A Practical Activity Report Submitted
for IOT Based Systems (UEC715)
by

Name of student: Pratham Singla
Roll number: 102115190
Group: 4NC6

Submitted to
Dr. Karmjit Singh Sandha

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY, (A DEEMED TO BE UNIVERSITY), PATIALA, PUNJAB
INDIA
Jul-Dec 2024

Table Of Content

Experiment: 1

Objective: Familiarise with Arduino UNO and interface LED with Arduino and write a program to turn ON LED for 1 sec after every 2 seconds.

Software Used: Arduino IDE

Components Used: Arduino UNO R3, LEDs, breadboard, jumper wires

Theory:

Introduction to Arduino UNO: Arduino UNO is a microcontroller board based on the ATmega328P. It is one of the most popular and widely used boards in the Arduino family, particularly well-suited for beginners and educational purposes due to its simplicity and ease of use. The Arduino UNO features 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header, and a reset button. It can be powered via a USB cable or an external power supply.

LED (Light Emitting Diode): LEDs are semiconductor devices that emit light when an electric current passes through them. They have two terminals: the anode (positive) and the cathode (negative). When interfacing an LED with the Arduino, a current-limiting resistor is typically used to prevent excessive current from damaging the LED.

Breadboard and Jumper Wires: A breadboard is a construction base for prototyping electronics. It allows you to create circuits without soldering by inserting components and wires into interconnected holes.

Circuit Diagram:

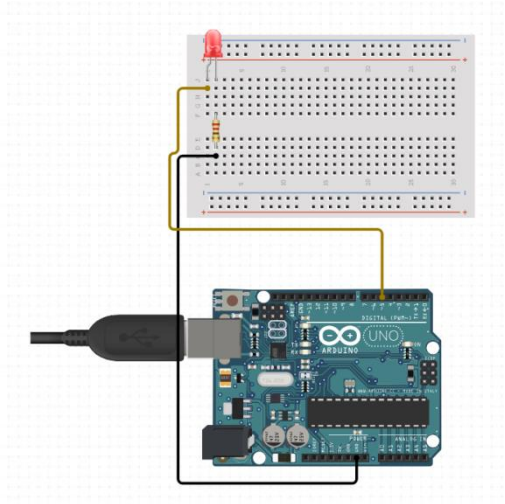

Fig1. LED connected to Arduino UNO using a breadboard

Code:

```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(5, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(5, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for 1 second
  digitalWrite(5, LOW);    // turn the LED off by making the voltage LOW
  delay(2000);                       // wait for 2 seconds
}
```

Output:

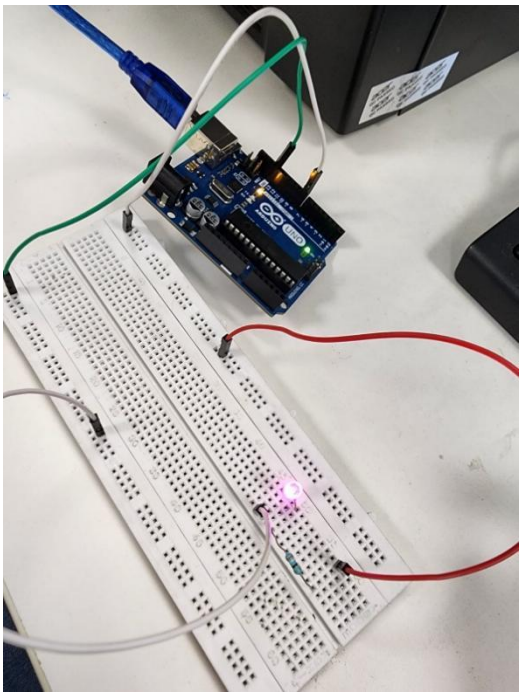

Fig2. LED turned on for 2 seconds

Conclusions: In this experiment, we successfully familiarized ourselves with the Arduino UNO by interfacing an LED and programming it to turn on for 1 second and off for 2 seconds in a continuous loop. This simple exercise demonstrated the basics of digital output control using the Arduino platform, laying a foundation for more complex projects involving various sensors and actuators. Through this experiment, we gained hands-on experience with the Arduino IDE, circuit design using a breadboard, and the fundamental concepts of timing and control in embedded systems.

Experiment: 2

Objective: To interface the Push button with Arduino, write a program to turn ON the LED when a push button is pressed.

Software Used: Arduino IDE

Components Used: Arduino UNO R3, LEDs, breadboard, jumper wires, push button

Theory:

Push Button: A push button is a simple switch mechanism used to temporarily make or break a connection in an electronic circuit. When the button is pressed, it completes the circuit, allowing current to flow. When the button is released, the circuit is broken, stopping the current flow. Push buttons are commonly used in various electronic devices to provide user input, such as turning a device on or off, or as an interface for more complex interactions.

Circuit Diagram:

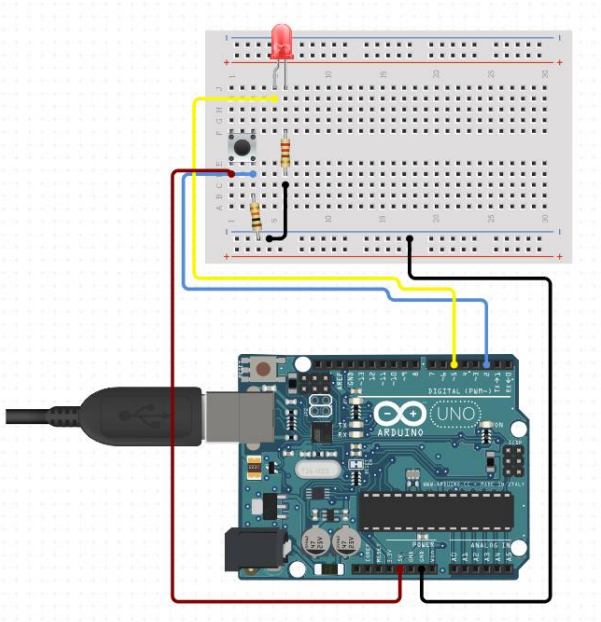

Fig1. LED and Push Button connected to Arduino UNO using a breadboard

Code:

```
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  5;      // the number of the LED pin
int buttonState = LOW;         // variable for reading the pushbutton status
void setup() {
  // initialize the LED pin as an output:
```

```
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}
void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

Output:

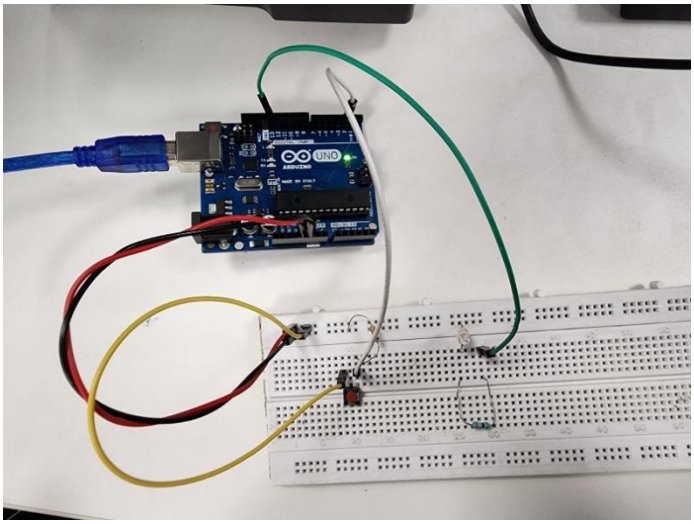

Fig2. LED will turn ON when button is pressed

Conclusions: In this experiment, we successfully interfaced a push button with the Arduino UNO to control an LED. We wrote a program that detects when the push button is pressed and responds by turning on the LED. This exercise illustrated the basic principle of using digital input to control digital output, demonstrating how external inputs can be used to interact with and control electronic components through the Arduino.

Experiment: 3

Objective: To interface the LDR sensor with Arduino, write a program to print LDR readings on a serial monitor.

Software Used: Arduino IDE

Components Used: Arduino UNO R3, LEDs, breadboard, jumper wires, LDR sensor

Theory:

An LDR (Light Dependent Resistor), also known as a photoresistor, is a type of resistor whose resistance varies with the intensity of light falling on it. As the light intensity increases, the resistance of the LDR decreases, and vice versa. This characteristic makes the LDR a useful component for light-sensing applications, such as automatic lighting systems, light meters, and various other applications where light intensity needs to be measured. To interface an LDR with the Arduino using a digital pin, the LDR is connected in a voltage divider circuit with a fixed resistor. The voltage across the LDR changes with the light intensity, and this varying voltage is fed into one of the digital input pins on the Arduino.

Circuit Diagram:

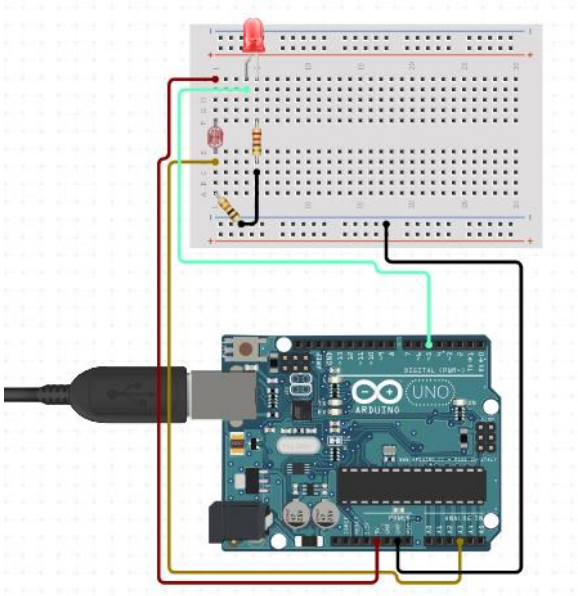

Fig1. LED and LDR Sensor connected to Arduino UNO using a breadboard

Code:

```
int ldr = 7;
int x;
```

```
int led = 13;
void setup() {
  Serial.begin(9600);
  pinMode(ldr, INPUT);
  pinMode(led, OUTPUT);
}
void loop() {
  x = digitalRead(ldr);
  Serial.println(x);
  if (x == HIGH) {
    digitalWrite(led, LOW);
  } else {
    digitalWrite(led, HIGH);
  }
}
```

Output:

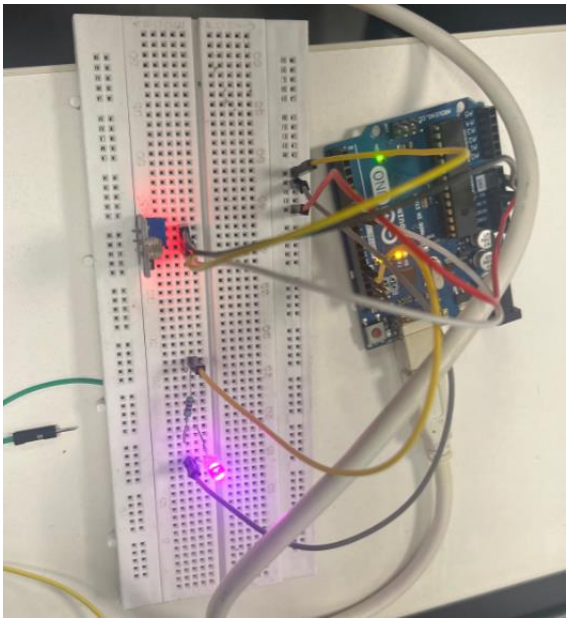

Fig2. LED will turn ON in absence of light

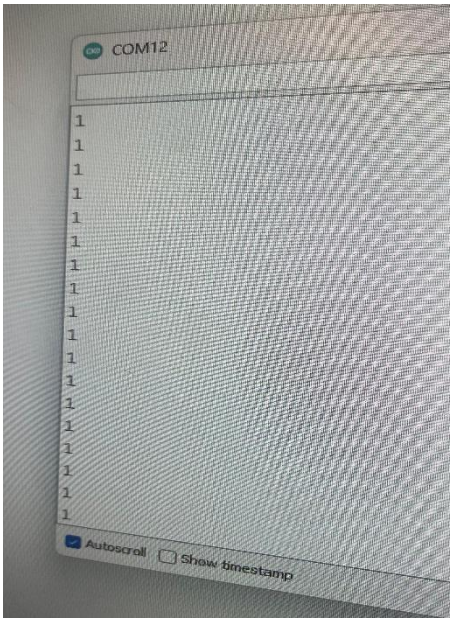

Fig3. Serial Monitor Output

Conclusions: The experiment successfully demonstrated a basic light-dependent control system using an LDR as a digital input. By treating the LDR as a simple on/off sensor, we were able to effectively control an LED based on ambient light conditions. The experiment provides a fundamental understanding of digital input/output operations and serves as a starting point for more sophisticated light-based projects that require a wider range of light level detection.

Experiment: 4

Objective: According to the LDR readings, write a program to turn ON & OFF multiple LEDs.

Software Used: Arduino IDE

Components Used: Arduino UNO R3, LEDs, breadboard, jumper wires, LDR sensor

Theory:

An LDR (Light Dependent Resistor), also known as a photoresistor, is a type of resistor whose resistance varies with the intensity of light falling on it. As the light intensity increases, the resistance of the LDR decreases, and vice versa. This characteristic makes the LDR a useful component for light-sensing applications, such as automatic lighting systems, light meters, and various other applications where light intensity needs to be measured. LDRs are made of semiconductor materials. When light photons hit the surface of the LDR, they excite the electrons in the semiconductor material, reducing its resistance. In the dark or low light conditions, the LDR exhibits high resistance, while in bright light, the resistance drops significantly. The change in resistance can be measured by an Arduino and used to trigger actions based on the light levels.

Circuit Diagram:

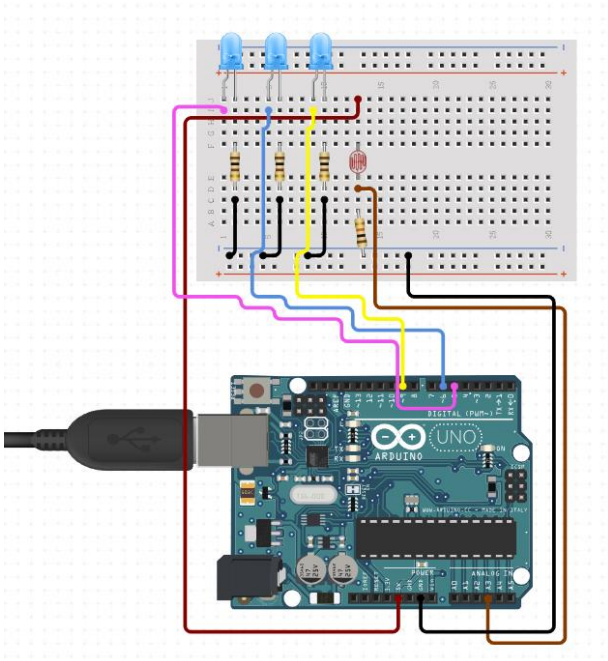

Fig1. Multiple LEDs and LDR Sensor connected to Arduino UNO using a breadboard

Code:

```
const int LEDPin1 = 5;
const int LEDPin2 = 6;
const int LEDPin3 = 7;
const int LDRPin = A0;

void setup() {
  Serial.begin(9600);
  pinMode(LEDPin1, OUTPUT); // Define pin as output
  pinMode(LEDPin2, OUTPUT);
  pinMode(LEDPin3, OUTPUT);
  pinMode(LDRPin, INPUT); // Define LDR pin as input
}

void loop() {
int ldrStatus = analogRead(LDRPin);
Serial.println(ldrStatus);
delay(1000);
if (ldrStatus >200 && ldrStatus<400){
  digitalWrite(LEDPin1, HIGH);
}
else if(ldrStatus>400 && ldrStatus<=600){
  digitalWrite(LEDPin1, HIGH);
  digitalWrite(LEDPin2, HIGH);
}
else if(ldrStatus>600){
  digitalWrite(LEDPin1, HIGH);
  digitalWrite(LEDPin2, HIGH);
  digitalWrite(LEDPin3, HIGH);
}
else{
  digitalWrite(LEDPin1, LOW);
  digitalWrite(LEDPin2, LOW);
  digitalWrite(LEDPin3, LOW);
}
}
```

Output:

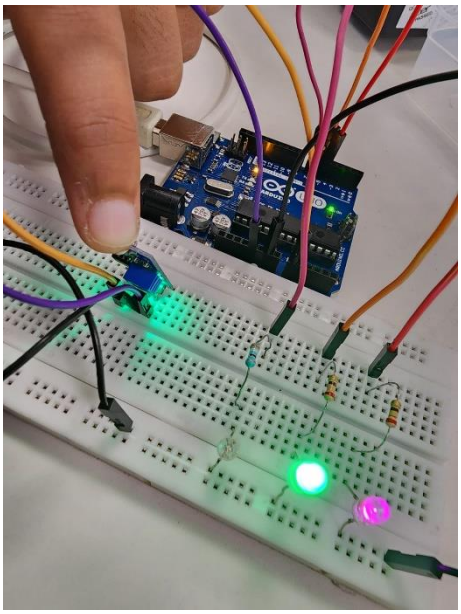

Fig2. LEDs will turn ON depending upon the analog readings of LDR

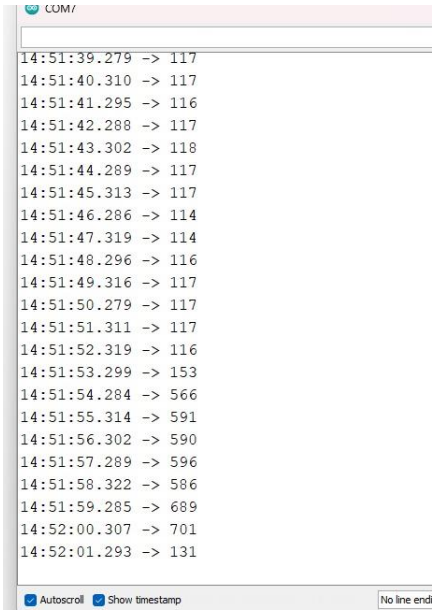

Fig3. Serial Monitor Output

Conclusions: The experiment successfully demonstrated a multi-level light-dependent control system using an LDR as an analog input. By setting specific light thresholds, we were able to control multiple LEDs based on varying ambient light conditions. The experiment provided a practical understanding of analog input/output operations and effectively showcased how to manipulate multiple outputs (LEDs) according to different light intensity levels. This serves as a foundational step towards more complex light-based projects that require precise and graduated control over multiple outputs.

Experiment: 5

Objective: To interface the DHT11 sensor with Arduino, write a program to print temperature and humidity readings on a serial monitor.

Software Used: Arduino IDE

Components Used: Arduino UNO R3, breadboard, jumper wires, DHT11 sensor

Theory:

The DHT11 sensor is a widely used device for measuring temperature and humidity, making it a popular choice in DIY electronics and hobbyist projects. It offers reliable readings within a temperature range of 0°C to 50°C and a humidity range of 20% to 90%, with a simple digital output that facilitates easy integration with microcontrollers like Arduino. The DHT11 operates using a capacitive humidity sensor and a thermistor to detect changes in environmental conditions, converting these into a digital signal that can be processed by a microcontroller.

Circuit Diagram:

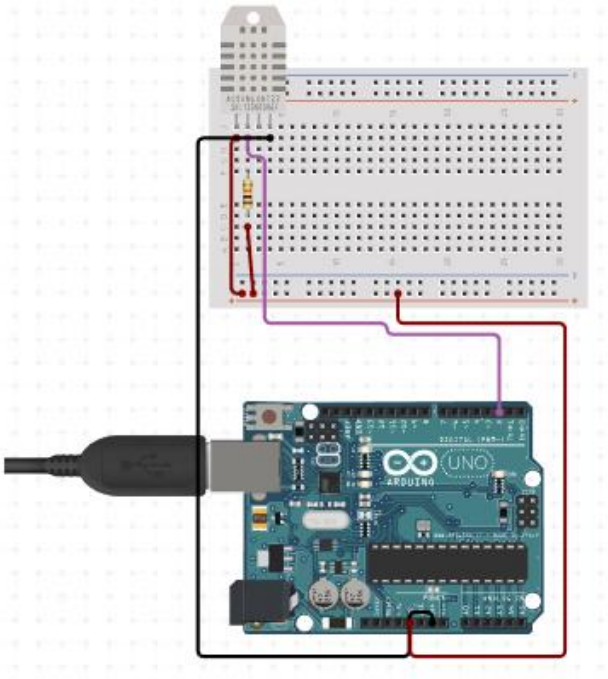

Fig1. DHT11 Sensor connected to Arduino UNO using a breadboard

Code:

```
// include the EduIntro library
#include <EduIntro.h>
```

```
DHT11 dht11(D7);  // creating the object sensor on pin 'D7'
int C;   // temperature C readings are integers
float F; // temperature F readings are returned in float format
int H;   // humidity readings are integers
void setup()
{
  // initialize serial communications at 9600 bps
  Serial.begin(9600);
}
void loop()
{
  dht11.update();
  C = dht11.readCelsius();
  F = dht11.readFahrenheit();
  H = dht11.readHumidity();
  Serial.print("H: ");
  Serial.print(H);
  Serial.print("\tC: ");
  Serial.print(C);
  Serial.print("\tF: ");
  Serial.println(F);
  delay(1000);              // Wait one second before get another temperature reading
}
```

Output:-

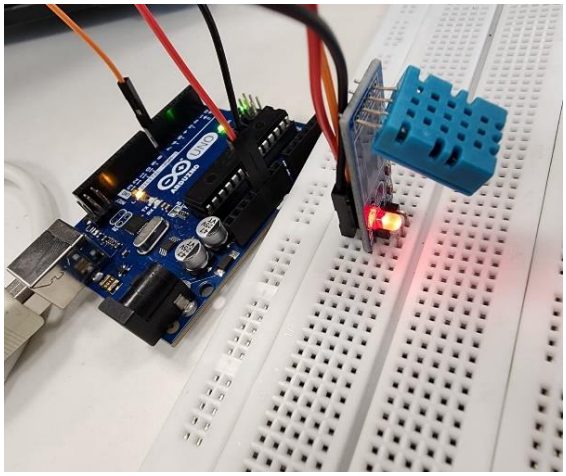

Fig2. DHT11 sensor measuring the parameters

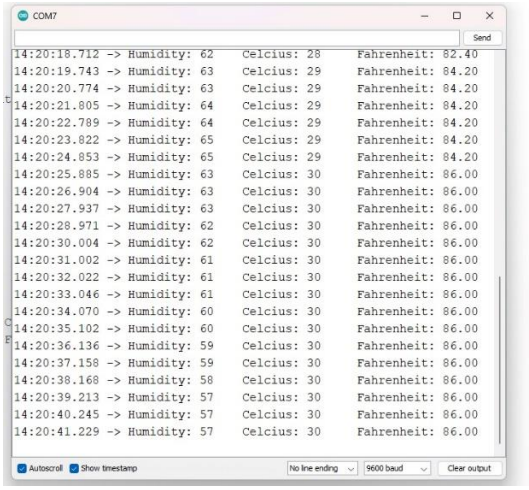

Fig3. Serial Monitor Output

Conclusions: The experiment successfully demonstrated the interfacing of a DHT11 sensor with an Arduino to measure and display temperature and humidity readings on a serial monitor. By utilizing the DHT library, we were able to efficiently read and process environmental data, showcasing how digital sensors can be used for real-time monitoring. The experiment provided a practical understanding of sensor integration and serial communication, effectively illustrating how to capture and display environmental conditions through simple programming. This serves as a foundational step towards more advanced projects involving environmental monitoring and control systems.

Experiment: 6

Objective: According to the DHT readings. write a program to turn ON & OFF multiple LEDs.

Software Used: Arduino IDE

Components Used: Arduino UNO R3, breadboard, jumper wires, DHT11 sensor, LEDs

Theory:

The DHT11 sensor is a widely used device for measuring temperature and humidity, making it a popular choice in DIY electronics and hobbyist projects. It offers reliable readings within a temperature range of 0°C to 50°C and a humidity range of 20% to 90%, with a simple digital output that facilitates easy integration with microcontrollers like Arduino. The DHT11 operates using a capacitive humidity sensor and a thermistor to detect changes in environmental conditions, converting these into a digital signal that can be processed by a microcontroller. Its ease of use, affordability, and straightforward single-wire communication make it an excellent tool for applications such as weather monitoring, home automation, and greenhouse management.

Circuit Diagram:

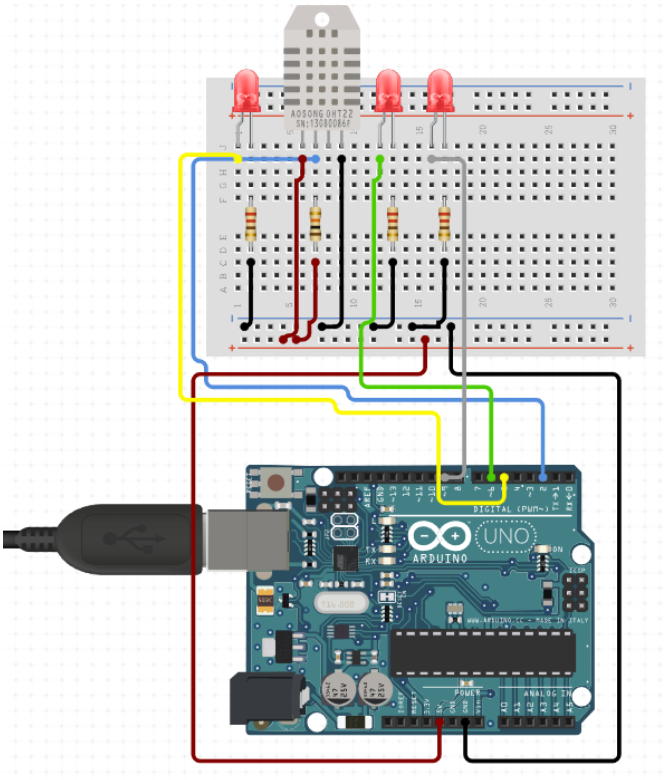

Fig1. DHT11 Sensor and LEDs connected to Arduino UNO using a breadboard

Code:

```
#include <DFRobot_DHT11.h>
DFRobot_DHT11 DHT;
#define DHT11_PIN 6
int redLED1 = 3;
int redLED2 = 4;
int redLED3 = 5;
void setup()
{
Serial.begin(9600);
pinMode(DHT11_PIN, INPUT);
pinMode(redLED1, OUTPUT);
pinMode(redLED2, OUTPUT);
pinMode(redLED3, OUTPUT);
}
void loop()
{
DHT.read(DHT11_PIN);
Serial.print("temperature:");
Serial.print(DHT.temperature);
Serial.print(", humidity:");
Serial.println(DHT.humidity);
if(DHT.humidity >= 80 || DHT.temperature >= 30)
{
digitalWrite(redLED1, HIGH);
digitalWrite(redLED2, HIGH);
digitalWrite(redLED3, HIGH);
}
else if(DHT.humidity >= 70 || DHT.temperature
< 26)
{
digitalWrite(redLED1, LOW);
digitalWrite(redLED2, HIGH);
digitalWrite(redLED3, HIGH);
}
else
{
digitalWrite(redLED1, LOW);
```

```
digitalWrite(redLED2, LOW);
digitalWrite(redLED3, HIGH);
}
delay(1000);
}
```

Output:

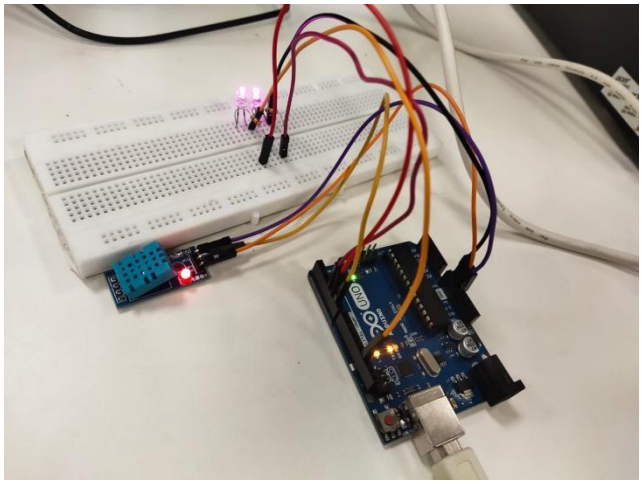

Fig2. DHT11 sensor measuring the parameters and LEDs turned ON accordingly

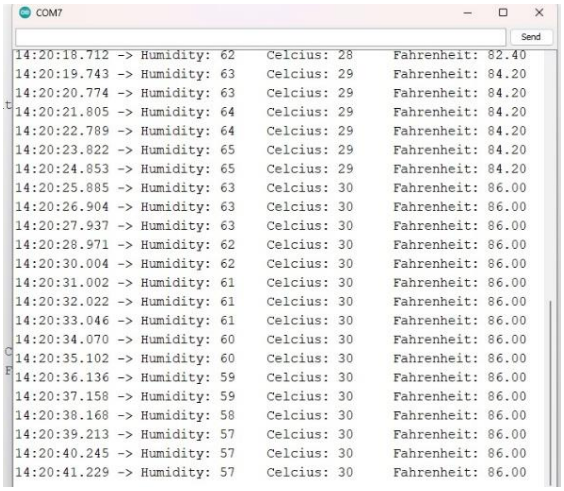

Fig3. Serial Monitor Output

Conclusions: The experiment successfully demonstrated the integration of a DHT11 sensor with an Arduino to control multiple LEDs based on temperature and humidity readings. By setting specific thresholds, the LEDs were activated or deactivated accordingly, showcasing the practical application of sensor data in environmental control systems. This experiment provided a foundational understanding of how digital sensors can be used for real-time monitoring and automated responses, laying the groundwork for more complex projects in environmental monitoring and smart systems.

Experiment: 7

Objective: To interface the HC-05 Bluetooth module with Arduino, write a program to turn ON & OFF multiple LEDs with a smartphone using Bluetooth.

Software Used: Arduino IDE, Arduino Bluetooth Controller(Mobile App)

Components Used: Arduino UNO R3, breadboard, jumper wires, HC-05 sensor, LED

Theory:

The HC-05 is a Bluetooth module commonly used for wireless communication between devices in various embedded systems and IoT applications. It operates on Bluetooth 2.0 technology, providing a cost-effective solution for establishing a serial communication link between microcontrollers, such as Arduino, and Bluetooth-enabled devices like smartphones or computers. The module supports both master and slave modes, allowing it to either initiate or receive connections. It operates over a 3.3V to 5V range and communicates via UART (Universal Asynchronous Receiver-Transmitter), making it compatible with most microcontroller platforms. Its versatility and ease of use make it a popular choice for remote control, data transmission, and wireless automation projects.

Circuit Diagram:

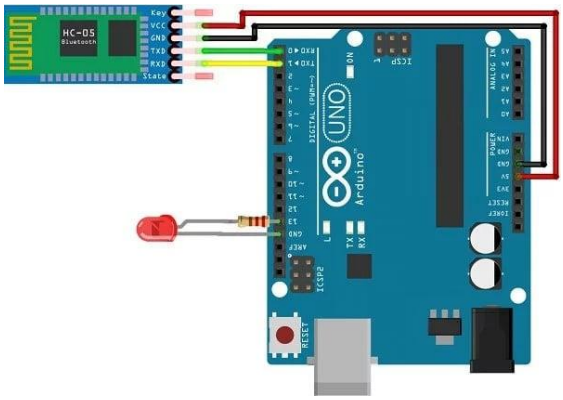

Fig1. HC-05 Bluetooth Sensor and LED connected to Arduino UNO using a breadboard

Code:

```
#define LED_PIN 9
void setup() {
  Serial.begin(9600);
  pinMode(LED_PIN, OUTPUT);
}
```

```
void loop() {
  if (Serial.available())
  {
    String command = Serial.readStringUntil('\n');
    if (command == "OFF")
    {
      digitalWrite(LED_PIN, LOW); // turn off LED
      Serial.println("LED is turned OFF");
    }
    else if (command == "ON")
    {
      digitalWrite(LED_PIN, HIGH); // turn on LED
      Serial.println("LED is turned ON");
    }
  }
}
```

Output:

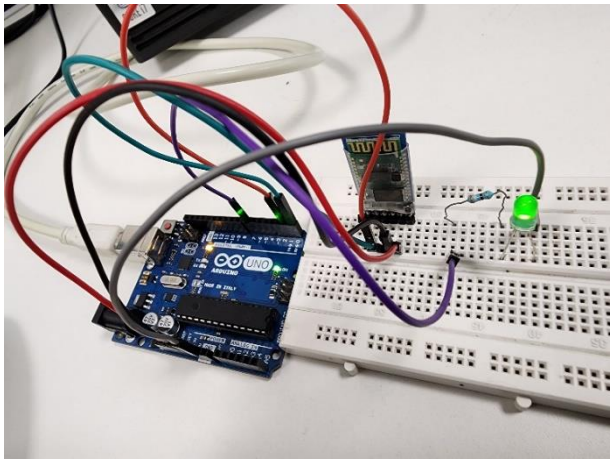

Fig2. LED turned ON from mobile app using HC-05 Bluetooth sensor

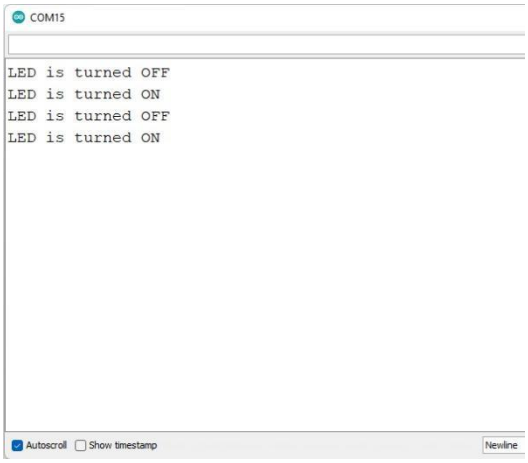

Fig3. Serial Monitor Output

Conclusions: In this experiment, we successfully demonstrated how to interface an HC-05 Bluetooth module with Arduino to wirelessly control an LED via a smartphone. By using basic serial communication, the module received specific commands from a Bluetooth terminal, allowing for precise control of each LED. This setup provides a foundation for building more advanced wireless control systems in IoT projects.

References:

- https://www.geeksforgeeks.org/all-about-hc-05-bluetooth-module-connection-with-android/
- https://www.arduino.cc/