Carpeta de Campo - Proyecto: Traductor de Código Morse con Tinkercad y Arduino

*Integrantes del proyecto: Hernandez Kiara, Tomás Fleitas, Benjamín Korstanje, Figallo Brunella*

******

### Introducción

### Este proyecto tiene como objetivo crear un sistema de comunicación basado en código Morse usando dos Arduinos. Uno de los Arduinos tomará un texto y lo convertirá en señales de código Morse (usando un buzzer y un LED). El otro Arduino recibirá las señales de Morse (por medio de un botón) y las decodifica el texto, mostrándolo en un LCD.

### Objetivos del Proyecto

Conversión de texto a código Morse: El primer Arduino convierte un texto ingresado en código Morse y lo transmite usando un buzzer y un LED.

Conversión de Morse a texto: El segundo Arduino recibe las señales de Morse y las convierte en texto, mostrándolo en un LCD.

Interactividad con un botón: El segundo Arduino recibe pulsaciones del botón, las convierte en Morse y las decodifica en texto.

### División de Tareas

Mi Tarea (Hernández):

Modelado 3D: Crear el modelo del circuito en Tinkercad.

Pruebas del circuito: Verificar la funcionalidad del sistema en Tinkercad y asegurar que el buzzer y el LED funcionen adecuadamente.

Investigación: Documentar el funcionamiento del código Morse y la teoría detrás de su uso.

Presentación: Colaborar en la presentación final del proyecto.

Otros Integrantes:

Fleitas: Encargado de las conexiones físicas del circuito.

Krawczyk: Responsable de la decodificación de Morse a texto y visualización en LCD.

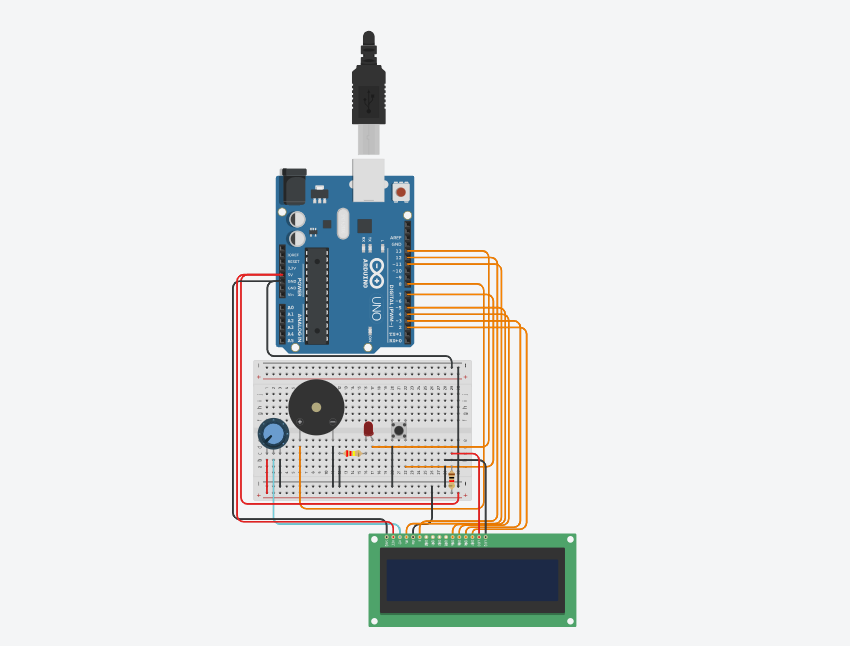
Korstanje: Apoyo general en pruebas y documentación.

### Conexiones del Circuito

#### Arduino 1 (Emisión de Código Morse):

1. Buzzer (1)
2. Resistencias (2)
3. LED (1)
4. Pantalla LCD 16x2 (1)
5. Botón (1)
6. Potenciómetro (1)
7. Cables (Alrededor de 25)

Inicial

****

Armado de cables y componentes inicialmente planteado por Korstanje, Fleitas y Hernandez

### Código Inicial Utilizado

| String letrasMor[] =  {".-","-...","-.-.","-..",".","..-.","--.","....",".." // A-I |
| --- |

Primero, tenemos el arreglo letrasMor[], donde cada posición representa una letra del alfabeto, desde la A hasta la Z. Por ejemplo:

La posición 0 corresponde a la letra A y su equivalente en Morse es ".-".

La posición 1 corresponde a la letra B, que en Morse es "-...".

Para que quede más claro, dividimos el alfabeto en tres partes (A-I, J-R u S-Z)

| ,".---","-.-",".-..","--","-.","---",".--.","--.-",".-." // J-R ,"...","-..-","...-",".--","-..-","-.--","--.."}; // S-Z   String numerosMor[] = |
| --- |

Después, hicimos lo mismo con los números en el arreglo numerosMor[]. Este arreglo tiene 10 elementos, uno para cada número del 0 al 9. Por ejemplo:

El 0 está en la posición 0 del arreglo y en Morse es "-----".

El 1 está en la posición 1 y en Morse es ".----".

| {"-----",".----","..---","...--","....,",".....","-...." // 0-6 ,"--...","---..","----."}; // 7-9 |
| --- |

*Lo que hicimos fue crear dos arreglos de cadenas, uno para guardar el alfabeto en código Morse y otro para los números en Morse.*

*Ese fue el planteamiento inicial del código.*

*Una vez aplicado lo anterior mencionado, el string quedo asi:*

*Aca fue organizamos todo de manera más clara y práctica para trabajar con traducciones entre texto, números y código Morse. En lugar de tener un solo arreglo, separamos las letras, los códigos Morse de letras, los números, y los códigos Morse de números en arreglos distintos para que sea más fácil de manejar.*

| String letras[] = { "a" , "b" , "c" , "d" , "e" , "f" , "g" , "h" , "i" , "j" , "k" , "l" , "m" , "n" , "o" , "p" , "q" , "r" , "s" , "t" , "u" , "v" , "w" , "x" , "y" , "z" }; String codigo[] = { ".-", "-..." , "-.-." , "-.." , "." , "..-." , "--." , "...." , ".." , ".---" , "-.-" , ".-.." , "--" , "-." , "---" , ".--." , "--.-" , ".-." , "..." , "-" , "..-" , "...-" , ".--" , "-..-" , "-.--" , "--.." };  codigo[]: Este contiene los equivalentes en código Morse para cada letra del alfabeto, también en orden.  letras[]: Este arreglo contiene todas las letras del alfabeto en minúsculas, de la a a la z, en orden.  int numeros[] = { 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 };  numeros[]: Este arreglo simplemente contiene los números del 0 al 9, en orden.  String codigoNum[] = { "-----" , ".----" , "..---" , "...--" , "....-" , "....." , "-...." , "--..." , "---.." , "----." };  codigoNum[]: Este tiene los códigos Morse correspondientes para cada número. |
| --- |

| ".--/-...//-.-" |
| --- |

Por parte de mi compañero Martín, ha compartido el avance de su código el cual planteando juntos quedó así (Cabe aclarar que esto lo hizo sólo Korstanje y Romero.):



| #include <LiquidCrystal.h> LiquidCrystal LCD(2,3,9,10,11,12);  #include <LiquidCrystal.h>: Incluye la biblioteca que permite manejar una pantalla LCD.  LiquidCrystal LCD(2, 3, 9, 10, 11, 12);: Inicializa la pantalla LCD conectada a los pines digitales especificados.  #define boton 4 #define buz 7  #define boton 4: Define el pin 4 como el botón que leerá las pulsaciones del usuario.  #define buz 7: Define el pin 7 como el buzzer que emitirá sonidos.    String vecPalabra[1]; String letrasMor[] =  {".-","-...","-.-.","-..",".","..-.","--.","....",".." // A-I  ,".---","-.-",".-..","--","-.","---",".--.","--.-",".-." // J-R  ,"...","-", "..-","...-",".--","-..-","-.--","--.."}; // S-Z  String letras[] =  {"A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","V","W","X","Y","Z"}; String numerosMor[] =  {"-----",".----","..---","...--","....,",".....","-...." // 0-6 ,"--...","---..","----."}; // 7-9 String numeros[] =  {"0","1","2","3","4","5","6","7","8","9"}; String palabra = ""; int contPos = 0; char caracter;  letrasMor[] y letras[]: Contienen las letras del alfabeto y sus equivalentes en código Morse.  numerosMor[] y numeros[]: Contienen los números del 0 al 9 y sus equivalentes en código Morse.  Esto permite buscar una letra o número en texto y encontrar su representación Morse, o viceversa  String palabra = "";  int contPos = 0;  char caracter;  palabra: Almacena el código Morse completo ingresado por el usuario.  contPos: Lleva el conteo de la posición actual en el LCD.  caracter: Guarda el carácter actual (., -, o /) que el usuario genera.  void setup() {  pinMode(boton, INPUT\_PULLUP);  pinMode(buz, OUTPUT);  Serial.begin(9600);  LCD.begin(16,2); }  void loop()  pinMode(boton, INPUT\_PULLUP);: Configura el botón como entrada con resistencia pull-up para evitar lecturas falsas.  pinMode(buz, OUTPUT);: Configura el buzzer como salida para emitir sonidos.  Serial.begin(9600);: Inicia la comunicación serial para depuración.  LCD.begin(16, 2);: Configura la pantalla LCD con 16 columnas y 2 filas.  {  while(true){  LCD.setCursor(0,0);  unsigned int valor = pulseIn(boton, LOW);  if (valor > 0 && valor < 40000){  caracter = '.';  digitalWrite(buz, HIGH);  delay(250);  digitalWrite(buz, LOW);  }else if(valor >= 40000 && valor < 80000){  caracter = '-';  digitalWrite(buz, HIGH);  delay(500);  digitalWrite(buz, LOW);  }  else if(valor >= 80000 && valor < 120000){  caracter = '/';  }else if(valor >= 120000){  break;  }  palabra = palabra + caracter;  Serial.print(valor);  Serial.print("=");  Serial.println(caracter);  }  pulseIn(boton, LOW): Mide el tiempo que el botón está presionado.  Menor a 40 ms: Se interpreta como un punto (.).  Entre 40-80 ms: Se interpreta como una raya (-).  Entre 80-120 ms: Se interpreta como un separador (/).  Más de 120 ms: Finaliza la captura.  Los valores se agregan a palabra y se imprimen en el monitor serial para depuración.  int longPalabra = palabra.length();  String letra = "";    for(int i = 0; i < longPalabra; i++){  if(palabra[i] != '/'){  letra = letra + palabra[i];  }  else if (palabra[i] == '/'){  for(int i = 0; i < 26; i++){  if(letra == letrasMor[i]){  LCD.print(letras[i]);  contPos++;  }  }  for(int i = 0; i < 10; i++){  if(letra == numerosMor[i]){  LCD.print(numeros[i]);  contPos++;  }  if(i == 9 && letra != numerosMor[i]){  LCD.print("?");  contPos++;  }  }  letra = "";  }else if((i + 1 < longPalabra) && (palabra[i] == '/' && palabra[i+1] == '/')){  LCD.print(" ");  }if(contPos == 15){  LCD.setCursor(0,1);  }   }  }  Cada carácter se agrega a letra hasta encontrar /, que indica el final de una letra.  Doble / indica un espacio entre palabras.  Traducción:  Compara letra con los códigos en letrasMor[] y numerosMor[] para encontrar coincidencias.  Si no coincide, imprime un ? para indicar un error.  Control del LCD:  Después de 15 caracteres, cambia de línea en el LCD.  C++ code // void setup() //  {  Serial.begin(9600); }  void loop() // void loop() datos desde 3.  {    if( Serial.available() )  {   String g = "";  while( Serial.read() != -1){  g = g + (char)Serial.read();  }    Serial.println( g );  }  } |
| --- |

Esta sección permite leer datos desde la consola serial y mostrarlos directamente, útil para depurar**.**

**Resumen**

**Captura:** Usa el botón para generar puntos (.), rayas (-), y separadores (/).

**Traducción:** Convierte los códigos Morse en letras y números usando arreglos.

**Pantalla LCD:** Muestra el texto traducido en tiempo real.

**Serial:** Imprime información útil para depuración.

#### (Morse a Texto)

Este código recibe las pulsaciones del botón como señales de código Morse, las traduce a texto y las muestra en una pantalla LCD. En el código actualizado, hay algunas diferencias o detalles que no expliqué anteriormente.

| #include <LiquidCrystal.h>  LiquidCrystal LCD(2, 3, 9, 10, 11, 12); #define boton 4 #define buz 7  String letrasMor[] = { ".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....", "..",   ".---", "-.-", ".-..", "--", "-.", "---", ".--.", "--.-", ".-.",   "...", "-", "..-", "...-", ".--", "-..-", "-.--", "--.." };  String letras[] = { "A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M",   "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z" };  String palabra = ""; int contPos = 0; char caracter;  void setup() {  pinMode(boton, INPUT\_PULLUP);  pinMode(buz, OUTPUT);  Serial.begin(9600);  LCD.begin(16, 2); }  void loop() {  while (true) {  LCD.setCursor(0, 0);  unsigned int valor = pulseIn(boton, LOW);   if (valor > 0 && valor < 40000) {  caracter = '.';  digitalWrite(buz, HIGH);  delay(250);  digitalWrite(buz, LOW);  } else if (valor >= 40000 && valor < 80000) {  caracter = '-';  digitalWrite(buz, HIGH);  delay(500);  digitalWrite(buz, LOW);  } else if (valor >= 80000 && valor < 120000) {  caracter = '/';  } else if (valor >= 120000) {  break;  }   palabra = palabra + caracter;  Serial.print(valor);  Serial.print("=");  Serial.println(caracter);  }   int longPalabra = palabra.length();  String letra = "";  for (int i = 0; i < longPalabra; i++) {  if (palabra[i] != '/') {  letra = letra + palabra[i];  } else if (palabra[i] == '/') {  for (int i = 0; i < 26; i++) {  if (letra == letrasMor[i]) {  LCD.print(letras[i]);  contPos++;  }  }  letra = "";  } else if ((i + 1 < longPalabra) && (palabra[i] == '/' && palabra[i + 1] == '/')) {  LCD.print(" ");  }  if (contPos == 15) {  LCD.setCursor(0, 1);  }  } } |
| --- |

palabra = palabra + caracter;

Aca se acumula cada carácter (., -, o /) en la variable palabra, formando así una cadena completa que contiene el código Morse ingresado por el usuario.

Ejemplo: Si el usuario pulsa el botón para ingresar .-, luego -..., y después un /, la variable palabra contendrá la cadena ".-/-...".

**Decodificación de Palabra Completa**

longPalabra: Obtiene la longitud de la cadena palabra para recorrerla carácter por carácter.

letra: Actúa como una variable temporal donde se almacenan los puntos (.) y rayas (-) de cada letra antes de decodificarlas.

**Separación y Decodificación de letras**

Cuando no se encuentra un separador (/):

Los caracteres (. y -) se van acumulando en letra.

Cuando se encuentra un separador (/):

La cadena acumulada en letra se compara con los códigos Morse en letrasMor[].

Si coincide, se imprime la letra correspondiente de letras[] en el LCD.

Después de decodificar, se reinicia letra para procesar la siguiente.

**Manejo de Espacios entre Palabras**

Detecta espacios dobles (//):

Si se encuentran dos separadores consecutivos, se interpreta como un espacio entre palabras y se imprime un espacio en blanco en el LCD.

**Control de Línea en el LCD**

Cuando la cantidad de caracteres mostrados alcanza el límite de 15 en la primera línea del LCD, el cursor salta a la segunda línea para continuar mostrando texto.

Estos detalles refinan el manejo de la palabra y la decodificación de letras y espacios en el código Morse

**Un breve resumen de los dos códigos**

Proyecto: Decodificador de Código Morse

En nuestro equipo desarrollamos un programa que permite ingresar mensajes en código Morse utilizando un botón y luego decodificarlos para mostrarlos en una pantalla LCD.

1. Primer Código:

Creamos una estructura básica que detecta las pulsaciones del botón para interpretar puntos (.), rayas (-), y separadores (/). Usamos arreglos para almacenar las equivalencias entre el alfabeto y el código Morse. Al procesar las entradas, el mensaje decodificado se muestra en el LCD, permitiendo traducir las pulsaciones en tiempo real.

2. Segundo Código:

Refinamos nuestro diseño para gestionar palabras completas, detectar espacios entre palabras (doble /) y optimizar la decodificación. También agregamos control de línea para manejar el límite de caracteres del LCD. Este enfoque mejorado permitió interpretar mensajes más largos de manera fluida y eficiente.

Resultado Final:

Nuestro sistema traduce las pulsaciones del botón a texto claro, manteniendo un diseño funcional y amigable. Es una solución práctica para aprender o practicar código Morse de forma interactiva.

### **Simulaciones en Tinkercad**

En Tinkercad, se ha creado el modelo 3D del circuito, incluyendo todos los componentes como Arduino, LED, buzzer, botón y LCD. Se realizaron pruebas para verificar el funcionamiento del sistema.  
En Tinkercad, el proyecto permite simular la emisión de señales de código Morse y su decodificación correctamente.

### Investigación sobre Código Morse

El código Morse es un sistema de codificación que usa secuencias de puntos (.) y rayas (-) para representar letras, números y otros caracteres. Es un método de comunicación muy utilizado en situaciones de emergencia o cuando otras formas de comunicación no están disponibles.

Ejemplo de Código Morse:

A = . -

B = - . . .

1 = .----

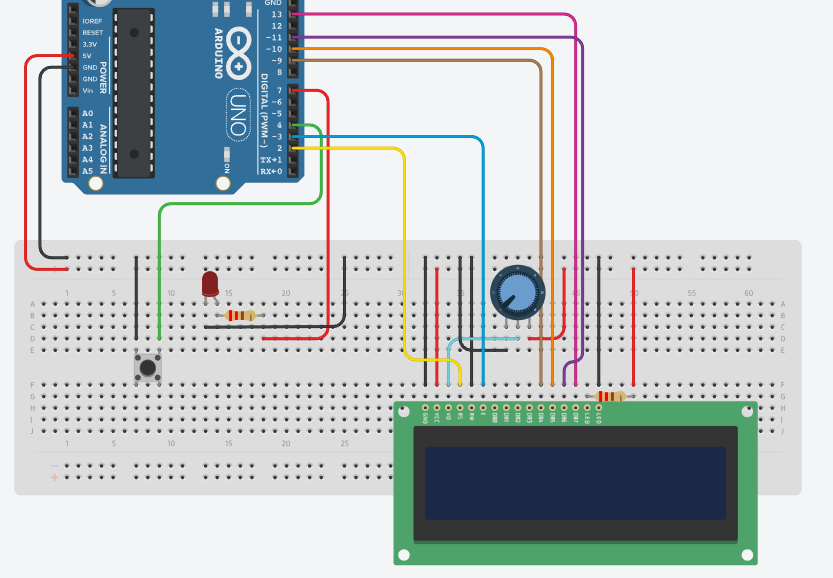
0 = -----

Espacios:

Entre letras: Pausa corta.

Entre palabras: Pausa más larga.

Conexiones del arduino:

****

Link de Arduino: [**https://www.tinkercad.com/things/4A22je0iBqV-codigo-morse/editel?returnTo=https%3A%2F%2Fwww.tinkercad.com%2Fdashboard**](https://www.tinkercad.com/things/4A22je0iBqV-codigo-morse/editel?returnTo=https%3A%2F%2Fwww.tinkercad.com%2Fdashboard)

Mis investigaciones:

Video 1: ¿Qué es el Código Morse?

Descripción: Un video explicativo que describe cómo funciona el código Morse, su historia, cómo se modifican las letras, los números y los signos de puntuación. Introducción para entender la teoría detrás del proyecto.

Buscador de YouTube: "Cómo funciona el código Morse"

Enlace:<https://www.youtube.com/watch?v=H4j_f_spV-A>

#### Vídeo 2: Introducción al Código Morse con Arduino

#### Descripción: Un video donde se explica cómo implementar un sistema de código Morse básico usando un Arduino, un LED y un buzzer. Tutoriales de como convertir texto a Morse de manera visual y sonora.

Buscador de YouTube: "Código morse con Arduino y buzzer"

Enlace: https://www.youtube.com/watch?v=lunnM6lPkyo&t=574s

#### Video 6: Comunicación serial entre dos Arduinos

Descripción: Un tutorial sobre cómo configurar la comunicación entre dos Arduinos utilizando puertos seriales. El video enseña cómo transmitir datos entre los dos Arduinos, uno convierte texto a Morse y el otro lo decodifica.

Buscador de YouTube: "Comunicacion serial entre dos Arduinos"

Enlace sugerido:https://youtu.be/B8OPTLKNIJc?si=Mq0yR\_-W\_jkMQZ0m

#### Video 7: Comunicación entre Arduinos vía Bluetooth

Descripción: Comunicación inalámbrica, este video sobre la comunicación Bluetooth con Arduino es útil. Es un tutorial sobre como usar esta comunicación para enviar mensajes de texto o Morse entre dos dispositivos.

Buscador de YouTube: "Comunicación Bluetooth entre dos Arduinos"

Enlace:https://youtu.be/JB3sldVkP6E?si=FmImAaDBfWzT4P9t

### Wikipedia - Código Morse

La página de Wikipedia sobre el código Morse es una excelente fuente para entender su historia, su funcionamiento y sus aplicaciones modernas. Además, proporciona una tabla de referencia completa con las letras, números y símbolos utilizados en código Morse.

Enlace:https://es.wikipedia.org/wiki/C%C3%B3digo\_morse

Contenido relevante:

Historia y contexto del código Morse.

Caracteres y símbolos utilizados en Morse.

Diferentes aplicaciones del código Morse en telecomunicaciones.

### Instructables - Guías y Proyectos DIY

Instructables es una plataforma en línea donde los usuarios comparten tutoriales paso a paso sobre cómo hacer proyectos electrónicos, de programación y de hardware. Hay muchos proyectos relacionados con Arduino y código Morse, desde básicos hasta más avanzados.

Enlace: https://www.instructables.com/Enviando-c%C3%B3digo-morse-com-o-Arduino/

Contenido relevante:

Tutoriales completos: Instructables tiene guías detalladas sobre cómo hacer traductores de código Morse con Arduino, incluyendo los diagramas de conexión, el código y los pasos a seguir.

Proyectos de Arduino con pantalla LCD y buzzer: Muchos proyectos de Arduino incluyen la utilización de pantallas LCD y buzzer

Exposición con Canva:

### Presentación en Canva sobre el Proyecto, realizada con los siguientes integrantes: Korstanje, Romero y Fleitas.

Para la exposición de nuestro proyecto, estuvimos creando una presentación en Canva, donde vamos a explicar todo el proceso de desarrollo del traductor de código Morse con Arduino. Esta presentación servirá como apoyo visual durante la exposición, y está organizada para que la audiencia entienda claramente los pasos seguidos, los objetivos alcanzados y cómo funciona el sistema. A continuación, detallo la estructura que tiene la presentación y qué contenido se me ocurrió para incluir en cada parte.

#### Portada

En la primera diapositiva se me ocurrió incluir el título del proyecto: "Morse grupal". También, colocar los nombres del equipo: Tomas Fleitas, Kiara Hernandez, Benjamin Korstanje,Martin Krawczyk, Brunella Figallo. La fecha de la exposición 3/12 y un ícono representativo del proyecto, como el logo de Arduino o algún símbolo relacionado con el código Morse.

#### Introducción al Proyecto

En esta diapositiva vamos a explicar brevemente de qué se trata el proyecto. Mencionaremos que el objetivo es poder traducir texto a código Morse utilizando Arduino y permitir que otro Arduino reciba ese código Morse y lo traduzca nuevamente

#### Objetivos del Proyecto

Aca explicamos los objetivos específicos que buscábamos cumplir con este proyecto

#### Materiales y Componentes

En esta sección, describimos los materiales y componentes utilizados en el proyecto. Colocaré imágenes de cada uno de los componentes para hacerlo visualmente atractivo

#### Esquema del Circuito

En esta diapositiva pensé en agregar un diagrama de conexiones del circuito. Esto mostrará cómo están conectados los diferentes componentes entre sí y con el Arduino. Puedo usar el diagrama de Tinkercad que ya he creado, o también puedo hacer uno más simple usando herramientas en Canva. Asegurarme de destacar las conexiones importantes

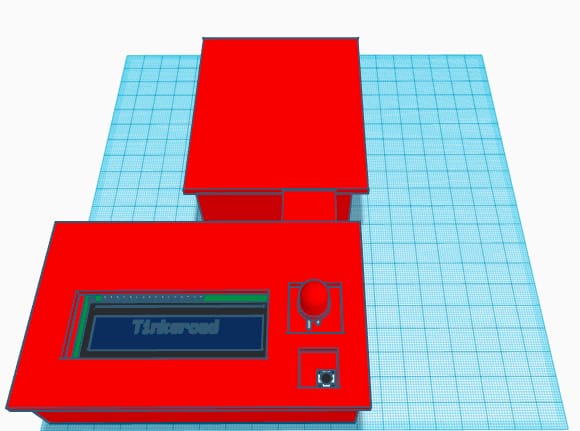
#### Desafíos y Soluciones

Se me ocurrió agregar esta sección, para explicar algunos de los desafíos que encontramos durante el desarrollo del proyecto y cómo los resolvimos

Resultados y conclusiones.

**Modelo 3D del Proyecto Decodificador de Código Morse final.**

Nuestro equipo diseñó y desarrolló este modelo físico para representar el decodificador de código Morse, integrando todos los componentes necesarios para su funcionalidad. Aquí explicamos cada detalle, destacando el trabajo en equipo detrás de su diseño:



Caja Principal (Frontal):

Esta es la parte visible del dispositivo, diseñada para facilitar la interacción del usuario.

Pantalla LCD:

Ubicada en el centro-izquierda de la superficie superior, muestra en tiempo real los mensajes decodificados ingresados en código Morse.

Trabajamos en ajustar las dimensiones de la pantalla para que quede embebida en la caja de manera segura, optimizando la estética y el funcionamiento.

Botón Pulsador:

Situado en la parte superior derecha, el usuario lo utiliza para ingresar puntos (.), rayas (-), y separadores (/) al mantenerlo presionado por diferentes intervalos de tiempo.

Nuestro equipo seleccionó este tipo de botón por su durabilidad y precisión al detectar las pulsaciones.

Buzzer:

Posicionado junto al botón, emite un sonido para confirmar cada entrada, proporcionando una retroalimentación inmediata al usuario.

Diseñamos el espacio para que el buzzer quedara ligeramente elevado, asegurando un sonido claro y sin interferencias.

Caja Secundaria (Posterior):

Este compartimiento posterior fue diseñado para alojar los componentes internos del proyecto:

Microcontrolador (Arduino): Es el núcleo del sistema, donde se procesa la lógica del programa para interpretar las entradas Morse y generar la salida en texto.

Placa de Prototipos y Cables: Incluye las conexiones necesarias para integrar todos los componentes. Nuestro equipo se esforzó en minimizar el uso de cables y organizar el diseño interno para evitar interferencias y facilitar el mantenimiento.

Fuente de Alimentación (Batería): El espacio también contempla el almacenamiento de una batería o conexión externa para garantizar la portabilidad del dispositivo.

Base del Diseño:

Diseñamos la base para garantizar estabilidad durante el uso. La superficie amplia y el peso distribuido evitan que el dispositivo se desplace mientras el usuario interactúa con él.

También previmos ventilación y ranuras internas para evitar sobrecalentamiento de los componentes electrónicos.

Proceso de Diseño en Equipo:

Diseño CAD:

Utilizamos herramientas como TinkerCAD para modelar y ajustar las dimensiones del proyecto. Cada miembro del equipo aportó ideas para equilibrar funcionalidad y estética.

Prototipado:

Realizamos pruebas físicas con materiales como plástico o acrílico para garantizar que la disposición de los componentes fuera cómoda y eficiente.

Optimización:

Iteramos el diseño para reducir espacio sin comprometer el rendimiento, logrando un dispositivo compacto, funcional y fácil de transportar.

Resultado Final:

El modelo combina diseño intuitivo y funcionalidad, permitiendo que cualquier usuario practique y decodifique mensajes en código Morse de forma interactiva. Es una muestra del esfuerzo y colaboración de nuestro equipo en todas las etapas del proyecto.

Apartir del resultado final, pense en algunos consejos para mejorar y optimizar tanto el diseño del dispositivo como su funcionalidad, aunque se nos complica para terminarlo asi que quedara como esta.

Optimización del Hardware

Mejorar la retroalimentación sonora del buzzer.

Se podria agregar tonos diferenciados para puntos (.) y rayas (-), haciendo más clara la interacción.

Considerar un buzzer piezoeléctrico con volumen ajustable.

Usar un botón más ergonómico

Optar por un botón más grande o con mejor respuesta táctil, para evitar fatiga al usuario durante sesiones prolongadas.

Integrar LEDs como apoyo visual:

Incorporar un LED que parpadee con cada pulsación, ayudando a usuarios con dificultades auditivas.

Asegurar mejor disipación de calor:

Agregar rejillas de ventilación en la caja secundaria para evitar que el microcontrolador y otros componentes se sobrecalienten.

Ampliar la Funcionalidad del Software

Agregar soporte para números y caracteres especiales:

Extender los arreglos de traducción para incluir dígitos (0-9) y signos como puntos, comas, etc., haciéndolo más versátil.

Incluir un modo de aprendizaje:

Se podria agregar un sistema que enseñe al usuario cómo escribir en código Morse, mostrando la letra asociada a cada pulsación en tiempo real.

Mejorar el control de pantalla:

Implementar desplazamiento automático en el LCD si se excede el espacio disponible, evitando pérdida de información.

Mejorar el Diseño del Modelo 3D

Compactar el diseño:

Integrar la caja secundaria con la principal, reduciendo el tamaño total del dispositivo. Esto lo haría más portátil.

Agregar accesos para mantenimiento:

Diseñar una tapa removible en la parte inferior o trasera para acceder fácilmente a los componentes internos.

Personalizar el diseño:

Incluir una cubierta con un diseño atractivo, como grabados o colores relacionados con el proyecto (por ejemplo, un gráfico de código Morse).

Incorporar Conectividad

Agregar Bluetooth o WiFi:

Permitir la conexión con un dispositivo móvil para que el mensaje decodificado pueda ser enviado o recibido en tiempo real.

Guardar mensajes decodificados:

Incluir una memoria interna o tarjeta SD para almacenar los mensajes traducidos. Esto permitiría revisar un historial de mensajes.

Hacerlo más Inclusivo

Modo de vibración:

Para usuarios con discapacidad visual o auditiva, podrías incluir un motor vibrador que indique puntos y rayas mediante pulsos.

Soporte multilingüe:

Extender el software para que soporte otros idiomas que usen alfabetos distintos o incluir guías de pronunciación.

Con estas mejoras, se podría llevar el proyecto a otro nivel, haciéndolo más interactivo, accesible y funcional.

**Código Final**

| #include <LiquidCrystal.h> LiquidCrystal LCD(2, 3, 9, 10, 11, 13);  #define boton 4 #define led 7  String letrasMor[] = {".-","-...","-.-.","-..",".","..-.","--.","....","..", //A-I   ".---","-.-",".-..","--","-.","---",".--.","--.-",".-.", //J-R   "...","-","..-","...-",".--","-..-","-.--","--.."}; // S-Z String letras[] = {"A","B","C","D","E","F","G","H","I",  "J","K","L","M","N","O","P","Q","R",  "S","T","U","V","W","X","Y","Z"};  String numerosMor[] = {"-----",".----","..---","...--","....-" // 0-4  ,".....","-....","--...","---..","----."}; // 5-9 String numeros[] = {"0","1","2","3","4",  "5","6","7","8","9"};  int contPos = 0;  void setup() {  pinMode(boton, INPUT\_PULLUP);  pinMode(led, OUTPUT);  Serial.begin(9600);  LCD.begin(16, 2); }  void loop() {  principal(); }  String lectura() {  String a = "";  while (Serial.available()) {   char caracter = Serial.read();  a += caracter;  delay(2);  }  Serial.println(a);  return a; }  bool buscar(String string, int hastaDonde, String vectorComparar[], String vectorImprimir[]) {  bool encontrado = false;  for (int n = 0; n < hastaDonde; n++) {  if (string == vectorComparar[n]) {  LCD.print(vectorImprimir[n]);  encontrado = true;  break;  }  }  return encontrado; }  String buscarMorse(String caracter, int hastaDonde, String vectorComparar[], String vectorImprimir[]) {  String morse = "";  for (int n = 0; n < hastaDonde; n++) {  if (caracter == vectorComparar[n]) {  morse = vectorImprimir[n];  break;  }  }  return morse; }  void principal(){  LCD.clear();  LCD.setCursor(0,0);  LCD.print("1 - Boton");  LCD.setCursor(0,1);  LCD.print("2 - Texto");  LCD.setCursor(0,0);  while(!Serial.available()){    }  if (Serial.available()) {  String opcion = lectura();  if(opcion == "1"){  LCD.clear();  BotonATXT();  }  else if(opcion == "2"){  LCD.clear();  LCD.print("1 - Morse a TXT");  LCD.setCursor(0,1);  LCD.print("2 - TXT a Morse");  LCD.setCursor(0,0);  while(!Serial.available()){    }  if(Serial.available()){  String opcion2 = lectura();  if(opcion2 == "1"){  LCD.clear();  MorseATXT();  }  else if(opcion2 == "2"){  LCD.clear();  TXTAMorse();  }  else{  Serial.println("No existe esa opcion.");  }  }  }  else{  Serial.println("No existe esa opcion.");  }  } }  void MorseATXT(){  while(!Serial.available()){    }  if (Serial.available()) {  LCD.clear();  String caracter = "";  contPos = 0;  String texto = lectura();  int longTexto = texto.length();  for (int i = 0; i < longTexto; i++) {  if (texto[i] != '/') {  caracter += texto[i];  }   if (texto[i] == '/' || i + 1 == longTexto) {  if (contPos < 16) {  LCD.setCursor(contPos, 0);  } else {  LCD.setCursor(contPos - 16, 1);  }    bool encontrado = buscar(caracter, 26, letrasMor, letras);  if (!encontrado) {  encontrado = buscar(caracter, 10, numerosMor, numeros);  }  if (!encontrado && caracter != "") {  LCD.print("?");  }    contPos++;  caracter = "";   if (i < longTexto - 1 && texto[i] == '/' && texto[i + 1] == '/') {  LCD.print(' ');  i++;  contPos++;  }   if (contPos > 31) {  Serial.println("El texto es demasiado largo. Se borrara la pantalla y continuara en 3 segundos");  delay(3000);  LCD.clear();  contPos = 0;  }  }  }  Serial.println("Mostrando el texto por los siguientes 5 segundos");  delay(5000);  } }   void TXTAMorse(){  while(!Serial.available()){    }  if (Serial.available()) {  LCD.clear();  contPos = 0;  int longMorse = 0;  String letraActual;  String texto = lectura();  texto.toUpperCase();  int longTexto = texto.length();  for (int i = 0; i < longTexto; i++) {  if (contPos < 16) {  LCD.setCursor(contPos, 0);  } else {  LCD.setCursor(contPos - 16, 1);  }  letraActual = texto[i];  String morse = buscarMorse(letraActual, 26, letras, letrasMor);  if (morse == "") {  morse = buscarMorse(letraActual, 10, numeros, numerosMor);  }  if(morse == "" && letraActual == " "){  LCD.print(' ');   }  else if (morse == "" && letraActual != "") {  LCD.print("?");  }  if(morse != "" && letraActual != ""){  LCD.print(letraActual);  }  longMorse = morse.length();  for(int m = 0; m < longMorse; m++){  if (morse[m] == '.'){  digitalWrite(led, HIGH);  delay(250);  digitalWrite(led, LOW);  delay(250);  }  else if(morse[m] == '-'){  digitalWrite(led,HIGH);  delay(500);  digitalWrite(led,LOW);  delay(250);  }  }    contPos++;  if (contPos > 31) {  Serial.println("El texto es demasiado largo. Se borrara la pantalla y continuara en 3 segundos");  delay(3000);  LCD.clear();  contPos = 0;  }    }  }  Serial.println("Mostrando el texto por los siguientes 5 segundos");  delay(5000);   }    void BotonATXT(){  char caracter;  String texto = "";  String palabra = "";  while(true){  caracter;  float valor = pulseIn(boton, LOW, 3000000);  if (valor == 0){  Serial.println("No se registro ningun valor (no presiono el boton durante mucho tiempo)");  }  if (valor > 0 && valor < 100000){  caracter = '.';  texto = texto + caracter;  digitalWrite(led, HIGH);  delay(250);  digitalWrite(led, LOW);  delay(250);  }else if(valor >= 100000 && valor < 150000){  caracter = '-';  texto = texto + caracter;  digitalWrite(led, HIGH);  delay(500);  digitalWrite(led, LOW);  delay(250);  }  else if(valor >= 150000 && valor < 200000){  caracter = '/';  texto = texto + caracter;  }else if(valor >= 200000){  break;  }  }   contPos = 0;  int longTexto = texto.length();  for (int i = 0; i < longTexto; i++) {  if (texto[i] != '/') {  palabra += texto[i];  }  if (texto[i] == '/' || i + 1 == longTexto) {  if (contPos < 16) {  LCD.setCursor(contPos, 0);  }   else {  LCD.setCursor(contPos - 16, 1);  }  bool encontrado = buscar(palabra, 26, letrasMor, letras);  if (!encontrado) {  encontrado = buscar(palabra, 10, numerosMor, numeros);  }  if (!encontrado && palabra != "") {  LCD.print("?");  }  contPos++;  palabra = "";  if (i < longTexto - 1 && texto[i] == '/' && texto[i + 1] == '/') {  LCD.print(' ');  i++;  contPos++;  }   if (contPos > 31) {  Serial.println("El texto es demasiado largo. Se borrara la pantalla y continuara en 3 segundos");  delay(3000);  LCD.clear();  contPos = 0;  }  }  }  Serial.println("Mostrando el texto por los siguientes 5 segundos");  delay(5000); } |
| --- |

**Explicación del código**

Este es el resumen y explicación del proyecto que realizamos como equipo, centrado en un sistema que convierte el código Morse en texto y viceversa, utilizando un botón, texto de entrada y una pantalla LCD para mostrar los resultados. El código también permite que el usuario interactúe a través de la comunicación serial. Partes clave de la implementación:

Declaración de variables y configuración

Librerías y objetos inicializados:

Usamos la librería LiquidCrystal para manejar la pantalla LCD. Se configuró la pantalla en los pines 2, 3, 9, 10, 11 y 13 del Arduino.

Definimos un botón en el pin 4 y un LED en el pin 7, que se utilizarán para interactuar con el usuario.

Arreglos de caracteres en Morse y sus equivalentes en texto:

Los arreglos letrasMor y numerosMor contienen las representaciones en código Morse de las letras y los números, respectivamente.

Los arreglos letras y numeros contienen las letras y números correspondientes en formato de texto.

Función setup()

Configuramos los pines del botón (entrada) y del LED (salida).

Inicializamos la comunicación serial para recibir y enviar datos desde y hacia el computador.

Configuramos el LCD para mostrar información en una pantalla de 16x2 caracteres.

Función loop()

Llama a la función principal(), que gestiona las opciones del programa y la entrada del usuario. Esto incluye elegir entre diferentes modos de conversión: convertir Morse a texto o texto a Morse.

Función lectura()

Lee los datos recibidos por la comunicación serial y los almacena en una variable tipo String, retornándolos para su procesamiento posterior.

Función buscar()

Se utiliza para comparar una cadena de texto con los arreglos de código Morse. Si encuentra una coincidencia, imprime el texto correspondiente en el LCD.

Función buscarMorse()

Similar a la función anterior, pero en este caso, devuelve la cadena en Morse correspondiente a un carácter en texto.

Función principal()

Es la función central que maneja el menú principal y las opciones del programa:

Opción 1: "Botón a texto" convierte las pulsaciones del botón en código Morse y luego las traduce a texto.

Opción 2: "Texto a Morse" y "Morse a texto" permite al usuario ingresar texto o Morse y realizar la conversión respectiva.

Función MorseATXT()

Esta función recibe un texto en Morse a través de la comunicación serial y lo convierte en texto.

Usa los arreglos letrasMor y numerosMor para encontrar las correspondencias y mostrar el texto en el LCD. Si se encuentra un error o un símbolo no reconocido, muestra un signo de interrogación (?).

Función TXTAMorse()

Convierte un texto ingresado por el usuario en código Morse.

Cada letra es convertida a su equivalente en Morse, y el LED parpadea en secuencia con la duración correcta de los puntos (250 ms) y las rayas (500 ms). Luego, imprime la letra o símbolo en el LCD.

Función BotonATXT()

Convierte las pulsaciones de un botón en código Morse.

Si el botón se presiona brevemente, se interpreta como un punto (.).

Si se mantiene presionado un poco más, se interpreta como una raya (-).

Si el botón se mantiene presionado por un largo período, se interpreta como un espacio entre palabras (/).

Al finalizar la entrada de pulsaciones, el código convierte el código Morse a texto, utilizando las funciones buscar() y buscarMorse(). Si el texto es demasiado largo, se muestra un mensaje de advertencia.

Flujo de trabajo y control del LCD

El LCD muestra los resultados de las conversiones, y en cada interacción, se asegura de que no se sobrecargue de información. Si se excede el límite de caracteres, la pantalla se borra y continúa mostrando el siguiente conjunto de datos.

Consejos para el uso del código y el proyecto:

El código está diseñado para manejar distintas entradas (botón o texto a través de comunicación serial), lo que hace que el dispositivo sea flexible y accesible.

Se incluye un control para evitar que el LCD se sobrecargue de información, y se ofrece un breve retraso antes de borrar la pantalla si el texto es demasiado largo.

El sistema también permite que el LED actúe como una herramienta visual de apoyo para aquellos que prefieren un indicador visual sobre el sonido de los puntos y rayas en Morse.