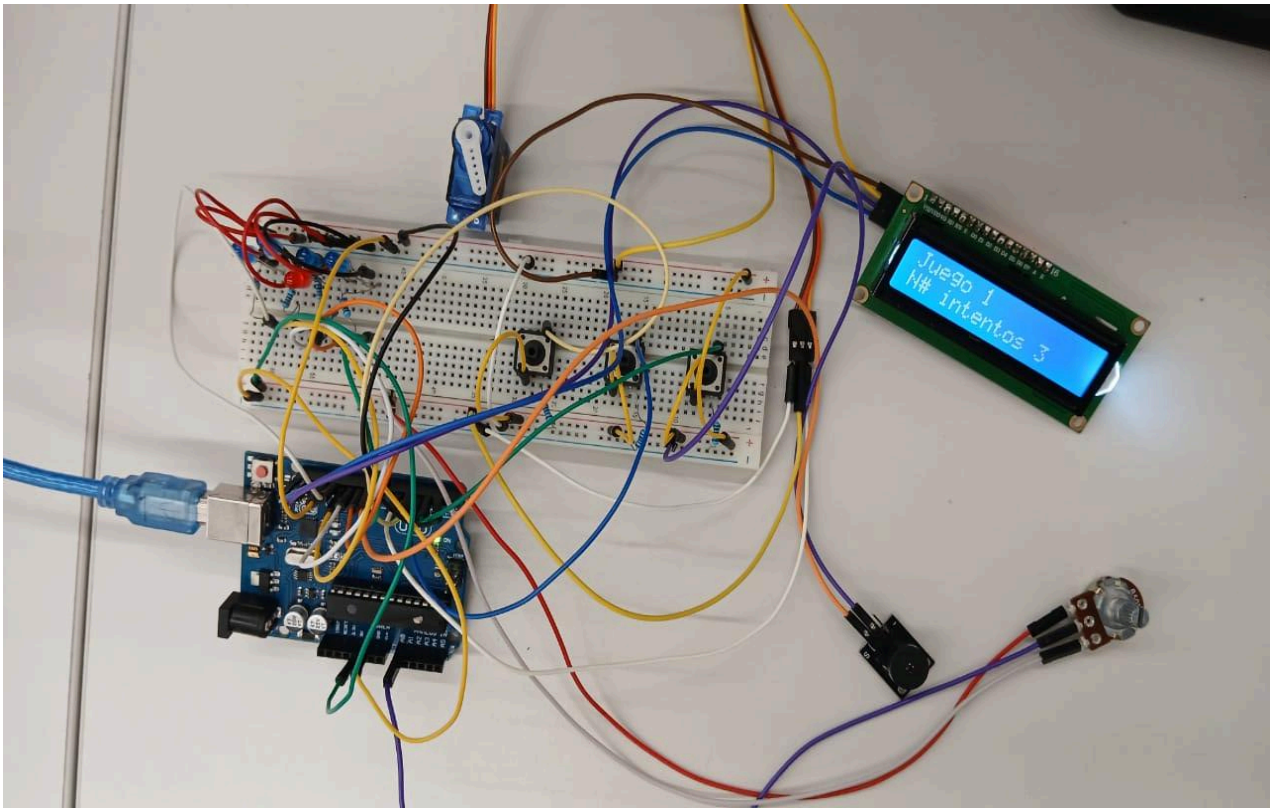
Grupo 2
Juego de reacción y Juego de secuencia.

Integrantes: Zarate Alaciel, Ferrando Florencia, Alfonzo Tamara, Quiroga Daiana



Materia: Laboratorio de Computación 1.

Docentes: Matias Jose Gagliardo, Pedro Facundo Iriso.

2025

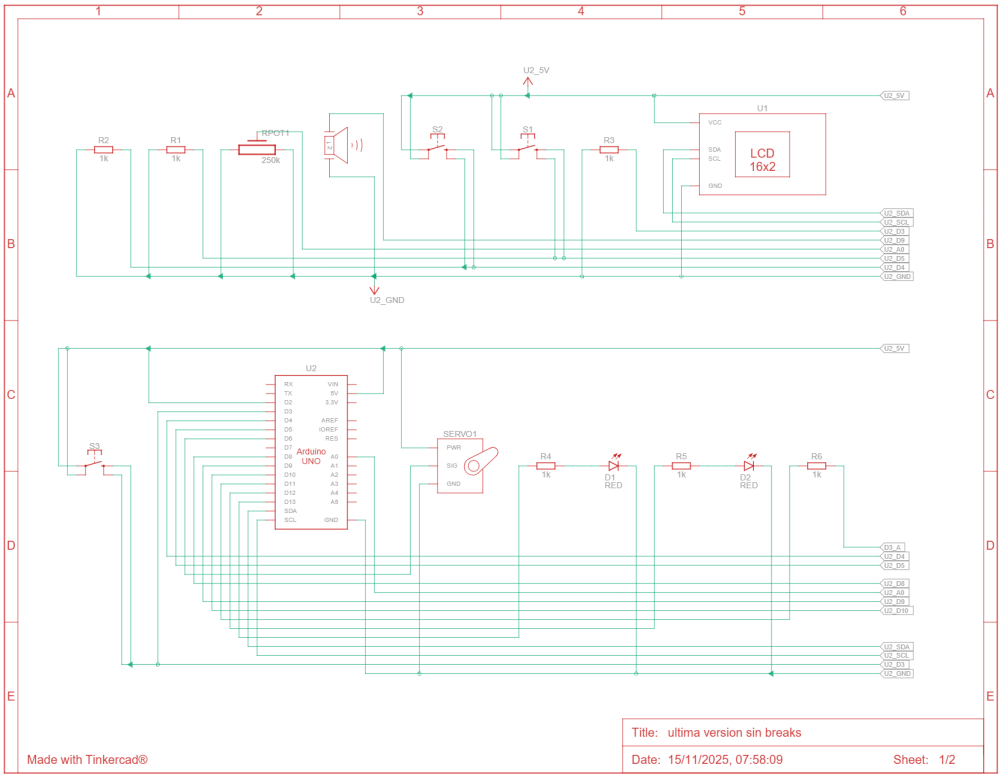
Resumen del proyecto:

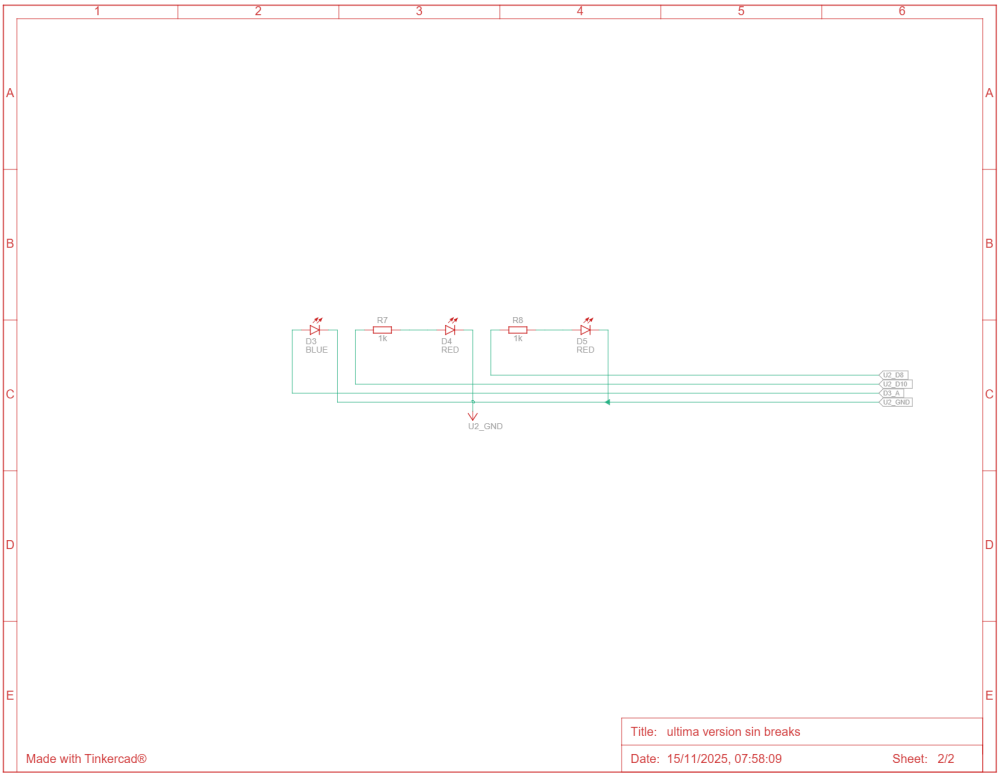
Este sistema de Arduino propone 2 juegos de entretenimiento mediante un sistema de luces leds y botones. El primer juego consta de 5 luces, para desafiar los reflejos del usuario estas luces se prenden y apagan en secuencia, el objetivo es tocar el botón cuando la luz del medio se encuentre encendida, en el momento exacto. El usuario cuenta con 3 intentos para lograrlo y avanzar al 2do juego. Cada intento fallido añade dificultad a los intentos posteriores.
El segundo juego es similar a uno de "simón dice", el usuario deberá replicar tres secuencias de luces diferentes, repitiendo cada patrón al pulsar los botones correspondientes, cada secuencia añade una led. El usuario cuenta con 2 intentos para lograrlo. Puede regularse la dificultad de este último con el potenciómetro.
Al ganar ambos juegos, se activa un servo con un premio.

Descripción técnica del hardware:

1	PCF8574-based, 39 (0x27) LCD 16 x 2 (I2C)
1	Arduino Uno R3
4	Red LED
1	Blue LED
1	Buzzer
1	Potenciómetro 250 kΩ
3	Botones
8	Resistencias 1 kΩ
1	Servo
32	Cables

Diagrama esquemático o conexiones:



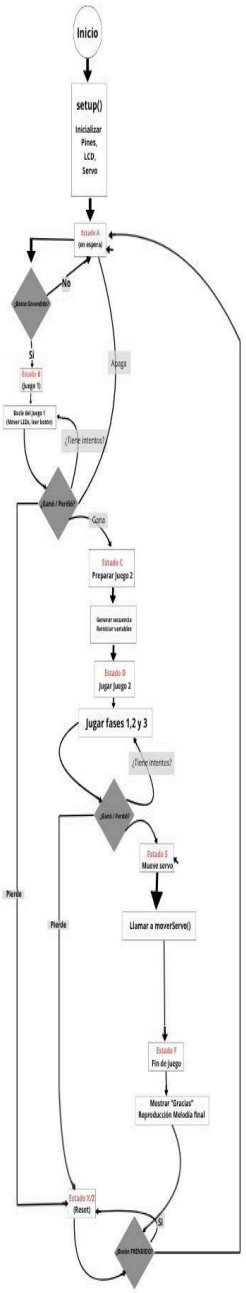


Descripción del software y su estructura:

Este código implementa juegos de reacción con un display, leds, botones, un potenciómetro, un buzzer y un servo, estructurado en una máquina de estados para gestionar los diferentes momentos del juego.
Librerías: LiquidCrystal_I2C (para el display) y Servo (para el servo).
Pines: Definidos para leds (13, 12, 11, 10, 8), botones (PRENDIDO, BTN1, BTN2, BTN3), servo (PIN_SERVO), potenciómetro (POTENCIOMETRO) y buzzer (BUZZER).
Variables: Almacenan el estado de los leds, velocidad del juego, intentos, estados de los botones, secuencias aleatorias para el juego 2, y el estado general del juego.
Funciones Principales:
setup: Inicializa el puerto serial, configura todos los pines (leds como OUTPUT, botones como INPUT), inicializa el display, el servo (a 0°), el generador de números aleatorios (randomSeed) y el buzzer.
todosApagados/controladorLeds: Funciones auxiliares para gestionar el encendido y apagado de los leds.
limpiarPantalla: Limpia el display cuando la variable limpiar es true.
generarSecuenciaAleatoria: Crea secuencias de leds aleatorias para el Juego 2.
mostrarSecuencia: Enciende y apaga leds siguiendo una secuencia. La velocidad de la secuencia es ajustable mediante el potenciómetro.
leerSecuencia: Espera la entrada del usuario a través de los botones, guardando la secuencia que presiona.
Melodías (melodiaVictoria, melodiaDerrota, melodiaFinal): Reproducen sonidos a través del buzzer para indicar victoria, derrota o el final del juego.
moverServo: Controla el servo en una secuencia deseada (a 90°, espera, a 0°). Está implementada como una máquina de estados usando millis().
Flujo del Juego (Máquina de Estados en loop):
Estado 'A' Inicio/Espera: Espera a que se presione el botón PRENDIDO.
Estado 'B' Juego 1 - "apreta el led":
Los leds se encienden y apagan uno por uno. El jugador debe apretar BTN1 cuando el led objetivo (pin 11) está encendido.
Si presiona bien, gana y pasa al Juego 2.
Si falla, pierde un intento y la velocidad de los leds aumenta.
Si se quedan sin intentos, pierde el juego y pasa al estado de derrota 'X'.
Estado 'C' (Inicio Juego 2): reinicia variables y genera las secuencias aleatorias para las tres fases del Juego 2.
Estado 'D' (Juego 2):
El juego consta de tres fases con secuencias de leds que van aumentando (3, 4 y 5 leds).
En cada fase, mostrarSecuencia reproduce una secuencia y leerSecuencia espera la entrada del jugador.
Si el jugador falla, pierde un intento y tiene una segunda oportunidad en la misma fase.
Si el jugador queda sin intentos, pierde el juego.
Si completa las tres fases, gana y pasa al estado servo ('S').
Estado 'S' (Control del Servo): Ejecuta la función moverServo para mover el servo a 90°.
Estado 'F' (Final): Muestra un mensaje final, reproduce una melodía de victoria y pasa al estado 'Z'

Estado 'X' (Derrota): Muestra un mensaje de derrota, reproduce una melodía y pasa a 'default'.
Estado 'Z' (Reinicio): Permite al usuario reiniciar el juego.
Default: Permite al usuario reiniciar el juego.

Diagrama de máquina de estados:



Descripción del contador de flancos y control por tiempo:

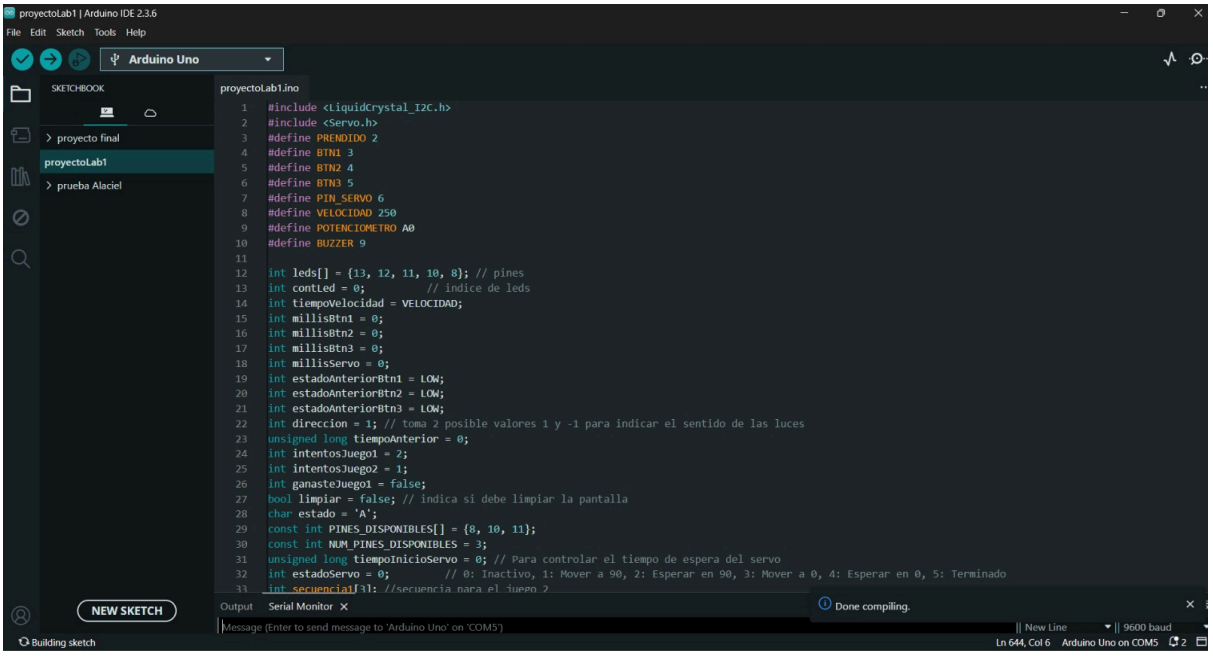
El código cuenta con un contador de flancos principalmente definido con las variables “estadoBtn” más el número correspondiente a cada uno, donde se verifica si un botón esta

presionado; la variable “tiempoActual” junto a “milisBtn” (más el número correspondiente al botón), calcula durante cuánto tiempo se presiona cada uno.
Este proceso detecta cuando se presiona un botón, omitiendo los rebotes.

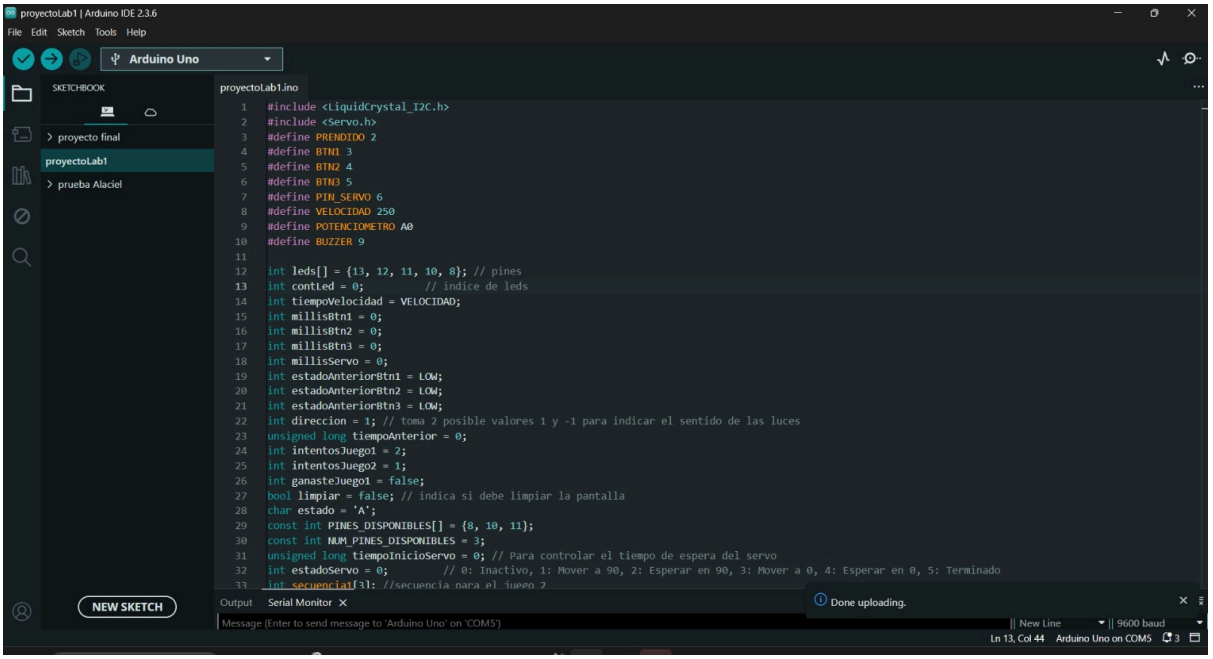
```
if (estadoBtn1 == HIGH && estadoAnteriorBtn1 == LOW)
                {
                        if (tiempoActual - millisBtn1 > 10)
```

Capturas del sistema funcionando:

sistema compilado:



sistema ejecutado:



Mediante el siguiente enlace pueden visualizarse videos del sistema, cada elemento del mismo y ejemplos de su funcionamiento:
https://drive.google.com/drive/folders/12KCKWD86lNKBZqg8CSxYaXRbglvW2ETn

Conclusiones del grupo:

Aprendimos sobre funciones y el alcance del Arduino Uno, adquirimos nuevos conocimientos sobre electrónica y programación, diseño, planificación y conexión de elementos del Arduino, de plataformas como GitHub y Tinkercad, junto a sus usos básicos y funciones útiles.

Reforzamos nuestros conocimientos y metodologías para investigar y redactar. Como primer proyecto relacionado a la carrera que queremos seguir, nos llevamos una enseñanza memorable, con nuevas habilidades de comunicación y manejo en equipo.