



Tradutor de código morse

Haría: Especialidad en Computación,

Establecimientos: Escuela Técnica 32 "General José de San Martín"

Año: 4to3ra

Materias: Proyecto Informático

Profesor: Gonzalo N. Consorti

Integrantes: Leonard Rubio - Marcos Torres - Jimena Aysana

## Descripción general

El objetivo es desarrollar un sistema que permita a los usuarios ingresar texto y recibir la traducción en código Morse a través de señales visuales (LED) o sonoras (buzzer).y una comunicación entre dos equipos a través de bluetooth.

## Las herramientas utilizadas en el proyecto son:

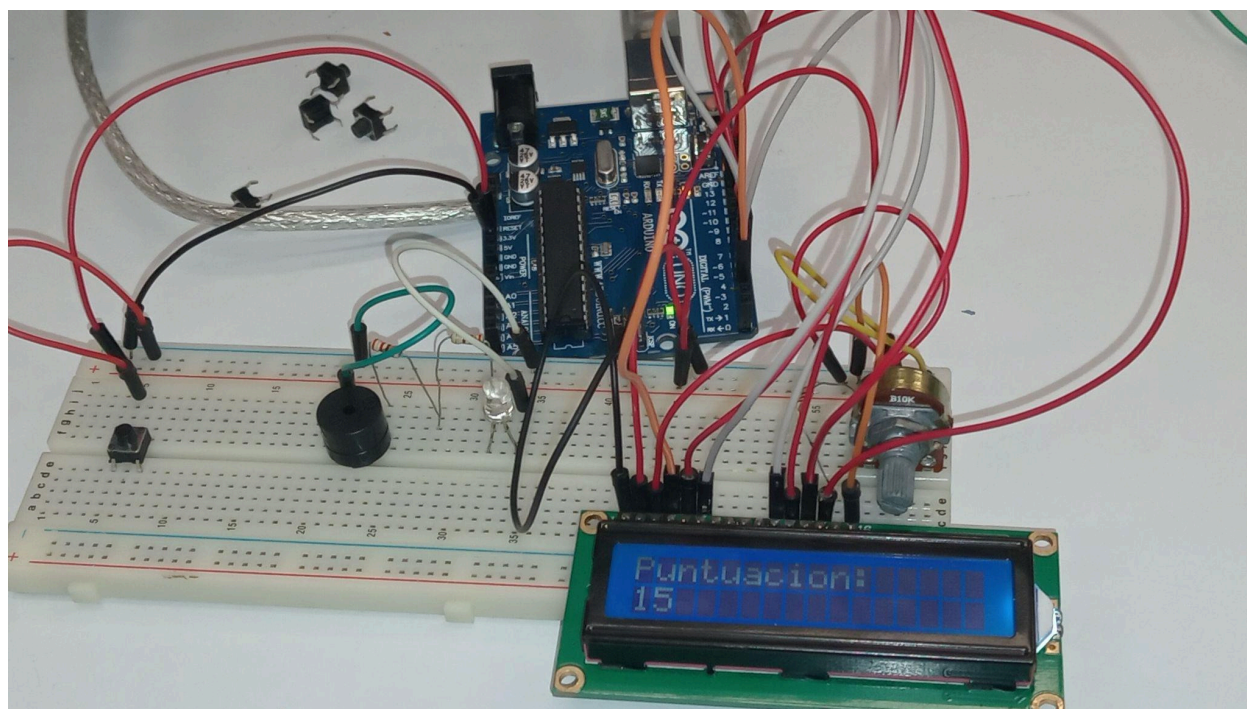
- (2) Arduino Uno R3
- (2) Cable USB para programar
- (2) LCD 16x2
- (2) Potentiometer
- (2) Piezo
- (4) Pushbutton
- (10) Resistencia 220 ohms
- (2) módulo bluetooth HC-05
- (2) LED
- (2) Protoboard

## El primer día:

Me dediqué a sentarme y analizar cómo conectar todos los componentes necesarios para el proyecto. Revisé los trabajos prácticos anteriores para verificar si en alguno de ellos había algo

que pudiera necesitar. Afortunadamente, encontré el TP en el que habíamos trabajado con la pantalla LCD, lo cual resultó ser de gran utilidad.

Recordé que una de las primeras cosas que probamos en clase fue que, al presionar el botón, el LED y el piezoeléctrico responden correctamente. Ese sistema sigue hasta hoy me dediqué a sentarme y analizar cómo conectar todos los componentes necesarios para el proyecto. Revisé los trabajos prácticos (TP) anteriores para verificar si en alguno de ellos había algo que pudiera necesitar. Afortunadamente, encontré el TP en el que habíamos trabajado con la pantalla LCD, lo cual resultó ser de gran utilidad.



Recordé que una de las primeras cosas que probamos en clase fue que, al presionar el botón, el LED y el piezoeléctrico responden correctamente. Ese sistema sigue siendo parte del proyecto incluso en esta etapa tan avanzada. Luego, me enfoqué en iniciar la conexión de la

pantalla LED. Sin embargo, debo admitir que los cables complicaron bastante la tarea, dificultando que todo quedara ordenado y funcional.

Después de realizar estas conexiones, comencé a trabajar en las modificaciones del código. Esta parte también representó un desafío, ya que tuve que ajustar varios detalles para garantizar que cada componente respondiera correctamente según lo planeado. lastimosamente el código que cree no fue el definitivo.

### Librerías y Pines:

`#include <LiquidCrystal.h>`: Esta línea incluye una biblioteca que permite controlar pantallas LCD, que son dispositivos que muestran información visualmente.

`#define`: Estas líneas asignan nombres a los pines del microcontrolador que se utilizan para conectar botones, un LED y un buzzer (un pequeño altavoz). También se definen dos tiempos: uno para un "punto" y otro para una "raya" en el código Morse.

### Inicialización de la Pantalla LCD:

`LiquidCrystal lcd(7, 8, 9, 10, 11, 12);`: Aquí se inicializa la pantalla LCD especificando cuáles pines del microcontrolador están conectados a ella.

`lcd.begin(16, 2);`: Establece que la pantalla tiene 16 columnas y 2 filas, lo que significa que puede mostrar hasta 32 caracteres a la vez.

### Estructura y Alfabeto Morse

`struct MorseCode`: Esta parte define una estructura que contiene un par de valores: el código Morse (como un punto o una raya) y la letra correspondiente en el alfabeto.

`morseAlphabet[]`: Es un listado de todos los pares de códigos Morse y sus letras asociadas. Por ejemplo, "."- representa la letra "A".

### Funciones Principales

#### `setup()`:

En esta función se configuran los pines como entradas o salidas. También muestra un mensaje inicial en la pantalla LCD que dice "Código Morse" y luego limpia la pantalla.

### loop():

Esta es la función principal que se ejecuta repetidamente. Llama a dos funciones:

**handleMorseInput()**: Captura las señales del código Morse (los puntos y las rayas).

**handleConfirmLetter()**: Confirma y traduce el código Morse ingresado a una letra.

### Funciones Secundarias

#### **handleMorseInput()**:

Monitorea el botón de entrada. Cuando se presiona, mide cuánto tiempo se mantiene presionado:

Si es breve (menos tiempo que el definido para un punto), añade un "." al código Morse.

Si es más largo (pero no tanto como para ser una raya), añade un "-".

Luego hace una pausa para evitar lecturas rápidas consecutivas.

#### **handleConfirmLetter()**:

Monitorea otro botón que confirma la letra ingresada. Cuando se presiona:

Traduce el código Morse actual a una letra usando la función **morseToChar(morseInput)**.

Muestra la letra en la pantalla LCD.

Hace parpadear un LED como señal visual de confirmación.

Limpia el código Morse actual para permitir la entrada de un nuevo carácter.

#### **morseToChar(String morse):**

Esta función busca en el listado de códigos Morse para encontrar cuál coincide con el ingresado.

Si encuentra una coincidencia, devuelve la letra correspondiente; si no, devuelve un signo de interrogación ("?") como indicador de error.

### Funcionamiento del Sistema

**Ingreso del Código Morse:** El usuario presiona el botón correspondiente. Según cuánto tiempo lo mantenga presionado:

Un pulso breve se interpreta como un punto (.)

Un pulso largo se interpreta como una raya (-)

Confirmación de la Letra: Una vez que el usuario ha terminado de ingresar un carácter, presiona otro botón para confirmar. El sistema traduce el código ingresado a una letra usando el alfabeto Morse y lo muestra en la pantalla LCD.

Retroalimentación Visual: El LED parpadea para indicar que el sistema ha procesado la letra.

Reinicio para Próxima Entrada: El código Morse actual se borra para que el usuario pueda ingresar una nueva letra.

Este sistema es útil para aprender y practicar el código Morse de manera interactiva y visual.

Fuentes:

[LiquidCrystal Library Documentation](#)

[LiquidCrystal Library Functions Tour](#)

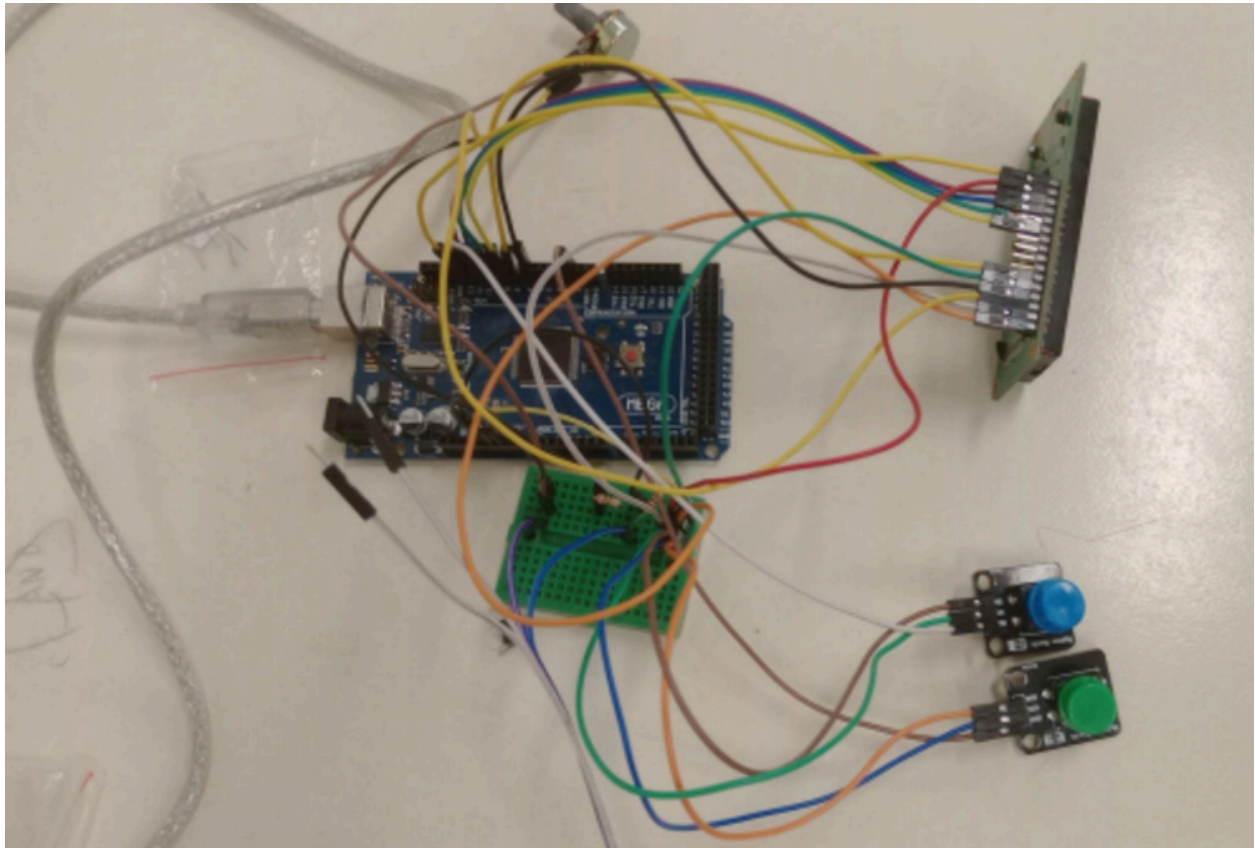
[Arduino PulseIn Reference](#)

### El segundo Arduino:

El segundo Arduino resultó ser una fuente de frustración debido a que la caja de componentes solo contenía los materiales necesarios para un solo dispositivo, y esto no siempre era suficiente. Esta limitación se convirtió en un desafío considerable, ya que cada vez que necesitábamos realizar pruebas o modificaciones, nos veíamos obligados a buscar los componentes necesarios de manera apresurada. Esta presión constante para encontrar lo que necesitábamos generó varios contratiempos que afectaron nuestro progreso.

Además, al igual que en el primer Arduino, los cables se convirtieron en un problema recurrente. La protoboard que teníamos a nuestra disposición era demasiado pequeña para acomodar todas las conexiones requeridas, lo que complicó aún más la organización del circuito. La falta de espacio hizo que los cables se enredan y se cruzaran, lo que aumentaba el riesgo de errores en las conexiones. En un intento por solucionar este problema, incluso tuve que recurrir a llevar cinta adhesiva para asegurar los cables en su lugar y evitar que se movieran accidentalmente durante las pruebas.






### La conexión bluetooth:

Esta fue la parte más difícil del proyecto. El principal desafío fue que en Tinkercad, la herramienta de simulación que utilizamos para probar el diseño y la programación, no está disponible el módulo Bluetooth HC-05, lo que limitó nuestras posibilidades de trabajar con él de manera virtual. Esto significaba que solo podíamos usar el módulo físicamente durante las sesiones de clase, lo que nos obligaba a ser extremadamente organizados y eficientes.

Antes de cada clase, era crucial preparar el código de antemano para aprovechar al máximo el tiempo con el hardware. Esto incluía investigar, planificar y escribir los fragmentos de código necesarios para las pruebas. Una vez en clase, usábamos el módulo HC-05 para



conectar el sistema y verificar si la programación funcionaba según lo esperado. Este proceso requería ajustar rápidamente los errores o realizar modificaciones sobre la marcha, lo que hacía que cada sesión fuera intensa y desafiante.

La fuente principal que utilicé para codificar fue una guía específica sobre cómo configurar y utilizar el módulo HC-05 con Arduino. Esta fuente explicó cómo establecer la comunicación entre el módulo y el microcontrolador, y cómo enviar y recibir datos utilizando comandos AT y la biblioteca [SoftwareSerial](#). Además, incluía ejemplos básicos que pude adaptar a las necesidades de nuestro proyecto. [módulo bluetooth HC-05](#)