**Experiment 1 Aim:**

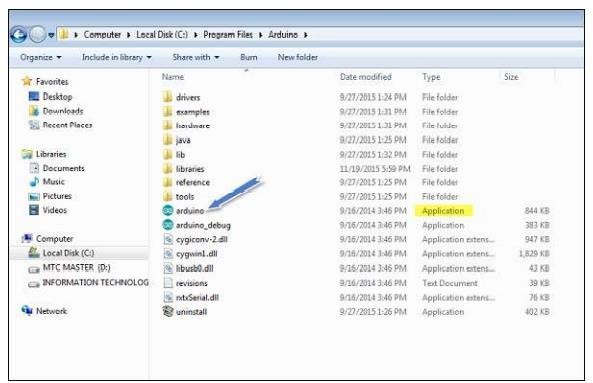
**To familiarize myself with Arduino and perform the necessary software installation.**

**Description:**

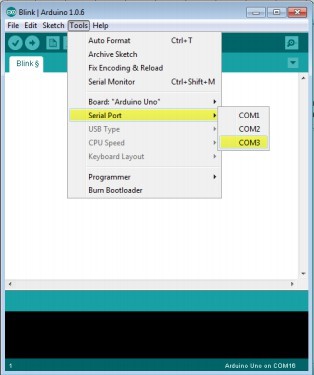
In this experiment, I learned about Arduino and Raspberry Pi platforms. I installed the Arduino IDE software, explored its environment, and executed a sample program. This helped me understand how to set up the development environment and upload a basic program to the Arduino board.

**Procedure:**

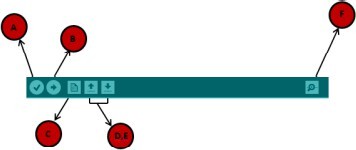
1. I downloaded the Arduino IDE software from the official website “https://[www.arduino.cc](http://www.arduino.cc/)”.
2. I launched the Arduino IDE on my system.
3. After the download, I unzipped the folder and located the application file with the infinity label (application.exe). I double-clicked on the icon to start the IDE.



1. Once the IDE opened, I had two options: create a new project or open an existing example. I chose to open an existing project by navigating to **File → Examples → Basics → Blink**. This program makes the LED turn on and off with a time delay.

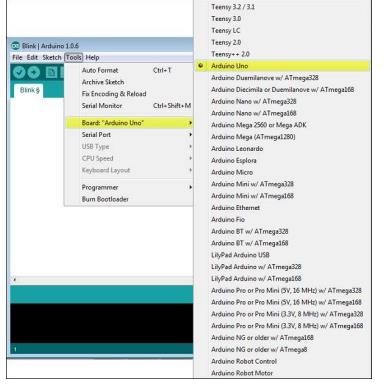


1. I selected my Arduino board by going to **Tools → Board** and choosing **Arduino Uno**, as it matched the board I was using.
2. I selected the correct serial port by navigating to **Tools → Serial Port**. I verified this by disconnecting and reconnecting my Arduino board and then choosing the correct COM port.
3. I became familiar with the Arduino IDE toolbar functions:



* 1. Verify (check for errors)
  2. Upload (send program to the board)
  3. New Sketch (create a new project)
  4. Open (open existing examples)
  5. Save (save the sketch)
  6. Serial Monitor (communicate with the board)

1. clicked the **Upload** button to send the Blink program to the Arduino board.



1. observed the RX and TX LEDs flashing on the board. After a few seconds, the message **“Done uploading”** appeared in the status bar.

**Result:**

I successfully installed the Arduino IDE, configured the board and serial port, and uploaded the Blink program. The Arduino Uno executed the program correctly, and the onboard LED blinked with a time delay.

**Conclusion:**

Through this experiment, I understood how to install and set up the Arduino IDE, select the appropriate board and port, and upload a program to the Arduino. This gave me the basic knowledge required to start working on IoT applications.

**Experiment 2 Aim:**

**To interface an LED/Buzzer with Arduino and write a program to turn ON the LED for 1 second after every 2 seconds.**

**Requirements:**

* Standard 5mm LED
* One Resistor
* Arduino Uno microcontroller board (ATmega328p)
* Jumper wires

**Description:**

In this experiment, I interfaced an LED with the Arduino Uno microcontroller. I connected the LED through a resistor to avoid excess current and wrote a program to blink the LED such that it stays ON for 1 second and remains OFF for 2 seconds in a continuous loop. This simple task helped me understand how digital pins control external devices in IoT applications.

**Procedure:**

1. I opened [www.wokwi.com](http://www.wokwi.com/) in my browser and selected the **Arduino Uno**

microcontroller.

1. In the simulation section, I added the required components (LED, resistor, jumper wires, and Arduino Uno) by clicking the **“+”** symbol.
2. I used jumper wires to make the connections: I connected the **Anode (A-pin)** of the LED to **digital pin 3** of the Arduino board through the resistor, and I connected the **Cathode (C-pin)** of the LED to **Ground (GND)**.
3. I wrote the following program in the **sketch.ino** file:

**Code:**

/\*

Experiment: Interfacing an LED with Arduino Uno

Purpose: To turn ON the LED for 1 second after every 2 seconds Name: P.AKSHAYA

Roll No: 24M11MC126

\*/

void setup() {

pinMode(3, OUTPUT); // Set pin 3 as OUTPUT for LED

}

void loop() {

digitalWrite(3, HIGH); // Turn LED ON delay(1000); // Keep ON for 1 second digitalWrite(3, LOW); // Turn LED OFF delay(2000); // Keep OFF for 2 seconds

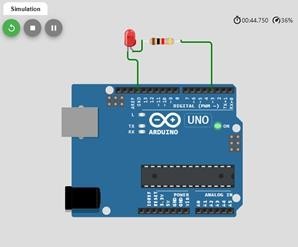
}

1. I compiled and uploaded the program to the Arduino Uno board in the simulation.
2. I observed the LED blinking – it turned ON for 1 second and remained OFF for 2 seconds repeatedly.

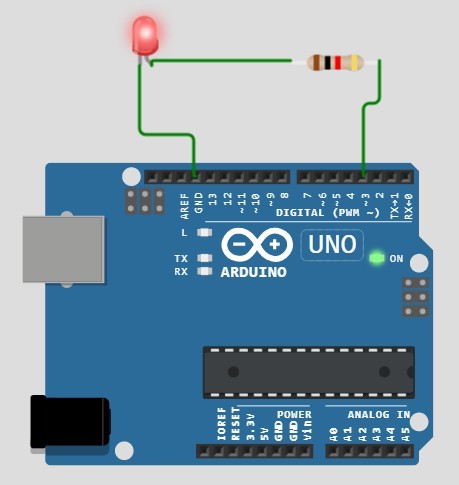
**Result:**

I successfully interfaced the LED with the Arduino Uno board and observed that the LED turned ON for 1 second and remained OFF for 2 seconds, working as expected.

**Before:**

****

**After:**

****

**Conclusion:**

Through this experiment, I learned how to interface external output devices like LEDs with Arduino and control them using digital pins. I also understood how to use pinMode(), digitalWrite(), and delay() functions for timing control in Arduino programs.

**Experiment 3 Aim:**

**To interface a Push Button/Digital Sensor (IR/LDR) with Arduino and write a program to turn ON the LED when the push button is pressed or at sensor detection.**

**Requirements:**

* Standard 5mm LED
* Two Resistors
* Arduino Uno microcontroller board (ATmega328p)
* Jumper wires
* Push button

**Description:**

In this experiment, I interfaced a push button (digital input device) with the Arduino Uno microcontroller to control an LED. The LED turned ON when the push button was pressed. This experiment helped me understand how to use digital inputs in Arduino, read button states, and control output devices based on user interaction or sensor detection.

**Procedure:**

1. I opened [www.wokwi.com](http://www.wokwi.com/) in my browser and selected the **Arduino Uno**

microcontroller.

1. In the simulation section, I added the required components (LED, resistors, jumper wires, push button, and Arduino Uno) by clicking the **“+”** symbol.
2. I connected the **Anode (A-pin)** of the LED to **digital pin 3** of the Arduino microcontroller through **resistor1**, and I connected the **Cathode (C-pin)** of the LED to **Ground (GND)**.
3. I provided a **5V power supply** to the push button pin **1:1L**.
4. I connected the push button pin **1:1R** both to **Ground through resistor2** and to

**digital pin 5** of the Arduino board.

1. I wrote the following program in **sketch.ino**:

**Code:**

/\*Experiment 3: Interfacing a Push Button/Digital Sensor with Arduino Uno

Purpose: To turn ON the LED when the push button is pressed or at sensor detection

Name: P.AKSHAYA

Roll No: 24M11MC126

\*/

int flag = 0; void setup() {

pinMode(3, OUTPUT); // LED connected to pin 3 pinMode(5, INPUT); // Push button connected to pin 5

}

void loop() {

int value = digitalRead(5); // Read push button state

if (value == 1 && flag == 0) { digitalWrite(3, HIGH); // Turn LED ON delay(1000);

flag = 1;

}

else if (value == 1 && flag == 1) { digitalWrite(3, LOW); // Turn LED OFF delay(1000);

flag = 0;

}

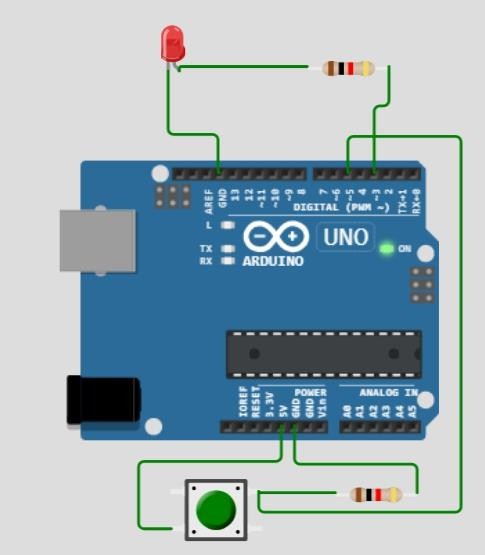
}

1. I compiled and uploaded the program to the Arduino Uno board in the simulation.
2. I observed that when I pressed the push button, the LED turned ON. On pressing it again, the LED turned OFF.

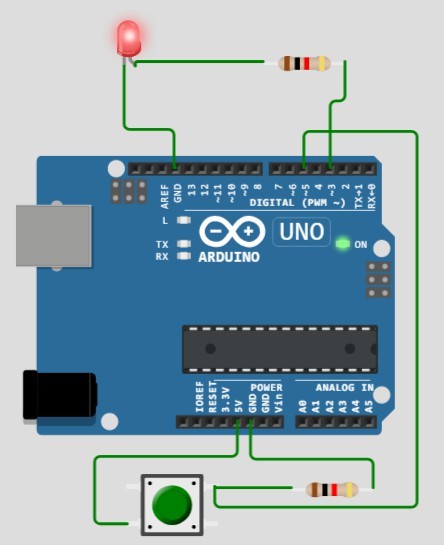
**Result:**

I successfully interfaced a push button with the Arduino Uno. When I pressed the push button, the LED turned ON, and on pressing again, it turned OFF.

**Before:**

****

**After:**

****

**Conclusion:**

Through this experiment, I learned how to interface a push button (digital input) with Arduino and use its state to control an output device. I understood the use of digitalRead() for inputs and how conditional logic can toggle device states in IoT applications.

**Experiment 4 Aim:**

**To interface the DHT22 sensor with Arduino and write a program to print temperature and humidity readings.**

**Requirements:**

* DHT22 sensor
* Arduino Uno microcontroller board (ATmega328p)
* Jumper wires

**Description:**

In this experiment, I interfaced a DHT22 sensor with the Arduino Uno to measure environmental temperature and humidity. The DHT22 sensor is widely used in IoT projects for weather monitoring and home automation.

**Procedure:**

1. I opened [www.wokwi.com](http://www.wokwi.com/) in my browser and selected the **Arduino Uno**

microcontroller.

1. In the simulation section, I added the required components (DHT22 sensor, jumper wires, and Arduino Uno) by clicking the **“+”** symbol.
2. I made the following connections using jumper wires:
   * I connected **VCC pin of DHT22** to **5V power supply** of Arduino.
   * I connected **SDA pin of DHT22** to **digital pin 3** of Arduino Uno.
   * I connected **GND pin of DHT22** to **Ground (GND)** of Arduino Uno.
3. I wrote the following program in **sketch.ino**:

**Code:**

/\*

Experiment 4: Interfacing the DHT22 sensor with Arduino Uno Purpose: To print temperature and humidity readings

Name: P.AKSHAYA

Roll No: 24M11MC126

\*/

#include <Adafruit\_Sensor.h> #include <DHT.h>

#include <DHT\_U.h>

DHT Sensor(3, DHT22);

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

// Display student details before starting readings Serial.println("=====================================");

Serial.println(" Experiment 4: DHT22 Sensor Interface "); Serial.println(" Name : MD SHAMEEM"); Serial.println(" Roll No : 24M11MC102");

Serial.println("=====================================");

delay(2000); // Pause for 2 seconds before readings

}

void loop() {

float tempr = Sensor.readTemperature(); float humid = Sensor.readHumidity();

Serial.print("Temperature = "); Serial.print(tempr); Serial.print(" °C, "); Serial.print("Humidity = "); Serial.print(humid); Serial.println(" %");

delay(1000); // Update every second

}

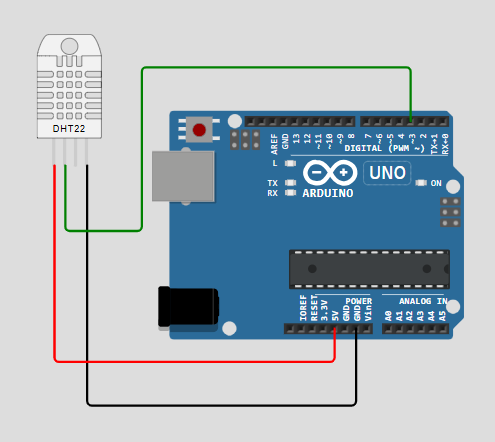
1. I compiled and uploaded the program to the Arduino Uno in the simulation.
2. I opened the **Serial Monitor** and observed the temperature and humidity values displayed continuously.

**Result:**

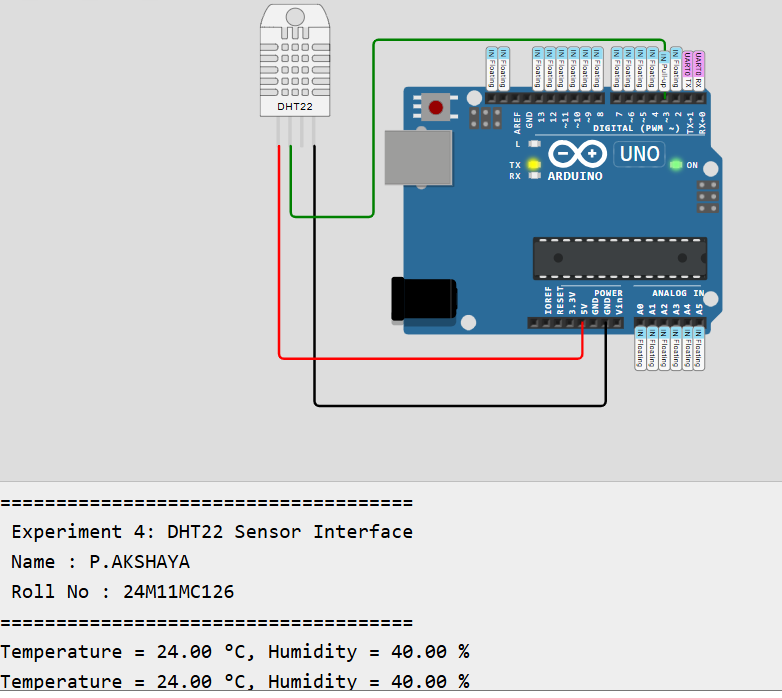
I successfully interfaced the DHT22 sensor with the Arduino Uno and observed real-time

**temperature and humidity readings** displayed on the serial monitor.

**Before:**

****

**After:**



**Conclusion:**

From this experiment, I understood how to interface digital sensors like the DHT22 with Arduino. I learned to use sensor libraries, read environmental parameters, and display them through the serial monitor, which is useful in IoT-based weather monitoring and automation systems.

**Experiment 5 Aim:**

**To interface an OLED with Arduino and write a program to print temperature and humidity readings on it.**

**Requirements:**

* DHT22 sensor
* Arduino Uno microcontroller board (ATmega328p)
* Jumper wires
* SSD1306 OLED displa

**Description:**

In this experiment, I interfaced an OLED display with the Arduino Uno along with the DHT22 sensor. The DHT22 provided real-time temperature and humidity values, and the OLED displayed these readings. This experiment helped me understand how to use I²C communication in Arduino and how to display sensor data on graphical displays, which is useful in IoT applications like smart weather stations and home automation systems.

**Procedure:**

1. I opened [www.wokwi.com](http://www.wokwi.com/) in my browser and selected the **Arduino Uno**

microcontroller.

1. In the simulation section, I added the required components (DHT22 sensor, SSD1306 OLED, jumper wires, and Arduino Uno) by clicking the **“+”** symbol.
2. I made the following connections using jumper wires:
   1. I connected **VCC pin of DHT22** to **5V power supply** of Arduino.
   2. I connected **SDA pin of DHT22** to **digital pin 3** of Arduino.
   3. I connected **GND pin of DHT22** to **Ground (GND)** of Arduino.
   4. I connected the **OLED display** as follows:
      1. OLED DATA → Analog pin **A4**
      2. OLED CLK → Analog pin **A5**
      3. OLED VIN → **5V power**
      4. OLED GND → **Ground**
3. I downloaded and included the **Adafruit SSD1306**, **Adafruit GFX**, and **DHT sensor libraries** in the Arduino IDE.
4. I wrote the following program in **sketch.ino**:

/\*

Experiment 5: Interfacing OLED with Arduino Uno

Purpose: To print temperature and humidity readings on OLED Name: P.AKSHAYA

Roll No: 24M11MC126

\*/

#include <Adafruit\_Sensor.h> #include <Adafruit\_GFX.h> #include <Adafruit\_SSD1306.h> #include <DHT.h>

#include <DHT\_U.h> #include <Wire.h>

// OLED settings

Adafruit\_SSD1306 display(128, 64, &Wire);

// DHT22 settings

DHT Sensor(3, DHT22);

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

// Initialize OLED

if (!display.begin(SSD1306\_SWITCHCAPVCC, 0x3C)) { Serial.println("SSD1306 allocation failed");

for (;;);

}

display.clearDisplay(); display.setTextColor(WHITE);

// Show student details first display.setTextSize(1); display.setCursor(0, 10);

display.println("Experiment 5"); display.println("OLED with DHT22"); display.println(" ");

display.println("Name: MD SHAMEEM"); display.println("Roll No: 24M11MC102"); display.display();

delay(3000); // Show info for 3 seconds display.clearDisplay();

}

void loop() {

float tempr = Sensor.readTemperature(); float humid = Sensor.readHumidity();

// Display sensor readings display.setTextSize(1); display.setCursor(0, 10); display.print("Temp: "); display.print(tempr); display.println(" C");

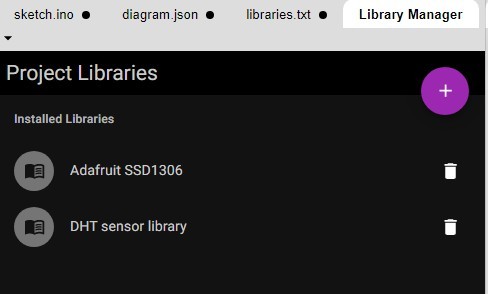
display.print("Humidity: "); display.print(humid); display.println(" %");

display.display();

delay(1000); // Update every second display.clearDisplay();

}

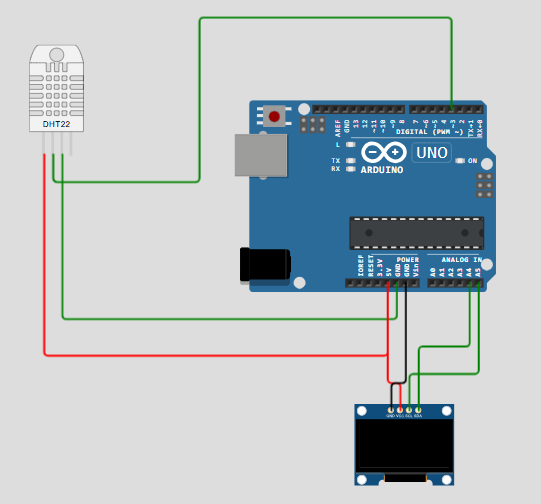
1. I compiled and uploaded the program to the Arduino Uno in the simulation.
2. I observed that the OLED displayed the real-time **temperature and humidity values** from the DHT22 sensor.



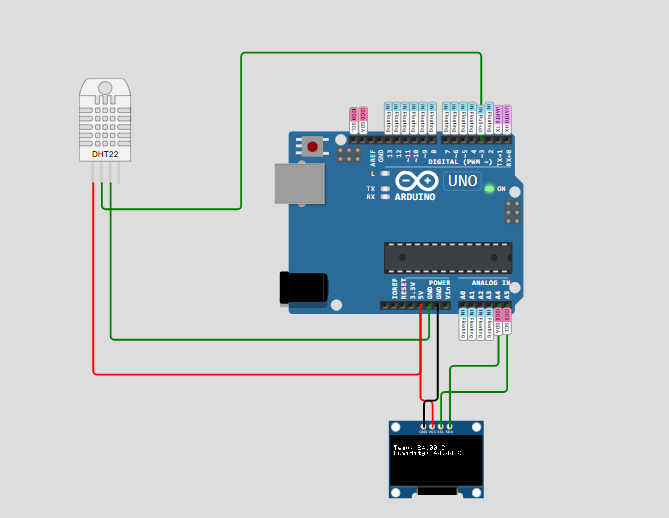
**Result:**

I successfully interfaced the OLED display with the Arduino Uno and observed that the real-time temperature and humidity readings from the DHT22 sensor were displayed on the OLED screen.

**Before:**



**After:**





**Conclusion:**

Through this experiment, I learned how to interface and program an OLED display with Arduino using the I²C protocol. I also understood how to collect real-time data from a sensor and display it on a graphical interface, which is an essential part of IoT applications.