

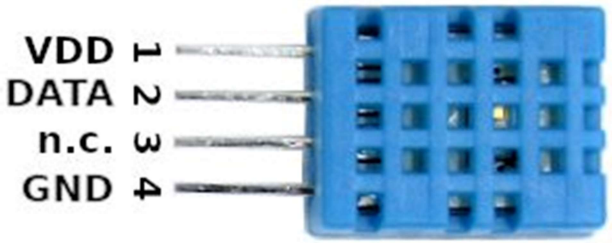
Experiment No. 6

Aim: To measure temperature and humidity using DHT sensor

Theory

DHT11 -Low-cost digital sensor for sensing temperature and humidity. It is easily interfaced with any microcontroller such as Arduino, Raspberry Pi, etc... to measure humidity and temperature instantaneously. It is available as a sensor and as a module. Difference between sensor and module is of Pull-up resistor and a power-on LED. It has resistive humidity sensing and negative temperature coefficient (NTC). 8 bit MCU is connected in it which is responsible for its fast response. DHT11 sensor consists of a capacitive humidity sensing element and a thermistor for sensing temperature. Humidity sensing capacitor has two electrodes with a moisture-holding substrate as a dielectric between them. Change in capacitance value occurs with change in humidity levels. IC measure, process this changed resistance values and change them into digital form. For measuring temperature this sensor uses Negative Temperature coefficient thermistor, which causes a decrease in its resistance value with an increase in temperature. To get a larger resistance value even for smallest change in temperature, this sensor is usually made up of semiconductor ceramics or polymers.

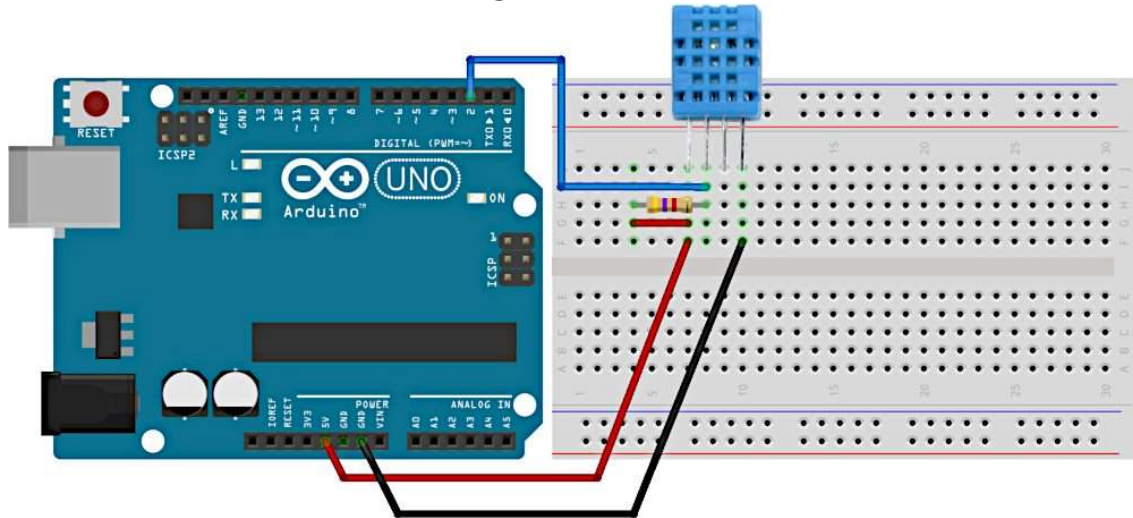
DHT Sensor Pin out

PIN 1,2,3,4 (from left to right) <ul style="list-style-type: none"> • PIN 1- 3.3V-5V Power supply • PIN 2- Data • PIN 3- Null • PIN 4- Ground 	
---	--

DHT Sensor Library

- Arduino supports a special library for DHT11 and DHT22 sensors
- Provides function to read temperature and humidity values from data pin
- `dht.readHumidity()`
- `dht.readTemperature()`
- Install DHT Sensor Library
- Go to sketch -> Include Library -> Manage Library

Arduino and DHT Sensor Circuit Diagram



DHT 22Code

```
#include <DHT.h>
DHT dht(2, DHT22);
void setup() {
  Serial.begin(9600);
  dht.begin();
}

void loop() {
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.print("%  Temperature: ");
  Serial.print(temperature);
  Serial.println("°C ");
  delay(2000);
}
```

DHT 11Code

```
#include "DHT.h"
#define DHTPIN 2
#define DHTTYPE DHT11
// DHT dht(2,DHT11)
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  Serial.println("DHT11 test!");
  dht.begin();
}
void loop() {
  delay(2000);
  float humidity = dht.readHumidity();
```

VIPS-TC

```
float temperature = dht.readTemperature();
if (isnan(humidity) || isnan(temperature))
//isnan = is NOT A NUMBER which return true when it is not a
number
{
    Serial.println("Failed to read from DHT sensor!");
    return;
}
```

Result

PASTE YOUR SCREENSHOT OF THE SUCCESSFUL RUN OF PROGRAM INCLUDING HARDWARE CONNECTION AND ARDUINO IDE

Experiment No. 7

Aim: Custom functions that can be created for specific Needs. Using a RGB LED create a custom function program.

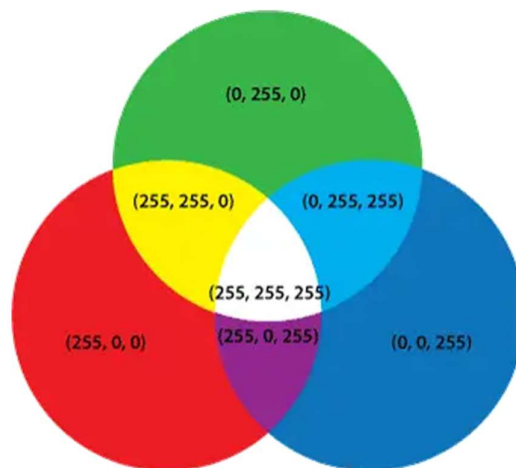
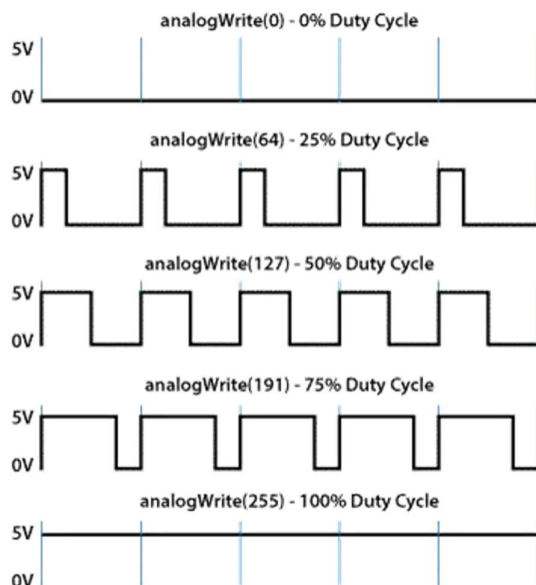
Theory

RGB - Stands for "Red Green Blue." RGB refers to three hues of light that can be mixed together to create different colors. Combining red, green, and blue light is standard method of producing color images on screens, such as TVs, computer monitors, and smartphone screens. Types of RGB led's: Common cathode and Common anode. In common cathode RGB led, cathode of all led's is common and PWM signals is given to anode of led's. In common anode RGB led, anode of all led's is common and PWM signals is given to cathode of led's.

How does an RGB LED work?

- RGB LED can emit different colors by mixing 3 basic colors red, green and blue.
- Consists of 3 separate LEDs red, green and blue packed in a single case.
- Comprises of 4 leads, one lead for each of 3 colors and one common cathode or anode depending on RGB LED type.
- Cathode will be connected to ground and 3 anodes will be connected through 220 Ohms resistors to 3 digital pins on Arduino Board that can provide PWM signal.
- Using PWM for simulating analog output will provide different voltage levels to LEDs to get desired colors

PWM - Pulse Width Modulation



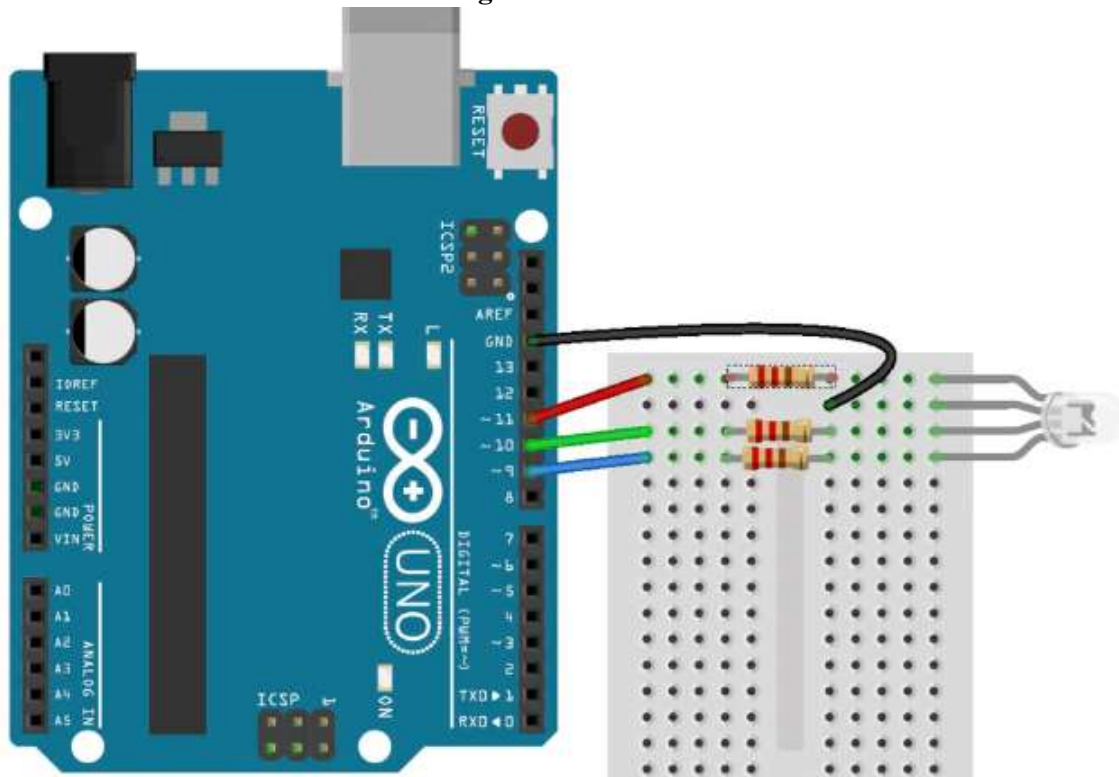
RGB Sensor Pin out

PIN 1,2,3,4 (from left to right)

- Pin D5 - R
- Pin D4 - G
- Pin D3 - B



Arduino and RGB Sensor Circuit Diagram



RGB LED Code

```
#define LED1RED 5
#define LED1BLUE 3
#define LED1GREEN 4
void setup() {
  pinMode(LED1RED, OUTPUT);
  pinMode(LED1BLUE, OUTPUT);
  pinMode(LED1GREEN, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  digitalWrite(LED1RED, HIGH);
  Serial.println("LED1RED ");
  delay(1000);
  digitalWrite(LED1RED, LOW);
```

IOT LAB

```

Serial.println("LED1RED");
delay(1000);
digitalWrite(LED1BLUE, HIGH);
Serial.println("LED1BLUE");
delay(1000);
digitalWrite(LED1BLUE, LOW);
Serial.println("LED1BLUE");
delay(1000);
digitalWrite(LED1GREEN, HIGH);
Serial.println("LED1GREEN");
delay(1000);
digitalWrite(LED1GREEN, LOW);
Serial.println("LED1GREEN");
delay(1000);
}

```

Using RGB sensor to create functions for special needs such as setcolor in the following case:

```

int redPin= 7;
int greenPin = 6;
int bluePin = 5;
void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}
void loop() {
  setColor(255, 0, 0); // Red Color
  delay(1000);
  setColor(0, 255, 0); // Green Color
  delay(1000);
  setColor(0, 0, 255); // Blue Color
  delay(1000);
  setColor(255, 255, 255); // White Color
  delay(1000);
  setColor(170, 0, 255); // Purple Color
  delay(1000);
}
void setColor(int redValue, int greenValue, int blueValue) {
  analogWrite(redPin, redValue);
  analogWrite(greenPin, greenValue);
  analogWrite(bluePin, blueValue);
}

```

Result

PASTE YOUR SCREENSHOT OF THE SUCCESSFUL RUN OF PROGRAM INCLUDING HARDWARE CONNECTION AND ARDUINO IDE