**JUEGO SIMON**

**grupo 4**

**integrantes**

Leandro Delgado - Daiana Rivera - Rodrigo Acevedo- Adrian Hidalgo-Ramiro Averbuj

especialidad en Computación, Escuela Técnica N°32 D.E. 14

4to2da: Proyecto Informático

Profesor : Gonzalo N. Consorti

empieza el 24 de septiembre

**Introducción AL juego de Simon Dice**

bueno esta es una introducción de lo que sería nuestro proyecto de proyecto informática conformada por gonzalo consorti , en esta ocasión con mi grupo decidimos elegir hacer el juego simon , sacamos diferentes opiniones y es la que nos pareció más factible para nuestro grupo

el juego de simon trata de los leds que puedas en lo posible más de 2 mínimo un lcd y botones para cada led y para que inicie el juego, nosotros vamos a usar un lcd(por ahora es uno) para explicar como va el juego como ejemplo : el tiempo “ siguiente ronda”, “bienvenido” , “perdiste”, etc. Para los que no saben que es el juego de simon , trata de un juego de memoria en la que tenes por ejemplo colores y botones se ilumina uno y apretas el boton del que se iluminó y ahi suman otro con el mismo que se iluminó anteriormente con otro nuevo y así en bucle hasta que pierdas, eso es lo que vamos a tratar de hacer pero más divertido y entendible para que no se pierdan y puedan disfrutar del juego sin que se aburran, dicho esto les voy a pasar a explicar los componentes que vamos a usar y para que los vamos a usar en la estructura de este arduino para que no se pierdan y entiendan cómo funciona cada cosa ,

**objetivos:**

Aumento Progresivo de la Dificultad: El juego debe incrementar la dificultad de manera que vaya poco a poco aumentando la longitud de la secuencia de colores y reduciendo el tiempo de espera entre cada color a medida que el jugador avanza. Esto asegura que el juego crezca progresivamente, manteniendo el interés del jugador haciendolo sentir muy frustrado.

Fluidez del Juego: Se debe garantizar que el juego funcione sin retrasos ni fallos, asegurando que la interacción entre las secuencias de LEDs, los sonidos emitidos por el buzzer y las respuestas del jugador sea continua y sin interrupciones. La velocidad de respuesta y la sincronización entre los elementos del juego son esenciales para que la experiencia de usuario sea óptima.

Reducción de Errores: Es fundamental reducir los errores tanto en la lectura de las entradas del jugador (con técnicas como el debouncing para evitar pulsaciones incorrectas) como en la comparación de las secuencias ingresadas con las mostradas. Además, se debe asegurar que el sistema de verificación sea preciso para evitar que el jugador avance con una secuencia incorrecta o que se registre un error cuando no debería haberlo.

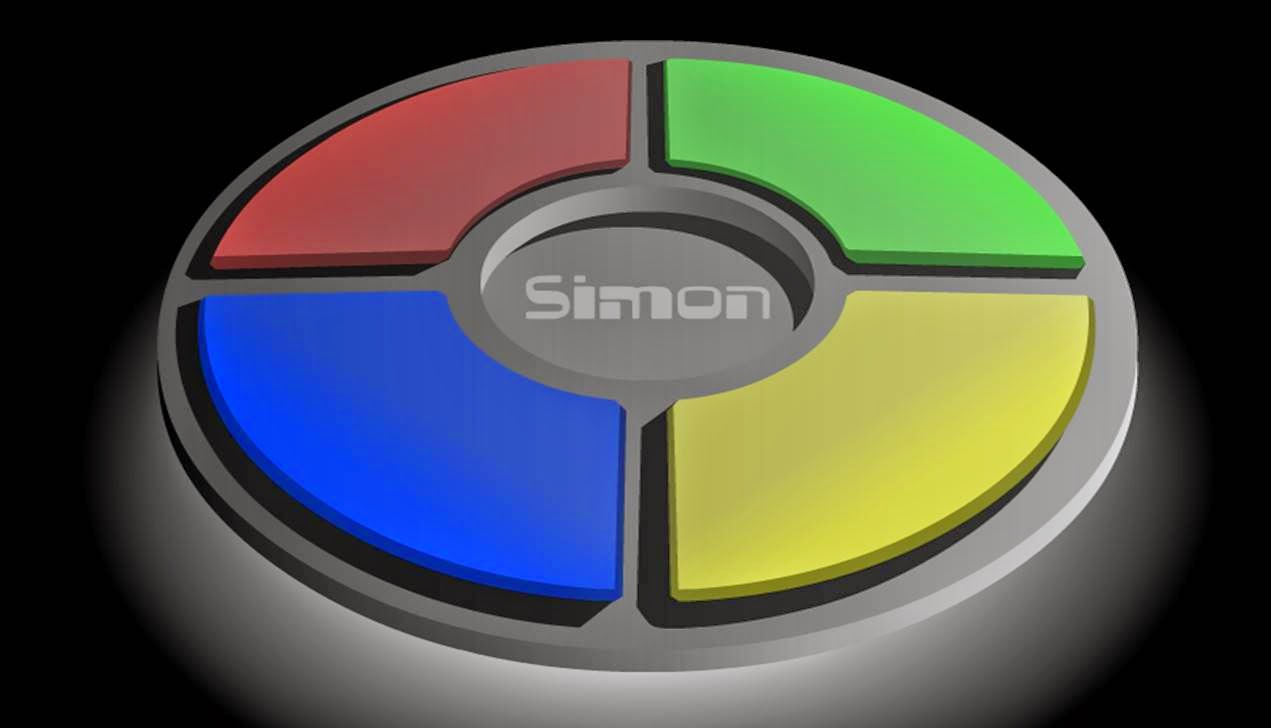
Facilidad en la jugabilidad : se enfocará en implementar señales de indicación claras. Esto incluye el uso de LEDs y sonidos para indicar correctamente si el jugador ha acertado o cometido un error. Los LEDs se iluminarán con colores específicos según la secuencia y el estado del juego, mientras que el buzzer emitirá tonos diferenciados para distinguir entre aciertos, errores y el final de cada secuencia. Estas señales deben ser intuitivas para que el jugador entienda rápidamente el estado de su progreso en el juego.

Velocidad de Juego: La velocidad con la que se muestran las secuencias debe ajustarse progresivamente para mantener el juego desafiante. A medida que el jugador avanza en los niveles, el intervalo entre los colores de la secuencia se acortará, incrementando la dificultad de manera equilibrada y sin que se vuelva tan dificil , igual es facil..

Bueno al principio nos turnamos , que ibamos a ser yo ya tenia claro que iba a hacer que era algo del codigo y la parte mas divertida que viene a ser lo fisico , lo bueno es que en mi grupo todos trabajan y no me tengo porque preocupar .

Al principio no me costó elegir un lugar por el cual empezar ya que no había mucho que hacer , asi que en base a los materiales que nos pidieron usar ,investigamos , bueno cada uno tenía su fortaleza yo al principio no tenía mucho en lo que aportar en base al código pero , estuve un poco se podría decir ayudando a leandro que es el que más hizo sobre el código y ayudando en lo errores básicamente , después en lo físico ,ahi ya no es tan dificil , bueno esto es mi recorrido.

Yo de chiquito tenia el memotest o simon dice cuadrado pero tambien tenia este que es lo mismo , pero tenia uno que hablaba por el agujerito donde estaba



**Lista de componentes:**

Botones:vamos a necesitar 4 botones los que se va a poder interactuar en el juego ,Cada uno de los botones está vinculado a un color específico, como rojo, verde, azul o amarillo, y su función principal es permitir que el jugador ingrese su respuesta al intentar repetir una secuencia de luces mostrada por el las luces.

El sistema genera una secuencia aleatoria de luces y sonidos, y el jugador debe presionar los botones en el orden correcto para replicar esa secuencia. Cuando el jugador presiona un botón, el sistema verifica si la respuesta es correcta comparando la secuencia ingresada con la secuencia original. Si el jugador se equivoca, el sistema activa una secuencia de error con luces titilando y un sonido para indicar el fallo. En caso de éxito, el sistema avanza a la siguiente ronda, aumentando la longitud de la secuencia. Además, los botones proporcionan una respuesta visual y sonora, permitiendo que el jugador reciba información sobre su desempeño. También se utilizan para detectar el tiempo en que el jugador realiza la acción, lo que puede influir en la dificultad del juego.



Pantalla LCD: durante el juego. El LCD se utiliza para mostrar información clave, como el nivel actual del juego, el tiempo transcurrido, o mensajes que indican si el jugador ha cometido un error o ha completado correctamente una secuencia. Cuando el jugador inicia el juego, el LCD muestra un mensaje de bienvenida, y conforme avanza en los niveles, se actualiza con el número de nivel actual. Además, el LCD se usa para mostrar el tiempo transcurrido en cada ronda, lo que permite al jugador ver cuánto ha jugado y cuánta dificultad ha alcanzado. Si el jugador comete un error en la secuencia, el LCD puede mostrar un mensaje de error, indicándole que la secuencia no fue correcta y que debe intentar nuevamente. Proporcionándole la información necesaria para seguir participando. En los niveles más altos, el LCD puede mostrar el número de intentos o cuántas secuencias ha completado correctamente el jugador. Además, el LCD contribuye a mejorar la experiencia del juego al hacerla más visual, ayudando a que el jugador se enfoque en los objetivos y avance en el juego de manera estructurada y clara.

Buzzer: para que suene un ruido cada vez que pierdas , un ruido para los colores distintos y ruidos cuando ganas .

El buzzer cumple un papel importante al proporcionar ruido al jugador, lo que mejora la experiencia del juego y facilita la interacción.

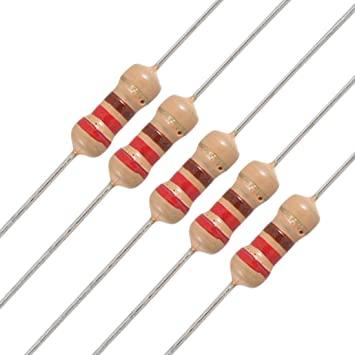
Emite diferentes tonos y duraciones de sonido en función de la acción que se realice en el juego. Por ejemplo, cuando el sistema muestra una secuencia de luces, el buzzer emite un tono específico para cada color o LED activado, lo que ayuda al jugador a asociar cada botón con un sonido particular. Este sonido sirve como una pista para recordar la secuencia. Además, el buzzer también se utiliza para indicar errores. Si el jugador presiona un botón incorrecto, el buzzer emite un tono de error, lo que avisa al jugador de que ha cometido un fallo.

Los tonos emitidos por el buzzer pueden variar en frecuencia, lo que permite al sistema generar una variedad de sonidos para diferenciar las acciones dentro del juego. Sin el buzzer, el juego perdería una parte fundamental de la audicion, y la interacción sería menos divertida, ya que solo se utilizarían los LEDs para comunicar el progreso del jugador. El buzzer también ayuda a crear una atmósfera entretenida, ya que la combinación de luces y sonidos incrementa la emoción del juego y hace que el jugador se concentre en completar las secuencias correctamente.

Led: bueno los leds van a ser 4 y vamos a usarlos , primero se va iluminar uno de forma random y apretas su boton , despues se reinicia y se a iluminar el que ya tocaste y otro random así hasta que pierdas en el juego , los LEDs normales juegan un papel fundamental como medio visual para guiar al jugador a través del juego. Cada LED está asociado con un color específico, como rojo, verde, azul o amarillo, y estos colores se utilizan para mostrar las secuencias que el jugador debe repetir. Cuando el juego inicia, el sistema enciende los LEDs de forma secuencial, y el jugador debe presionar los botones correspondientes en el mismo orden en el que los LEDs se iluminaron. Los LEDs proporcionan una forma clara de mostrar la secuencia de colores, haciendo que el jugador pueda asociar un color con una acción de presionar un botón. Además, los LEDs sirven para dar retroalimentación visual inmediata sobre las acciones del jugador. Si el jugador presiona el botón correcto, el LED correspondiente se ilumina y, en algunos casos, se puede acompañar de un tono emitido por el buzzer. En caso de error, los LEDs pueden parpadear o apagarse brevemente, indicando al jugador que algo salió mal. También, los LEDs se utilizan para mostrar el progreso del juego, ya que con cada nivel, el sistema enciende más LEDs en una secuencia cada vez más larga y compleja. Esto hace que el jugador esté constantemente informado sobre cómo debe proceder. Además, los LEDs se utilizan para mostrar un error de forma visual, parpadeando rápidamente o activándose todos los colores al mismo tiempo como señal de fallo. Sin los LEDs, el jugador no tendría un medio visual para interactuar con el juego, l. Los LEDs son, por tanto, un componente esencial para el funcionamiento del juego, proporcionando tanto pistas para la secuencia como indicaciones claras del rendimiento del jugador

Resistencias:

resistencias para los componentes, mayormente lo usamos con 220 ohms



Las resistencias en el juego protegen los componentes electrónicos y garantizan el funcionamiento adecuado del circuito. Son necesarias para limitar la cantidad de corriente que pasa a través de los LEDs, los botones y otros componentes, evitando que se dañen por sobrecarga. En el caso de los LEDs, las resistencias se colocan en serie con los mismos para evitar que fluya demasiada corriente y para mantener su funcionamiento dentro de parámetros seguros. Sin estas resistencias, los LEDs podrían quemarse debido a una corriente excesiva. Además, las resistencias también pueden usarse para ajustar las señales de entrada de los botones, asegurando que las señales que llegan a la placa de circuito no sean demasiado fuertes o débiles.



Potenciómetro:para configurar el LCD y así aplicarle una resistencia específica aca pongo imagen que es muy distinta a lo de tinkercad. El potenciómetro en el juego de se utiliza como un componente interactivo que permite al usuario ajustar ciertos parámetros del juego.

Un potenciómetro es un tipo de resistencia variable que puede modificar su valor dependiendo de la posición de su eje, haciendo que los colores cambien más rápido o más lentamente dependiendo del valor configurado por el jugador.

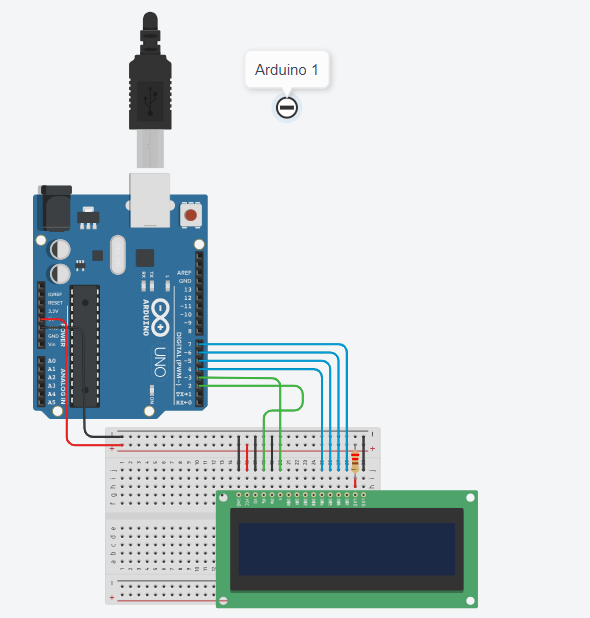
Esto lo vamos a utilizar mayormente para cambiar la intensidad de los LEDs, permitiendo que el jugador tenga una experiencia más personalizada. En general, el potenciómetro proporciona una forma sencilla y efectiva de ajustar parámetros en tiempo real, mejorando la interactividad y la flexibilidad del juego.

**24 de septiembre de 2024**

Lo primero que hice fue investigar en youtube sobre el LCD que para mi es algo importante en el proyecto , bueno a lo que quería aspirar era poner el tiempo , cambios de los rounds , y información como” bienvenido” o mensajes , bueno aca mando el primer intento de lo que vi en youtube y lo que se.

Esta estructura la voy a usar con los 3 códigos aviso, porque no puedo poner texto y mucha multimedia y prefiero avisarlo antes para que no queden dudas, igual dejo la imagen para que vean como se conecta un lcd con mis conocimientos (POR AHORA) .

Seguramente falte algo porque en los ejercicios que hicimos usabamos el potenciometro que yo entendi que lograba hacer con eso y me guié más por videos de youtube , pero voy a preguntarle al profe despues si está bien conectado



**1er testeo**

Lo que intente hacer en este intento es como contar el tiempo en una ronda y saber cómo colocar cada elemento y donde conectarlo esto se usaría para cuando estás en la ronda y te cuente el tiempo todavía no sabemos si vas a tener un tiempo máximo para completar eso estaria bueno como una dificultad para tenerlo en cuenta, lo hice con un tutorial de youtube   
void setup()  
{  
lcd.begin(16,2);  
}  
void loop ()  
{  
 lcd.setCursor(0,0);  
lcd.print("Hola, Han pasado");  
lcd.setCursor(0,1);  
lcd.print(millis() / 1000);  
lcd.print ("seg.");  
}

el lcd.setCursor lo que hace es poner en las coordenadas(0,0) , el lcd.print osea va a poner Hola, Han pasado” y después va a ir seguido en la coordenada (0,1) van a aparecer los segundo en los que tardas en completar el nivel y se te van sumando a una cuenta ,hasta que pierdas la secuencia

**2do testeo**

Con el segundo intento de aprender hice un print en el lcd que diga “bien hecho” y parpadee ,como por ejemplo usarlo cuando te completaste el round, igual no creo que lo vayamos a usar porque es muy simple y te aparezca para saber que lo completaste y que vas bien no hay mas explicacion 🍷 , bueno aca lo voy a mostrar , no es muy difícil de hacer pero se te puede conectar con otro print es lo complicado lo que vamos a tener que usar es lcd.clear() para borrar lo que pusiste y que no se quede guardado en el lcd

| void setup() { lcd.begin(16,2); } void loop () { lcd.setCursor(0,0); lcd.print("!Bien hecho"); lcd.display(); delay(200); lcd.noDisplay() delay(200); } |
| --- |

**3er testeo**

y ahora el último intento que hice fue algo un toquecito mas dificil por que aparte de poner “bien hecho ” hice que después que parpadee 3 veces te ponga otra palabra que diga , bueno esto capaz , o seguramente no se use en el código porque es mejor hacer lo principal para poder terminarlo y no agregarle distintas cosas que puedan afectar al código, y intentar hacer lo más sencillo posible para no tomar mucho tiempo , aparte el lcd lo que si vamos a usar seguramente sea el tiempo para contabilizar los segundos que tardas en perder , y las rondas las vamos

contando , hasta que pierdas y se finaliza, de manera que cuando pierdas , se reinicia 🙂

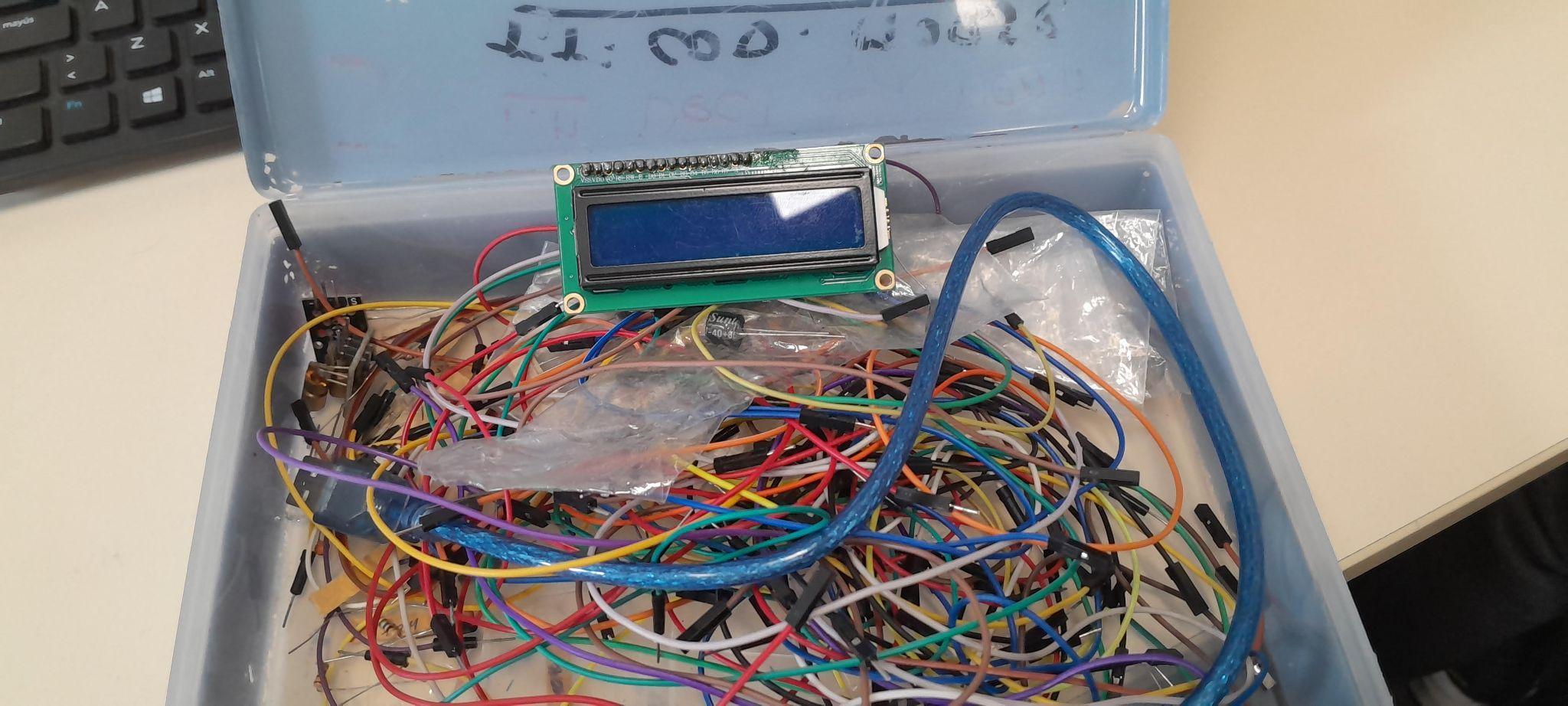
| void setup() {  lcd.begin(16, 2);  }  void loop()  {  if (counter < 3)  {  lcd.setCursor(0, 0);  lcd.print("!Bien hecho");  lcd.display();  delay(200);  lcd.noDisplay();  delay(200);  counter++; |  |
| --- | --- |

falta, igual tendría que investigar mas por que es lo que más se usa en este código ya que tendria que poner ,por ejemplo , un menú de opciones de que te diga si aprietas el botón del amarilllo haces tal cosa y así con los otros colores o ver por ejemplo no creo que se agregue por que es muy difícil, un ranking del que más lejos llegó , a ver esto del ranking es difícil hasta lo que yo se o suponer , no creo que lo hagamos pero es una buena idea por si alguien lee esto, y la quiera aplicar para su proyecto de Juego Simon ,estas ideas son buenas pero están alejadas de lo que vamos a hacer , por ahora. y estaria bueno poder usar el lcd ya que te acerca mas al juego y tenes mas informacion de lo mencionado,

Finalización de la clase 24/9/2024

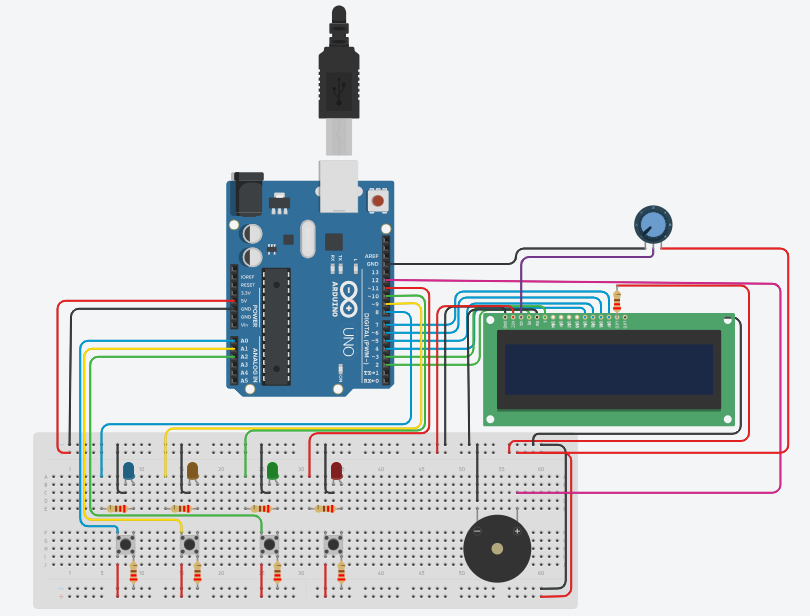
**8 de octubre de 2024**

En esta clase ya nos dieron los materiales para armar el juego desde el arduino



esta cajita traía cables , un LCD(como se puede ver en la pantalla) un arduino , resistencias y mas, tambien nos dieron otra caja llena de materiales , como led, buzzer y botones , que no le llegue a sacar fotos.

Después nos juntamos con el grupo y empezamos a hacer la estructura del juego empezamos con el código , por el cual lo primero que intentamos es hacer lo de las luces y los botones porque vendría siendo lo principal en esta estructura de código por el cual va a funcionar este juego, bueno haciendo lo del LCD le consultamos a nuestro profe sobre el potenciómetro , porque yo no lo había puesto y si iba. ¿Por qué? Lo que nos contestó el profe fue : lo que hace este potenciómetro es iluminar el contraste del LCD , lo único malo(difícil) es que hay que encontrar una resistencia específica para que se ilumine bien y no tenga errores de no poder ver bien el contraste. Después el buzzer lo usamos para que tenga sonido en los errores en las luces por ejemplo, el rojo tiene un ruido , el verde tiene otro y así variando con los otros colores y estaría bueno agregar un menú al lcd para que puedas elegir que queres hacer .

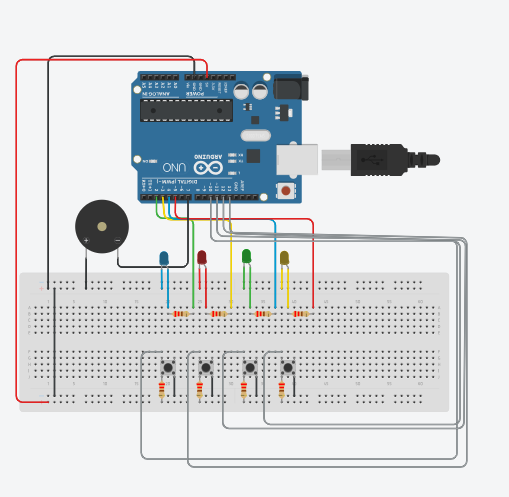


asi es como va quedando la estructura del simon dice en tinkercad en realidad no va a ser asi

Finalización de la clase 8/10/2024

**15 de octubre de 2024**

En esta clase me concentré más en completar la carpeta de campo para ponerme al dia lo unico nuevo que me llegó fue que dos compañeros hicieron una estructura y código pero que está mal porque falta el LCD pero igual lo podemos usar como estructura del juego de simon aca mando🙂



el codigo esta bien hecho por ahora pero lo inicias y siempre empieza amarillo y despues rojo, pero si erras se reinicia empieza por amarillo y despues no da rojo eso esta bien, lo que hay que cambiar es que empiece por diferentes colores .

Esto es lo que hicimos en esta clase . Avanzamos en cuestión código pero nos falta una barbaridad para terminar.

Finalización de la clase 15/10/2024

**22 de octubre de 2024**

En esta clase avanzamos muchísimo comparado con las anteriores clases en este dia terminamos lo que viene siendo la estructura y no falta un poco más de código , nuestro objetivo era avanzar

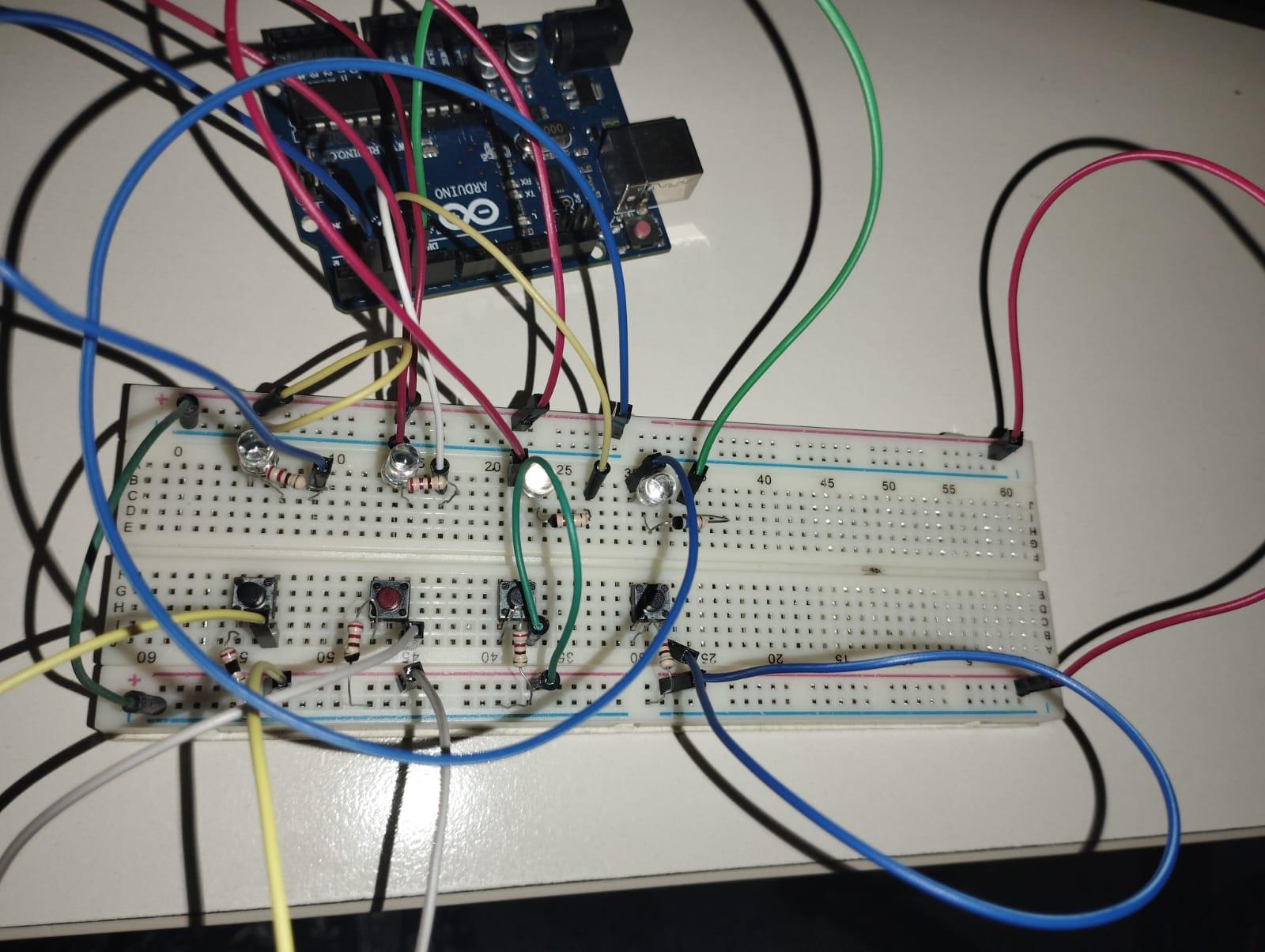
en el codigo asi no nos quedamos tan corto Y corregir los errores así en la próxima clase

lo tenemos limpito para seguir con el código y no tener que estar media hora buscando un error, y simplificarlo más , esta imagen es a lo que llegamos a hacer , de paso cambiamos la estructura de la anterior imagen , bueno llegamos a conectar todo y nos quedaria el codigo terminarlo más allá ,

Finalización de la clase 22/10/2024

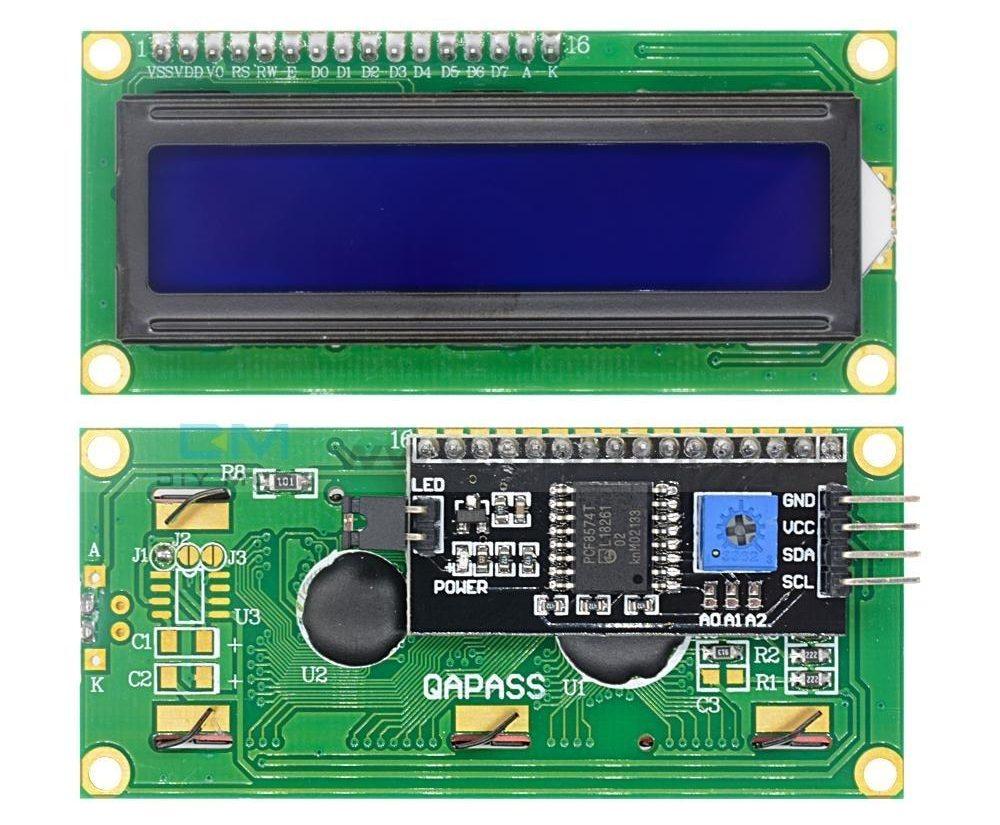
**29 de octubre de 2024**

Bueno cuando estábamos armando en fisico el juego de simon , tuvimos un problema y es que no funcionaba , no me acuerdo el error aca dejo una imagen de cómo era la estructura pero teníamos



varios problemas como era que corríamos el codigo y solo se prendia una luz y los botones no servían de nada seguramente , conectamos mal algunas cosas, y había otro problema y es que no nos alcanzaban los pines para poner el lcd y el buzzer ese parte de nuestros problemas , hasta que llegó consorti y no dio uno ya armado que tiene diferentes botones y luces , nosotros veníamos haciendo con luces led , pero nos dimo cuenta que había que hacerlo con luces RGB , para los colores y mas , asi que tuvimos que cambiar el código para que se adapte a este nueva estructura que nos dio consorti , l

a verdad lo veo más apto para que sea un juego y ahora tenemos el problema del lcd , y me acorde que existen dos lcd uno con 12 pines , que son los que deberíamos conectar al arduino ,pero no nos sirve y otro con 4(cuatro) pines hembras que es el más rentable y creo que más fácil de usar o a mi gusto fue más simple, que se llama LCD 16X2(I2C)



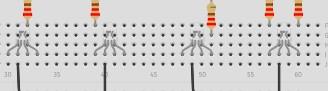
<https://www.electronics-lab.com/using-16x2-i2c-lcd-display-esp32/>

16:21 2/12/2024

LCD 16x2 con interfaz I2C es un componente para mostrar información de forma visual al jugador. Este tipo de pantalla tiene una resolución de 16 columnas por 2 filas, lo que permite mostrar textos y números de manera clara y compacta. La interfaz I2C es una forma eficiente de comunicarse con el microcontrolador utilizando solo dos pines (SCL y SDA), lo que simplifica el cableado y reduce el número de pines necesarios en comparación con las pantallas LCD tradicionales.

En el juego, el LCD se utiliza para mostrar el nivel actual del jugador, el tiempo transcurrido y mensajes de estado, como "Error" o "Nivel Completo", lo que proporciona una retroalimentación visual directa. Por ejemplo, al comenzar el juego, el LCD puede mostrar un mensaje de bienvenida, y luego actualizarse para reflejar el progreso del jugador en cada nivel. Además, el LCD puede ser utilizado para mostrar el tiempo restante para completar una secuencia, lo que ayuda al jugador a mantener un ritmo adecuado. También se puede emplear para mostrar mensajes de error, como cuando un jugador presiona el botón incorrecto, o para indicar que el nivel ha terminado. De este modo, el LCD contribuye significativamente a la experiencia del juego, proporcionando información clara y oportuna en todo momento.

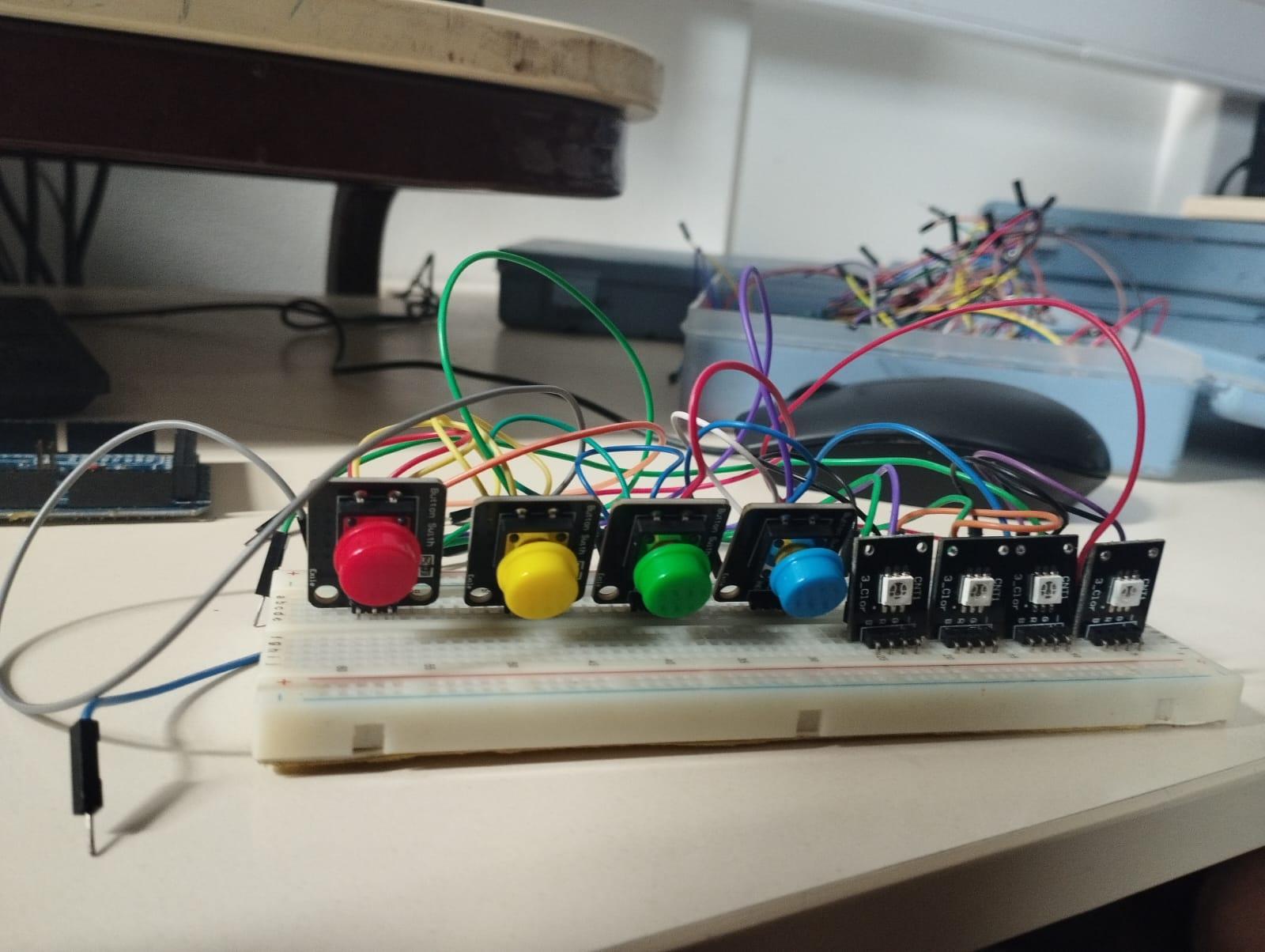
El uso del LCD mejora la interacción con el jugador, haciendo el juego más intuitivo y accesible. Sin él, sería difícil proporcionar información de manera eficaz, y la experiencia del jugador sería mucho menos atractiva.. Este lo usamos porque ahorra los pines y simplifica parte del código, ya que el uso de la librería I2C simplifica el código. Solo necesitas enviar instrucciones específicas para controlar la pantalla, y el controlador I2C se encarga de la comunicación en segundo plano.

También Cambiamos el led normal por el LED RGB

LED que combina tres diodos en uno: uno rojo, uno verde y uno azul. Esto permite generar una amplia gama de colores al ajustar la intensidad de cada uno de estos diodos. Cada LED RGB tiene tres pines de control, uno para cada color, que pueden ser modulados usando un microcontrolador como Arduino.

La combinación de estos colores puede producir colores como blanco, celeste, amarillo, entre otros, dependiendo de la intensidad de cada uno de los tres LEDs. Para lograr una variabilidad de colores, se suele usar la técnica de PWM (modulación por ancho de pulso), que ajusta la cantidad de tiempo que cada LED está encendido, lo que permite controlar la mezcla de colores. En el caso del juego "Simón Dice", los LEDs RGB se utilizan para mostrar diferentes colores como indicaciones visuales durante el juego, brindando una retroalimentación clara al jugador.

Al cambiar el color de los LEDs, el juego se hace más visualmente atractivo y fácil de seguir, mejorando la experiencia del usuario. Además, el uso de LEDs RGB permite programar efectos de luz personalizados, como parpadeos o transiciones de colores, que pueden hacer que el juego sea más interactivo y dinámico.

ya dicho esto paso a mostrar la estructura que nos concedió Consorti

Bueno por un lado tenemos los botones , que vendría a ser las entradas en este código y por otro lado tenemos las luces LED RGB a la derecha , en esta imagen todavía no sabíamos lo del LCD bueno ya con esto nuestro código cambia obviamente así que voy a pasar a explicar TODO el codigo proceso por proceso.

tengo un problema ahora y es que estoy usando la conexión de la escuela en mi casa , porque no tengo wifi y no me deja pasar el codigo a code block , y se entrega este lunes así que voy ponerlo normal y si puedo cambiarlo lo cambie espero leas esto profe , Bueno voy a pasar a explicar primero las declaraciones de variables y pines de este código de Simon Dice.Va basicamente todo el codigo pero resumido y en algunas partes con explicaciones adentro del codigo. Lo primero vienen a ser variables mas una biblioteca.

CODIGO

#include <LiquidCrystal\_I2C.h>  
#define BUZZER 7  
#define ENTRADA\_A A0  
#define ENTRADA\_B A1  
#define ENTRADA\_C A2  
#define ENTRADA\_D A3  
#define SDA A4  
#define SCL A5

Primero tenemos la Librería liquidCrystal I2C.h que sirve para manejar una pantalla LCD que se utiliza para mostrar el estado del jeugo como , el nivel o el tiempo. despues tenemos todos los pines de entrada que vienen a ser los botones que se presionan para replicar la secuencia de LEDs. Se configuraron en INPUT PULLUP , lo que significa que se activan al presionar.

ahora los Pines del Led: los pusimos como led1R,ledB,led2R ,led1G y led2G estos controlan los LEDs RGB y se utilizaron para mostrar el color especifico que le pusiste en la secuencia y sea mas dinamico para el usuario entender la consigna del juego ledB viene a ser azul solo , led1R viene a ser rojo solo, led1G viene a ser verde solo y led2R y led2G vienen a formar el color amarillo. También tenemos las variables SDA y SCL que son parte del lcd 16X2 ic2 que son las conexiones.

const int NIVEL\_MAX = 50; esta linea define una constante NIVEL\_MAX con el valor de 50. Esto establece el número máximo de niveles o elementos en la secuencia del juego.

int secuencia[NIVEL\_MAX]; declara un arreglo llamado secuencia de tipo int con un tamaño de NIVEL\_MAX (que es 50). Esto significa que el arreglo puede almacenar hasta 50 elementos enteros. Se utiliza para guardar la secuencia de colores o acciones que el jugador debe seguir en el juego. El tamaño del arreglo está limitado por el valor de NIVEL\_MAX, asegurando que el número de elementos no supere ese límite

int secuenciaUsuario[NIVEL\_MAX]; esto declara un arreglo llamado secuenciaUsuario de tipo int con un tamaño de NIVEL\_MAX (50). Este arreglo se utiliza para almacenar la secuencia de colores o acciones que el jugador ingresa durante el juego. Al igual que secuencia, su tamaño está limitado lo que asegura que el jugador solo pueda ingresar hasta 50 elementos en su secuencia

unsigned long tiempoInicio; La línea unsigned long tiempoInicio; declara una variable llamada tiempoInicio de tipo unsigned long. Esta variable se utiliza para almacenar el tiempo de inicio de algún evento, como el comienzo de una secuencia en el juego. El tipo unsigned long permite almacenar valores grandes y positivos, lo cual es útil para medir tiempos en milisegundos, como el transcurso de tiempo desde el inicio del juego.

Estas líneas de código definen la estructura básica de las variables que controlan el nivel, la secuencia de luces y la entrada del jugador, así como la secuencia del tiempo en el jeugo.

secuencia[ ] : almacena la secuencia de colores generados por el sistema secuenciaUsuario [ ] almacena la secuencia de colores generada por el sistema

tiempoInicio: mide el tiempo transcurrido desde que el juego comenzó.

todo esto permite que el juego sea más dinámico para que el juegue , no sea tan fácil y así va aumentando el nivel en cada paso que vas haciendo así no sea tan repetitivo y así a medida que va avanzando el que juega pueda ir progresando.

tenemos primero int melodia[ ] los numero adentro de las llaves son las frecuencias de notas musicales en hertzios en nuestro caso pusimos{262.196,196,220,196,0,247,262}; dentro del arreglo representan frecuencias de notas musicales en Hz (hertzios). Por ejemplo, 262 Hz es la frecuencia de la nota C , 196 Hz es la nota G, y 0 viene a ser un silencio.

también tenemos la duracionNotas[ ] ={4,8,8,,4,4,4,4,4 };

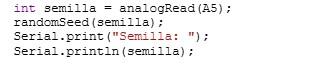
Esto contiene la duración de cada nota en la melodía. el 4 es un cuarto de tiempo y el 8 es una nota de octavo de tiempo

int nivelActual = 0;

Se inicia en 0 y representa el nivel actual de jugador en el juego. A medida que el jugador acierta, esta variable incrementa y va aumentando la dificultad



Es un valor que controla la rapidez con la que se muestran las secuencias de luces en el juego. Un valor de 500 indica un intervalo de 500 milisegundo entre cada paso de la secuencia. la velocidad puede ajustarse a medida que el jugador progresa.

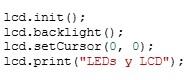


La línea int semilla = analogRead(A5) lee el valor analógico desde el pin A5 del Arduino, que está conectado a una fuente de "ruido". Este ruido es esencialmente que ocurre cuando un pin no está conectado a una fuente de voltaje estable. El valor leído es un número entre 0 y 1023, dependiendo de la variación de voltaje en el pin A5 en ese momento. Esta lectura se usa como una "semilla" para inicializar el generador de números aleatorios.

La función randomSeed(semilla)toma ese valor leído y lo usa para iniciar el generador de números aleatorios del Arduino. Una semilla en este contexto es un valor inicial que se usa para generar una secuencia de números aleatorios.

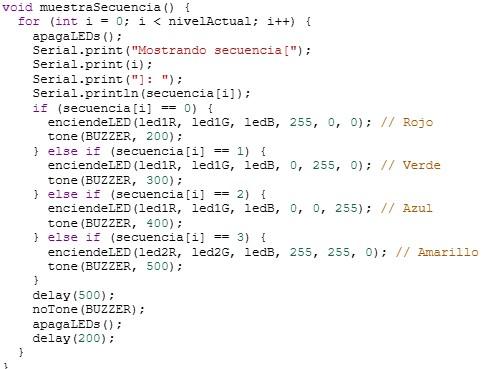
Sin una semilla variable, los números aleatorios generados serían siempre los mismos en cada ejecución del programa, lo cual haría que los resultados sean predecibles.

Al usar una semilla basada en un valor que cambia constantemente, como el de A5, se asegura que cada vez que el programa se ejecute, el generador de números aleatorios produzca una secuencia diferente, Esto nos permite que sea aleatorio las luces , Esto lo pusimos para que no sea aleatorio y no sea repetitivo , asi cada que jugas sea unico , el unico error es que siempre la primera secuencia es igual pero cuando le erras cambia, no sabemos por que pasa esto en nuestro codigo , pero lo importante es que después de eso anda bien. 👍



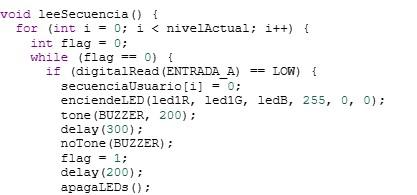
La función lcd.init(); inicia la pantalla LCD, configurandola para que reciba y muestre datos , sin esta inicialización , la pantalla no podría mostrar nada, ya que no sabría como manejar los datos que le mandamos lcd.backlight();enciende la retroiluminación de la pantalla, lo que hace que el texto sea visible. Sin esto se no se podría ver bien , bueno y el lcd.setCursor(0,0) establece la posicion inicial , en este caso es la primera fila y la primera columna (0, 0) es como decirle al LCD donde debe comenzar

cd.print(“LEDs y LCD” ); imprime el texto ya dicho comenzando desde la posición que se estableció con setCursor esto es útil para mostrar mensajes de información o instrucciones al usuario, esto permite que el programa interactúa con el usuario mostrándole informacion importante en la pantalla .



ahora empezamos con el void muestraSecuencia() La función se encarga de mostrar una secuencia de luces y sonidos en funcion de los valores almacenados, comienza con un for que recorre hasta el (nivelActual) primero apaga todos los LEDs utilizando la funcion apagaLEDs().dependiendo del valor en secuencia[i] enciende un LED específico y emite un tono con el buzzer: rojo para 0, verde para 1, azul para 2 y amarillo para 3.

Cada vez que se enciende un LED, se emite un tono correspondiente a esa luz, con una frecuencia que aumenta con cada color. Después de mostrar la luz y el sonido, la función espera medio segundo con delay(500 )apaga el sonido con no tone despues , apaga todos los LEDs y espera 200 milisegundos antes de continuar con la siguiente parte de la secuencia. Esto permite al jugador visualizar y escuchar cada paso de la secuencia que debe replicar y así no es tan difícil y por lo menos se guia un poco .



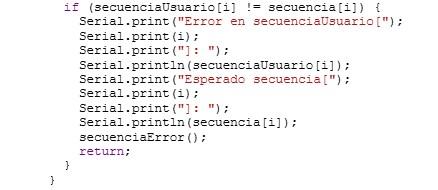
La función leeSecuencia() permite al jugador replicar la secuencia de luces que se muestra en el jeugo , paso a paso , y verificar si la respuesta es correcta. Al inicio, recorre cada elemento de la secuencia, representada por los colores de los LEDs, con un ciclo for . Dentro de cada interacción, la función espera que el jugador presione uno de los botones conectados a los pines

Estos botones están asociados a los colores que corresponden a los valores de la secuencia. Para cada color, la función verifica si el jugador presiona el botón correcto, igual en fisico estan cada uno con su color correspondiente osea si el boton que apretas es de color rojo el que se va a prender es rojo.

Si el jugador presiona el botón correctamente, la función enciende el LED correspondiente al color de la secuencia y emite un tono con el buzzer para confirmar la selección. Luego, la función espera un corto período de tiempo para dar tiempo al jugador a ver la respuesta antes de que continúe.

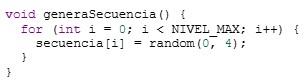
Si el jugador elige un botón incorrecto, se enciende un conjunto de LEDs en señal de error,y se empieza a escuchar la melodiaError() Esto indica al jugador que ha cometido un error en su secuencia, y el juego se reinicia.

En caso de que el jugador presione el botón correcto, la función continúa y pasa al siguiente paso en la secuencia.



La comparación entre las elecciones del jugador (SecuenciaUsuario[i]) y los valores almacenados en la secuencia original (secuencia[i]) se realiza en cada interacción. el juego para verificar si el jugador está siguiendo correctamente la secuencia mostrada por el sistema

Una vez que el jugador haya replicado correctamente toda la secuencia, la función termina y llama a la función secuenciaCorrecta(). Esto aumenta el nivel del jeugo , mostrando una nueva secuencia más larga que el jugador deberá repetir en el siguiente ciclo. La función garantiza que el jugador solo avance si la secuencia ingresada es completamente correcta.

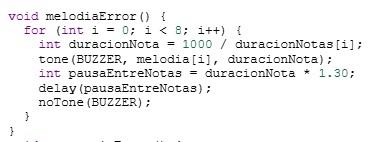


ahora paso a explicar la generarSecuencia(), esto genera una secuencia aleatoria de colores que el jugador debe replicar en el juego. Utiliza un ciclo for que va hasta el valor de Nivel\_Max, que es 50. Dentro del ciclo, se utiliza la función random(0,4)para generar un número aleatorio entre 0 y 3. Cada número corresponde a un color específico:

* 0: Rojo
* 1: Verde
* 2: Azul
* 3: Amarillo

El número generado se guarda en el arreglo secuencia[i] , que almacena la secuencia de colores a medida que se genera. Cada vez que se ejecuta el programa, la secuencia es diferente, lo que hace que el juego sea impredecible.

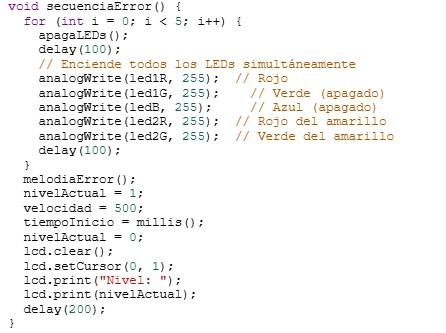
Al final de la función, el arreglo secuencia[] contiene una secuencia aleatoria de colores que el jugador tendrá que seguir en el siguiente paso del juego. La longitud de la secuencia depende del nivel actual del juego, lo que hace que la secuencia sea más larga y desafiante a medida que se avanza.



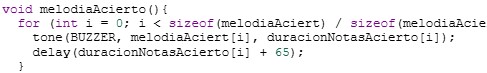
La función melodiaError() se encarga de reproducir una melodía cuando el jugador comete un error en la secuencia del juego. La melodía es un conjunto de notas que se tocan en orden para indicar que ha ocurrido un fallo. En primer lugar, la función recorre un ciclo for que se ejecuta 8 veces, ya que el arreglo melodia[] contiene 8 elementos. Cada elemento en este arreglo es una frecuencia correspondiente a una nota musical.

La duración de cada nota está definida en el arreglo duracionNotas[ ], y la duración de la nota se calcula dividiendo 1000 por el valor en duracionNotas[i] , lo que determina el tiempo que debe durar cada nota. Luego, la función tone() tone() genera el sonido del buzzer con la frecuencia especificada por melodia[i] durante la duración calculada. Después de reproducir la nota, se calcula un pequeño retraso entre las notas, el cual se ajusta con el valor de (pausaEntreNotas) , que es 1.3 veces la duración de la nota. Esto permite que haya una pequeña pausa entre las notas para hacer la melodía más fluida.

Al finalizar el ciclo, el buzzer deja de sonar con la función notone() y la melodía de error ha sido reproducida, indicando que el jugador ha cometido un error.



La función secuenciaError() se ejecuta cuando el jugador comete un error al intentar reproducir la secuencia de luces. Primero, apaga todos los LEDs y luego los enciende en todos los colores (rojo, verde, azul y amarillo) de forma simultánea, haciendo que parpadeen varias veces para indicar el error. Esto ocurre cinco veces, con una pausa de 100 milisegundos entre cada parpadeo. A continuación, se reproduce una melodía de error usando el buzzer, lo que proporciona una retroalimentación auditiva del fallo. Luego, el nivel del juego se restablece a 1,



Está diseñada para reproducir una melodía de éxito usando un buzzer, indicando que el jugador ha ingresado correctamente la secuencia de luces.

Primero, la función utiliza un bucle for para recorrer las notas que conforman la melodía. El número de notas es determinado por el tamaño del arreglo melodiaAciert[], calculado con sizeof(melodiaAciert) / sizeof(melodiaAciert[0]).

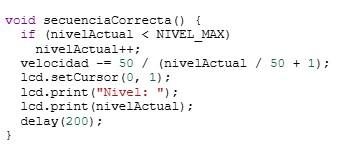
Dentro del bucle, se utiliza la función tone() para reproducir cada nota de la melodía. El primer parámetro es el pin donde está conectado el buzzer, el segundo es la frecuencia de la nota a emitir, tomada de melodiaAciert[], y el tercer parámetro es la duración de esa nota, tomada del arreglo duracionNotasAcierto[]. Esto asegura que cada nota suene durante el tiempo adecuado.

Después de cada nota, se usa un delay() para establecer una pausa entre ellas. El tiempo de la pausa es igual a la duración de la nota más 65 milisegundos adicionales, lo que asegura una separación clara entre los tonos.



y la velocidad del juego se restablece a su valor inicial (500). También se actualiza el tiempo de inicio para reiniciar el conteo de tiempo del juego.

Finalmente, se limpia la pantalla LCD, mostrando el texto "Nivel: 0", lo que indica que el jugador comienza nuevamente desde el primer nivel. Después de un breve retraso, el juego vuelve a estar listo para que el jugador intente nuevamente la secuencia desde el inicio. Esta función proporciona una clara retroalimentación visual y auditiva al jugador y restablece el juego para un nuevo intento.



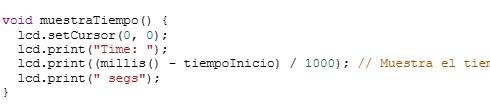
La función secuenciaCorrecta() se ejecuta cuando el jugador ha replicado correctamente la secuencia de colores del juego.

Su propósito es gestionar el avance al siguiente nivel después de que el jugador haya completado correctamente una secuencia. Incremento de nivel: Si el jugador ha completado correctamente la secuencia y el nivel actual es menor que el máximo (Nivel\_Max), el nivel se incrementa en uno.

Esto significa que el jugador se enfrentará a una secuencia más larga y desafiante en el siguiente ciclo. Ajuste de velocidad: La velocidad del juego (el tiempo entre la aparición de los LEDs) se ajusta para que el juego se haga más difícil a medida que el jugador avanza. La fórmula velocidad -= 50 / (nivelActual / 50 + 1); esto disminuye la velocidad gradualmente conforme el nivel aumenta, haciendo que los tiempos entre los colores de la secuencia se reduzcan. Actualización en la pantalla LCD: La función también actualiza la pantalla LCD para mostrar el

nuevo nivel al jugador. Esto se hace mediante lcd.setCursor(0,1) y lcd.print (“”) lcd.setCursor(0, 1) y lcd.print(“Nivel:”)para mostrar el nivel actual en la segunda línea de la pantalla.

Después de actualizar el nivel en la pantalla, se coloca un pequeño retraso de 200 milisegundos con delay(200) para permitir que el jugador vea el nivel actualizado antes de que la secuencia del siguiente nivel comience.



La función muestraTiempo() se utiliza para mostrar en la pantalla LCD el tiempo que ha transcurrido desde que comenzó el juego.

Su principal objetivo es proporcionar al jugador una referencia del tiempo durante el juego. Cada vez que se llama a esta función, calcula el tiempo en segundos desde el inicio y lo muestra en la pantalla LCD.

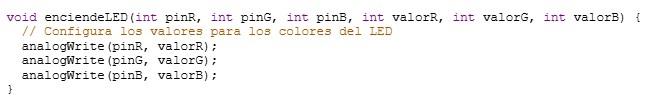
El tiempo se calcula restando el valor almacenado en tiempoInicio del valor actual de millis(), que representa el número de milisegundos desde que se encendió la placa. Esta diferencia nos da el tiempo transcurrido en milisegundos, y al dividirlo por 1000, lo convertimos a segundos.

El valor calculado se imprime en la primera línea de la pantalla LCD, puesto como la palabra

"Time: ". Esto permite que el jugador vea de manera continua cuánto tiempo ha estado jugando, lo que es útil para seguir su progreso y ayudar a crear una sensación de urgencia o desafío.

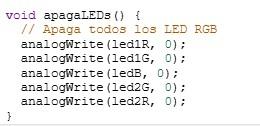
La función se ejecuta repetidamente dentro del ciclo principal del programa para mantener el tiempo actualizado en la pantalla mientras el jugador sigue jugando. Así, cada vez que la pantalla se actualiza, el jugador tiene una visualización en tiempo real del tiempo transcurrido desde el inicio del juego.

Este indicador de tiempo es esencial en muchos juegos para monitorear la duración de las rondas o para crear desafíos basados en el tiempo. Además, el tiempo mostrado se actualiza de forma constante.



Las funciones enciendeLED() y apagaLEDs() son cruciales en el funcionamiento visual del juego, ya que controlan los LEDs RGB conectados al sistema. Los LEDs RGB permiten crear una variedad de colores combinando tres colores básicos: rojo, verde y azul. A través de estas funciones, se logra que el jugador vea una secuencia visual que debe replicar en el juego.

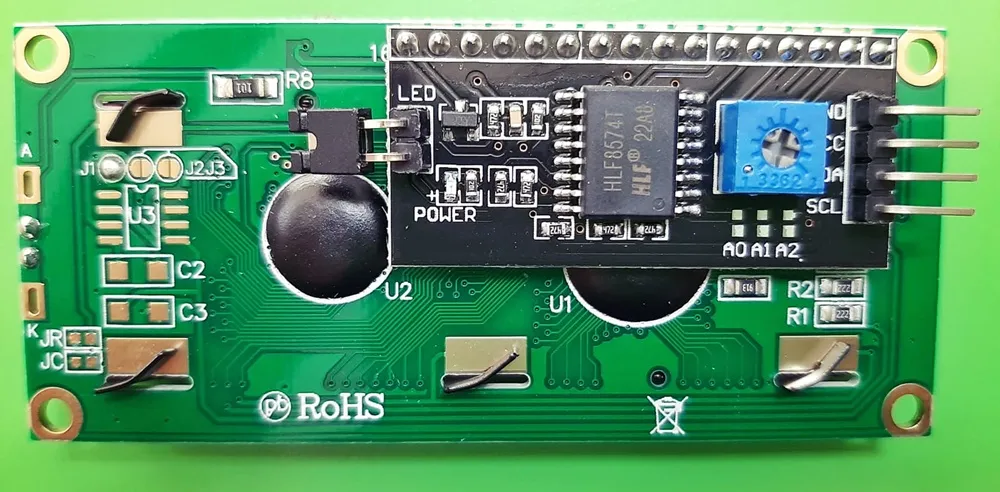
La función enciendeLED() se utiliza para encender un LED en un color específico, y para ello toma seis parámetros: los pines correspondientes a los colores rojo, verde y azul del LED, así como los valores que determinan la intensidad de cada color. La función analogWrite() se emplea para controlar la intensidad de cada color, permitiendo que el LED se ilumine con la mezcla de los colores especificados. Se puede obtener cualquier color en el espectro visible dependiendo de los valores que se asignen a los colores.



Por otro lado, la función apagaLEDs() se encarga de apagar todos los LEDs del sistema. Esto se logra ajustando el valor de los pines correspondientes a los colores de los LEDs a 0, lo que hace que el LED se apague completamente. Esta función es importante porque asegura que no haya confusión o distracción visual durante el juego cuando no se necesita ningún color encendido.

En el contexto del juego, la función enciendeLED() se utiliza para mostrar los colores de la secuencia que el jugador debe memorizar y reproducir. Cada vez que el juego genera una nueva secuencia, los LEDs se encienden con colores específicos que corresponden a las posiciones de la secuencia.

Esto sirve como una guía visual para el jugador. Después de que el LED se enciende y se muestra por un corto tiempo, se apaga con la función apagaLEDs ()

El ciclo de encender y apagar los LEDs no solo es importante para que el jugador pueda ver y recordar la secuencia, sino que también está sincronizado con el buzzer, que emite un sonido diferente según el color que se muestra. De esta forma, el jugador tiene memoria para recordar la secuencia. Bueno esto vendría hacer una explicación resumida del código si lo mando todo se hace muy largo. ya con esta explicacion, voy a pasar hablar de lo último que hice, nos pasó que un martes habíamos hecho todas las conexiones y no nos andaba bien a lo último vino el profe y YO había hecho una conexión mal o varias entonces el jueves decidí quedarme hasta las 4 de la tarde viendo si funcionaba hice 

todo , descargue una biblioteca para el lcd en arduino y , finalmente funciono , lo unico que anda mal ahora viene a ser el buzzer.

Porque es como que se escucha la melodía de fondo pero se escucha mas una interferencia que aturde y otra que la termine arreglando es la del lcd , anda bien pero vi que el mio tenia brillo bajito y vi el de otro grupo y era más iluminado resulta que en la parte de atrás al lcd IC2 hay 2 pines hembras que hay que conectarlas mutuamente y con un destornillador girar en la parte de adentro en la cajita azul como se puede ver en la imagen a la izquierda del todo se puede ver las hembras. . 🙂