Universidad Nacional Autónoma de México

Facultad de Ingeniería

Laboratorio de Fundamentos de Sistemas Embebidos

Proyecto Final

Brigada 5	
318283195	López Cantarell Diego Emir
318192950	Moreno Chalico Edgar Ulises
318263522	Nava Alberto Vanessa
318086721	Reyes Valderrama Rodrigo Miguel

Grupo : 02

Profesor: Ing. Samuel Gandarilla Pérez

Fecha de entrega: 21 de mayo de 2025

Semestre 2025-2

Índice

Proyecto a presentar: Juego Simón Dice

Planteamiento del problema

En la actualidad, los juegos interactivos son una excelente herramienta educativa y de entretenimiento. Sin embargo, gracias a la evolución constante de la tecnología la mayoría de estos juegos son frente a pantallas de dispositivos móviles y aunque algunos de estos son accesibles para la mayor parte de la gente, algunas desventajas de estos son: la publicidad excesiva, consumo excesivo de batería y recursos, monetización, violencia gráfica, requieren hardware costoso o plataformas complejas para su desarrollo y casi ninguno de estos juegos fomenta la concentración, estimulación de la memoria, mejora de aprendizaje junto con la resiliencia.

Gracias a los nuevos juegos, redes sociales y toda la disponibilidad que podemos tener en nuestros dispositivos móviles, nos hemos dado cuenta de que cada vez aumenta la dificultad para poder concentrarse en una sola cosa por más de 10 minutos, y esto pasa tanto en adultos, niños o adolescentes y esto es debido a que lo que se consume en internet suele ser información fugaz como reels, tik tok, facebook, etc.

> *La Nación (2025) opina que, en primer lugar, la atención es un proceso indispensable para procesar información y memorizar. Sin ella, las distracciones constantes evitan que el cerebro se enfoque y registre recuerdos. Y en la actualidad, uno de los grandes potenciadores de las distracciones son las redes sociales. La UNESCO advirtió en un informe que las redes sociales afectan al bienestar y el aprendizaje de muchos niños y adolescentes. Aunque esto no solo se limita a ellos, dado que los adultos también sufren las consecuencias.*

De acuerdo con esto podemos decir que el problema radica en la falta de juegos interactivos sencillos pero estimulantes que puedan ser utilizados como recurso educativo con el fin de estimular la memoria, la concentración, la mejora de aprendizaje y la resiliencia de las personas. Todo esto teniendo en cuenta que se puedan utilizar recursos económicos para el desarrollo del juego.

Planteamiento de la solución

Como equipo planteamos una solución en particular. La solución propuesta consiste en la creación de un juego interactivo de "Simón Dice" basado en colores, ya que opinamos que es una forma entretenida y divertida de estimular habilidades de memoria, coordinación y concentración, ya sea jugado individualmente o en entre dos personas.

El juego será implementado con recursos sencillos de programar y económicos de obtener. Se plantea que el juego tenga un menú inicial en el cual tendrá dos modos de juego: uno individual, donde el jugador se rete a sí mismo y vea hasta dónde es posible llegar siguiendo las secuencias, y uno en pareja, donde dos jugadores deberán seguir secuencias más complejas y competir para ver quién mantiene mejor la memoria. Al final de cada partida se mostrará el puntaje más alto se mostrará en la pantalla.

Este diseño juego tiene como finalidad la competencia amistosa y la mejora continua para la estimulación de las habilidades, como se mencionó anteriormente.

Funcionamiento del juego:

- Al iniciar el juego, el usuario podrá seleccionar entre jugar de forma individual o en pareja. En el modo individual, la máquina generará las secuencias de LEDs, mientras que en el modo en pareja, el juego mostrará secuencias para ambos jugadores.

- El juego consistirá en una serie de secuencias de colores que los jugadores deben memorizar y reproducir presionando los botones correspondientes. La secuencia aumentará en longitud y complejidad a medida que el juego avanza.

- Si un jugador presiona el botón incorrecto, pierde y el juego termina. Se mostrará en la pantalla el puntaje final y el ganador si se juega en pareja.

- El puntaje más alto de cada partida se guardará en memoria y será visible en el menú principal al inicio de cada nueva partida.

Descripción de la solución

El juego se planea desarrollar en el lenguaje de programación C + +. Todo el funcionamiento del juego, desde el manejo del display, hasta las funciones para el manejo de los controles.

Para poder llevar a cabo el desarrollo del juego vamos a necesitar componentes electrónicos como push buttons, leds de colores, PCF8574 Expansor E/S I2C Módulo I²C, protoboard, resistencias y cable de cobre los cuales nos van a servir de ayuda para armar el cuerpo de los controles de juego. También vamos a utilizar una Raspberry Pi 4,una pantalla SPI TFT para

mostrar la interfaz del juego, para realizar la programación y unión lógica de todos los componentes para poder realizar el juego.

Estructura del programa:

La estructura principal del programa del juego se divide en 3 secciones principales y elementos extra para su funcionamiento. Las secciones son las clases: Display, Simon_dice, controller; mientras que los elementos extra son funciones de utils y defines en common.

Secciones del programa:

- Display: Permite controlar una pantalla TFT ST7789 de 240x240 píxeles, conectada mediante SPI. Incluye configuraciones de pines, parámetros de comunicación y funciones para mostrar elementos gráficos como menús, contadores en juego y la pantalla de "Game Over". También maneja la conversión de colores RGB a formato RGB565 para algunos colores adicionales a los que vienen por defecto. La clase está diseñada para integrarse en un juego con opciones como un jugador, multijugador y récords, facilitando la interfaz gráfica desde la Raspberry Pi.

- Simon_dice: Gestiona la lógica del juego. Por un lado se define la estructura básica de la clase, incluyendo métodos para iniciar el juego, agregar elementos a la secuencia, verificar si la secuencia ingresada por el usuario es correcta, y mantener el nivel actual, por otro lado se implementan los métodos: iniciar juego, generación de una secuencia aleatoria que el jugador debe memorizar y reproducir, además de que cada nivel que se avanza se agrega un nuevo número a la secuencia y la muestra al usuario con pausas temporales. Si el jugador introduce la secuencia correctamente avanzará al siguiente nivel; si falla, el juego termina mostrando el nivel alcanzado. En general esta clase, lleva todo el manejo del juego desde el puntaje, nivel, creación de secuencia, etc.

- Controller: Interactúa con un sistema de entrada/salida mediante comunicación I2C, típicamente. La clase Controller, representa el controlador de un jugador, y contiene métodos para escribir valores en salidas (con los LEDs) y leer valores de entradas (con los push buttons). También haremos uso de la biblioteca wiringPi, inicializando dos dispositivos I2C (uno para entrada y otro para salida), y configurando el archivo descriptor para cada uno. El método Write() envía un valor al dispositivo de salida, mientras que ReadInput() lee el valor del dispositivo de entrada y le aplica la máscara definida para omitir los bits que no requerimos, permitiendo identificar qué botones están siendo presionados. Esta es una interfaz sencilla para controlar el hardware del control a través del bus I2C.

Componentes

- Raspberry Pi 4
- Push buttons
- LEDs
- Pantalla SPI TFT
- Módulos I2C PCF8574

Descripción de los componentes

- **Raspberry Pi4:**

Es una microcomputadora de placa única desarrollada por la Raspberry Pi Foundation, la cuál es un sistema embebido. Además la Raspberry Pi tiene la habilidad de interactuar con el mundo exterior, puede ser usada en una amplia variedad de proyectos digitales, desde reproductores de música y video, hasta estaciones meteorológicas.

Procesador: Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz.
RAM: 2GB, 4GB u 8GB LPDDR4.
Puertos: 2 USB 3.0, 2 USB 2.0, 2 micro HDMI (4K), Gigabit Ethernet, GPIO de 40 pines, con soporte para SPI, I2C, UART, PWM.

Se va a utilizar como la plataforma principal que controla la lógica del juego, la interfaz de usuario y la comunicación con los componentes periféricos.

- **Push buttons:**

Es el elemento más básico y el más usado en los sistemas de control. Estos dispositivos en su configuración básica necesitan tener una presión mecánica en ellos para mantenerse conectados o desconectados.

Funcionamiento: Cuando se presionan, cierran un circuito, permitiendo el paso de corriente (estado lógico bajo o alto, según configuración).
Conexión típica: Se conectan entre un pin GPIO y tierra (GND), usualmente con una resistencia.

Se utilizarán botones físicos para que los jugadores interactúen con el juego, presionando el color correspondiente al LED encendido.

- **LEDs:**

Un diodo LED es un semiconductor que emite luz cuando una corriente eléctrica lo atraviesa.

Funcionamiento: Requieren una resistencia en serie para limitar la corriente (típicamente 220Ω a 1kΩ).

Colores comunes: Rojo, verde, azul, ámbar, blanco.
Control: Encendidos y apagados por salidas GPIO en estado ALTO o BAJO.

Se emplearán LEDs de colores (rojo, azul, verde y amarillo) para mostrar las secuencias que los jugadores deben seguir. Cada LED se iluminará durante un tiempo específico, y los jugadores deberán presionar el botón correspondiente a ese color.

- **Pantalla SPI TFT:**

La pantalla tiene una estructura tipo sándwich con material de cristal líquido relleno entre dos placas de vidrio, donde cada píxel en una matriz activa está emparejado con un transistor que incluye un capacitor que le da a cada subpíxel la capacidad de retener su carga, en lugar de requerir que se envíe una carga eléctrica cada vez que sea necesario cambiarlo. La capa TFT controla el flujo de luz, un filtro de color muestra el color y una capa superior alberga la pantalla visible.

Resolución: 240x240, hasta 65K colores.

Se usará una pantalla TFT SPI ST7789 para mostrar el menú principal del juego y los resultados de la partida, como el puntaje más alto y el jugador ganador.

- **Módulos I2C PCF8574:**

El PCF8574 expansor de E/S es útil en aplicaciones donde se requiere controlar múltiples dispositivos con un número limitado de pines, como en sistemas embebidos, automatización doméstica, control de sensores, y paneles de control. Su capacidad para simplificar el diseño de circuitos y reducir la complejidad del cableado lo convierte en una opción popular para proyectos que involucran microcontroladores como Arduino y Raspberry Pi.

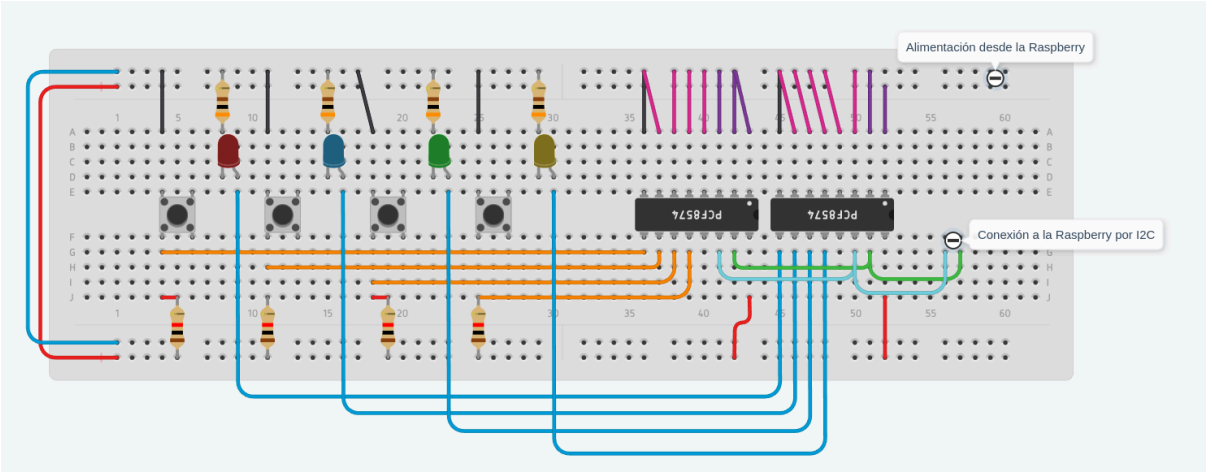
Interfaz: I2C (usa solo SDA y SCL), hasta 8 dispositivos en el mismo bus.
Pines de E/S: 8 pines configurables como entrada o salida digital.
Dirección configurable: Hasta 8 direcciones I2C usando 3 pines de dirección (A0-A2).

Se utilizarán para tener más entradas y salidas mediante I2C.

Descripción de la implementación del juego:

Conexión Hardware

Diagrama de conexión del control



El control utiliza dos módulos PCF8774, utilizando dos direcciones en el programa. El primer módulo controla las entradas de los botones, mientras que el segundo controla la iluminación de los leds. Los controles se conectan al bus I²C de la Raspberry (GPIO 2 - SDA y GPIO 3 - SCL).

La pantalla ST7789 se conecta mediante SPI utilizando los pines GPIO 10 para MOSI, GPIO 11 para SCLK, GPIO 23 para el reset y GPIO 24 para DC. El pin BLK es para ajustar la intensidad de brillo, pero en este caso no lo utilizamos ya que no era esencial.

Métodos de interconexión de los componentes

Para conectar los leds y los botones se utilizarán dos módulos PCF8574, que son expansores I²C. El módulo para los leds solo se utilizará como salida, mientras que el de los botones será la entrada. Cada control requiere de dos PCF8574, uno para entrada y uno para salida, físicamente se deben de configurar sus direcciones mediante los pines A0, A1 y A2. Para este

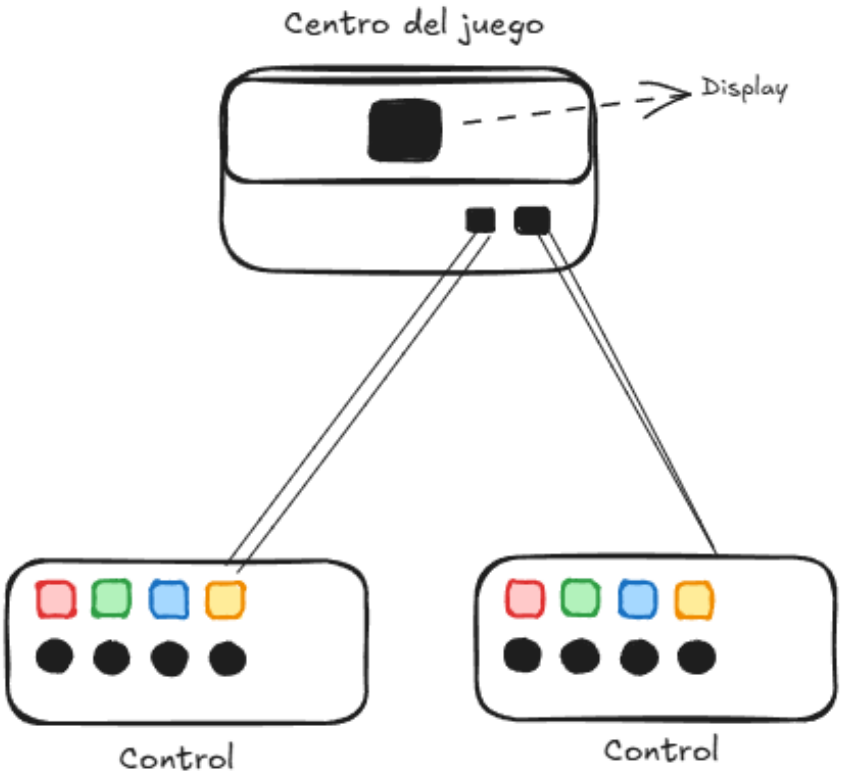
proyecto el primer control tiene las direcciones 0x20 y 0x21, mientras que el segundo tiene las direcciones 0x22 y 0x23.

El display utilizado es el ST7789, que es una pantalla TFT RGB, la cual utiliza el protocolo de comunicación SPI en el modo 3. Para utilizarla incorporamos al proyecto la biblioteca Display Lib RPI, que tiene controladores para esta y otras pantallas similares en el lenguaje C++, es importante mencionar que esta biblioteca requiere compilar con el estándar C++23.

El siguiente código son las funciones que usamos para manejar el control:

```
#include "Controller.h"
#include <cstdint>
#include <cstdio>
#include <wiringPi.h>
#include <wiringPiI2C.h>
#include <unistd.h>

Controller::Controller(const uint32_t input_direction, const uint32_t output_direction,
const uint8_t mask)
{
    input_fd = wiringPiI2CSetup(input_direction);
    output_fd = wiringPiI2CSetup(output_direction);
    this->mask = mask;
}

Controller::~Controller()
{
    wiringPiI2CWrite(input_fd, 0);
    wiringPiI2CWrite(output_fd, 0);
    close(input_fd);
    close(output_fd);
}

void Controller::write_in_leds(const uint8_t value)
{
    wiringPiI2CWrite(output_fd, value);
}

uint8_t Controller::ReadInput()
{
    uint8_t valueRead = wiringPiI2CRead(input_fd);
    valueRead &= mask;
    return valueRead;
}
```

Diagrama del prototipo



Software

Por parte del software tenemos nuestro proyecto de github, donde se puede revisar todo el código. Aquí en este archivo solo pondré las clases principales y una explicación rápida.

La siguiente función maneja principalmente cuando se selecciona el juego, revisa si están jugando dos jugadores.

```
bool gaming_simon_says(int number_players, std::shared_ptr<SimonDice> game,
std::shared_ptr<Display> display, std::shared_ptr<Controller> contrls_p1,
std::shared_ptr<Controller> contrls_p2 = nullptr) {

    if (game == nullptr || display == nullptr || contrls_p1 == nullptr) {
        std::cerr << "Error: Game, Display or Controller is null." << std::endl;

        if (number_players == 2 && contrls_p2 == nullptr)
            std::cerr << "Error: Second Controller is null." << std::endl;

        return false;
    }

    std::string print_info = number_players == 1 ? "1P" : "2P";
    std::cout << "Iniciando juego en modo de " << print_info << "\n";

    game->initialize_sequence();
```

```
    do
    {
        //display->DrawInGameCounter(game->get_current_level() * 10, YELLOW);
        if (!game->play_level(contrls_p1, display))
            break;

        if (number_players == 2 && !game->play_level(contrls_p2, display, false))
            break;

    } while (!stopProgram);

    display->DrawGameOver((game->get_current_level() - 1) * 10);

    if (((game->get_current_level() - 1) * 10) > max_score)
    {
        max_score = ((game->get_current_level() - 1) * 10);
        write_score_file("data/score.txt", max_score);
        std::cout << "Nuevo record: " << max_score << "\n";
    }
    std::this_thread::sleep_for(std::chrono::seconds(2));
    display->DrawMenu();

    return true;
}
```

El siguiente while es para manejar el menú principal del programa, donde recibe las entradas del control y se va moviendo mediante la interfaz y dependiendo de cual sea seleccionado manda a ejecutar la funciones para eso.

```
    while (!stopProgram)
    {
        uint8_t input = 0;
        int selection = 0;

        while (input == 0)
            input = contrls_p1->ReadInput();

        contrls_p1->write_in_leds(input);

        printf("Value: %u\n", input);
        selection = static_cast<int>(input);

        while (input != 0)
            input = contrls_p1->ReadInput();

        contrls_p1->write_in_leds(0x0);

        if (selection == INPUT_VALUE_YELLOW)
        {
            display->DrawGameOver(420); //TODO: Crear función para despedirse
            stopProgram = true;
        }
        else if (selection == INPUT_VALUE_GREEN)
```

```
        {

            if (current_menu == SINGLEPLAYER)
            {
                gaming_simon_says(1, game, display, contrls_p1);
            }
            else if (current_menu == MULTIPLAYER)
            {
                gaming_simon_says(2, game, display, contrls_p1, contrls_p2);
            }
            else if (current_menu == RECORDS)
            {
                std::cout << "Mostrando records\n";

                display->DrawInGameCounter(max_score, YELLOW); //TODO: Crear funcion para
mostrar el record
                std::this_thread::sleep_for(std::chrono::seconds(3));
                display->DrawMenu();
            }
        }
        else if (selection == INPUT_VALUE_BLUE)
        {
            current_menu = (current_menu - 1 + 3) % 3;
        }
        else if (selection == INPUT_VALUE_RED)
        {
            current_menu = (current_menu + 1) % 3;
        }

        display->SelectOption(static_cast<MENU_OPTIONS>(current_menu));

        std::this_thread::sleep_for(std::chrono::milliseconds(400));
    }
```

La siguiente función es la que se encarga de jugar el nivel, a su vez modifica el display.

```
bool SimonDice::play_level(std::shared_ptr<Controller> cntrls, std::shared_ptr<Display>
display, bool is_player_1) {

    if (cntrls == nullptr || display == nullptr) {
        std::cerr << "Error: Controller or Display is null." << std::endl;
        return false;
    }

    if (is_player_1)
        add_to_sequence();

    std::cout << "Nivel " << current_level << ". Memoriza la secuencia." << std::endl;
    display->DrawInGameCounter(get_current_level() * 10, YELLOW);

    for (int i = 0; i < sequence.size(); ++i) {
        if (color_name_map.find(sequence[i]) != color_name_map.end()) {
```

```
            std::cout << color_name_map.at(sequence[i]) << std::flush << std::endl;
            cntrls->write_in_leds(color_output_map.at(sequence[i]));
        } else {
            std::cerr << "Error: Key not found in color_name_map for sequence element " <<
sequence[i] << std::endl;
            return false;
        }
        std::this_thread::sleep_for(std::chrono::seconds(1));
        cntrls->write_in_leds(0x0);
        std::this_thread::sleep_for(std::chrono::milliseconds(350));
    }

    display->DrawInGameCounter(get_current_level() * 10, BLUE);

    std::vector<int> user_sequence;
    for (size_t i = 0; i < sequence.size(); ++i) {
        uint8_t input = 0;
        std::cout << "Introduce el elemento " << i << " de la secuencia: " << std::flush <<
std::endl;

        while (input == 0) {
            input = cntrls->ReadInput();
        }
        cntrls->write_in_leds(input);

        if (color_input_map.find(input) != color_input_map.end()) {
            user_sequence.push_back(color_input_map.at(input));
        } else {
            std::cerr << "Error: input inesperado (" << static_cast<int>(input) << ") no
está en color_input_map.\n";
        }

        std::cout << "Valor presionado: " << static_cast<int>(input) << std::flush <<
std::endl;

        while (input != 0) {
            input = cntrls->ReadInput();
        }
        cntrls->write_in_leds(0x0);
        std::this_thread::sleep_for(std::chrono::milliseconds(400));
    }

    if (check_sequence(user_sequence)) {
        std::cout << "¡Correcto! Pasas al siguiente nivel." << std::endl;
        return true;
    } else {
        std::cout << "Incorrecto. Fin del juego." << std::endl;
        return false;
    }
}
```

Es mucho código para ponerlo documentado aquí en este archivo, pero en la liga del repositorio viene todo el software desarrollado para hacer posible este proyecto.

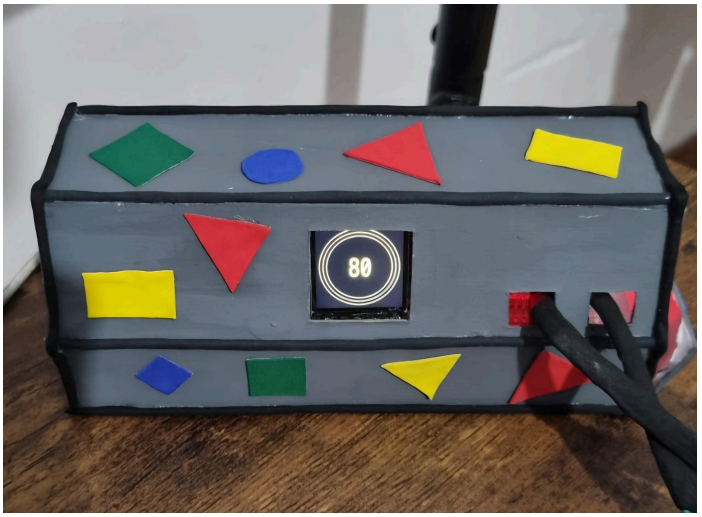
Evidencias de funcionamiento

EvidenciaSimonDiceEmbebido.mp4

Prototipo consola y control, menú inicial mostrado en pantalla.

Control de juego encendido.

Consola de juego, pantalla "Game Over".



Consola de juego, pantalla mostrando puntaje.

Conclusión generales

Este proyecto presenta una solución sencilla, pero divertida y educativa, utilizando herramientas como la Raspberry Pi 4, LEDs y push buttons. A través del juego de "Simón Dice" versión de colores, los jugadores podrán mejorar su memoria y coordinación mientras disfrutan de una competencia interactiva. Además, al ser un proyecto donde se ponen en práctica todos los conocimientos adquiridos durante el laboratorio de Fundamentos de Sistemas Embebidos. Usamos i2c para la conexión y quedó muy claro que con 2 pines de la raspberry podemos crear varios esclavos para diferentes cosas y no estar limitados con el número de pines. En la parte de software fue fácil programar el juego y quedó claro cómo programar los GPIOs de la raspberry.

Conclusiones individuales

Moreno Chalico Edgar Ulises

En este proyecto se resumen la mayoría de los temas abarcados en el curso para la creación de este producto enfocado en mejorar la agilidad mental de una manera desafiante y entretenida; principalmente nos centramos en utilizar los principales métodos de comunicación que ofrece la Raspberry PI, en este caso con I²C y SPI. Personalmente la parte que más me gustó experimentar fue el manejo del display, ya que este tipo de componentes suelen ser algo complicados de configurar correctamente pero se logró añadir este elemento para ofrecer un poco más de retroalimentación visual. Durante el desarrollo de este proyecto no hubo muchos problemas, en general el flujo de trabajo fue ágil y logramos completar satisfactoriamente este proyecto.

Nava Alberto Vanessa

El proyecto fue una manera estratégica para poder juntar todo lo aprendido en el laboratorio y la teoría visto a lo largo del semestre. Además de que logramos coordinar de manera sencilla el trabajo en equipo, logramos coordinarnos de manera en que pudimos resolver los problemas que se presentaron durante el desarrollo del proyectoen sí, no fue un proyecto tan complicado de implementar. Los mayores retos que tuvimos en la implementación del proyecto principalmente fue conseguir los módulos I2C, por alguna razón no lograbamos conseguir los módulos, pero lo solucionamos pidiéndolos por Internet. Fuera de eso no tuvimos complicaciones técnicas ni durante la creación del prototipo.

Reyes Valderrama Rodrigo Miguel

El proyecto sirvió mucho para poder implementar lo aprendido durante el semestre en la clase de Fundamentos de Sistemas de Embebidos, la parte de software fue muy fácil crearla. La parte interesante de implementar fue la programación y la conexión por i2c, fue increíble cómo de dos pines de la Raspberry Pi sacabamos varios esclavos para tener mas entradas y

salidas. No tuvimos muchas complicaciones, fue muy divertido de programar y fue satisfactorio ver el proyecto concluido. Con nuestro juego creado me di cuenta que tengo que ejercitar mi memoria.

López Cantarell Diego Emir

En este proyecto se aplicaron los conocimientos adquiridos durante el semestre, para la conexión usamos I2C lo cual nos generó un atraso ya que no encontrábamos el componente en ninguna tienda, la programación del proyecto igual que durante el semestre fue lo más sencillo, es importante ver hasta donde se puede llegar con un sistema embebido fuera de las prácticas normales de la materia y este proyecto nos da una pequeña muestra de ello.

Bibliografía

Colaboradores de La Nación. (2025). *"Cómo recuperar la memoria y la concentración dañada por las redes sociales y el multitasking"*. La Nación. Recuperado de: https://www.lanacion.com.ar/lifestyle/cuidado-cuerpo-belleza/como-recuperar-la-memoria-y-la-concentracion-danada-por-las-redes-sociales-y-el-multitasking-nid08012025/

Colaboradores de MASAM . (2022). *"Control electromecánico: Push buttons"*. Facultad de estudios Superiores Cuautitlán. Recuperado de: https://masam.cuautitlan.unam.mx/dycme/ce/push-buttons/

Colaboradores de MCI Electronics. (S/F). *"¿Qué es la Raspberry Pi?"*. Raspberrypi.cl. Recuperado de: https://raspberrypi.cl/que-es-raspberry/

Gerard. (2024). *"¿Qué es un Diodo LED y cómo funciona?"*. Staria technologies. Recuperado de: https://stariatechnologies.com/que-es-un-diodo-led-y-como-funciona/

Colaboradores de Orient Display. (2022). *"¿Qué es una pantalla LCD TFT? Conocimiento Básico"*. Orient Display. Recuperado de: https://www.orientdisplay.com/es/knowledge-base/tft-basics/what-is-thin-film-transistor-tft/

Colaboradores de Unit Electronics. (S/F). *"PCF8574 Expansor E/S I2C Módulo o IC."* UElectronics. Recuperado de: https://uelectronics.com/producto/pcf8574-expansor-e-s-i2c-modulo-o-ic/

Raspberry Pi Foundation. (S/F.). *Raspberry Pi 4 Model B*. Raspberry Pi. Recuperado de: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

Texas Instruments. (S/F). "*PCF8574 Remote 8-Bit I/O Expander for I²C Bus*". Recuperado de: https://www.ti.com/lit/ds/symlink/pcf8574.pdf