PROYECTO PACMAN-32

Sistemas Embebidos

3 de Diciembre 2022

Berdeja Ruiz Rolando Fabricio
Ingeniería Macatrónica
Universidad Católica Boliviana
rolando.berdeja@ucb.edu.bo

Santelices Roncal Cristian Bryan
Ingeniería Macatrónica
Universidad Católica Boliviana
cristian.santelices@ucb.edu.bo

Diego Mauricio Pinto Mendoza
Ingeniería Macatrónica
Universidad Católica Boliviana
diego.pinto.m@ucb.edu.bo

***Resumen*—En el presente informe se vera la implementacion del juego PACMAN en una pantalla OLED utilizando un microcontrolador y pulsadores, se podrán observar los objetivos y también se contemplaran los limites y alcances del proyecto elegido. El cual basado en ciertos criterios investigados, se planteara el desarrollar un juego de PACMAN con el kit que se entrego en la asignatura, que consta de un nucleo-L432KC y una panatalla OLED de 128x64. El proyecto se realizo con todos estos materiales, cumpliendo con los objetivos planteados y teniendo la solida base de una investigacion, concluyendo con la muestra de la pantalla OLED el juego de Pacman, y el resto del circuito fisico.**

***Index Terms*—Proyecto, STM32, Codigo C, nucleo, oled.**

I. Objetivos

Desarrollar e implementar juego de Pacman en un nucleo-L432KC con pantalla OLED 128x64.

- Desarrollar el código deseado para cargarlo al nucleo.
- Implementar jugabilidad con 4 pulsadores.
- Diseñar el código para la pantalla OLED que entre en el tamaño de 128x64.
- Utilizar y configurar los pines deseados.
- Hacer que los códigos del nucleo y la pantalla funcionen.
- Analizar posibles mejoras y cambios.

II. Limites y alcances

En los limites se busca:

- Modificación de los mapas por tamaño del display de 128x64.
- Crear nueva jugabilidad con el puntaje y niveles como el juego real.
- Capacidad de niveles totales.

En los alcances se busca:

- Desarrollar una mini consola de Pac-Man.
- Implementar códigos de entrada y salida en tiempo real sobre la pantalla.
- Aplicación y entendimiento del manejo del display con el STM32.

III. Introducción

III-A. Pacman

El proyecto que se tiene pensado realizar un juego clásico. Pac-Man es un videojuego de acción en el que el jugador controla al personaje homónimo en un laberinto cerrado.El videojuego es ya una leyenda y ha sabido adaptarse a los nuevos tiempos. Se han publicado adaptaciones para todas las consolas y formatos, e incluso se llegó a crear un personaje femenino inspirado en el Pac-Man original, Ms. Pac-Man .

El protagonista del videojuego Pac-Man es un círculo amarillo al que le falta un sector, por lo que parece tener boca. Aparece en laberintos donde debe comer puntos pequeños, puntos mayores y otros premios con forma de frutas y otros objetos. El objetivo del juego es comer todos los puntos colocados en el laberinto y evitar los fantasmas de cuatro colores: Blinky (rojo), Pinky (rosa), Inky (cian) y Clyde (naranja), que persiguen a Pac-Man. Sin embargo, cuatro fantasmas o monstruos, recorren el laberinto para intentar capturar a Pac-Man.

Cuando Pac-Man se ha comido todos los puntos, el jugador entra en un nuevo nivel. Los niveles se indican con símbolos de fruta en la parte inferior de la pantalla. Entre nivel y nivel hay escenas cortas en las que Pac y Blink se meten en situaciones divertidas.

Figura 1. Juego clasico de PACMAN.

Cuando Pac-Man es atrapado por un fantasma, pierde una vida; cuando se pierden todas las vidas, el juego termina. Cada uno de los cuatro fantasmas tiene su propia inteligencia artificial (IA) o ”personalidad”: Blinky sigue directamente a Pac-Man; Pinky e Inky intentan ponerse delante de Pac-Man, a menudo arrinconándolo; Ryder alterna entre seguir a Pac-Man y escapar.

III-B. Núcleo-L432KC

Para este proyecto necesitamos un microcontrolador, en este caso utilizaremos el núcleo-L432KC. NUCLEO-L432KC es una placa de desarrollo Nucleo-32 que incluye el microcontrolador de potencia ultra baja STM32L432KCU6. Permite crear prototipos y desarrollar ideas utilizando la MCU STM32, con posibilidad de elegir entre diferentes combinaciones de rendimiento, consumo de potencia.

Las placas STM32 Nucleo-32 ofrecen una forma asequible y flexible para que los usuarios prueben nuevos conceptos y construyan prototipos con los microcontroladores STM32, elige entre varias combinaciones de rendimiento, consumo de energía y características. Las placas STM32 Nucleo-32 no requiere ningún hardware adicional ya que integra el depurador / programador ST-LINK. La placa STM32 Nucleo se suministra con la biblioteca HAL completa de software de STM32 junto con diversos ejemplos de software comercial, así como acceso directo a recursos online de mbed. Tiene un cortex ARM Cortex M4 con interfaz USB.

Características del Nucleo L432KC de STMicroelectronics: el microcontrolador STM32 tiene 32 pines, el oscilador es de cristal de 24 MHz, tiene LED de usuario 1x, tiene botón de reinicio, tiene conector de expansión Arduino Nano V3, conector USB Micro-AB para ST-LINK, ST-LINK integrado con capacidad de remuneración USB: almacenamiento masivo, puerto COM virtual y puerto de depuración, tiene bibliotecas completas de software libre.

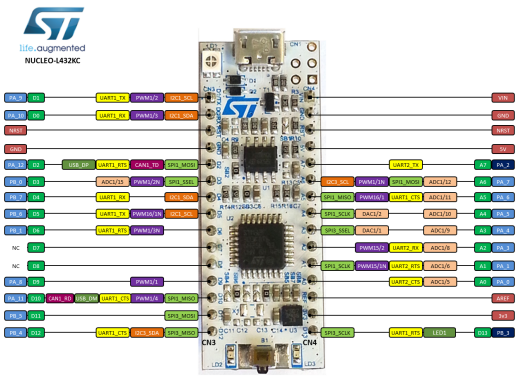

Figura 2. El microcontrolador y el pinout.

III-C. OLED de 128x64

También haremos uso del display o pantalla que se entrego en el kit, siendo este el OLED 128x64 0,96”. Las pantallas OLED se destacan por su gran contraste, mínimo consumo de energía y buena calidad de imagen. El display Oled 0.96Ï2C SSD1306 posee una resolución de 128*64 píxeles, permitiendo controlar cada píxel individualmente y mostrar tanto texto como gráficos. Además por ser de tipo OLED no necesita de retroiluminación (Backlight) como los LCD, lo que hace que su consumo de energía sea mucho menor y aumenta su contraste.

Las pantallas OLED se caracterizan por su gran contraste, bajo consumo 0.04w, resolución de 128*64 pixeles, permitiendo controlar cada pixel ideal para texto o gráficos. A diferencia de las pantallas LCD las pantallas OLED no necesitan retroiluminación, lo que significa mejor contraste y claridad.

El display posee interfaz de comunicación de tipo I2C. Diseñado para trabajar a 5V directamente gracias a su regulador de voltaje en placa y puede trabajar con sistemas de 3.3V o 5V sin necesidad de conversores. Debemos tener en cuenta que los pines I2C son diferentes para cada modelo de Arduino, por lo que debemos revisar el Pinout de nuestro Arduino para saber cuales son sus pines I2C, por ejemplo en Arduino Uno son los pines A4(SDA: data) y A5(SCL: clock).

Para manejar la pantalla Oled es necesario utilizar un microcontrolador con al menos 1K de RAM, este espacio cumple la función de buffer para el display. El driver de la pantalla es el SSD1306, con una librería lista para usarse en Arduino. La librería permite mostrar texto, mapas de bits, píxeles, rectángulos, círculos y líneas. A pesar de usar 1K de RAM, el funcionamiento es muy rápido y el código es fácilmente portable a distintas plataformas de microcontroladores.(naylampmechatronics).

Figura 3. OLED128x64 proporcionado en el kit.

El software que lo utilizamos para realizar los procedimientos es el STM32CubeIDE, dado que nos permite escoger nuestro controlador y configurar los pines deseados para así poder transmitir a la placa del microcontrolador el código.

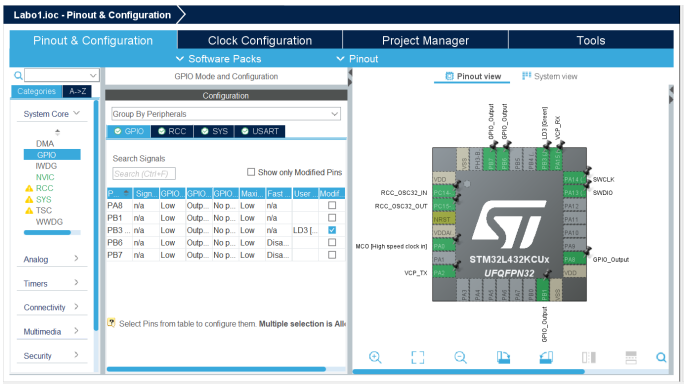

Figura 4. La configuración de los I/O en el IDE.

IV. Marco Teórico

IV-A. Historia de Pac-Man

Lanzado el 21 de mayo de 1980, Pac-Man fue diseñado por Toru Iwatani, un hombre japonés que trabajaba en el desarrollador Namco en ese momento, quien imaginó al personaje mientras comía pizza y lo vio sin una ”tajada”. Ese sería el origen de la imagen inequívoca del héroe amarillo.

Figura 5. Imagen de una maquina Arcade.

Toru está alterando el diseño original de su Puck-Man para ampliar la jugabilidad tanto como sea posible. Un juego que se vuelve aburrido después de unos minutos no sirve para nada. Lo primero que hace es aumentar los puntos obtenidos por Puck-Man. 10 por punto comestible, 200 por bola de energía. 200 si come fantasmas, 400 si atrapa dos, 800 si tiene tres, 1600 si llena los cuatro antes de que se acabe el tiempo.

La velocidad de movimiento es más lenta en el modo de pánico y el tiempo de caza se reduce gradualmente. La fruta aparecerá varias veces dependiendo del momento determinado por la oportunidad equivocada.

También hizo cambios en el laberinto hasta que se decidió por una versión que le gustaba. Para facilitar el escape, agregó un túnel; Puck-Man entra y sale, pero si los fantasmas lo siguen, disminuyen la velocidad.

Luego vino el mayor éxito: cada fantasma tiene su propia personalidad, expresada en la forma en que trata de cazar monstruos. Tienen nombres, apodos y colores: Oikake (Akabei) es rojo y agresivo, Machibuse (Pinky) es rosa y bueno para las emboscadas, Kimagure (Aosuke) es azul porque es impredecible, Otoboke (Guzuta) es un naranja que distrae, siempre ve al Otra manera.

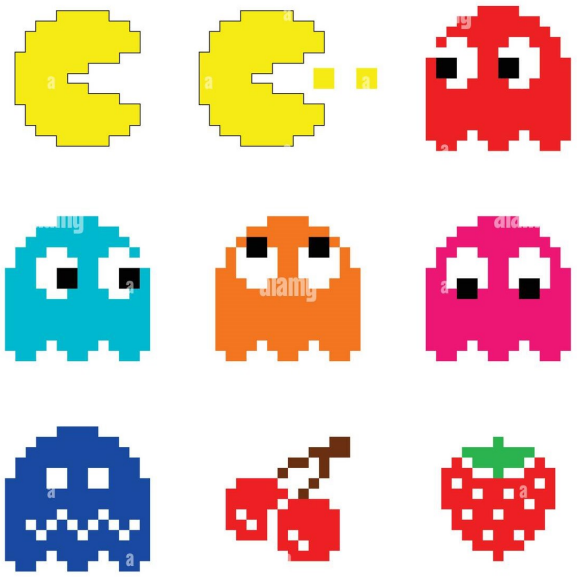

Figura 6. Iconos de Pac-Man y los fantasmas.

En la lógica de programación, el laberinto se divide en mosaicos invisibles para permitir el control del comportamiento. Los fantasmas siempre van en línea recta hasta que llegan a una bifurcación en la que deciden si girar y hacia dónde. Se les prohibió retroceder a menos que entraran en pánico. Si se dispersan, también invierten la dirección, lo que sucede varias veces por nivel.

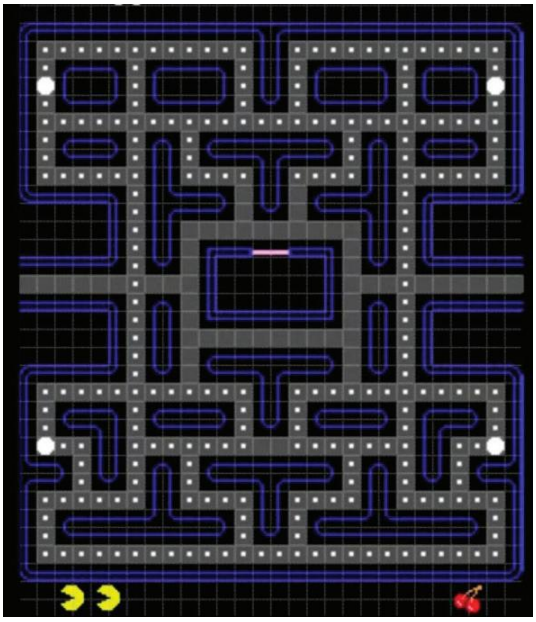

Figura 7. Laberinto del videojuego

Pac-man fue originalmente producido gracias a algoritmos bastante simples pero que hacían difícil predecir cómo se moverían los fantasmas, al igual que muchos juegos de arcade de los 80, se creó usando lenguaje ensamblador y con la evolución de la tecnología avanzó hacia el lenguaje C y C++.

IV-B. Pulsadores

Un interruptor o botón eléctrico es un elemento eléctrico que permite o impide el paso de corriente cuando se acciona o presiona. El botón solo se enciende o apaga cuando el usuario lo presiona y lo mantiene presionado. Cuando lo sueltas, vuelve a su posición original.

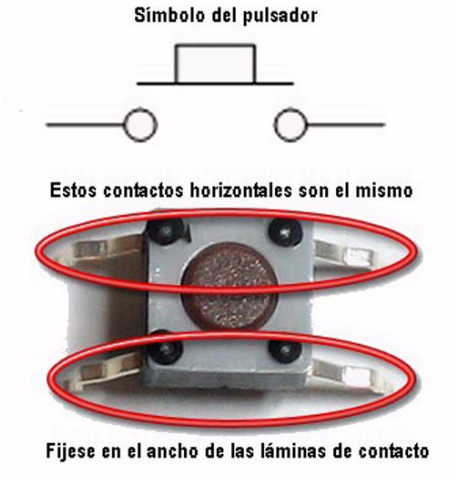

Figura 8. Imagen de un pulsador.

Para que un botón funcione, debe tener un resorte o resorte que lo devuelva a su posición anterior cuando se presiona. El ejemplo más evidente es un botón para activar el timbre de la casa.

Lo presionas y deja fluir la electricidad activando la campana, pero una vez que lo sueltas, vuelve a su posición inicial, evitando que la campana suene. El paso o cierre de corriente se logra a través de contactos eléctricos, también conocidos como ”terminales”, generalmente de cobre.

Cada contacto eléctrico del botón tiene 2 posiciones, abierto y cerrado. Cerrado: Los dos terminales están juntos y el botón permite que fluya la corriente. Circuito abierto: 2 terminales están separados, el botón corta o no deja pasar la corriente.

IV-C. Lenguaje C

El desarrollo inicial de C tuvo lugar en AT-y-T Bell Laboratories de 1969 a 1973; Según Dennis Ritchie, el período más creativo fue en 1972. Fue nombrado Ç”porque muchas de sus características fueron tomadas de un lenguaje más antiguo llamado ”B”.

Hay muchas leyendas sobre el origen de C y el sistema operativo estrechamente relacionado con él, Unix. Algunos de ellos son:

El desarrollo de C fue el resultado del deseo de los programadores de jugar con Space Travel. Estaban jugando en el mainframe de su empresa, pero debido a la baja potencia de procesamiento y la necesidad de admitir 100 usuarios, Ken Thompson y Dennis Ritchie no pudieron controlar la nave para evitar la colisión de un asteroide, por lo que decidieron mover el juego al no utilizado. escritorio PDP-7; pero esta máquina no tenía un sistema operativo, por lo que decidieron escribir uno. Al final decidieron portar el sistema operativo PDP-11 que tenían en su oficina, pero era muy caro, ya que todo el código estaba escrito en lenguaje ensamblador. Entonces decidieron usar un lenguaje portátil y de alto nivel para que el sistema operativo pudiera ser portado fácilmente de una computadora a otra, consideraron usar B pero carecía de la funcionalidad necesaria para aprovechar algunas de las características avanzadas del PDP-11 , por lo que comenzaron a crear un nuevo lenguaje: C.

Figura 9. Icono del lenguaje C.

La razón fundamental para usar las primeras computadoras para desarrollar Unix era crear un sistema que pudiera automatizar la presentación de patentes. La versión original de Unix se desarrolló en lenguaje ensamblador. Posteriormente se desarrolló el lenguaje C para poder reescribir el sistema operativo. Para 1973, el lenguaje C se había vuelto tan poderoso que la mayor parte del kernel de Unix, originalmente escrito en lenguaje ensamblador PDP-11, fue reescrito en C.1 Fue uno de los primeros sistemas operativos del kernel. ensamblador. idioma. (Los casos anteriores fueron el sistema Multics, escrito en PL/I, y el programa de control principal para Burroughs B5000, escrito en ALGOL en 1961).

En términos de aplicabilidad, fue creado principalmente para la programación competente en sistemas Unix. También se utiliza para desarrollar otros sistemas operativos como Windows o GNU/Linux. De manera similar, para aplicaciones de escritorio como GIMP, cuyo lenguaje de programación principal es C.

Del mismo modo, es muy utilizado en aplicaciones científicas (para experimentos computacionales, física, química, matemáticas, entre otras, una parte de las cuales se denomina modelado y simulación), industriales (robótica, cibernética, sistemas de información y bases de datos para la industria petrolera y petroquímica, prevalece también todo lo relacionado con la simulación de maquinaria de producción), la simulación de vuelo (esta es la más delicada, porque se utilizan demasiados recursos materiales y de software para desarrollar aplicaciones de simulación de vuelo de aeronaves de la vida real). Por lo que se aplica en varios campos que son desconocidos para la mayoría de los usuarios novatos.

Las computadoras de finales de la década de 1990 eran más potentes que las máquinas para las que C se desarrolló originalmente. Los programas escritos en lenguajes de entrada dinámicos y fáciles de codificar (Ruby, Python, Perl...) solían ser demasiado lentos pero lo suficientemente rápidos como para reemplazar el C que se usaba. Aun así, todavía puedes encontrar código C en el increíble desarrollo de animación, modelado 3D y escenas en películas y otras aplicaciones multimedia.

C es un lenguaje de programación popular para sistemas integrados [cita requerida] El código ligero generado por compiladores de C, combinado con la capacidad de acceder a capas de software cercanas al hardware, es la razón de su popularidad en estas aplicaciones.

Una característica que C muestra una usabilidad particularmente valiosa en sistemas integrados es la manipulación de bits. El sistema contiene registros mapeados en memoria (MMR) a través de los cuales se configuran los dispositivos. Estos registros mezclan varias configuraciones en la misma dirección de memoria, aunque en bits diferentes. Con C es fácil cambiar uno de estos bits sin cambiar el resto.

Otras características de C que se consideran desventajas en la programación de PC, como la falta de controles de memoria automáticos, se convierten en una ventaja cuando los sistemas integrados necesitan código pequeño y optimizado. Este es el caso de los sistemas basados en microcontroladores de bajo consumo como el Intel 8051 o muchos sistemas ARM.

IV-D. OLED

OLED es un tipo de pantalla relativamente nuevo para televisores, teléfonos inteligentes y computadoras portátiles. Inventado en 1987, OLED ya es una de las dos tecnologías de visualización líderes en la industria. OLED significa diodo emisor de luz orgánico. Esta tecnología de visualización utiliza compuestos orgánicos (carbono) que emiten luz cuando pasa una corriente eléctrica a través de ellos.

La tecnología OLED es conocida por su alta calidad de imagen y precio premium. El principal competidor de OLED es la pantalla de cristal líquido barata, o LCD para abreviar. Sin embargo, el alto contraste de color y los amplios ángulos de visión de OLED lo hacen ideal para la electrónica de alta gama. Los paneles OLED también son más ligeros y flexibles que las pantallas LCD.

También es muy duradero, con una vida útil de aproximadamente 22 años cuando se usa 6 horas al día. Hay muchos tipos diferentes de pantallas OLED. Los tipos más comunes son OLED de matriz pasiva (PMOLED) y OLED de matriz activa (AMOLED). La principal diferencia entre estos dos es que los OLED de matriz activa tienen transistores de película delgada (TFT), mientras que los OLED de matriz pasiva no los tienen.

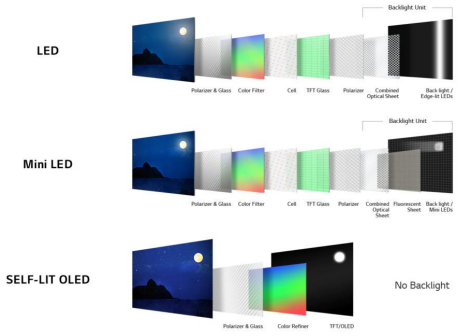

Figura 10. Diferencias entre pantallas.

Los transistores de película delgada sirven como semiconductores en las pantallas OLED. AMOLED es ideal para pantallas grandes, ya que TFT mejora la eficiencia energética. Otro tipo de OLED es el OLED fosforescente o PMOLED. Los OLED fosforescentes son más eficientes que los OLED fluorescentes. Los OLED fosforescentes de alta eficiencia se degradan más lentamente al desvanecerse en lugar de desvanecerse instantáneamente.

También hay otro tipo de OLED, a menudo llamado PLED, que utiliza polímeros emisores de luz en lugar de moléculas orgánicas. La mayoría de los OLED usan sustratos de vidrio, pero los dispositivos OLED también se pueden fabricar con sustratos de plástico. Esto es ideal para OLED flexibles ya que es más delgado.

OLED significa diodo emisor de luz orgánico. La materia orgánica en este caso se refiere a su definición química, no a los alimentos. Las moléculas están compuestas por cadenas o anillos de carbono y otros elementos. El acrónimo restante de LED significa Light Emitting Diode. Se refiere a cualquier sistema de dos electrodos que emite luz en presencia de una corriente eléctrica.

Los electrodos tienen carga opuesta. El electrodo cargado positivamente se llama cátodo y el electrodo cargado negativamente se llama ánodo. Se dispone una capa orgánica entre los electrodos. Entonces, cuando los electrones se mueven del ánodo al cátodo y crean una corriente eléctrica, atraviesan el material orgánico y emiten luz de colores. Las pantallas OLED usan materiales orgánicos amarillos y azules, y otros colores se crean usando muchos filtros de color.

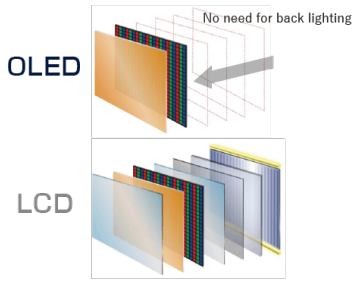

Figura 11. Esquema de una pantalla OLED.

En 1950, el químico francés André Bernanost descubrió las propiedades electroluminiscentes de los compuestos orgánicos. Pero este descubrimiento no se puso en práctica hasta 1987, cuando los científicos Ching Tang y Steven Van Slyke inventaron las últimas pantallas OLED. Pasó otra década antes de que OLED se usara comercialmente. Pioneer comenzó a usarlo en pantallas de automóviles y un año después en televisores. Finalmente, en 2000, Sanyo, LG, NEC y Samsung comenzaron la producción en masa de OLED.

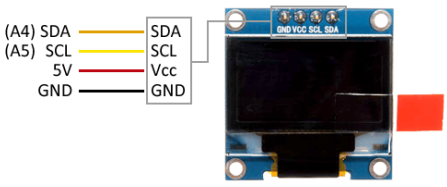

Figura 12. Conexion oled.

V. Marco Practico

V-A. Desarrollo de Hardware

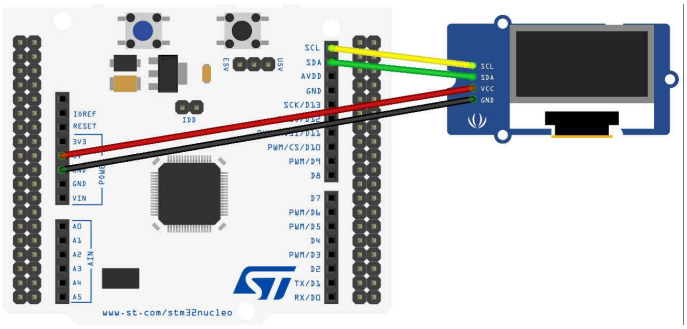

Figura 13. Esquema de conexión referencial STM y pantalla OLED

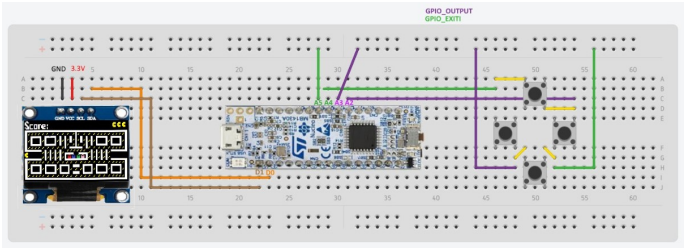

Figura 14. Las conexiones en el protoboard del microcontrolador, pulsadores y la pantalla oled.

En el desarrollo del proyecto se necesito 4 pulsadores para el movimiento del pacman, la pantalla OLED para visualizar el juego como tal y el microcontrolador STM32 para hacer toda la programacion y lo requerido para que el juego funcione, con estos componentes se logro el armado del pequeño emulador para que asi mostrar en la pantalla el juego de Pac man.

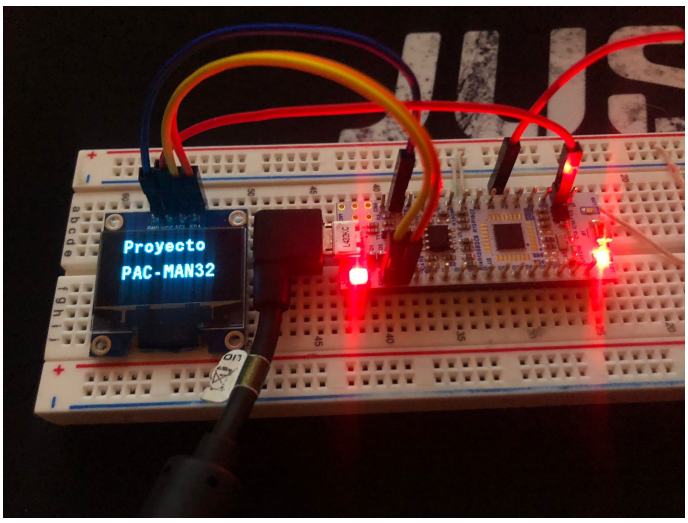

Figura 15. Circuito armado, el STM32 con todos sus pines configurasos y las conexiones a la pantalla oled, Proyecto PAC MAN32

En este circuito se logró una implementación de los botones como un sistema de matrices, donde todos los botones están conectados en serie entre sí, logrando obtener sus pulsaciones a través de la configuración de puertos. Además de buscar emular los botones clásicos del juego como una cruz según su sentido.

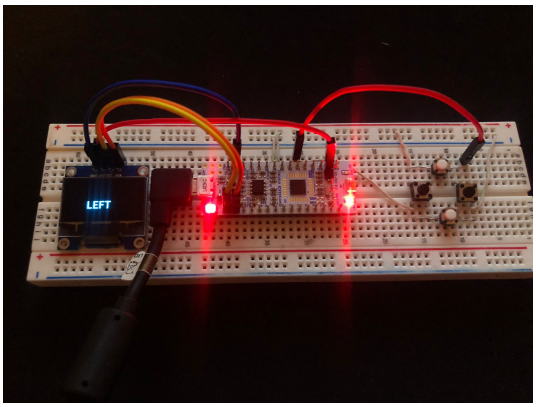

Figura 16. Funcionamiento botón izquierdo.

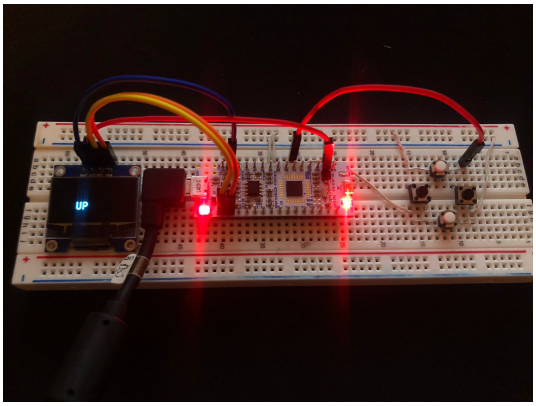

Figura 17. Funcionamiento botón arriba.

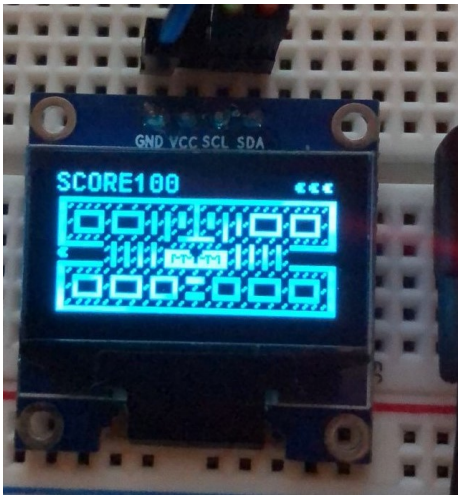

Figura 18. Visualisacion del juego en la pantalla.

V-B. Desarrollo de software

Primeramente se realizaron todas las configuraciones necesarias en el IDE, como los times y demas, para poder realizar el código necesario.

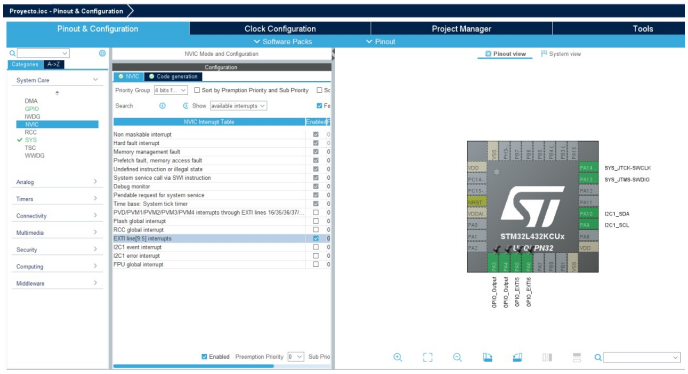

Figura 19. Configuración de pines en el IDE

En el proyecto se logro implementar un código capaz de detectar las pulsaciones e imprimir en la pantalla el comando realizado a través del pulsador. Así logrando imprimir funciones y observar que si se están ejecutando las pulsaciones debidamente. Luego se implemento la lógica del juego, es decir la aplicación de niveles, score y fantasmas como se muestra en la pantalla oled.

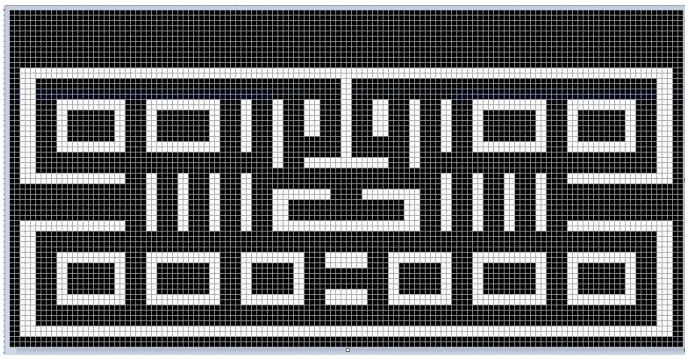

Figura 20. Se muestra el diseño inicial del mapa de pacman.

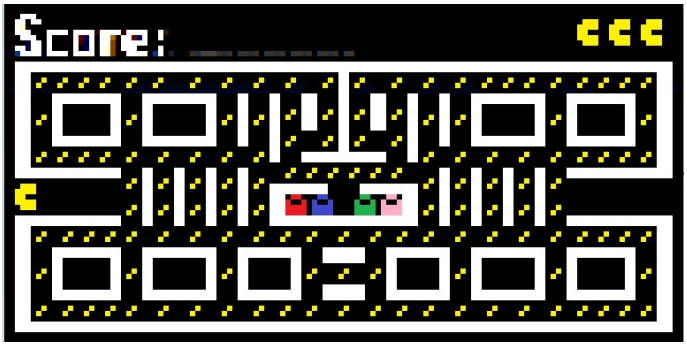

Figura 21. Se muestra el diseño del mapa con la comida, fantasmas y el pacman.

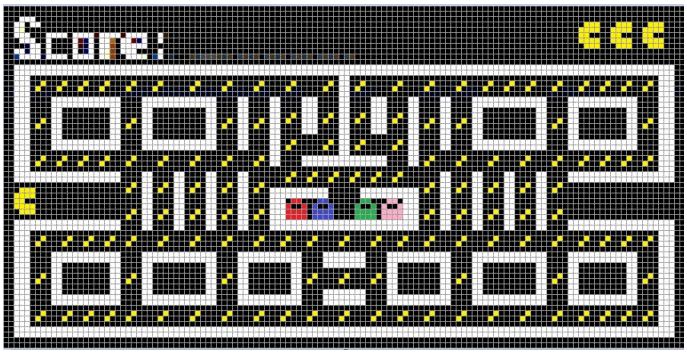

Figura 22. Se muestra ya el mapa en escala para que entre en la pantalla oled segun sus pixecel.

V-C. Código

A continuación se mostrara todo el código y los pixecel para la pantalla oled desarrollado para el juego.

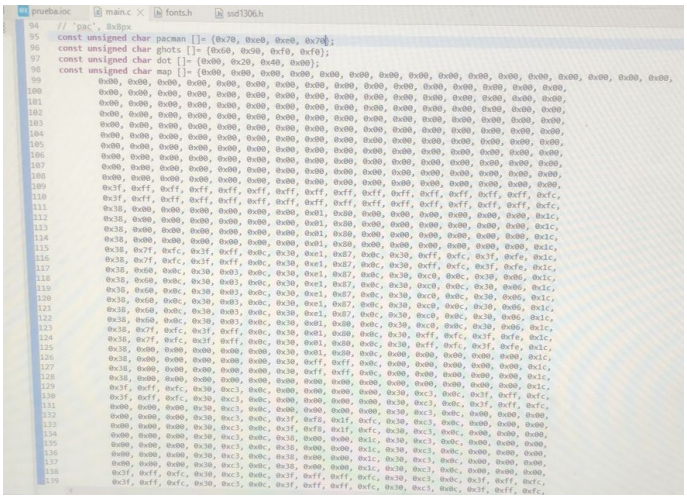

Figura 23. Se muestra una parte de los pixeles donde gracias a esto podemos ver el mapa, los fantasmas, la comida y el pacman, todo dimensionado y en escala.

```
/* USER CODE END Header */
/* Includes ------------------------------------------
#include "main.h"

/* Private includes ----------------------------------
/* USER CODE BEGIN Includes */
#include "fonts.h"
#include "ssd1306.h"
/* USER CODE END Includes */

/* Private typedef -----------------------------------
```

Figura 24. Parte 1 del código.

```
/* USER CODE BEGIN PV */
GPIO_InitTypeDef GPIO_InitStructPrivate = {0};
uint32_t previousMillis = 0;
uint32_t currentMillis = 0;
uint8_t keyPressed = 0;
/* USER CODE END PV */

/* Private function prototypes -----------------
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_I2C1_Init(void);
/* USER CODE BEGIN PFP */
```

Figura 25. Parte 2 del código.

```
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------

  /* Reset of all peripherals, Initializes the Flash i
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_I2C1_Init();
  /* USER CODE BEGIN 2 */
  SSD1306_Init();
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, 1);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, 1);
  SSD1306_GotoXY (0,0);
  SSD1306_Puts ("Proyecto", &Font_11x18, 1);
  SSD1306_GotoXY (0, 30);
  SSD1306_Puts ("PAC-MAN32", &Font_11x18, 1);
  SSD1306_UpdateScreen();
  HAL_Delay (1000);
  SSD1306_ScrollRight(0,7);
  HAL_Delay(3000);
  SSD1306_ScrollLeft(0,7);
  HAL_Delay(3000);
  SSD1306_Stopscroll();
  SSD1306_Clear();
  SSD1306_GotoXY (35,0);
  SSD1306_Puts ("PLAY", &Font_11x18, 1);
  SSD1306_UpdateScreen();
  /* USER CODE END 2 */
```

Figura 26. Parte 3 del código.

```
/* USER CODE BEGIN WHILE */
while (1)
{

  /* USER CODE END WHILE */
    if(keyPressed==85){
        SSD1306_Clear();
        SSD1306_GotoXY (35,35);
        SSD1306_Puts ("UP", &Font_11x18, 1);
        SSD1306_UpdateScreen();
        HAL_Delay (1000);
    }
    else if(keyPressed==82){
        SSD1306_Clear();
        SSD1306_GotoXY (35,35);
        SSD1306_Puts ("RIGHT", &Font_11x18, 1);
        SSD1306_UpdateScreen();
        HAL_Delay (1000);
    }
    else if(keyPressed==76){
        SSD1306_Clear();
        SSD1306_GotoXY (35,35);
        SSD1306_Puts ("LEFT", &Font_11x18, 1);
        SSD1306_UpdateScreen();
        HAL_Delay (1000);
    }
    else if(keyPressed==68){
        SSD1306_Clear();
        SSD1306_GotoXY (35,35);
        SSD1306_Puts ("DOWN", &Font_11x18, 1);
        SSD1306_UpdateScreen();
        HAL_Delay (1000);
    }
```

Figura 27. Parte 4 del código.

```
/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  currentMillis = HAL_GetTick();
  if (currentMillis - previousMillis > 10) {
    /*Configure GPIO pins : PB6 PB5 to GPIO_INPUT*/
    GPIO_InitStructPrivate.Pin = GPIO_PIN_5|GPIO_PIN_6;
    GPIO_InitStructPrivate.Mode = GPIO_MODE_INPUT;
    GPIO_InitStructPrivate.Pull = GPIO_NOPULL;
    GPIO_InitStructPrivate.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStructPrivate);

    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, 1);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, 0);
    if(GPIO_Pin == GPIO_PIN_5 && HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_5))
    {
      keyPressed = 85; //ASCII value of Up
    }
    else if(GPIO_Pin == GPIO_PIN_6 && HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_6)
    {
      keyPressed = 82; //ASCII value of Right
    }

    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, 0);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, 1);
    if(GPIO_Pin == GPIO_PIN_5 && HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_5))
    {
      keyPressed = 76; //ASCII value of Left
    }
    else if(GPIO_Pin == GPIO_PIN_6 && HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_6)
    {
      keyPressed = 68; //ASCII value of Down
    }

    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, 1);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, 1);

    /*Configure GPIO pins : PB6 PB5 back to EXTI*/
    GPIO_InitStructPrivate.Mode = GPIO_MODE_IT_RISING;
    GPIO_InitStructPrivate.Pull = GPIO_PULLDOWN;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStructPrivate);
    previousMillis = currentMillis;
  }
}
/* USER CODE END 4 */
```

Figura 28. Parte 5 del código.

VI. Conclusiones

No pudimos concluir con el proyecto ya que tuvimos algunos conflictos con el tema de los movimientos, los pulsadores si son reconocidos pero no generan moviemiento, respecto desde el inicio, si se pudo implementar codigo para que la pantalla oled te muestre el mapa, los fantasmas, la comida, y el pacman, toda la base esta, pero al momento de codificar los botones, no genera ningun movimiento, y muchas veces no generaba peligro ni errores pero tampoco generaba movimiento, en eso tuvimos falla ya que si nos reconocio los botones pero no genero ningun movimiento.

Basicamente aplicamos gran parte de lo aprendido en la materia y el conocimiento esta presente, con el tema de memorias, ALU, RAM, ROM, arquitectura computacional, maquinas de turing, CPU, automatas, las interfaces de comunicacion, UART, SPI, I2P, I2C, y assembly x86.

VII. Anexos

Link de la presentación en Canva