Proyecto : Codigo Morse

Alumno : Marcos Torres

4to 1ra : Proyecto Informático

Escuela Técnica Nª32 D.E. 14

Integrantes : Leonard Rubio - Jimena Aysana - Marcos Torres

Introducción :

¿ Qué es el Código Morse?

El código Morse es un sistema de comunicación que traduce letras, números y símbolos en combinaciones de pulsos cortos y largos llamados puntos (.) y rayas (-). Desarrollado en el siglo XIX por Samuel Morse y Alfred Vail, fue diseñado inicialmente para el envío de mensajes a través de telégrafos, permitiendo la transmisión de información a larga distancia de manera eficiente.

¿Cuál es su función?

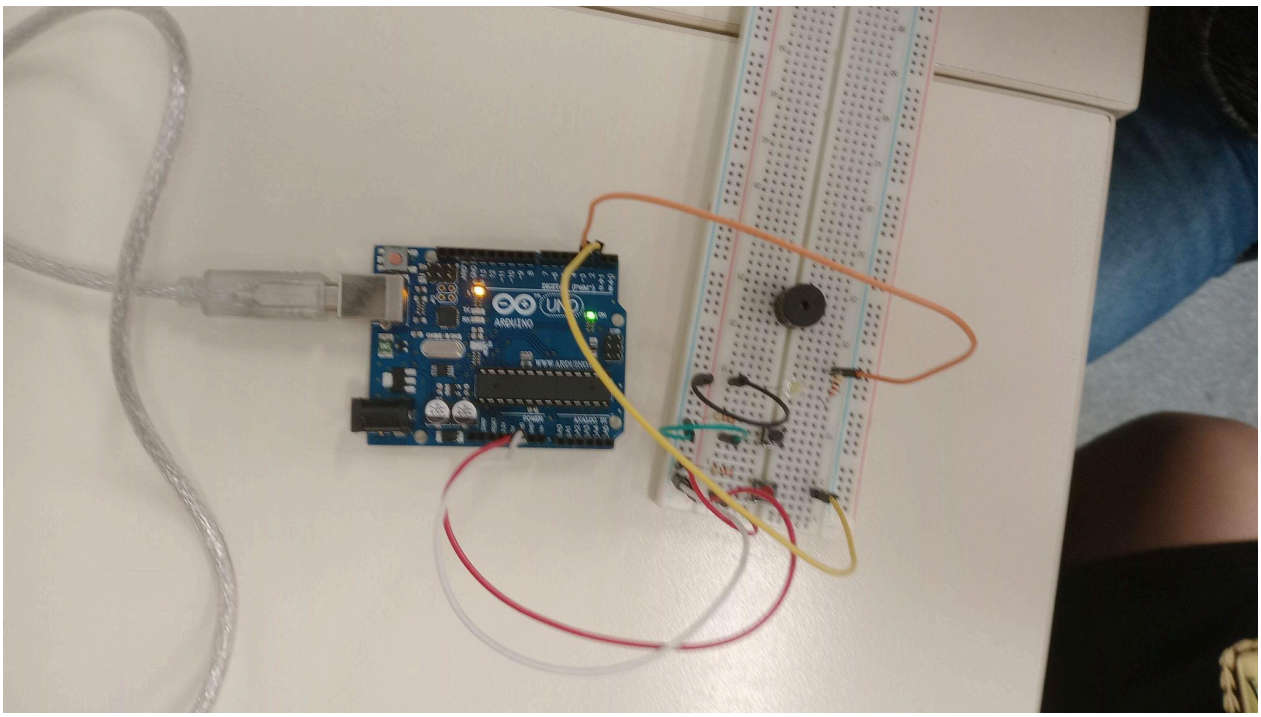
La funcionalidad principal del código Morse es transmitir mensajes de forma rápida y efectiva a través de señales audibles, visuales o eléctricas. Su diseño permite que la información se comunique incluso en condiciones difíciles, como interferencias o ruido de fondo. Esto lo hace especialmente útil en situaciones de emergencia, donde la simplicidad y la claridad son cruciales.

Lo que vamos hacer en el proyecto de código Morse en Arduino es que permita ingresar puntos y rayas mediante un botón, donde la duración de la presión determina si se agrega un punto (“.”) o una raya (“-”). Al confirmar la entrada con otro botón, el sistema traduce el código Morse a la letra correspondiente utilizando un array de estructuras que contiene pares de códigos Morse y letras. La letra decodificada se muestra en una pantalla LCD, y un LED parpadea para indicar la confirmación de la entrada. Esta implementación modular no solo facilita el mantenimiento y la expansión del código, sino que también hace que el proyecto sea interactivo y efectivo.

Al principio iba hacer tocábamos un botón y dependiendo la duración del botón presionado era un punto (“.”) o una raya (“-”) y eso se mostraba con un LED y también con el sonido del buzzer.

Es igual a lo que vamos hacer ahora pero cambia porque cuando presionamos el otro botón es para mandar el dato a la pantalla LCD y que el LED parpadee para indicar la confirmación de entrada.

```
int buttonState = 0;
#define LED 4

void setup()
{
  pinMode(2, INPUT);
  pinMode(4, OUTPUT);
}
v
void loop()
{
    buttonState = digitalRead(2);
    if (buttonState == HIGH)
  {
        digitalWrite(LED, LOW);
  } else
  {

    digitalWrite(LED, HIGH);
  }
  delay(10);
}
```



El primer día, estuve trabajando en el código para que el botón funcione correctamente en el proyecto. Me aseguré de que al presionar el botón, el Arduino lo detectara adecuadamente y respondiera como se espera. Esto involucró configurar el pin del botón, leer su estado y manejar las transiciones para que se registrará la presión del botón de manera eficiente. Además, realicé pruebas para garantizar que el sistema reconociera la entrada y que el botón funcionará de manera fluida, sentando las bases para las siguientes etapas del proyecto.

```
void manejarEntradaMorse() {
  if (digitalRead(PIN_BOTON_MORSE) == LOW) {
    unsigned long duracionPulsacion = pulseIn(PIN_BOTON_MORSE, LOW);

    if (duracionPulsacion < DURACION_PUNTO) {
      entradaMorse += ".";  // Agrega punto
    } else if (duracionPulsacion >= DURACION_PUNTO &&
duracionPulsacion <= DURACION_RAYA) {
```

```
      entradaMorse += "-";  // Agrega raya
    }
    delay(300);  // Pausa entre entradas
  }
}

// Función para manejar el botón de confirmación de letra
void manejarConfirmarLetra() {
  if (digitalRead(PIN_BOTON_CONFIRMAR) == LOW) {
    String letra = morseACaracter(entradaMorse);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Letra:");
    lcd.setCursor(6, 0);
    lcd.print(letra);

    digitalWrite(PIN_LED_CONFIRMAR, HIGH); // LED parpadea para
confirmar
    delay(500);
    digitalWrite(PIN_LED_CONFIRMAR, LOW);

    entradaMorse = ""; // Reinicia el código Morse para la próxima
letra
    delay(500);
  }
}
```

Este código hace que puedas escribir letras usando código Morse con botones en un Arduino. Cuando mantienes un botón presionado, el programa mide cuánto tiempo lo tienes apretado para decidir si es un punto o una raya. Luego, ese código se guarda en una cadena llamada “entradaMorse”. Si presionas otro botón para confirmar, el programa busca la letra que corresponde a ese código y la muestra en una pantalla LCD. También prende un LED para indicar

que la letra se ha guardado. Luego de eso, el código Morse se reinicia para que puedas escribir la siguiente letra.

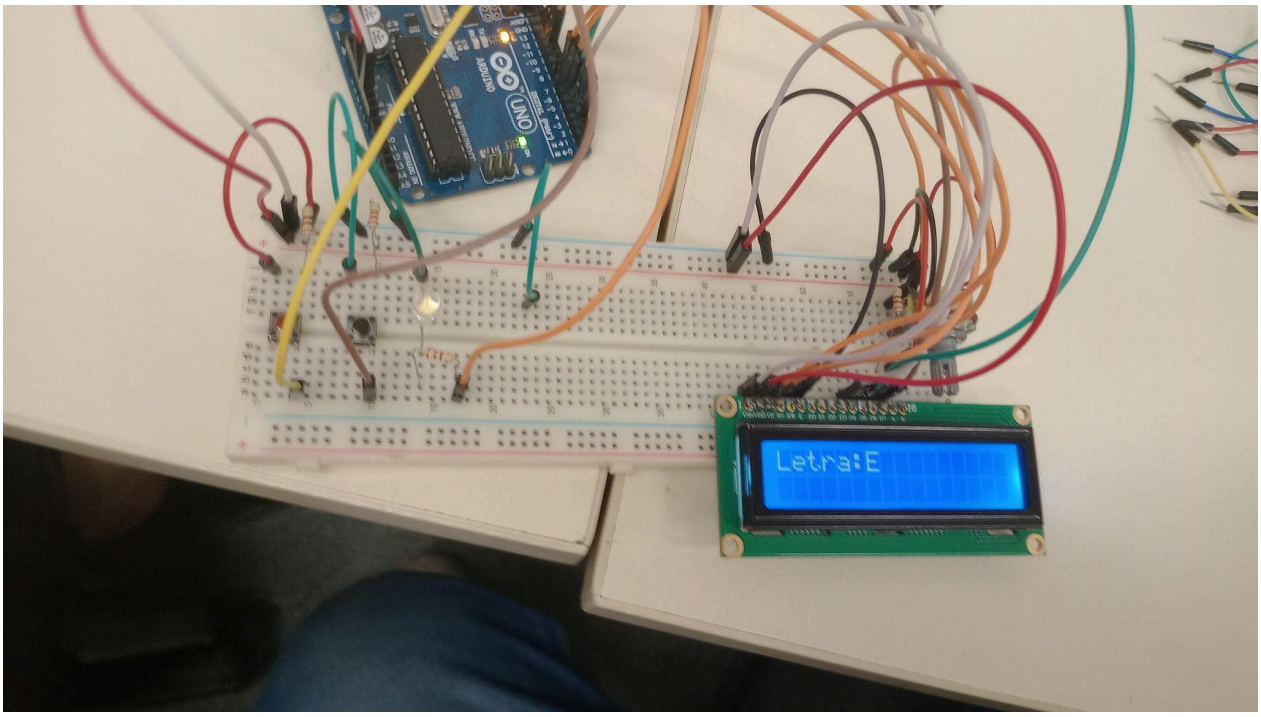
El for sirve para recorrer el array alfabetoMorse, que tiene todas las letras con su código Morse. Lo que hace es comparar lo que escribiste en Morse (morse) con cada código del array. Si encuentra uno igual, devuelve la letra correspondiente y termina. Si no encuentra ninguno después de revisar todo, devuelve un signo de interrogación ("?") para decir que el código no es válido o no existe. Es como un buscador que revisa letra por letra.

La función manejarEntradaMorse se encarga de leer las pulsaciones del botón asignado para ingresar el código Morse. Primero, detecta si el botón está presionado (digitalRead(PIN_BOTON_MORSE) == LOW) y luego mide el tiempo que se mantiene presionado con pulseIn. Si la pulsación dura menos que un tiempo definido como "punto" (DURACION_PUNTO), agrega un punto (".") al código Morse. Si la pulsación dura más, pero no tanto como para ser una raya, se agrega una raya ("-"). Después de registrar el punto o la raya, hace una pausa de 300 milisegundos antes de que puedas presionar el botón nuevamente.

La función manejarConfirmarLetra se encarga de cuando presionas el botón de confirmación para validar la letra que has escrito en Morse. Si el botón se presiona (digitalRead(PIN_BOTON_CONFIRMAR) == LOW), se convierte el código Morse ingresado en una letra usando la función morseACaracter(). Luego, la letra es mostrada en el LCD. Además, para confirmar la acción, se enciende un LED durante medio segundo y luego se apaga.

Finalmente, se limpia el código Morse (entradaMorse = "";) para empezar a escribir una nueva letra y se agrega un pequeño retraso antes de que puedas hacer otra acción.

La línea return resultado; en la función morseACaracter sirve para devolver la letra que corresponde al código Morse que ingrese. La función busca el código Morse en una lista; si lo encuentra, guarda la letra en resultado. Luego, return resultado; manda esa letra de vuelta al programa principal, donde se llamó la función. Si no encuentra nada que coincida, resultado queda en "?", lo que significa que el código ingresado no se reconoce como una letra válida.



Aca solo podía ingresar letras así que tuve que hacer cambios para que pueda funcionar, Porque no había implementado el espacio y porque solo me dejaba mandar solo una letra.

```
void manejarEspacio()
```

```
{
  if (digitalRead(PIN_BOTON_ESPACIO) == LOW) {
    delay(20);
    if (digitalRead(PIN_BOTON_ESPACIO) == LOW) {
      palabraActual += " ";

      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Palabra:");
      lcd.setCursor(0, 1);
      lcd.print(palabraActual);

      digitalWrite(PIN_LED_CONFIRMAR, HIGH);
      delay(500);
      digitalWrite(PIN_LED_CONFIRMAR, LOW);
      delay(500);
    }
  }
}

String morseACaracter(String morse) {
  String resultado = "?";
  int tamanoAlfabeto = sizeof(codigosMorse) /
sizeof(codigosMorse[0]); // Tamaño del array
  for (int i = 0; i < tamanoAlfabeto; i++) {
    if (codigosMorse[i] == morse) {
      resultado = letras[i];
      break;
    }
  }
  return resultado;
}
```

Después también elimine el botón de espacio que previamente se usaba para separar palabras en el código Morse. En su lugar, agregue la funcionalidad de detección de un espacio en el código Morse mismo. Ahora, si el usuario deja presionado el botón de enviar el dato por un tiempo más

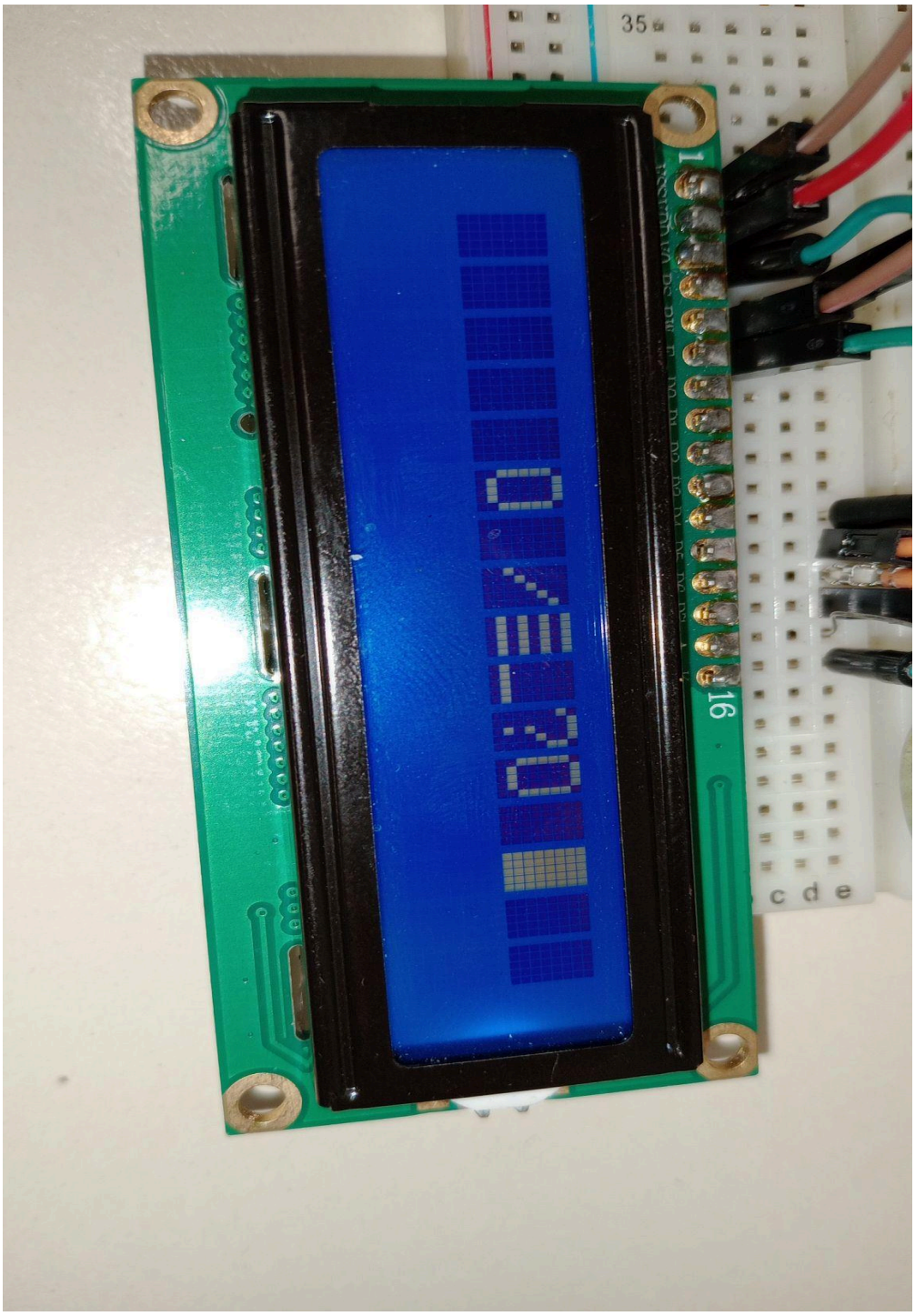
largo, el sistema lo interpretará como un espacio entre palabras. Esta duración es controlada por la variable DURACION_ESPACIO. Si se detecta un tiempo de pulsación adecuado, el código Morse agrega un espacio " " a la palabra actual. Esta modificación elimina la necesidad de un botón separado para ingresar espacios, haciendo que la entrada sea más eficiente y fluida.

Aca agregue el espacio para que funcione en el código manejarEspacio se encarga de detectar cuando presionas el botón asignado para poner un espacio entre palabras. Primero, verifica si el botón está presionado, y si lo está, agrega un espacio a la palabra que estás escribiendo. Luego, actualiza la pantalla LCD para mostrar la nueva palabra con el espacio añadido. Además, prende un LED por medio segundo para confirmar que se registró el espacio, y después apaga el LED. Todo esto tiene un pequeño retraso para evitar que el botón se registre más de una vez en un solo clic.

```
// Función que busca en el array de CodigoMorse para encontrar la letra
correspondiente
String morseACaracter(String morse) {
  int tamanoAlfabeto = sizeof(alfabetoMorse) /
sizeof(alfabetoMorse[0]); // Tamaño del array
  for (int i = 0; i < tamanoAlfabeto; i++) {
    if (alfabetoMorse[i].codigo == morse) {
      return alfabetoMorse[i].letra;  // Retorna la letra si hay
coincidencia
    }
  }
  return "?";  // Retorna '?' si el código no es reconocido
}
```

Este código convierte un mensaje en Morse (hecho con puntos y rayas) a su letra correspondiente. Básicamente, busca el código dentro de un arreglo llamado alfabetoMorse, que

tiene pares de códigos y letras. Si lo encuentra, devuelve la letra, y si no, devuelve un "?" para decir que no reconoce el código.



Probe si funcionaba el código pero no me funcionaba en el arduino, en el tinkercad si me funcionaba. Algo habré hecho mal, o por ahí conecte algo mal.

Este código ahora tiene la capacidad de manejar tanto letras como números en el código Morse. Los números 0-9 se agregaron al array codigosMorse[], y el sistema automáticamente los manejará como parte de la conversión de Morse a texto. Al confirmar una letra o número, se muestra en el LCD y se enciende un LED como confirmación.

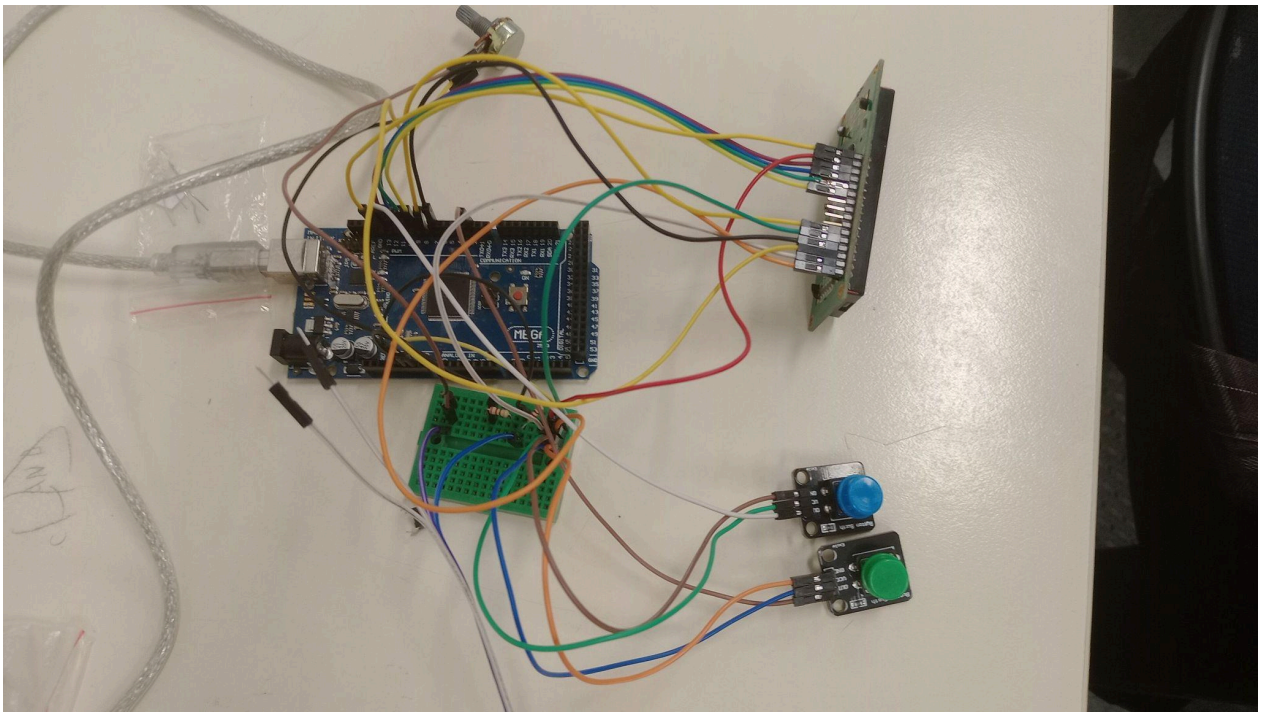
```
void manejarConfirmarEliminarOEspacio() {
  if (digitalRead(PIN_BOTON_CONFIRMAR) == LOW) {
    delay(20);
    unsigned long tiempoInicio = millis();

    while (digitalRead(PIN_BOTON_CONFIRMAR) == LOW) {
      if (millis() - tiempoInicio > TIEMPO_ELIMINAR) {
        if (palabraActual.length() > 0) {
          palabraActual.remove(palabraActual.length() - 1);
          lcd.clear();
          lcd.setCursor(0, 0);
          lcd.print("Palabra:");
          lcd.setCursor(0, 1);
          lcd.print(palabraActual);
        }
        delay(500);
        return;
      }
    }

    if (entradaMorse == "") {
      palabraActual += " ";
    } else {
      String letra = morseACaracter(entradaMorse);
      if (letra != "?") {
```

```
        palabraActual += letra;
      }
    }

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Palabra:");
    lcd.setCursor(0, 1);
    lcd.print(palabraActual);

    digitalWrite(PIN_LED_CONFIRMAR, HIGH);
    delay(500);
    digitalWrite(PIN_LED_CONFIRMAR, LOW);

    entradaMorse = "";
    delay(500);
  }
}
```

En el código de Arduino, usamos unsigned long para manejar números grandes que solo pueden ser positivos. Esto es súper útil cuando queremos medir tiempos con la función millis(), que nos dice cuántos milisegundos han pasado desde que encendimos el Arduino. Por ejemplo, nos sirve para calcular cuánto tiempo se mantuvo presionado un botón y decidir si es un punto o una raya en código Morse. Es como un cronómetro dentro del programa, y nos permite trabajar con tiempos sin interrumpir otras cosas que el Arduino esté haciendo al mismo tiempo.

Teniendo el código funcionando con la confirmación de letras y la adición de espacios, decidimos agregar una función para eliminar la última letra que se muestra en el LCD. En lugar de usar un botón extra, lo hicimos con el mismo botón de confirmación. La idea es que si mantienes presionado el botón durante más de un segundo, se eliminará la última letra de la palabra que se

está mostrando en la pantalla. Esto lo logramos con un control de tiempo que, si el botón permanece presionado, elimina el último carácter de la cadena usando el método remove(). Así, la palabra en el LCD se actualiza para reflejar el cambio, y se evita agregar otro botón físico extra, simplificando el diseño.



En este día estuvimos trabajando con otro Arduino además del principal, ya que necesitábamos uno adicional para que recibiera los datos enviados desde el proyecto y los mostrara mediante una conexión Bluetooth. La idea era transmitir las palabras o letras en código Morse procesadas en el primer Arduino hacia el segundo, utilizando el módulo Bluetooth como enlace entre ambos, lo que amplía la funcionalidad del sistema y permite la visualización o el uso de los datos en un dispositivo remoto.

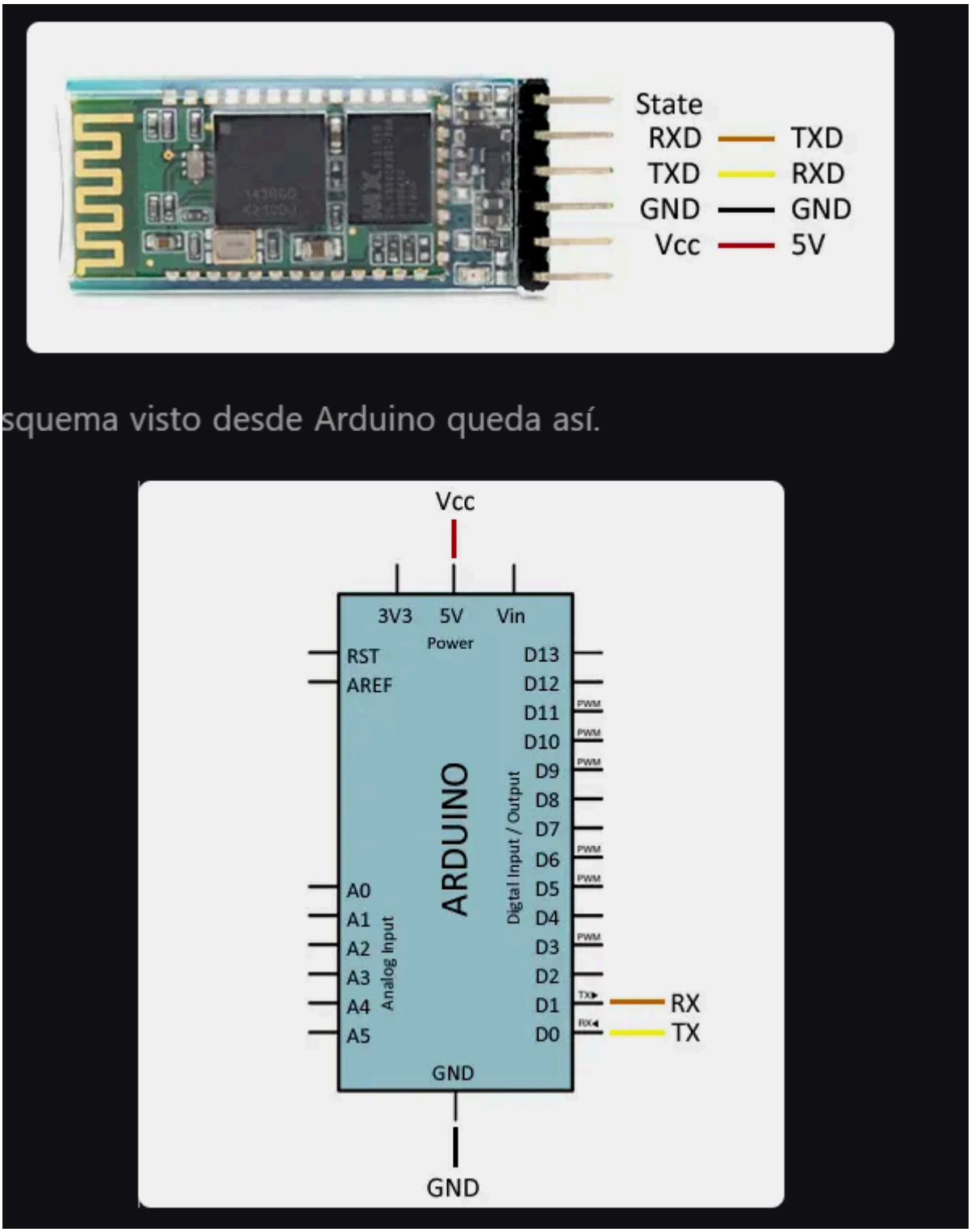
El segundo Arduino toma estos datos, como las palabras procesadas en código Morse, y los muestra en una pantalla LCD conectada. Para esto, usamos un código que configura el módulo Bluetooth con funciones básicas para enviar y recibir datos. Por ejemplo, se utiliza Serial1.print() para transmitir los datos desde el Arduino principal, y el segundo Arduino, al recibirlos con Serial1.read(), los procesa para mostrarlos en la pantalla. Esto nos permite completar la comunicación entre ambos dispositivos de manera inalámbrica.

El código Bluetooth en este proyecto se usa para enviar y recibir información de forma inalámbrica entre dos Arduinos. Usamos un módulo como el HC-05 para la conexión. En el Arduino que envía los datos (el emisor), se utiliza Serial.print() para mandar las letras o símbolos generados en Morse, y en el Arduino receptor, usamos Serial.read() para recibir esa información.

Básicamente, el Bluetooth se maneja con tres funciones importantes: Serial.begin() que inicia la comunicación, Serial.print() para enviar los datos y Serial.read() para recibirlos. El Bluetooth se conecta a los pines RX y TX del Arduino, lo que permite transmitir y recibir datos sin cables.

Cuando estuvimos trabajando con los módulos Bluetooth para que los dos Arduinos se comunicaran, no logramos que funcionara. Intentábamos enviar palabras desde uno y recibirlas en el otro, pero algo no estaba bien. Revisamos las conexiones, el código, y hasta intentamos cambiar la configuración del *baud rate*, pero seguía sin responder. Fue bastante frustrante porque no encontrábamos dónde estaba el error, y después de varios intentos tuvimos que dejarlo para

seguir probando otro día. Aunque no salió como esperábamos, entendimos más sobre cómo funcionan los módulos y qué cosas revisar en el futuro.



Le pedí a Chat GPT que me explicara cómo configurar el Bluetooth en Arduino, específicamente con el módulo HC-05. Me explicó que este dispositivo se puede configurar como maestro o esclavo utilizando comandos AT. Para eso, hay que entrar en el **modo AT**, conectando el pin KEY/EN a 3.3V y enviando los comandos desde el monitor serie del Arduino IDE. Me enseñó cómo cambiar entre los modos maestro (AT+ROLE=1) y esclavo (AT+ROLE=0), además de modificar el nombre del módulo (AT+NAME=NuevoNombre) y emparejar dispositivos en el modo maestro.

Aunque entendimos el proceso, no llegamos a verificar si realmente el Bluetooth estaba funcionando bien, así que quedó pendiente probarlo más a fondo y asegurarnos de que la comunicación entre los dispositivos sea correcta.

En estos días, lo único que estuvimos haciendo fue armar los dos arduinos con sus conexiones y tratar de que el Bluetooth funcionara. Nos dimos cuenta de que no sabíamos que había que usar comandos AT para configurar los módulos HC-05 como maestro y esclavo, y eso es súper importante porque sin esa configuración no se comunican entre ellos. Fue como darnos cuenta de algo básico que nos faltaba saber, y al final no pudimos verificar si todo estaba funcionando porque nos quedamos en esa parte.

A continuación voy a poner un código que permite establecer comunicación entre dos Arduinos utilizando módulos Bluetooth. Este código se divide en dos partes: una para enviar datos y otra para recibirlos. El primer Arduino tomará información (por ejemplo, caracteres o mensajes) desde el monitor serie y la enviará a través de Bluetooth. El segundo Arduino, por su parte, estará configurado para recibir esos datos y mostrarlos en su propio monitor serie.

```
#include <SoftwareSerial.h> // Librería para comunicación serie por
pines distintos a RX/TX

SoftwareSerial bluetooth(10, 11); // Definir pines para comunicación
Bluetooth (RX, TX)

void setup() {
  Serial.begin(9600); // Inicializa el puerto serie para la PC a 9600
baudios
  bluetooth.begin(9600); // Inicializa el módulo Bluetooth a 9600
baudios
  Serial.println("Bluetooth listo para enviar."); // Mensaje en el
monitor serie
}

void loop() {
  if (Serial.available()) { // Verifica si hay datos ingresados desde
el monitor serie
    char data = Serial.read(); // Lee un carácter del monitor serie
    bluetooth.print(data); // Envia el carácter al módulo Bluetooth
    Serial.print("Enviado: "); // Muestra en el monitor serie el dato
enviado
    Serial.println(data);
  }
}
```

Este código permite al Arduino enviar datos al módulo Bluetooth HC-05 o HC-06. Los datos se escriben en el monitor serie de la computadora, el Arduino los lee y los transmite mediante el módulo Bluetooth. Esto podría ser útil, por ejemplo, para enviar comandos o mensajes a otro Arduino o dispositivo conectado al Bluetooth.

```
SoftwareSerial bluetooth(10, 11);
void setup() {
 Serial.begin(9600); // Inicializa el puerto serie para la PC a 9600 baudios
 bluetooth.begin(9600); // Inicializa el módulo Bluetooth a 9600 baudios
 Serial.println("Esperando datos por Bluetooth..."); // Mensaje en el monitor serie
}

void loop() {
 if (bluetooth.available()) { // Verifica si hay datos recibidos desde Bluetooth
  char data = bluetooth.read(); // Lee un carácter del Bluetooth
  Serial.print("Recibido: "); // Muestra en el monitor serie el dato recibido
  Serial.println(data);
 }
}
```

Esto configura el Arduino para recibir datos a través del módulo Bluetooth y mostrarlos en el monitor serie. Por ejemplo, si otro dispositivo envía una letra, esta se mostrará en el monitor. Es útil para pruebas o para construir sistemas de comunicación más avanzados entre dispositivos.

En este proyecto estuvimos trabajando con dos Arduinos que se comunican entre sí usando módulos Bluetooth. Configuramos cada Arduino para que pueda enviar y recibir datos, permitiendo que una palabra ingresada en uno se muestre en el otro. Esto implicó aprender sobre el código Morse, manejar pantallas LCD y entender cómo funciona la configuración maestro-esclavo en los módulos Bluetooth. Aunque enfrentamos problemas al inicio, como no

saber sobre los comandos AT para configurar los módulos, logramos avanzar y desarrollar un sistema funcional.