FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

B. E. (COMPUTER SCIENCE AND ENGINEERING)

VII Semester

22CSCP706 – Embedded Systems and Internet of Things (IoT) Lab

Name : .....

Reg. No. : .....

ANNAMALAI UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

B. E. (COMPUTER SCIENCE AND ENGINEERING)

VII Semester

22CSCP706 – Embedded Systems and Internet of Things (IoT) Lab

Certified that this is a bonafide record of work done by

Mr./Ms.....

Reg. No. ..... of B.E. (CSE) in the 22CSCP706 - EMBEDDED SYSTEMS AND INTERNET OF THINGS (IoT) LAB during the odd semester of the academic year 2025–26.

Staff-in-charge

Internal Examiner

Place: Annamalai Nagar
Date :

External Examiner

CONTENTS

S. No.	Date	List of Experiments	Page No	Mark	Sign
01		DISTANCE MEASUREMENT USING ARDUINO	01		
02		IDENTIFYING MOISTURE CONTENT IN AGRICULTURAL LAND USING ARDUINO	05		
03		MOTION DETECTION USING ARDUINO	09		
04		IDENTIFYING ROOM TEMPERATURE AND HUMIDITY USING ARDUINO	12		
05		COLOUR RECOGNITION USING ARDUINO	16		
06		FIRE ALARM INDICATOR USING ARDUINO	19		
07		SOUND DETECTION USING ARDUINO	22		
08		INTERFACING FLEX SENSOR WITH ARDUINO	25		
09		INTERFACING FORCE PRESSURE SENSOR WITH ARDUINO	28		
10		IDENTIFYING ROOM TEMPERATURE AND HUMIDITY USING RASPBERRY PI	31		
11		PIR MOTION SENSOR INTERFACING WITH RASPBERRY PI	34		
12		MEASURING SOIL MOISTURE WITH RASPBERRY PI	36		
13		SOUND SENSOR INTERFACING WITH RASPBERRY PI	38		
14		ANALOG SENSOR WITH ESP32 MICROCONTROLLER	41		
15		DHT11 SENSOR WITH ESP 32	43		
				Total:	
				Average:	

Annamalai University
Department of Computer Science and Engineering

VISION

To provide a congenial ambience for individuals to develop and blossom as academically superior, socially conscious and nationally responsible citizens.

MISSION

- Impart high quality computer knowledge to the students through a dynamic scholastic environment wherein they learn to develop technical, communication and leadership skills to bloom as a versatile professional.
- Develop life-long learning ability that allows them to be adaptive and responsive to the changes in career, society, technology, and environment.
- Build student community with high ethical standards to undertake innovative research and development in thrust areas of national and international needs.
- Expose the students to the emerging technological advancements for meeting the demands of the industry.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO	PEO Statements
PEO1	To equip the graduates with fundamental concepts and impart problem solving skills that will help them to pursue professional careers in Computer Science and Engineering.
PEO2	To provide the graduates with the requisite knowledge to pursue higher education and carry out research in the field of Computer Science and Engineering.
PEO3	To equip the graduates with the ability to acquire new skills in emerging areas of Computer Science and Engineering such as artificial intelligence, machine learning and data science and enable them to become successful professionals and entrepreneurs.
PEO4	To ensure that the graduates exhibit high levels of ethical and moral behavior as computer engineers in addressing societal problems and providing sustainable development for the betterment of society.

PROGRAM OUTCOMES (POs)

S. No.	Program Outcomes
PO1	**Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**Problem Analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.
PO3	**Design/Development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	**Conduct Investigations of Complex Problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	**Modern Tool Usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	**The Engineer and Society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	**Environment and Sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and Team Work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10	**Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Project Management and Finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	**Life-long Learning:** Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

S. No.	Program Specific Outcomes
PSO1	Ability to investigate challenging problems in various domains and exhibit programming skills to build automation solutions.
PSO2	Ability to apply the algorithms and computational techniques to construct robust and resilient computer systems and applications.
PSO3	Ability to comprehend and implement the contemporary trends in industry and research environment paving the way for innovative solutions to existing and emerging problems.

Rubrics for Laboratory Examination (Internal/External)

(Internal: Two tests - 15 marks each, **External:** Two questions - 25 marks each)

Rubric	Poor Up to (1/2)	Average Up to (2/4)	Good Up to (3/6)	Excellent Up to (5/8*)
<u>Syntax and Logic</u> Ability to understand, specify the data structures appropriate for the problem domain	Program does not compile with typographical errors and incorrect logic leading to infinite loops.	Program compiles that signals major syntactic errors and logic shows severe errors.	Program compiles with minor syntactic errors and logic is mostly correct with occasional errors.	Program compiles with evidence of good syntactic understanding of the syntax and logic used.
<u>Modularity</u> Ability to decompose a problem into coherent and reusable functions, files, classes, or objects (as appropriate for the programming language and platform).	Program is one big Function or is decomposed in ways that make little/no sense.	Program is decomposed into units of appropriate size, but they lack coherence or reusability. Program contains unnecessary repetition.	Program is decomposed into coherent units, but may still contain some unnecessary repetition.	Program is decomposed into coherent and reusable units, and unnecessary repetition are eliminated.
<u>Clarity and Completeness</u> Ability to code formulae and algorithms that produce appropriate results. Ability to apply rigorous test case analysis to the problem domain.	Program does not produce appropriate results for most inputs. Program shows little/no ability to apply different test cases.	Program approaches appropriate results for most inputs, but contain some miscalculations. Program shows evidence of test case analysis, but missing significant test cases or mistaken some test cases.	Program produces appropriate results for most inputs. Program shows evidence of test case analysis that is mostly complete, but missed to handle all possible test cases.	Program produces appropriate results for all inputs tested. Program shows evidence of excellent test case analysis, and all possible cases are handled appropriately.

* 8 marks for syntax and logic, 8 marks for modularity, and 9 marks for Clarity and Completeness.

Rubric for CO3

Rubric for CO3 in Laboratory Courses				
Rubric	**Distribution of 10 Marks for CIE/SEE Evaluation Out of 40/60 Marks**			
	Up To 2.5 Marks	**Up To 5 Marks**	**Up To 7.5 Marks**	**Up To 10 marks**
Demonstrate an ability to listen and answer the viva questions related to programming skills needed for solving real-world problems in Computer Science and Engineering.	Poor listening and communication skills. Failed to relate the programming skills needed for solving the problem.	Showed better communication skill by relating the problem with the programming skills acquired but the description showed serious errors.	Demonstrated good communication skills by relating the problem with the programming skills acquired with few errors.	Demonstrated excellent communication skills by relating the problem with the programming skills acquired and have been successful in tailoring the description.

Ex. No: 01 **DISTANCE MEASUREMENT USING ARDUINO**
Date:

AIM:

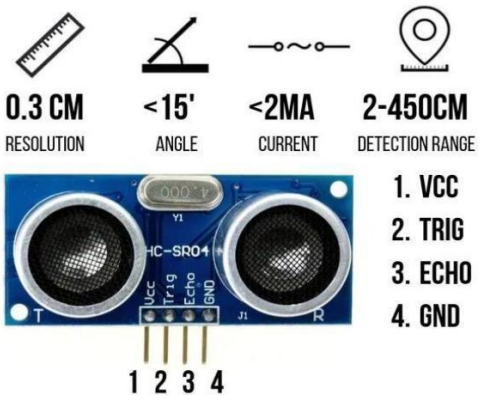
To measure the Distance of an Object using Ultrasonic Sensor HC–SR04.

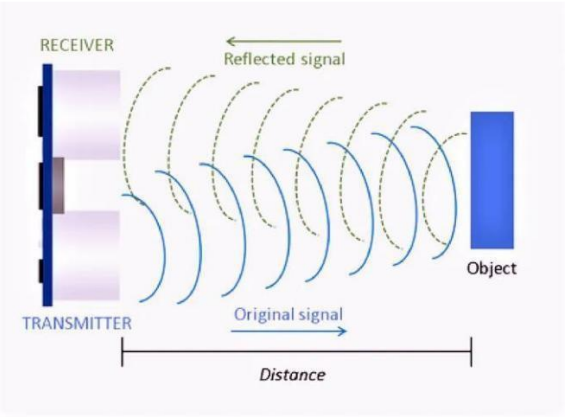
LIST OF COMPONENTS:

1. Arduino UNO
2. Ultrasonic Sensor HC-SR04
3. Solderless Breadboard
4. Jumper Wires
5. Buzzer
6. 2x LEDs
7. 2x Resistor 220Ω

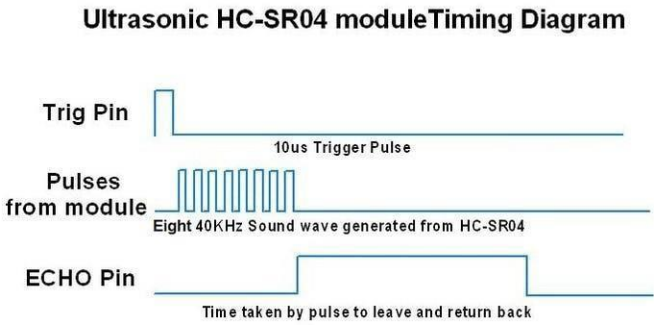
WORKING DESCRIPTION:

Ultrasonic Sensor HC-SR04 is a sensor that can measure distance. It emits an ultrasound at 40 000 Hz (40 kHz) which travels through the air and if there is an object or obstacle on its path It will bounce back to the module. Considering the travel time and the speed of the sound you can calculate the distance. The configuration pin of HC-SR04 is VCC (1), TRIG (2), ECHO (3), and GND (4). The supply voltage of VCC is +5V and you can attach TRIG and ECHO pin to any Digital I/O in your Arduino Board.



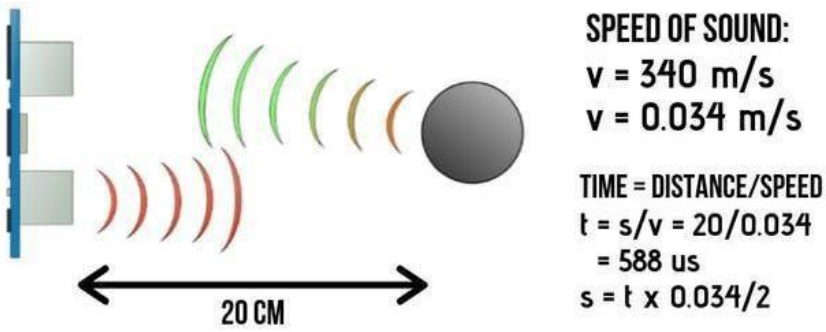


In order to generate the ultrasound we need to set the Trigger Pin on a High State for 10 µs. That will send out an 8 cycle sonic burst which will travel at the speed sound and it will be received in the Echo Pin. The Echo Pin will output the time in microseconds the sound wave travelled.

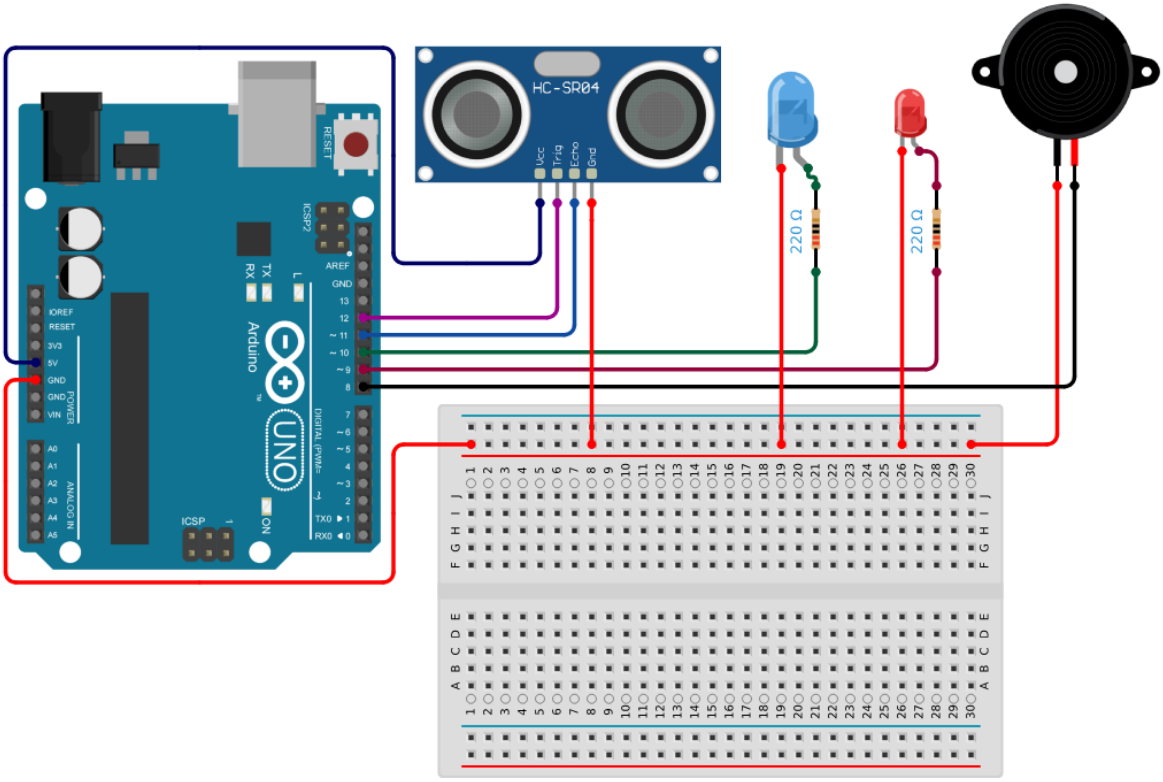


For example, if the object is 20 cm away from the sensor, and the speed of the sound is 340 m/s or 0.034 cm/µs the sound wave will need to travel about 588 microseconds. But what you will get from the Echo

pin will be double that number because the sound wave needs to travel forward and bounce backward. So in order to get the distance in cm we need to multiply the received travel time value from the echo pin by 0.034 and divide it by 2.



CONNECTION DIAGRAM:



CONNECTIONS:

- **Ultrasonic sensor to Arduino:**
  - Vcc to 5V
  - Trig to pin 12
  - Gnd to GND
  - Echo to pin 11
- **LED BLUE to Arduino:**
  - Negative to GND
  - Positive to pin 10 and 220Ω Resistor
- **LED RED to Arduino:**

- Negative to GND
- Positive to pin 9 and 220Ω Resistor

➢ **Buzzer to Arduino:**

- Negative to GND
- Positive to pin 8

PROGRAM:

```
const int buzzer = 8;
const int redLED = 9;
const int blueLED = 10;
const int echoPin = 11;
const int trigPin = 12;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(blueLED, OUTPUT);
  pinMode(redLED, OUTPUT);
  pinMode(buzzer, OUTPUT);
  Serial.begin(9600);
  Serial.println("Distance Measurement Ready");
}

void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH, 30000);
  int distance = duration * 0.034 / 2;

  if (duration == 0) {
    Serial.println("Out of range");
    digitalWrite(blueLED, LOW);
    digitalWrite(redLED, LOW);
    noTone(buzzer);
  } else {
    Serial.print("Distance: "); Serial.print(distance); Serial.println(" cm");
    digitalWrite(blueLED, LOW);
    digitalWrite(redLED, LOW);
    noTone(buzzer);

    if (distance <= 10) {
      Serial.println("VERY CLOSE!");
      digitalWrite(blueLED, HIGH);
      tone(buzzer, 1000);
```

```
      delay(500);
      noTone(buzzer);
    } else if (distance <= 50) {
      Serial.println("NEAR RANGE");
      digitalWrite(redLED, HIGH);
    } else {
      Serial.println("FAR RANGE");
    }
  }
  delay(100);
}
```

OUTPUT:

```
Distance Measurement Ready
Distance: 8 cm
VERY CLOSE!
Distance: 40 cm
NEAR RANGE
Distance: 100 cm
FAR RANGE
```

RESULT:

Thus, the Distance of an Object was measured using Arduino successfully.

Ex. No: 02 IDENTIFYING MOISTURE CONTENT IN AGRICULTURAL LAND USING ARDUINO
Date:

AIM:

To identify the Moisture content of soil in Agricultural Land using Soil Moisture Sensor.

LIST OF COMPONENTS:

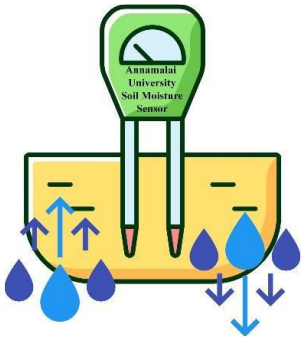
1. Arduino Uno
2. Soil Moisture Sensor (Analog)
3. Solderless Breadboard
4. Buzzer
5. 3 x LED
6. 3 x 220Ω Resistors
7. Jumper Wires

WORKING DESCRIPTION:

Soil moisture sensors measure the volumetric water content in soil. Since the direct gravimetric measurement of free-soil moisture requires removing, drying, and weighing of a sample, soil moisture sensors measure the volumetric water content indirectly by using some other property of the soil, such as electrical resistance, dielectric constant, or interaction with neutrons, as a proxy for the moisture content.

The relation between the measured property and soil moisture must be calibrated and may vary depending on environmental factors such as soil type, temperature, or electric conductivity. Reflected microwave radiation is affected by the soil moisture and is used for remote sensing in hydrology and agriculture. Portable probe instruments can be used by farmers or gardeners.

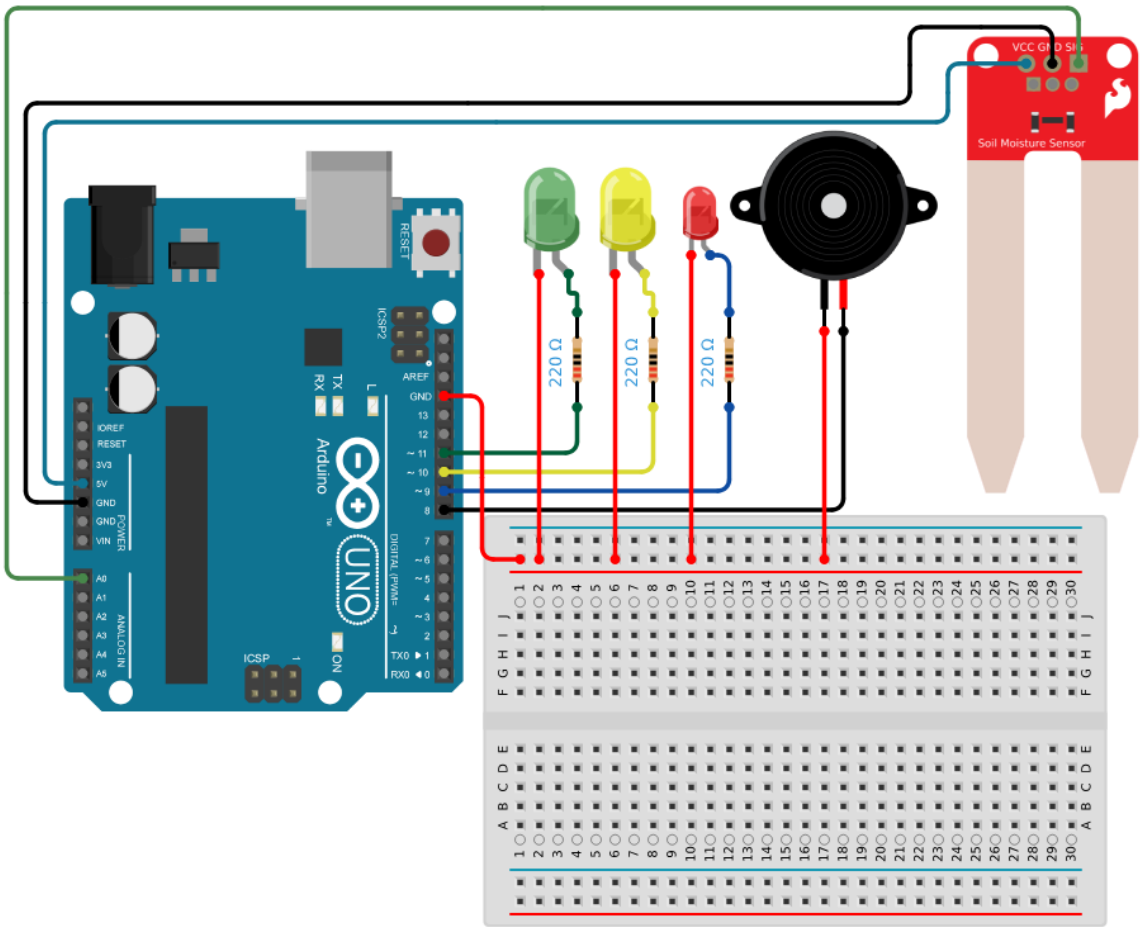
Soil moisture sensors typically refer to sensors that estimate volumetric water content. Another class of sensors measure another property of moisture in soils called water potential; these sensors are usually referred to as soil water potential sensors and include tensiometers and gypsum blocks.



There are different types of soil moisture sensor on the market, but their working principal are all similar. All of these sensors have at least three pins: VCC, GND, and AO. The AO pin changes according to the amount of moisture in the soil and increases as there is more water in the soil. Some models have an additional base called DO. If the moisture amount is less than the permissible amount (which can be changed by the potentiometer on the sensor) the DO pin will be “1”, otherwise will remain “0”.

In this Experiment, we have used the Soil Moisture Sensor. It has a detection length of 38mm and a working voltage of 2V-5V. It has a Fork-like design, which makes it easy to insert into the soil. The analog output voltage boosts along with the soil moisture level increases.

CONNECTION DIAGRAM:



CONNECTIONS:

- **Soil Moisture Sensor to Arduino:**
  - VCC to 5V
  - GND to GND
  - AO to A0
- **Red LED to Arduino:**
  - Negative to GND
  - Positive to pin 9 and 220Ω Resistor
- **Yellow LED to Arduino:**
  - Negative to GND
  - Positive to pin 10 and 220Ω Resistor
- **Green LED to Arduino:**
  - Negative to GND
  - Positive to pin 11 and 220Ω Resistor
- **Buzzer to Arduino:**
  - Negative to GND
  - Positive to pin 8

PROGRAM:

```
const int buzzer = 8;
const int ledDry = 9;
const int ledHumid = 10;
const int ledWet = 11;

void setup() {
  pinMode(A0, INPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(ledDry, OUTPUT);
  pinMode(ledHumid, OUTPUT);
  pinMode(ledWet, OUTPUT);
  Serial.begin(9600);
  Serial.println("Soil Moisture Monitoring Ready");
}

void loop() {
  int sensorValue = analogRead(A0);

  digitalWrite(ledDry, LOW);
  digitalWrite(ledHumid, LOW);
  digitalWrite(ledWet, LOW);
  noTone(buzzer);

  if (sensorValue >= 1000) {
    Serial.println("Not in soil or disconnected");
  }
  else if (sensorValue >= 600) {
    Serial.println("DRY SOIL");
    digitalWrite(ledDry, HIGH);
    tone(buzzer, 1000);
    delay(1000);
    noTone(buzzer);
  }
  else if (sensorValue >= 370) {
    Serial.println("HUMID SOIL");
    digitalWrite(ledHumid, HIGH);
  }
  else {
    Serial.println("WATER SOIL");
    digitalWrite(ledWet, HIGH);
  }

  Serial.println("----------");
  delay(2000);
}
```

OUTPUT:

```
Soil Moisture Monitoring Ready
DRY SOIL
----------
HUMID SOIL
----------
WATER SOIL
----------
```

RESULT:

Thus, the Moisture content in Agricultural Land was measured using Arduino successfully.

Ex. No: 03 **MOTION DETECTION USING ARDUINO**
Date:

AIM:

To build a Motion Detection System using Arduino.

LIST OF COMPONENTS:

1. Arduino Uno
2. PIR Motion Sensor
3. Buzzer
4. LED (Red)
5. 220Ω Resistor
6. Jumper Wires

WORKING DESCRIPTION:

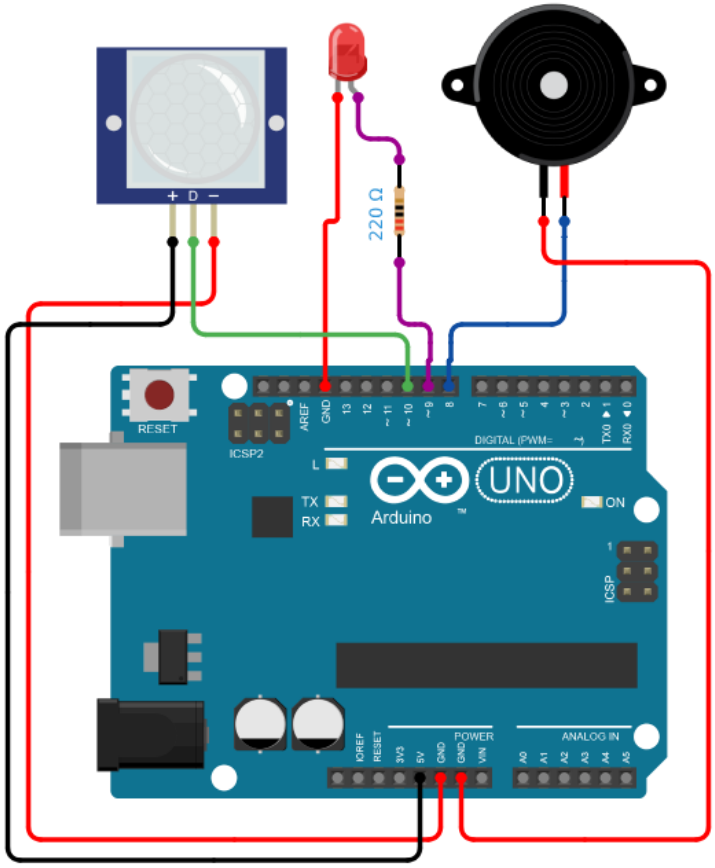
Home Automation is what makes us call our place a Smart Home. We can control all the appliances at our home using automation. The devices within the home automation system connect with each over a local network. When connected with the Internet, home devices are an important constituent of the Internet of Things. Mainly Home automation is to control or monitor signals from different appliances. Simply, this helps to make our lives easy.

An LDR (Light Dependent Resistor) is a component that has a (variable) resistance that changes with the light intensity that falls upon it. This allows them to be used in light sensing circuits. A photoresistor is made of a high resistance semiconductor. In the dark, a photoresistor can have a resistance as high as several megohms (MΩ), while in the light, a photoresistor can have a resistance as low as a few hundred ohms. If incident light on a photoresistor exceeds a certain frequency, photons absorbed by the semiconductor give bound electrons enough energy to jump into the conduction band.

The resulting free electrons (and their hole partners) conduct electricity, thereby lowering resistance. This example demonstrates the use of a LDR as a switch. Each time you cover the photocell, the LED (or whatever) is turned on or off.

PIR sensor detects a human being moving around within approximately 10m from the sensor. This is an average value, as the actual detection range is between 5m and 12m.PIR are fundamentally made of a pyro electric sensor, which can detect levels of infrared radiation. For numerous essential projects or items that need to discover when an individual has left or entered the area. PIR sensors are incredible, they are flat control and minimal effort, have a wide lens range, and are simple to interface with.

CONNECTION DIAGRAM:



CONNECTIONS:

- **PIR Sensor to Arduino:**
  - VCC to 5V
  - GND to GND
  - OUT to pin 10
- **Red LED to Arduino:**
  - Negative to GND
  - Positive to Pin 9 and 220Ω Resistor
- **Buzzer to Arduino:**
  - Negative to GND
  - Positive to pin 8

PROGRAM:

```
const int buzzer = 8;
const int led = 9;
const int pirPin = 10;
bool motionDetected = false;

void setup() {
  Serial.begin(9600);
  pinMode(pirPin, INPUT);
  pinMode(led, OUTPUT);
  pinMode(buzzer, OUTPUT);
```

```
  Serial.println("Motion Detection Ready");
}

void loop() {
  int motion = digitalRead(pirPin);
  if (motion == HIGH) {
    if (!motionDetected) {
      Serial.println("Motion Detected");
      digitalWrite(led, HIGH);
      tone(buzzer, 1000);
      delay(500);
      noTone(buzzer);
      motionDetected = true;
    }
  } else {
    if (motionDetected) {
      Serial.println("Motion stopped!");
      digitalWrite(led, LOW);
      motionDetected = false;
    }
  }
  delay(100);
}
```

OUTPUT:

```
Motion Detection Ready
Motion Detected
Motion stopped!
```

RESULT:

Thus, the Motion detection System was built successfully using Arduino.

Ex. No: 04 IDENTIFYING ROOM TEMPERATURE AND HUMIDITY USING ARDUINO
Date:

AIM:

To identify the Room Temperature and Humidity using DHT11 Sensor.

LIST OF COMPONENTS:

1. Arduino Uno
2. DHT11 Sensor
3. Solderless Breadboard
4. Buzzer
5. 3 x LED
6. 3 x 220Ω Resistors
7. Jumper Wires

WORKING DESCRIPTION:

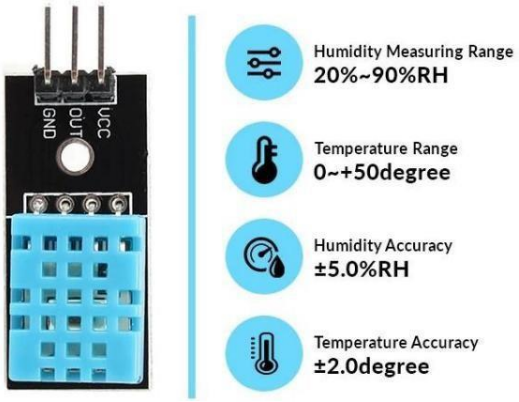
DHT11 is a low-cost digital sensor for sensing temperature and humidity. This sensor can be easily interfaced with any micro-controller such as Arduino, Raspberry Pi etc. to measure humidity and temperature instantaneously. DHT11 humidity and temperature sensor is available as a sensor and as a module. The difference between this sensor and module is the pull-up resistor and a power-on LED. DHT11 is a relative humidity sensor. To measure the surrounding air this sensor uses a thermistor and a capacitive humidity sensor.

WORKING PRINCIPLE OF DHT11 SENSOR:

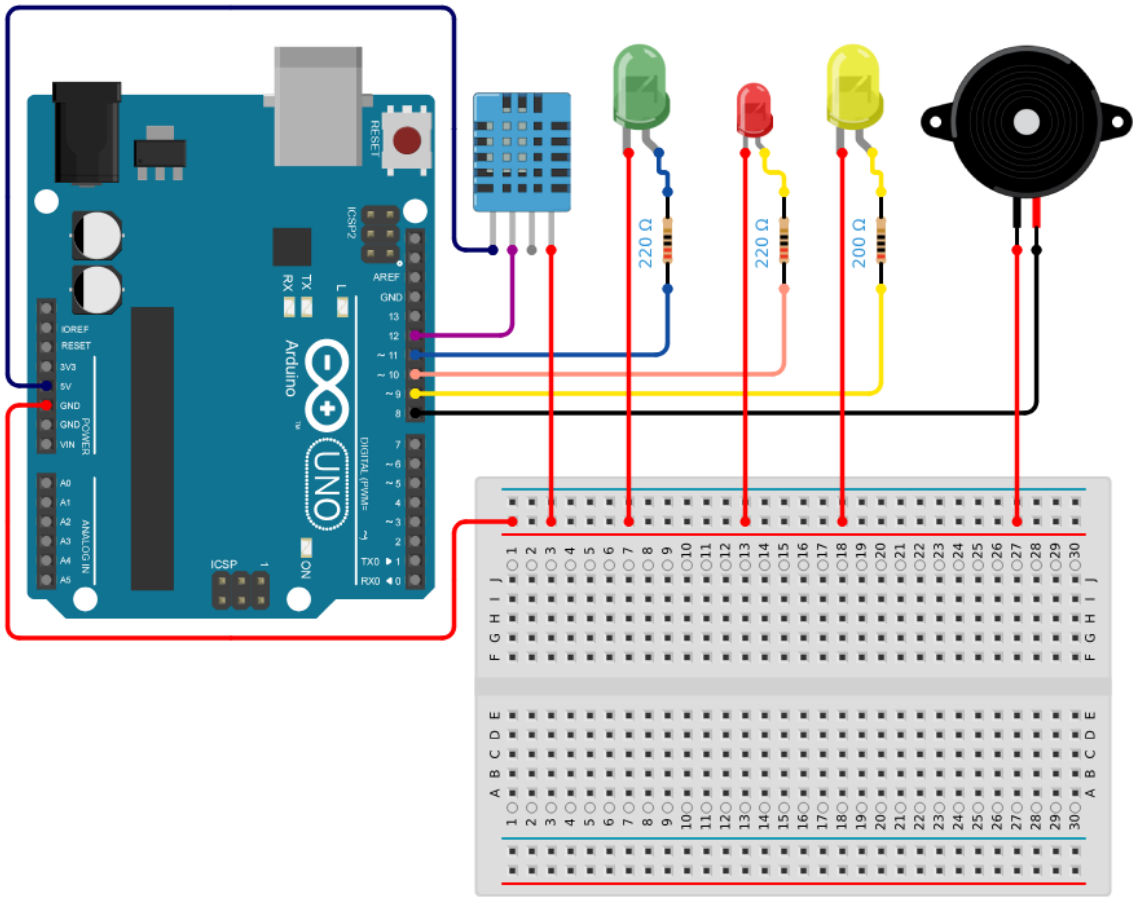
DHT11 sensor consists of a capacitive humidity sensing element and a thermistor for sensing temperature. The humidity sensing capacitor has two electrodes with a moisture holding substrate as a dielectric between them. Change in the capacitance value occurs with the change in humidity levels. The IC measure, process this changed resistance values and change them into digital form.

For measuring temperature this sensor uses a Negative Temperature coefficient thermistor, which causes a decrease in its resistance value with increase in temperature. To get larger resistance value even for the smallest change in temperature, this sensor is usually made up of semiconductor ceramics or polymers.

The temperature range of DHT11 is from 0 to 50 degree Celsius with a 2-degree accuracy. Humidity range of this sensor is from 20 to 80% with 5% accuracy. The sampling rate of this sensor is 1Hz .i.e. it gives one reading for every second. DHT11 is small in size with operating voltage from 3 to 5 volts. The maximum current used while measuring is 2.5mA.



CONNECTION DIAGRAM:



CONNECTIONS:

- **DHT11 Sensor to Arduino:**
  - VCC to 5V
  - GND to GND
  - Data to pin 12
- **Error LED to Arduino:**
  - Negative to GND
  - Positive to pin 9 and 220Ω Resistor
- **High Temp LED to Arduino:**
  - Negative to GND
  - Positive to pin 10 and 220Ω Resistor
- **Normal LED to Arduino:**
  - Negative to GND
  - Positive to pin 11 and 220Ω Resistor
- **Buzzer to Arduino:**
  - Negative to GND
  - Positive to pin 8

PROGRAM:

```
#include "DHT.h"
#define DHTPIN 12
#define DHTTYPE DHT11
```

```
DHT dht(DHTPIN, DHTTYPE);

const int buzzer = 8;
const int ledError = 9;
const int ledHighTemp = 10;
const int ledNormal = 11;

void setup() {
  Serial.begin(9600);
  dht.begin();
  pinMode(buzzer, OUTPUT);
  pinMode(ledError, OUTPUT);
  pinMode(ledHighTemp, OUTPUT);
  pinMode(ledNormal, OUTPUT);
  Serial.println("Climate Monitoring Ready");
}

void loop() {
  digitalWrite(ledError, LOW);
  digitalWrite(ledHighTemp, LOW);
  digitalWrite(ledNormal, LOW);
  noTone(buzzer);

  delay(2000);
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  if (isnan(h) || isnan(t)) {
    Serial.println("Sensor Error!");
    digitalWrite(ledError, HIGH);
    delay(1000);
    return;
  }

  Serial.print("Humidity: "); Serial.print(h);
  Serial.print("% Temp: "); Serial.print(t); Serial.println("°C");

  if (t > 29) {
    Serial.println("HIGH TEMP ALERT!");
    digitalWrite(ledHighTemp, HIGH);
    tone(buzzer, 1000);
    delay(500);
    noTone(buzzer);
  } else {
    Serial.println("Normal Climate");
    digitalWrite(ledNormal, HIGH);
  }
  delay(1000);
}
```

OUTPUT:

```
Climate Monitoring Ready
Humidity: 45% Temp: 28.00°C
Normal Climate
Humidity: 45% Temp: 28.00°C
Normal Climate
Humidity: 45% Temp: 28.00°C
Normal Climate
Humidity: 50% Temp: 31.00°C
HIGH TEMP ALERT!
Humidity: 50% Temp: 31.00°C
HIGH TEMP ALERT!
Humidity: 50% Temp: 31.00°C
HIGH TEMP ALERT!
```

RESULT:

Thus, the Room Temperature and Humidity was measured successfully using DHT11 sensor.

Ex. No: 05 **COLOUR RECOGNITION USING ARDUINO**
Date:

AIM:

To identify the different colours using Colour Recognition Sensor.

LIST OF COMPONENTS:

1. Arduino Uno
2. TCS3200 Color Sensor
3. Jumper Wires

WORKING DESCRIPTION:

Colour sensors provide more reliable solutions to complex automation challenges. They are used in various industries including the food and beverage, automotive and manufacturing industries for purposes such as detecting material, detecting colour marks on parts, verifying steps in the manufacturing process and so on.

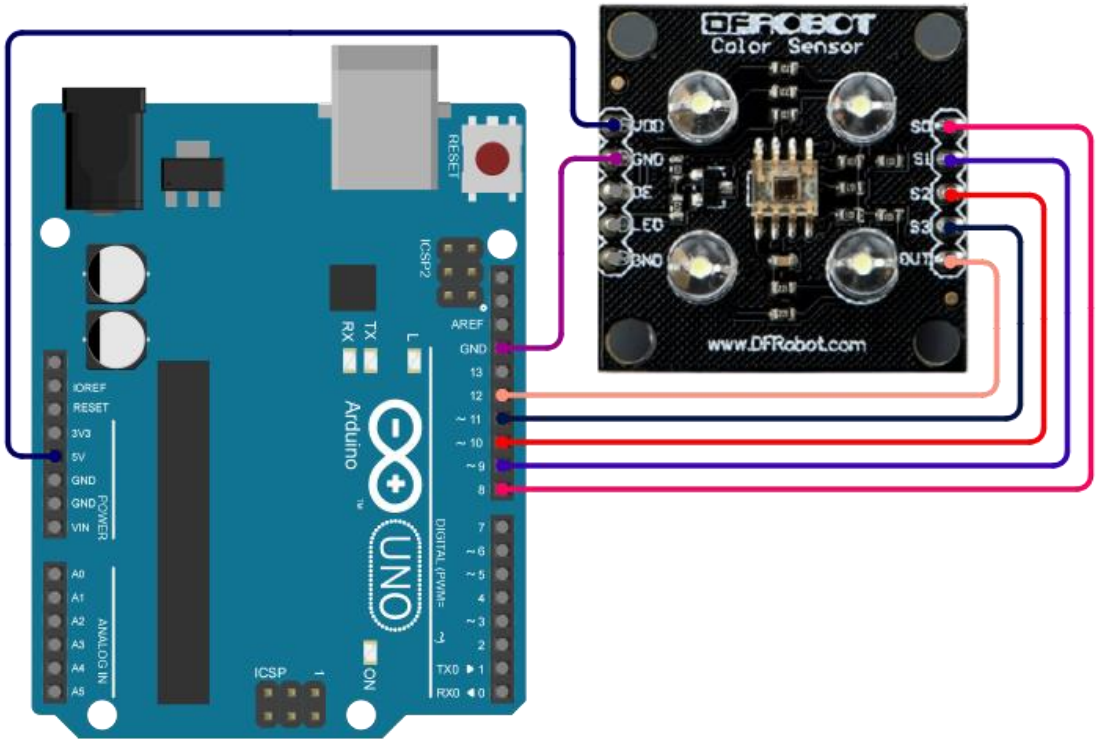
WORKING PRINCIPLE OF COLOUR RECOGNITION SENSOR:

The TCS230 Color sensor (also branded as the TCS3200) is quite popular, inexpensive and easy to use. At the heart of the module is an inexpensive RGB sensor chip from Texas Advanced Optoelectronic Solutions – TCS230. The TCS230 Color Sensor is a complete colour detector that can detect and measure an almost infinite range of visible colours.

The sensor itself can be seen at the centre of the module, surrounded by the four white LEDs. The LEDs light up when the module is powered up and are used to illuminate the object being sensed. Due to these LEDs, the sensor can also work in complete darkness to determine the colour or brightness of the object. The TCS230 operates on a supply voltage of 2.7 to 5.5 volts and provides TTL logic- level outputs.

The TCS230 detects colour with the help of an 8 x 8 array of photodiodes, of which sixteen photodiodes have red filters, 16 photodiodes have green filters, 16 photodiodes have blue filters, and remaining 16 photodiodes are clear with no filters.

CONNECTION DIAGRAM:



CONNECTIONS:

- **TCS3200 Sensor to Arduino:**
  - VCC to 5V
  - GND to GND
  - S0 to pin 8
  - S1 to pin 9
  - S2 to pin 10
  - S3 to pin 11
  - OUT to pin 12

PROGRAM:

```
const int s0 = 8, s1 = 9, s2 = 10, s3 = 11, out = 12;

void setup() {
  pinMode(s0, OUTPUT); pinMode(s1, OUTPUT);
  pinMode(s2, OUTPUT); pinMode(s3, OUTPUT);
  pinMode(out, INPUT);
  Serial.begin(9600);
  digitalWrite(s0, HIGH); digitalWrite(s1, HIGH);
  Serial.println("Color Recognition Ready");
}

void loop() {
  digitalWrite(s2, LOW); digitalWrite(s3, LOW);
  Serial.print("Red = "); readColor();
  digitalWrite(s2, LOW); digitalWrite(s3, HIGH);
```

```
  Serial.print("Blue = "); readColor();
  digitalWrite(s2, HIGH); digitalWrite(s3, HIGH);
  Serial.print("Green = "); readColor();
  Serial.println(); delay(2000);
}

void readColor() {
  long d = pulseIn(out, LOW, 30000);
  if (d == 0) Serial.print("NoSignal");
  else Serial.print(d);
  Serial.print("\t"); delay(1000);
}
```

OUTPUT:

```
Color Recognition Ready
Red = 1250 Blue = 980      Green = 1100
Red = 1250 Blue = 980      Green = 1100
Red = 600  Blue = 1500     Green = 700
Red = 600  Blue = 1500     Green = 700
```

RESULT:

Thus, the Colour Recognition Sensor has been interfaced with Arduino successfully.

Ex. No: 06 **FIRE ALARM INDICATOR USING ARDUINO**
Date:

AIM:

To develop a Fire Alarm Indicator using the Flame sensor.

LIST OF COMPONENTS:

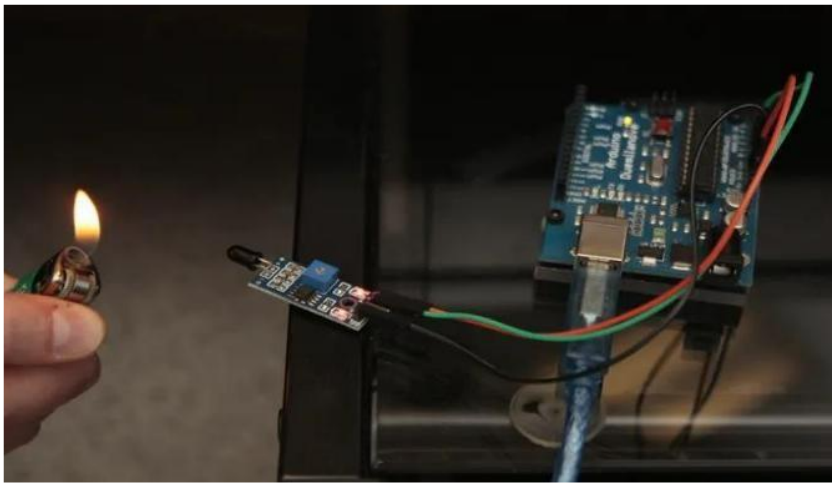
1. Arduino Uno
2. Flame Sensor (Analog)
3. Solderless Breadboard
4. Buzzer
5. 3 x LED
6. 3 x 220Ω Resistors
7. Jumper Wires

WORKING DESCRIPTION:

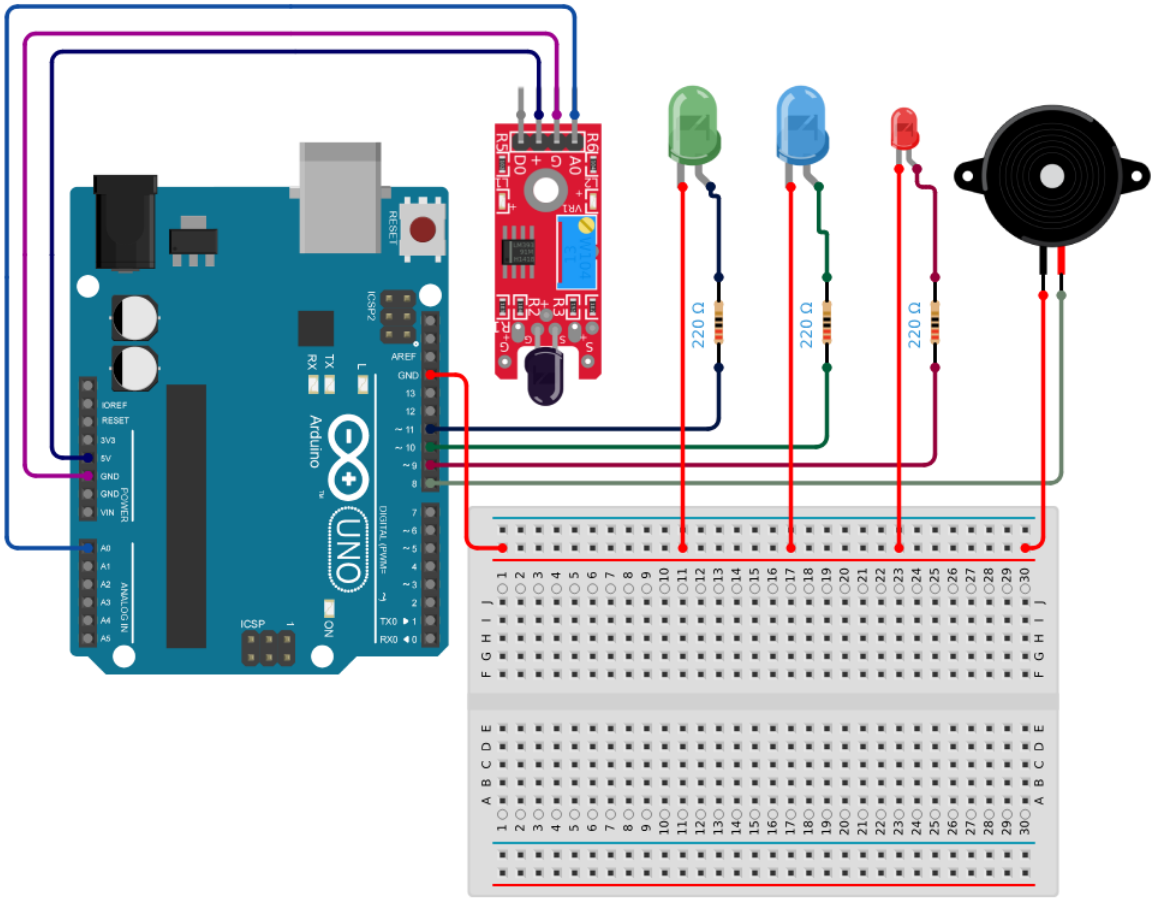
A flame-sensor is one kind of detector which is mainly designed for detecting as well as responding to the occurrence of a fire or flame. The flame detection response can depend on its fitting. It includes an alarm system, a natural gas line, propane & a fire suppression system. This sensor is used in industrial boilers. The main function of this is to give authentication whether the boiler is properly working or not. The response of these sensors is faster as well as more accurate compare with a heat/smoke detector because of its mechanism while detecting the flame.

WORKING PRINCIPLE:

This sensor/detector can be built with an electronic circuit using a receiver like electromagnetic radiation. This sensor uses the infrared flame flash method, which allows the sensor to work through a coating of oil, dust, water vapor, otherwise ice.



CONNECTION DIAGRAM:



CONNECTIONS:

- **Flame Sensor to Arduino:**
  - VCC to 5V
  - GND to GND
  - AO to A0
- **Close Fire LED to Arduino:**
  - Negative to GND
  - Positive to pin 9 and 220Ω Resistor
- **Distant Fire LED to Arduino:**
  - Negative to GND
  - Positive to pin 10 and 220Ω Resistor
- **No Fire LED to Arduino:**
  - Negative to GND
  - Positive to pin 11 and 220Ω Resistor
- **Buzzer to Arduino:**
  - Negative to GND
  - Positive to pin 8

PROGRAM:

```
const int buzzer = 8;
const int ledClose = 9;
```

```
const int ledDistant = 10;
const int ledNone = 11;

void setup() {
  Serial.begin(9600);
  pinMode(buzzer, OUTPUT);
  pinMode(ledClose, OUTPUT);
  pinMode(ledDistant, OUTPUT);
  pinMode(ledNone, OUTPUT);
  Serial.println("Fire Alarm Ready");
}

void loop() {
  int flame = analogRead(A0);

  digitalWrite(ledClose, LOW);
  digitalWrite(ledDistant, LOW);
  digitalWrite(ledNone, LOW);
  noTone(buzzer);

  if (flame < 200) {
    Serial.println("Close Fire");
    digitalWrite(ledClose, HIGH);
    tone(buzzer, 1000);
    delay(500);
    noTone(buzzer);
  }
  else if (flame < 500) {
    Serial.println("** Distant Fire **");
    digitalWrite(ledDistant, HIGH);
  }
  else {
    Serial.println("No Fire");
    digitalWrite(ledNone, HIGH);
  }

  delay(500);
}
```

OUTPUT:

```
Fire Alarm Ready
No Fire
No Fire
** Distant Fire **
Close Fire
```

RESULT:

Thus, the Fire Alarm Indicator was successfully implemented using Flame Sensor.

Ex. No: 07 **SOUND DETECTION USING ARDUINO**
Date:

AIM:

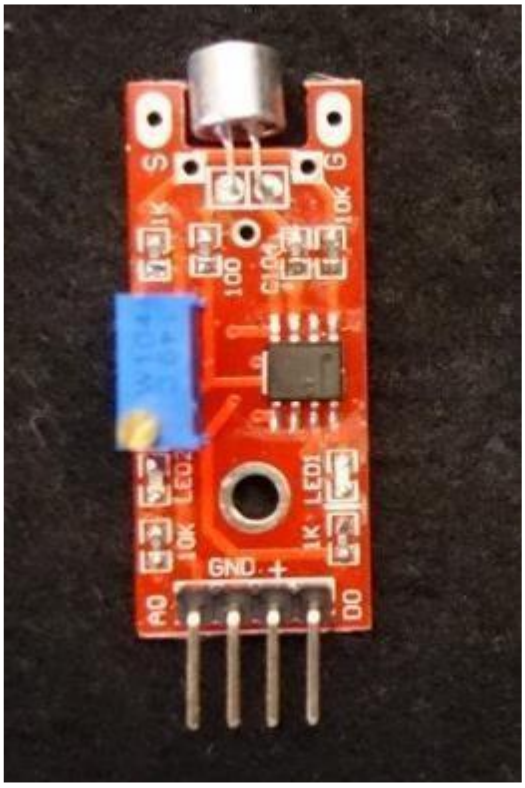
To develop a Sound Detection System using the Sound Sensor.

LIST OF COMPONENTS:

1. Arduino Uno
2. KY-038 Sound Sensor
3. LED
4. 220Ω Resistor
5. Jumper Wires

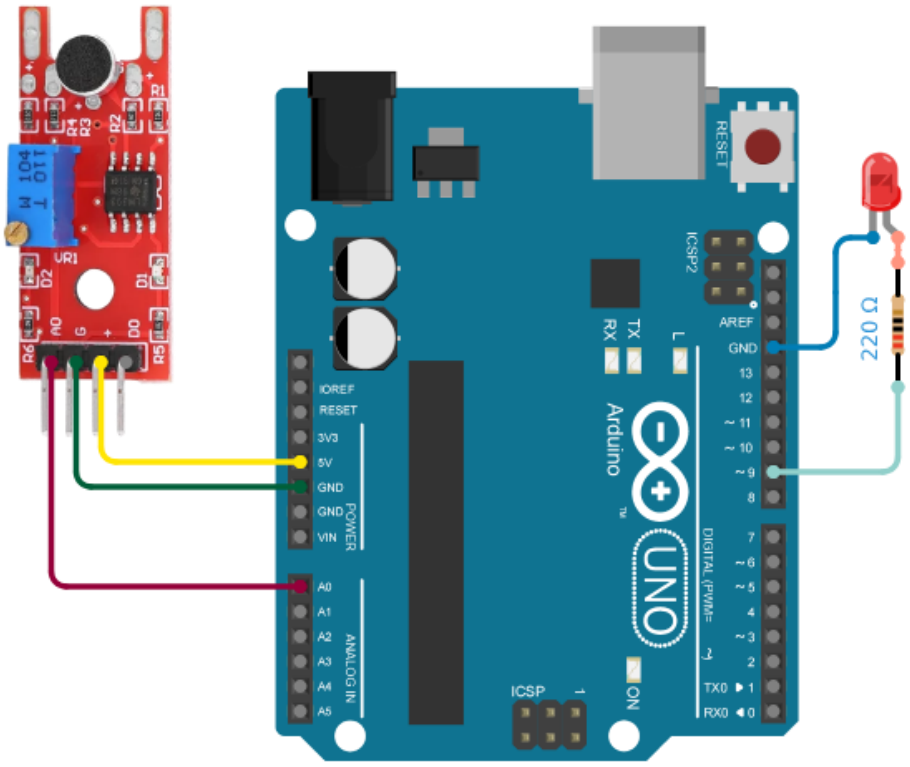
WORKING DESCRIPTION:

The microphone sound sensor, as the name says, detects sound. It gives a measurement of how loud a sound is. There are a wide variety of these sensors. In the figure below you can see the most common used with the Arduino.



In this example, a microphone sensor will detect the sound intensity of your surroundings and will light up an LED if the sound intensity is above a certain threshold.

CONNECTION DIAGRAM:



CONNECTIONS:

- **Sound Sensor to Arduino:**
  - VCC to 5V
  - GND to GND
  - AO to A0
- **LED to Arduino:**
  - Negative to GND
  - Positive to pin 9 and 220Ω Resistor

PROGRAM:

```
const int soundPin = A0;
const int ledPin = 9;
const int threshold = 600;

void setup() {
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
  Serial.println("Sound Detection Ready");
}

void loop() {
  int value = analogRead(soundPin);
  Serial.print("Sound Level: "); Serial.println(value);

  if (value > threshold) {
```

```
    digitalWrite(ledPin, HIGH);
    delay(200);
  } else {
    digitalWrite(ledPin, LOW);
  }
  delay(100);
}
```

OUTPUT:

```
     Sound Detection Ready
     Sound Level: 320
     Sound Level: 450
     Sound Level: 330
     Sound Level: 500
     Sound Level: 850  ← Clap!
     Sound Level: 600
     Sound Level: 320
```

RESULT:

Thus, the Sound Detection System using Arduino was implemented successfully.

Ex. No: 08 **INTERFACING FLEX SENSOR WITH ARDUINO**
Date:

AIM:

To interface a Flex Sensor with Arduino board.

LIST OF COMPONENTS:

1. Arduino Uno
2. Flex Sensor
3. LED
4. 10kΩ Resistor
5. 220Ω Resistor
6. Jumper Wires

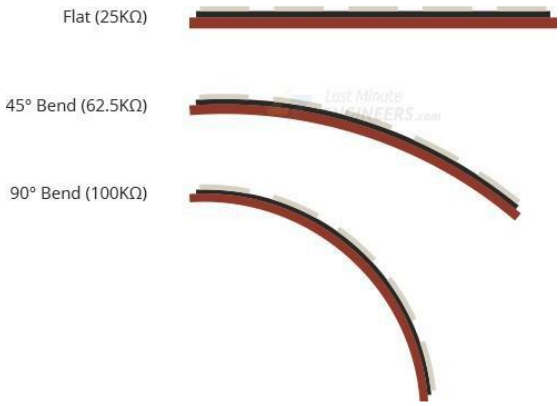
WORKING DESCRIPTION:

A flex sensor is a kind of sensor which is used to measure the amount of defection otherwise bending. The carbon surface is arranged on a plastic strip as this strip is turned aside then the sensor's resistance will be changed. Thus, it is also named a bend sensor.

A flex-sensor could be used to check a door or window is opened or not. This sensor can be arranged at the edge of the door and once the door opens then this sensor also gets flexed. When the sensor bends then its parameters automatically change which can be designed to give an alert.

This sensor works on the bending strip principle which means whenever the strip is twisted then its resistance will be changed. This can be measured with the help of any controller. This sensor works similar to a variable resistance because when it twists then the resistance will be changed. The resistance change can depend on the linearity of the surface because the resistance will be dissimilar when it is level.

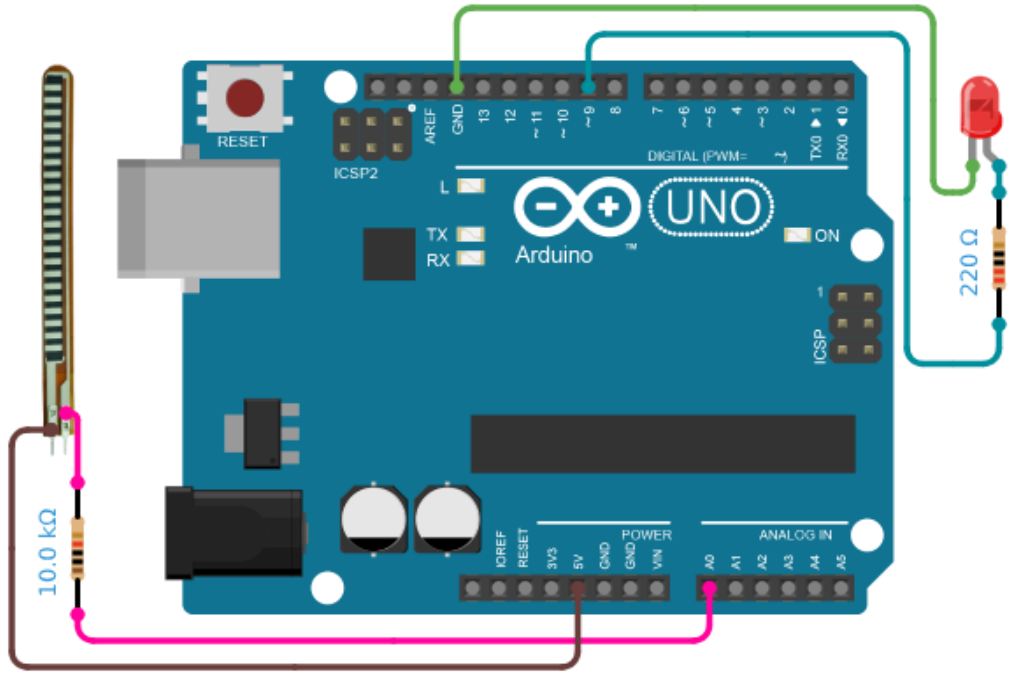
Flex sensors are designed to flex in only one direction – away from ink (as shown in the figure). Bending the sensor in another direction may damage it.



When the sensor is straight, this resistance is about 25k. When the sensor is bent, conductive layer is stretched, resulting in reduced cross section (imagine stretching a rubber band). This reduced cross section results in an increased resistance. At 90° angle, this resistance is about 100KΩ.

When the sensor is straightened again, the resistance returns to its original value. By measuring the resistance, we can determine how much the sensor is bent.

CONNECTION DIAGRAM:



CONNECTIONS:

- **Flex Sensor to Arduino:**
  - Pin 1 to 5V
  - Pin 2 to A0 and 10kΩ Resistor
- **10kΩ Resistor:**
  - One end to A0 and Flex Pin 2
  - Other end to GND
- **LED to Arduino:**
  - Negative to GND
  - Positive to pin 9 and 220Ω Resistor

PROGRAM:

```
const int flexPin = A0;
const int ledPin = 9;

void setup() {
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
  Serial.println("Flex Sensor Ready");
}

void loop() {
  int value = analogRead(flexPin);
  Serial.print("Flex Value: "); Serial.println(value);

  if (value > 890) {
    digitalWrite(ledPin, HIGH);
  } else {
```

```
    digitalWrite(ledPin, LOW);
  }
  delay(100);
}
```

OUTPUT:

```
Flex Sensor Ready
Flex Value: 520  ← Straight
Flex Value: 621
Flex Value: 679
Flex Value: 758
Flex Value: 865
Flex Value: 920  ← Bent → LED ON
Flex Value: 689
Flex Value: 564
```

RESULT:

Thus, the Flex Sensor was interfaced with Arduino successfully.

Ex. No: 09 **INTERFACING FORCE PRESSURE SENSOR WITH ARDUINO**
Date:

AIM:

To interface a force pressure sensor with Arduino board.

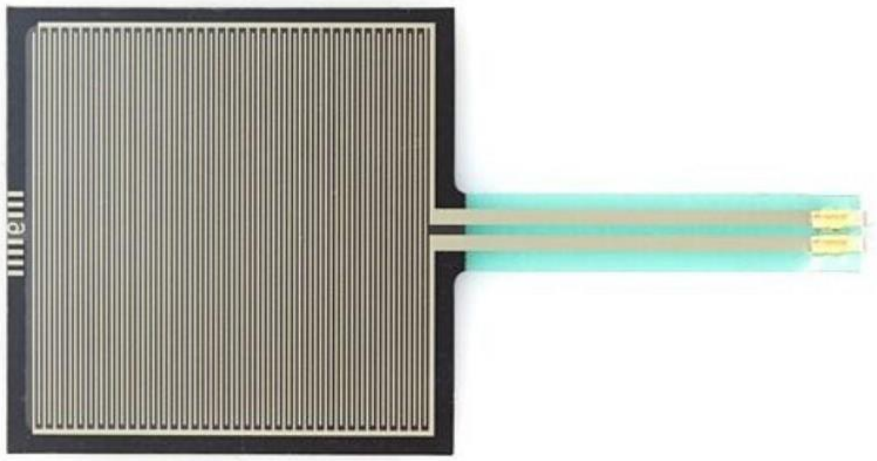
LIST OF COMPONENTS:

1. Arduino Uno
2. FSR
3. 10kΩ Resistor
4. Jumper Wires

WORKING DESCRIPTION:

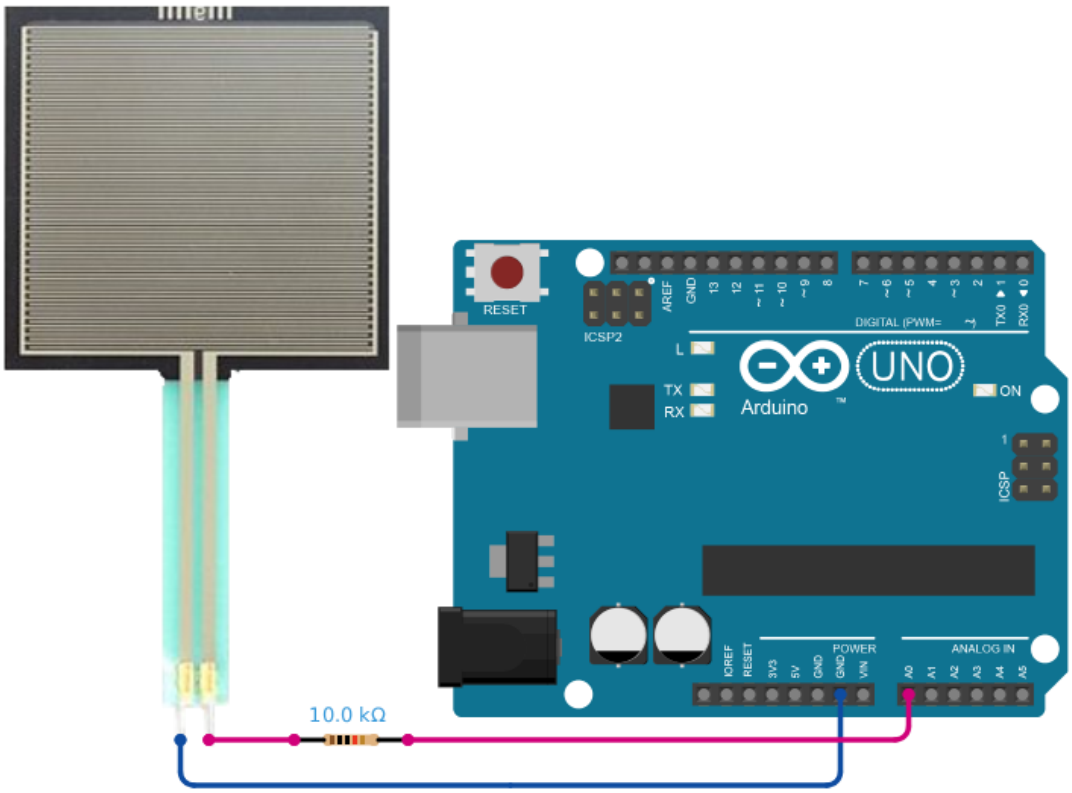
The working principle of a force sensor is that it responds to the applied force, as well as converts the value to a measurable quantity. Most force sensors are created with the use of force-sensing resistors. Such sensors consist of electrodes and sensing film.

Force-sensing resistors are based on contact resistance. These contain a conductive polymer film, which changes its resistance in a predictable way once force is applied on the surface.



In road traffic, force measurement plays an essential role. For instance, force sensors are being used in trucks. This means that the axle load may be determined precisely to enable effective and fast monitoring. Different force sensors are in use in automobiles. For instance, force sensors in the area of trailer couplings offer the possibility to know the trailer's load, and to determine static information in relation to dynamic driving behavior on the road.

CONNECTION DIAGRAM:



CONNECTIONS:

- **FSR to Arduino:**
  - Pin 1 to 5V
  - Pin 2 to A0 and 10kΩ resistor
- **10kΩ Resistor:**
  - One end to A0 and FSR Pin 2
  - Other end to GND

PROGRAM:

```
const int fsrPin = A0;

void setup() {
  Serial.begin(9600);
  Serial.println("Force Sensor Ready");
}

void loop() {
  int reading = analogRead(fsrPin);
  Serial.print("Force Reading: "); Serial.print(reading);

  if (reading < 10) Serial.println(" → No Pressure");
  else if (reading < 200) Serial.println(" → Light Touch");
  else if (reading < 500) Serial.println(" → Light Pressure");
  else if (reading < 800) Serial.println(" → Medium Pressure");
  else Serial.println(" → Heavy Pressure");
```

```
  delay(200);
}
```

OUTPUT:

```
Force Sensor Ready
Force Reading: 5 → No Pressure
Force Reading: 150 → Light Touch
Force Reading: 600 → Medium Pressure
Force Reading: 950 → Heavy Pressure
```

RESULT:

Thus, the Force Pressure Sensor was interfaced with Arduino successfully.

Ex. No: 10 IDENTIFYING ROOM TEMPERATURE AND HUMIDITY USING RASPBERRY PI
Date:

AIM:

To identify the Room Temperature and Humidity using Raspberry Pi.

LIST OF COMPONENTS:

1. Raspberry Pi
2. DHT 11 Sensor
3. Jumper Wire
4. Thingspeak cloud

WORKING DESCRIPTION:

1. Here, we are going to interface DHT11 sensor with Raspberry Pi 3 and display Humidity and Temperature on terminal.
2. We will be using the DHT Sensor Python library by Adafruit from GitHub. The Adafruit Python DHT Sensor library is created to read the Humidity and Temperature on Raspberry Pi or Beaglebone Black. It is developed for DHT series sensors like DHT11, DHT22 or AM2302.
3. Extract the library and install it in the same root directory of downloaded library by executing following command.

```
sudo python setup.py install
```

4. Once the library and its dependencies has been installed, open the example sketch named simple test from the library kept in examples folder.
5. In this code, raspberry Pi reads Humidity and Temperature from DHT11 sensor and prints them on terminal. But, it read and display the value only once. So, the program is developed in such a way to print value continuously.

Notes:

Assign proper sensor type to the sensor variable in this library. Here, we are using DHT11 sensor.

```
sensor = Adafruit_DHT.DHT11
```

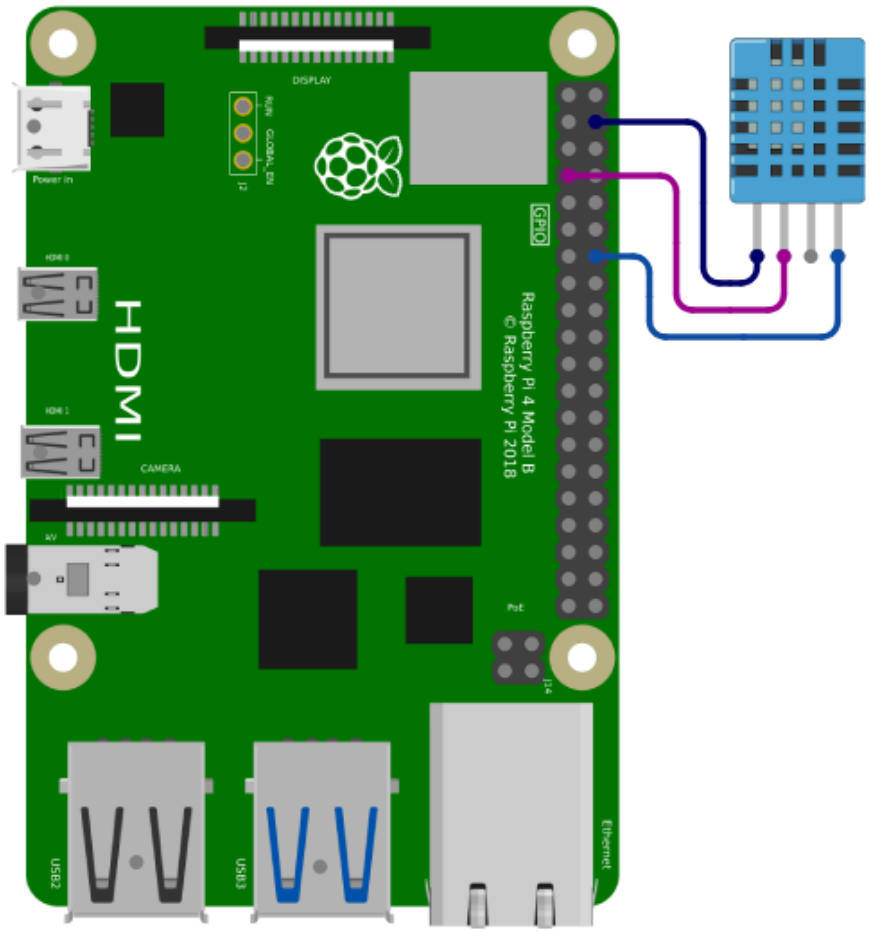
6. If anyone is using sensor DHT22 then we need to assign Adafruit_DHT.DHT22 to the sensor variable shown above.
7. Also, comment out Beaglebone pin definition and uncomment pin declaration for Raspberry Pi.
8. Then assign pin no. to which DHT sensor's data pin is connected. Here, data out of DHT11 sensor is connected to GPIO4.
9. Next these commands have to be executed in cmd.

```
sudo apt-get update
sudo apt-get install build-essential python-dev python-openssl git
git clone https://github.com/adafruit/Adafruit_Python_DHT.git
cd Adafruit_Python_DHT
sudo python setup.py install.
```

10. Using Raspberry Pi ThingSpreak Library
11. In order to be able to use the service, it is possible to simply send the data via "POST" or retrieve via "GET". Functions are available in just about any programming language and with a little bit of knowledge, data transfer should be fast. The answers are in principle in JSON.
12. Alternative command

```
sudo pip install thingspeak
```

CONNECTION DIAGRAM:



CONNECTIONS:

- **DHT11 sensor to Raspberry Pi:**
  - VCC (Pin 1) to Pin 4 (5V)
  - DATA (Pin 2) to Pin 7 (GPIO4)
  - GND (Pin 4) to GND

PROGRAM:

```
import Adafruit_DHT
import time

SENSOR = Adafruit_DHT.DHT11
PIN = 4  # GPIO4

while True:
    humidity, temperature = Adafruit_DHT.read(SENSOR, PIN)
    if humidity is not None and temperature is not None:
        print('Temp={0:0.1f}°C  Humidity={1:0.1f}%'.format(temperature, humidity))
    else:
        print('Failed to get reading. Try again!')
    time.sleep(2)
```

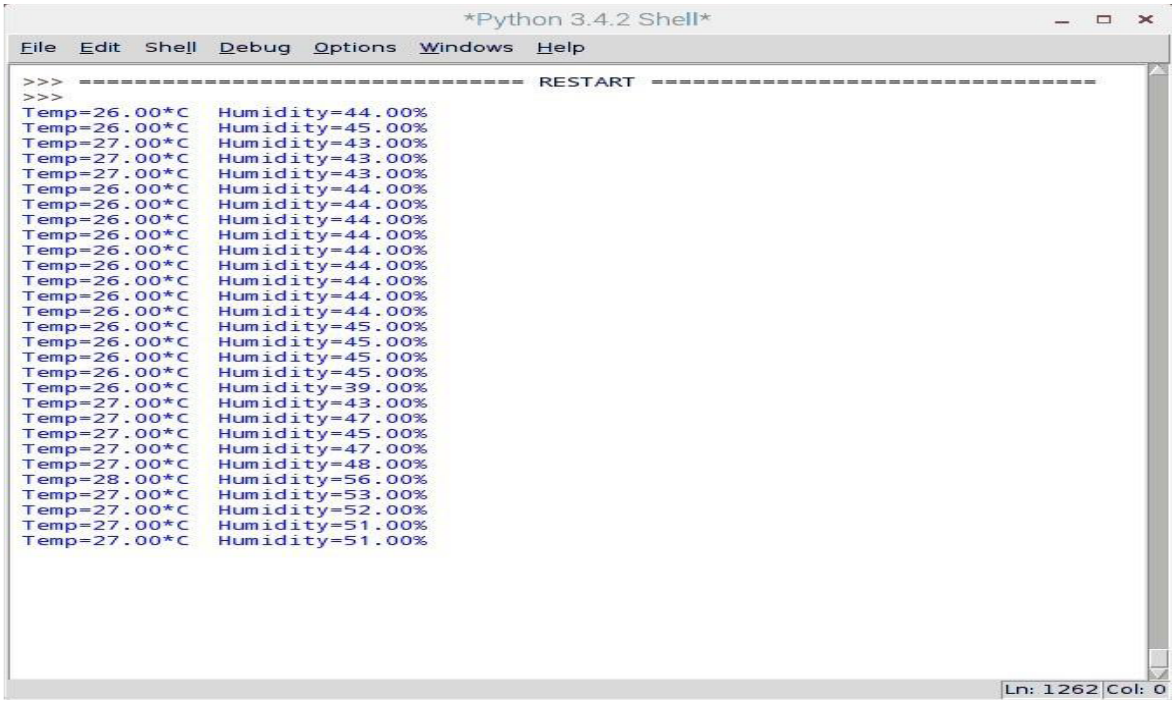
```
import time
import Adafruit_DHT
import requests

channel_id = 1521416
write_key = 'WND956XF9P9OX5IS'
read_key = 'YSSKCVIKZDZRHIYS'
PIN = 4
SENSOR = Adafruit_DHT.DHT11

def measure(channel):
      try:
            humidity, temperature = Adafruit_DHT.read_retry(SENSOR, PIN)
            response = channel.update({'field1': temperature, 'field2':
                                                humidity})
            read = channel.get({})
            print("Read:", read)
      except:
            print("connection failed")

if __name__ == "__main__":
      while True:
            measure(channel)
            time.sleep(15)
```

OUTPUT:



RESULT:

Thus, the Room temperature and Humidity was measured using Raspberry Pi successfully.

Ex. No: 11 **PIR MOTION SENSOR INTERFACING WITH RASPBERRY PI**
Date:

AIM:

To build Motion Detection System using PIR Sensor.

LIST OF COMPONENTS:

1. Raspberry Pi
2. PIR Motion Sensor (HC-SR501)
3. Jumper Wire

WORKING DESCRIPTION:

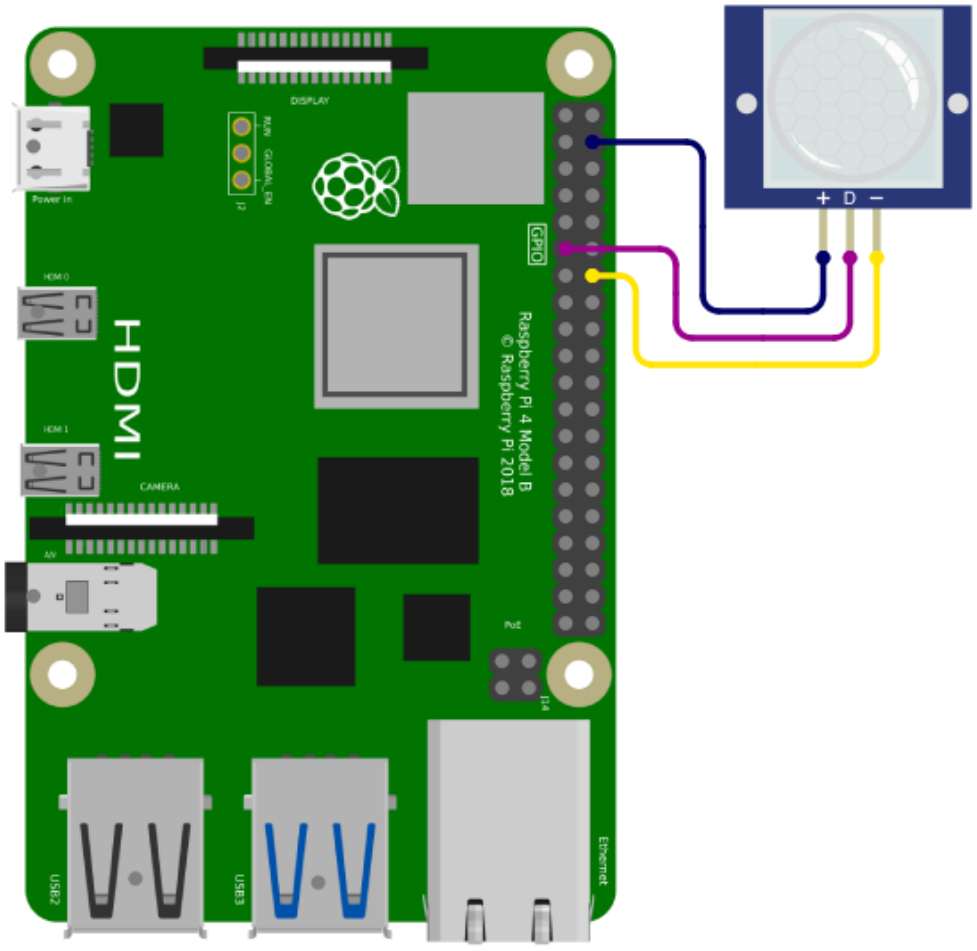
PIR sensor is used for detecting infrared heat radiations. This makes them useful in the detection of moving living objects that emit infrared heat radiations. The output (in terms of voltage) of PIR sensor is high when it senses motion; whereas it is low when there is no motion (stationary object or no object).

PIR sensors are used in many applications like for room light control using human detection, human motion detection for security purpose at home, etc.

Setup:

When motion is detected, PIR output goes HIGH which will be read by Raspberry Pi. So, we will turn on LED when motion is detected by PIR sensor. Here, LED is connected to GPIO12 (pin no. 32) whereas PIR output is connected to GPIO5 (pin no. 29).

CONNECTION DIAGRAM:



CONNECTIONS:

- **PIR sensor to Raspberry Pi:**
  - VCC (Pin 1) to Pin 4 (5V)
  - DATA (Pin 2) to Pin 11 (GPIO 17)
  - GND (Pin 4) to Pin 14 GND

PROGRAM:

```
import time
import RPi.GPIO as GPIO

PIR_pin = 17

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(PIR_pin, GPIO.IN)

print("Motion Detection Started")

try:
    while True:
        if GPIO.input(PIR_pin):
            print("Motion detected!")
        else:  # If no motion (LOW signal)
            print("No motion detected.")
        time.sleep(1)

except KeyboardInterrupt:
    print("Program interrupted by user.")

finally:
    GPIO.cleanup()
```

OUTPUT:

```
Motion Detection Started
No motion detected.
Motion detected!
Motion detected!
No motion detected.
```

RESULT:

Thus, the Motion Detector was built successfully using PIR sensor.

Ex. No: 12 **MEASURING SOIL MOISTURE WITH RASPBERRY PI**
Date:

AIM:

To identify the Moisture content of soil in Agricultural Land using Soil Moisture Sensor.

LIST OF COMPONENTS:

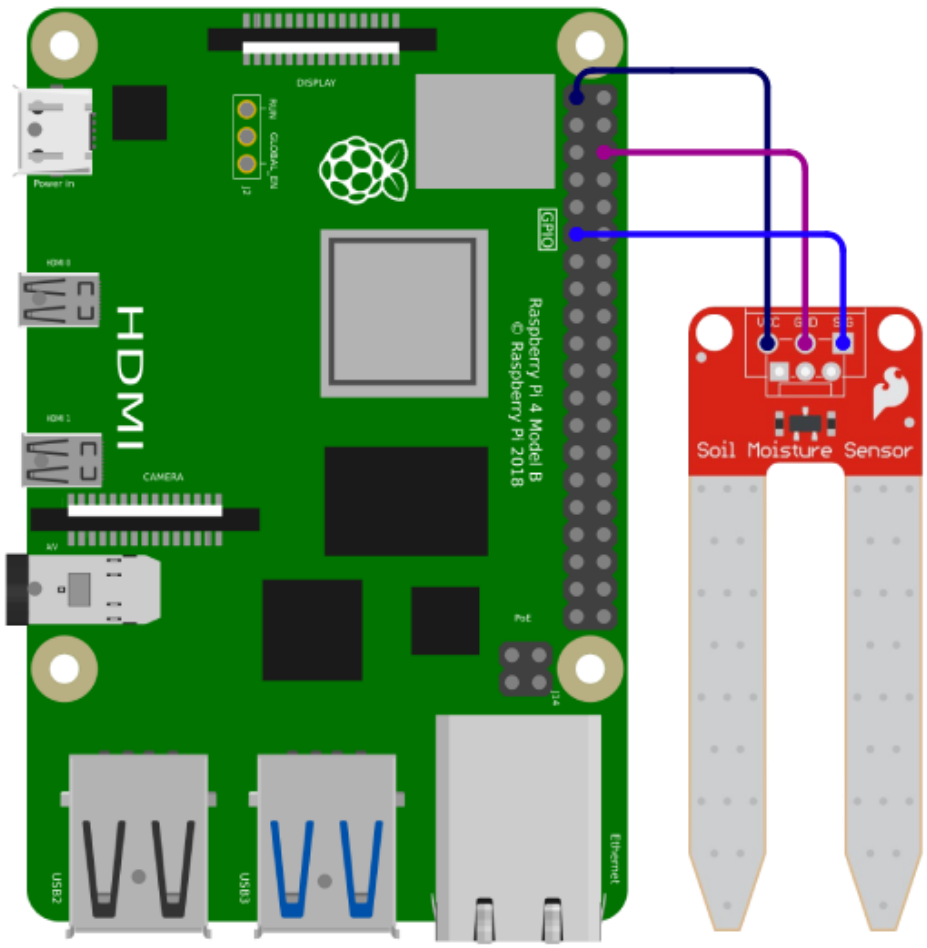
1. Raspberry Pi
2. Soil Moisture Sensor (Analog)
3. Jumper Wire

WORKING DESCRIPTION:

The soil moisture sensor is used to detect the presence of water content in the soil. It consists of two probes that measure the electrical resistance of the soil. When the soil is wet, it conducts electricity better, resulting in a LOW digital output signal. When the soil is dry, the sensor outputs a HIGH digital signal.

In this setup, the sensor's digital output pin is connected to the Raspberry Pi's GPIO17 pin. The Raspberry Pi continuously reads this digital signal. When the input is LOW, it indicates the soil is wet, and the program prints "Soil is wet." When the input is HIGH, it means the soil is dry, and the program prints "Soil is dry." This simple method allows for real-time monitoring of soil moisture content, which is crucial for agricultural applications to optimize irrigation and prevent overwatering or drought stress in plants.

CONNECTION DIAGRAM:



CONNECTIONS:

- **Soil Moisture Sensor to Raspberry Pi:**
  - VCC (Pin 1) to Pin 1 (3.3V)
  - DATA (Pin 2) to Pin 11 (GPIO 17)
  - GND (Pin 4) to Pin 6 GND

PROGRAM:

```
import RPi.GPIO as GPIO
import time

SENSOR_PIN = 17  # GPIO17 (Pin 11)

GPIO.setmode(GPIO.BCM)
GPIO.setup(SENSOR_PIN, GPIO.IN)

try:
    while True:
        if GPIO.input(SENSOR_PIN) == 0:
            print("Soil is wet")
        else:
            print("Soil is dry")
        time.sleep(1)
except KeyboardInterrupt:
    GPIO.cleanup()
```

OUTPUT:

```
Soil is wet
Soil is wet
Soil is dry
Soil is dry
Soil is wet
```

RESULT:

Thus, the Moisture content in Agricultural land was measured using Raspberry Pi successfully.

Ex. No: 13 **SOUND SENSOR INTERFACING WITH RASPBERRY PI**
Date:

AIM:

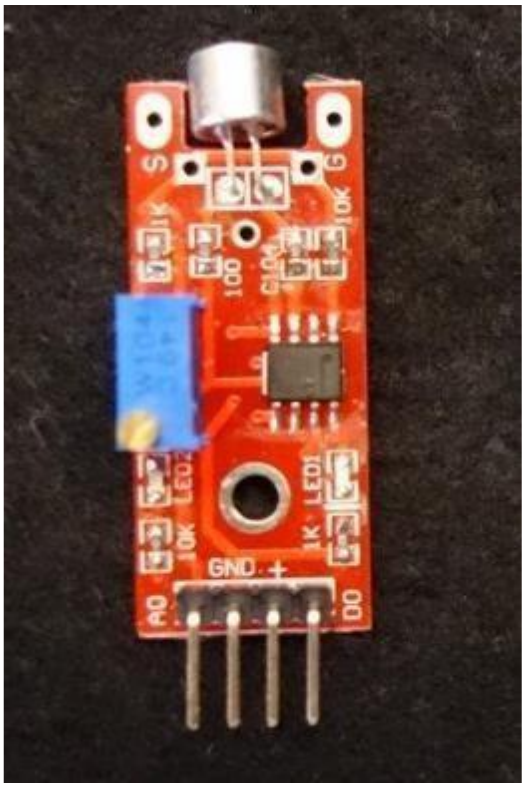
To Interfacing Sound Sensor with Raspberry Pi.

LIST OF COMPONENTS:

1. Raspberry Pi
2. Sound Sensor
3. LED
4. Jumper Wire

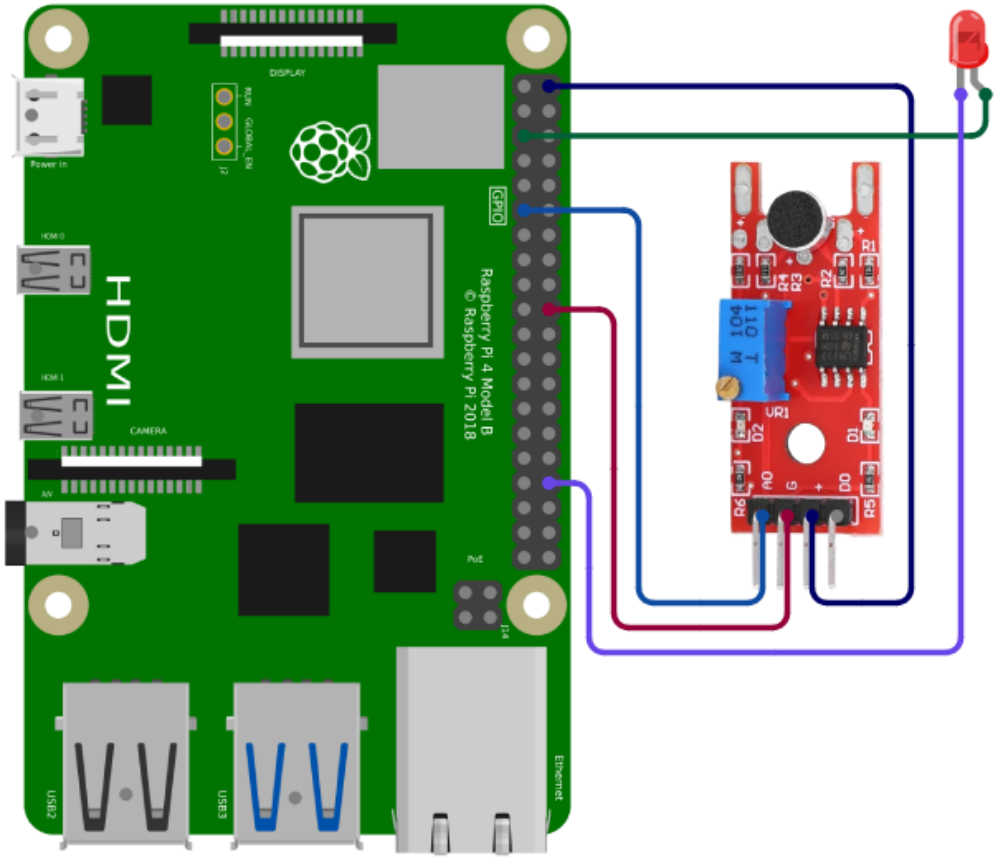
WORKING DESCRIPTION:

The microphone sound sensor, as the name says, detects sound. It gives a measurement of how loud a sound is. There are a wide variety of these sensors. In the figure below you can see the most common used with the Arduino.



In this example, a microphone sensor will detect the sound intensity of your surroundings and will light up an LED if the sound intensity is above a certain threshold.

CONNECTION DIAGRAM:



CONNECTIONS:

- **Sound Sensor to Raspberry Pi:**
  - VCC to Pin 2 (5V)
  - DATA to Pin 11 (GPIO 17)
  - GND to Pin 20 GND
- **Red LED to Arduino:**
  - Negative to GND
  - Positive to Pin 5 (GPIO 3)

PROGRAM:

```
from time import sleep
import RPi.GPIO as GPIO

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)

SOUND_PIN = 11
LED_PIN = 3

GPIO.setup(SOUND_PIN, GPIO.IN)
GPIO.setup(LED_PIN, GPIO.OUT)
```

```
print("Sound sensor test is running...")

while True:
    if GPIO.input(SOUND_PIN) == True:
        GPIO.output(LED_PIN, False)
        print("No sound detected")
    else:
        GPIO.output(LED_PIN, True)
        print("Sound detected!")

    sleep(1)
```

OUTPUT:

```
Sound sensor test is running...
No sound detected
No sound detected
Sound detected!
Sound detected!
No sound detected
```

RESULT:

Thus, the Sound Sensor interfacing using Raspberry Pi successfully.

Ex. No: 14 **ANALOG SENSOR WITH ESP32 MICROCONTROLLER**
Date:

AIM:

To Interface an analog sensor (MQ135) with an ESP32 Microcontroller.

LIST OF COMPONENTS:

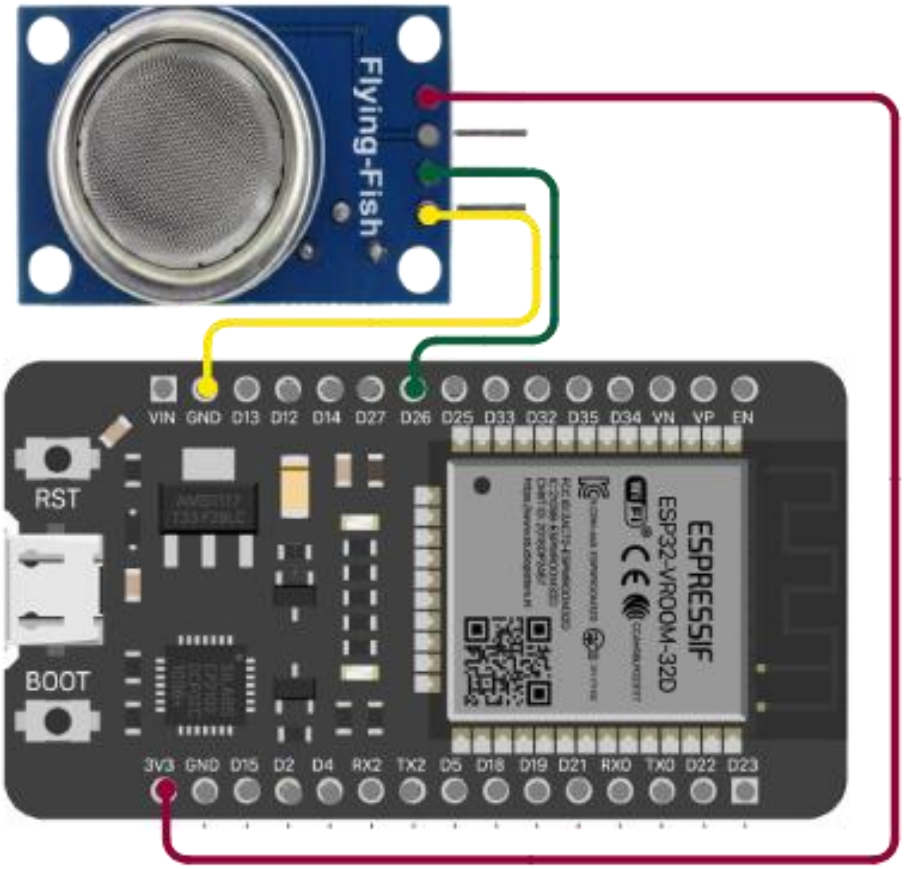
1. ESP32 Microcontroller
2. MQ135 Gas Sensor
3. Jumper wires

WORKING DESCRIPTION:

Sensor (MQ135) Working Description the MQ135 is an analog gas sensor that detects air quality or the presence of certain gases in the air.

1. The MQ135 sensor is connected to GPIO pin 26 (analog input) of the ESP32.
2. The analogRead(26) function reads the analog output of the MQ135 sensor.
3. The sensor's analog output varies based on the concentration of gases detected.
4. The code prints this analog value to the Serial Monitor for monitoring or further processing.

CONNECTION DIAGRAM:



CONNECTIONS:

- **MQ135 Gas Sensor to ESP32:**
  - VCC to (3.3V)
  - AOUT to Pin D26
  - GND to Pin GND

PROGRAM:

```
void setup() {
  Serial.begin(9600);
  pinMode(26, INPUT);
}

void loop() {
  int sensorValue = analogRead(26);
  Serial.println(sensorValue);
  delay(1);
}
```

OUTPUT:

```
      2105
      2150
      2203
      2287
      2340
      2410
```

RESULT:

Thus, the Analog Sensor show analog value from the ESP32 using successfully.

Ex. No: 15 **DHT11 SENSOR WITH ESP 32**
Date:

AIM:

To interface DHT11 with ESP32 to read Temperature and Humidity.

LIST OF COMPONENTS:

1. ESP32 Microcontroller
2. DHT11 Sensor
3. Jumper Wire

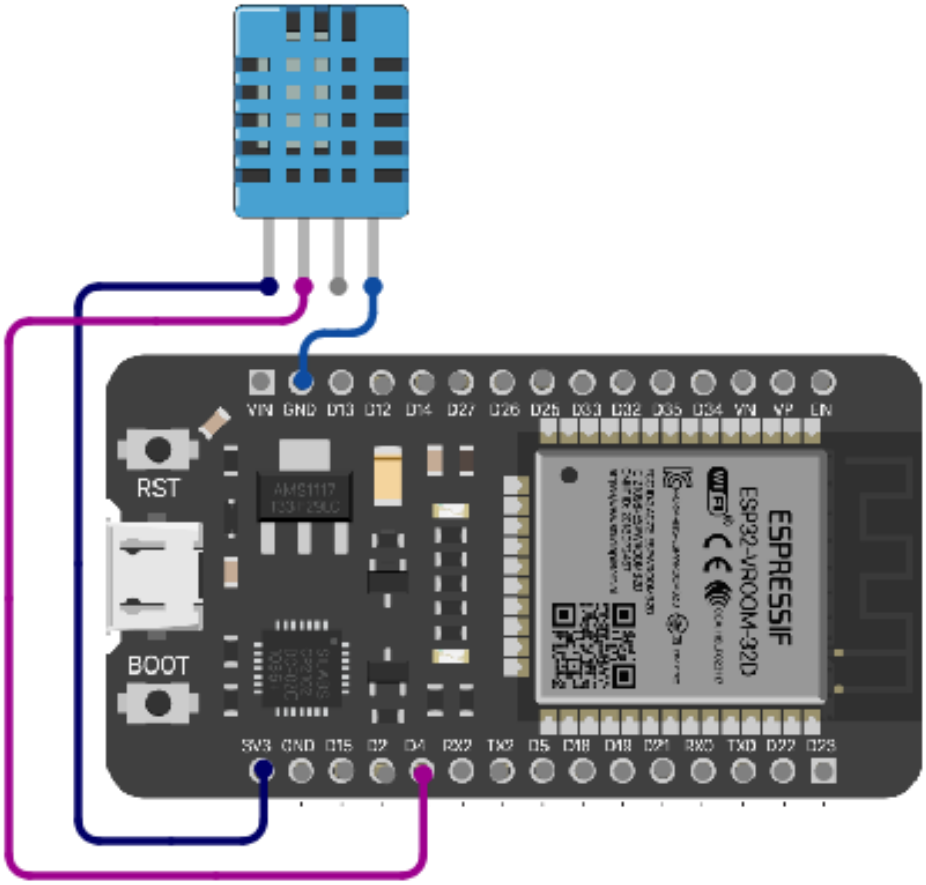
WORKING DESCRIPTION:

The DHT11 sensor measures temperature and humidity. When connected to an ESP32, it sends digital signals representing the temperature and humidity values. The ESP32 reads these signals and prints the values to the Serial Monitor.

1. The DHT11 sensor captures temperature and humidity data.
2. The ESP32 reads the digital signal from the DHT11.
3. The ESP32 processes the data and prints it to the Serial Monitor.

This setup allows for monitoring environmental conditions using the DHT11 sensor with the ESP32.

CONNECTION DIAGRAM:



CONNECTIONS:

- **DHT11 Sensor to ESP32:**
  - VCC to (3.3V)
  - DATA to Pin D4
  - GND to Pin GND

PROGRAM:

```
#include "DHT.h"

#define DHT11PIN 4
DHT dht(DHT11PIN, DHT11);

void setup() {
  Serial.begin(115200);
  dht.begin();
}

void loop() {
  float humi = dht.readHumidity();
  float temp = dht.readTemperature();

  // Print the readings
  Serial.print("Temperature: ");
  Serial.print(temp);
  Serial.println(" ºC");

  Serial.print("Humidity: ");
  Serial.print(humi);
  Serial.println(" %");

  delay(1000);
}
```

OUTPUT:

```
Temperature: 27.4 ºC
Humidity: 58.0 %

Temperature: 27.6 ºC
Humidity: 57.8 %

Temperature: 27.5 ºC
Humidity: 58.1 %
```

RESULT:

Thus, the Temperature and Humidity readings using DHT11 from the ESP32 using successfully.