www.sjcit.ac.in

|| Jai Sri Gurudev ||
Sri Adichunchanagiri Shikshana Trust®

SJC INSTITUTE OF TECHNOLOGY

An Autonomous Institution under VTU from 2024-25

AICTE Approved, Accredited by NBA [CSE, ISE, ECE, ME, CV, AE] and NAAC with A+ Grade, QS I-Gauge Gold rated

P.B. No. 20, B.B. Road, Chikkaballapur - 562 101, Karnataka.

Estd. 1986

Department Of

Computer Science and Engineering

VII SEMESTER

Internet of Things BCS 701

SJC INSTITUTE OF TECHNOLOGY

VISION

Preparing Competent Engineering and Management Professionals to Serve the Society.

MISSION

- **Providing Students with Sound Knowledge in Fundamentals of their branch of Study.**
- **Promoting Excellence in Teaching, Training, Research and Consult**
- **Exposing Students to Emerging Frontiers in various domains enabling Continuous Learning.**
- **Developing Entrepreneurial acumen to venture into Innovative areas.**
- **Imparting Value based Professional Education with a sense of Social Responsibility.**

Department of C S E

Vision Building up-skilled Computer Professionals, enriched with interactive design skills as to serve the dynamic needs of the Industry and Society.

Mission

- **Develop innovative, Proficient and ethically strong Computer Design Engineers with design skills to meet Universal Challenges.**
- **Nurture competence as well as applied Research activities to solve concrete problems.**
- **Aspire for constant up-gradation of engineering skills to cater the needs of the corporate and Society.**
- **Imbibing the spirit of teamwork, core skills, professionalism and confidence to the Leadership role.**
- **Inculcate research culture to design and develop smart computing solutions for humanity and nation.**

INSTRUCTIONS

1. **Arrive on time with all necessary materials like Pen, Observation.**
2. **Log in with your assigned credentials and log out before leaving.**
3. **Save your work frequently and maintain backups in a folder named with your USN.**
4. **Follow the given tasks and faculty instructions during the session.**
5. **Write clean, structured, and well-commented code.**
6. **Seek help from faculty or lab assistants when facing difficulties.**
7. **Maintain silence and avoid distracting others.**
8. **Negligence of one candidate will result in penalty for the whole batch.**
9. **Use the systems only for academic purposes and avoid unauthorized installations.**
10. **Write original code and avoid copying from peers or online sources.**
11. **Log out, clean your workspace, and leave the equipment in good condition.**

||Jai Srigurudev||
SJC Institute of Technology, Chickballapur

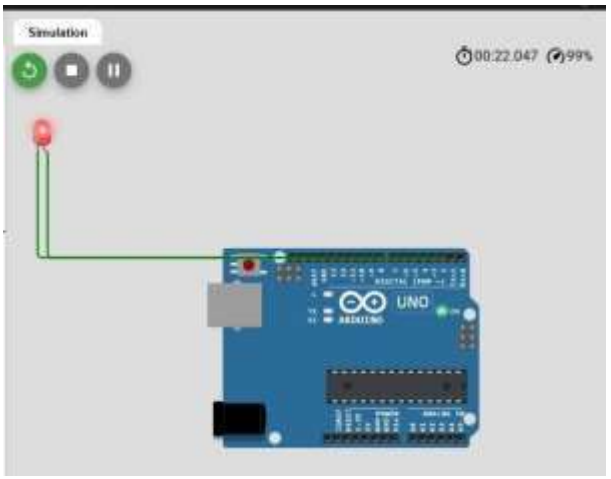
Computer Science and Engineering
7th semester
Internet of Things (BCS701)

PRACTICAL COMPONENT OF IPCC

1. Develop a program to blink 5 LEDs back and forth.
2. Develop a program to interface a relay with Arduino board.
3. Develop a program to deploy an intrusion detection system using Ultrasonic and sound sensors.
4. Develop a program to control a DC motor with Arduino board.
5. Develop a program to deploy smart street light system using LDR sensor.
6. Develop a program to classify dry and wet waste with the Moisture sensor (DHT22).
7. Develop a program to read the pH value of a various substances like milk, lime and water.
8. Develop a program to detect the gas leakage in the surrounding environment.
9. Develop a program to demonstrate weather station readings using Arduino.
10. Develop a program to setup a UART protocol and pass a string through the protocol.
11. Develop a water level depth detection system using Ultrasonic sensor.
12. Develop a program to simulate interfacing with the keypad module to record the keystrokes.

Sample Program 1

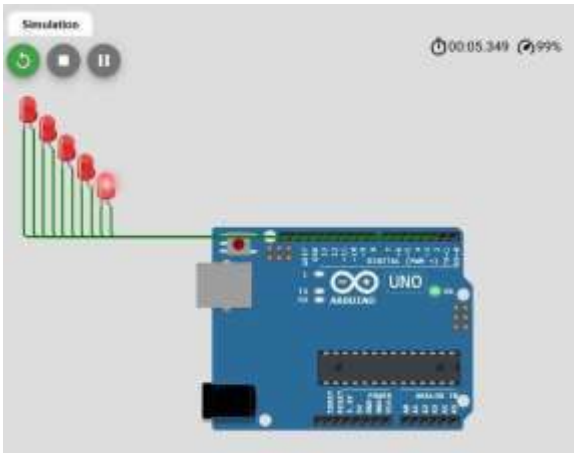
```
// Define LED pins
int led1 = 2;  // LED1 connected to pin 2

void setup() {
  // Set pins as output
  pinMode(led1, OUTPUT);
                }

void loop() {
  // Blink LEDs one by one
  digitalWrite(led1, HIGH);
  delay(500);
  digitalWrite(led1, LOW);
  delay(500);
                }
```



Sample Program 2 (5LEDs)

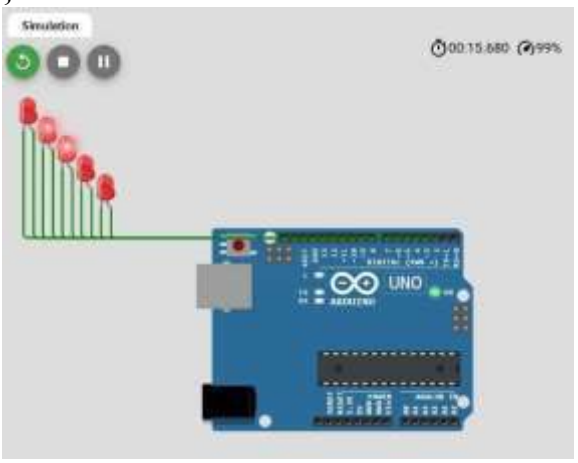
```
// Define LED pins
int led1 = 2;  // LED1 connected to pin 2
int led2 = 3;  // LED2 connected to pin 3
int led3 = 4;  // LED3 connected to pin 4
int led4 = 5;  // LED4 connected to pin 5
int led5 = 6;  // LED5 connected to pin 6

void setup() {
  // Set pins as output
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
```

```
  pinMode(led4, OUTPUT);
  pinMode(led5, OUTPUT);
}

void loop() {
  // Blink LEDs one by one
  digitalWrite(led1, HIGH);
  delay(500);
  digitalWrite(led1, LOW);

  digitalWrite(led2, HIGH);
  delay(500);
  digitalWrite(led2, LOW);

  digitalWrite(led3, HIGH);
  delay(500);
  digitalWrite(led3, LOW);

  digitalWrite(led4, HIGH);
  delay(500);
  digitalWrite(led4, LOW);

  digitalWrite(led5, HIGH);
  delay(500);
  digitalWrite(led5, LOW);
}
```



1. Develop a program to blink 5 LEDs back and forth.

```
// Define LED pins in an array
int leds[] = {2, 3, 4, 5, 6};
int numLeds = 5;   // total LEDs
int delayTime = 200;  // delay in ms

void setup() {
  // Set all LED pins as OUTPUT
  for (int i = 0; i < numLeds; i++) {
  pinMode(leds[i], OUTPUT);
  }
}

void loop() {
  // Move forward
  for (int i = 0; i < numLeds; i++) {
    digitalWrite(leds[i], HIGH);
    delay(delayTime);
    digitalWrite(leds[i], LOW);
  }

  // Move backward
  for (int i = numLeds - 2; i > 0; i--) {
    digitalWrite(leds[i], HIGH);
    delay(delayTime);
    digitalWrite(leds[i], LOW);
  }
}
```

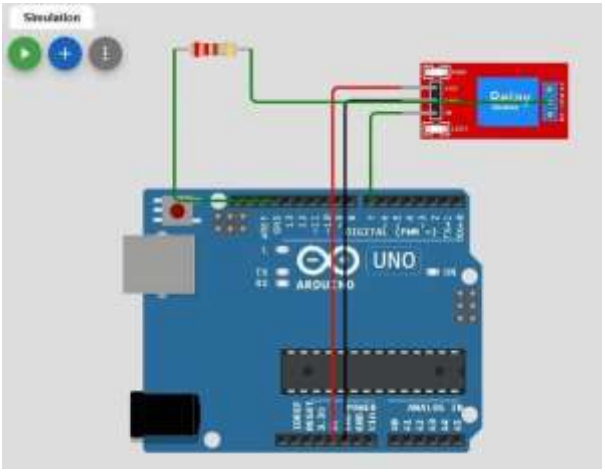


2. Develop a program to interface a relay with Arduino board.

```
// Relay Interfacing with Arduino (Wokwi / Hardware)
int relayPin = 7;  // Relay connected to digital pin 7
void setup() {
  pinMode(relayPin, OUTPUT);   // Set relay pin as output
}
void loop() {
  digitalWrite(relayPin, HIGH); // Turn ON relay
  delay(2000);                  // Wait 2 seconds

  digitalWrite(relayPin, LOW);  // Turn OFF relay
  delay(2000);                  // Wait 2 seconds
}
```

Connections:

- Relay **IN → Arduino pin 7**
- Relay **VCC → 5V**
- Relay **GND → GND**
- LED (Load) → Connect to relay **NO (Normally Open)** and **COM** terminals with a resistor.



3. Develop a program to deploy an intrusion detection system using Ultrasonic and sound sensors.

```
// Intrusion Detection System using Ultrasonic + Sound Sensor
// Works in Wokwi (https://wokwi.com)

#define TRIG_PIN 9
#define ECHO_PIN 8
#define SOUND_SENSOR A0
#define BUZZER 7
#define LED 6

long duration;
int distance;
int soundValue;

void setup() {
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(BUZZER, OUTPUT);
  pinMode(LED, OUTPUT);
  pinMode(SOUND_SENSOR, INPUT);

  Serial.begin(9600);
}

void loop() {
  // --- Ultrasonic Sensor Measurement ---
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  duration = pulseIn(ECHO_PIN, HIGH);
  distance = duration * 0.034 / 2;  // cm

  // --- Sound Sensor Measurement ---
  soundValue = analogRead(SOUND_SENSOR);

  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.print(" cm | Sound Value: ");
  Serial.println(soundValue);

  // --- Intrusion Detection Logic ---
  if (distance < 30 || soundValue > 500) {  // Adjust thresholds
    digitalWrite(BUZZER, HIGH);
    digitalWrite(LED, HIGH);
```

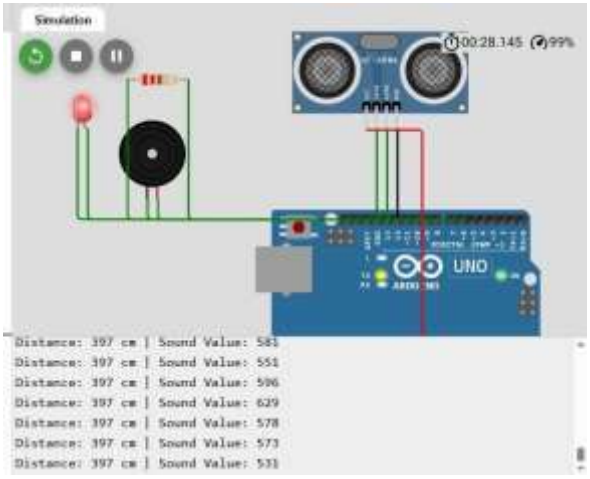
```
  } else {
    digitalWrite(BUZZER, LOW);
    digitalWrite(LED, LOW);
  }

  delay(200);
}
```

Components:

- Arduino Uno
- HC-SR04 Ultrasonic Sensor
- Sound Sensor (use Microphone Sensor in Wokwi)
- Buzzer
- LED + Resistor
- Breadboard + Wires

Wire connections:

- **Ultrasonic Sensor**
  - VCC → 5V
  - GND → GND
  - TRIG → D9
  - ECHO → D8
- **Sound Sensor**
  - VCC → 5V
  - GND → GND
  - OUT (Analog) → A0
- **Buzzer** → D7
- **LED + Resistor** → D6



4. Develop a program to control a DC motor with Arduino board.

```
// DC Motor Control using L293D Motor Driver and Arduino
int motorPin1 = 9;  // L293D Pin 2
int motorPin2 = 10;  // L293D Pin 7
int enablePin = 11;  // L293D Pin 1 (Enable A)
void setup() {
  // Set motor control pins as output
  pinMode(motorPin1,  OUTPUT);
  pinMode(motorPin2,  OUTPUT);
  pinMode(enablePin, OUTPUT);
  // Enable the motor driver
  digitalWrite(enablePin, HIGH);
}
void loop() {
  // Rotate motor clockwise
  digitalWrite(motorPin1, HIGH);
  digitalWrite(motorPin2, LOW);
  delay(2000);
  // Stop motor
  digitalWrite(motorPin1, LOW);
  digitalWrite(motorPin2, LOW);
  delay(1000);
  // Rotate motor anti-clockwise
  digitalWrite(motorPin1, LOW);
  digitalWrite(motorPin2, HIGH);
  delay(2000);
  // Stop motor
  digitalWrite(motorPin1, LOW);
  digitalWrite(motorPin2, LOW);
  delay(1000);  }
```

Components Required

1. **Arduino UNO**
2. **DC Motor**
3. **L293D Motor Driver IC** (or L298N)
4. **External Power Supply (9V/12V Battery – optional for motor)**
5. **Jumper Wires**
6. **Breadboard**

The following components from the left panel:

- Arduino UNO
- DC Motor
- L293D Motor Driver
- Breadboard & Wires

Connect **L293D Motor Driver**:

- Pin 1 (Enable A) → Arduino Pin 11
- Pin 2 → Arduino Pin 9
- Pin 7 → Arduino Pin 10
- Pin 3 & 6 → Connect to DC Motor terminals
- Pin 4, 5, 12, 13 → GND
- Pin 8 → External 9V Vcc (for motor power)
- Pin 16 → +5V (Arduino Vcc)
- Pin 9 (GND of Arduino) → Common Ground with Motor Driver

5. Develop a program to deploy smart street light system using LDR sensor.

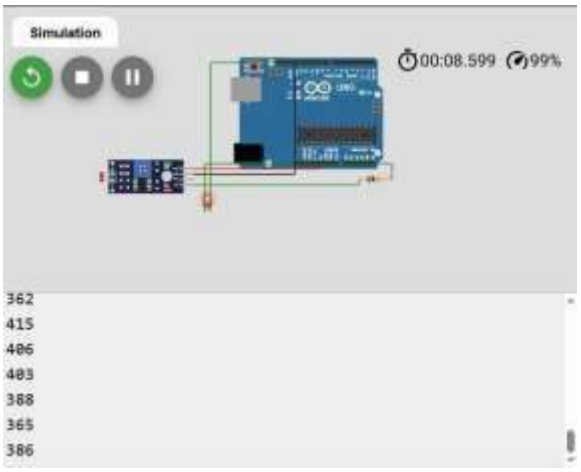
```
// Smart Street Light System using LDR
int LDRPin = A0;     // LDR connected to Analog pin A0
int LEDPin = 9;      // LED connected to Digital pin 9
int LDRValue = 0;    // Variable to store LDR value
int threshold = 500; // Threshold for darkness (adjustable)

void setup() {
  pinMode(LEDPin, OUTPUT);
  Serial.begin(9600); // For monitoring LDR values
}

void loop() {
  LDRValue = analogRead(LDRPin); // Read LDR value
  Serial.println(LDRValue);       // Print LDR value in Serial Monitor
  if (LDRValue < threshold) {
    digitalWrite(LEDPin, HIGH); // Turn ON light (dark condition)
  }
else
{
    digitalWrite(LEDPin, LOW);  // Turn OFF light (bright condition)
  }
  delay(200); // Small delay
}
```

Components Required

1. **Arduino Uno**
2. **LDR Sensor (Photoresistor)**
3. **10kΩ Resistor** (for LDR voltage divider)
4. **LED** (acts as street light)
5. **220Ω Resistor** (for LED current limiting)
6. **Breadboard & Jumper wires**

Circuit Connection

1. Connect **LDR** in series with **10kΩ resistor** to form a voltage divider.
   - One side of LDR → **5V**
   - Other side of LDR → **A0 (Arduino Analog Pin)** and one side of 10kΩ resistor.
   - Other side of 10kΩ resistor → **GND**.
2. **LED connection**:
   - Anode of LED → **Pin 9 (PWM Pin)** of Arduino through a **220Ω resistor**.
   - Cathode of LED → **GND**.



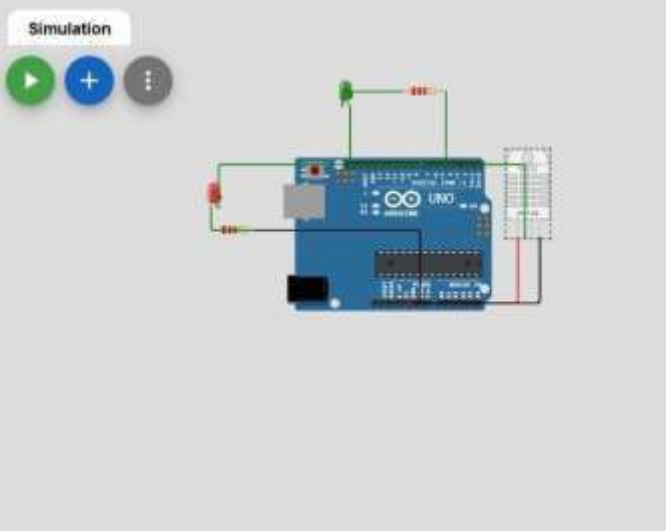
6. Develop a program to classify dry and wet waste with the Moisture sensor (DHT22).

```
#include "DHT.h"
#define DHTPIN 2        // DHT22 sensor connected to digital pin 2
#define DHTTYPE DHT22   // DHT22 (AM2302)
#define LED_DRY 3       // Green LED for Dry
#define LED_WET 4       // Red LED for Wet

DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  dht.begin();
  pinMode(LED_DRY, OUTPUT);
  pinMode(LED_WET, OUTPUT);
}
void loop() {
  float humidity = dht.readHumidity();
  if (isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.println("%");

  // Threshold: Assume >60% = Wet waste, <=60% = Dry waste
  if (humidity > 60) {
    digitalWrite(LED_WET, HIGH);
    digitalWrite(LED_DRY, LOW);
    Serial.println("Classification: Wet Waste");
```

```
  } else {
    digitalWrite(LED_DRY, HIGH);
    digitalWrite(LED_WET, LOW);
    Serial.println("Classification: Dry Waste");
  }
  delay(2000); // Read every 2 seconds
}
```

Components Required (in Wokwi)

- Arduino Uno (main controller)
- DHT22 Sensor (for humidity and temperature)
- 2 LEDs (Green for Dry, Red for Wet)
- 2 Resistors (220Ω) (for LEDs)
- Breadboard & Jumper wires



7. **Develop a program to read the pH value of a various substances like milk, lime and water.**

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
// Pin for pH sensor (analog)
const int phPin = A0; // Sensor connected to A0

// LCD display setup (if using LCD)
LiquidCrystal_I2C lcd(0x27, 16, 2);  // Change the I2C address if necessary

void setup() {
  Serial.begin(9600);  // For serial monitor
  lcd.begin(16, 2);     // Initialize the LCD
  lcd.print("pH Sensor"); // Display message on LCD
  delay(2000);          // Wait for 2 seconds
}

void loop() {
  // Read the analog value from the pH sensor
  int sensorValue = analogRead(phPin);
  float voltage = sensorValue * (5.0 / 1023.0);  // Convert to voltage (0-5V)

  // Assuming the sensor has a linear output for pH (check your sensor datasheet for calibration)
  float pH = 3.5 * voltage;  // You may need to calibrate this formula based on sensor specs

  // Print the pH value to the Serial Monitor
  Serial.print("pH Value: ");
  Serial.println(pH);
```

```
  // Display the pH value on LCD (if using LCD)
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("pH: ");
  lcd.print(pH);

  delay(1000);  // Wait for 1 second before taking the next reading
}
```

Components

1. **pH Sensor** (e.g., **Gravity: Analog pH Sensor / Meter V2**)
2. **Arduino Uno** or any compatible Arduino board
3. **Jumper wires** for connections
4. **LCD Display** (Optional, to display the pH value directly on the screen)
5. **Breadboard** (Optional for ease of connection)

Execution Steps in Wokwi Simulator

1. Setting up the Components

- **pH Sensor:**
  - Connect the **pH sensor's analog output** to the **A0 pin** on the Arduino. The sensor will output an analog value corresponding to the pH of the liquid.
  - Connect the **VCC** of the pH sensor to **5V** on the Arduino.
  - Connect the **GND** of the pH sensor to **GND** on the Arduino.
- **LCD Display** (Optional):
  - If you want to display the pH value on an LCD screen, you can use a 16x2 LCD, and connect the pins as follows:
    - **VSS** to **GND**
    - **VDD** to **5V**
    - **SDA** to **A4**
    - **SCL** to **A5**
    - **V0** for contrast adjustment (use a potentiometer for this).

8. Develop a program to detect the gas leakage in the surrounding environment.

```
// Pin Definitions
const int mq2Pin = A0;        // MQ-2 sensor analog pin (A0)
const int buzzerPin = 9;      // Buzzer connected to pin 9
const int ledPin = 13;        // LED connected to pin 13 (optional)
int gasThreshold = 400;       // Threshold value for gas detection (calibration needed)
void setup() {
  // Initialize serial communication for debugging
  Serial.begin(9600);
  // Set the buzzer and LED pins as outputs
  pinMode(buzzerPin, OUTPUT);
  pinMode(ledPin, OUTPUT);
  // Initial state of LED (off) and buzzer (silent)
  digitalWrite(buzzerPin, LOW);
  digitalWrite(ledPin, LOW);
}
void loop() {
  // Read the analog value from the MQ-2 sensor
  int gasValue = analogRead(mq2Pin);
  // Print the gas sensor value to the serial monitor for debugging
  Serial.print("Gas Sensor Value: ");
  Serial.println(gasValue);
  // Check if the gas value exceeds the threshold (indicating gas leakage)
  if (gasValue > gasThreshold) {
    // If gas is detected, activate the buzzer and turn on the LED
    digitalWrite(buzzerPin, HIGH); // Sound the buzzer
    digitalWrite(ledPin, HIGH);    // Turn on the LED
    Serial.println("Gas leakage detected!");
  } else {
    // If no gas detected, deactivate the buzzer and turn off the LED
```

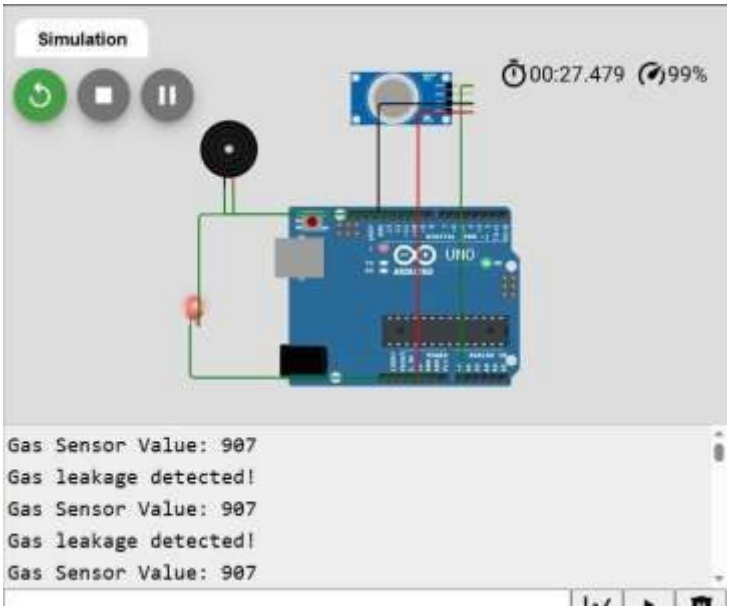
```
    digitalWrite(buzzerPin, LOW);
    digitalWrite(ledPin, LOW);
  }
  // Wait for 500 milliseconds before reading again
  delay(500);
}
```

Components

1. **MQ-2 Gas Sensor** (or MQ-7, depending on the gases you want to detect)
2. **Arduino Uno** (or any compatible Arduino board)
3. **Buzzer** (to sound an alarm when gas is detected)
4. **LED** (to visually indicate if gas is detected)
5. **Jumper wires**
6. **Breadboard** (optional for ease of connection)

Pin Connections

- **MQ-2 Gas Sensor:**
  - **VCC** to **5V** on Arduino.
  - **GND** to **GND** on Arduino.
  - **Analog Output (A0)** to **A0** on Arduino (used for detecting gas concentration).
- **Buzzer:**
  - **Positive (long leg)** to **pin 9** on Arduino.
  - **Negative (short leg)** to **GND** on Arduino.
- **LED** (Optional, for indication):
  - **Anode (long leg)** to **pin 13** on Arduino.
  - **Cathode (short leg)** to **GND** through a 220-ohm resistor.



9. Develop a program to demonstrate weather station readings using Arduino.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <Adafruit_BMP280_U.h>

// Pin Definitions
#define DHTPIN 2              // Pin connected to DHT sensor
#define DHTTYPE DHT11         // Type of DHT sensor (DHT11 or DHT22)
DHT dht(DHTPIN, DHTTYPE);       // Initialize DHT sensor
Adafruit_BMP280_Unified bmp;// Initialize BMP280 sensor
// LCD Setup (I2C Address 0x27)
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
  Serial.begin(9600);  // For serial monitor output
    // Initialize DHT sensor
  dht.begin();
    // Initialize BMP280 sensor
  if (!bmp.begin()) {
    Serial.println("BMP280 not detected. Check wiring.");
    while (1);
  }
  // Initialize LCD
  lcd.begin(16, 2);
  lcd.print("Weather Station");
  delay(2000); // Wait 2 seconds
    lcd.clear();
  lcd.print("Temp: ");
```

```
  lcd.setCursor(0, 1);
  lcd.print("Humidity: ");
}
void loop() {
  // Read temperature and humidity from DHT sensor
  float temperature = dht.readTemperature(); // in Celsius
  float humidity = dht.readHumidity();  // in %

  // Check if any readings failed and exit early
  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  // Read temperature and pressure from BMP280 sensor
  float pressure = bmp.readPressure() / 100.0F;  // in hPa
  // Print readings to Serial Monitor
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print(" °C  ");
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.print(" %  ");
  Serial.print("Pressure: ");
  Serial.print(pressure);
  Serial.println(" hPa");
  // Display readings on LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(temperature);
```

```
  lcd.print("C");
  lcd.setCursor(0, 1);
  lcd.print("Hum: ");
  lcd.print(humidity);
  lcd.print("%");
  delay(2000);  // Delay for 2 seconds before next read
}
```

Components

1. **DHT11/DHT22** - Temperature and Humidity Sensor
2. **BMP180/BMP280** - Atmospheric Pressure Sensor
3. **Arduino Uno** or any compatible Arduino board
4. **16x2 LCD Display** (for displaying data) or **Serial Monitor**
5. **Jumper wires**
6. **Breadboard** (optional for convenience)

Pin Connections

1. **DHT11/DHT22 Sensor:**
   - **VCC** to **5V** on Arduino.
   - **GND** to **GND** on Arduino.
   - **DATA** to **pin 2** on Arduino (you can change the pin if needed).
2. **BMP180/BMP280 Sensor:**
   - **VCC** to **3.3V** or **5V** on Arduino (check the sensor's datasheet for voltage requirements).
   - **GND** to **GND** on Arduino.
   - **SCL** to **A5** on Arduino (SCL pin).
   - **SDA** to **A4** on Arduino (SDA pin).
3. **LCD 16x2:**
   - **VSS** to **GND**
   - **VDD** to **5V**
   - **V0** (contrast) connected to a potentiometer (optional, for adjusting contrast).
   - **SDA** to **A4** on Arduino (if using I2C)
   - **SCL** to **A5** on Arduino (if using I2C)

Libraries Required

- **DHT Sensor Library** (for DHT11/DHT22)
- **Adafruit BMP280 Library** (for BMP180/BMP280)
- **LiquidCrystal_I2C Library** (for LCD display)

10. Develop a program to setup a UART protocol and pass a string through the protocol.

```
void setup() {
  // Initialize the Serial communication at 9600 baud rate
  Serial.begin(9600);
  // Wait for the serial connection to establish
  while (!Serial) {
    ; // Wait for serial port to connect (only needed for Leonardo and other boards with native USB)
  }
  // Send a welcome message through UART
  Serial.println("UART Communication Setup Complete!");
  Serial.println("Sending String: Hello, UART Protocol!");
}
void loop() {
  // Send a string over UART
  Serial.println("Hello, UART Protocol!");
  // Delay for 2 seconds before sending the string again
  delay(2000);
}
```

Components Required:

- **Arduino Uno** (or any compatible Arduino board)
- **USB cable** (to connect Arduino to the computer for serial communication)

Basic Concept :

- **TX Pin (Transmit)**: Sends data from Arduino to another device (typically, the computer).
- **RX Pin (Receive)**: Receives data sent from another device (like the computer).
- **Ground (GND)**: Common reference point for communication.

Pin Connections:

- **TX Pin (Arduino Pin 1)** is used for transmitting data to the computer.
- **RX Pin (Arduino Pin 0)** is used to receive data from the computer (not necessary for just sending a string).

11. Develop a water level depth detection system using Ultrasonic sensor.

```
#include <LiquidCrystal_I2C.h>
// Pin Definitions for HC-SR04 Ultrasonic Sensor
const int trigPin = 9;  // Trigger pin for ultrasonic sensor
const int echoPin = 10; // Echo pin for ultrasonic sensor
// LCD Setup (I2C Address 0x27, 16x2 LCD)
LiquidCrystal_I2C lcd(0x27, 16, 2);
// Constants
long duration;  // Duration for the ultrasonic pulse to return
int distance;   // Distance from the ultrasonic sensor to the water surface
int maxHeight = 200; // Maximum height of the water tank in cm (adjust based on your setup)

void setup() {
  // Start serial communication for debugging
  Serial.begin(9600);
  // Initialize the ultrasonic sensor pins
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  // Initialize the LCD
  lcd.begin(16, 2);
  lcd.print("Water Level Depth");
  delay(2000); // Wait for 2 seconds to display the message
  lcd.clear();
}
void loop() {
  // Send a pulse to the ultrasonic sensor
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2); // Wait for a brief moment
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10); // Send a 10-microsecond pulse
```

```
  digitalWrite(trigPin, LOW);

  // Measure the time for the pulse to return
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance in cm
  distance = duration / 58.2;  // Convert time to distance in cm (Speed of sound is 34300 cm/s)

  // Calculate the water level depth based on the maximum height
  int waterLevel = maxHeight - distance; // Water level = Max height - Distance from the sensor

  // Print the distance and water level to Serial Monitor
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.print(" cm  ");
  Serial.print("Water Level Depth: ");
  Serial.print(waterLevel);
  Serial.println(" cm");

  // Display the water level on the LCD
  lcd.setCursor(0, 0);
  lcd.print("Level: ");
  lcd.print(waterLevel);
  lcd.print(" cm");
  lcd.setCursor(0, 1);
  lcd.print("Distance: ");
  lcd.print(distance);
  lcd.print(" cm");

  // Delay for 1 second before the next reading
```

```
  delay(1000);
}
```

Components Required:

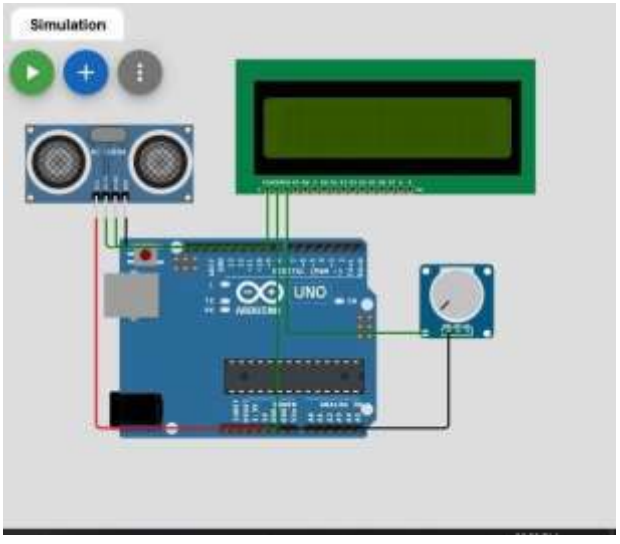
1. **Arduino Uno** (or any compatible Arduino board)
2. **HC-SR04 Ultrasonic Sensor** (to measure distance)
3. **16x2 LCD Display** (to display water depth) or **Serial Monitor**
4. **Jumper Wires**
5. **Breadboard** (optional)

Pin Connections for HC-SR04 Sensor:

- **VCC → 5V** on Arduino
- **GND → GND** on Arduino
- **Trig → Pin 9** on Arduino
- **Echo → Pin 10** on Arduino

16x2 LCD (Optional for Display):

- **VSS → GND**
- **VDD → 5V**
- **V0** → Potentiometer (for contrast adjustment)
- **SDA → A4** on Arduino
- **SCL → A5** on Arduino



12. Develop a program to simulate interfacing with the keypad module to record the keystrokes.

```
#include <Keypad.h>
// Define the keymap for a 4x4 Keypad
const byte ROW_NUM    = 4; // Four rows
const byte COLUMN_NUM = 4;  // Four columns
// Define the pinout for the keypad
char keys[ROW_NUM][COLUMN_NUM] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
// Define the pin connections for the keypad
byte pin_rows[ROW_NUM] = {2, 3, 4, 5};      // Connect rows to digital pins 2 to 5
byte pin_column[COLUMN_NUM] = {6, 7, 8, 9}; // Connect columns to digital pins 6 to 9
// Create an instance of the Keypad library
Keypad keypad = Keypad(makeKeymap(keys), pin_rows, pin_column, ROW_NUM,
COLUMN_NUM);
void setup() {
  // Start the Serial communication
  Serial.begin(9600);
  Serial.println("Keypad Interface - Recording Keystrokes");
}
void loop() {
  // Check if a key is pressed
  char key = keypad.getKey();
  // If a key is pressed, print it to the Serial Monitor
  if (key) {
    Serial.print("Key Pressed: ");
    Serial.println(key);      }  }
```

Components Required:

1. **Arduino Uno** (or any compatible Arduino board)
2. **4x4 Keypad Module**
3. **Jumper Wires**
4. **Breadboard** (optional for convenience)

Pin Connections:

The **4x4 Keypad** consists of 8 pins: 4 for the rows and 4 for the columns. You'll connect these pins to any digital pins on your Arduino.

For example:

- **Row Pins**: R1, R2, R3, R4 → **Pins 2, 3, 4, 5** on Arduino
- **Column Pins**: C1, C2, C3, C4 → **Pins 6, 7, 8, 9** on Arduino

Library to Use:

- **Keypad Library**: This library helps manage the keypad matrix and provides an easy-to-use interface to read key presses.

To install the **Keypad library**:

1. Go to **Sketch → Include Library → Manage Libraries**.
2. Search for **Keypad** and install the library by **Mark Stanley** and **Alexey Dykhno**.

Keypad Layout:

The 4x4 keypad has 16 keys, which can be represented as:

1	2	3	A
4	5	6	B
7	8	9	C
*	0	#	D

Code to Interface with Keypad and Record Keystrokes