

ATME COLLEGE OF ENGINEERING

13th Km, Mysore Kanakapura-Bangalore-Road, Mysore-570028



ATME

College of Engineering

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

2026-2027

INTERNET OF THINGS

(BCS701)

Laboratory Manual - Integrated Course

(Effective from the academic year 2022 -2023)

SEMESTER– VII

Composed By

Verified By

Approved by

Mr. Vinay Kumar M N
Programmer

Mrs. Roopa B, Mr. Gadi Divyasree
Faculties Coordinators

Dr. Puttegowda D
HOD, CSE

INSTITUTIONAL VISION AND MISSION

OBJECTIVES

- To provide quality education and groom top-notch professionals, entrepreneurs and leaders for different fields of engineering, technology and management.
- To open a Training-R & D-Design-Consultancy cell in each department, gradually introduce doctoral and postdoctoral programs, encourage basic & applied research in areas of social relevance, and develop the institute as a Centre of excellence.
- To develop academic, professional and financial alliances with the industry as well as the academia at national and transnational levels.
- To cultivate strong community relationships and involve the students and the staff in local community service.
- To constantly enhance the value of the educational inputs with the participation of student's faculty, parents and industry.

VISION

- Development of academically excellent, culturally vibrant, socially responsible and globally competent human resources.

MISSION

- To keep pace with advancements in knowledge and make the students competitive and capable at the global level.
- To create an environment for the students to acquire the right physical, intellectual, emotional and moral foundations and shine as torch bearers of tomorrow society.
- To strive to attain ever-higher benchmarks of educational excellence.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Vision of the Department

- To develop highly talented individuals in Computer Science and Engineering to deal with real world challenges in industry, education, research and society.

Mission of the Department

- To inculcate professional behavior, strong ethical values, innovative research capabilities and leadership abilities in the young minds & to provide a teaching environment that emphasizes depth, originality and critical thinking.
- Motivate students to put their thoughts and ideas adoptable by industry or to pursue higher studies leading to research.

Program outcomes (POs)

Engineering Graduates will be able to:

PO1. Engineering Knowledge: Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems.

PO2. Problem Analysis: Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4).

PO3. Design/Development of Solutions: Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5).

PO4. Conduct Investigations of Complex Problems: Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8).

PO5. Engineering Tool Usage: Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6).

PO6. The Engineer and The World: Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).

PO7. Ethics: Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)

PO8. Individual and Collaborative Team work: Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.

PO9. Communication: Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences.

PO10. Project Management and Finance: Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.

PO11. Life-Long Learning: Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change. (WK8).

Program Specific Outcomes (PSOs)

1. **PSO1:** Ability to apply skills in the field of algorithms, database design, web design, cloud computing and data analytics.
2. **PSO2:** Apply knowledge in the field of computer networks for building network and internet-based applications.

Program Educational Objectives (PEOS):

1. Empower students with a strong basis in the mathematical, scientific and engineering fundamentals to solve computational problems and to prepare them for employment, higher learning and R&D.
2. Gain technical knowledge, skills and awareness of current technologies of computer science engineering and to develop an ability to design and provide novel engineering solutions for software/hardware problems through entrepreneurial skills.
3. Exposure to emerging technologies and work in teams on interdisciplinary projects with effective communication skills and leadership qualities.
4. Ability to function ethically and responsibly in a rapidly changing environment by applying innovative ideas in the latest technology, to become effective professionals in Computer Science to bear a life-long career in related areas.

INSTRUCTIONS TO STUDENTS

- Students should leave their foot wares outside in the shoe rack.
- Students should keep their bags in the bag rack.
- Students must take care of their valuable things.
- Students must bring Observation book, record and manual along with pen, pencil, and eraser Etc., no borrowing from others.
- Students must handle the personal computers and other components carefully, as they are expensive.
- Before entering to lab, must prepare for Viva for which they are going to conduct experiment.
- After the completion of the experiment and before leaving the lab, check whether PCs have shutdown, switch off the monitors, fans, light and keep the chairs properly.

RULES FOR MAINTAINING LABORATORY RECORD

1. Put your Name, USN and Subject Name & Subject Code on the outside front cover of the record.
Put that same information on the first page inside.
2. Update Table of Contents every time you start each new experiment/programs/Sketch/Code.
3. Always use pen and write neatly and clearly.
4. Start each new topic (experiment, notes, calculations, waveforms etc.) on a right-side page.
5. Obvious care should be taken to make it readable, even if you have bad handwriting.
6. Date to be written every page on the top right-side corner.
7. On each right-side page
Title of experiment Aim/Objectives
Components Required if any
Theory/Programs
Procedure described clearly in steps
Result
8. On each left side Page
Pin Diagrams
Circuit Diagrams/Problems
Truth Tables
Graphs/Waveforms

Use label and captions for figures and tables.
9. Use the correct (i.e., IEEE standard) abbreviations for all units.
10. Attach printouts and plots of data as needed. Stick printouts (A4 Size) on the right side of the lab record.
11. Observation and record should be signed before the next lab session.
12. Strictly maintain the instructions given by the Faculty/ Lab Instructor.

Internet of Things		Semester	VII
Course Code	BCS701	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	3:0:2:0	SEE Marks	50
Total Hours of Pedagogy	40 hours Theory + 8-10 Lab slots	Total Marks	100
Credits	04	Exam Hours	03
Examination nature (SEE)	Theory/practical		

Sl.No.	Experiments
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.

Course objectives:

- Understand about the fundamentals of Internet of Things and its building blocks along with their characteristics.
- Understand the recent application domains of IoT in everyday life.
- Understand the protocols and standards designed for IoT and the current research on it.
- Understand the other associated technologies like cloud and fog computing in the domain of IoT.
- Improve their knowledge about the various cutting-edge technologies in the field IoT and machine learning applications.
- Gain insights about the current trends of machine learning and AI techniques used in IoT to orient towards the present industrial scenario.

Course outcome (Course Skill Set):

At the end of the course, the student will be able to:

- Explain the evolution of IoT, IoT networking components, and addressing strategies in IoT.
- Analyze various sensing devices and actuator types.
- Demonstrate the processing in IoT.
- Apply different connectivity technologies.
- Elaborate the need for Data Analytics and Security in IoT.

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together.

CIE for the theory component of the IPCC (maximum marks 50)

- IPCC means practical portion integrated with the theory of the course.
- CIE marks for the theory component are 25 marks and that for the practical component is 25 marks.
- 25 marks for the theory component are split into 15 marks for two Internal Assessment Tests (Two Tests, each of 15 Marks with 01-hour duration, are to be conducted) and 10 marks for other assessment methods mentioned in 22OB4.2. The first test shall be held at the end of 40-50% coverage of the syllabus and the second test after covering 85-90% of the syllabus.
- Scaled-down marks of the sum of two tests and other assessment methods will be CIE marks for the theory component of IPCC (that is for 25 marks).
- The student has to secure 40% of 25 marks to qualify in the CIE of the theory component of IPCC.

CIE for the practical component of the IPCC

- 15 marks for the conduction of the experiment and preparation of laboratory record, and 10 marks for the test to be conducted after the completion of all the laboratory sessions.
- On completion of every experiment/program in the laboratory, the students shall be evaluated including viva-voce and marks shall be awarded on the same day.
- The CIE marks awarded in the case of the Practical component shall be based on the continuous evaluation of the laboratory report. Each experiment report can be evaluated for 10 marks. Marks of all experiments' write-ups are added and scaled down to 15 marks.
- The laboratory test (duration 02/03 hours) after completion of all the experiments shall be conducted for 50 marks and scaled down to 10 marks.
- Scaled-down marks of write-up evaluations and tests added will be CIE marks for the laboratory component of IPCC for 25 marks.
- The student has to secure 40% of 25 marks to qualify in the CIE of the practical component of the IPCC.

Sl No.	Experiment Name List	Page No.
1.	INTRODUCTION	1 - 2
2.	LABORATORY SESSION 1 – Study of Arduino Board & Sensors, Sketch	3 - 7
Sl No.	EXPERIMENTS	Page No.
1.	Develop a program to blink 5 LEDs back and forth.	8 - 9
2.	Develop a program to interface a relay with Arduino board.	10 - 11
3.	Develop a program to deploy an intrusion detection system using Ultrasonic and sound sensors.	12 - 13
4.	Develop a program to control a DC motor with Arduino board.	14 - 15
5.	Develop a program to deploy smart street light system using LDR sensor.	16 - 17
6.	Develop a program to classify dry and wet waste with the Moisture sensor (DHT22).	18 - 19
7.	Develop a program to read the pH value of a various substances like milk, lime and water.	20 - 22
8.	Develop a program to detect the gas leakage in the surrounding environment.	23 - 24
9.	Develop a program to demonstrate weather station readings using Arduino.	25 - 31
10.	Develop a program to setup a UART protocol and pass a string through the protocol.	32 - 33
11.	Develop a water level depth detection system using Ultrasonic sensor.	34 - 35
12.	Develop a program to simulate interfacing with the keypad module to record the keystrokes.	36 - 38

INTRODUCTION

INTERNET OF THINGS

The Internet of Things (IoT) describes the network of physical objects “things” that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet. These devices range from ordinary household objects to sophisticated industrial tools.

How IoT works

IoT connects the physical and digital worlds. The process generally includes:

1. **Data Collection:** Devices with sensors gather data from their environment.
2. **Data Transmission:** Collected data is sent to a system or cloud platform for storage and processing using various connectivity technologies like Wi-Fi, Bluetooth, or cellular networks.
3. **Data Analysis and Processing:** Algorithms and machine learning analyse the data to find patterns and identify issues.
4. **Action and Response:** Based on analysis, devices can communicate and perform actions, such as adjusting a thermostat or triggering alerts.

Essential components of IoT systems include:

- **Sensors and Actuators:** Sensors collect data, and actuators enable devices to respond.
- **Connectivity Technologies:** Various wired and wireless options connect devices.
- **Data Storage and Processing:** Cloud platforms are used for storing and processing the large amounts of data.
- **IoT Applications and Services:** These interfaces allow users to manage connected devices.

Benefits of IoT

IoT offers benefits across various sectors:

- **Increased Efficiency and Productivity:** Automation leads to greater efficiency.
- **Enhanced Operational Visibility:** Real-time data provides insights for better decision-making.
- **Cost Savings:** Automation and optimized resource use reduce costs.
- **Improved Safety:** Monitoring and automated responses enhance safety.
- **Enhanced Customer Experience:** Personalized services and proactive support improve satisfaction.
- **Data-Driven Decision Making:** Analysing data provides insights for strategic decisions.

Challenges and Future of IoT

IoT faces challenges like security, privacy, interoperability, scalability, regulatory issues, and high implementation costs. However, the future looks promising with emerging trends.

Integration with 5G, AI, ML, edge computing, and blockchain are driving forces. These advancements enable faster data transfer, more intelligent decision-making, and enhanced security.

IoT is changing how people live and work, leading to a more connected and efficient future.

Technologies have made IoT possible?

- Access to low-cost, low-power sensor technology.
- Connectivity.
- Cloud computing platforms.
- Machine learning and analytics.
- Artificial intelligence (AI).

ARCHITECHTURE OF IOT

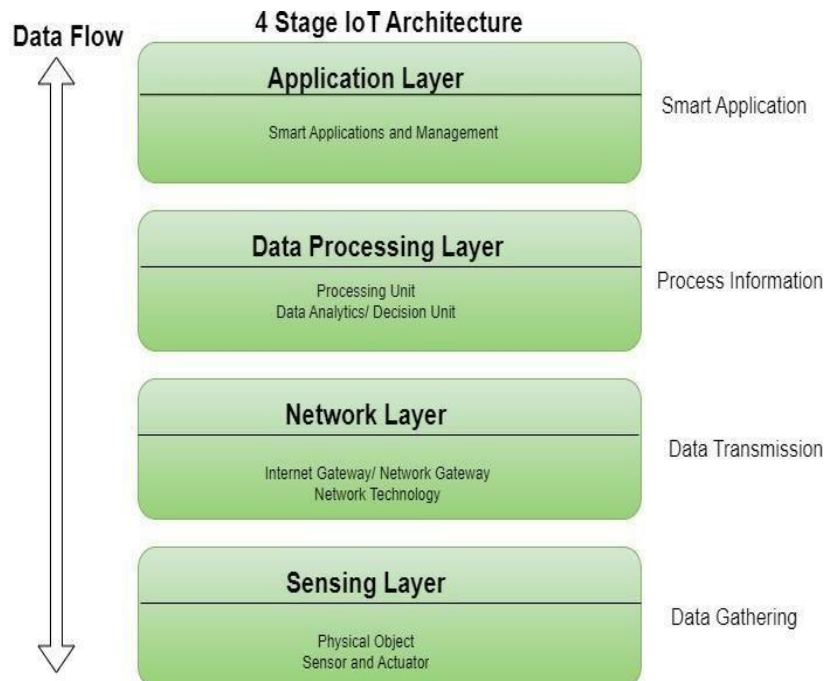


Diagram 1

Arduino Board

Arduino Board

An Arduino board is a microcontroller-based kit designed for building interactive projects. It's a popular open-source platform, meaning its hardware and software are freely available for anyone to use, modify, and distribute. Arduino boards are used to read input from sensors, process that data, and then control output devices like LEDs, motors, and more.

Key Features:

Microcontroller: At the heart of the board is a microcontroller (like the Atmel AVR on the Arduino Uno) that executes the code you write.

Input/Output (I/O) Pins: These pins allow you to connect sensors, buttons, LEDs, and other components to the board.

Memory:

Arduino boards have different types of memory:

- **Flash Memory:** Stores the program code.
- **SRAM:** Stores temporary data while the program is running.
- **EEPROM:** Stores data that needs to be saved even when the board is powered off.

Programming: Arduino boards are programmed using the Arduino IDE (Integrated Development Environment), which is based on the Wiring language (a simplified version of C++).

Common Arduino Boards:

- **Arduino Uno:** A popular and widely used board, often considered a good starting point.
- **Arduino Mega:** A board with more I/O pins and memory, suitable for more complex projects.
- **Arduino Nano:** A smaller, compact board.

How it Works:

1. **Code Development:** You write your code in the Arduino IDE, specifying how the board should interact with the connected components.
2. **Code Upload:** You upload the code to the Arduino board via a USB connection.
3. **Input Processing:** The microcontroller reads data from the input pins (e.g., from a sensor).
4. **Logic Execution:** Based on the code, the microcontroller performs calculations and makes decisions.
5. **Output Control:** The microcontroller sends signals to the output pins, controlling devices like LEDs, motors, or other actuators.

The Arduino uno3 board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, Power Pins Include 5V, 3.3V, GND, Vin, and IOREF, Other Pins Include Reset, AREF, TX, and RX pins are also present. The digital pins can be used for both input and output, with some also offering PWM functionality (pins 3, 5, 6, 9, 10, and 11) and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable.

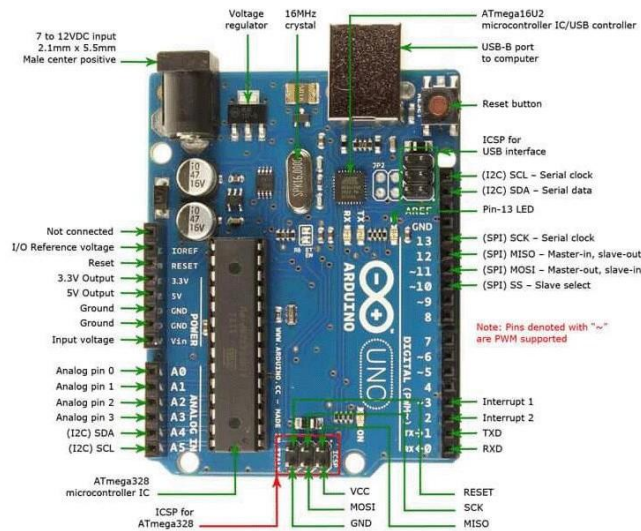
Analog Pins

Pin	Function	Type	Description
1	NC	NC	Not connected
2	IOREF	IOREF	Reference for digital logic V - connected to 5V
3	Reset	Reset	Reset
4	+3V3	Power	+3V3 Power Rail
5	+5V	Power	+5V Power Rail
6	GND	Power	Ground
7	GND	Power	Ground
8	VIN	Power	Voltage Input
9	A0	Analog/GPIO	Analog input 0 /GPIO
10	A1	Analog/GPIO	Analog input 1 /GPIO
11	A2	Analog/GPIO	Analog input 2 /GPIO
12	A3	Analog/GPIO	Analog input 3 /GPIO
13	A4/SDA	Analog input/I2C	Analog input 4/I2C Data line
14	A5/SCL	Analog input/I2C	Analog input 5/I2C Clock line

Digital Pins

Pin	Function	Type	Description
1	D0	Digital/GPIO	Digital pin 0/GPIO
2	D1	Digital/GPIO	Digital pin 1/GPIO
3	D2	Digital/GPIO	Digital pin 2/GPIO
4	D3	Digital/GPIO	Digital pin 3/GPIO
5	D4	Digital/GPIO	Digital pin 4/GPIO
6	D5	Digital/GPIO	Digital pin 5/GPIO
7	D6	Digital/GPIO	Digital pin 6/GPIO
8	D7	Digital/GPIO	Digital pin 7/GPIO
9	D8	Digital/GPIO	Digital pin 8/GPIO
10	D9	Digital/GPIO	Digital pin 9/GPIO
11	SS	Digital	SPI Chip Select
12	MOSI	Digital	SPI1 Main Out Secondary In
13	MISO	Digital	SPI Main In Secondary Out
14	SCK	Digital	SPI serial clock output
15	GND	Power	Ground
16	AREF	Digital	Analog reference voltage
17	A4/SD4	Digital	Analog input 4/I2C Data line (duplicated)
18	A5/SD5	Digital	Analog input 5/I2C Clock line (duplicated)

Arduino UNO Board



SENSORS: IoT sensors are devices that detect and measure physical parameters like temperature, humidity, motion, light, and more, then transmit this data over a network to a central system for analysis and action. These sensors are crucial for enabling smart applications across various industries.

What they are:

- **Data Acquisition:** IoT sensors act as the eyes and ears of IoT systems, continuously monitoring and gathering data from the physical world.
- **Connectivity:** They are typically connected to the internet (or other networks) via technologies like Wi-Fi, Bluetooth, or cellular, allowing them to transmit data to a central server or the cloud for processing.
- **Smart Capabilities:** Some sensors are "smart" or intelligent, incorporating processing capabilities and even the ability to make decisions based on the data they collect.

Key Types of IoT Sensors:

- **Temperature Sensors:** Monitor ambient temperature, equipment temperature, or specific process temperatures.
- **Humidity Sensors:** Measure the amount of moisture in the air or a specific environment.
- **Proximity/Motion Sensors:** Detect the presence or movement of objects or people.
- **Pressure Sensors:** Measure force applied to a surface or within a system.
- **Gas Sensors:** Detect the presence and concentration of various gases (e.g., carbon monoxide, methane, etc.).
- **Light Sensors:** Measure the intensity of light, often used in automated lighting systems.
- **Water Quality Sensors:** Monitor parameters like pH, turbidity, and flow in water systems.

- **Vibration Sensors:** Detect vibrations in machinery or structures, useful for predictive maintenance.
- **Image Sensors:** Capture visual data, often used in security systems or for object recognition.
- **Chemical Sensors:** Detect specific chemical compounds, often used in environmental monitoring.
- **Level Sensors:** Measure the level of liquids or solids in a container.
- **Smoke Sensors:** Detect the presence of smoke, often used in fire detection systems.
- **Ultrasonic Sensor:** Is an instrument that measures the distance to an object using ultrasonic sound waves.
- **Sound Sensor:** A sound sensor detects sound waves and converts them into an electrical signal, enabling devices to respond to audio input, much like a human ear.
- **Accelerometer/Gyroscope Sensors:** Measure acceleration and rotational motion, used in navigation and movement tracking. Etc....



Figure: 1

Applications:

- **Smart Homes:** Control lighting, temperature, security systems, and appliances.
- **Healthcare:** Monitor patient vital signs, track medication adherence, and enable remote patient monitoring.
- **Agriculture:** Optimize irrigation, monitor crop health, and manage livestock.
- **Manufacturing:** Improve production efficiency, monitor equipment health, and optimize supply chains.
- **Environmental Monitoring:** Track air and water quality, detect pollution, and manage resources.
- **Transportation:** Improve traffic flow, manage logistics, and track vehicle location.
- **Retail:** Enhance customer experience, manage inventory, and optimize store layouts.

The Arduino software is open-source. The source code for the Java environment is released under the General Public License (GPL) and the C/C++ microcontroller libraries are under the Lesser General Public License (LGPL).

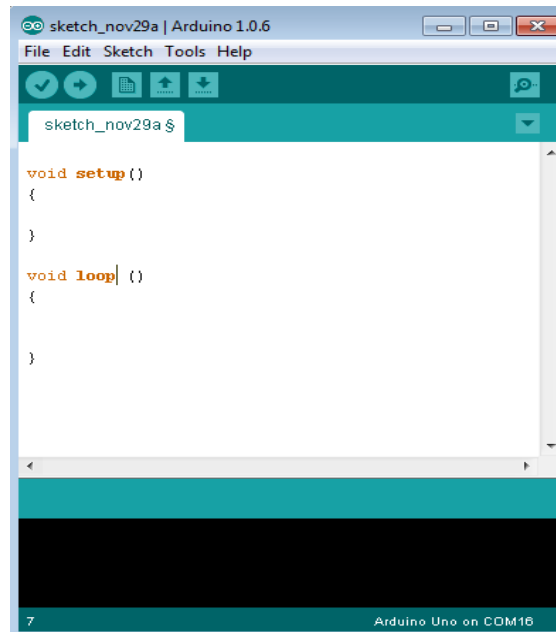
Sketch - The first new terminology is the Arduino program called **Sketch**.

Structure

Arduino programs can be divided in three main parts: **Structure, Values (variables and constants), and Functions**. In this tutorial, we will learn about the Arduino software program, step by step, and how we can write the program without any syntax or compilation error.

Let us start with the **Structure**. Software structure consists of two main functions –

- Setup() function
- Loop() function



```
Void setup () {  
  
}
```

- PURPOSE – The **setup()** function is called when a sketch starts. Use it to initialize the variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board.

```
Void setup () {  
  
}
```

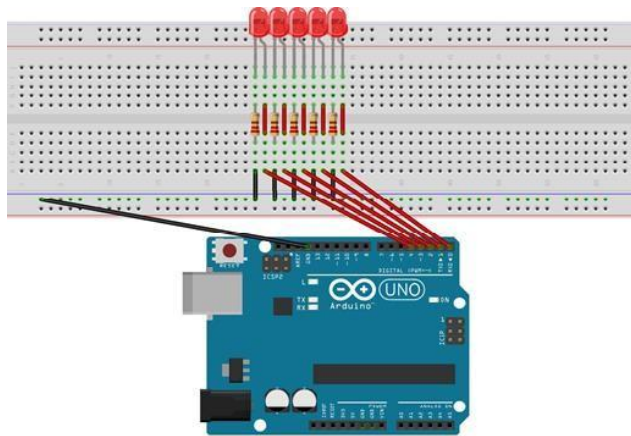
- PURPOSE – After creating a **setup()** function, which initializes and sets the initial values, the **loop()** function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

Experiment 1

Aim: Develop a program to blink 5 LEDs back and forth.

Components Required: Arduino UNO Board with USB cable, 220 Ω Resistors, Jumper Wires, Bread Board, LEDs

Schematic:



Sketch:

```
int led1 = 0;

int led2 = 1;

int led3 = 2;

int led4 = 3;

int led5 = 4;

// the setup routine runs once when you press reset:

void setup() {

  // initialize the digital pin as an output.

  pinMode(led1, OUTPUT);

  pinMode(led2, OUTPUT);

  pinMode(led3, OUTPUT);

  pinMode(led4, OUTPUT);

  pinMode(led5, OUTPUT);
```

```
}  
  
// the loop routine runs over and over again forever:  
  
void loop() {  
  
    digitalWrite(led1, HIGH); // turn the LED on (HIGH is the voltage level)  
  
    delay(80);                // wait for 1/2 a second  
  
    digitalWrite(led1, LOW);  // turn the LED off by making the voltage LOW  
  
    digitalWrite(led2, HIGH); // turn the LED on (HIGH is the voltage level)  
  
    delay(80);                // wait for 1/2 a second  
  
    digitalWrite(led2, LOW);  // turn the LED off by making the voltage LOW  
  
    digitalWrite(led3, HIGH); // turn the LED on (HIGH is the voltage level)  
  
    delay(80);                // wait for 1/2 a second  
  
    digitalWrite(led3, LOW);  // turn the LED off by making the voltage LOW  
  
    digitalWrite(led4, HIGH); // turn the LED on (HIGH is the voltage level)  
  
    delay(80);                // wait for 1/2 a second  
  
    digitalWrite(led4, LOW);  // turn the LED off by making the voltage LOW  
  
    digitalWrite(led5, HIGH); // turn the LED on (HIGH is the voltage level)  
  
    delay(80);                // wait for 1/2 a second  
  
    digitalWrite(led5, LOW);  // turn the LED off by making the voltage LOW  
  
    delay(500);               // wait for a second  
  
}
```

Output & Result: Program to blink 5 LEDs back and forth is developed, executed and the output is verified successfully.

Experiment 2

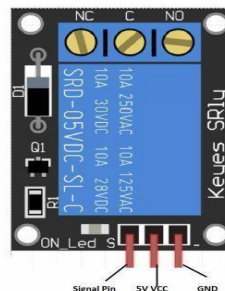
Aim: Develop a program to interface a relay with Arduino board.

Components Required: Arduino UNO Board with USB cable, 5V Relay Module, 220Ω Resistors, Bread Board, Jumper Wires, LED

Theory: A relay is generally an electrically operated switch. The principle used by the relays is an electromagnet to mechanically operate the switch. So basically, it operates a switch using an electromagnet which needs only less power like 5V, 12V or 24V. Different kinds of relays are available in the market like SPDT, DPDT, SPST, 5V, 12V, 24V and with various high current/voltage driving capacity.

Pinouts: Let's dive deep into the 5V relay module which we are using. This module has total 6 pins out of which 3 are used for controlling the relay (low voltage side). While other 3 are the output of the relay (usually high voltage side), where we will connect the device which we want to control.

Relay:



As shown in the above image:

- **NO: Normally Open** – This is the normally open terminal of the relay, means if we don't energize the relay there won't be any contact with Common terminal. But it will establish electrical contact with Common terminal once the relay is energized.
- **C: Common terminal**
- **NC: Normally Closed** – This is the normally closed terminal of the relay; means it will have electrical contact with common terminal whenever the relay is not energized. And there won't be electrical contact when the relay is energized.
- **GND: Ground Pin**
- **Signal:** Actuation signal to control the relay.
- **5V VCC:** Operating voltage for the relay.

Working: The principle behind the relay is electromagnetism, a switch operated by an electromagnet. So, a low voltage is enough for an electromagnet to get activated. This small voltage will be given to the relay by Arduino Uno or using some intermediate driver if required.

NO, NC and COM Terminals: Usually SPDT (Single Pole Double Throw) relays have 3 output terminals, these are the 3 terminals of internal SPDT electromagnetic switch.

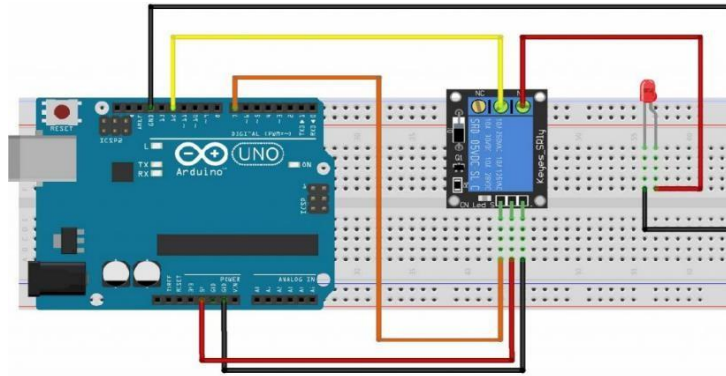
Common (COM): This is the commonly terminal. This terminal will be connected to either of other 2

terminals (NO or NC) based on the state of relay.

NormallyOpen(NO): As the name indicates this is normally open terminal, ie. if the relay is not energized (not ON), this pin will be open. We can say that the switch is OFF by default and when the relay is energized it will become ON.

NormallyClosed(NC): As the name indicates it is normally closed terminal, ie. if the relay is not energized (not ON), this pin will be closed. We can say that the switch is ON by default and when the relay is energized it will become OFF.

Schematic:



Sketch:

```
int relay_pin = 7;
int led_pin = 12;

void setup()
{
  pinMode(relay_pin,OUTPUT);
  pinMode(led_pin,OUTPUT);
  digitalWrite(led_pin,HIGH);
}

void loop()
{
  digitalWrite(relay_pin,HIGH);
  delay(2000);
  digitalWrite(relay_pin,LOW);
  delay(2000);
}
```

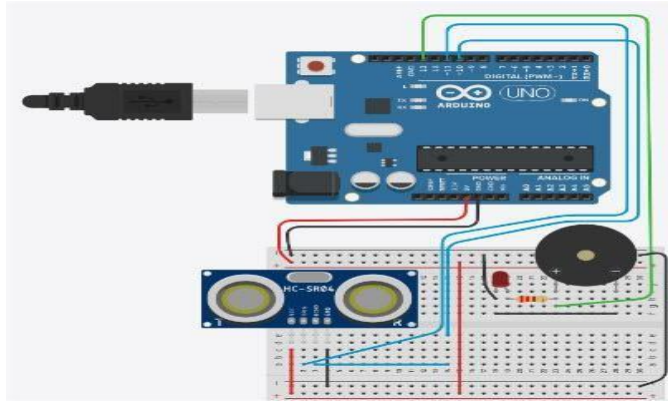
Output & Result: Program to interface a relay with Arduino board is developed, executed and the output is verified successfully.

Experiment 3

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

Components Required: Arduino UNO Board with USB cable, 220Ω Resistor, Jumper Wires, Bread Board, LED, Ultrasonic Sensor and Sound Sensor (Buzzer).

Schematic:



Sketch:

```
#define TRIG_PIN 11

#define ECHO_PIN 10

int LEDpin = 13;

SR04 sr04 = SR04(ECHO_PIN,TRIG_PIN);

long a;

void setup() {

  pinMode(LEDpin, OUTPUT);

  Serial.begin(9600);

  delay(1000);

}

void loop() {

  a=sr04.Distance();

  Serial.print(a);

  Serial.println("cm");
```

```
delay(1000);  
  
if(a < 200) {  
    digitalWrite(LEDpin, HIGH);  
}  
  
else{  
    digitalWrite(LEDpin, LOW);  
}  
}
```

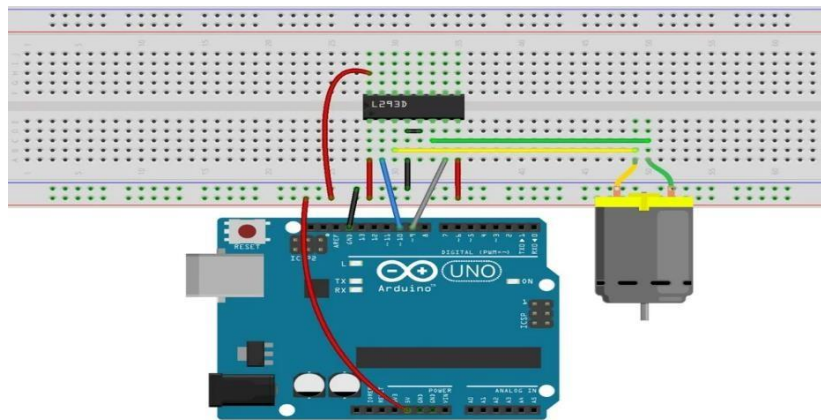
Output & Result: Program to deploy an intrusion detection system using Ultrasonic and sound sensors is developed, executed and the output is verified successfully.

Experiment 4

Aim: Develop a program to control a DC motor with Arduino board.

Components Required: Arduino UNO Board with USB cable, L298N Motor Driver, DC Motor, Bread Board, Jumper Wires.

Schematic:



Sketch:

```
const int motorIn1= 9; //attach to one of the pins of the motor

const int motorIn2= 10; //attach to another pin of the motor

/*****/

void setup()
{
    pinMode(motorIn1,OUTPUT); //initialize the motorIn1 pin as output
    pinMode(motorIn2,OUTPUT); //initialize the motorIn2 pin as output
}

/*****/

void loop()
{
    clockwise(200); //rotate clockwise

    delay(1000); //wait for a second
```

```
counterclockwise(200); //rotate counterclockwise

delay(1000); //wait for a second

}

/*****

//The function to drive motor rotate clockwise

void clockwise(int Speed)

{

    analogWrite(motorIn1,Speed); //set the speed of motor

    analogWrite(motorIn2,0); //stop the motorIn2 pin of motor

}

//The function to drive motor rotate counterclockwise

void

counterclockwise(int Speed)

{

    analogWrite(motorIn1,0); //stop the motorIn1 pin of motor

    analogWrite(motorIn2, Speed); //set the speed of motor

}
```

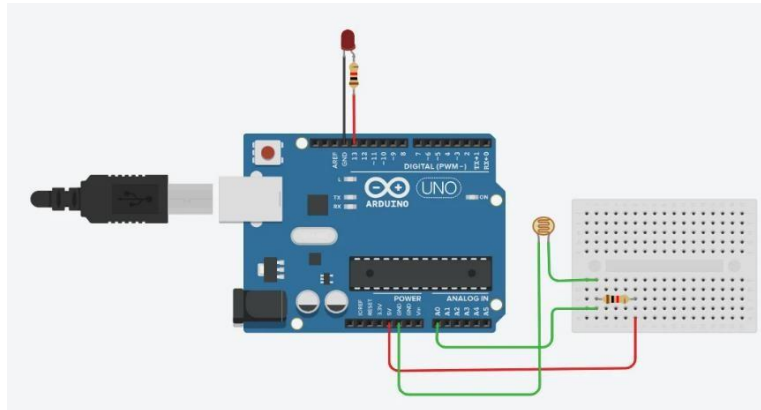
Output & Result: Program to control a DC motor with Arduino board is developed, executed and the output is verified successfully.

Experiment 5

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

Components Required: Arduino UNO Board with USB cable, LDR, LED, 220 Ω Resistor, Bread Board, Jumper Wires.

Schematic:



Sketch:

```
void setup()
{
  pinMode(13,OUTPUT);
  pinMode(A0,INPUT);
  Serial.begin(9600);
}

void loop()
{
  int signal;

  signal=analogRead(A0);

  Serial.println(signal);

  if(signal<400)
  {
```

```
    Serial.println("LED Light On");  
  
    digitalWrite(13,HIGH);  
  
    }  
  
    else  
  
    {  
  
    Serial.println("LED Light OFF");  
  
    digitalWrite(13,LOW);  
  
    }  
  
    delay(10);  
  
    }
```

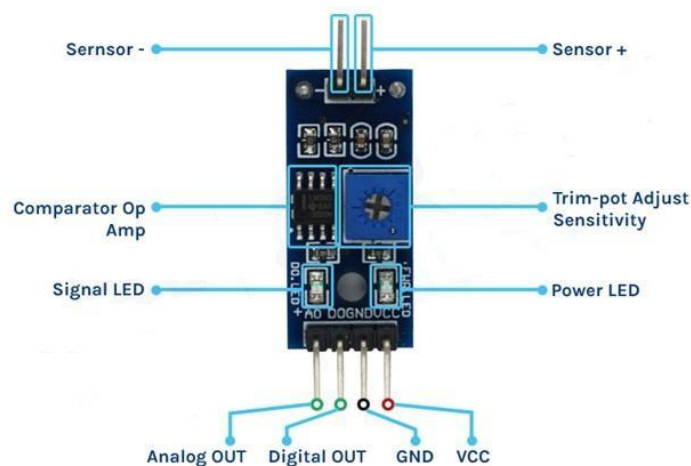
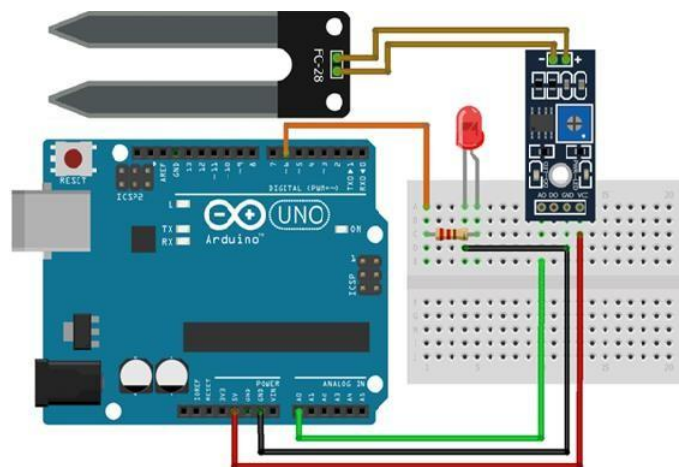
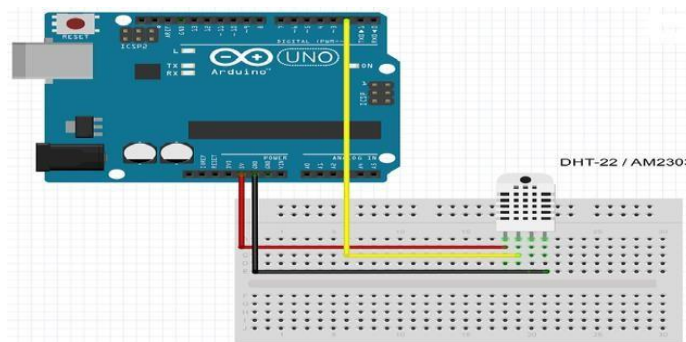
Output & Result: Program to deploy smart street light system using LDR sensor is developed, executed and the output is verified successfully.

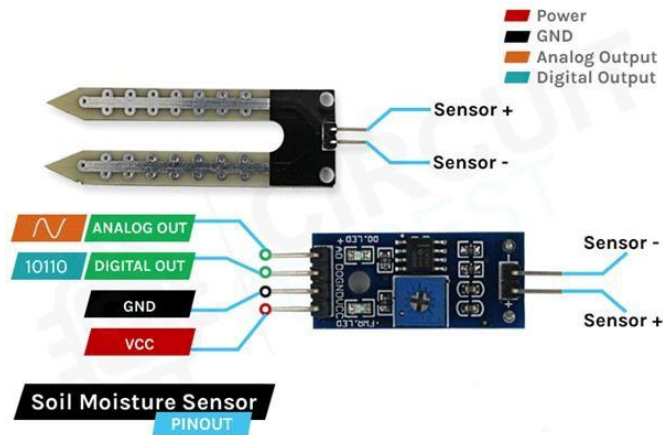
Experiment 6

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

Components Required: Arduino UNO Board with USB cable, Moisture Sensor, Bread Board, 16X2 LCD with I2C Module (if needed) Jumper Wires.

Schematic:



**Sketch:**

```
// Sensor pins pin D6 LED output, pin A0 analog Input

#define ledPin 6

#define sensorPin A0

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

  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);
}

void loop() {
  Serial.print("Analog output: ");
  Serial.println(readSensor());

  delay(500);
}

int readSensor() {
  int sensorValue = analogRead(sensorPin); // Read the analog value from sensor

  int outputValue = map(sensorValue, 0, 1023, 255, 0); // map the 10-bit data to 8-bit data

  analogWrite(ledPin, HIGH); // generate PWM signal

  return outputValue;          // Return analog moisture value
```

```
}

/* How to use the DHT-22 sensor with Arduino uno Temperature and humidity sensor */

//Libraries

#include <DHT.h>;

//Constants

#define DHTPIN 7    // what pin we're connected to

#define DHTTYPE DHT22 // DHT 22 (AM2302)

DHT dht(DHTPIN, DHTTYPE); /// Initialize DHT sensor for normal 16mhz Arduino

//Variables

int chk;

float hum; //Stores humidity value

float temp; //Stores temperature value

void setup()

{

    Serial.begin(9600);

    dht.begin();

}

void loop()

{

    delay(2000);

    //Read data and store it to variables hum and temp

    hum = dht.readHumidity();

    temp= dht.readTemperature();

    //Print temp and humidity values to serial monitor

    Serial.print("Humidity: ");
```

```
Serial.print(hum);  
  
Serial.print(" %, Temp: ");  
  
Serial.print(temp);  
  
Serial.println(" Celsius");  
  
delay(10000); //Delay 2 sec.
```

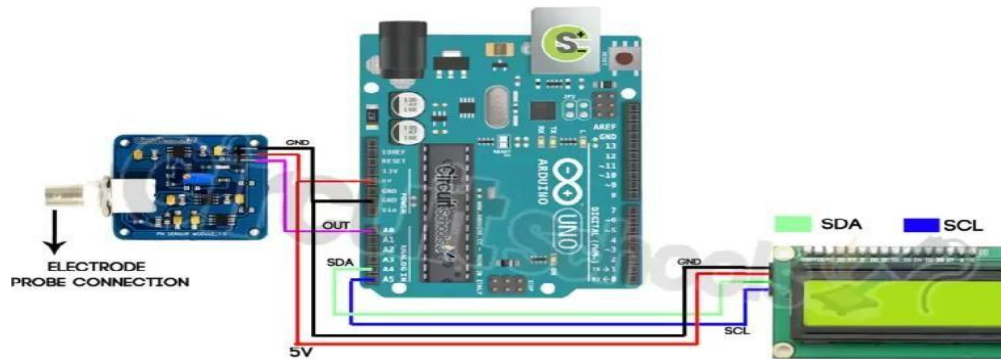
Output & Result: Program to classify dry and wet waste with the Moisture sensor (DHT22) is developed, executed and the output is verified successfully.

Experiment 7

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

Components Required: Arduino UNO Board with USB cable, Analog pH Sensor Module, 16x2 LCD module with I2C, Bread Board, Jumper Wires.

Schematic:



Sketch:

```
#include <LiquidCrystal_I2C.h>
```

```
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

```
int phval = 0; // the pH meter Analog output is connected with the Arduino's A0
```

```
unsigned long int avgval; //Store the average value of the sensor feedback
```

```
int buffer_arr[10],temp;
```

```
void setup()
```

```
{
```

```
  Serial.begin(9600);
```

```
  lcd.init();
```

```
  lcd.begin(16, 2);
```

```
  lcd.backlight();
```

```
  lcd.setCursor(0, 0);
```

```
  lcd.print("For more projects");
```

```
    lcd.setCursor(0, 1);

    lcd.print("CircuitSchools");

    delay(2000);

    lcd.clear();
}

void loop() {

    for(int i=0;i<10;i++) //Get 10 sample value from the sensor for smooth the value

    {

        buffer_arr[i]=analogRead(A0);

        delay(30);

    }

    for(int i=0;i<9;i++) //sort the analog from small to large

    {

        for(int j=i+1;j<10;j++)

        {

            if(buffer_arr[i]>buffer_arr[j])

            {

                temp=buffer_arr[i];

                buffer_arr[i]=buffer_arr[j];

                buffer_arr[j]=temp;

            }

        }

    }

    avgval=0;

    for(int i=2;i<8;i++) //take the average value of 6 center sample
```

```
avgval+=buffer_arr[i];  
  
float volt=(float)avgval*5.0/1024/6; //convert the analog into millivolt  
  
float ph_act = 3.5*volt; //convert the millivolt into pH value  
  
lcd.setCursor(0, 0);  
  
lcd.print("pH Val:");  
  
lcd.setCursor(8, 0);  
  
lcd.print(ph_act);  
  
delay(1000);  
  
}
```

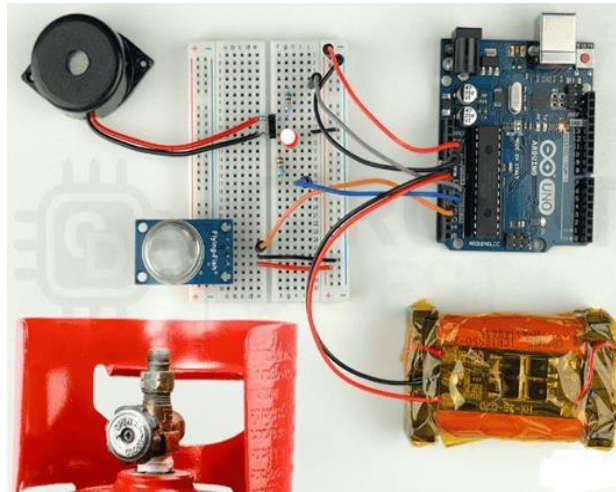
Output & Result: Program to read the pH value of a various substances like milk, lime and water is developed, executed and the output is verified successfully.

Experiment 8

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

Components Required: Arduino UNO Board with USB cable, MQ2 Gas Sensor Module, Buzzer, LED, 220Ω Resistor, Bread Board, Jumper Wires, 16X2 LCD with I2C Module (if needed).

Schematic:



Sketch:

```
const int gasSensorDigitalPin = A2;

const int ledPin = A1;

const int buzzerPin = A0;

void setup()

{

  pinMode(gasSensorDigitalPin, INPUT);

  pinMode(ledPin, OUTPUT);

  pinMode(buzzerPin, OUTPUT);

  Serial.begin(9600);

  Serial.println("Gas Detection System Initialized (Digital Mode)");

}

void loop (){

  int gasState = digitalRead(gasSensorDigitalPin);
```

```
if (gasState == LOW) {  
    digitalWrite(ledPin, HIGH);  
    digitalWrite(buzzerPin, HIGH);  
    Serial.println("!! Gas Leak Detected !!");  
} else {  
    digitalWrite(ledPin, LOW);  
    digitalWrite(buzzerPin, LOW);  
    Serial.println("Environment Normal");  
}  
delay(800);  
}
```

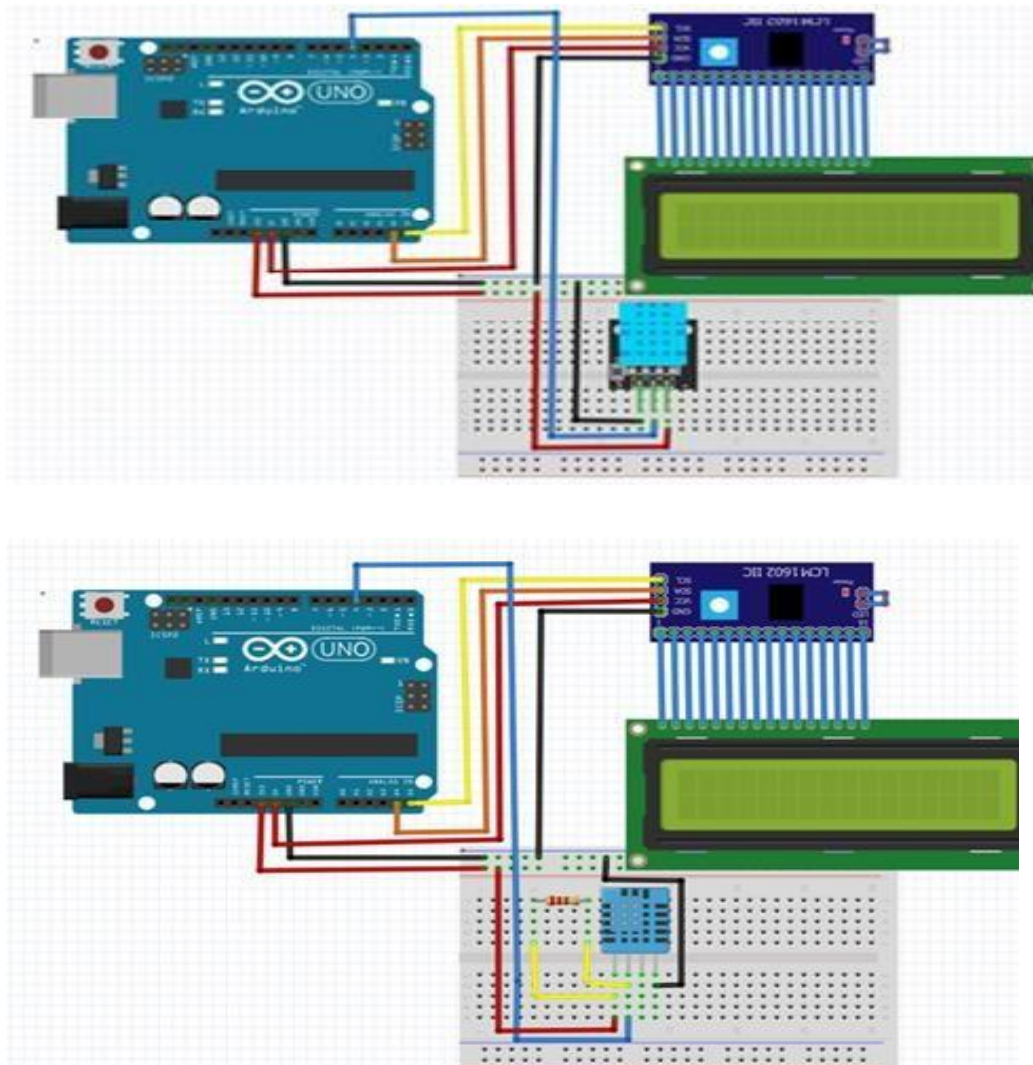
Output & Result: Program to detect the gas leakage in the surrounding environment is developed, executed and the output is verified successfully.

Experiment 9

Aim: Develop a program to demonstrate weather station readings using Arduino.

Components Required: Arduino UNO Board with USB cable, DHT22 Sensor Module, 16X2 LCD with I2C Module, Rain Sensor Module, Bread Board, Jumper Wires.

Schematic:



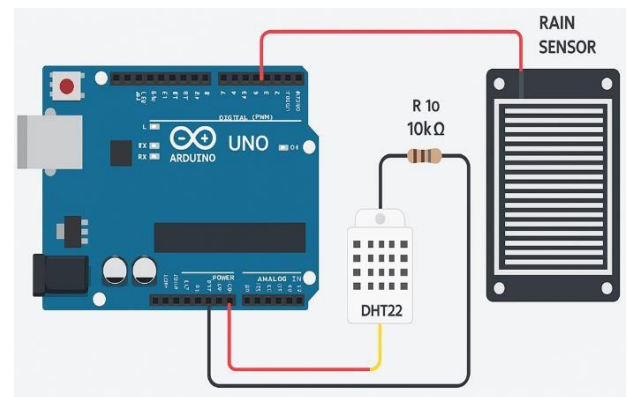
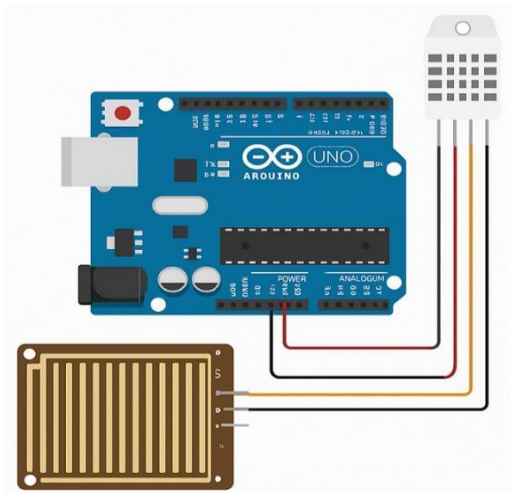
If you are using 4 pins DHT22 Connection are as follows

DHT22 Arduino UNO Vcc 3.3V Out PIN4 (Digital) GND GND NC --

Connect a 10K Ohm resistor between Vcc and Out Pin of DHT22.

If you are using 3 pins DHT22 Connection are as follows

DHT22 Arduino UNO Vcc 3.3V Out PIN4 (Digital) GND GND

**Sketch:**

```
#include <DHT.h>;

#define RAIN_SENSOR_PIN A0

#define RAIN_THRESHOLD 600

//Constants

#define DHTPIN 7    // what pin we're connected to
#define DHTTYPE DHT22 // DHT 22 (AM2302)

DHT dht(DHTPIN, DHTTYPE); /// Initialize DHT sensor for normal 16 mhz Arduino

//Variables

int chk;

float hum; //Stores humidity value

float temp; //Stores temperature value

void setup() {

    pinMode(RAIN_SENSOR_PIN, INPUT);

    Serial.begin(9600);

    Serial.println("weather station");

    dht.begin();

}
```

```
void loop() {  
    int sensorValue = analogRead(RAIN_SENSOR_PIN);  
  
    Serial.print("Sensor Value: ");  
  
    Serial.println(sensorValue);  
  
    if (sensorValue < RAIN_THRESHOLD) {  
        Serial.println("RAIN DETECTED!");  
    } else {  
        Serial.println("No rain detected!");  
    }  
  
    delay(2000);  
  
    //Read data and store it to variables hum and temp  
    hum = dht.readHumidity();  
    temp= dht.readTemperature();  
  
    //Print temp and humidity values to serial monitor  
    Serial.print("Humidity: ");  
  
    Serial.print(hum);  
  
    Serial.print(" %, Temp: ");  
  
    Serial.print(temp);  
  
    Serial.println(" Celsius");  
  
    delay(1000); //Delay 2 sec.  
}
```

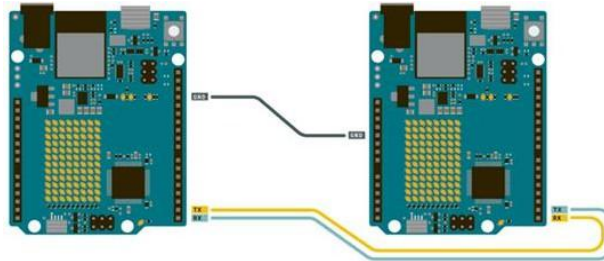
Output & Result: Program to demonstrate weather station readings using Arduino is developed, executed and the output is verified successfully.

Experiment 10

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

Components Required: Arduino UNO Board with USB cable 2 quantity, Push Button Module 2 quantity (if needed), LEDs 2 quantities (if needed).

Schematic:



Sketch:

```
String sendMessage;  
  
String receivedMessage;  
  
void setup()  
{  
    Serial.begin(9600); // Initialize the Serial monitor for debugging  
    Serial1.begin(9600); // Initialize Serial1 for sending data  
}  
  
void loop()  
{  
    while (Serial1.available() > 0) {  
        char receivedChar = Serial1.read();  
        if (receivedChar == '\n')
```

```
{  
    Serial.println(receivedMessage); // Print the received message in the Serial monitor  
    receivedMessage = ""; // Reset the received message  
}  
else  
{  
    receivedMessage += receivedChar; // Append characters to the received message  
}  
}  
if (Serial.available() > 0)  
{  
    char inputChar = Serial.read();  
    if (inputChar == '\n')  
    {  
        Serial1.println(sendMessage); // Send the message through Serial1 with a newline character  
        sendMessage = ""; // Reset the message  
    }  
else  
    {  
        sendMessage += inputChar; // Append characters to the message  
    }  
}  
}
```

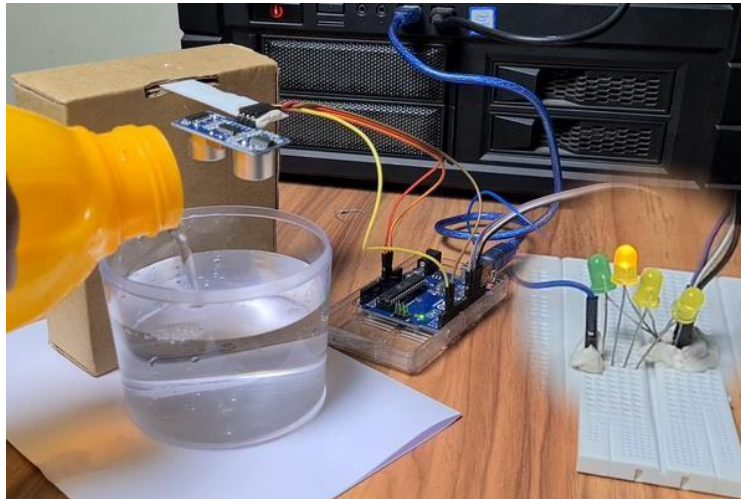
Output & Result: Program to setup a UART protocol and pass a string through the protocol is developed, executed and the output is verified successfully.

Experiment 11

Aim: Develop a water level depth detection system using Ultrasonic sensor.

Components Required: Arduino UNO Board with USB cable, Ultrasonic Sensor, Bread Board, Jumper Wires, LEDs 5 quantity (if needed), 220 Ω Resistors 5 quantity (if needed).

Schematic:



Sketch:

```
int trig = 12;

int echo = 11;

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

    pinMode(trig, OUTPUT);

    pinMode(echo, INPUT);
}

void loop()
{
    long t = 0, h = 0, hp = 0;
```

```
// Transmitting pulse

digitalWrite(trig, LOW);

delayMicroseconds(2);

digitalWrite(trig, HIGH);

delayMicroseconds(10);

digitalWrite(trig, LOW);

// Waiting for pulse

t = pulseIn(echo, HIGH);

// Calculating distance

h = t / 58;

h = h - 6; // offset correction

h = 50 - h; // water height, 0 - 50 cm

hp = 2 * h; // distance in %, 0-100 %

// Sending to computer

Serial.print(hp);

// Serial.print(" cm\n");

Serial.print("\n");

delay(1000);

}
```

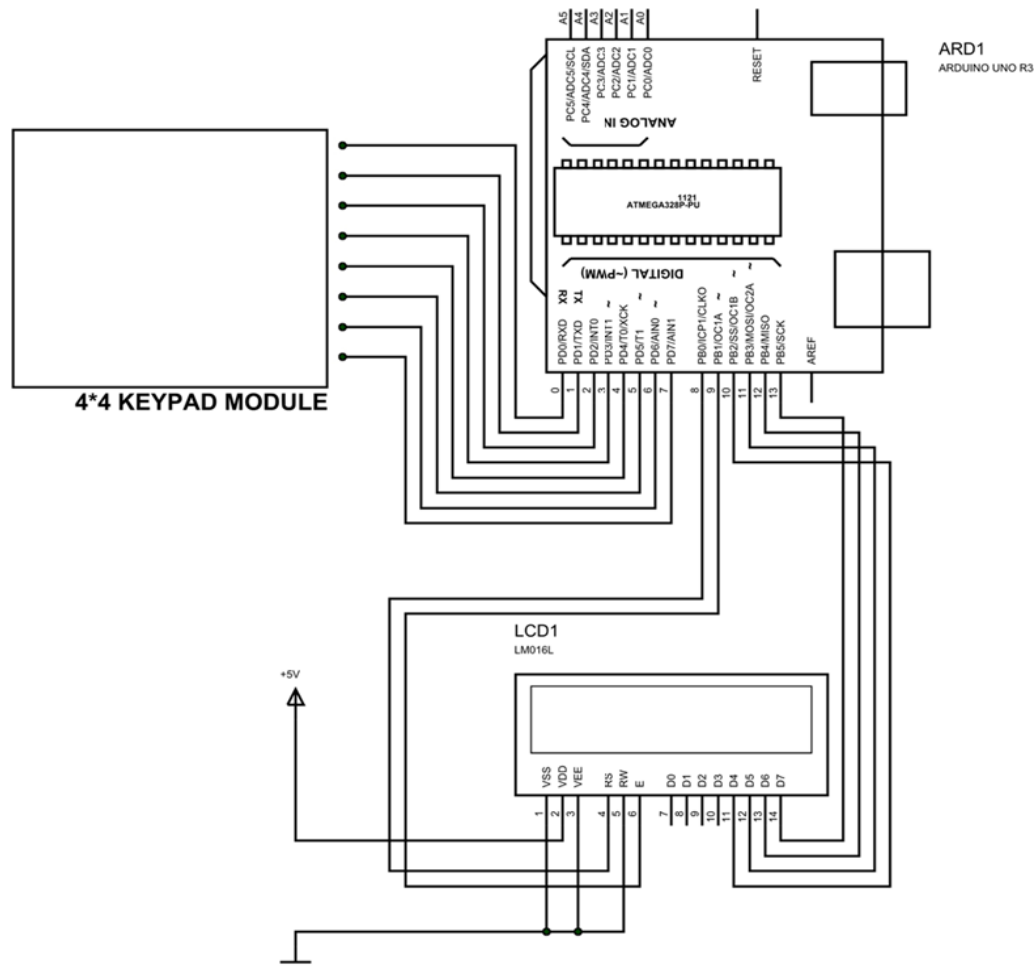
Output & Result: Program to water level depth detection system using Ultrasonic sensor is developed, executed and the output is verified successfully.

Experiment 12

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

Components Required: Arduino UNO Board with USB cable, 4x4 Keypad Module, 16X2 LCD with I2C Module, Bread Board, Jumper Wires

Schematic:



The connections which are done for LCD are given below:

PIN1 or VSS to ground

PIN2 or VDD or VCC to +5v power

PIN3 or VEE to ground (gives maximum contrast best for a beginner)

PIN4 or RS (Register Selection) to PIN8 of ARDUINO UNO

PIN5 or RW (Read/Write) to ground (puts LCD in read mode eases the communication for user)

PIN6 or E (Enable) to PIN9 of ARDUINO UNO

PIN11 or D4 to PIN10 of ARDUINO UNO

PIN12 or D5 to PIN11 of ARDUINO UNO

PIN13 or D6 to PIN12 of ARDUINO UNO

PIN14 or D7 to PIN13 of ARDUINO UNO

Sketch:

```
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins

LiquidCrystal lcd(8, 9, 10, 11, 12, 13); //RS,EN,D4,D5,D6,D7

#include <Keypad.h> //header for keypad commands enabling

const byte ROWS = 4; // Four rows

const byte COLS = 4; // Three columns

// Define the Keymap

char keys[ROWS][COLS] = {

  {'1','2','3','A'},

  {'4','5','6','B'},

  {'7','8','9','C'},

  {'#','0','*','D'}

};

// Connect keypad ROW0, ROW1, ROW2 and ROW3 to these Arduino pins.

byte rowPins[ROWS] = { 0, 1, 2, 3 };

// Connect keypad COL0, COL1 and COL2 to these Arduino pins.

byte colPins[COLS] = { 4, 5, 6, 7 };

// Create the Keypad

Keypad kpd = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

void setup()

{

  for(int k=8;k<14;k++)

  {

    pinMode(k,OUTPUT); //pins 8-14 are enabled as output
```

```
}

lcd.begin(16, 2); //initializing LCD

}

void loop()

{

    char key = kpd.getKey(); //storing pressed key value in a char

    if (key != NO_KEY)

    {

        lcd.print(key); //showing pressed character on LCD

    }

}
```

Output & Result: Program to simulate interfacing with the keypad module to record the keystrokes is developed, executed and the output is verified successfully.