Group No:15	Course Code: CSE 368
	Course Title: Microprocessor and Interfacing Lab.

Title of Project: Humidity Controller

Shahjalal University of Science & Technology, Sylhet

Department of Computer Science & Engineering

Authority	Submitted to
Student ID Name 2020331030-Nobel Ahmad Badhon 2020331056-Tazbir Hossain Akash 2020331062-Gazi Maksudur Rahman 2020331092-Abdullah Al Mahadi Apurbo 2020331108 -Elias Ahahammed	Abdullah Al Noman Lecturer, Department of CSE, SUST.

Authority Names:

1. Nobel Ahmed Badhon
2. Gazi Maksudur Rahman
3. Elias Ahahammed
4. Tazbir Ahmed Akash
5. Abdullah Mahadi Apurbo

Abstract:

The Humidity Controller is a microprocessor-based project designed to regulate room humidity efficiently. This project utilizes a microcontroller to control a vaporization generator, aiming to increase the room's humidity levels. The key component responsible for monitoring the humidity levels is a hygrometer. By employing this system, the project ensures precise and effective control over the room's humidity automatically, providing a reliable solution for environments requiring specific humidity conditions.

Introduction:

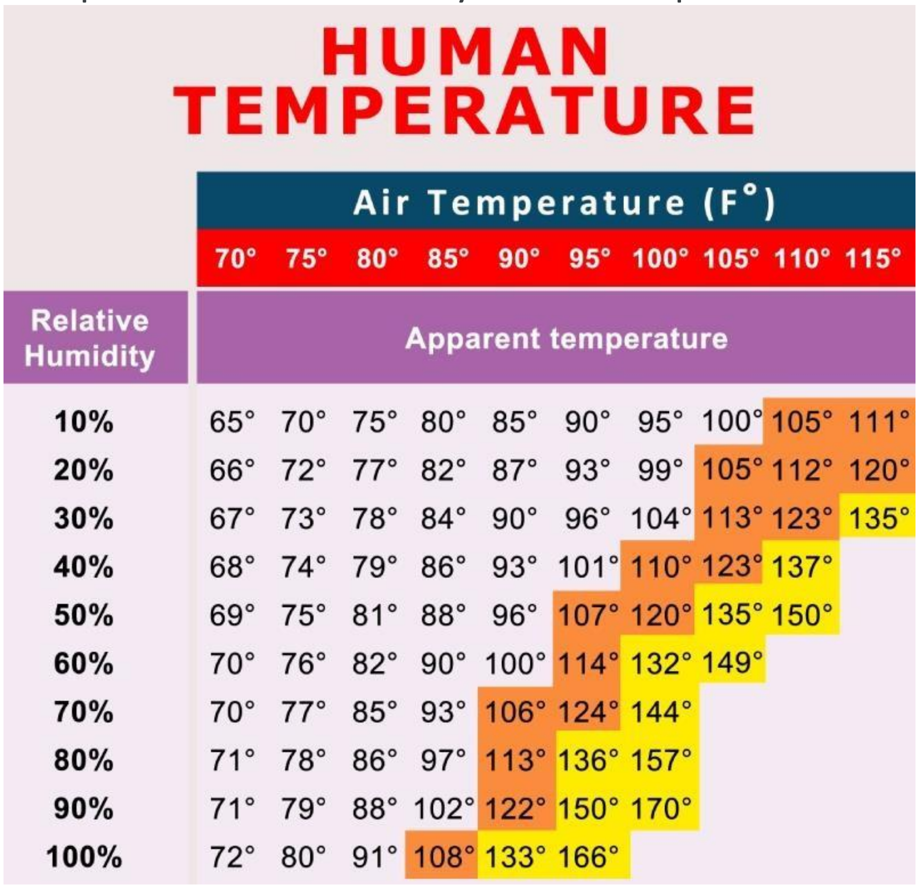
Bangladesh is facing severe mid-winter cold, impacting daily life with dense nighttime fog. This has led to an increase in cold-related illnesses, particularly affecting children and the elderly. Additionally, the winter season brings very low humidity levels, emphasizing the importance of maintaining optimal humidity in controlled environments, greenhouses, and industrial processes. The Humidity Controller project addresses the need for a sophisticated system to manage room humidity. This project focuses on achieving this control by utilizing a microcontroller to regulate a vaporization generator. The microcontroller processes information from a hygrometer, a sensor designed to measure humidity accurately. The microcontroller processes information from a hygrometer, a sensor designed to measure humidity accurately. By interfacing the microcontroller with the vaporization generator, the system can adjust the humidity levels in the room as required.

Related works:

- Making water vapor by heater.
- Measure of temperature across humidity
- Measure of Humidity across water vapor
- Fixed a humidity level when it's lower, the humidity controller machine starts boiling water by heater.

Proposed Solution:

Temperature and Humidity Relationships chart:



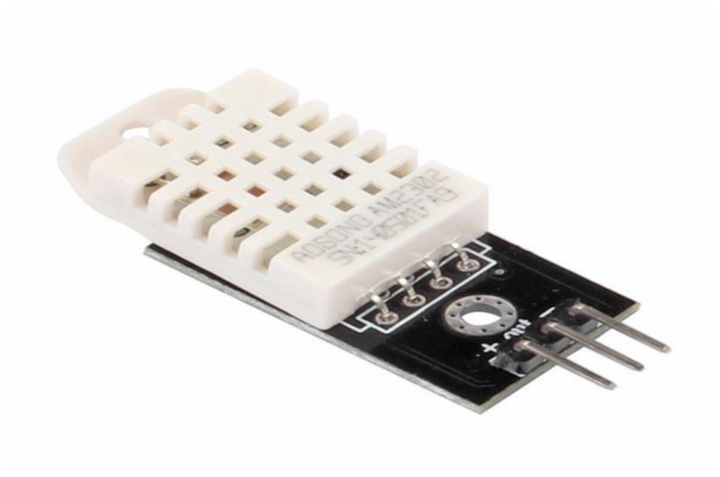
HUMAN TEMPERATURE

	Air Temperature (F°)									
	70°	75°	80°	85°	90°	95°	100°	105°	110°	115°
Relative Humidity	Apparent temperature									
10%	65°	70°	75°	80°	85°	90°	95°	100°	105°	111°
20%	66°	72°	77°	82°	87°	93°	99°	105°	112°	120°
30%	67°	73°	78°	84°	90°	96°	104°	113°	123°	135°
40%	68°	74°	79°	86°	93°	101°	110°	123°	137°	
50%	69°	75°	81°	88°	96°	107°	120°	135°	150°	
60%	70°	76°	82°	90°	100°	114°	132°	149°		
70%	70°	77°	85°	93°	106°	124°	144°			
80%	71°	78°	86°	97°	113°	136°	157°			
90%	71°	79°	88°	102°	122°	150°	170°			
100%	72°	80°	91°	108°	133°	166°				

When the humidity level is lower than fixed level, the humidity controller machine starts boiling water so that it can create water vapors. It will continue up to across the fixed level. If we control humidity of a room environment, we can overcome cold, roughness for low level temperature.

Necessary Hardware:

- DHT22 humidity and temperature sensor

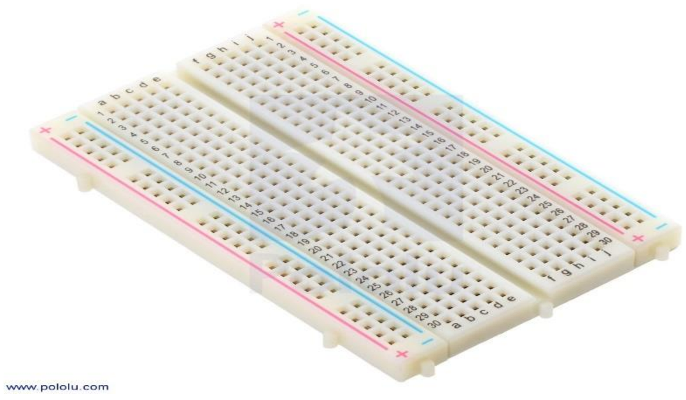


- Arduino Uno microcontroller

- 5V relay module

- Humidifier

- Breadboard



- jumper wires

- 10kΩ resistor

- Power supply (5V)

- LED light

- 10K Potentiometer

- **LiquidCrystal_I2C Display**

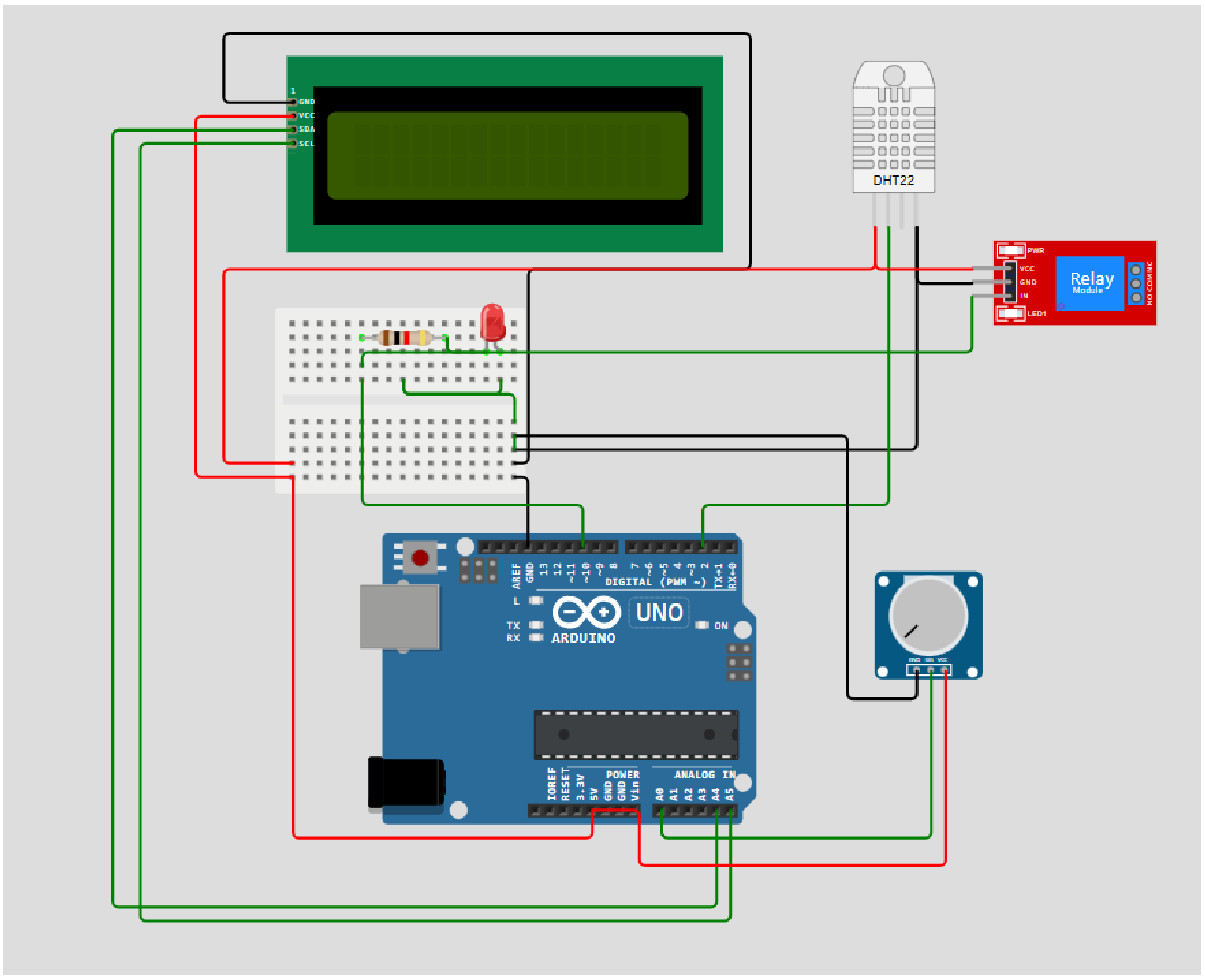
Necessary Software:

- **Arduino IDE**

Preferable Program Code:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include "DHT.h"
#define red_led 10

// Set the LCD number of columns and rows
int lcdColumns = 16;
int lcdRows = 2;
const int potpin = A0;
int potValue = 0;

// Set LCD address, number of columns and rows
// If you don't know your display address, run an I2C scanner sketch
LiquidCrystal_I2C lcd(0x27, lcdColumns, lcdRows);

#define DHTPIN 2     // Digital pin connected to the DHT sensor
#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);
  lcd.init();       // Initialize LCD
  lcd.backlight();  // Turn on LCD backlight
  dht.begin();      // Initialize DHT sensor
  pinMode(red_led, OUTPUT);
  digitalWrite(red_led, HIGH);
}

void loop() {
  // Wait a few seconds between measurements
  delay(2000);

  // Reading temperature or humidity takes about 250 milliseconds
  // Sensor readings may also be up to 2 seconds 'old' (it's a slow sensor)
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  h = roundf(h * 10.0) / 10.0;
  t = roundf(t * 10.0) / 10.0;

  char humidityStr[6]; // Array to hold the string for humidity (3 digits + decimal
point + null terminator)
  char temperatureStr[6]; // Array to hold the string for temperature (3 digits +
decimal point + null terminator)
  dtostrf(h, 4, 1, humidityStr); // Convert float to string with one decimal point
  dtostrf(t, 4, 1, temperatureStr);
  bool status = false;
  potValue = analogRead(potpin);

  potValue = map(potValue, 0, 1023, 0, 100);

  // Check if any reads failed and exit early (to try again)
  if (isnan(h) || isnan(t)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  if(h<potValue){
    digitalWrite(red_led, LOW);
    status =true;
  }else{
    digitalWrite(red_led,HIGH);
    status =false;
  }

  // Display temperature and humidity on LCD
  lcd.setCursor(0, 0);
  lcd.print(F("T:"));
  lcd.print(temperatureStr);
  lcd.print(F("C H:"));
  lcd.print(humidityStr);
  lcd.print(F("%"));

  // Move to the next line and print additional message
  lcd.setCursor(0, 1);
  lcd.print("Tg: ");
  lcd.print(potValue);
  lcd.print(" St: ");
  lcd.print(status);

}
```

Circuit Design:



Connections:

- **LiquidCrystal_I2C Display:**
  - **VCC** to **5V** on Arduino
  - **GND** to **GND** on Arduino
  - **SDA** to **A4** on Arduino
  - **SCL** to **A5** on Arduino
- **DHT22 Sensor:**
  - **VCC** to **5V** on Arduino
  - **GND** to **GND** on Arduino
  - **DATA** to **Digital Pin 2** on Arduino
- **Potentiometer:**
  - **VCC** to **5V** on Arduino
  - **GND** to **GND** on Arduino
  - **Wiper (middle pin)** to **A0** on Arduino

- **Red LED:**
    - **Anode (long leg)** to **Digital Pin 10** on Arduino (through a 10kΩ resistor)
    - **Cathode (short leg)** to **GND** on Arduino
- **Relay:**
    - **VCC** to **5V** on Arduino;
    - **GND** to **GNT** on Arduino;
    - **IN** to Digital **Pin 10** on Arduino;
- **Humidifier Connect with Relay:**
    - **5 V** of Humidifier connect with Relay **ON**;
    - **GND** to **GND** on Arduino;
    - **COM** of Relay connect with Humidifier **VCC;**

Code Explanation:

The source code for this project consists of setup and loop functions, along with the necessary library imports and initializations.

1. Libraries and Definitions:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include "DHT.h"
#define red_led 10
```

- Wire.h and LiquidCrystal_I2C.h are included for I2C communication with the LCD.
- DHT.h is included to interface with the DHT22 sensor.
- The red LED is defined to be connected to digital pin 10.

2. LCD and Sensor Initialization:

```
int lcdColumns = 16;
int lcdRows = 2;
const int potpin = A0;
int potValue = 0;
LiquidCrystal_I2C lcd(0x27, lcdColumns, lcdRows);
#define DHTPIN 2
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
```

- LCD is set to have 16 columns and 2 rows.
- Potentiometer is connected to analog pin A0.
- LCD address is set to 0x27.
- DHT22 sensor is connected to digital pin 2.

3. Setup Function:

```
void setup() {
  Serial.begin(115200);
  lcd.init();
  lcd.backlight();
  dht.begin();
  pinMode(red_led, OUTPUT);
  digitalWrite(red_led, HIGH);
}
```

- Serial communication is initialized at 115200 baud rate.
- LCD and DHT22 sensor are initialized.
- Red LED is set as output and initially turned off (HIGH).

4. Loop Function:

```
void loop() {
  delay(2000);
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  h = roundf(h * 10.0) / 10.0;
  t = roundf(t * 10.0) / 10.0;
  char humidityStr[6];
  char temperatureStr[6];
  dtostrf(h, 4, 1, humidityStr);
  dtostrf(t, 4, 1, temperatureStr);
  bool status = false;
  potValue = analogRead(potpin);
  potValue = map(potValue, 0, 1023, 0, 100);

  if (isnan(h) || isnan(t)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  if(h < potValue){
    digitalWrite(red_led, LOW);
    status = true;
  } else {
    digitalWrite(red_led, HIGH);
    status = false;
  }

  lcd.setCursor(0, 0);
  lcd.print(F("T:"));
  lcd.print(temperatureStr);
  lcd.print(F("C H:"));
  lcd.print(humidityStr);
  lcd.print(F("%"));

  lcd.setCursor(0, 1);
  lcd.print("Tg: ");
  lcd.print(potValue);
  lcd.print(" St: ");
  lcd.print(status);
}
```

- The system waits for 2 seconds between measurements.
- Temperature and humidity are read from the DHT22 sensor and rounded to one decimal place.
- Values are converted to strings for display.
- The potentiometer value is read and mapped to a range of 0-100.
- If the humidity is less than the potentiometer value, the LED is turned on (LOW), otherwise, it remains off (HIGH).

- Temperature, humidity, threshold value, and status are displayed on the LCD.

Results:

The constructed humidity controller effectively maintained the desired humidity level within the test environment. The DHT22 sensor provided accurate humidity readings, and the Arduino-controlled relay successfully activated and deactivated the humidifier based on the measured humidity.

Data

Firstly, we set humidity with 10k potentiometer at 90%. Then it has been happening:

Time (min)	Humidity (%)	Humidifier Status
0	85	ON
5	88	ON
10	90	OFF
15	95	OFF
20	89	ON
25	93	OFF

Discussion

The humidity controller demonstrated reliable performance in maintaining the desired humidity level. The system's response time was adequate for the test environment, and the hysteresis implemented in the software prevented rapid cycling of the humidifier. The DHT22 sensor's accuracy was sufficient for this application, though higher precision sensors could be considered for more demanding requirements.

Conclusion

This project successfully developed a humidity controller using an Arduino, DHT22 sensor, and relay module. The system effectively maintained the desired humidity level, proving the feasibility of using low-cost components for humidity control applications. Future improvements could include integrating a display for real-time humidity monitoring and expanding the system to control multiple humidifiers for larger environments.