PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS DIGITALES (IEE256)

INFORME FINAL

THE LAST HOPE

HORARIO H-052B

ASESOR:

Ing. Rodia Santivañez Santivañez

INTEGRANTES:

Huayapa Leon, Victor Javier 20181501

Meza Rojas, Harif Joe 20180785

FECHA: 10/07/2020

2020-1

Índice

THE LAST HOPE

1. Resumen del proyecto

El microcontrolador TM4C123GH6PM nos ofrece una amplia gama de aplicaciones, entre ellas, el desarrollar videojuegos. ”The Last Hope” toma como inspiración el legendario y antiquísimo juego Space Invaders, del cual adquiere la temática de la defensa del planeta Tierra frente a un ataque “alienígena”.
The Last Hope utiliza los módulos atendidos a lo largo del curso Sistemas Digitales: GPIO, UART, Y SYSTICK; adicionalmente, trabaja de manera conjunta con otros tales como SSI, DAC, así como la interacción con la pantalla ST7735 y diversas dinámicas propias del juego.
A nivel de organización lógica, el juego incluye un menú principal con diversas opciones, cada una de las cuales cumple una finalidad específica; Dentro de ellas, la más relevante viene a ser la de ‘Iniciar Partida’, ya que esta abre un submenú que contiene la tabla de posiciones (o leaderboard) y la opción para iniciar una nueva partida (ingresar al juego en sí). El juego como tal consiste en la acción de una nave manipulada por el usuario que enfrentará a 12 aliens distribuidos en 3 filas de 4 cada una y se desarrolla en 3 niveles. En cuanto el jugador pierda un nivel o gane todo el juego, se le solicitará los datos para registrar su puntaje en el leaderboard y, posteriormente, se le retornará al menú principal repitiendo el proceso.

Objetivo general:

Diseñar, implementar y simular un videojuego tipo arcade, con un entorno amigable, sencillo de usar y entretenido.
El videojuego se desarrolla utilizando el microcontrolador TM4C123GH6PM y el kit de desarrollo LaunchPad Evaluation Board (EK-TM4C123GXL); a nivel de software, se trabaja con el IDE Keil uVision.

Objetivos específicos:

- Desarrollar los módulos que sean indispensables para el desarrollo del videojuego
- Programar en lenguaje C la lógica del programa, teniendo en consideración y como meta la eficiencia y fácil legibilidad del código
- Elaborar los diagramas requeridos para la implementación física del proyecto, buscando siempre la máxima calidad y minimizando costos
- Comprobar a través de la simulación en KEIL uVision el correcto funcionamiento de juego y proponer optimizaciones a futuro

Al desarrollar la codificación se optó por regirse por el paradigma de la programación modular, en el que se ataca el problema dividiéndolo en módulos cuya manipulación resulta ser más sencilla; estos son desarrolladas en subprogramas y/o subfunciones. Esto se evidencia a lo largo del proyecto y en todos los niveles de jerarquía del mismo. Por ejemplo, el archivo main.c (que contiene el programa principal y el Systick_Handler) ocupa menos de 100 líneas de código incluyendo la presentación del trabajo. Además, hay jerarquía de funciones que, apoyadas mediante la creación y adaptación de librerías pertinentes, permitió que el proyecto se desarrolle de una manera organizada y sea entendible incluso para alguien que revise el código por primera vez y quiera implementarlo o modificarlo

2. Metodología empleada

División de trabajo:
A continuación se presenta el cronograma de actividades. Las actividades se referirán al alumno M y alumno H como responsables. Esta información estará entre paréntesis, así como la sesión de laboratorio en que se espera que esté listo (ejm: (M-8) Desarrollar… [Alumno M debe presentar en la semana 8 la tarea...])
Alumno M: Harif Meza
Alumno H: Victor Huayapa

1) Desarrollar la biblioteca relacionadas con el uso del tiempo (SysTick):
    - (M-8)Desarrollar la función void SysTick_Init()
    - (M-8)Desarrollar la función void SysTick_wait(uint32_t delay)
    - (M-8)Desarrollar la función void SysTick_wait10ms(uint32_t delay)
    - (M-8)Analizar si se puede simplificar la programación del código final mediante una nueva función que trabaje con el módulo empleado e implementarla
    - (HyM-8)Desarrollaron la función void ConfiguraSystick_1ms()
2) Desarrollar las funciones relacionadas con el uso de la pantalla LCD ST7735. Tomados de Valvano, Jonathan.
    - (H-8)Preparar los arreglos que permitirán mostrar la animación de los aliens
    - (H-8)Preparar los arreglos que permitirán mostrar la nave del jugador
    - (H-8)Preparar los arreglos que permitirán mostrar los proyectiles
    - (H-8)Preparar los arreglos que permitirán mostrar los mensajes especiales según circunstancia
    - (M-8)Preparar los arreglos que permitirán mostrar las instrucciones
    - (M-8)Preparar los arreglos que permitirán mostrar los menús
    - (H-8)Investigación y correcta utilización de las funciones que muestran mapas de bits y ubican cursor:
      void ST7735_DrawBitmap(int16_t x, int16_t y, const uint16_t *image, int16_t w, int16_t h)
      ST7735_SetCursor(uint32_t newX, uint32_t newY)
    - (M-8) Investigación y correcta utilización de las funciones que escriben cadenas y borran la pantalla:
      uint32_t ST7735_DrawString(uint16_t x, uint16_t y, char *pt, int16_t textColor)
      ST7735_FillScreen(0)
    - (MyH-8)Analizar si se puede simplificar la programación del código final mediante una nueva función que trabaje con el módulo empleado o los componentes empleados e implementarla
3) Desarrollar las funciones relacionadas con el uso de los pulsadores y el led
    - (H-8)Desarrollar la función void config_puertoF(void)
    - (H-8)Desarrollar la función void sondea_pulsador(uint8_t *antSW,uint8_t mask_SW,uint8_t *flanco_bajadaSW)

- (H-8)Desarrollar la función void maneja_led(uint8_t color,uint8_t encender)
- (H-8)Desarrollar la función void parpadea_led(uint8_t color, uint32_t retardo)
- (HyM-8)Analizar si se puede simplificar la programación del código final mediante una nueva función que trabaje con el módulo empleado o los componentes empleados e implementarla

4) Desarrollar las funciones relacionadas con el uso del teclado
   - (M-8)Desarrollar la función void UART_Init(void)
   - (M-8)Desarrollar la función uint8_t UART_Caracter()
   - (M-8)Desarrollar la función void UART_OutString(char *pt)
   - (M-8)Desarrollar la función void UART_InString(char *bufPt, uint16_t max)
   - (MyH-8)Analizar si se puede simplificar la programación del código final mediante una nueva función que trabaje con el módulo empleado los componentes empleados e implementarla
   - (H-8)Desarrolló la función void sondea_UART(uint8_t *flag_hay_caracter,uint8_t *caracter)

5) Desarrollar las funciones relacionadas con el uso del parlante. Tomado en parte de Valvano, Jonathan y adaptada para el trabajo con el microcontrolador y archivo de depuración
   - (M-8)Investigar y desarrollar la función void DAC_Init(void)
   - (M-8)Desarrollar la función void DAC_Out(unsigned long data)
   - (MyH-8)Analizar si se puede simplificar la programación del código final mediante una nueva función que trabaje con el módulo empleado o los componentes empleados e implementarla
6) Desarrollo e implementación de las funciones a mayor escala que se usarán en el programa main
   - (M-9)Desarrollar la función void Menu()
     - (M-9)Desarrolló void nueva() //accede a “juego”
     - (M-9)Desarrolló void leaderboard() //accede a “maneja_leaderboard”
     - (M-9)Desarrolló void retornar ()
     - (M-9)Desarrolló void partida(int8_t *opc_menu)
     - (M-9)Desarrolló void instrucciones(int8_t *opc_menu)
     - (M-9)Desarrolló void mutear(int8_t *opc_menu,uint8_t *flag_sonido)
     - (M-9)Desarrolló void salir(int8_t *opc_menu,uint8_t *flag_start)
     - (H-9)Desarrolló la función void maneja_leaderboard (uint8_t opcion,uint16_t puntaje_ind)
     - (H-9) Desarrolló la función void ordena_puntajes(uint8_t letra1[],uint8_t letra2[],uint8_t letra3[],uint16_t puntaje[])
     - (H-9) Desarrolló la función void juego(uint8_t nivel,uint8_t *flag_gano,uint8_t *flag_perdio)
     - (H-9) Desarrolló la función void trabaja_jugador(uint8_t *jug_x,uint8_t *jug_y,uint8_t *cant_proy_jug, uint8_t jug_proy_x[],uint8_t jug_proy_y[])

7) (H-9)Desarrollar la función void Modo_Juego() //es toda la información que contiene void juego
    - ❖ (H-9)Desarrolló la función mueve_aliens(uint8_t *alien_x, uint8_t alien_y,int8_t *sentido,uint8_t vida_aliens[])
    - ❖ (H-9)Desarrolló la función genera_proyectil(uint8_t *cant_proy,uint8_t proy_x[], uint8_t proy_y[],uint8_t alien_x,uint8_t alien_y,uint8_t jug_x,uint8_t vida_aliens[])
    - ❖ (H-9)Desarrolló la función void mueve_proyectiles(uint8_t cant_proy, uint8_t proy_x[],uint8_t proy_y[],uint8_t caso)
    - ❖ (H-9)Desarrolló la función void modifica_arreglo_proy(uint8_t det,uint8_t *cant_proy2,uint8_t proy_x[],uint8_t proy_y[])
    - ❖ (H-9)Desarrolló la función void borra_proyectiles(uint8_t np,uint8_t *cant_proy,uint8_t proy_x[],uint8_t proy_y[],uint8_t caso)
    - ❖ (H-9)Desarrolló la función void proyectil_a_alguien(uint8_t *cant_proy,uint8_t proy_x[],uint8_t proy_y[], uint8_t jug_x,uint8_t jug_y,uint8_t *vidas_jugador,uint8_t caso)
    - ❖ (H-9)Desarrolló la función void proyectil_a_aliens(uint8_t *cant_proy_jug,uint8_t jug_proy_x[],uint8_t jug_proy_y[],uint8_t alien_x,uint8_t alien_y,uint8_t vida_aliens[])
    - ❖ (H-9)Desarrolló la función void evalua_flags(uint8_t *flag_gano,uint8_t vida_aliens[],uint8_t *flag_perdio,uint8_t vidas_jugador)
    - ❖ (HyM-9)Analizar si se puede simplificar la programación del código final mediante una nueva función e implementarla
8) Desarrollo de funciones de interrupcion, así como algunas “librerias” complementarias.
    - (M-9)Desarrollar la función void mostrar_ocultar
    - (MyH-9)Desarrollar la función void SysTick_Handler(void) //M: parte del DAC, H: parte del puntaje
9) Implementación del programa main
    - (M-10)Desarrollar la función int main(void)
    - (MyH-10)Desarrollo y demostración de la consistencia y funcionalidad del juego

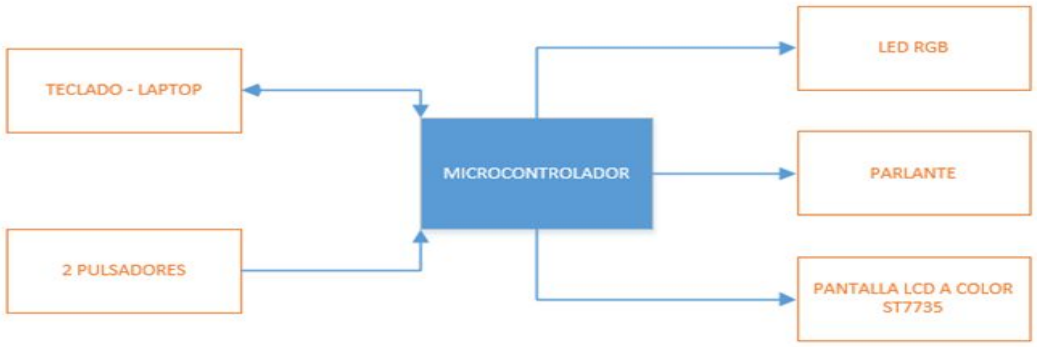
Distribución del informe final:

**	Carátula	(M-10)
**	Índice	(M-10)
1.	Resumen del proyecto	(H-10)
2.	Metodología empleada	(MyH-10)
3.	Descripción del Hardware y Software del proyecto	
3.1	Hardware	
3.1.1	Diagrama de bloques	(M-10)
3.1.2	Diagrama esquemático	(M-10)
3.1.3	Explicación del funcionamiento de hardware	(MyH-10)
3.1.4	Costo de módulos o componentes	(H-10)
3.2	Software	
3.2.1	Diagrama de flujo del programa principal y SysTick_Handler	(M-10)
3.2.2	Diagrama de flujo de las funciones relevantes	(MyH-10)
3.2.3	Explicación de algoritmos utilizados	(MyH-10)
4.	Resultados	(MyH-10)
5.	Conclusiones	(MyH-10)
6.	Observaciones y Recomendaciones	(MyH-10)
7.	Bibliografía	(MyH-10)

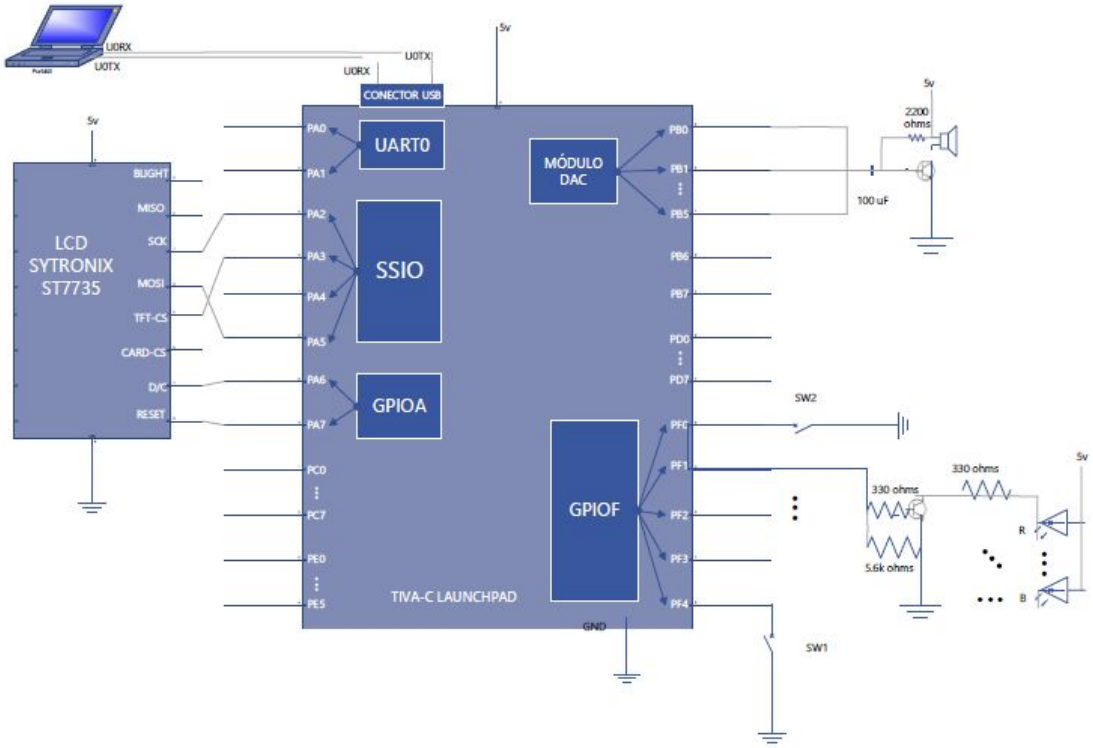
3. Descripción del Hardware y Software del proyecto

3.1. Hardware

3.1.1. Diagrama de bloques



3.1.2. Diagrama esquemático



3.1.3. Explicación del funcionamiento del hardware

Las conexiones de hardware tienen en común una fuente de alimentación de 5v.

El funcionamiento del hardware del programa es desarrollado de la siguiente manera: En primer lugar, se tiene una laptop conectada por comunicación serial al microcontrolador . Del mismo, se realiza una conexión hacia la pantalla ST7735 y hacia el DAC, la señal de este último es amplificada con un arreglo de impedancias, transistores y un speaker; en una implementación real, el sonido sería lo suficientemente potente para ser escuchado siempre y cuando el usuario no seleccione la opción mutear.

Cuando se pida ingresar alguna letra, la acción se realiza mediante el uso del teclado de un computador personal. Para interactuar con el juego, la nave se desplazará cuando el usuario presione la letra ‘a’ o ‘d’; además, cuando presione el SW1 o el SW2, la nave realizará disparos a través del cañón izquierdo y derecho, respectivamente. Cualquier otra instrucción será explicada por el mismo juego. Por ejemplo, ingresar 3 letras para actualizar el leaderboard (tabla de puntuaciones). La imagen en todo momento se ve a través de la pantalla Sitronix y la tabla de puntuaciones se ve a través del monitor serial.

3.1.4. Costo de módulos o componentes

PANTALLA LCD A COLOR SITRONIX ST7735: $19.95
https://www.adafruit.com/product/358

TERMINAL SERIAL: sin costo, se puede instalar en un ordenador

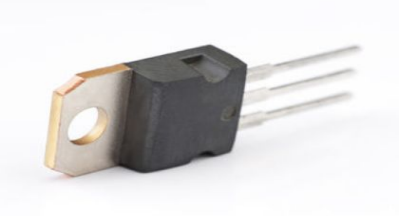
CAPACITOR 100uF 16V 0.25 EUR

https://www.electrocomponentes.es/condensadores-electroliticos/condensadores-electroliticos-100uf-16v-20-5x11mm-124-.htmlv

RESISTENCIA 2.2K Ohms 0.5$
https://tostatronic.com/store/es/componentes-pasivos/888-resistencia-22k-ohms-14w.html

AMPLIFICADOR 2SC5200N(S1,E,S) **$1.12**

https://www.avnet.com/shop/us/products/toshiba/2sc5200n-s1-e-s-307445



7345625897710/

SPEAKER 0.2W 86±3dBA $2.43

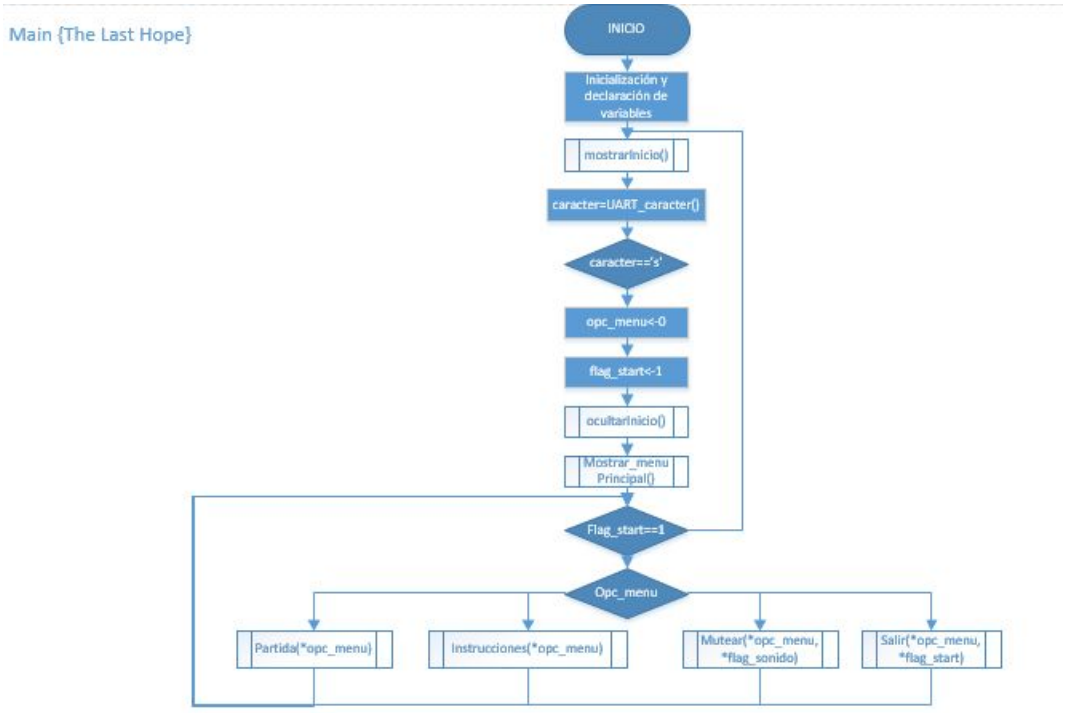
https://www.avnet.com/shop/us/products/pui-audio/ads02008mr-r-3074457345626706551/

Costo total: $24.3 = 85.57 soles

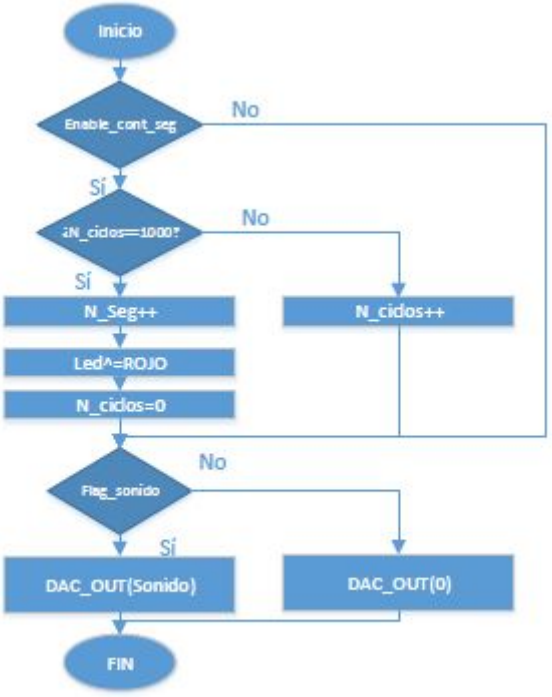
3.2. Software

En caso se desee realizar una mejor inspección de los diagramas de flujo de todas las funciones, estos están adjuntados al informe en un archivo .rar

3.2.1. Diagrama de flujo del programa principal y SysTick Handler

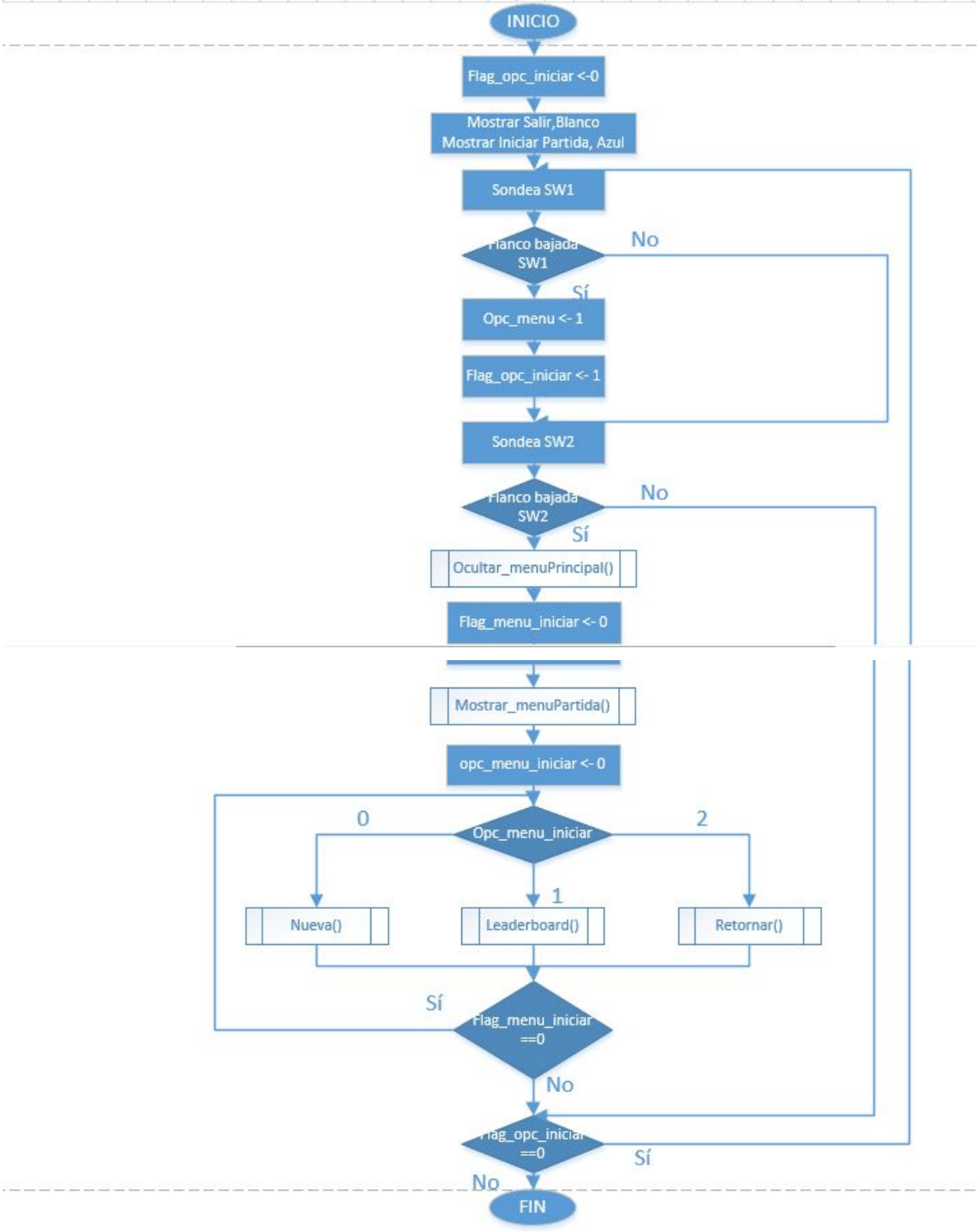


SysTick_Handler

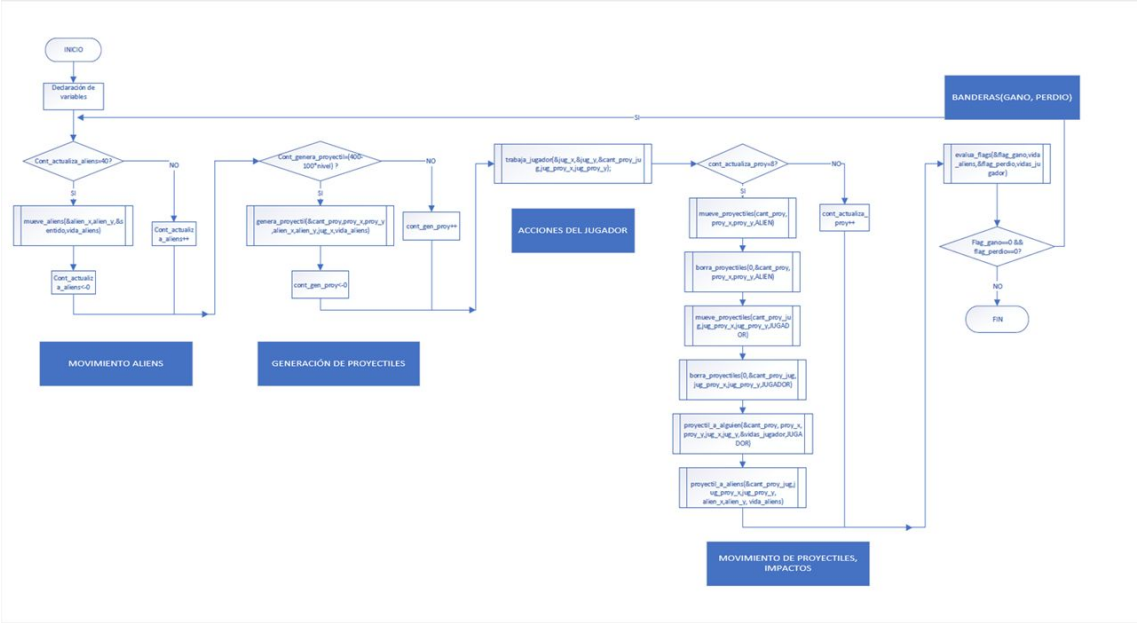


3.2.2. Diagrama de flujo de las funciones relevantes

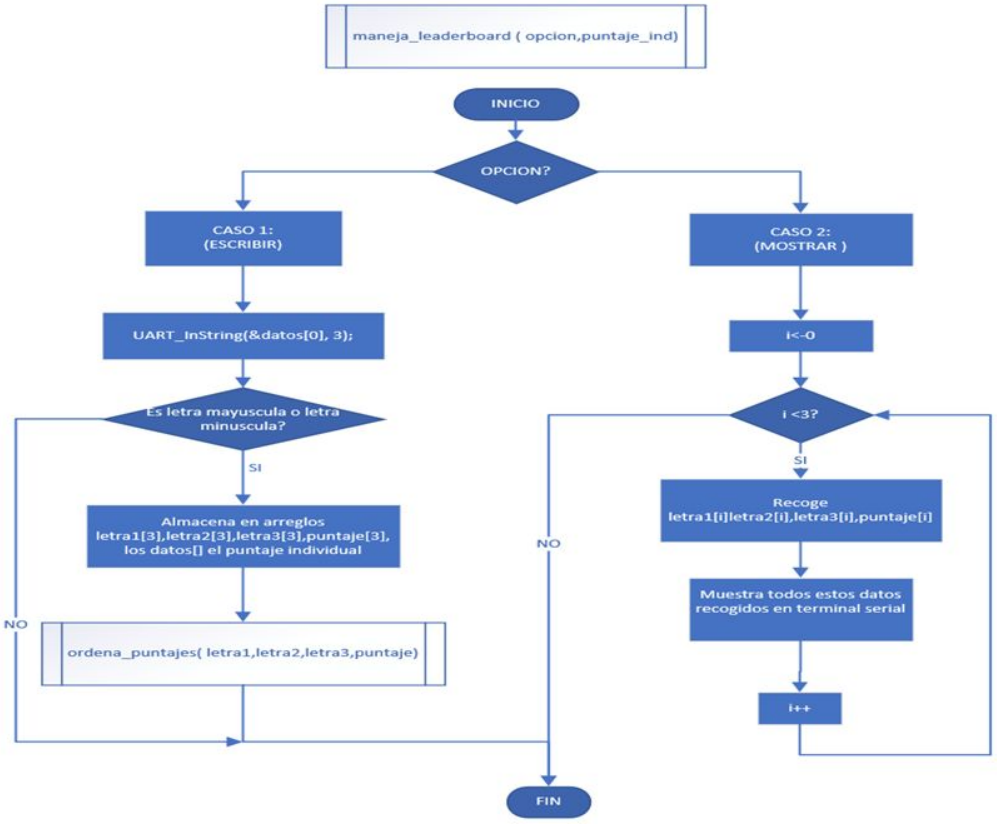
Función Partida



Función Juego



Función maneja_leaderboard



3.2.3. Explicación del algoritmos utilizados

Los algoritmos utilizados en el proyecto son los siguientes:

- Cálculo de puntaje y salida de audio: El tiempo que transcurre en segundos mientras se está en el juego (cálculo mediante interrupciones).El cálculo de puntaje se realiza de acuerdo a 2 banderas, si gano se le otorga más puntaje a quien lo hizo en menos tiempo. Si perdió se le da más puntaje a quien duró más tiempo.

- Salida de audio: La salida de audio es otra aplicación de las interrupciones, ya que nos permite reproducir sonido en cualquier momento (en caso de no ser muteado por el usuario). Se verifica en el juego el estado de una bandera, la cual informa si fue o no muteado y, según su estado, se genera o no el sonido.

- Leaderboard: Si bien es cierto el leaderboard es una tabla de posiciones, para su tratamiento total se requiere una parte que recepcione los datos del jugador y otra que muestre la tabla completa tal cual. En lo que corresponde a la recepción, se recogen los datos ingresados por el jugador, se los almacena en arreglos y se realiza un ordenamiento de los jugadores (sus puntajes) por el “Ordenamiento de burbuja”. Para mostrar la tabla tal cual, simplemente se imprimen los elementos de los arreglos de los jugadores, ya ordenados en la etapa “recepción”.

- Análisis de la dinámica de juego: Es cíclica y posee 2 banderas de salida: gano, perdio. Para las etapas de movimiento se optó por dividirlo en 3 categorías: Aliens, jugador, proyectiles. Al ser los aliens idénticos, solo se requiere la posición de un alien “base” para poder moverlos todos en un sentido u otro. Con respecto al jugador, se realiza un sondeo de teclas y pulsadores para tomar las acciones correspondientes. Los proyectiles se podría considerar lo más complicado, lo que se hizo fue utilizar arreglos que almacenen las posiciones de los proyectiles de los aliens y del jugador.Cada cierto tiempo se actualizan dichos arreglos y se evalúa si hay alguna colisión con los aliens o el jugador, de ser el caso, se elimina ese elemento particular del arreglo.

- Animaciones: El efecto del movimiento de los personajes se logra borrando y re dibujando el alien, la nave defensora o el misil según sea el caso. Esto se logra guardando la última dirección conocida y desplazando las coordenadas según la trayectoria (caso alien y misiles) u obteniendo información de la trayectoria mediante UART para el caso nave.

- El efecto de transiciones en los diferentes menús se logra escribiendo de color blanco la opción anterior y de color azul la opción actual. Además para borrar un menú, se escribe de negro (color del fondo) todas sus opciones: esto debido a que el simulador ocupa menos tiempo en pintar ciertos bits que al pintar todos los bits de color negro.

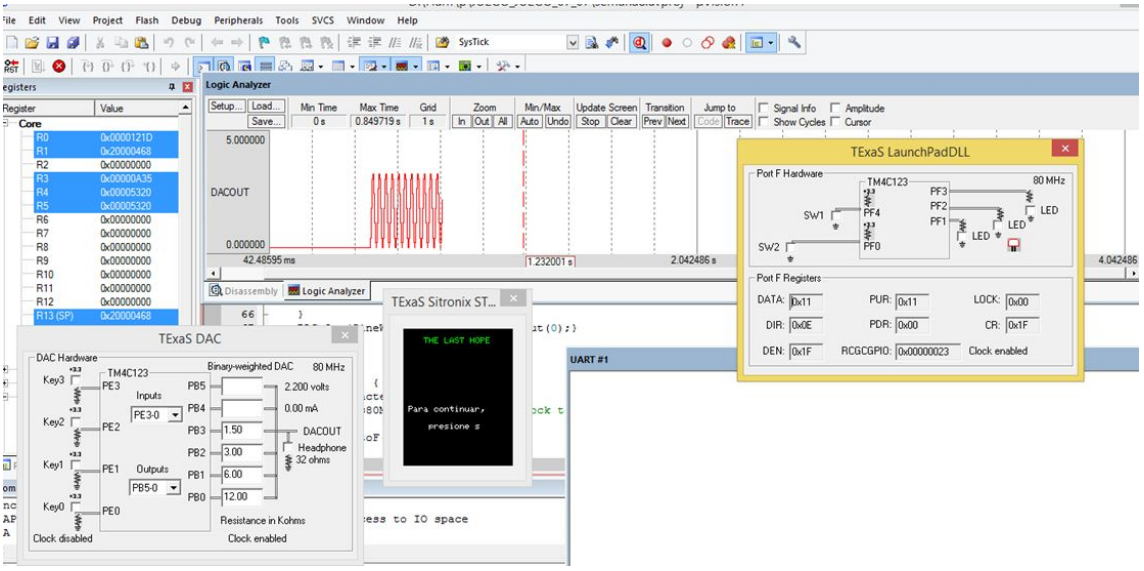
4. Resultados:

De la codificación y simulación llevadas a cabo, se puede observar que el proyecto ha resultado exitoso en cada aspecto que se llegó a plantear

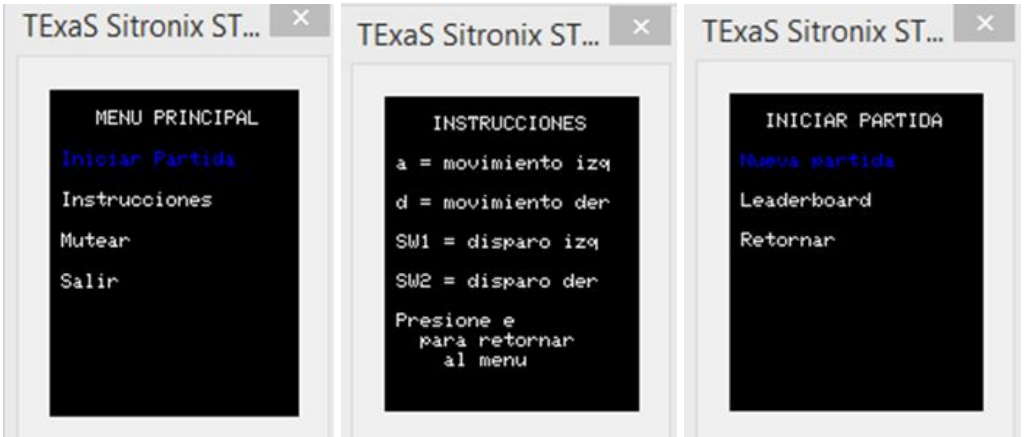
El proyecto cumple con la utilización y correcto uso de los módulos GPIO, SSI, UART, DAC Y SYSTICK (uso de interrupciones)

Cada subfunción y librería ha sido correctamente implementada, por lo que el programa cumple con lo esperado

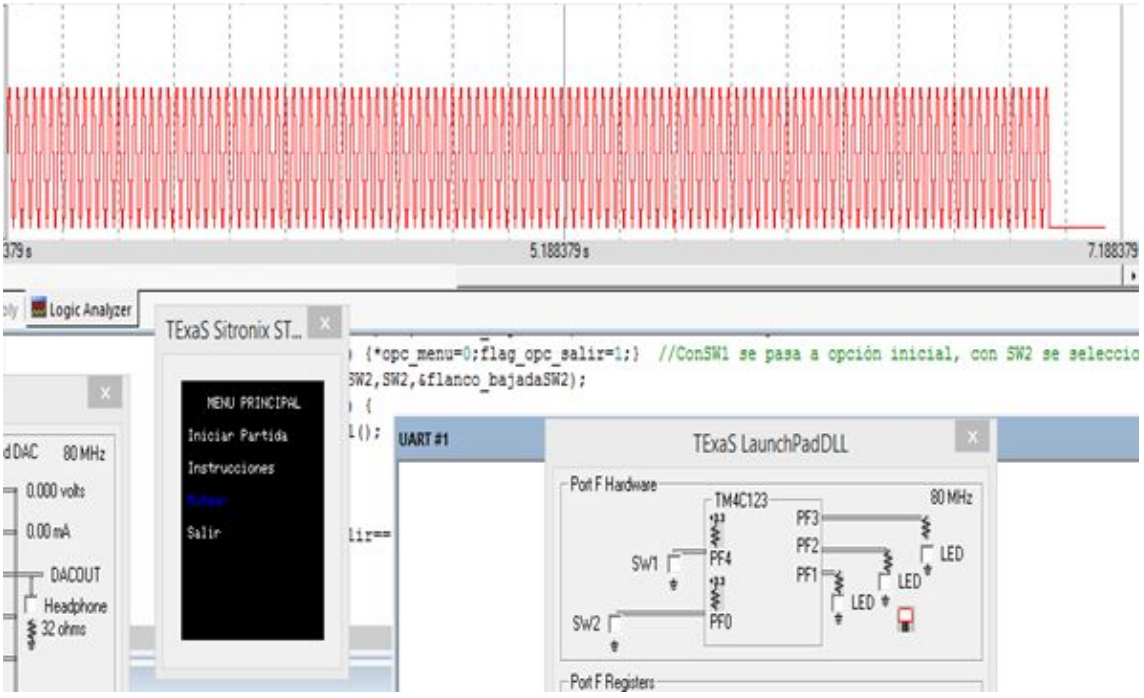
Se logró, mediante el uso de la técnica “modulación”, desarrollar un programa que resulta de fácil comprensión y que no es agotador para alguien que quiera entenderlo, ya sea para recrearlo o para modificarlo (Ejm. El archivo main.c ocupa menos de 100 líneas de código).



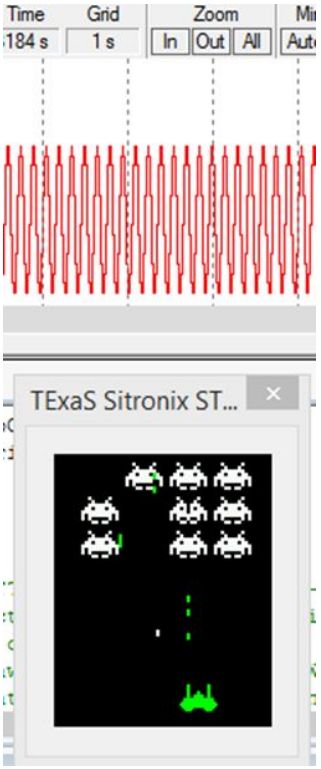
En esta imagen se puede observar el correcto funcionamiento de todos los módulos implementados en conjunto.



En estas imágenes se puede observar el correcto funcionamiento de los menús y de la presentación de las instrucciones.

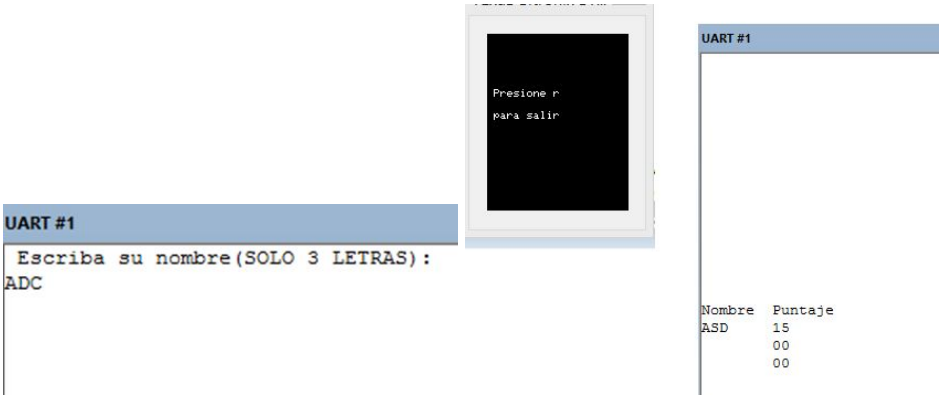


En esta imagen se puede observar el correcto funcionamiento de la función mutear



En esta imagen se observa que el DAC funciona siempre

En la primera imagen se observa la correcta detección del impacto de los disparos y la capacidad de desplazarse por parte del usuario. Adicionalmente, se muestran los mensajes en caso de victoria o derrota en cada nivel



En estas imágenes se demuestra el correcto funcionamiento de la actualización del leaderboard, así como el correcto funcionamiento de la opción leaderboard en el menú Iniciar Partida

Finalmente, para observar el funcionamiento se ofrecen los siguientes links:
https://youtu.be/zKV7WxCSOMg
https://youtu.be/B8Zsw2wV6Lk

5. Conclusiones:

Del proyecto desarrollado, se concluye:

Resulta muy importante la planificación previa para desarrollar un proyecto de sistemas embebidos, ya que esto permite fijar objetivos claros y metas realistas y determinar los módulos a usar para desarrollar el proyecto. La elaboración del diagrama esquemático, de módulos y, sobre todo, de los diagramas de flujo es clave para tener una guía de referencia que servirá para asegurar la realización y culminación exitosa del proyecto.

Respecto a la programación, se concluye que la programación modular es efectiva para microcontroladores, ya que hace fácilmente ubicable los módulos, asimismo el diseño jerárquico permite poder revisar el código de manera sencilla y realizar modificaciones sin perderse en este. Esto último incluso beneficia a aquellos que ven por primera vez el código y quieren interactuar con él

Se logró desarrollar un videojuego funcional que cumple con los requerimientos planteados y se podría implementar como un videojuego real

Pese a las limitaciones encontradas, se pudo concluir de forma exitosa el proyecto, ya que se pudo dar solución a todos los problemas o bien se pudo proponer alternativas que dirigieron a una culminación exitosa al proyecto

6. Observaciones y recomendaciones:

Se debe tener en mente que el videojuego está operativo en cuanto a simulación respecta, por lo que a continuación se explicará la manera de "jugar" desde el simulador. En los menús, pulsar SW1 sirve para desplazarse hacia la siguiente opción y pulsar SW2 para seleccionar la opción. La verificación del correcto funcionamiento de la salida de sonido se realiza haciendo uso del Logic Analizer (señal DACOUT). El videojuego se muestra en la pantalla SITRONIX ST7735. Durante el juego, el terminal serial se usa para desplazarse (a=mov izq y d=mov der) y se debe pulsar los SWs para disparar (SW1 cañón izq, SW2 cañón der). Adicionalmente, el terminal serial se usa para registrar el nombre del jugador y mostrar la tabla de puntajes.
Una limitación del proyecto es que no se puede simular el hecho de guardar el leaderboard "para siempre", la única manera conocida de atacar este problema sería utilizando una tarjeta física, por lo que esta limitación corresponde netamente al contexto virtual en que nos encontramos.

A continuación, se presentan recomendaciones:

En el código:

Para la simulación en KEIL uVision, procurar que sea la única aplicación abierta y tener el internet apagado (en nuestra experiencia, mejora la simulación)

En el supuesto de ser implementado:

Se sugiere que, de querer implementarse, se verifique que los componentes son compatibles en cuanto a corriente y tensión de trabajo para evitar que los componentes se dañen.
Se sugiere tener en cuenta el delay por rebote mecánico en el caso de la presión de SW1 y SW2 si se va implementar de manera física, asimismo, es recomendable disminuir la frecuencia de movimiento de los personajes en el sitronix, ya que lo que hace que se vea lento el juego no es la lógica en sí, sino el tiempo de simulación (1seg simulación > 3 seg vida real).
Se sugiere que, de querer implementarse, se verifique que las conexiones hacia la tarjeta Tiva C estén realizadas de una manera correcta, ya que de lo contrario, no se asegura el correcto funcionamiento del videojuego e incluso se podría dañar la tarjeta.

7. Bibliografía

Valvano, Jonathan W.
Embedded systems : real-time interfacing to ARM ¿Cortex? - M microcontrollers
https://pucp.ent.sirsi.net/client/es_ES/campus/search/detailnonmodal/ent:$002f$002fSD_ILS$002f0$002fSD_ILS:551535/one

Texas Instruments
Tiva™ TM4C123GH6PM Microcontroller DATA SHEET

.Revista Épsilon
Suárez Mora, D. R., Aparicio Gómez, A., Gallego Ibáñez, Y. K . y Ramírez Salcedo, J. C. (2015).
Integración de los sistemas embebidos Raspberry Pi y Arduino para el manejo de un brazo robótico mediante una aplicación Android. Épsilon,(25)

Valvano, Jonathan W.
Consulta:28/06/2020
https://youtu.be/JYKvC6p8yl0