor de movimiento y los requerimientos del sonido a mezclar y tirar los dados;
- Tirada con dados seleccionados y deseleccionados: Corrobora los requerimientos relacionados al táctil y la interacción con el sistema, guardando y liberando datos para tirar;
- Juegos posibles : Corrobora los requerimientos relacionados a mostrar todos los juegos posibles con sus respectivos puntajes (para facilitar esta demostración se forzaron los resultados con un código especial);
- Generala!: Corrobora el requerimiento de sonido al obtener el juego *Generala* con los 5 dados.

Se pueden apreciar en los videos, además, que el proyecto cumple con los requerimientos de Hardware solicitados. En cuanto a los requerimientos no funcionales, los relacionados a la apariencia física del proyecto se pueden ver en el video. Hubo 2 requerimientos no funcionales que fueron modificados a lo largo del proyecto:

1. *"El código debe ser desarrollado con los entornos de trabajo "CIAA Launcher" e "IDE Eclipse".* Este requerimiento no fue respetado estrictamente durante todo el proyecto, pues también resultó cómodo trabajar con el entorno *"Embedded IDE"* dentro del *"CIAA Launcher"*;
2. *"Deben estar incluidos los nombres o firmas de los desarrolladores del proyecto".* Si bien este requerimiento no fue cumplido dentro de la interfaz del Software, sí se encuentra la firma de los desarrolladores dentro del diseño de Hardware.

CONCLUSIONES

A partir de las validaciones realizadas, se puede concluir en que los objetivos primarios del proyecto han sido cumplidos. Se ha logrado un circuito embebido que recrea los principios básicos del juego de la Generala.

Más allá de que no es una implementación del juego completo, por no tener el límite de 3 tiros para armar un juego y no simular las tiradas limitadas en las cuales se desarrolla una partida, permite recrear la función que tiene el cubilete en sí, que era la meta de este primer prototipo. Además, se ha complementado esta funcionalidad con elementos audiovisuales que no sólo constituyen una estética llamativa, si no que también vuelven interesante la propuesta, mostrando su potencial de cara al público general.

Desde una perspectiva implementativa, ciertos aspectos del proyecto podrían haberse desarrollado con mayor precisión. Se podría haber perfeccionado la aleatoriedad del proyecto utilizando una señal de ruido para generar la semilla del programa, aunque la solución por Software propuesta resulta satisfactoria con creces para los fines del prototipo. Además, por falta de documentación adecuada y por cuestiones de cronograma, no se ha podido integrar la lectura del táctil con su activación por interrupción, lo cual llevó a implementar una lectura por polling, la cual genera algo de latencia hasta detectar un toque y complejiza la detección de un toque en un dado o botón. A pesar de estas inconveniencias, la implementación final sigue siendo adecuada para la interacción humana con la pantalla, pues se ha minimizado el tiempo de espera a uno que no resulte molesto en el uso cotidiano. No obstante, de retomar el trabajo en el prototipo, trabajar en estos aspectos simplificaría la implementación y solucionaría futuros inconvenientes que puedan surgir.

El proyecto resulta un buen primer prototipo para una potencial consola de datos; como posible extensión, se podría tomar la simulación de tiradas de Generala para implementar una partida completa, con turnos acotados para lograr todos los juegos y acumulando el puntaje dentro de una única partida. Se podría incluso implementar, contemplando los cambios en Hardware necesarios, la conexión a Internet, permitiendo guardar puntajes en la nube o habilitando un modo multijugador con otra consola. El proyecto incluso puede expandirse más allá de la generala: se pueden incorporar múltiples juegos al firmware actual, pudiendo seleccionar al inicio el juego que más prefiera jugar el usuario en ese momento.

Gracias al tiempo invertido y al trabajo en equipo de sus desarrolladores, se ha podido construir un sistema sencillo que permite simular un clásico juego popular. GENERATOR resultó ser un dispositivo electrónico llamativo para los niños y jóvenes, invitando a las generaciones digitales a redescubrir la diversión en los dados y cumpliendo así su misión principal.

REFERENCIAS BIBLIOGRÁFICAS

- [Pseudo-Randomización de Números en C](#)
- [Generación de Números Aleatorios en las Computadoras](#)
- [Proyecto CIAA](#)
- [Información General - EDU CIAA](#)
- [Pines de la placa EDU-CIAA](#)
- [EDU-CIAA-datasheet_fabricante_LPC435X_3X_2X_1X.pdf](#)
- [Sensor de movimiento: MPU6050 - DataSheet](#)
- [Amplificador LM386 - DataSheet](#)
- [Display LCD: ILI9341 - DataSheet](#)
- [Librería del controlador para ILI9341](#)
- [Membrana táctil: XPT2046 - DataSheet](#)
- [Possible powerbank para alimentación](#)
- [Software con el cual se desarrolló el esquemático](#)
- [Librería de referencia para la implementación del táctil XPT2046](#)
- [Video 1: referencia para rediseñar el circuito eléctrico para los efectos de sonido](#)
- [Guía de usuario de la librería implementada para ILI9341](#)

ANEXO

A. DIVISIÓN DE TAREAS

A continuación, se especifican las distintas tareas del proyecto y su división entre los integrantes del grupo:

Tabla A: División de Tareas y responsabilidades

Tarea	Responsable	Colaborador
Elección del proyecto y elección de periféricos a utilizar	Garcia Iacovelli Tobias, Madera Ramiro, Pereira Ulises	Fernandez M. Juan Ignacio
Especificación de requerimientos	Garcia Iacovelli Tobias, Madera Ramiro, Pereira Ulises	-
Investigación y análisis de plataforma EDU-CIAA y herramientas anexas	Garcia Iacovelli Tobias, Madera Ramiro, Pereira Ulises	-
Corrección y supervisión de la documentación	Pereira Ulises	Garcia Iacovelli Tobias, Madera Ramiro
Investigación de protocolo de comunicación SPI	Madera Ramiro	-
Investigación de protocolo I2C	Garcia Iacovelli Tobias	-
Investigación y cálculos del consumo energético del sistema	Garcia Iacovelli Tobias, Madera Ramiro	-
Investigación sobre el funcionamiento del display y controlador ILI9341	Madera Ramiro	-
Implementación de librerías para controlar el display	Madera Ramiro	-
Ampliación de funciones gráficas del display	Madera Ramiro	-
Investigación de	Garcia Iacovelli Tobias	-

características del acelerómetro MPU6050		
Desarrollo y testeo del módulo controlador del MPU6050	Garcia Iacobelli Tobias	-
Investigación de características de los componentes de sonido	Garcia Iacobelli Tobias	-
Rediseño del circuito eléctrico para la implementación de efectos de sonido	Garcia Iacobelli Tobias	-
Investigación sobre el funcionamiento del touch y controlador XPT2046	Pereira Ulises	-
Implementación y prueba de librerías para el uso del XPT2046	Pereira Ulises	-
Investigación respecto a la generación de números aleatorios	Pereira Ulises	-
Capacitación sobre la herramienta KiCad y la creación de esquemas	Madera Ramiro	-
Diseño del circuito esquemático	Madera Ramiro	Garcia Iacobelli Tobias, Pereira Ulises
Investigación de diseño de huellas y PCB en KiCad	Madera Ramiro	-
Diseño del circuito impreso (PCB)	Madera Ramiro	
Fabricación y corrección del circuito impreso	Garcia Iacobelli Tobias, Madera Ramiro, Pereira Ulises	-
Soldadura y ensamblaje del dispositivo	Garcia Iacobelli Tobias, Madera Ramiro, Pereira Ulises	-

Pruebas y correcciones del Hardware	Garcia Iacovelli Tobias, Madera Ramiro, Pereira Ulises	-
Diseño de la Arquitectura de Software y la jerarquía de módulos	Pereira Ulises	Garcia Iacovelli Tobias, Madera Ramiro
Diseño de la máquina de estados y pseudocódigo del programa	Pereira Ulises	Garcia Iacovelli Tobias, Madera Ramiro
Programación de la máquina de estados	Pereira Ulises	Garcia Iacovelli Tobias, Madera Ramiro
Diseño gráfico del juego	Madera Ramiro	Garcia Iacovelli Tobias, Pereira Ulises
Diseño de efectos de sonido	Garcia Iacovelli Tobias	-
Incorporación del táctil al diseño gráfico e interacción con datos y botones	Pereira Ulises	-
Implementación del cálculo de puntajes	Pereira Ulises	-
Diseño e implementación del randomizado de datos	Pereira Ulises	Madera Ramiro
Pruebas de verificación y validación	Garcia Iacovelli Tobias, Pereira Ulises	Madera Ramiro
Armado de informes de avance	Garcia Iacovelli Tobias, Madera Ramiro, Pereira Ulises	-
Armado de informe final	Garcia Iacovelli Tobias, Madera Ramiro, Pereira Ulises	-

B. CRONOGRAMA FINAL

A continuación, se deja el cronograma de trabajo final que se ha adoptado en el proyecto, contemplando los imprevistos y las modificaciones mencionadas en el informe:

 Diagrama de Gantt - Grupo 8 - Taller de Proyecto I



FECHA	AGOSTO			SEPTIEMBRE				OCTUBRE					NOVIEMBRE				DICIEMBRE			FEBRERO		
	15	22	29	5	12	19	26	3	10	17	24	31	7	14	21	28	5	12	3	10	17	
SEMANA:	1	2		3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Elección de Grupo y Proyecto																						
Objetivos y Requerimientos																						
Diseño de Software																						
Investigación de periféricos																						
Programación de drivers																						
Simulación y Pruebas																						
Integración del código																						
Diseño Esquemático																						
Diseño PCB																						
Fabricación																						
Ensamble																						
Ens Ayos																						
Informe Final																						
Muestra																						
Informe inicial de proyecto																						
Primer Informe de Avance																						
Segundo Informe de Avance																						

Tabla B: Cronograma final del proyecto

C. CONSUMOS ELÉCTRICOS DE LOS COMPONENTES

Para estimar la autonomía del dispositivo, se consideraron los siguientes consumos eléctricos de los componentes principales:

- MCU con todos sus periféricos: 200 mA
- Sensor de movimiento MPU6050: 3,9 mA
- Display táctil TFT: 100 mA (consumo máximo)
- Amplificador LM386: 3,6 mA

Esto da un consumo total aproximado de 308 mA en funcionamiento.

El dispositivo es alimentado por un powerbank de 5100 mAh, lo que nos permite estimar la duración de la batería de la siguiente manera:

$$\text{Autonomía} = \frac{\text{Capacidad de la batería}}{\text{Consumo total del sistema}} = \frac{5100 \text{ mAh}}{308 \text{ mA}} \approx 16.56 \text{ horas}$$

Esto significa que el sistema podría funcionar de manera continua durante aproximadamente 16 horas y 30 minutos antes de requerir una recarga. Sin embargo, esta estimación supone un uso constante y máximo de todos los componentes. En un uso real, donde algunos periféricos pueden no estar activos todo el tiempo, la autonomía efectiva podría ser mayor.

D. SELECCIÓN DE DATOS Y BOTÓN DE INICIO

La implementación del controlador táctil tuvo como objetivo poder seleccionar los datos dentro del juego, así como poder interactuar con un botón de inicio del juego. Para ello, se utilizó un programa de lectura por *polling* cada 25ms, que devolviera los valores detectados en la lectura. La lectura por defecto al no tocar la pantalla devolvía valores que

cambiaban constantemente, pero que pueden repetirse una pequeña cantidad de lecturas consecutivas.

Para poder detectar lecturas concretas, se declaró una estructura de valor de pantalla, en la cual se guardan las coordenadas actuales, así como la cantidad de lecturas consecutivas en las que las mismas han sido leídas. Se definió que al superar el umbral de 5 lecturas, la lectura devolverá por única vez que la pantalla está siendo tocada, solucionando no solo la diferenciación de lecturas no significativas sino también la repetición no deseada de una acción por mantener presionado el mismo punto. A este contador se le agregó además una tolerancia de una lectura disidente si la misma difería en una unidad una sola de las 2 coordenadas, visto que en las pruebas del driver se detectó que esto solía pasar por error humano de mover levemente el dedo al tocar la pantalla.

Una vez implementada la detección del toque en un punto, resta poder identificar si el punto leído se encuentra dentro de un botón. Para ello, se creó la estructura *Rectangle*, que almacena las 4 esquinas de un botón, y la función *PointInRectangle()*, que dada una lectura y las esquinas de un botón permite ver, si la lectura marca que se está tocando un punto, si el punto está dentro del botón dado. Esta función, que devuelve un valor booleano, se usa en la función *InStart()*, que chequea si el punto se encuentra en el botón de inicio, y en la función *SelectDado()*, que la llama una vez por cada uno de los 5 dados y, de devolver verdadero en algún caso, devuelve el número de dado que encontró; si no encuentra ninguno, devuelve el número 0.

Los 5 dados y el botón de inicio tienen coordenadas constantes y predeterminadas, por lo cual su asignación se hace en la inicialización del táctil y no hace falta pasar sus parámetros para hacerla.

E. CÁLCULO DE JUEGOS Y PUNTAJES

Dentro del código principal del programa, se desarrolló un código que, dados los 5 valores que tengan los dados en un momento dado, calcule el tipo de juego que forman y el puntaje del mismo. Esto contempla tanto los juegos especiales como los juegos de números. Los juegos fueron definidos como:

- Generala (GEN): 5 dados del mismo número. Vale 50 puntos;
- Póker (POK): 4 dados del mismo número y uno distinto. Vale 40 puntos;
- Full (FUL): 3 dados iguales y otros 2 dados distintos, pero iguales entre sí. Vale 30 puntos;
- Escalera: 5 dados con números consecutivos. Hay 2 tipos:
 - Escalera Mayor (EMA): Del 2 al 6. Vale 25 puntos;
 - Escalera Menor (EME): Del 1 al 5. Vale 20 puntos;
- Juegos Normales: son los juegos que se seleccionan en caso de no darse ningún juego especial, y consisten en tomar uno de los números y calcular el puntaje como

el producto entre dicho número y su cantidad de apariciones entre los 5 dados. Se toma, de todos los posibles, el juego con mayor puntaje. Hay 6 tipos según el número:

- Juego de unos (UNO);
- Juego de dos (DOS);
- Juego de tres (TRE);
- Juego de cuatro (CUA);
- Juego de cinco (CIN);
- Juego de seis (SEI);

La estrategia para poder diferenciar cada uno de estos juegos se basó en calcular 2 histogramas diferentes; uno que cuente cuántas veces aparece cada número, y otro que cuente cuántos números tienen cierta cantidad de apariciones. A modo de ejemplo, el histograma de una tirada *full* que conste de los dados "4 4 4 5 5" va a tener dentro del primer histograma el valor 3 para el número 4 y el valor 2 para el número 5; dentro del segundo histograma, va a tener el valor 1 tanto para las categoría de 3 apariciones (el 4 aparece 3 veces) como para 2 apariciones (el 5 aparece 2 veces). Este segundo histograma de cantidad de apariciones es el que será clave para poder distinguir los juegos especiales, pues cada juego tiene una configuración específica en este histograma:

- Tener Generala implica que el histograma de apariciones tendrá un 1 para las 5 apariciones, pues habrá un solo número que aparece 5 veces;
- Tener Póker implica un 1 en la categoría de 4 apariciones;
- Tener Full implica un 1 en las categorías de 3 y 2 apariciones;
- Tener escalera implica tener un 5 en la categoría de una aparición. Sin embargo, este caso puede implicar una escalera mayor, menor o una secuencia cortada que no sea ningún juego especial. Para ello, se lleva registro en 2 variables booleanas *uno* y *seis* respecto a si los números 1 y 6 aparecen en los dados. Con esta información, sí se puede deducir el tipo de escalera:
 - La Escalera Mayor lleva el 6 y no el 1, por lo que cuando *uno* es falso y *seis* es verdadero, se tiene una Escalera Mayor;
 - La Escalera Menor lleva el 1 y no el 6, por lo que cuando *uno* es verdadero y *seis* es falso, se tiene una Escalera Menor;
 - Si *uno* y *seis* son verdaderos a la vez, se va a tener una secuencia cortada en la cual no aparece uno de los valores entre 2 y 5, por lo que no hay escalera alguna;
 - Si *uno* y *seis* son falsos a la vez, es imposible tener un 5 en el histograma de 1 aparición, pues habrá un número con mínimo 2 apariciones y otros 2 que no salen (1 y 6), por lo que este caso no se tiene en cuenta;

En caso de no encontrar un juego especial en estas situaciones, se rescata el mejor de los juegos normales; estos se calculan a la vez que se calcula el histograma de



apariciones, calculando el puntaje del juego normal para cada número y quedándose con el máximo de estos 6 puntajes.

Para cada uno de los juegos, se guarda el resultado del juego en una variable enumerativa que declara todos estos juegos, la cual es el valor de retorno de la función y es con la cual se selecciona el texto a imprimir en pantalla. El puntaje, por otro lado, es devuelto como parámetro por referencia al programa principal. Cabe destacar que, para el primer estado de reposo, se declara el juego de tipo *nada*, el cual indica que aún no hay juego y permite imprimir el texto “GO!” en la pantalla.