Unidad Académica Multidisciplinaria Mante

Examen Primer Parcial

PROGRAMACION DE INTERFACES Y PUERTOS

Alumna: Fernanda Marisela Montes Rivera

6°J

26/02/2025

Ejercicio 13

Para esta práctica, se trabajó con un sensor de temperatura analógico conectado a un Arduino UNO. La idea principal era poder leer la temperatura del ambiente y mostrarla en el monitor serie del IDE de Arduino.

Se comenzó colocando el sensor de temperatura en la protoboard, asegurándonos de que estuviera bien sujeto y con suficiente espacio para conectar los cables.

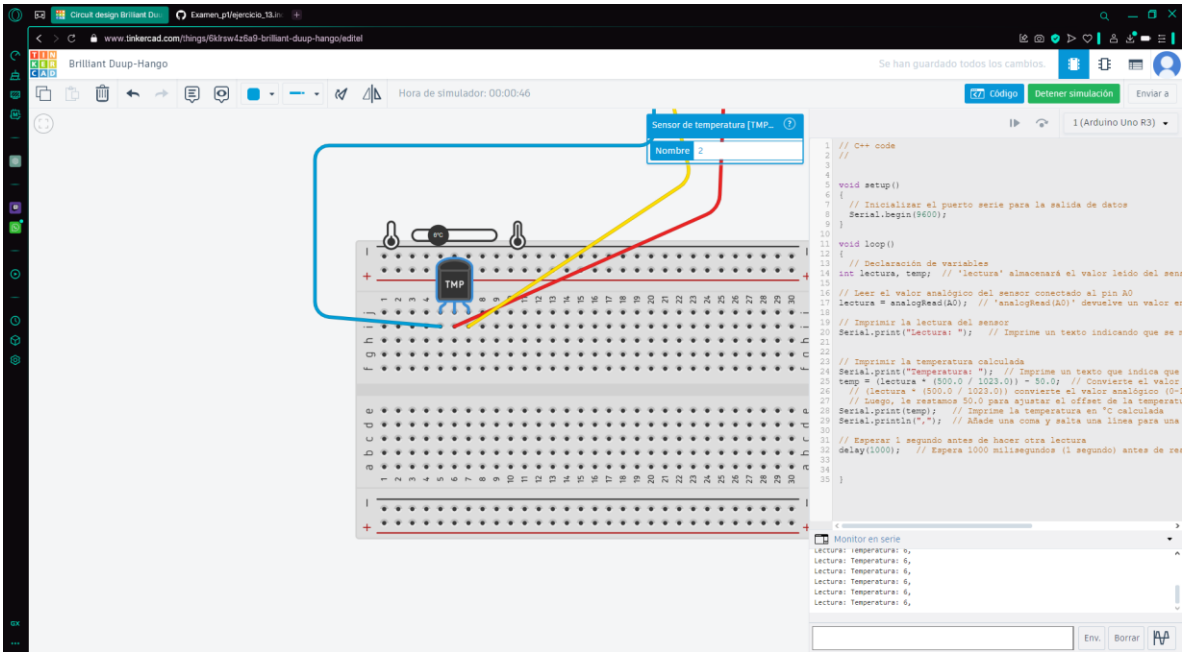
Luego, se realizaron las conexiones necesarias. El pin izquierdo del sensor se conectó a 5V en el Arduino para su alimentación. El pin central del sensor se conectó al puerto analógico A0 del Arduino, ya que este es el que enviará los datos de temperatura. El pin derecho se conectó a GND para completar el circuito.

Con las conexiones listas, se procedió a programar el Arduino. Se escribió un código para leer los valores del sensor, convertirlos a temperatura en grados Celsius y mostrarlos en el monitor serie. El código 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 cargó el código en la placa Arduino y se abrió el monitor serie del IDE para visualizar los valores de temperatura en tiempo real. Se pudo observar que el sensor analógico devuelve un valor numérico que, mediante una fórmula matemática, se convierte en temperatura en grados Celsius. Además, se comprendió la importancia de la comunicación serie para visualizar los datos y la necesidad de realizar conversiones adecuadas para interpretar correctamente las lecturas del sensor.

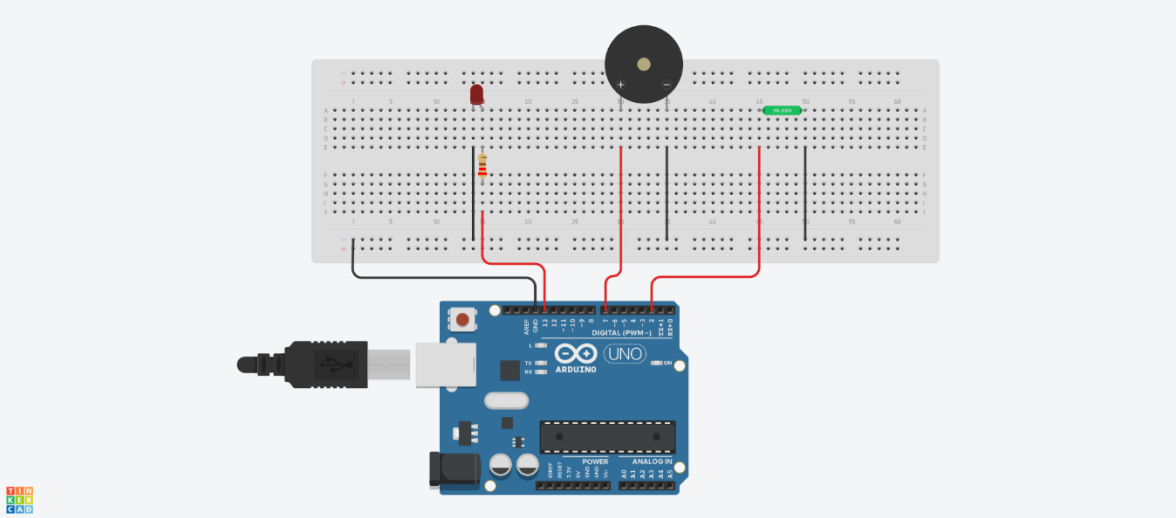


Ejercicio 14

Primeramente, Se puso una placa de pruebas (protoboard) y también sacamos el Arduino UNO R3 y se conecto desde el puerto GND a un puerto negativo de la protoboard, se colocó una led en las coordenadas A14 Y A15, también una resistencia de 220Ω (ohm) en las coordenadas E15 Y G15 que estaba enlazada al lado positivo de la led, conectando ambas partes desde la coordenada J15 al puerto 13 del Arduino y el lado negativo de la LED, del lado negativo de la protoboard.

Se coloco un buzzer en las coordenadas A30 y A35 y un sensor de inclinación a las coordenadas alineados con la led, de modo que pueda conectarse de forma mas organizada cada componente al Arduino, el lado positivo del buzzer se conecta al puerto 7 del Arduino, aso como el lado negativo e conecta al

En el sensor de inclinacion se colocó en las coordenadas A45 y A50 de modo que al estar alineada con la LED y el Buzzer, se organiza mejor la conexión, pues el siguiente paso fue conectar el lado positivo de el sensor al puerto 2 del Arduino y viceversa el lado negativo, se conecta al puerto de tierra (negativo) 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);                 }
}
```

De este modo, ya con la implementación del código y la realización del modelo con los componentes implementados, se ejecuta la simulación de modo que la led estara encendida, pero al pasar al sensor de inclinación, conforme este se desliza, la led poco a poco perderá intensidad en su luz hasta llegar el punto en que esta se apagara por completo y el buzzer mediante vibraciones comenzara a hacer sonido de “chicharra” y este no se detendra hasta que el sensor de inclinación sea acomodado y la luz de la led vuelva a estar encendida.

Ejercicio 15

En esta práctica, se diseñó un sistema utilizando un sensor de ultrasonido HC-SR04 y un Arduino UNO para medir distancias. Cuando un objeto se encuentra a menos de 10 cm, un LED se enciende como indicador visual. Para lograrlo, se programó el Arduino para realizar la medición de la distancia y activar el LED según la condición establecida.

Para comenzar, se conectó el sensor de ultrasonido a la protoboard y se realizaron las conexiones con el Arduino. El HC-SR04 cuenta con cuatro pines:

VCC: Se conectó al pin de 5V del Arduino para su alimentación.

GND: Se conectó al pin GND del Arduino.

Trigger: Se conectó al pin digital 10 del Arduino.

Echo: Se conectó al pin digital 11 del Arduino.

Luego, se colocó un LED en la protoboard con una resistencia de 220 ohmios en serie para evitar daños. El ánodo del LED se conectó al pin digital 13 del Arduino y el cátodo a GND.

Una vez realizadas las conexiones, se cargó el siguiente código en el Arduino:

```
const int Trigger = 10;    // Pin digital 10 para el
Trigger del sensor

const int Echo = 11;       // Pin digital 11 para el Echo
del sensor

const int Led = 13;        // Pin digital 13 para el LED


void setup() {
```

```
  Serial.begin(9600);       // Iniciamos la comunicación serie
  pinMode(Trigger, OUTPUT);// Pin Trigger como salida
  pinMode(Echo, INPUT);     // Pin Echo como entrada
  pinMode(Led, OUTPUT);     // Pin LED como salida
  digitalWrite(Trigger, LOW); // Inicializamos el Trigger en LOW
}

void loop() {
  long t; // Tiempo que demora en llegar el eco
  long d; // Distancia en centímetros

  digitalWrite(Trigger, HIGH);
  delayMicroseconds(10);   // Enviamos un pulso de 10us
  digitalWrite(Trigger, LOW);

  t = pulseIn(Echo, HIGH); // Obtenemos el ancho del pulso
  d = t / 59;              // Convertimos el tiempo a distancia en cm

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

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

  delay(100);  // Pequeña pausa de 100ms
}
```

Finalmente, se abrió el Monitor Serie del IDE de Arduino para observar las mediciones en tiempo real. Se colocaron objetos frente al sensor para probar su funcionamiento. Al acercar un objeto a menos de 10 cm, el LED se encendió correctamente.