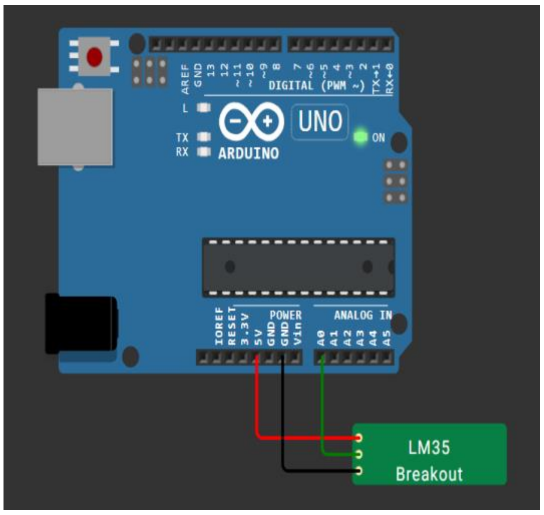
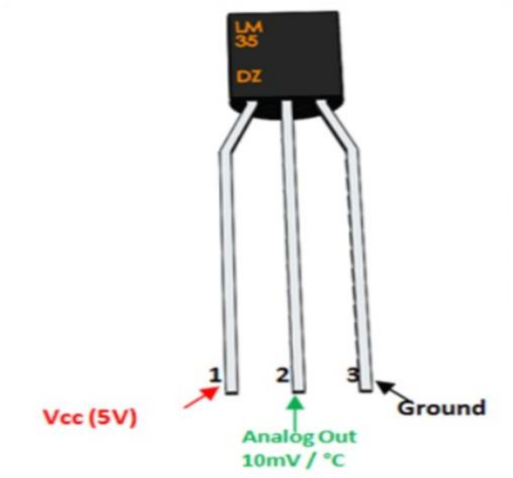
Experiment No. 4

Aim: To quantify ambient thermal parameters through analog (LM35) and digital (DHT11) temperature sensors.

Simulator: Wokwi Simulator.

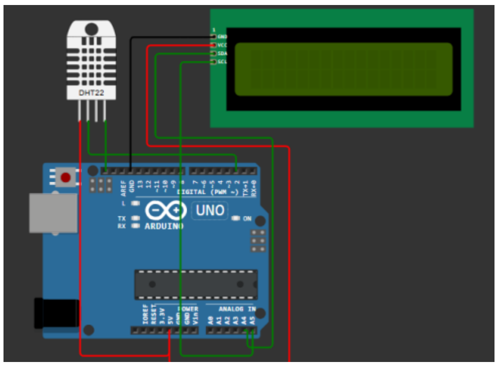
Circuit Layout:





(a) (b)

Figure 1: (a) Interfacing LM35 with Arduino Uno (b) Pin description of LM35



(a) (b)

Figure 2: (a) Interfacing DHT22 with Arduino Uno (b) Pin description of DHT22

Theory:

LM35 – Analog Temperature Sensor

The LM35 is a precision analog temperature sensor that outputs a voltage linearly proportional to the ambient temperature in Celsius. It provides 10 millivolts per degree Celsius, making it straightforward to interface with microcontrollers that have analog-to-digital converters (ADCs). With a typical accuracy of ±0.5°C and minimal self-heating, the LM35 is ideal for precise thermal measurements. Its linear response and calibration-free operation make it suitable for embedded systems and industrial applications requiring analog precision.

DHT22 – Digital Temperature and Humidity Sensor

The DHT22, also known as AM2302, is a digital sensor designed to measure both temperature and relative humidity with high accuracy and stability. It operates using a capacitive humidity sensing element and an NTC thermistor for temperature measurement. These components detect changes in environmental conditions and convert them into electrical signals, which are then processed by an onboard microcontroller. The sensor outputs data in a digital format using a single-wire communication protocol, making it easy to interface with microcontrollers such as Arduino or ESP32.

DHT11 VS DHT22

While the LM35 excels in high-resolution analog temperature measurement, the DHT11 offers the added benefit of humidity sensing and digital communication. Using both sensors together allows for cross-validation of temperature readings and provides a more comprehensive understanding of ambient thermal conditions. This dual-sensor setup enhances reliability and data richness, especially in applications like IoT-based monitoring, smart agriculture, and HVAC diagnostics.

<u>Code Used:</u>

Lm35.chip.json

```
{
  "name": "LM35",
  "author": "Maverick",
  "pins": [
    "VCC",
    "OUT",
    "GND",
    "",
    ""
  ],
  "controls": [
    {
      "id": "temperature",
      "label": "Temperature",
      "type": "range",
      "min": 2,
      "max": 150,
    "step": 1
   }
 ]
}
```

Lm35.chip.c

```
#include "wokwi-api.h"
#include <stdlib.h>
// Chip data.
typedef struct
{
  pin_t VCC;
  pin_t OUT;
  pin_t GND;
  uint32_t temperature;
} chip_data_t;

// Returns true if the power source is connected correctly.
bool power_connected(void *data)
{
  chip_data_t *chip = (chip_data_t*)data;
  return pin_read(chip->VCC) && !pin_read(chip->GND);
}

// Timer function. Analog output based on temperature.
void chip_timer_callback(void *user_data)
{
  if (power_connected(user_data))
  {
    chip_data_t *chip_data = (chip_data_t*)user_data;
    uint32_t temperature = attr_read(chip_data->temperature);
    float volts = 0.01 * temperature;
    pin_dac_write(chip_data->OUT, volts);
  }
}

// Chip initialization.
void chip_init(void)
{
  chip_data_t *chip_data = malloc(sizeof(chip_data_t));
  chip_data->VCC = pin_init("VCC", INPUT);
  chip_data->GND = pin_init("GND", INPUT);
  chip_data->OUT = pin_init("OUT", ANALOG);
  chip_data->temperature = attr_init("temperature", 50);
  const timer_config_t config =
  {
    .callback = chip_timer_callback,
    .user_data = chip_data,
  };

  timer_t timer_id = timer_init(&config);
  timer_start(timer_id, 10000, true);
}
```

LM35

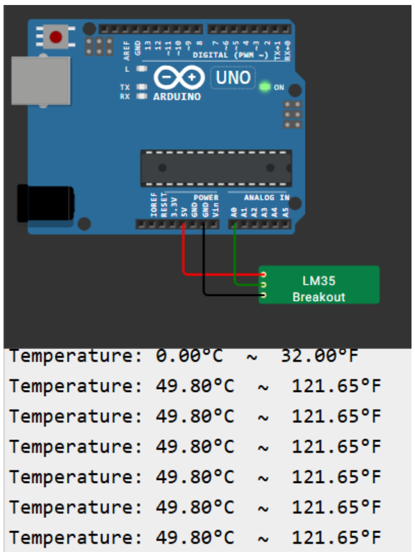
```
#define ADC_VREF_mV    5000.0 // in millivolt
#define ADC_RESOLUTION 1024.0
#define PIN_LM35       A0

void setup() {
  Serial.begin(9600);
}

void loop() {
  // get the ADC value from the temperature sensor
  int adcVal = analogRead(PIN_LM35);
  // convert the ADC value to voltage in millivolt
  float milliVolt = adcVal * (ADC_VREF_mV / ADC_RESOLUTION);
  // convert the voltage to the temperature in Celsius
  float tempC = milliVolt / 10;
  // convert the Celsius to Fahrenheit
  float tempF = tempC * 9 / 5 + 32;

  // print the temperature in the Serial Monitor:
  Serial.print("Temperature: ");
  Serial.print(tempC);   // print the temperature in Celsius
  Serial.print("°C");
  Serial.print("  ~  "); // separator between Celsius and Fahrenheit
  Serial.print(tempF);   // print the temperature in Fahrenheit
  Serial.println("°F");

  delay(1000);
}
```

DHT22

```
#include <DHT.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

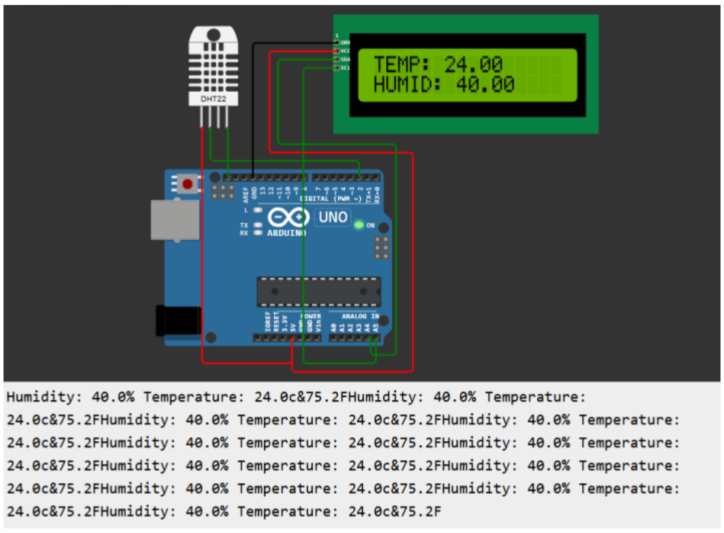
```
#define DHTPIN 2
#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);
LiquidCrystal_I2C lcd(0x27,16,2); // 0x3F or 0x27
void setup() {
Serial.begin(9600);
dht.begin();
lcd.init(); // LCD initialize
lcd.backlight(); // LCD backlight int
}

void loop() {
delay(1000); // Wait a few seconds between measurements.
float h = dht.readHumidity();
float t = dht.readTemperature();
float f= dht.readTemperature(true);
lcd.setCursor(0,0); lcd.print("TEMP: "); lcd.print(t);
lcd.setCursor(0,1); lcd.print("HUMID: "); lcd.print(h);
Serial.print("Humidity: ");
Serial.print(h,1);
Serial.print("% Temperature: ");
Serial.print(t,1);
Serial.print("c&");
Serial.print(f,1);
Serial.print("F" );

}
```

Simulation Outcome:



```
Temperature: 0.00°C  ~  32.00°F
Temperature: 49.80°C  ~  121.65°F
Temperature: 49.80°C  ~  121.65°F
Temperature: 49.80°C  ~  121.65°F
Temperature: 49.80°C  ~  121.65°F
Temperature: 49.80°C  ~  121.65°F
Temperature: 49.80°C  ~  121.65°F
```

Figure: 3 LM35 Temperature sensor simulation output



```
Humidity: 40.0% Temperature: 24.0c&75.2FHumidity: 40.0% Temperature:
24.0c&75.2FHumidity: 40.0% Temperature: 24.0c&75.2FHumidity: 40.0% Temperature:
24.0c&75.2FHumidity: 40.0% Temperature: 24.0c&75.2FHumidity: 40.0% Temperature:
24.0c&75.2FHumidity: 40.0% Temperature: 24.0c&75.2FHumidity: 40.0% Temperature:
24.0c&75.2FHumidity: 40.0% Temperature: 24.0c&75.2FHumidity: 40.0% Temperature:
24.0c&75.2FHumidity: 40.0% Temperature: 24.0c&75.2F
```

Figure 4: DHT22 sensor simulation output

<u>Result:</u>

In this experiment, we used two sensors with Arduino to measure temperature – LM35 (analog) and DHT11 (digital). The LM35 gave smooth, fast, and more precise temperature readings. It showed small changes clearly. The DHT11 gave digital readings of both temperature and humidity, but it was a bit slower and less detailed (rounded values). Still, it was easy to use and gave more info. Both sensors worked well. LM35 is better if you only need accurate temperature quickly. DHT11 is great if you also want humidity data in a simple way. This experiment helped us understand the difference between analog and digital sensors.