UNIDAD ACADÉMICA MULTIDISCIPLINARIA MANTE

Examen 1° Parcial

6° J

Vázquez Reyna Ángel Levi

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

Examen 1° Parcial

Ejercicio 13

En esta práctica, se utilizó un sensor de temperatura analógico conectado a un Arduino UNO con el objetivo de medir la temperatura ambiental y visualizarla en el monitor serie del IDE de Arduino.

Para comenzar, el sensor de temperatura se colocó en la protoboard, asegurándose de que estuviera bien sujeto y con suficiente espacio para las conexiones.

A continuación, se realizaron las conexiones necesarias:

El pin izquierdo del sensor se conectó a la salida de 5V del Arduino para su alimentación.

El pin central se conectó a la entrada analógica A0 del Arduino, ya que este es el encargado de enviar los datos de temperatura.

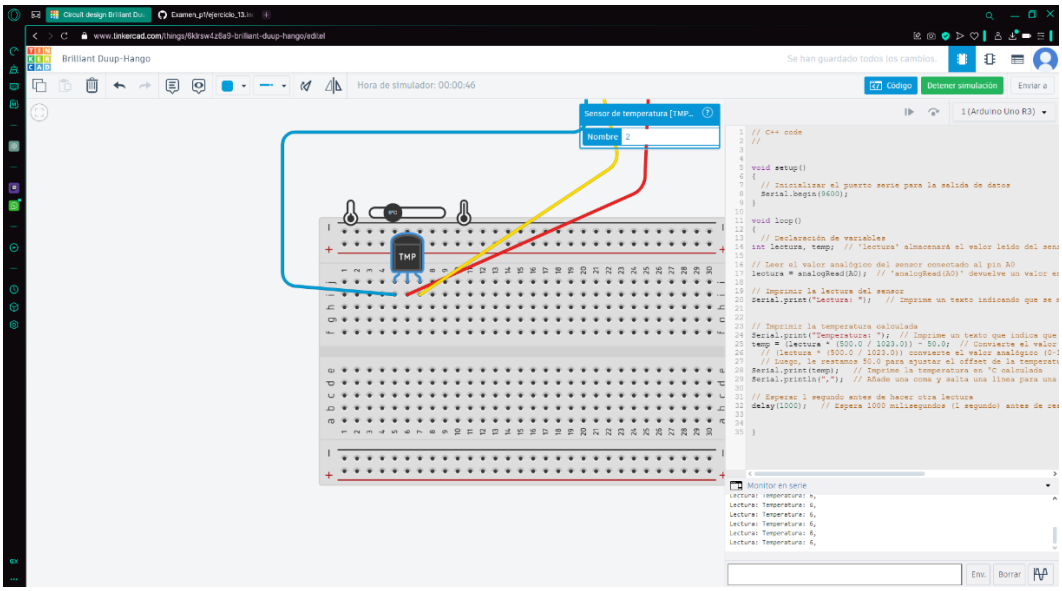
El pin derecho se conectó a GND para completar el circuito.

Una vez realizadas las conexiones, se procedió a escribir el código en el IDE de Arduino. Este programa se encargó de leer los valores del sensor, convertirlos en grados Celsius y mostrarlos en el monitor serie. El código utilizado fue el siguiente:

```
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);
}
```

Finalmente, se subió el código a la placa Arduino y se abrió el monitor serie del IDE para observar en tiempo real los valores de temperatura. Se verificó que el sensor analógico genera un valor numérico, el cual se transforma en grados Celsius utilizando una fórmula matemática. Además, se destacó la relevancia de la comunicación serie para la visualización de los datos y la importancia de aplicar conversiones adecuadas para interpretar correctamente las mediciones del sensor.



Ejercicio 14

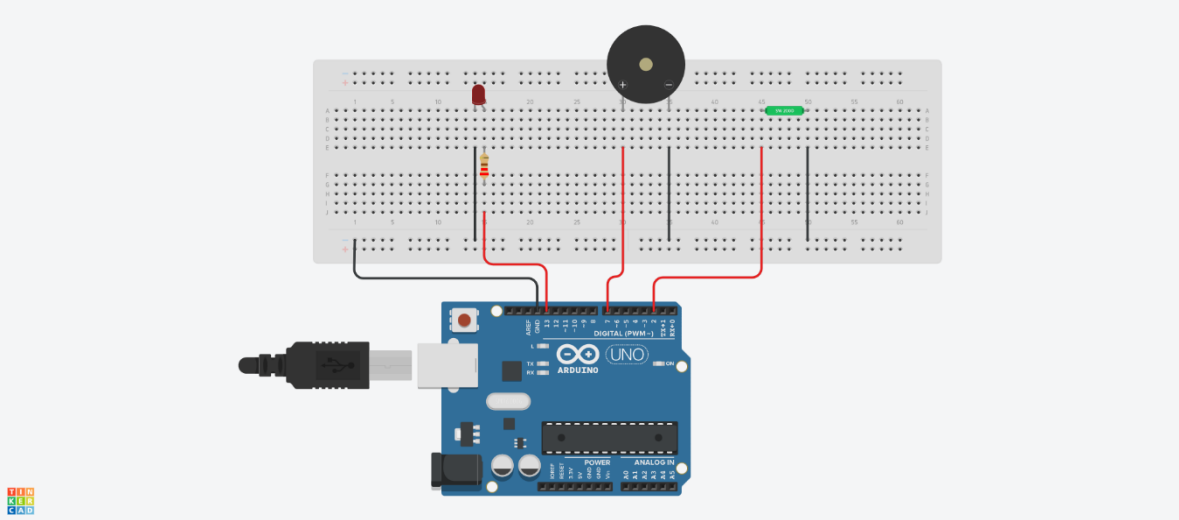
Para comenzar, se utilizó una placa de pruebas (protoboard) junto con un Arduino UNO R3. Se estableció una conexión desde el puerto GND del Arduino hacia una línea negativa de la protoboard.

A continuación, se colocó un LED en las coordenadas A14 y A15 de la protoboard, acompañado de una resistencia de 220Ω en las coordenadas E15 y G15. La resistencia se conectó al lado positivo del LED, mientras que el otro extremo de la resistencia, ubicado en J15, se conectó al puerto 13 del Arduino. Por otro lado, el lado negativo del LED se enlazó con la línea de tierra (negativa) de la protoboard.

Posteriormente, se instaló un buzzer en las coordenadas A30 y A35, asegurando una distribución organizada de los componentes. El terminal positivo del buzzer se conectó al puerto 7 del Arduino, mientras que el terminal negativo se dirigió a la línea de tierra.

Además, se incorporó un sensor de inclinación en las coordenadas A45 y A50, alineándolo con el LED y el buzzer para mantener una conexión ordenada. Se conectó el terminal positivo del sensor al puerto 2 del Arduino y el terminal

negativo a la línea de tierra de la protoboard.



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);                }
}
```

Con la implementación del código y el ensamblaje de los componentes, se procede a ejecutar la simulación. Inicialmente, el LED permanecerá encendido, pero al activar el sensor de inclinación, su luz comenzará a atenuarse gradualmente hasta apagarse por completo. En ese momento, el buzzer emitirá un sonido similar al de una "chicharra" y continuará sonando hasta que el sensor de inclinación vuelva a su posición original, permitiendo que el LED se encienda nuevamente.

Ejercicio 15

En esta práctica, se desarrolló un sistema para medir distancias utilizando un sensor de ultrasonido HC-SR04 y un Arduino UNO. El objetivo era encender un LED cuando un objeto estuviera a menos de 10 cm, funcionando como un indicador visual. Para ello, se programó el Arduino para realizar las mediciones de distancia y activar el LED según la condición establecida.

Para iniciar, se conectó el sensor de ultrasonido a la protoboard y se enlazó con el Arduino. El HC-SR04 tiene cuatro pines, los cuales se conectaron de la siguiente manera:

- **VCC**: al pin de 5V del Arduino para su alimentación.
- **GND**: al pin GND del Arduino.
- **Trigger**: al pin digital 10 del Arduino.
- **Echo**: al pin digital 11 del Arduino.

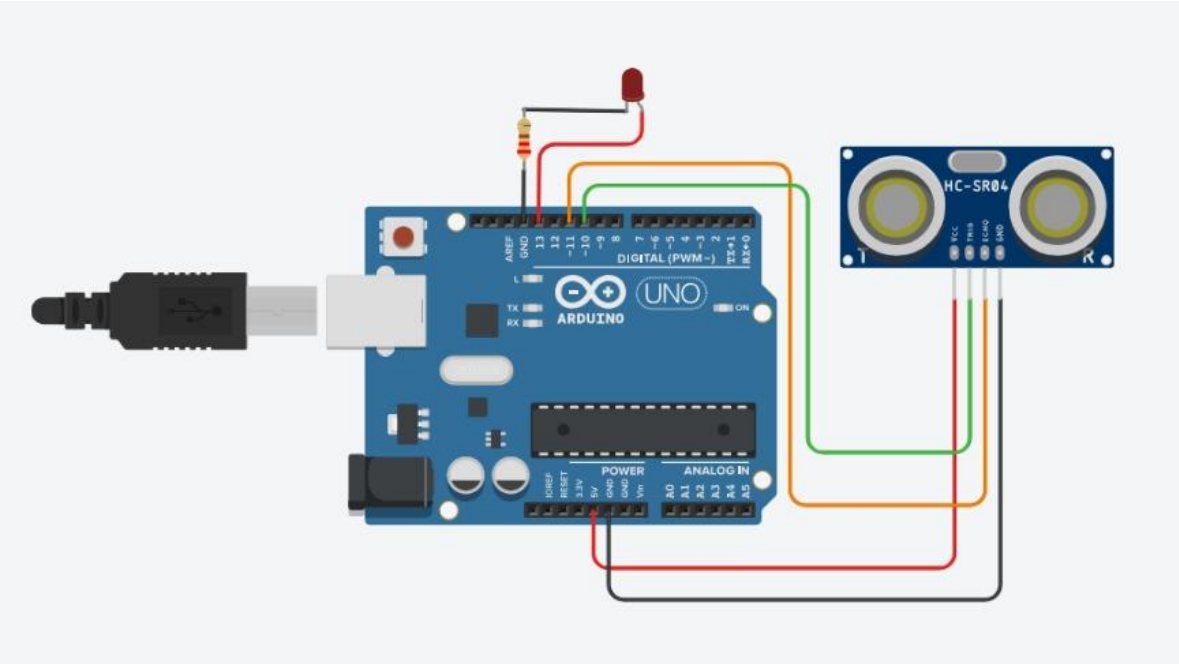
Además, se incorporó un LED en la protoboard con una resistencia de 220 ohmios en serie para proteger el circuito. El ánodo del LED se conectó al pin digital 13 del Arduino, mientras que el cátodo se unió a GND.

Una vez completadas las conexiones, se cargó el siguiente código en el Arduino.

```
const int Trigger = 10;   // Asignamos el pin digital 10 para el Trigger del sensor
const int Echo = 11;      // Asignamos el pin digital 11 para el Echo del sensor
const int Led = 13;       // Definimos el pin digital 13 para el LED

void setup() {
  Serial.begin(9600);      // Iniciamos la comunicación serial
  pinMode(Trigger, OUTPUT);// Configuramos el pin Trigger como salida
  pinMode(Echo, INPUT);    // Configuramos el pin Echo como entrada
  pinMode(Led, OUTPUT);    // Configuramos el pin del LED como salida
  digitalWrite(Trigger, LOW); // Establecemos el Trigger en estado LOW al inicio
}
```

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

  digitalWrite(Trigger, HIGH);
  delayMicroseconds(10);   // Generamos un pulso de 10 microsegundos
  digitalWrite(Trigger, LOW);

  t = pulseIn(Echo, HIGH); // Medimos la duración del pulso de respuesta
  d = t / 59;              // Convertimos el tiempo en distancia en centímetros

  Serial.print("Distancia: ");
  Serial.print(d);
  Serial.println(" cm");

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

  delay(100);  // Pausa breve de 100 milisegundos antes de la siguiente medición
}
```

Finalmente, se accedió al Monitor Serie del IDE de Arduino para visualizar las mediciones en tiempo real. Para probar el funcionamiento del sistema, se posicionaron objetos frente al sensor. Cuando un objeto se acercó a una distancia menor de 10 cm, el LED se encendió correctamente, confirmando el correcto desempeño del circuito.


HC-SR04
VCC
TRIG
ECHO
GND
DIGITAL (PWM~)
UNO
ARDUINO
TX
RX
ON
POWER
ANALOG IN