UNIDAD ACADÉMICA MULTIDISCIPLINARIA MANTE

EXAMEN PARCIAL 1

6° J

PROGRAMACION DE INTERFACES Y PUERTOS

Lopez Piña Daniel

Juarez Mayorga Ximena Guadalupe

El Mante, Tamaulipas, México. **Febrero 2025**

EJERCICIO 13

En esta actividad, se empleó un sensor de temperatura analógico conectado a un Arduino UNO con la finalidad de captar la temperatura del entorno y visualizar los datos obtenidos en la interfaz de monitor serie del software de Arduino. Este experimento permitió explorar el comportamiento de los sensores analógicos y su interacción con un microcontrolador.

El primer paso consistió en ubicar el sensor en una protoboard, asegurándonos de que quedara bien fijado y dejando espacio suficiente para conectar los cables de manera adecuada. Posteriormente, se llevaron a cabo las conexiones eléctricas necesarias para su correcto funcionamiento.

El terminal izquierdo del sensor se vinculó a la salida de 5V del Arduino para suministrarle energía. El pin central, responsable de transmitir la señal analógica con la lectura de temperatura, se enlazó con la entrada A0 del Arduino, encargada de procesar este tipo de señales. Finalmente, el terminal derecho se conectó a GND para cerrar el circuito eléctrico.

Una vez realizadas las conexiones, se procedió a programar el Arduino. Se desarrolló un código en el IDE de Arduino que permitía captar los valores del sensor, transformarlos en grados Celsius mediante una ecuación específica y posteriormente desplegarlos en la consola del monitor serie en tiempo real.

Para evaluar la efectividad del sistema, se realizaron diversas pruebas bajo distintas condiciones ambientales. Se verificó que las mediciones variaban en función de la temperatura del entorno, lo que confirmó la precisión del sensor. Asimismo, se implementaron ajustes en el código para optimizar la presentación de los datos, incluyendo mensajes descriptivos y un formato más estructurado.

Esta práctica facilitó el aprendizaje sobre la interpretación de señales analógicas, el uso del puerto serie para la visualización de información y la organización de un circuito en una protoboard. Además, sirvió como introducción a la conversión de valores eléctricos en información comprensible, permitiendo su aplicación en proyectos más avanzados.

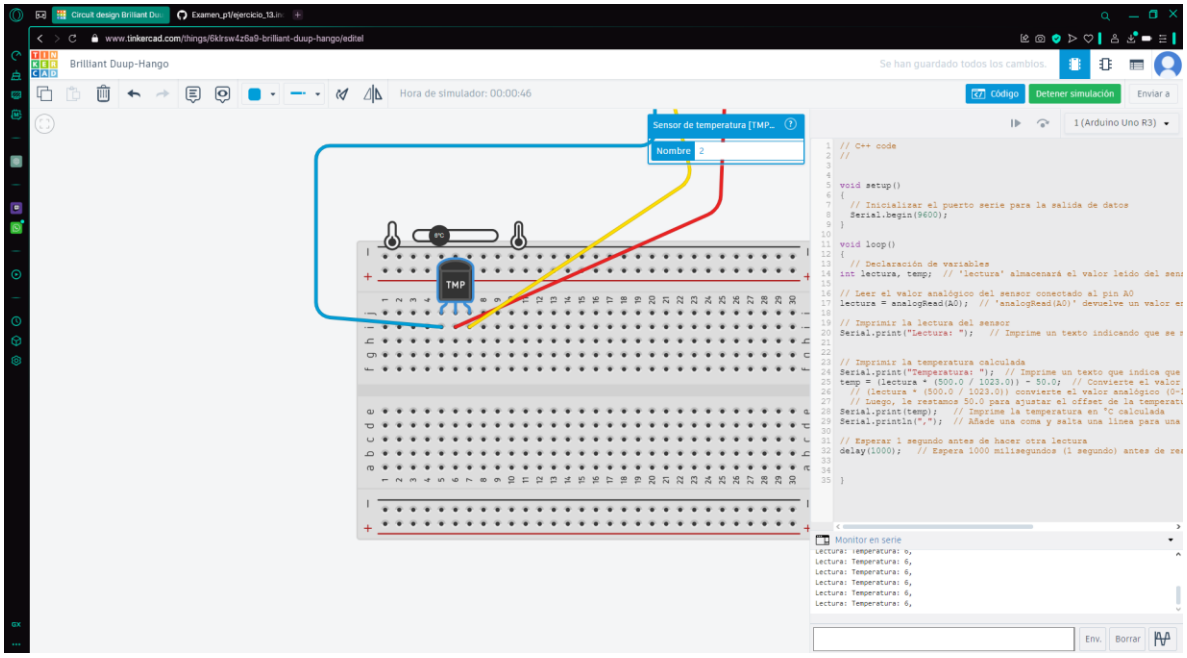
CÓDIGO

```
void setup()
{
  // Se inicia la comunicación serie para poder ver los valores en el monitor serie
  Serial.begin(9600);
}

void loop()
{
  // Se declaran variables para almacenar los datos
  int lectura, temp;

  // Se obtiene la lectura del sensor desde el pin A0
  lectura = analogRead(A0); // Devuelve un valor entre 0 y 1023

  // Se muestra la lectura del sensor en el monitor serie
  Serial.print("Lectura del sensor: ");
  Serial.print(lectura);

  // Se convierte la lectura a temperatura en grados Celsius
  temp = (lectura * (500.0 / 1023.0)) - 50.0;
  Serial.print("  Temperatura: ");
  Serial.print(temp);
  Serial.println(" °C");  // Se imprime la temperatura con su unidad

  // Se espera 1 segundo antes de la siguiente lectura para evitar valores muy seguidos
```

```
  delay(1000);
}
```

Como paso final, se cargó el código en la placa Arduino y se accedió al monitor serie del IDE para visualizar en tiempo real los valores registrados por el sensor de temperatura. Durante la prueba, se verificó que el sensor analógico genera un valor numérico en bruto que, al aplicarle una ecuación matemática específica, se convierte en una lectura en grados Celsius, permitiendo interpretar con precisión la temperatura del entorno.

Asimismo, se destacó la relevancia de la comunicación serie como una herramienta esencial para la visualización y análisis de los datos capturados por el sensor en tiempo real. También se comprendió la importancia de aplicar conversiones matemáticas adecuadas, ya que los valores originales proporcionados por el sensor requieren ajustes para que la información obtenida sea clara y útil en distintas aplicaciones. Esta práctica permitió afianzar conocimientos sobre el procesamiento de señales analógicas en microcontroladores y la correcta interpretación de mediciones para obtener resultados fiables y precisos.



EJERCICIO 14

Para iniciar con la configuración del circuito, se dispuso una protoboard en la superficie de trabajo y se extrajo un Arduino UNO R3, asegurándose de que todos los componentes estuvieran listos para ser instalados. Antes de conectar los dispositivos principales, se estableció la alimentación del circuito conectando el puerto GND del Arduino a una de las líneas de distribución negativa de la protoboard. Esto permitió que todos los componentes compartieran un mismo punto de referencia para el voltaje, asegurando un funcionamiento adecuado y evitando posibles fallos en la conexión.

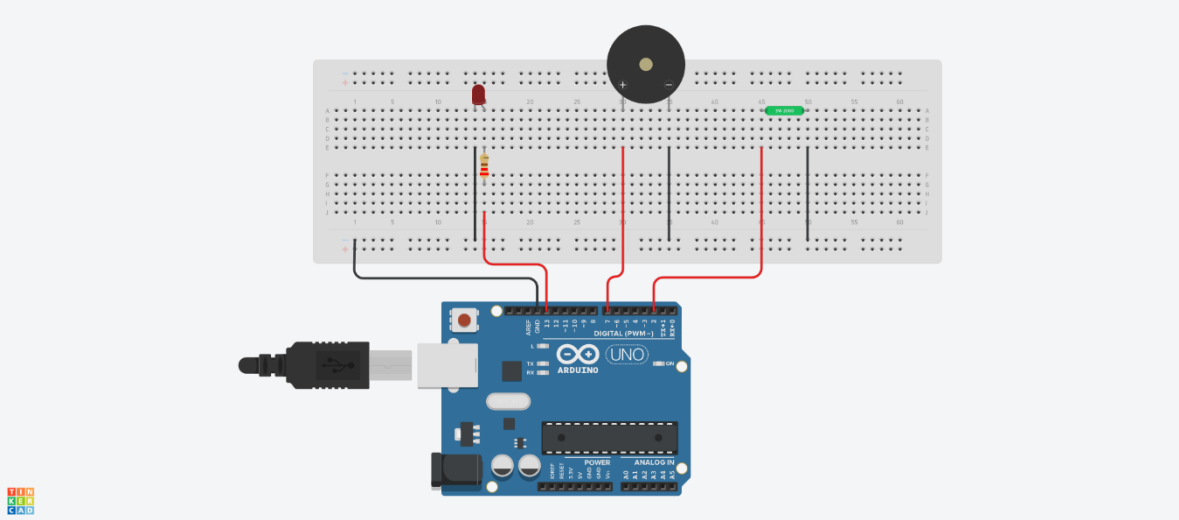
El siguiente paso fue la integración de un diodo LED, el cual se colocó en las coordenadas A14 y A15 de la protoboard. Para regular el flujo de corriente que pasaría a través del LED y evitar que se quemara debido a una sobrecarga, se añadió una resistencia de 220Ω en las posiciones E15 y G15. Esta resistencia quedó conectada en serie con la terminal positiva del LED, funcionando como un limitador de corriente. Posteriormente, se establecieron las conexiones necesarias para que el LED pudiera encenderse y apagarse mediante el Arduino: el lado positivo del LED fue conectado desde la coordenada J15 hasta el puerto digital 13 del Arduino, mientras que la terminal negativa se dirigió a la línea de tierra de la protoboard.

Una vez configurado el LED, se procedió a instalar un buzzer en las posiciones A30 y A35 de la protoboard. Para mantener el orden y la organización dentro del circuito, se decidió alinear este componente con el LED, lo que facilitó la distribución de los cables y conexiones. El buzzer se conectó de la siguiente manera: su terminal positiva fue enlazada al puerto digital 7 del Arduino, permitiendo que el microcontrolador pudiera activarlo cuando fuera necesario, mientras que la terminal negativa se vinculó directamente a la línea de tierra de la protoboard, asegurando que el componente estuviera correctamente integrado en el sistema.

Después de la instalación del buzzer, se incorporó un sensor de inclinación, que se colocó en las coordenadas A45 y A50 de la protoboard. Para optimizar la disposición del circuito y evitar cruces innecesarios de cables, se alineó este sensor con el LED y el buzzer. Su conexión al Arduino se realizó de la siguiente manera: el terminal positivo del sensor fue enlazado al puerto digital 2 del Arduino, mientras que el terminal negativo fue dirigido hacia la línea de tierra de la protoboard, asegurando su correcta funcionalidad.

Con todas las conexiones establecidas, el circuito quedó completamente armado y listo para la fase de programación. En este punto, se verificó que cada componente estuviera correctamente conectado y alineado para evitar posibles errores al cargar el código en el Arduino. Este montaje permitió entender mejor la importancia de una

distribución ordenada de los elementos en la protoboard, facilitando tanto el ensamblaje como la identificación de posibles fallos en el sistema. Además, se reforzó la comprensión sobre la manera en que los diferentes componentes electrónicos interactúan con el microcontrolador, lo que resultará útil en futuros proyectos y aplicaciones más complejas.



CODIGO

Descripción del código y como funcionan cada uno de sus componentes.

```
void setup() {
```

//Configura el pin 2 como una entrada con resistencia interna pull-up. Esto significa que el pin 2 esta configurado para leer el estado de un botón y la resistencia pull-up lo mantiene en un valor alto (HIGH) cuando el botón no está presionado.

```
 pinMode(2, INPUT_PULLUP);
```

//Configura el pin 7 como salida. Este pin está destinado a controlar un LED u otro dispositivo.

```
 pinMode(7, OUTPUT);
```

//Configura el pin 13 como salida, otro pin para controlar un LED u otro dispositivo.

```
 pinMode(13, OUTPUT);
}

void loop ()    {
```

//Lee el valor del pin 2. Si el valor es alto (HIGH), eso significa que el botón no está presionado.

```
 if (digitalRead(2) == HIGH) {
```

//Enciende el LED conectado al pin 13.

```
  digitalWrite(13, HIGH);
```

//Apaga el LED conectado al pin 7.

```
        digitalWrite(7, LOW);                  }
```

//Si el valor leído del pin 2 es bajo (LOW) es porque el botón está presionado.

```
 else {
```

```
  //Apaga el LED conectado al pin 13.
  digitalWrite(13, LOW);
  //Enciende el LED conectado al pin 7.
      digitalWrite(7, HIGH);                }
}
```

Una vez implementado el código y ensamblado el circuito con todos los componentes, se procede a ejecutar la simulación. En un inicio, el LED permanecerá encendido, indicando el estado normal del sistema. Sin embargo, al interactuar con el sensor de inclinación y moverlo gradualmente, la intensidad de la luz del LED comenzará a disminuir progresivamente.

Cuando el sensor alcanza cierto ángulo de inclinación, el LED se apagará por completo, activando inmediatamente el buzzer. Este emitirá un sonido similar al de una "chicharra" mediante vibraciones continuas. El buzzer permanecerá encendido hasta que el sensor de inclinación regrese a su posición original, momento en el cual el LED recuperará su brillo y el sonido del buzzer se detendrá, restableciendo así el estado inicial del sistema.

EJERCICIO 15

Para llevar a cabo esta práctica, se diseñó un sistema que combinaba un sensor de ultrasonido HC-SR04 con un Arduino UNO, cuyo propósito era medir distancias y generar una señal visual que indicara cuándo un objeto se encontraba a una distancia menor a 10 cm. El sistema estaba configurado para activar un LED como indicador visual en el momento en que un objeto se detectaba dentro de este umbral de distancia. Esta práctica no solo permitió conocer el funcionamiento de los sensores de ultrasonido, sino también cómo integrar estos componentes con un microcontrolador como el Arduino para crear una respuesta interactiva.

El proceso comenzó con la disposición de los componentes sobre una protoboard, lo que permitió realizar las conexiones de manera ordenada y segura. El sensor de ultrasonido HC-SR04, que consta de cuatro pines con funciones bien definidas, fue el primer componente en ser conectado. El pin **VCC** se conectó al pin de 5V del Arduino, lo que permitió suministrar la energía necesaria para el funcionamiento del sensor. Por otro lado, el pin **GND** se conectó al pin GND del Arduino, completando el circuito eléctrico del sensor. El pin **Trigger**, encargado de enviar una señal al sensor para iniciar el proceso de medición, se conectó al pin digital 10 del Arduino. Finalmente, el pin **Echo** se enlazó al pin digital 11 del Arduino, responsable de recibir la señal reflejada que, a su vez, permitirá calcular la distancia entre el sensor y el objeto. De esta forma, el sensor de ultrasonido estaba listo para medir la distancia en función del tiempo que tarda la señal en ir y regresar al sensor.

El siguiente paso fue la instalación de un LED en la protoboard. Este componente visual se utilizó como indicador de que un objeto se encontraba dentro del rango de distancia establecido. Para evitar daños por una corriente excesiva, se incluyó una resistencia de 220 ohmios en serie con el LED. El **ánodo** del LED se conectó al pin digital 13 del Arduino, que es el encargado de enviar la señal de encendido o apagado al LED. El **cátodo**, por su parte, se dirigió al pin GND de la protoboard, completando el circuito y asegurando el funcionamiento del LED.

Una vez que las conexiones estuvieron listas, se procedió a cargar el código en el Arduino. El programa que se escribió tenía la finalidad de medir la distancia utilizando el sensor de ultrasonido, y si esta distancia era inferior a 10 cm, encendería el LED. De esta manera, el sistema cumplía su función de proporcionar una señal visual clara en respuesta a la proximidad de un objeto. Además, el código permitió que el Arduino calculase el tiempo que tardaba la señal de ultrasonido en regresar al sensor y, en base a ese tiempo, calculaba la distancia mediante una fórmula matemática simple, lo que facilitó la creación de un sistema de medición preciso.

Este ejercicio no solo ayudó a comprender el funcionamiento básico del sensor de ultrasonido HC-SR04, sino también cómo interactuar con otros componentes como el LED a través de un microcontrolador. La práctica también brindó una introducción a la programación de Arduino, el cual permite realizar tareas de medición y control en sistemas electrónicos de manera sencilla. Además, aprendimos la importancia de realizar las conexiones correctamente para evitar fallos en el sistema y cómo pequeñas variaciones en el código pueden alterar el comportamiento de los componentes conectados. En resumen, este proyecto proporcionó una valiosa experiencia en la integración de sensores, microcontroladores y componentes visuales para crear sistemas interactivos y prácticos.

CODIGO

```
const int Trigger = 10;    // Pin digital 10 para enviar la señal de activación del sensor
const int Echo = 11;       // Pin digital 11 para recibir el eco del sensor
const int Led = 13;        // Pin digital 13 para controlar el encendido del LED

void setup() {
  Serial.begin(9600);      // Configura la comunicación serie a 9600 baudios
  pinMode(Trigger, OUTPUT); // Configura el pin Trigger como salida
  pinMode(Echo, INPUT);    // Configura el pin Echo como entrada
  pinMode(Led, OUTPUT);    // Configura el pin LED como salida
  digitalWrite(Trigger, LOW); // Inicializa el pin Trigger en estado bajo
}

void loop() {
  long t; // Variable para almacenar el tiempo del eco
  long d; // Variable para almacenar la distancia medida en centímetros

  digitalWrite(Trigger, HIGH); // Envia señal alta al Trigger para iniciar la medición
  delayMicroseconds(10);   // Pausa de 10 microsegundos para asegurar el pulso
  digitalWrite(Trigger, LOW); // Apaga el Trigger para finalizar el pulso

  t = pulseIn(Echo, HIGH); // Lee el tiempo que tarda el pulso en regresar
  d = t / 59;              // Calcula la distancia en centímetros usando el tiempo del eco

  Serial.print("Distancia: ");
  Serial.print(d);         // Muestra la distancia en la consola serie
```

```
  Serial.println(" cm");


  if (d <= 10) {
    digitalWrite(Led, HIGH);  // Enciende el LED si la distancia es igual o menor a 10 cm
  } else {
    digitalWrite(Led, LOW);   // Apaga el LED si la distancia es mayor a 10 cm
  }


  delay(100);  // Espera de 100ms antes de realizar la siguiente medición
}
```

Una vez que todas las conexiones estuvieron realizadas y el código cargado, se procedió a abrir el Monitor Serie del IDE de Arduino para visualizar las mediciones en tiempo real. Para verificar el funcionamiento del sistema, se colocaron varios objetos frente al sensor y se observó la respuesta del sistema. Al acercar un objeto a una distancia menor a 10 cm del sensor, el LED se encendió de manera correcta, confirmando que el sistema estaba funcionando como se esperaba.