Traductor de Código Morse con Arduino (LCD I2C + Buzzer + Pulsador)

Trabajo Práctico Final – Laboratorio de Computación I (UNSAM)

Grupo: 8 • Integrantes: Matias Mantiñan – Juan Fornes - Santino Pelle – Diego Mammana • Docentes: Matias Jose Gagliardo y Pedro Facundo Iriso

Resumen del proyecto

Se implementa un sistema basado en Arduino UNO que recibe entradas humanas mediante un pulsador y traduce dichas pulsaciones temporizadas al alfabeto Morse, reproduciendo puntos y rayas mediante un buzzer y mostrando el carácter reconocido en una pantalla LCD 16×2 vía I2C. El sistema opera sin demoras bloqueantes, empleando una máquina de estados de dos estados (UP/DOWN), un contador de flancos y control por tiempos para clasificar las pulsaciones (punto/raya) y los silencios (fin de letra / palabra).

Descripción técnica del hardware

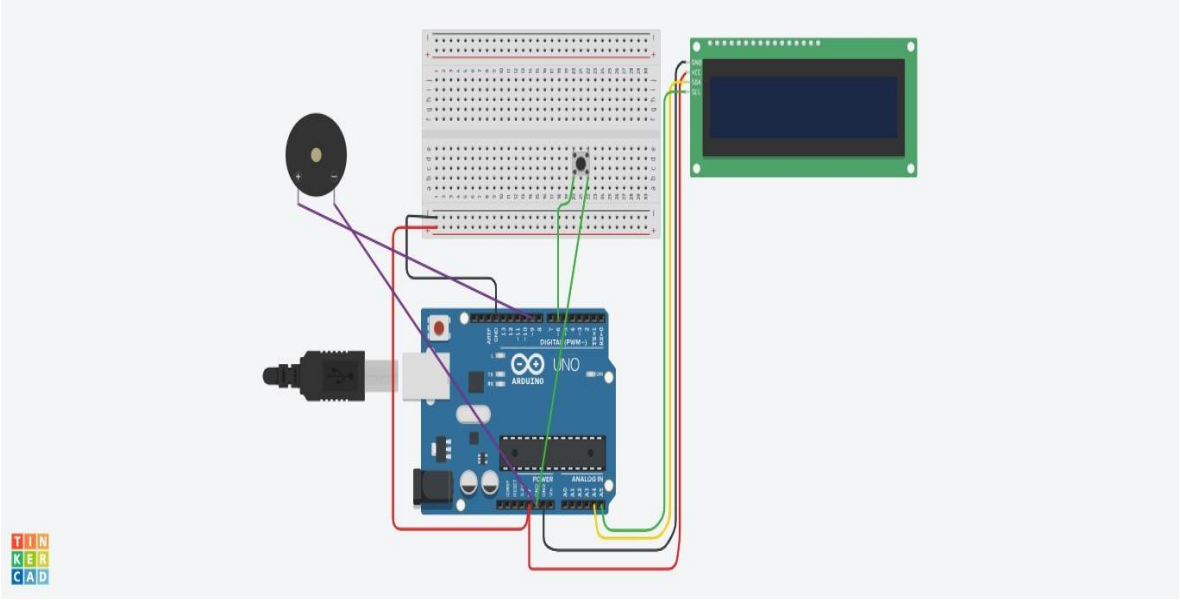
Placa: Arduino UNO R3.

Entradas: 1 pulsador en D6 con resistencia de pull-up interna (INPUT_PULLUP).

Salidas: 1 buzzer piezoeléctrico en D9 controlado con *tone()*; LCD 16×2 I2C (VCC, GND, SDA→A4, SCL→A5). Se comparte masa (GND) para asegurar referencia común entre módulos. La elección de A4/A5 se debe a que en UNO son los pines de hardware I2C (SDA/SCL).

Componente	Pin Arduino	Motivo
Pulsador (una pata a GND, la opuesta a señal) D6 (INPUT_PULLUP)	6	Entrada estable: 1 en reposo, 0 al presionar (a GND).
Buzzer (+)	D9 (PWM)	Salida de tono/feedback sonoro con tone().
Buzzer (–)	GND	Retorno común.
LCD I2C VCC	5V	Alimentación del módulo.
LCD I2C GND	GND	Referencia común.
LCD I2C SDA	A4	Datos I2C (SDA) – hardware UNO.
LCD I2C SCL	A5	Reloj I2C (SCL) – hardware UNO.

Diagrama esquemático / conexiones



Descripción del software y su estructura

El código se organiza en:

(1) Definiciones de tiempo (DOT_DURACION, TOLERANCIA, gaps de 1t/3t/7t),

(2) Entradas/Salidas (BOTON_PIN, BUZZER_PIN, LCD I2C),

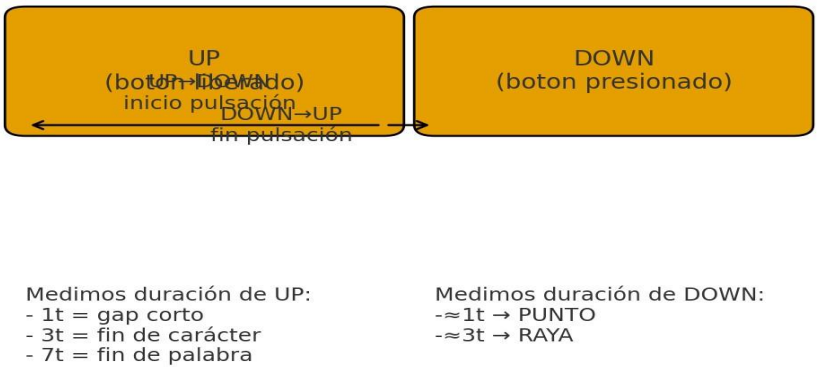
(3) Tabla del alfabeto Morse,

(4) Máquina de estados con *enum Estado {UP, DOWN}*, medida de tiempos por *millis()*,

(5) Funciones de utilidad: limpieza de buffer, clasificación punto/raya, decodificación de letra,

(6) Loop principal que detecta flancos, reproduce tono durante DOWN y escribe en LCD al cerrar letras/palabras.

Máquina de estados



Estados:

UP (botón liberado) y **DOWN** (botón presionado). En cada transición se mide la duración del estado previo: si la duración de *DOWN* ≈ 1t se interpreta *PUNTO*, si ≈ 3t se interpreta *RAYA*. Si la duración de *UP* ≈ 3t se cierra el *carácter* y se imprime; si ≈ 7t se agrega *espacio* (palabra). Se utiliza *INPUT_PULLUP* para lectura robusta del botón y *tone()* para feedback sonoro.

Contador de flancos y control por tiempo

El **contador de flancos** se implementa al comparar el estado leído con el estado actual del sistema: cuando *estado_nuevo != estado_actual* se detecta un cambio (flanco). En flanco **DOWN→UP** se almacena la duración de la pulsación (*down_duracion*) y se clasifica como punto o raya. En flanco **UP→DOWN** se evalúa la duración del silencio previo para decidir si corresponde cerrar la letra (3t) o palabra (7t). El **control por tiempo** parametriza *DOT_DURACION* y *TOLERANCIA*, permitiendo ajustar la velocidad del sistema.

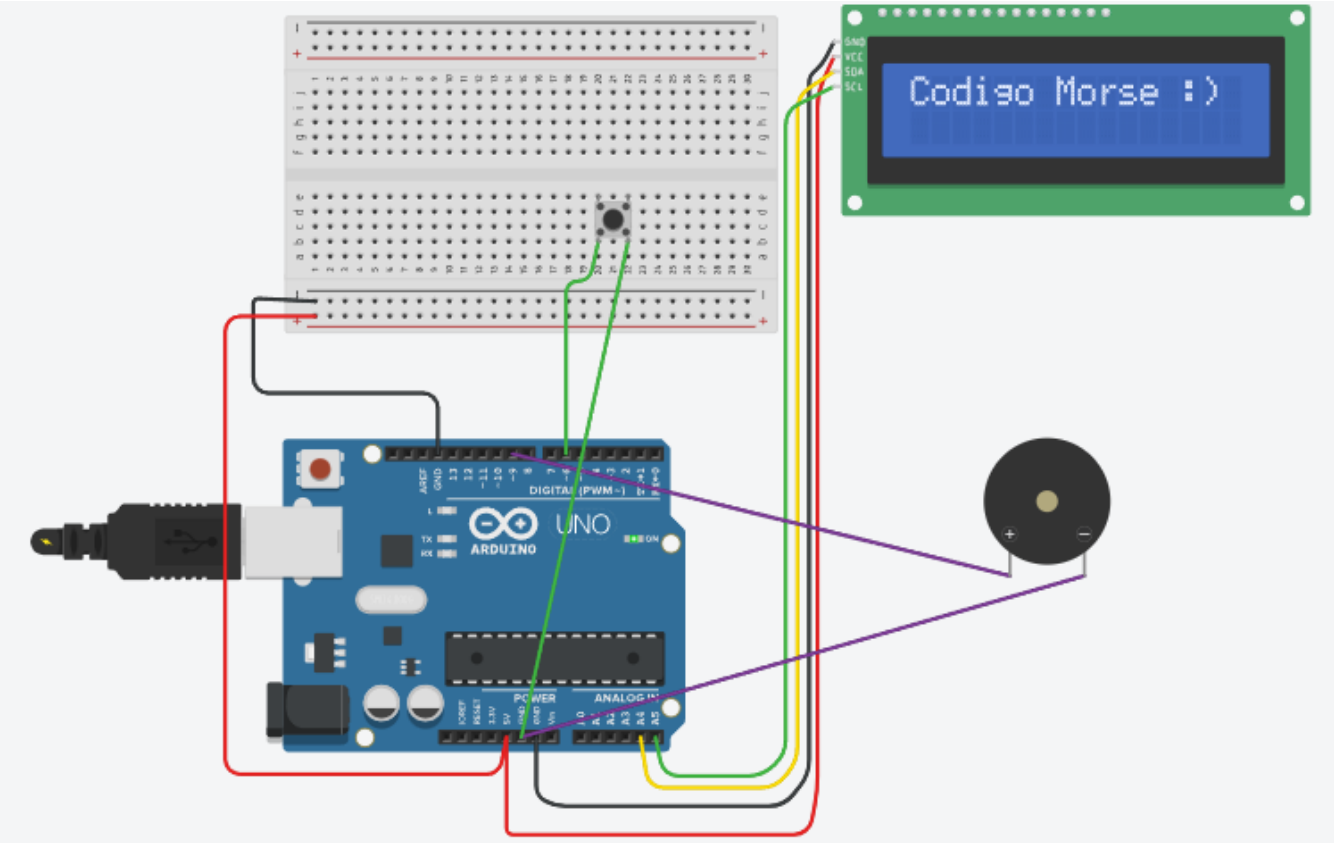
Apéndice A – Código fuente (versión de grupo)

```
#include <LiquidCrystal_I2C.h>

// Definiciones de tiempos de Morse

const unsigned long DOT_DURACION = 300
        DOT, DASH, DOT, CLEAR},
        {'G', DASH, DASH, DOT, CLEAR, CLEAR},
        {'H', DOT, DOT, DOT, DOT, CLEAR},
        TOLERANCIA && down_duracion <= DASH_DURACION + TOLERANCIA) {        morse_actual[morse_i++] = DASH;     }
        up_duracion = duracion;
        if (up_duracion >= LONG_GAP - TOLERANCIA) {
        lcd.print(
```

Capturas del sistema funcionando



Conclusiones del grupo

El sistema cumple los requisitos mínimos de la cátedra: control de múltiples E/S, contador de flancos, control lógico por tiempo y máquina de estados.

El enfoque por tiempos parametrizables y tolerancias permitió una decodificación robusta del Morse ingresado manualmente. Como mejora futura se propone:

(1) potenciómetro en A0 para ajustar la velocidad (DOT_DURACION),

(2) almacenamiento del texto en EEPROM o envío por Serial,

(3) soporte de números y símbolos adicionales,

(4) debounce por software con ventana temporal.