ANNAMALAI UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

B.E COMPUTER SCIENCE & ENGINEERING

B.E CSE (DATA SCIENCE)

SEMESTER – VII

DSPC706 - INTERNET OF THINGS (IoT) LAB

LABORATORY RECORD

(JULY 2022 – DECEMBER 2022)

Name :..

Reg. No :..

ANNAMALAI UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

B.E COMPUTER SCIENCE & ENGINEERING

B.E CSE (DATA SCIENCE)

SEMESTER – VII

DSPC706 - INTERNET OF THINGS (IoT) LAB

Bonafide Certificate

Certified that this is the Bonafide Record of work done by

Mr./Ms. ____

Reg. No. ____________________ *of VII semester B.E. Computer Science and Engineering (Data Science) in the* ***DSPC706 – Internet Of Things(IOT)*** *during the odd semester(July 2022 – December 2022).*

Staff In -Charge

Internal Examiner External Examiner

Place : Annamalai Nagar

Date :

Vision and Mission of the Department

VISION

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

MISION

M1: 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.

M2: Develop life-long learning ability that allows them to be adaptive and responsive to the changes in career, society, technology, and environment.

M3: Build student community with high ethical standards to undertake innovative research and development in thrust areas of national and international needs.

M4: Expose the students to the emerging technological advancements for meeting the demands of the industry.

Program Educational Objectives (PEOs)

PEOs	**PEO Statements**
PEO1	To prepare graduates with potential to get employed in the right role and/or become entrepreneurs to contribute to the society.
PEO2	To provide the graduates with the requisite knowledge to pursue higher education and carry out research in the field of Computer Science.
PEO3	To equip the graduates with the skills required to stay motivated and adapt to the dynamically changing world so as to remain successful in their career.
PEO4	To train the graduates with effectively, work collaboratively and exhibit high levels of professionalism and ethical responsibility.

COURSE OUTCOMES:

At the end of this course, the students will be able to

1. Comprehend the basic elements of Microcontroller and their Programming.
2. Use Raspberry Pi3 in Peripheral and in Trouble shooting.
3. Evaluate networking technologies for application within IOT.

Mapping of Course Outcomes with Programme Outcomes												
	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**
CO1	-	-	2	-	2	-	-	-	-	-	-	-
CO2	-	3	3	1	3	1	-	-	-	-	-	2
CO3	2	2	-	-	-	-	-	-	-	2	-	2

Rubric for CO3

Rubric for CO3 in Laboratory Courses				
Distribution of 10 Marks for CIE/SEE Evaluation Out of 40/60 Marks				
Rubric	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.

INDEX

INTERNET OF THINGS

Introduction:

Internet is a network of live persons. Internet of Things or IoT is a network of things and persons. IoT brings the things alive and there is interacting among themselves and with persons lively. Internet of Things is the network of devices such as vehicles, and home appliances that contain electronics, software, actuators, and connectivity which allows these things to connect, interact and exchange data. With the help of embedded technology, these things can communicate and interact over the Internet, and they can be remotely monitored and controlled. In the Internet of things, the precise geographic location and also the dimensions of a thing is critical. Therefore, sensors, transducers, locating devices and networks play very important role in IoT.

IoT makes virtually everything "smart," by improving aspects of our life with the power of data collection, AI algorithm, and networks. The thing in IoT can also be a person with a diabetes monitor implant, an animal with tracking devices, etc. There are many areas of applications of the Internet of Things like consumer, health, industrial, transportation, security, entertainment and many other. Security, Privacy, Complexity, Compliance, are key challenges of IoT

Components of IOT:

- **Sensors/Devices:** Sensors or devices are a key component that helps you to collect live data from the surrounding environment.
- **Connectivity:** All the collected data is sent to a cloud infrastructure. The sensors should be connected to the cloud using various mediums of communications like Bluetooth, WI-FI, WAN, etc.
- **Data Processing:** Once that data is collected, and it gets to the cloud, the software performs processing on the gathered data.
- **User Interface:** The information needs to be available to the end-user in some way which can be achieved by triggering alarms on their phones or sending them notification through email or text message.

Advantages of IOT:

- Technical optimization
- Improved data collection
- Reduced waste
- Improved customer engagement

EX.NO: 01	STUDY OF ARM EVALUATION SYSTEM
DATE :	

Aim:

To study of ARM processor system and describe the features of architecture.

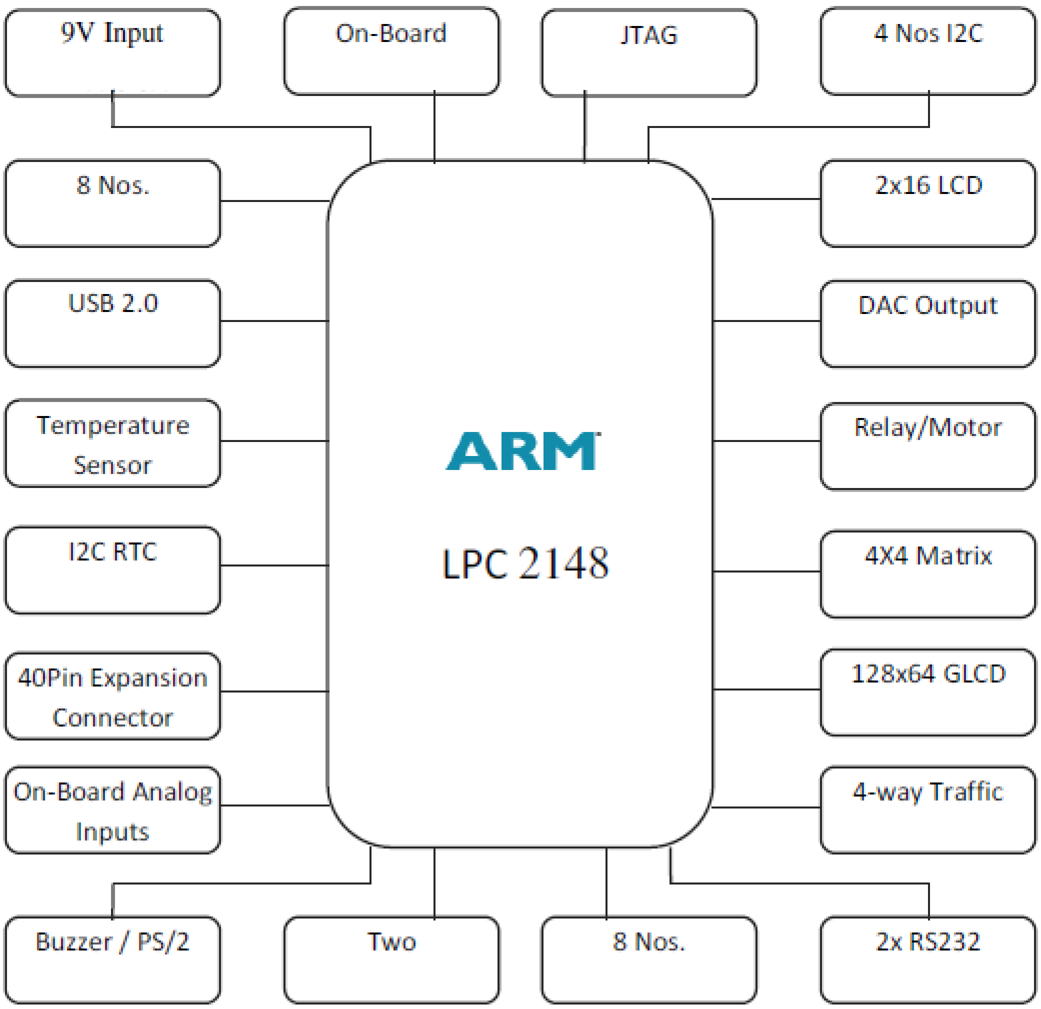
Architecture of ARM processor:

Features of ARM DEVELOPMENT KIT Processor

- 16-bit/32-bit ARM7TDMI-S microcontroller in a tiny LQFP64 package.8 kB to 40 kB of on-chip static RAM and 32 kB to 512 kB of on-chip flash memory. 128- bit wide interface/accelerator enables high-speed 60 MHz operation. In-System/In-Application Programming (ISP/IAP) via on-chip boot loader software.
- Single flash sector/full chip erase in 400 ms and programming of 256 bytes in 1 ms. USB 2.0 Full-speed compliant device controller with 2 kB of endpoint RAM. The LPC2146/48 provides 8 kB of on-chip RAM accessible to USB by DMA.
- One or two (LPC2141/42 vs. LPC2144/46/48) 10-bit ADCs provide a total of 6/14 analog inputs, with conversion times as low as 2.44 μs per channel. Single 10-bit DAC provides variable analog output (LPC2142/44/46/48 only).Two 32-bit timers/external event counters (with four capture and four compare channels each), PWM unit (six outputs) and watchdog.
- Low power Real-Time Clock (RTC) with independent power and 32 kHz clock input. Multiple serial interfaces including two UARTs (16C550), two Fast I2Cbus (400 kbit/s), SPI and SSP with buffering and variable data length capabilities.
- Vectored Interrupt Controller (VIC) with configurable priorities and vector addresses. Up to 45 of 5 V tolerant fast general purpose I/O pins in a tiny LQFP64 package. Up to 21 external interrupt pins available.
- 60MHz maximum CPU clock available from programmable on-chip PLL with settling time of 100μs.On-chip integrated oscillator operates with an external crystal from 1 MHz to 25 MHz. Power saving modes include Idle and Power-down.

- Individual enable/disable of peripheral functions as well as peripheral clock scaling for additional power optimization. Processor wake-up from Power-down mode via external interrupt or BOD. Single power supply chip with POR and BOD circuits: CPU operating voltage range of 3.0 V to 3.6 V (3.3 V ± 10 %) with 5 V tolerant I/O pads.

General Block Diagram:



Power supply:

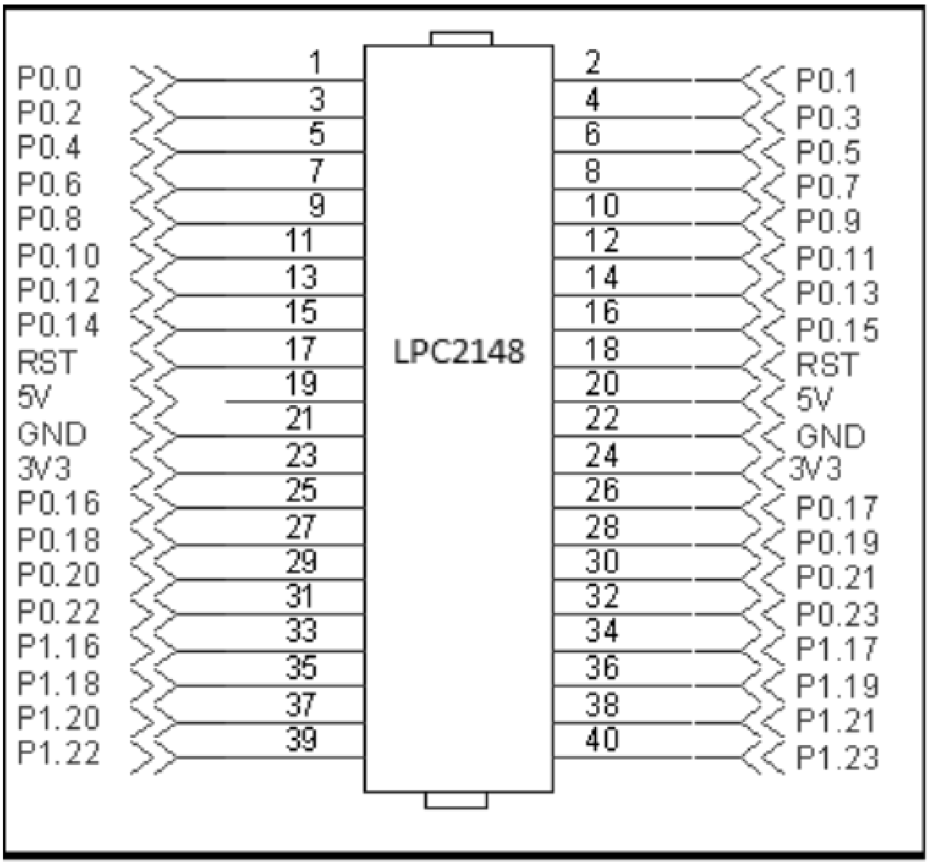
- The external power can be AC or DC, with a voltage between (9V/12V, 1A output) at 230V AC input. The ARM board produces +5V using an LM7805 voltage regulator, which provides supply to the peripherals.
- LM1117 Fixed +3.3V positive regulator used for processor & processor related peripherals.

Flash Programming Utility

- NXP (Philips)

NXP Semiconductors produce a range of Microcontrollers that feature both on-chip Flash memory and the ability to be reprogrammed using In-System Programming technology.

Pin configuration:



On-board Peripherals:

- 8-Nos. of Point LED’s (Digital Outputs)
- 8-Nos. of Digital Inputs (slide switch)
- 2 Lines X 16 Character LCD Display
- I2C Enabled 4 Digit Seven-segment display
- 128x64 Graphical LCD Display
- 4 X 4 Matrix keypad
- Stepper Motor Interface
- 2 Nos. Relay Interface
- Two UART for serial port communication through PC
- Serial EEPROM
- On-chip Real Time Clock with battery backup
- PS/2 Keyboard interface(Optional)
- Temperature Sensor
- Buzzer(Alarm Interface)
- Traffic Light Module(Optional)

Result:

Thus the study of ARM processor was done and ensured its composition with internal features specifically.

EX.NO:02	STUDY OF 8051 MICROCONTROLLER
DATE:	

Aim:

To study the features and pin structure of 8051MC.

Salient features of 8051

- A Microcontroller is a complete computer system built on a single chip.
- It contains all components like Processor (CPU), RAM, ROM, Serial port, Parallel port, Interrupt logic, Timer etc on chip.
- A Microcontroller saves cost, saves power consumption and makes the circuit compact.
- 8051 is an 8- bit Microcontroller
- On-Chip ROM = 4 KB (Program Memory).
- On-Chip RAM = 128 Bytes (Data Memory)
- Four 8 bit bi-directional I/O ports.
- Serial port
- Two 16 bit Up-Counters (Timers).
- It supports interrupts with two-level priority.
- Power saving modes.
- It is used in appliances such as Washing Machines, Microwaves, Mobile Phones, MP3 Players etc.

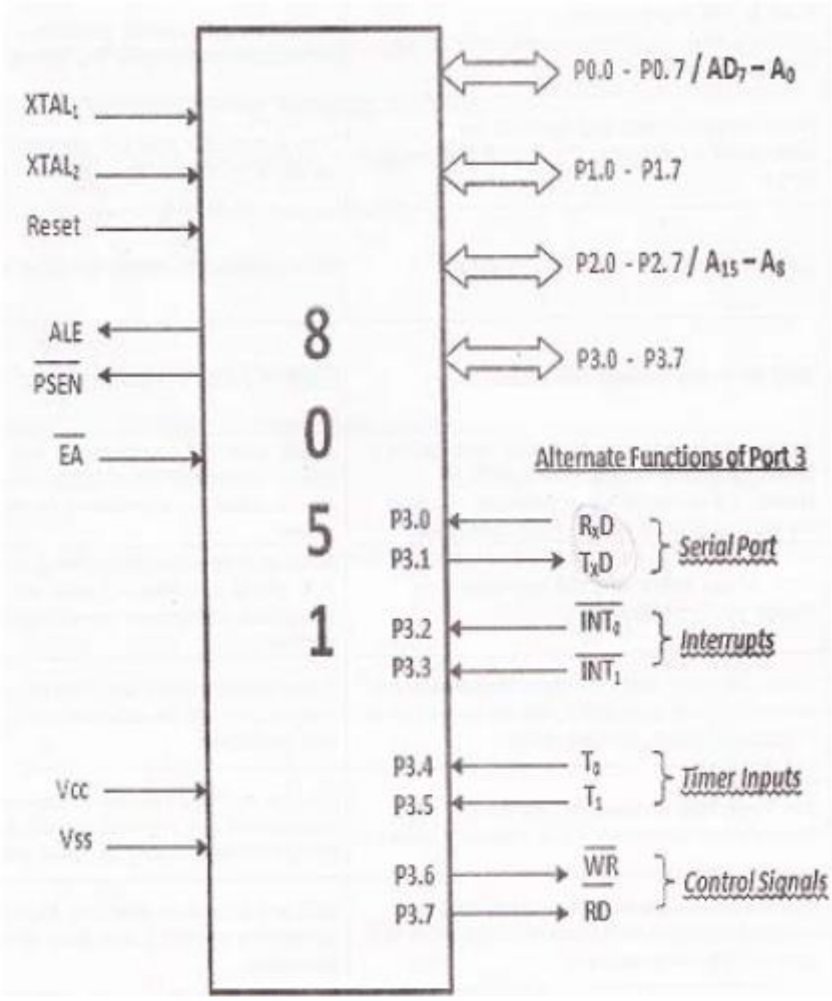
Pin diagram of 8051:

Port 0(p0.0 to p0.7): It is 8-bit bi-directional I/O port. It is bit/ byte addressable. During external memory access, it functions as multiplexed data and low-order address bus AD0-AD7

Port 1 (p1.0 to p1.7): It is 8-bit bi-directional I/O port. It is bit/ byte addressable. When logic '1' is written into port latch then it works as input mode. It functions as simply I/O port and it does not have any alternative function.

Port 2 (p2.0 to p2.7): It is 8-bit bi-directional I/O port. It is bit/ byte addressable. During external memory access it functions as higher order address bus (A8-A15).

Port 3(p3.0 to port 3.7): It is 8-bit I/O port. In an alternating function each pins can be used as a special function I/O pin.



P3.0-RxD: It is an Input signal. Through this I/P signal microcontroller receives serial data of serial communication circuit.

P3.1-TxD: It is O/P signal of serial port. Through this signal data is transmitted.

P3.2- (INT0): It is external hardware interrupt I/P signal. Through this user, programmer or peripheral interrupts to microcontroller.

P3.3-(INT1): It is external hardware interrupt I/P signal. Through this user, programmer or peripheral interrupts to microcontroller.

P3.4- T0: It is I/P signal to internal timer-0 circuit. External clock pulses can connects to timer-0 through this I/P signal.

P3.5-T1: It is I/P signal to internal timer-1 circuit. External clock pulses can connects to timer-1 through this I/P signal

P3.6-[WR(bar)]: It is active low write O/P control signal. During External RAM (Data memory) access it is generated by microcontroller. When [WR(bar)]=0, then performs write operation.

P3.7-[RD(bar)]: It is active low read O/P control signal. During External RAM (Data memory) access it is generated by microcontroller. When [RD(bar)]=0, then performs read operation from external RAM.

XTAL1 and XTAL2: These are two I/P line for on-chip oscillator and clock generator circuit. A resonant network as quartz crystal is connected between these two pin. 8051 microcontroller also drives from external clock, then XTAL2 is used to drive 8051 from external clock and XTAL1 should be grounded.

[EA(bar)]/VPP: It is and active low I/P to 8051 microcontroller. When (EA)= 0, then 8051 microcontroller access from external program memory (ROM) only. When (EA) = 1, then it access internal and external program memories (ROMS).

[PSEN(bar)]: It is active low O/P signal. It is used to enable external program memory (ROM). When [PSEN(bar)]= 0, then external program memory becomes enabled and microcontroller read content of external memory location. Therefore it is connected to (OE) of external ROM. It is activated twice every external ROM memory cycle.

ALE: Address latch enable: It is active high O/P signal. When it goes high, external address latch becomes enabling and lower address of external memory (RAM or ROM) latched into it. Thus it separates A0-A7 address from AD0-AD7. It provides properly timed signal to latch lower byte address. The ALE is activated twice in every machine cycle. If external RAM & ROM is not accessed, then ALE is activated at constant rate of 1/6 oscillator frequency, which can be used as a clock pulses for driving external devices.

RESET: It is active high I/P signal. It should be maintained high for at least two machine cycle while oscillator is running then 8051 microcontroller resets

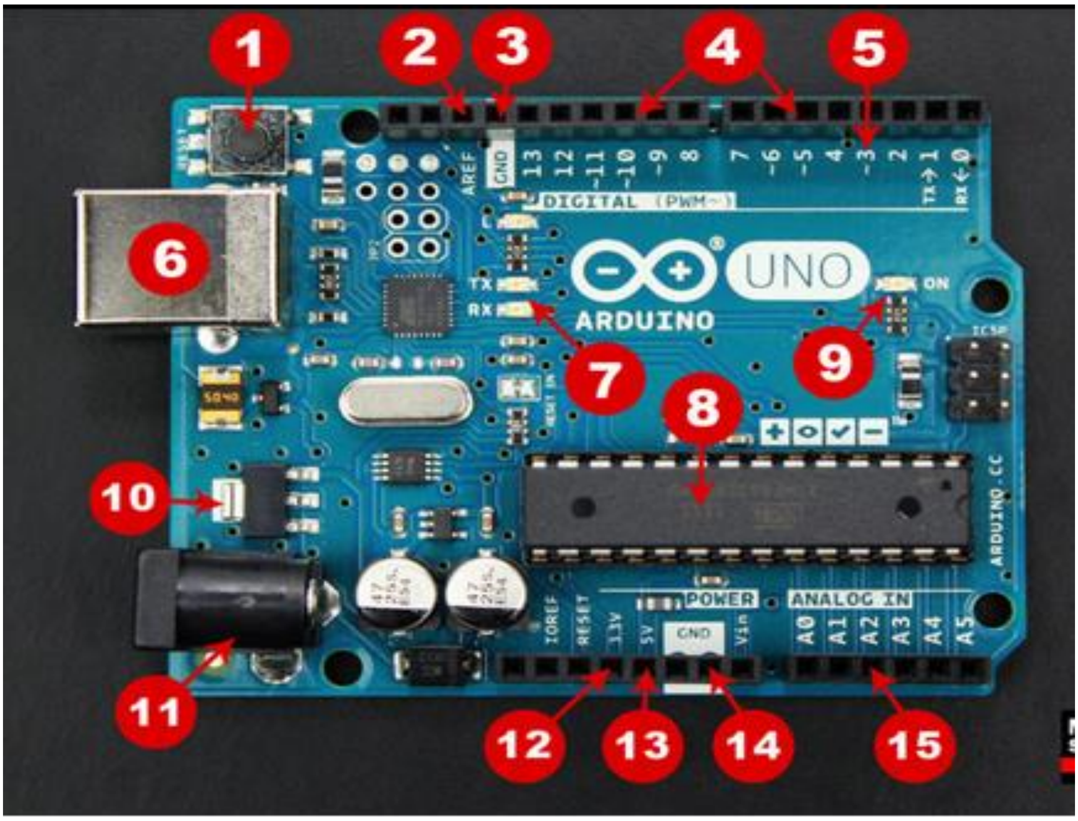
Result:

Thus the salient features and pin structure of 8051MC were studied in detail.

Arduino UNO

Introduction:

Arduino is an open source programmable circuit board that can be integrated into a wide variety of makerspace projects both simple and complex. This board contains a microcontroller which is able to be programmed to sense and control objects in the physical world. By responding to sensors and inputs, the Arduino is able to interact with a large array of outputs such as LEDs, motors and displays. Because of its flexibility and low cost, Arduino has become a very popular choice for makers and makerspaces looking to create interactive hardware projects.



Here are the components that make up an Arduino board and what each of their functions are.

1. **Reset Button** – This will restart any code that is loaded to the Arduino board

2. **AREF** – Stands for “Analog Reference” and is used to set an external reference voltage

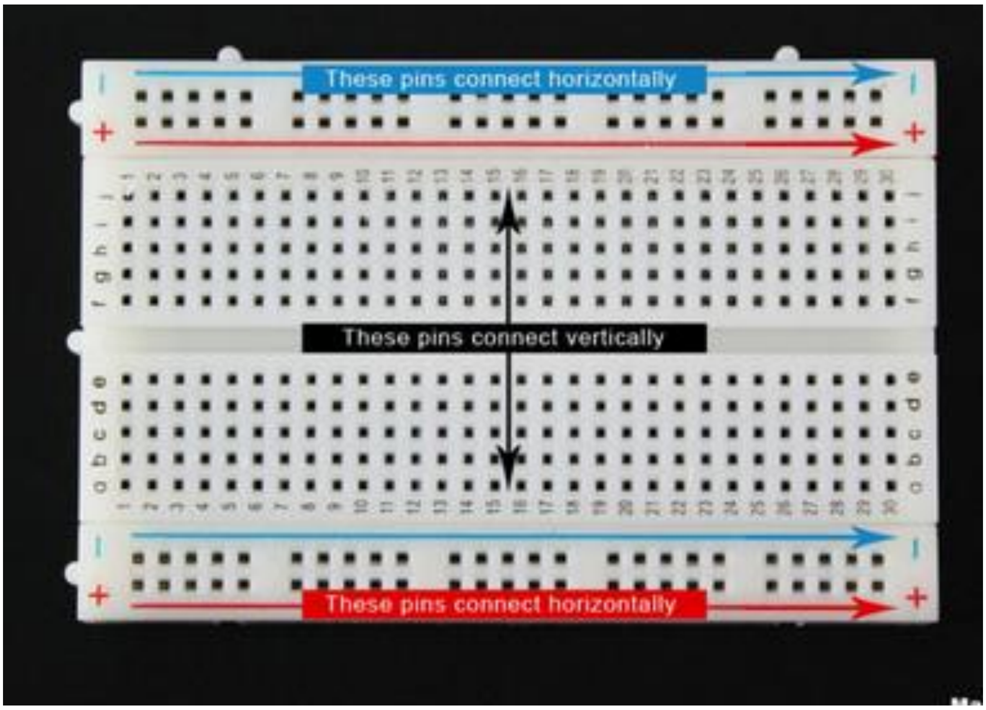
3. **Ground Pin** – There are a few ground pins on the Arduino and they all work the same
4. **Digital Input/Output** – Pins 0-13 can be used for digital input or output
5. **PWM** – The pins marked with the (~) symbol can simulate analog output
6. **USB Connection** – Used for powering up your Arduino and uploading sketches
7. **TX/RX** – Transmit and receive data indication LEDs
8. **ATmega Microcontroller** – This is the brains and is where the programs are stored
9. **Power LED Indicator** – This LED lights up anytime the board is plugged in a power source
10. **Voltage Regulator** – This controls the amount of voltage going into the Arduino board
11. **DC Power Barrel Jack** – This is used for powering your Arduino with a power supply
12. **3.3V Pin** – This pin supplies 3.3 volts of power to your projects
13. **5V Pin** – This pin supplies 5 volts of power to your projects
14. **Ground Pins** – There are a few ground pins on the Arduino and they all work the same
15. **Analog Pins** – These pins can read the signal from an analog sensor and convert it to digital

Power supply:

The Arduino Uno needs a power source in order for it to operate and can be powered in a variety of ways. The most common way is to connect the board directly to the computer via a USB cable.

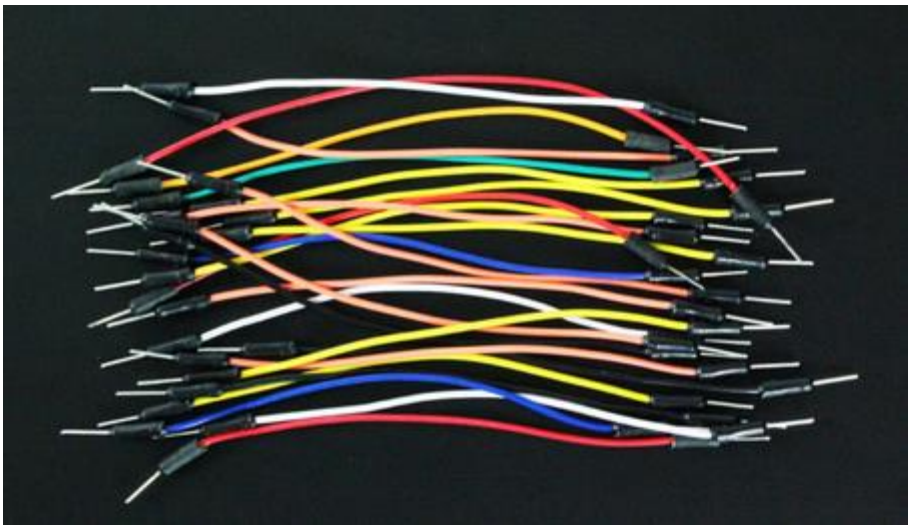
Arduino Breadboard

This device allows you to prototype your Arduino project without having to permanently solder the circuit together. Using a breadboard allows you to create temporary prototypes and experiment with different circuit designs. Inside the holes (tie points) of the plastic housing, are metal clips which are connected to each other by strips of conductive material.

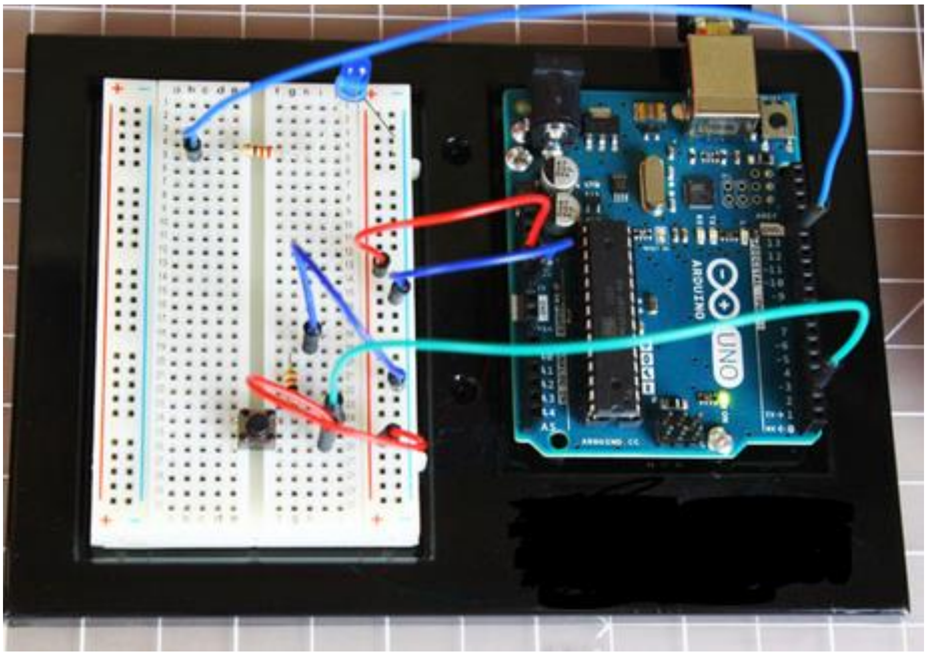


Jumper wire:

The breadboard is not powered on its own and needs power brought to it from the Arduino board using jumper wires. These wires are also used to form the circuit by connecting resistors, switches and other components together.



Here is a visual of what a completed Arduino circuit looks like when connected to a breadboard.



Once the circuit has been created on the breadboard, program (known as a sketch) need to be uploaded to the Arduino. The sketch is a set of instructions that tells the board what functions it needs to perform. An Arduino board can only hold and perform one sketch at a time. The software used to create

Arduino sketches is called the IDE which stands for Integrated Development Environment. The software is free to download and can be found at https://www.arduino.cc/en/Main/Software

Every Arduino sketch has two main parts to the program:

void setup() – Sets things up that have to be done once and then don't happen again.

void loop() – Contains the instructions that get repeated over and over until the board is turned off.

EX. NO: 03	DISTANCE MEASUREMENT USING ARDUINO
DATE :	

Aim:

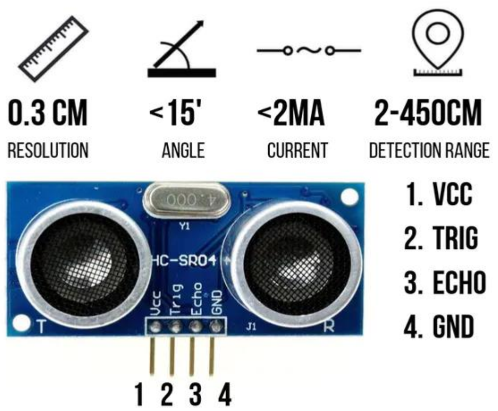
To measure the distance of an object using ultrasonic sensor HC–SR04.

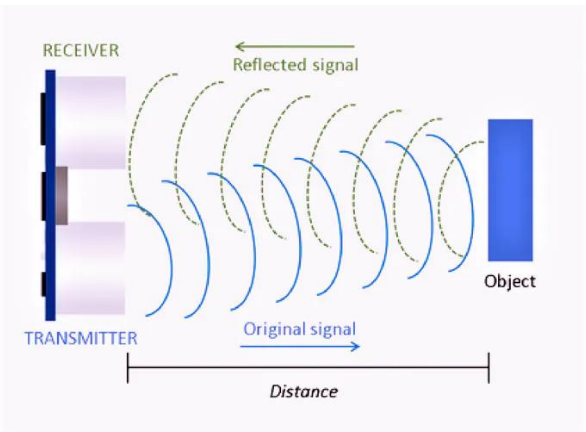
List of components:

1. Solderless Breadboard Full size
2. Jumper Wires
3. Arduino UNO
4. Buzzer
5. LEDs
6. LCD 16 X 2 12C pin
7. Ultrasonic Sensor HC-SR04

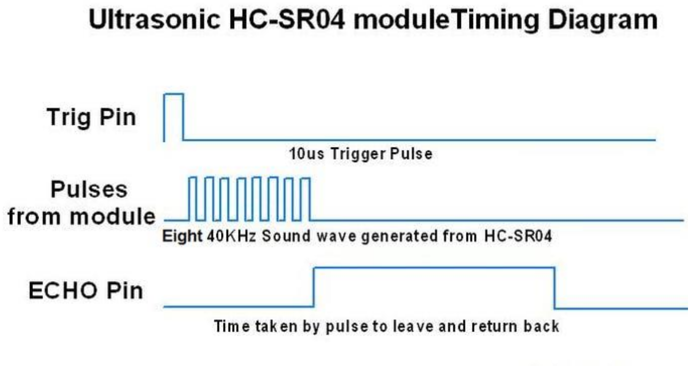
Working description:

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.

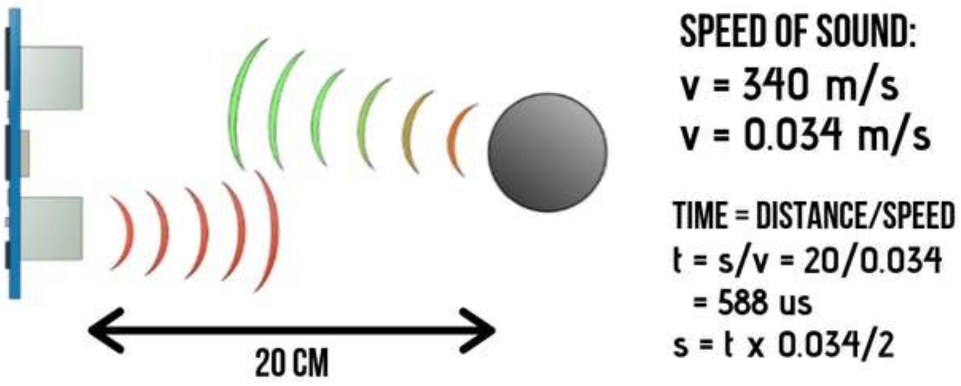




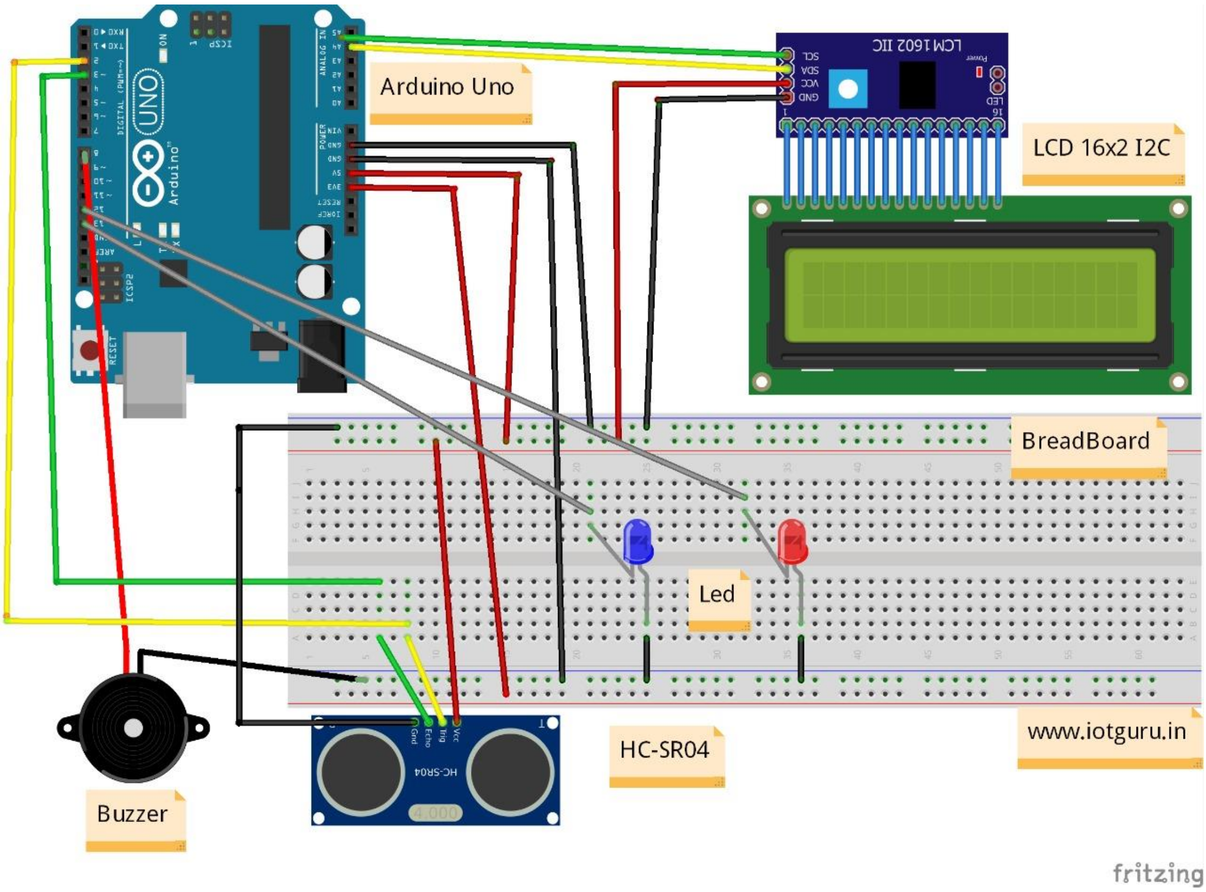
In order to generate the ultrasound we need to set the Trigger Pin on a High State for 10 µs. That will send out an 8 cycle sonic burst which will travel at the speed sound and it will be received in the Echo Pin. The Echo Pin will output the time in microseconds the sound wave travelled.



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.



Circuit diagram:



Connections:

Ultrasonic sensor to Arduino board:

Vcc to 5v

Trig(input) to pin 2

Gnd to gnd

Echo (output) to pin 3

LED BLUE to Arduino board:

Negative to gnd

Positive to pin 13

LED RED to Arduino board:

Negative to gnd

Positive to pin 12

Buzzer to Arduino board:

Negative to gnd

Positive to pin 8

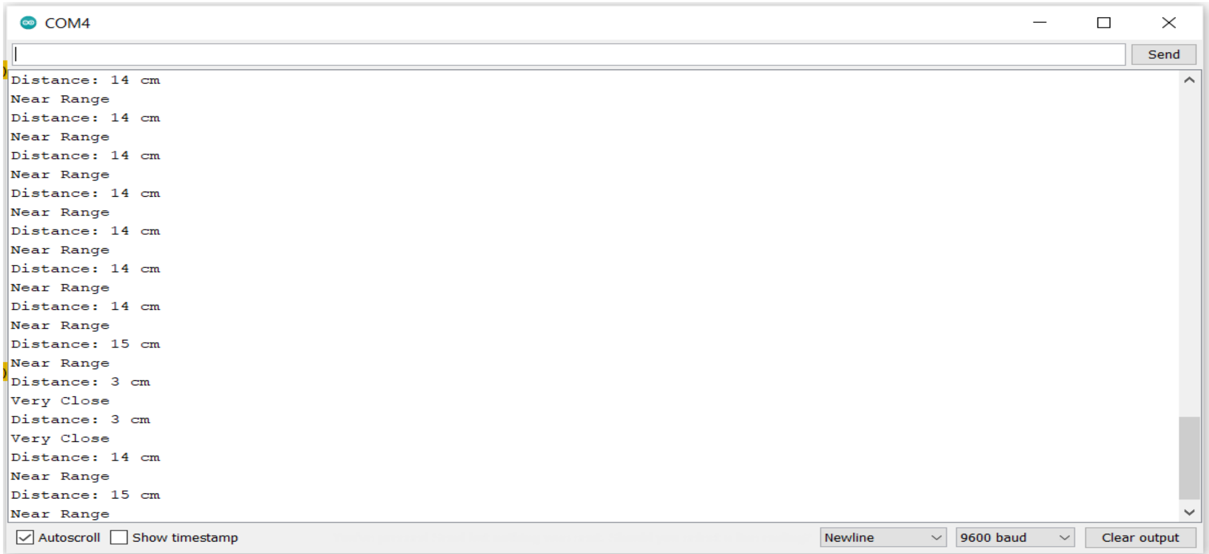
Code:

```
#include <Wire.h>
#define echoPin 3
#define trigPin 2
long duration;
int distance;
const int buzzer = 8;
const int light = 13;
const int light2 = 12;
void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
  pinMode(buzzer, OUTPUT);
  pinMode(light, OUTPUT);
  Serial.println("Annamalai University BE CSE 2022 "); // print some text in Serial Monitor
  Serial.println("Distance Measure Program");
}
```

```
void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2;
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
  digitalWrite(light, LOW);
  digitalWrite(light2, LOW);
  if (distance <= 10)
  {
      Serial.println("Very Close");
    digitalWrite(light, HIGH);
    tone(buzzer, 1000); // Send 1KHz sound signal...
    delay(1000);        // ...for 1 sec
    digitalWrite(light2, LOW);
    tone(buzzer, 500);
    noTone(buzzer);     // Stop sound...
  }
  else if (distance >=11 && distance <=50)
```

```
{
  Serial.println("Near Range");
  digitalWrite(light2, HIGH);
  digitalWrite(light, LOW);
  delay(1000);
}
else
{
  Serial.println("Far Range");
}}
```

Output:



Result:

Thus the distance of an object was measured using Arduino successfully.

EX. NO:04	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:

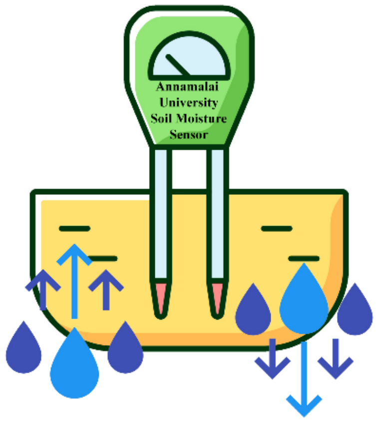
1. Solderless Breadboard Full Size
2. Jumper Wires
3. Arduino UNO
4. Buzzer
5. LEDs
6. LCD 16x2 I2C pin
7. LM393 High Precision Comparator
8. Soil Moisture sensor
9. 2 Channel Relay
10. Water Pump

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.

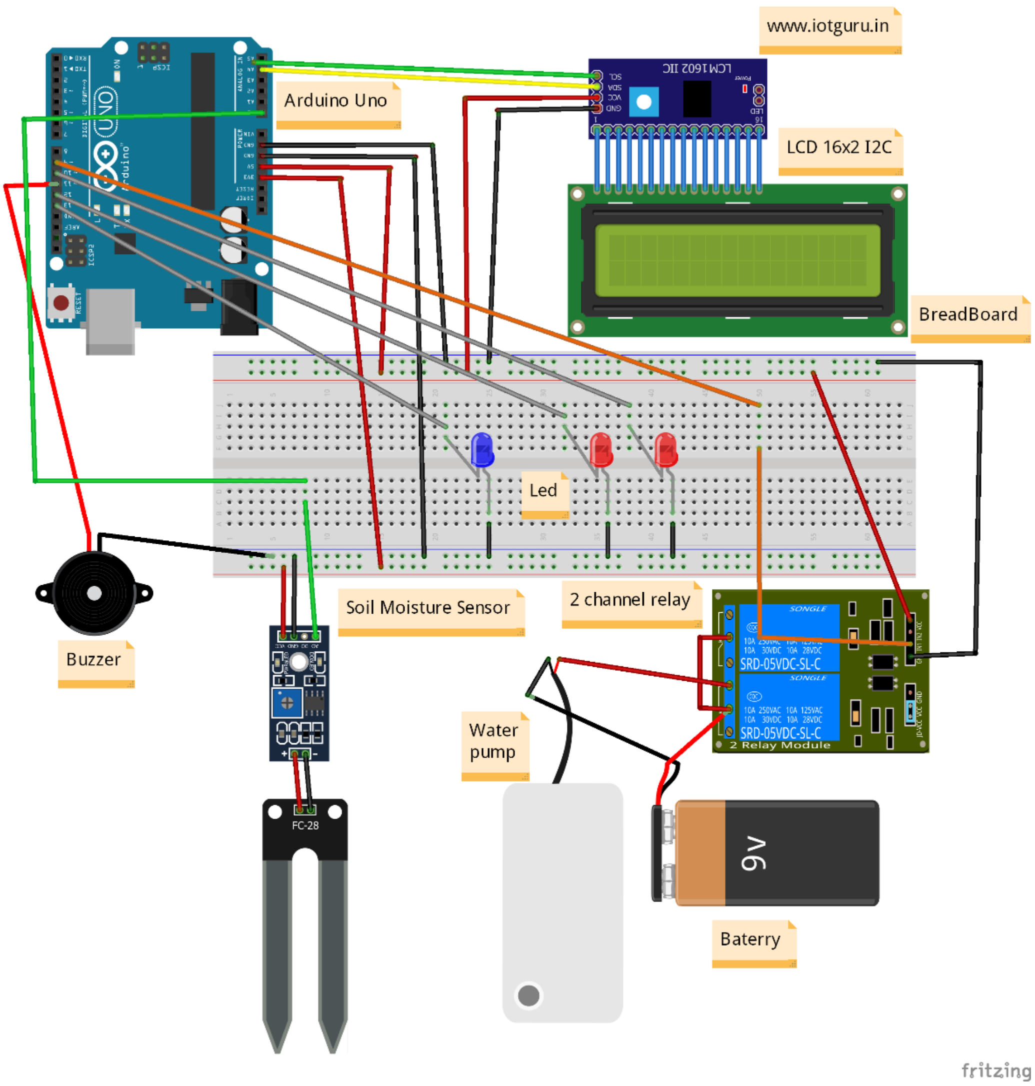
Soil moisture sensors typically refer to sensors that estimate volumetric water content. Another class of sensors measure another property of moisture in soils called water potential; these sensors are usually referred to as soil water potential sensors and include tensiometers and gypsum blocks.



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.

Circuit diagram:



Connections:

Soil moisture sensor to Arduino board:

AO to A0

Vcc to 5V

Gnd to gnd of buzzer

Buzzer to Arduino board:

+ve leg to pin 11

LED1 to Arduino board:

Gnd to gnd

+ve leg to pin 13

LED2 to Arduino board:

Gnd to gnd

+ve leg to pin 10

LED3 to Arduino board:

Gnd to gnd

+ve leg to pin 12

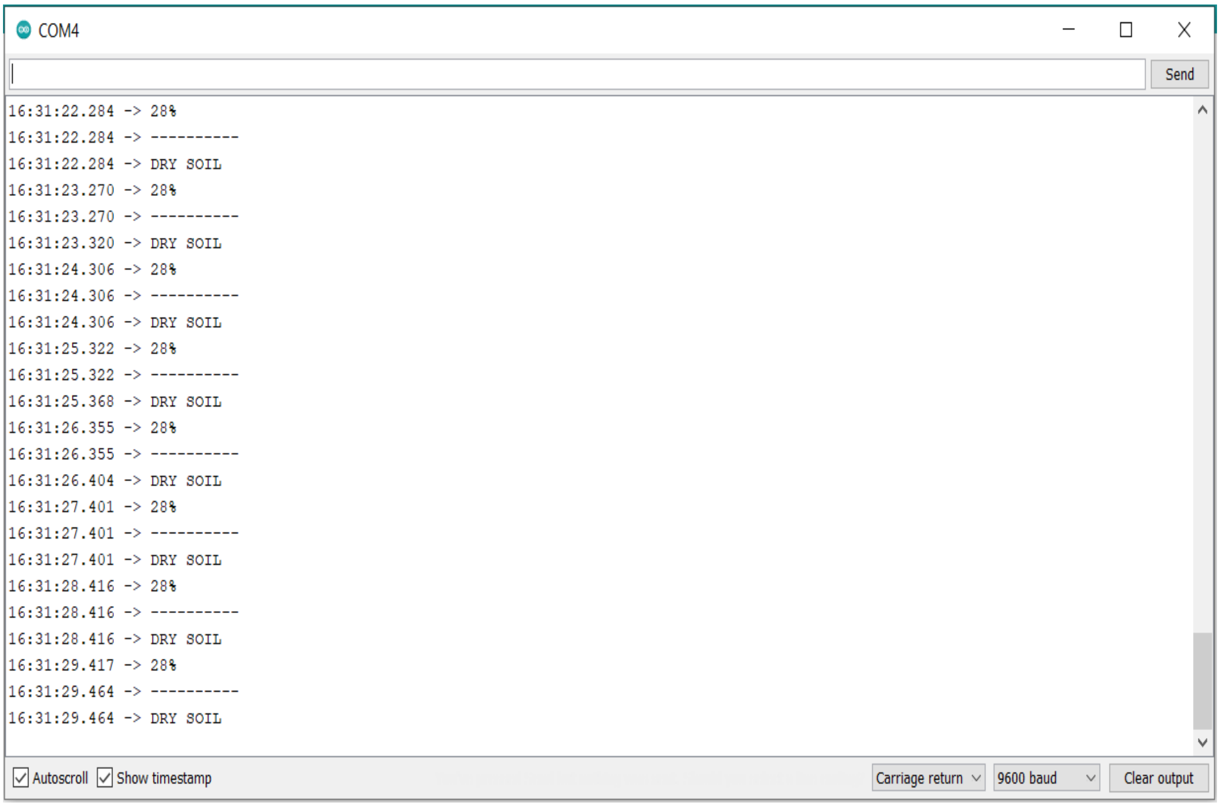
Code:

```
#include <Wire.h>

const int buzzer = 11;

const int light = 13;

const int light1 = 10;

const int light2 = 12;

int moisture;

void setup() {

  pinMode(A0, INPUT);

  Serial.begin(9600);

  pinMode(buzzer, OUTPUT);

  pinMode(light, OUTPUT);

  pinMode(light1, OUTPUT);

  pinMode(light2, OUTPUT);

  pinMode(9, OUTPUT);
```

```
  Serial.println("Annamalai University BE CSE 2022 "); // print some text in
Serial Monitor
  Serial.println("Soil Moisture Program");
}
void loop() {
  int SensorValue = analogRead(A0);
   moisture = (100 - ((SensorValue/1023.00) * 100));
  Serial.print(moisture);
  Serial.println("%");
  Serial.println("----------");
  if (SensorValue >= 1000)
  {
    Serial.println("Not in soil or disconnected");
  }
  if (SensorValue < 1000 && SensorValue >= 600)
  {
      digitalWrite(9, LOW);
      Serial.println("DRY SOIL");
      digitalWrite(light, HIGH);
      tone(buzzer, 1000); // Send 1KHz sound signal...
      delay(1000);        // ...for 1 sec
      digitalWrite(light1, LOW);
      digitalWrite(light2, LOW);
      tone(buzzer, 500);
      noTone(buzzer);     // Stop sound...
  }
```

```
 if (SensorValue < 600 && SensorValue >= 370)
 {
    digitalWrite(9, HIGH);
    Serial.println("HUMID SOIL");
    digitalWrite(light1, HIGH);
    digitalWrite(light2, LOW);
    digitalWrite(light, LOW);
    delay(1000);
 }
 if (SensorValue < 370)
 {
    digitalWrite(9, HIGH);
    Serial.println("WATER SOIL");
    digitalWrite(light2, HIGH);
    digitalWrite(light, LOW);
    digitalWrite(light1, LOW);
    delay(1000);
} }
```

Output:



Result:

Thus the moisture content in Agricultural land was measured using Arduino successfully.

EX. NO: 05	MOTION DETECTION USING ARDUINO
DATE :	

Aim:

To build a motion detection system using Arduino.

List of components

1. Solderless Breadboard Full Size
2. Jumper Wires
3. Arduino UNO
4. Buzzer
5. LEDs
6. LCD 16x2 I2C pin
7. LDR sensor
8. PIR Motion sensor
9. 2 Channel Relay
10. 5V Led Light
11. 5v FAN

Working Description

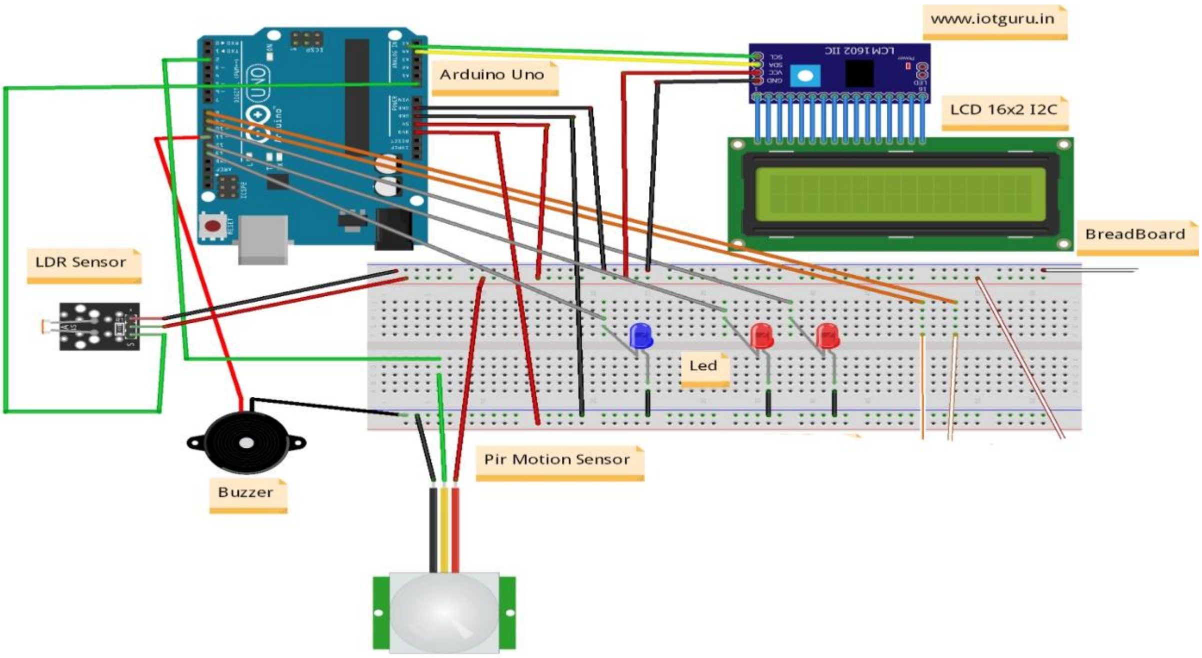
Home Automation is what makes us call our place a Smart Home. We can control all the appliances at our home using automation. The devices within the home automation system connect with each over a local network. When connected with the Internet, home devices are an important constituent of the Internet 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.

Circuit diagram:



Connections:

PIR sensor to Arduino board:

Gnd to gnd of buzzer

Vcc to 5V

Output to pin 2

LDR sensor to Arduino board:

Vcc to 5V

Gnd to gnd

Buzzer to Arduino board:

+ve pin to pin 11

LED1 to Arduino board:

Gnd to gnd

+ve leg to pin 13

LED2 to Arduino board:

Gnd to gnd

+ve leg to pin 12

LED3 to Arduino board:

Gnd to gnd

+ve leg to pin 10

Code:

```
#include <Wire.h>
const int buzzer = 11;
const int light = 13;
int sensor = 2;              // the pin that the sensor is atteched to
int state = LOW;             // by default, no motion detected
int val = 0;                 // variable to store the sensor status (value)
void setup() {
  Serial.begin(9600);
  pinMode(buzzer, OUTPUT);
  pinMode(light, OUTPUT);
  pinMode(sensor, INPUT);    // initialize sensor as an input
  Serial.println("Annamalai University BE CSE 2022 "); // print some text in Serial Monitor
  Serial.println("Home Automation Program");
}
    void loop() {
    val = digitalRead(sensor);   // read sensor value
    if (val == HIGH) {           // check if the sensor is HIGH
    delay(100);                  // delay 100 milliseconds
     if (state == LOW) {
     Serial.println("Motion Detected");
     Serial.println("");
     digitalWrite(light, HIGH);
     tone(buzzer, 1000); // Send 1KHz sound signal...
     delay(500);       // ...for 1 sec
```

```
    tone(buzzer, 500);
     noTone(buzzer);     // Stop sound...
     state = HIGH;       // update variable state to HIGH
   }
 }
 else {
     delay(100);             // delay 200 milliseconds
     if (state == HIGH){
       Serial.println("Motion stopped!");
       digitalWrite(light, LOW);
       state = LOW;       // update variable state to LOW
   }
 }
}
```

Output:

Motion not detected..

Motion detected..

Motion detected..

Motion not detected..

Result:

Thus the Motion detection System was built successfully using Arduino.

EX. NO: 06	IDENTIFYING ROOM TEMPERATURE AND HUMIDITY USING ARDUINO
DATE :	

Aim:

To identify the room temperature and humidity using DHT11 sensor.

List of components

1. Solderless Breadboard Full Size
2. Jumper Wires
3. Arduino UNO
4. Buzzer
5. LEDs
6. LCD 16x2 I2C pin
7. DHT11 Sensor

Working Description

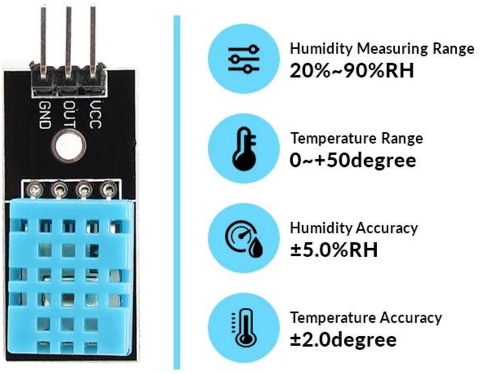
DHT11 is a low-cost digital sensor for sensing temperature and humidity. This sensor can be easily interfaced with any micro-controller such as Arduino, Raspberry Pi etc. to measure humidity and temperature instantaneously. DHT11 humidity and temperature sensor is available as a sensor and as a module. 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.



Connections:

DHT 11 sensor to Arduino board:

Gnd to gnd

Vcc to 5V

Out to pin 2

LED1 to Arduino board:

Gnd to gnd

+ve leg to pin 5

LED2 to Arduino board:

Gnd to gnd

+ve leg to pin 6

LED3 to Arduino board:

Gnd to gnd

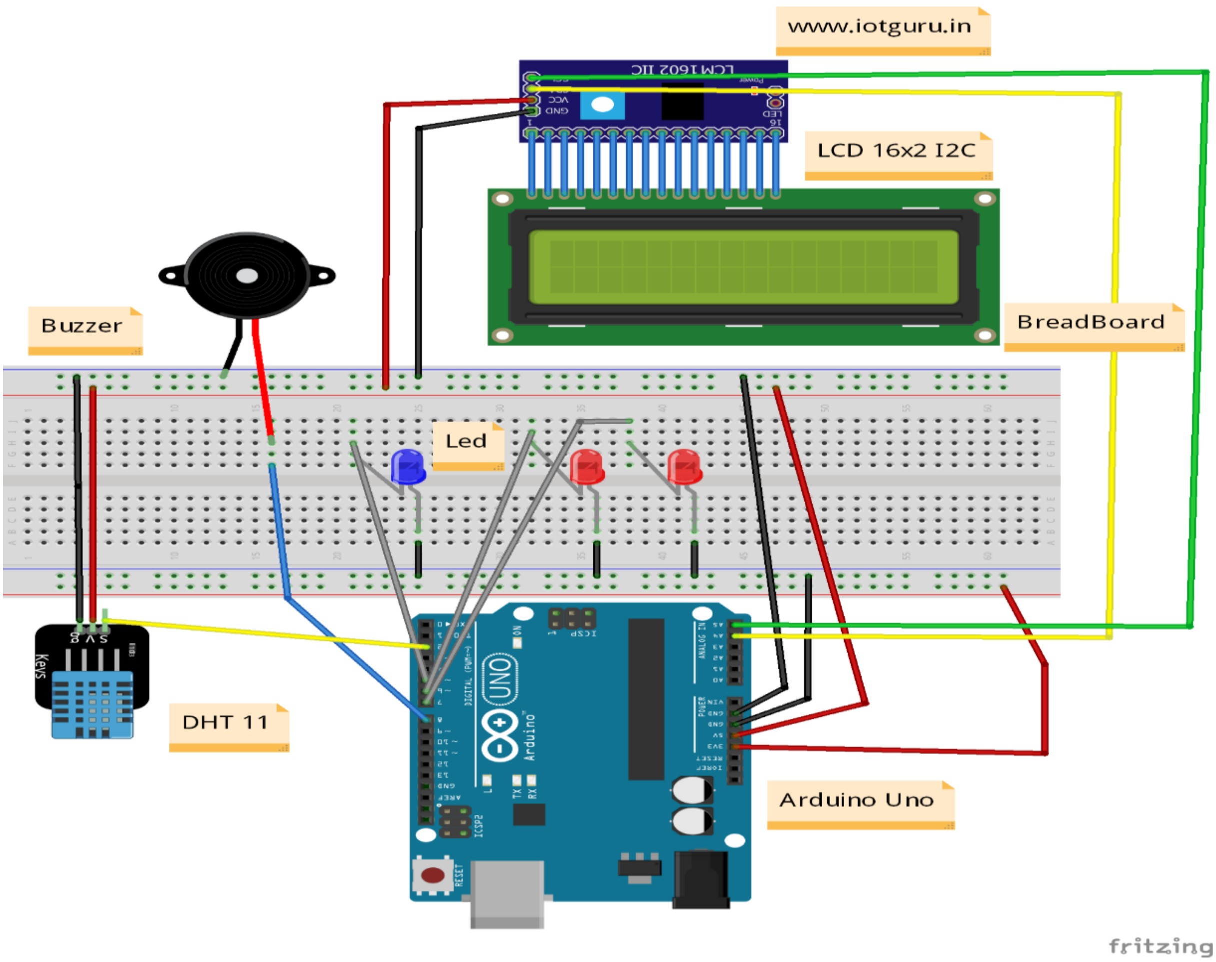
+ve leg to pin 7

Buzzer to Arduino board:

+ve pin to pin 8

-ve pin to gnd

Circuit diagram:



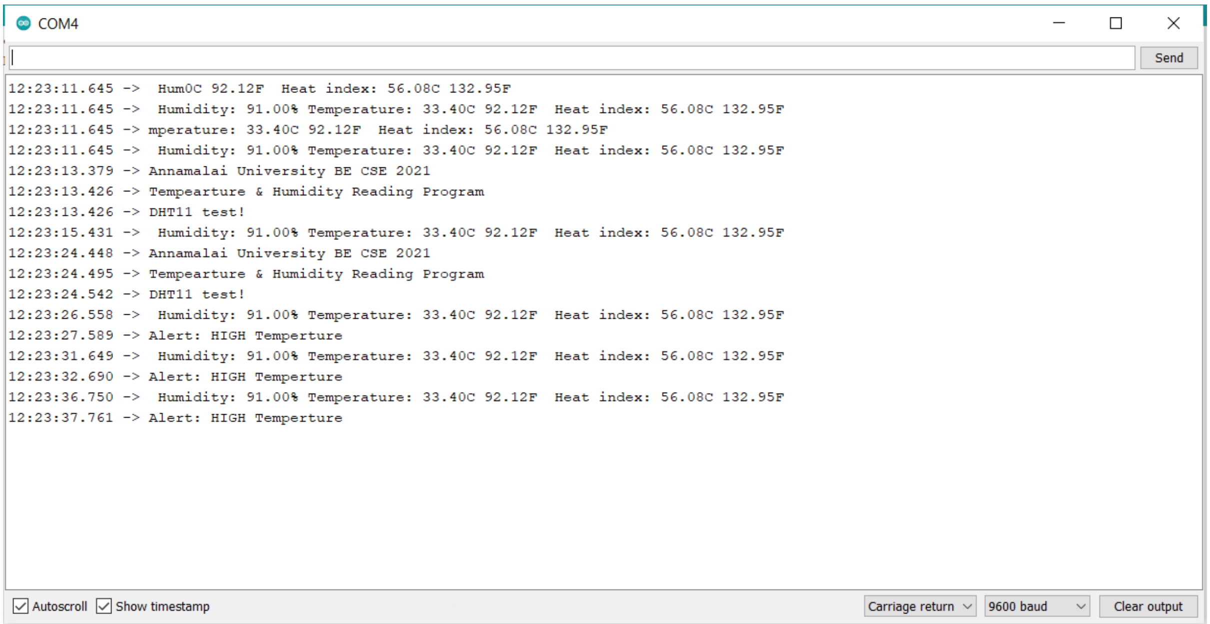
Code:

```
// REQUIRES the following Arduino libraries:
// - DHT Sensor Library: https://github.com/adafruit/DHT-sensor-library
// - Adafruit Unified Sensor Lib: https://github.com/adafruit/Adafruit_Sensor
#include <Wire.h>
#include "DHT.h"
#define DHTPIN 2     // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11   // DHT 11
DHT dht(DHTPIN, DHTTYPE);
const int buzzer = 8;
const int light = 7;
const int light1 = 6;
const int light2 = 5;
void setup() {
  Serial.begin(9600);
  pinMode(buzzer, OUTPUT);
  pinMode(light, OUTPUT);
  pinMode(light1, OUTPUT);
  pinMode(light2, OUTPUT);
  Serial.println("Annamalai University BE CSE 2021 "); // print some text in
Serial Monitor
  Serial.println("Tempearture & Humidity Reading Program");
  Serial.println(F("DHT11 test!"));
  dht.begin();
}
```

```
void loop() {
  digitalWrite(light, LOW);
  // Wait a few seconds between measurements.
  delay(2000);
  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  float h = dht.readHumidity(); // Read temperature as Celsius (the default)
  float t = dht.readTemperature(); // Read temperature as Fahrenheit
(isFahrenheit = true)
  float f = dht.readTemperature(true); // Check if any reads failed and exit early
(to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    digitalWrite(light, HIGH);
    return;  }
  float hif = dht.computeHeatIndex(f, h); // Compute heat index in Fahrenheit
(the default)
  float hic = dht.computeHeatIndex(t, h, false);  // Compute heat index in Celsius
(isFahreheit = false)
  Serial.print(F(" Humidity: "));
  Serial.print(h);
  Serial.print(F("% Temperature: "));
  Serial.print(t);
  Serial.print(F("C "));
  Serial.print(f);
  Serial.print(F("F  Heat index: "));
```

```
  Serial.print(hic);
  Serial.print(F("C "));
  Serial.print(hif);
  Serial.println(F("F"));
  if (t>29)
  { tone(buzzer, 1000); // Send 1KHz sound signal...
    delay(1000);        // ...for 1 sec
    digitalWrite(light1, HIGH);
    digitalWrite(light2, LOW);
    tone(buzzer, 500);
    noTone(buzzer);     // Stop sound...
    Serial.println("Alert: HIGH Temperture");  }
    else{
      Serial.println("Alert: Low Temperture");
      digitalWrite(light2, HIGH);
    }
  delay(2000);
  digitalWrite(light1,LOW);
}
```

Output:



Result:

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

EX. NO: 07	COLOUR RECOGNITION USING ARDUINO
DATE :	

Aim:

To identify the different colours using colour recognition sensor.

List of components

1. Jumper Wires
2. Arduino UNO
3. Colour recognition Sensor

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 colour 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 Colour 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.

Connections:

Colour recognition sensor to Arduino board:

Vcc to 5V

Gnd to gnd

Pin S0 to pin 8

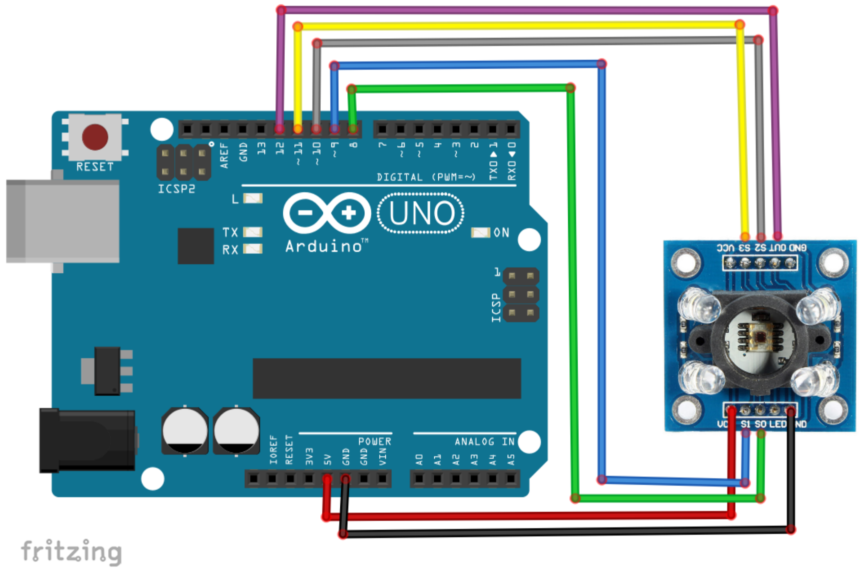
Pin S1 to pin 9

Pin S2 to pin 10

Pin S3 to pin 11

Out to pin 12

Circuit diagram:



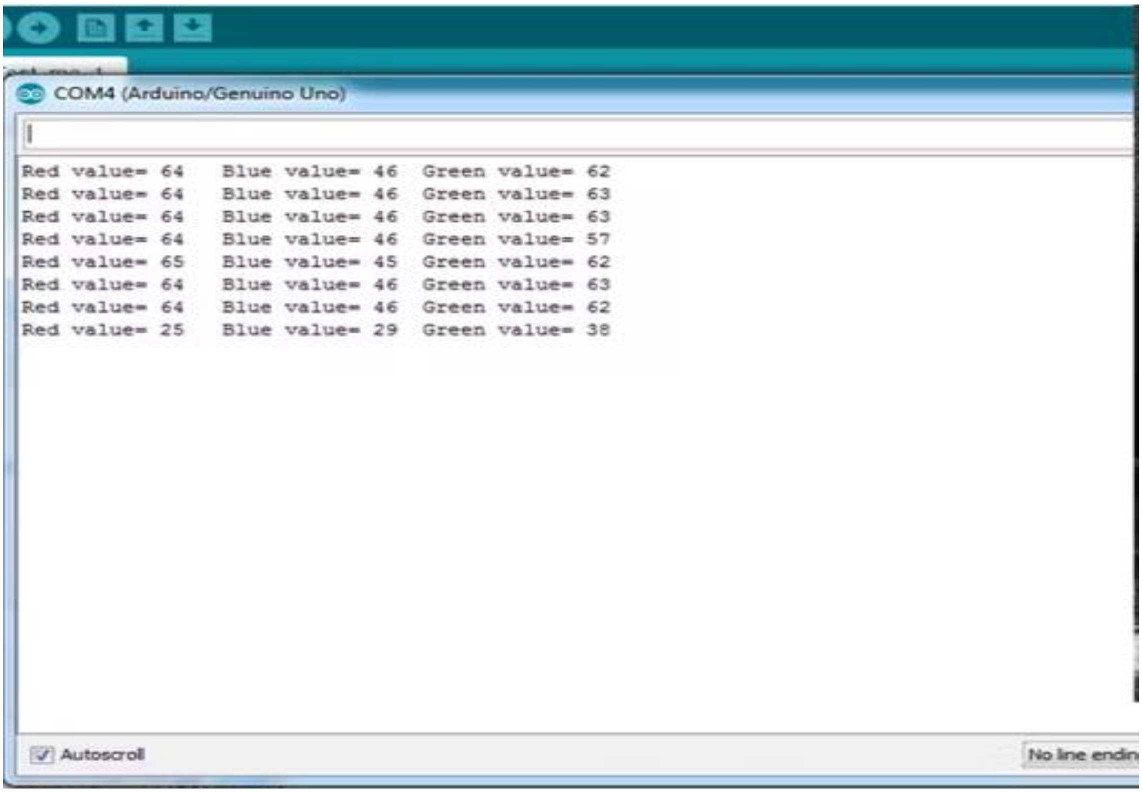
Code:

```
#define s0 8       //Module pins wiring
#define s1 9
#define s2 10
#define s3 11
#define out 12
int data=0;        //This is where we're going to stock our values
void setup()
{
   pinMode(s0,OUTPUT);    //pin modes
   pinMode(s1,OUTPUT);
   pinMode(s2,OUTPUT);
   pinMode(s3,OUTPUT);
   pinMode(out,INPUT);
   Serial.begin(9600);   //intialize the serial monitor baud rate
```

```
  digitalWrite(s0,HIGH); //Putting S0/S1 on HIGH/HIGH levels means the output frequency scalling is at 100% (recommended)

  digitalWrite(s1,HIGH); //LOW/LOW is off HIGH/LOW is 20% and LOW/HIGH is  2%

  }

void loop()                  //Every 2s we select a photodiodes set and read its data

{

  digitalWrite(s2,LOW);        //S2/S3 levels define which set of photodiodes we are using LOW/LOW is for RED LOW/HIGH is for Blue and HIGH/HIGH is for green

  digitalWrite(s3,LOW);

  Serial.print("Red value= ");

  GetData();                   //Executing GetData function to get the value

  digitalWrite(s2,LOW);

  digitalWrite(s3,HIGH);

  Serial.print("Blue value= ");

  GetData();

  digitalWrite(s2,HIGH);

  digitalWrite(s3,HIGH);

  Serial.print("Green value= ");

  GetData();

  Serial.println();

  delay(2000);

}
```

```
void GetData(){
  data=pulseIn(out,LOW);       //here we wait until "out" go LOW, we start measuring the duration and stops when "out" is HIGH again
  Serial.print(data);          //it's a time duration measured, which is related to frequency as the sensor gives a frequency depending on the color
  Serial.print("\t");          //The higher the frequency the lower the duration
  delay(20);
```

Output:



Result:

Thus, the colour recognition sensor has been interfaced with Arduino successfully.

EX. NO: 08	FIRE ALARM INDICATOR USING ARDUINO
DATE :	

Aim:

To develop a fire alarm indicator using the Flame sensor.

List of components:

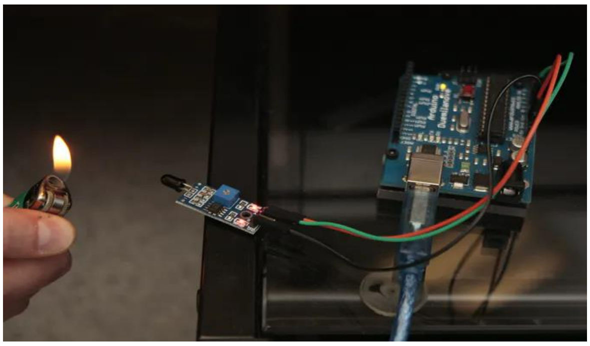
1. Solderless Breadboard Full Size
2. Jumper Wires
3. Arduino UNO
4. Buzzer
5. LEDs
6. LCD 16x2 I2C pin
7. Flame Sensor
8.

Working Description:

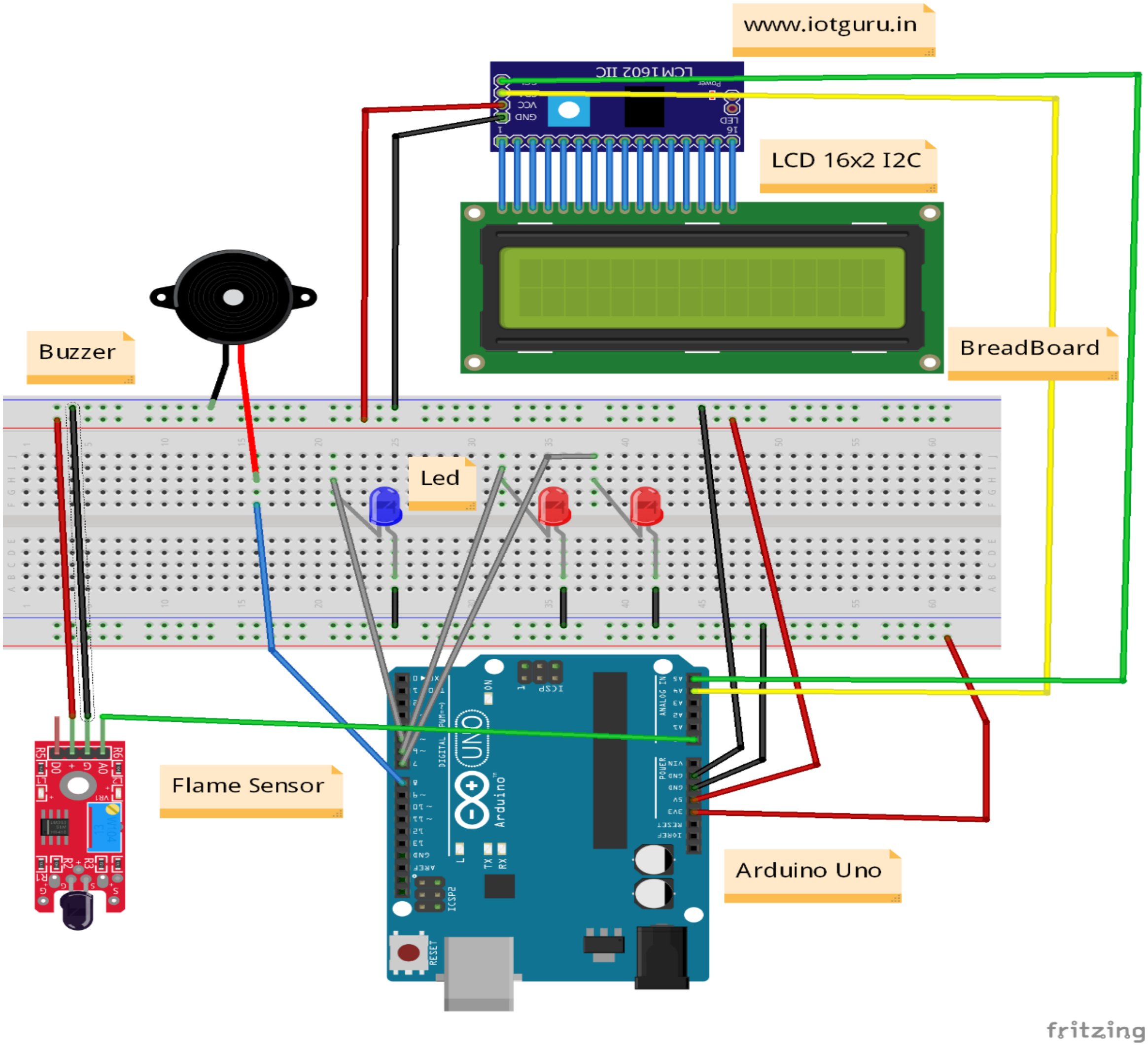
A flame-sensor is one kind of detector which is mainly designed for detecting as well as responding to the occurrence of a fire or flame. The flame detection response can depend on its fitting. It includes an alarm system, a natural gas line, propane & a fire suppression system. This sensor is used in 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 vapour, otherwise ice.



Circuit diagram:



Connections:

Flame sensor to Arduino board:

Gnd to gnd

+ve pin to 5V

AO to A0

LED1 to Arduino board:

Gnd to gnd

+ve leg to pin 5

LED2 to Arduino board:

Gnd to gnd

+ve leg to pin 6

LED3 to Arduino board:

Gnd to gnd

+ve leg to pin 7

Buzzer to Arduino board:

+ve pin to pin 8

-ve pin to gnd

Code:

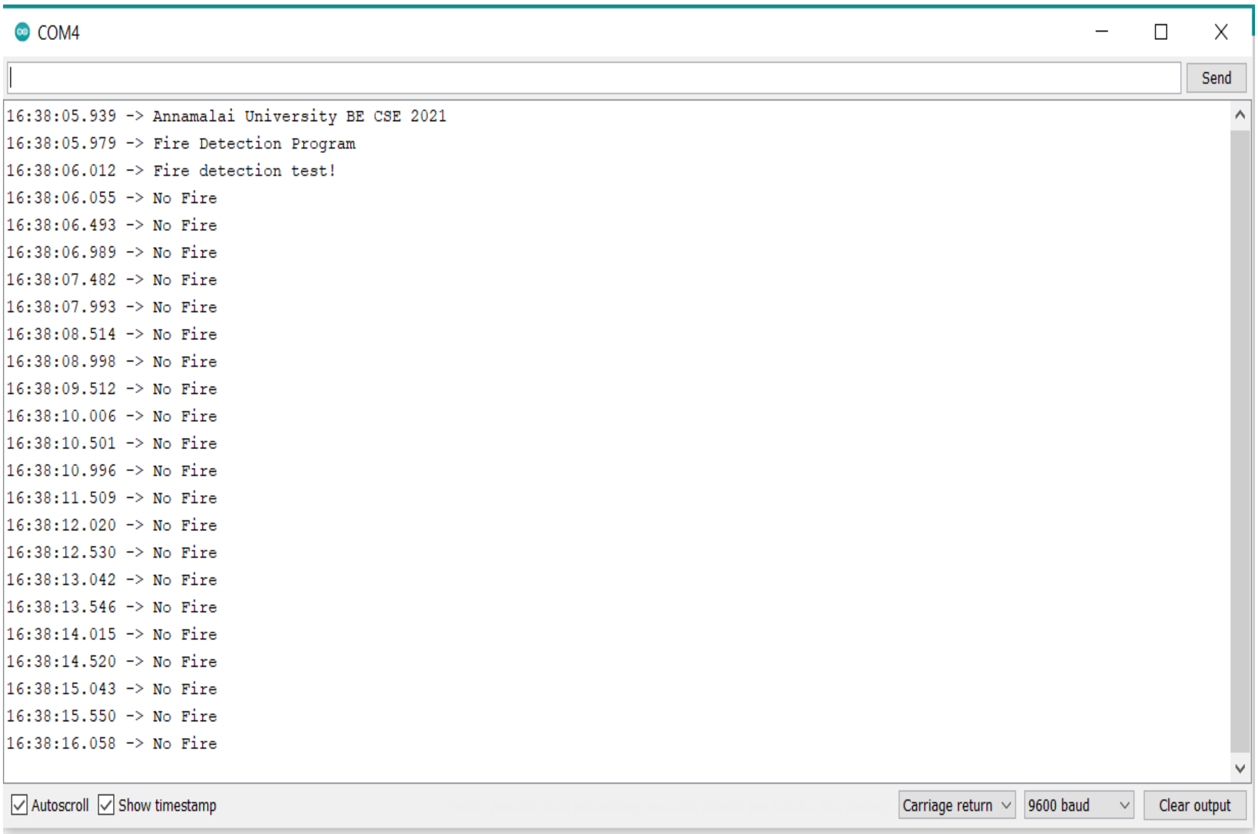
```
#include <Wire.h>

#include <LiquidCrystal_I2C.h>

// lowest and highest sensor readings:

const int sensorMin = 0;     // sensor minimum


const int sensorMax = 1024;  // sensor maximum
```

```
const int buzzer = 8;
const int light = 7;
const int light1 = 6;
const int light2 = 5;
//LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
void setup() {
//  lcd.begin(16,2);
//  lcd.backlight();
  Serial.begin(9600);
  pinMode(buzzer, OUTPUT);
  pinMode(light, OUTPUT);
  pinMode(light1, OUTPUT);
  pinMode(light2, OUTPUT);
  Serial.println("Annamalai University BE CSE 2021 "); // print some text in
Serial Monitor
  Serial.println("Fire Detection Program");
  Serial.println(F("Fire detection test!"));
//    lcd.clear();
//    lcd.setCursor(0,0);
//    lcd.print("Starting the");
//    lcd.setCursor(0,1);
//    lcd.print("Fire Sensor");
}
```

```
void loop() {
  // read the sensor on analog A0:
 int sensorReading = analogRead(A0);
  // map the sensor range (four options):
  // ex: 'long int map(long int, long int, long int, long int, long int)'
  int range = map(sensorReading, sensorMin, sensorMax, 0, 3);
  // range value:
  switch (range) {

  case 0:    // A fire closer than 1.5 feet away.
    Serial.println("Close Fire");
//     lcd.clear();
//     lcd.setCursor(0,0);
//     lcd.print("***CLOSE***");
//     lcd.setCursor(0,1);
//     lcd.print("****FIRE****");
     tone(buzzer, 1000); // Send 1KHz sound signal...
     delay(1000);
     digitalWrite(light, HIGH);
     tone(buzzer, 500);
     noTone(buzzer);     // Stop sound...
    break;

 case 1:
 // A fire between 1-3 feet away.
    Serial.println("** Distant Fire **");
```

```
//     lcd.clear();
//     lcd.setCursor(0,0);
//     lcd.print("***DISTANT***");
//     lcd.setCursor(0,1);
//     lcd.print("****FIRE****");
     digitalWrite(light1, HIGH);
    break;
  case 2:
// No fire detected.
    Serial.println("No Fire");
//     lcd.clear();
//     lcd.setCursor(0,0);
//     lcd.print("***NO***");
//     lcd.setCursor(0,1);
//     lcd.print("****FIRE****");
     digitalWrite(light2, HIGH);
     break;
  delay(500);  // delay between reads
  digitalWrite(light, LOW);
  digitalWrite(light1, LOW);
  digitalWrite(light2, LOW);
}
```

Output:



Result:

Thus the fire alarm indicator was successfully implemented using flame sensor

EX. NO: 09	SOUND DETECTION USING ARDUINO
DATE :	

Aim:

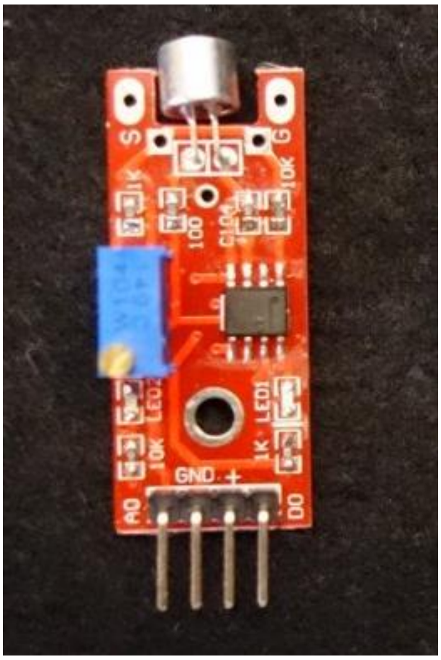
To develop a sound detection system using the sound sensor.

List of components:

1. Solderless Breadboard Full Size
2. Jumper Wires
3. Arduino UNO
4. LED
5. Sound Sensor

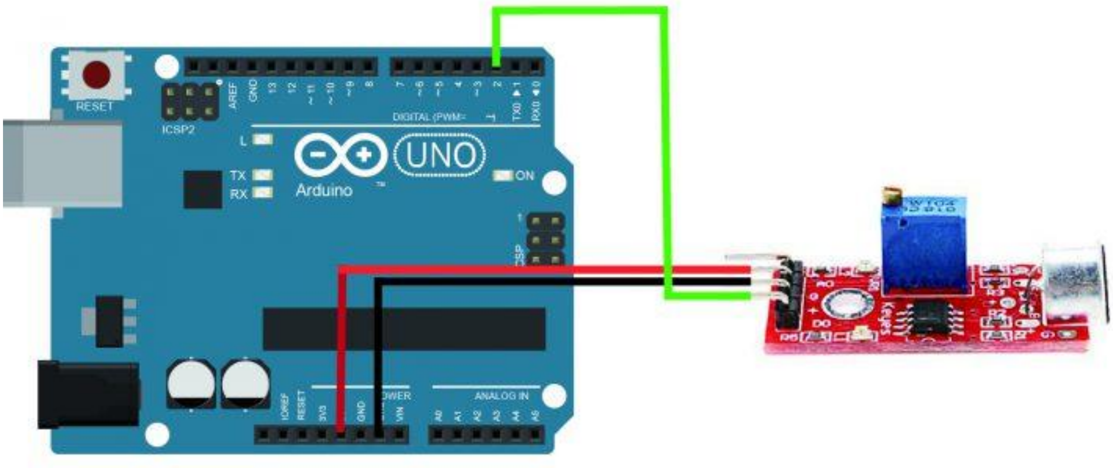
Working Description

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.

Circuit diagram:



Connections:

Sound sensor to Arduino board:

AO to A0

+ve pin to 5V

Gnd to gnd

LED to Arduino board:

+ve pin to pin 13

Gnd to gnd

Code:

const int ledPin = 13; //pin 13 built-in led

const int soundPin = A0; //sound sensor attach to A0

int threshold = 600; //Set minimum threshold for LED lit

void setup()

```
{
  pinMode(ledPin,OUTPUT);//set pin13 as OUTPUT
  Serial.begin(9600); //initialize serial
}
void loop()
{
  int value = analogRead(soundPin);//read the value of A0
  Serial.println(value);//print the value
  if(value > threshold) //if the value is greater than 600
  {
    digitalWrite(ledPin,HIGH);//turn on the led
    delay(200);//delay 200ms
  }
  else
  {
    digitalWrite(ledPin,LOW);//turn off the led
  }
  delay(1000);
```

Result:

Thus the sound detection system using Arduino was implemented successfully.

EX. NO: 10	INTERFACING FLEX SENSOR WITH ARDUINO
DATE :	

Aim:

To interface a flex sensor with Arduino board.

List of components:

1. Solderless Breadboard Full Size
2. Jumper Wires
3. Arduino UNO
4. LED
5. Flex Sensor

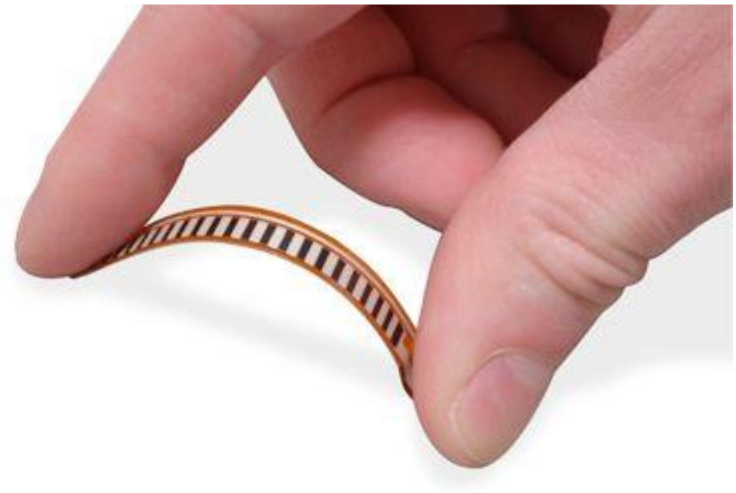
Working description:

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.

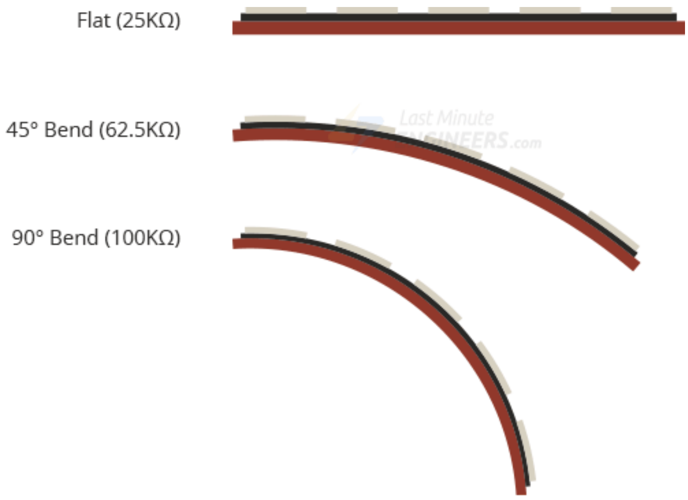
A flex-sensor could be used to check a door or window is opened or not. This sensor can be arranged at the edge of the door and once the door opens then this sensor also gets flexed. When the sensor bends then its parameters automatically change which can be designed to give an alert.

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.

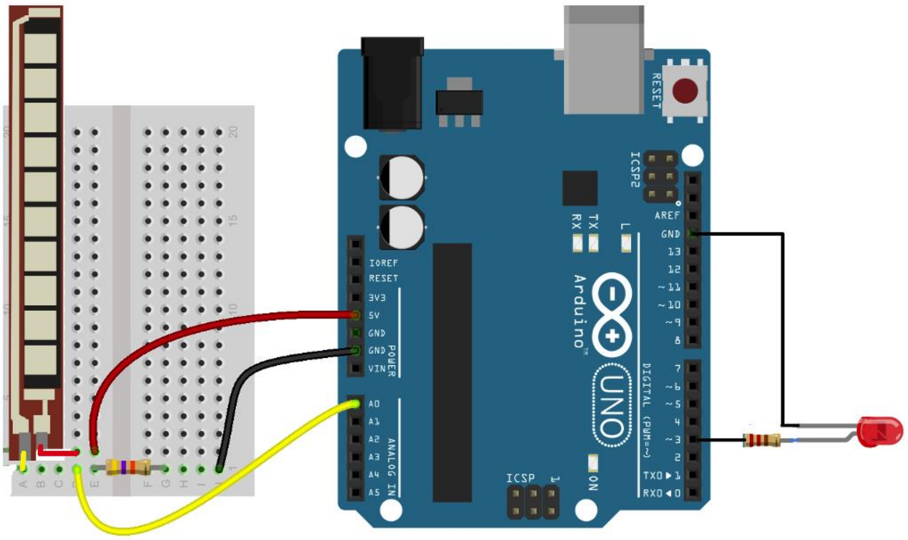


When the sensor is straight, this resistance is about 25k. When the sensor is bent, conductive layer is stretched, resulting in reduced cross section (imagine stretching a rubber band). This reduced cross section results in an increased resistance. At 90° angle, this resistance is about 100KΩ.



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.

Circuit diagram:



Connections:

Flex sensor to Arduino board

VCC (pin 2) to 5 V

Pin 1 to 10 kΩ resistor to GND

Pin 1 to pin A0

LED TO Arduino board

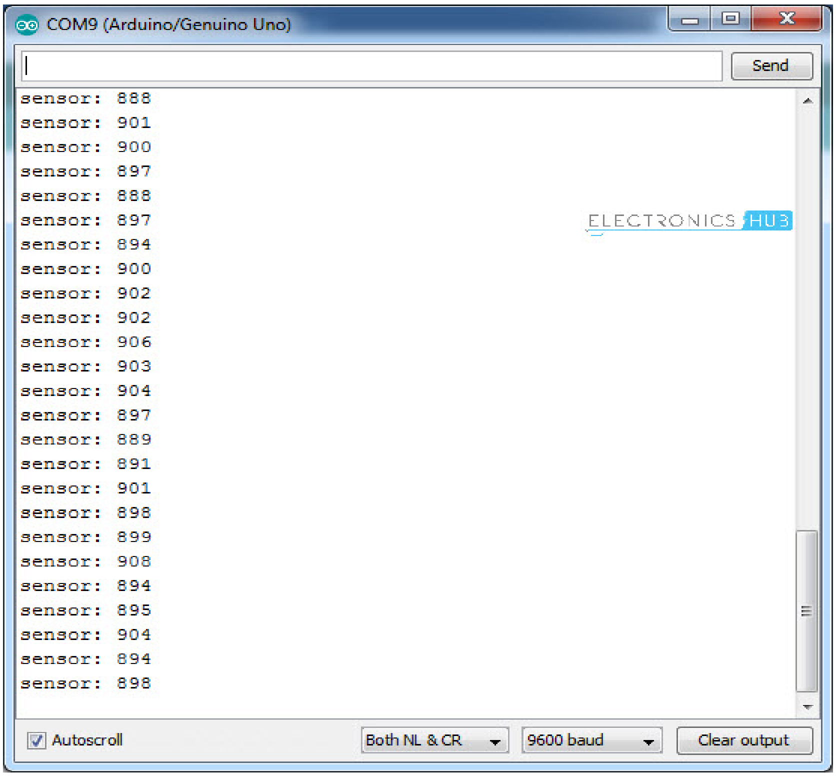
+ve pin to pin 3

-ve pin to GND

Code:

```
const int ledPin = 3;   //pin 3
const int flexPin = A0; //pin A0 to read analog input
int value; //save analog value
void setup(){
    pinMode(ledPin, OUTPUT);  //Set pin 3 as 'output'
  Serial.begin(9600);       //Begin serial communication
}
void loop(){
int flexValue;
  flexValue = analogRead(flexPin);
  Serial.print("sensor: ");
  Serial.println(flexValue);
    if(flexValue>890)
     digitalWrite(ledPin,HIGH);
  else
    digitalWrite(ledPin,LOW);
    delay(1000);
}
```

OUTPUT:



Result:

Thus the flex sensor was interfaced with Arduino successfully.

EX. NO: 11	INTERFACING FORCE PRESSURE SENSOR WITH ARDUINO
DATE :	

Aim:

To interface a force pressure sensor with Arduino board.

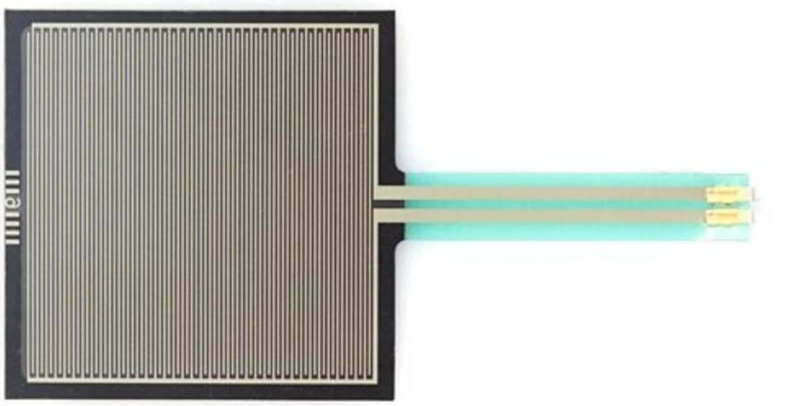
List of components:

1. Solderless Breadboard Full Size
2. Jumper Wires
3. Arduino UNO
4. Force pressure Sensor

Working description:

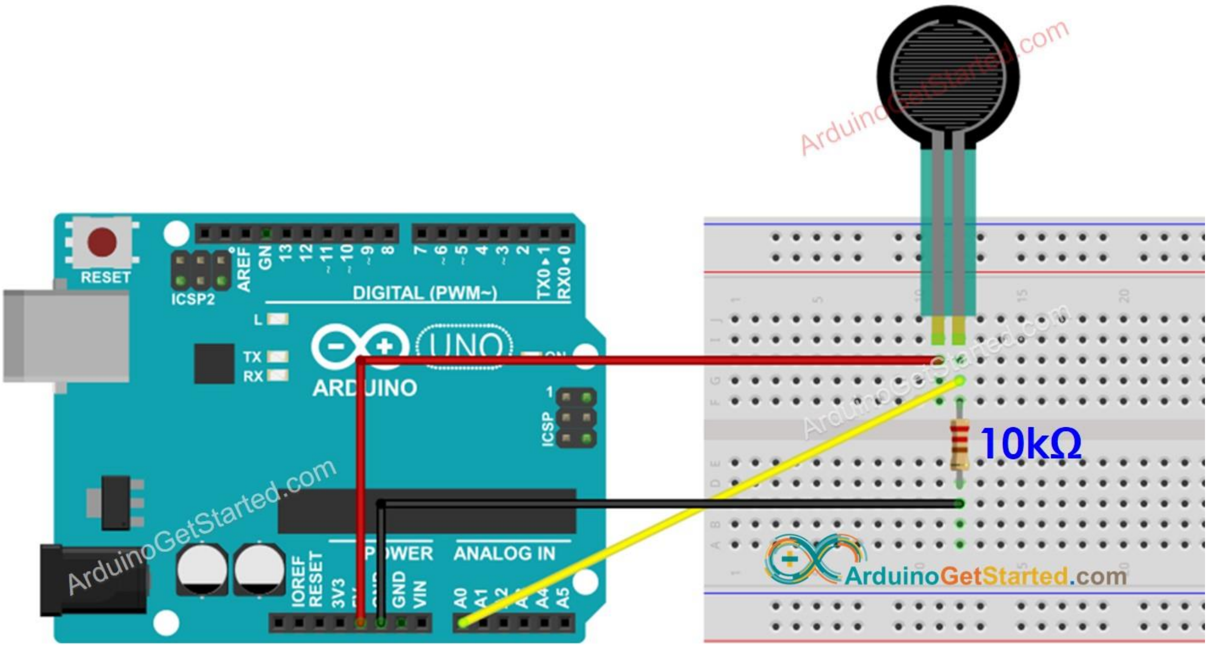
The working principle of a force sensor is that it responds to the applied force, as well as converts the value to a measurable quantity. Most force sensors are created with the use of force-sensing resistors. Such sensors consist of electrodes and sensing film.

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 behaviour on the road.

Circuit diagram:



Connections:

Flex sensor to Arduino board

VCC (pin 1) to 5 V

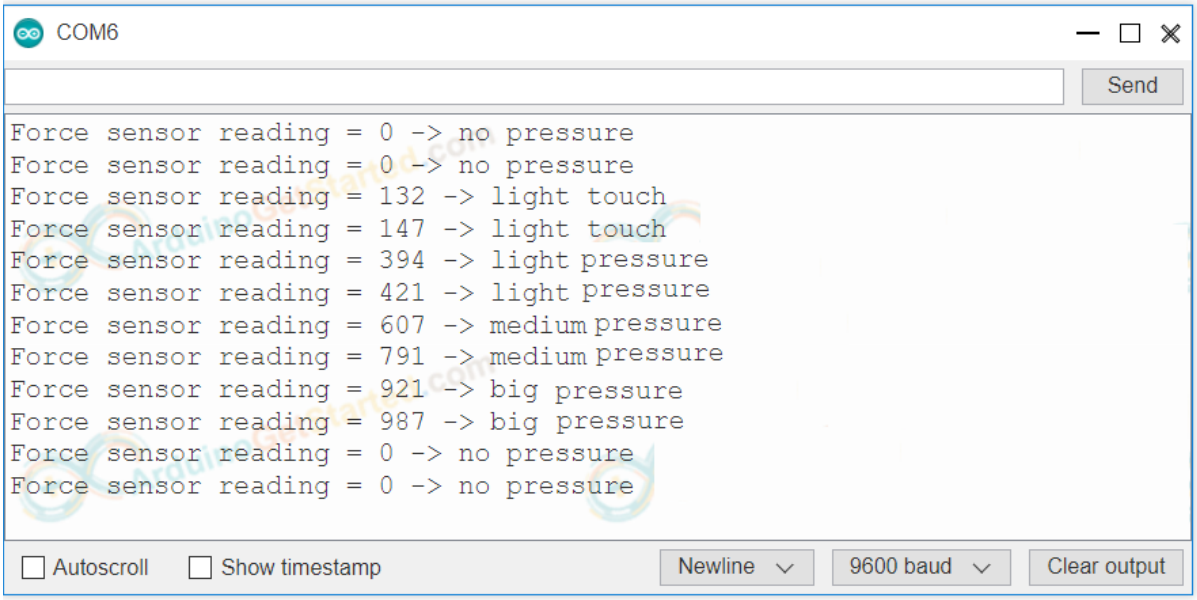
Pin 2 to 10 kΩ resistor to GND

Pin 2 to pin A0

Code:

```
#define FORCE_SENSOR_PIN A0
void setup() {
  Serial.begin(9600);
}
void loop() {
  int analogReading = analogRead(FORCE_SENSOR_PIN);
  Serial.print("Force sensor reading = ");
  Serial.print(analogReading); // print the raw analog reading
  if (analogReading < 10)       // from 0 to 9
    Serial.println(" -> no pressure");
  else if (analogReading < 200) // from 10 to 199
    Serial.println(" -> light touch");
  else if (analogReading < 500) // from 200 to 499
    Serial.println(" -> light pressure");
  else if (analogReading < 800) // from 500 to 799
    Serial.println(" -> medium pressure");
  else // from 800 to 1023
    Serial.println(" -> big pressure");
  delay(1000);
}
```

OUTPUT:



Result:

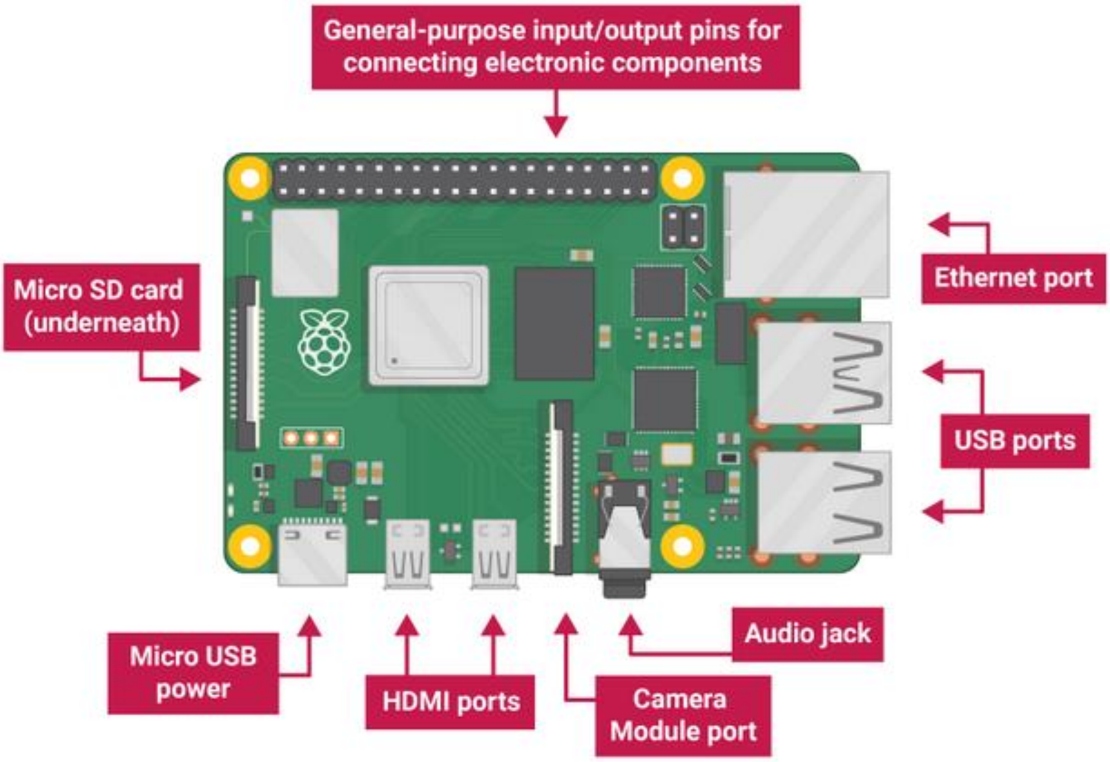
Thus the force pressure sensor was interfaced with Arduino successfully

Raspberry Pi

Introduction:

The Raspberry Pi is a low cost, **credit-card sized computer** that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It is capable of doing many tasks like browsing the internet and playing high-definition video, making spreadsheets, word-processing and playing games. The Raspberry Pi has the ability to interact with the outside world and has been used in a wide array of digital maker projects, from music machines and parent detectors to weather stations and tweeting birdhouses with infra-red cameras.

Pin configuration:



- **USB ports** — these are used to connect a mouse and keyboard. We can also connect other components, such as a USB drive.
- **SD card slot** — we can slot the SD card in here. This is where the operating system software and files are stored.

- **Ethernet port** — this is used to connect Raspberry Pi to a network with a cable. Raspberry Pi can also connect to a network via wireless LAN.
- **Audio jack** — we can connect headphones or speakers here.
- **HDMI port** — this is where we connect the monitor (or projector) that you are using to display the output from the Raspberry Pi. If the monitor has speakers, we can also use them to hear sound.
- **Micro USB power connector** — this is where we connect a power supply. We should always do this last, after we have connected all our other components.
- **GPIO ports** — these allow us to connect electronic components such as LEDs and buttons to Raspberry Pi.

Ex. No: 12	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. 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.

Note:

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 this commands have to be executed in cmd

   sudo apt-get update

   sudo apt-get install build-essential python-dev python-openssl git

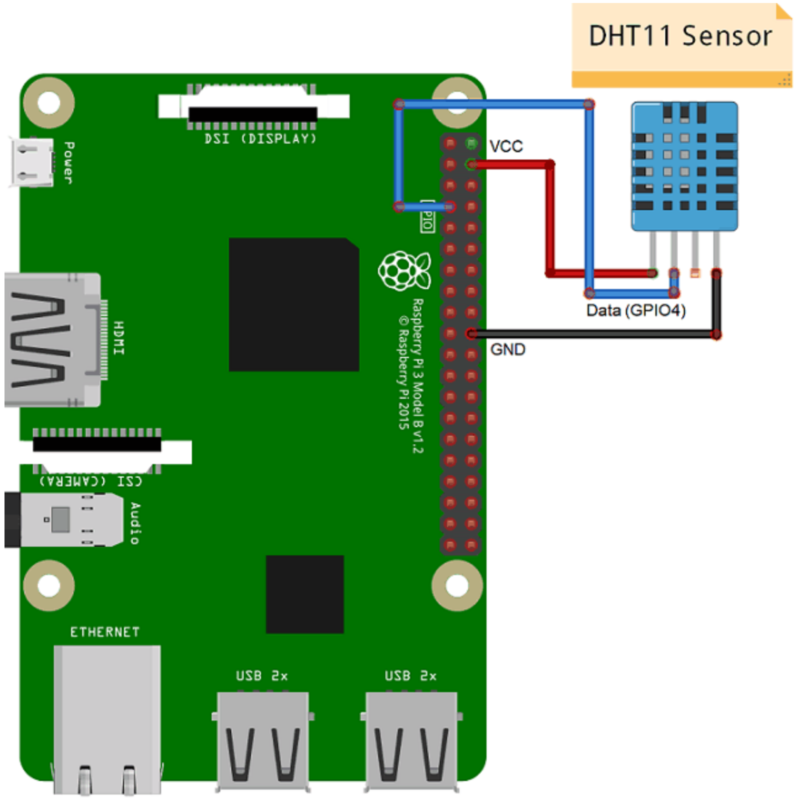
   git clone https://github.com/adafruit/Adafruit_Python_DHT.git && cd Adafruit_Python_DHT

   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

Circuit diagram:



Connections:

DHT11 sensor to Raspberry Pi

Vcc to pin 4 (5 V)

GND to GND

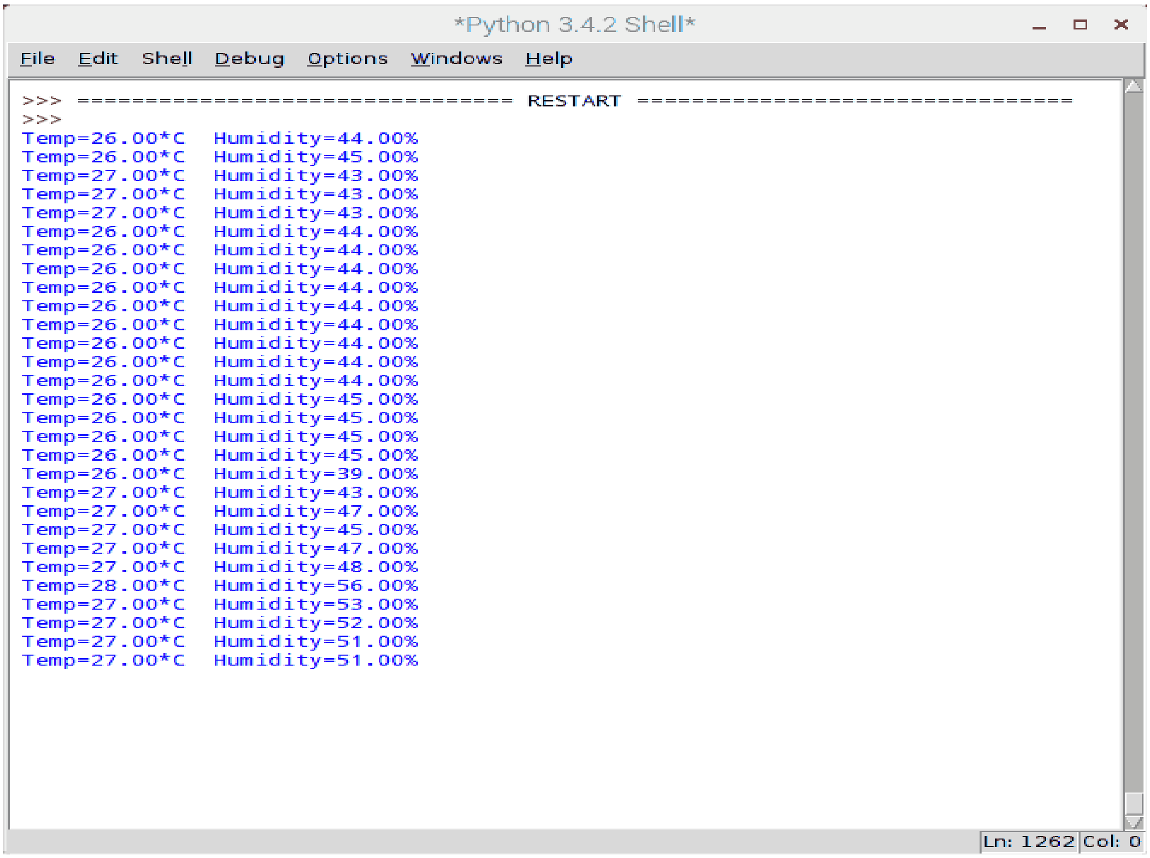
DATA pin to pin 7

Code:

```
import Adafruit_DHT
SENSOR=Adafruit_DHT.DHT11
PIN=4
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('fofjko')
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:



```
*Python 3.4.2 Shell*
File  Edit  Shell  Debug  Options  Windows  Help
>>> ================================ RESTART ================================
>>>
Temp=26.00*C  Humidity=44.00%
Temp=26.00*C  Humidity=45.00%
Temp=27.00*C  Humidity=43.00%
Temp=27.00*C  Humidity=43.00%
Temp=27.00*C  Humidity=43.00%
Temp=26.00*C  Humidity=44.00%
Temp=26.00*C  Humidity=44.00%
Temp=26.00*C  Humidity=44.00%
Temp=26.00*C  Humidity=44.00%
Temp=26.00*C  Humidity=44.00%
Temp=26.00*C  Humidity=44.00%
Temp=26.00*C  Humidity=44.00%
Temp=26.00*C  Humidity=44.00%
Temp=26.00*C  Humidity=44.00%
Temp=26.00*C  Humidity=45.00%
Temp=26.00*C  Humidity=45.00%
Temp=26.00*C  Humidity=45.00%
Temp=26.00*C  Humidity=45.00%
Temp=26.00*C  Humidity=39.00%
Temp=27.00*C  Humidity=43.00%
Temp=27.00*C  Humidity=47.00%
Temp=27.00*C  Humidity=45.00%
Temp=27.00*C  Humidity=47.00%
Temp=27.00*C  Humidity=48.00%
Temp=28.00*C  Humidity=56.00%
Temp=27.00*C  Humidity=53.00%
Temp=27.00*C  Humidity=52.00%
Temp=27.00*C  Humidity=51.00%
Temp=27.00*C  Humidity=51.00%
Ln: 1262 Col: 0
```

Result:

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

EX. NO: 13	PIR MOTION SENSOR INTERFACING WITH RASPBERRY PI
DATE :	

Aim:

To build motion detection system using PIR sensor.

List of components:

1. Raspberry Pi
2. LED
3. Jumper cable

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:

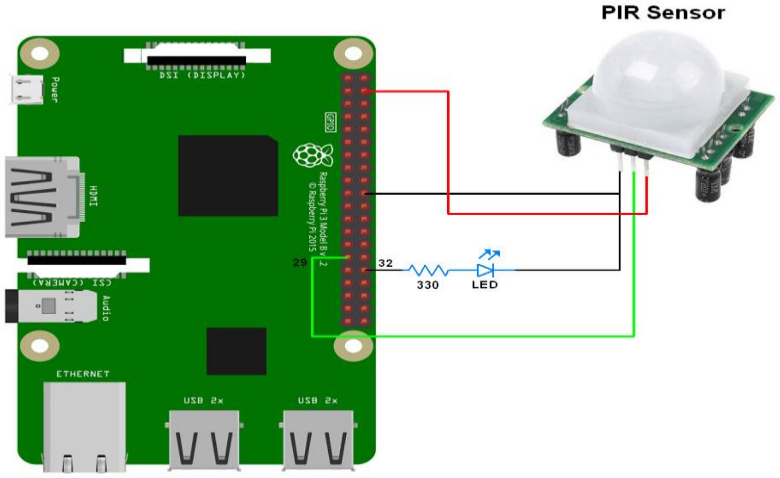
When motion is detected, PIR output goes HIGH which will be read by Raspberry Pi. So, we will turn on LED when motion is detected by PIR sensor. Here, LED is connected to GPIO12 (pin no. 32) whereas PIR output is connected to GPIO5 (pin no. 29).

Code:

```
import RPi.GPIO as GPIO

PIR_input=29

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BOARD)

GPIO.setup(PIR_input,GPIO.IN)

while True:
```

```
    if(GPIO.input(PIR_input)):
        print("Motion detected")
    else:
            print("MOtion not detected")
```

Circuit diagram:



Connections:

PIR sensor to Raspberry Pi:

VCC to VCC

GND to GND

DATA pin 29

Output:

Motion not detected..

Motion detected..

Motion not detected..

Motion not detected..

Result:

Thus the motion detector was built successfully using PIR sensor.

EX. NO: 14	SOUND SENSOR INTERFACING WITH RASPBERRY PI
DATE :	

Aim:

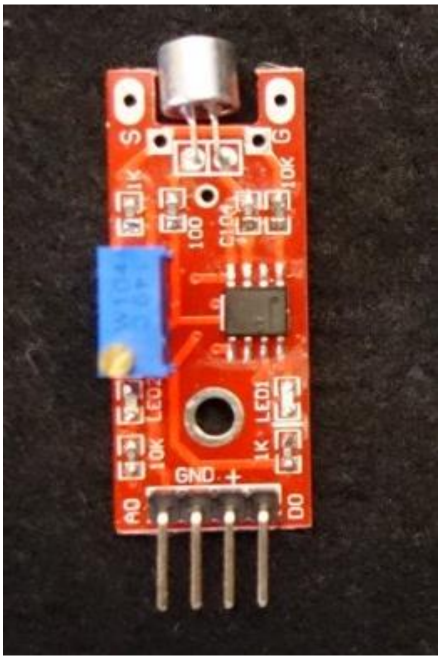
To interface Sound sensor with Raspberry Pi.

List of components:

1. Raspberry Pi
2. LED
3. Sound sensor

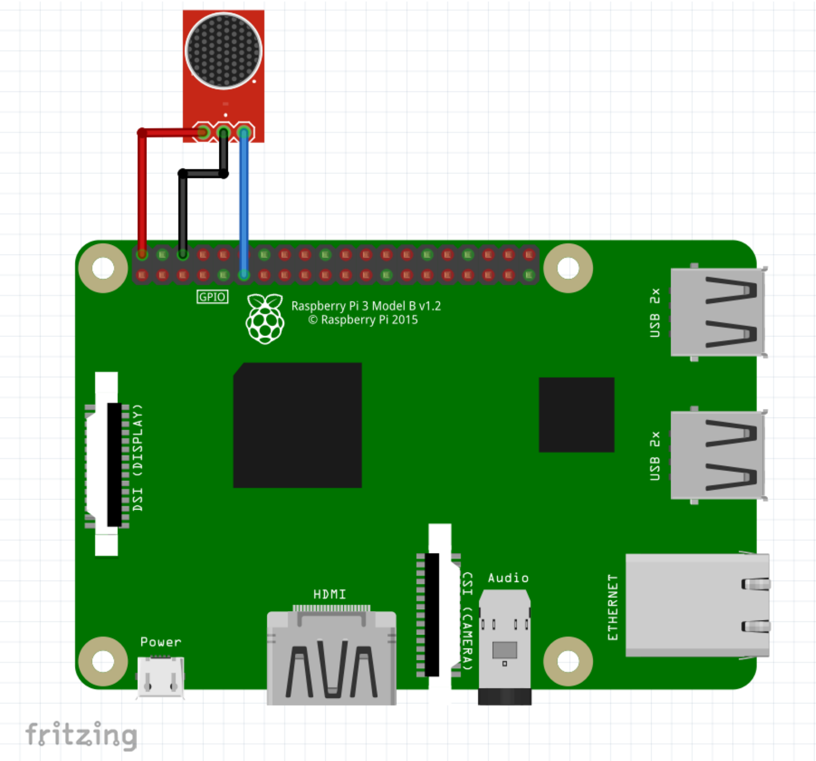
Working Description:

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 Sound sensor.



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.

Circuit diagram:



Connections

Sound sensor to Raspberry Pi

D0 to pin 11

VCC to VCC

GND to GND

LED to Raspberry Pi

+ve pin to pin 3

-ve pin to gnd

Code:

```
from time import sleep

import RPi.GPIO as GPIO

GPIO.setwarnings(False)
```

```
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11,GPIO.IN)
GPIO.setup(3,GPIO.OUT)
 while True:
    if(GPIO.input(11)== True):
        GPIO.output(3,False)
        print("NO sound");
        sleep(1)
    if(GPIO.input(11)== False):
        GPIO.output(3,True)
        print("sound");
        sleep(1)
```

Output:

No sound

No sound

Sound

Sound

Sound

Result:

Thus the sound sensor was interfaced with Raspberry Pi successfully

EX.NO: 15	**CLOUD BASED TEMPERATURE MONITOR WITH RASPBERRY PI**
DATE :	

Aim:

To write a python code for finding the Room temperature and humidity using Raspberry pi and show the outputs on the cloud system.

List of components:

1. Raspberry pi - full Kit
2. DHT sensor
3. Blynk IOT
4. Jumper wire

Working Description:

1. Connect the DHT sensor as shown in the circuit diagram.
2. Check the following python libraries are installed or not. If the libraries are not installed, open terminal and type pip3 install
<library Name (case sensitive)> and hit enter.
   a. blynk-library-python
   b. Adafruit_DHT
3. Type the given code and debug it.
4. Open browser and enter blynk.cloud and login your account. If you don't have a blynk account create new account and login again.
5. Then, go Templates section (on left-side bar) and select the new template (on right-side top corner). Enter the asked details and hit Done button.
6. Select the created template and go to Datastreams tab and select New Datastream and select virtual pin and enter the virtual pin name, Data type, Units (if needed), min, max and default value for the virtual pin. After all details are typed enter create button.
7. Create all virtual that program wants.

8. Then go to web dashboard select the necessary widgets and drop and drag in the dashboard. Select the widgets one by one, that widget show settings icon and click it choose the data stream for that widget. After widgets in the dashboard was fully configured click save and apply (top right-side corner) button. It will save your template and its configurations into cloud storage.
9. Then go to main dashboard using blynk.cloud/dashboard URL.
10. Click new device (top right-side corner) and click From template and choose the created template for template field and enter the name of the device and click create button.
11. Then go to Device Info section find the Auth token and copy It.
12. Go to python code and paste the copied auth token.
13. Run the Script.
14. Now the script connects to the cloud and shares the data to the cloud.
15. In Blynk web dashboard, we can see all the data that script sends.

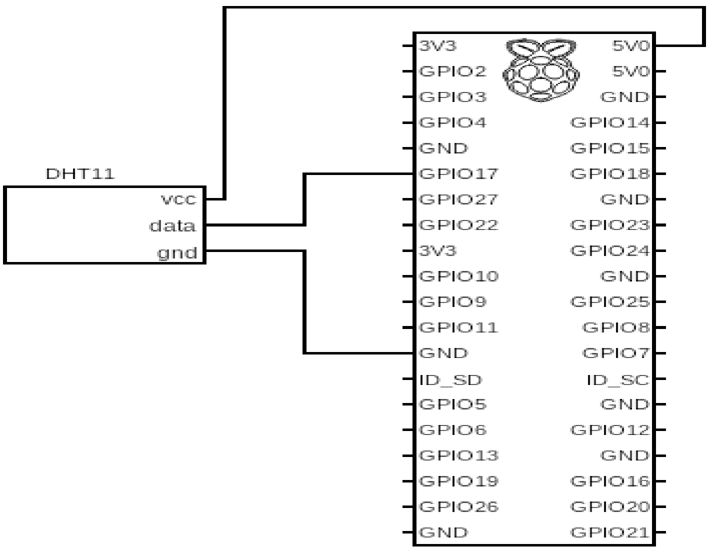
Connections:

Raspberry pi to DHT11 Sensor

Pin 2 (5v) to Vcc

Pin 11(GPIO17) to data or out Pin 25(Gnd) to Gnd

Circuit Diagram:

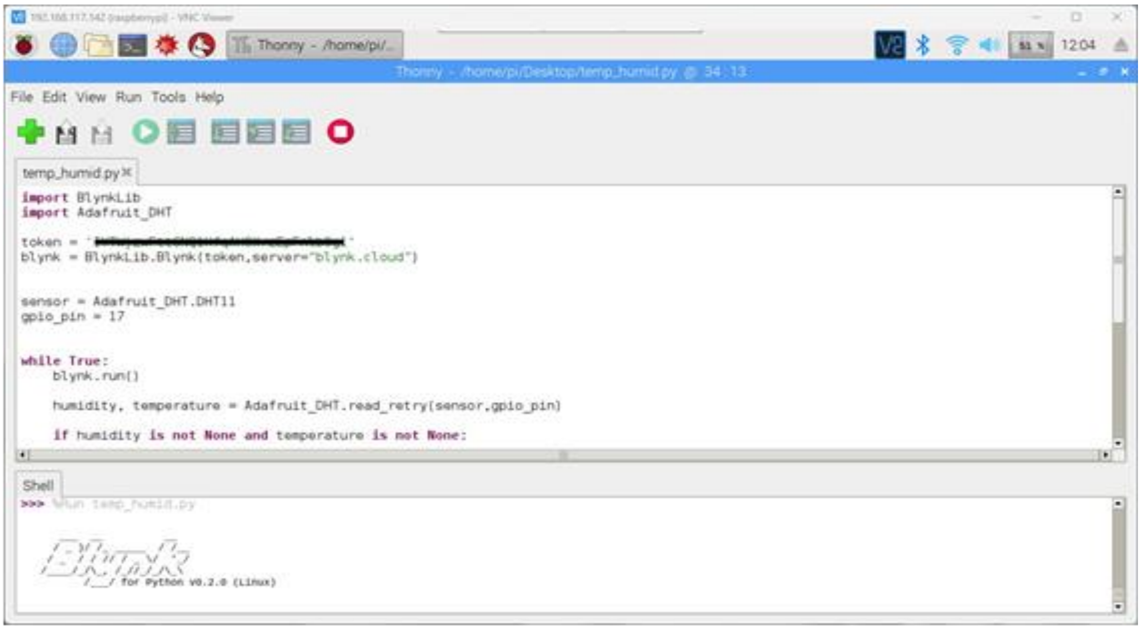


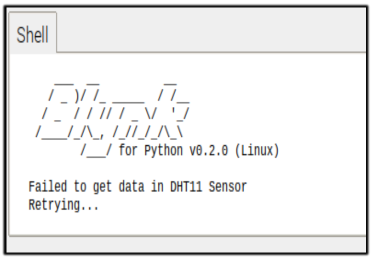
Note: The pin type maybe different from the dht11 sensor.

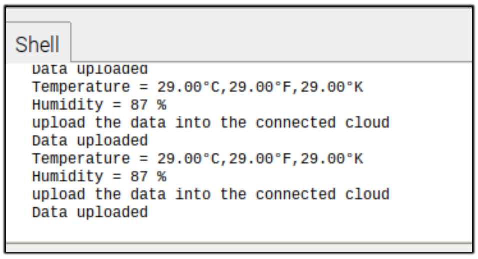
CODE:

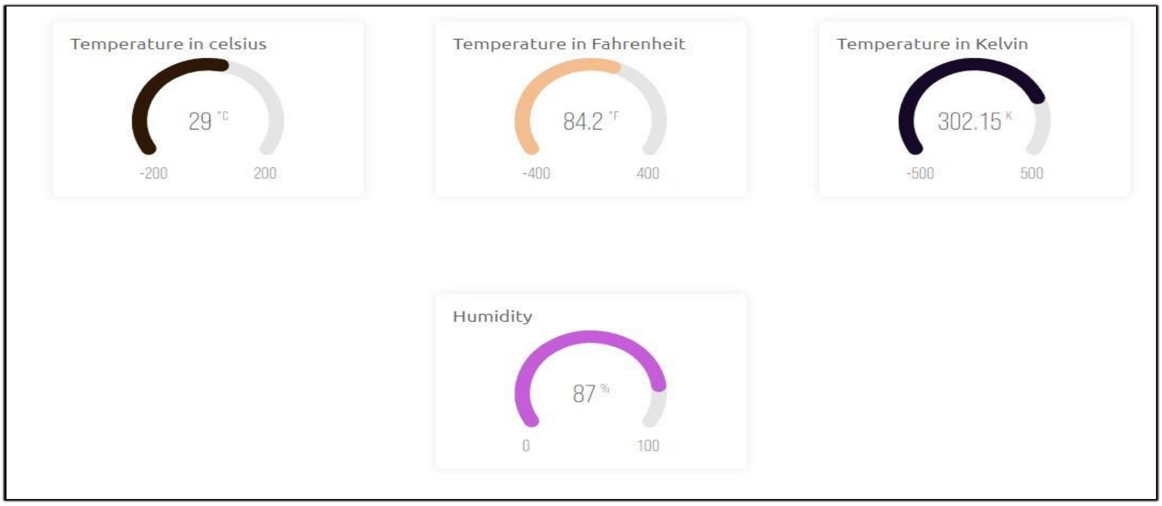
```
import  BlynkLib
import  Adafruit_DHT

token  =  '<paste  the  auth  token  here>'
blynk  =  BlynkLib.Blynk(token,server="blynk.cloud")

sensor  =  Adafruit_DHT.DHT11
gpio_pin  =  17

while  True:
        blynk.run()
        humidity, temperature  =  Adafruit_DHT.read_retry(sensor,gpio_pin)
        if  humidity  is  not  None  and  temperature  is  not  None:
                temperature=float('%.3f'%(temperature))
                temp_in_f  =  float('%.3f'%(temperature*1.8)+32)
                temp_in_k  =  float('%.3f'%(temperature+273.15))
                print('Temperature  =
{0:0.2f}°C,{0:0.2f}°F,{0:0.2f}°K'.format(temperature,temp_in_f,temp_in_k))
                print('Humidity  =',int(humidity),'%')
                print ('upload  the  data  into  the  connected  cloud')

                blynk.virtual_write(0,temperature)
                blynk.virtual_write(1,temperature_in_f)
                blynk.virtual_write(2,temperature_in_k)
                blynk.virtual_write(3,int(humidity))
                print("Data  uploaded")
        else:
                print("Failed  to  get  data  in  DHT11  Sensor\nRetrying...")
```

Sample Input /Output:









Result:

Thus, the python code was written successfully and room temperature and humidity was successfully shown in cloud system.

EX.NO:16	CLOUD BASED MOTION DETECTOR USING RASPBERRY PI
DATE :	

Aim:

Write a python code for finding the Room temperature and humidity using Raspberry pi and show the outputs on the IOT cloud.

Components Required:

1. Raspberry pi - full Kit
2. Motion sensor
3. Blynk IOT
4. Jumper wire
5. LED – 2

WORKING DESCRIPTION:

1. Connect the motion sensor and led as shown in the circuit diagram.
2. Check the following python libraries are installed or not. If the libraries are not installed, open terminal and type pip3 install

<library Name (case sensitive)> and hit enter.

a. blynk-library-python

3. Type the given code and debug it.
4. Open browser and enter blynk.cloud and login your account. If you don’t have a blynk account create new account and login again.
5. Then, go Templates section (on left-side bar) and select the new template (on right-side top corner). Enter the asked details and hit Done button.
6. Select the created template and go to Datastreams tab and select

New Datastream and select virtual pin and enter the virtual pin

name, Data type, Units (if needed), min, max and default value for the virtual pin. After all details are typed enter create button.

7. Create all virtual that program wants.

8. Then go to web dashboard select the necessary widgets and drop and drag in the dashboard. Select the widgets one by one, that widget show settings icon and click it choose the data stream for that widget. After widgets in the dashboard was fully configured click save and apply (top right-side corner) button. It will save your template and its configurations into cloud storage.

9. Then go to main dashboard using blynk.cloud/dashboard URL.

10. Click new device (top right-side corner) and click From template and choose the created template for template field and enter the name of the device and click create button.

11. Then go to Device Info section find the Auth token and copy It.

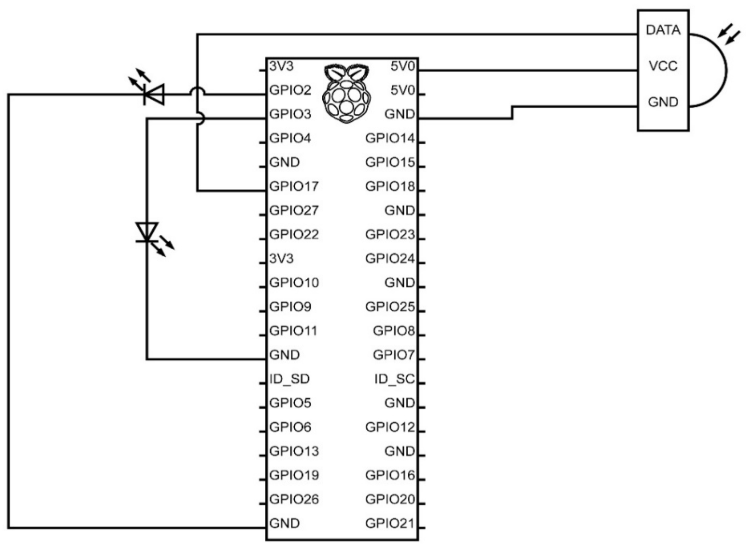
12. Go to python code and paste the copied auth token.

13. Run the Script.

14. Now the script connects to the cloud and shares the data to the cloud.

15. In Blynk web dashboard, we can see all the data that script sends.

CIRCUIT DIAGRAM:



Connections:

Raspberry pi to motion Sensor

Pin 2 (5v) to Vcc

Pin 11(GPIO17) to data or out

Pin 6(Gnd) to Gnd

Raspberry pi to LED (power Indicator)

Pin 3 (GPIO2) to +ve

Pin 39(Gnd) to -ve

Raspberry pi to LED (Motion Indicator)

Pin 5 (GPIO3) to +ve

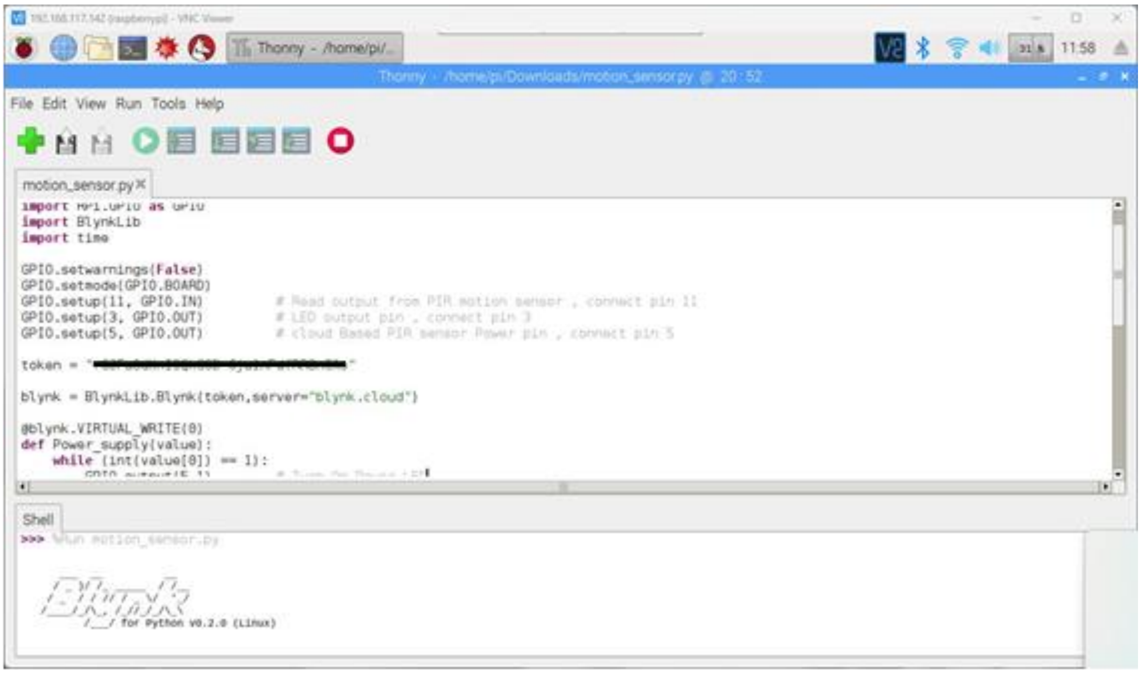
Pin 29(Gnd) to -ve

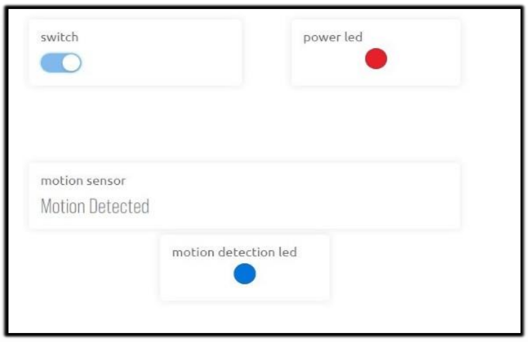
CODE:

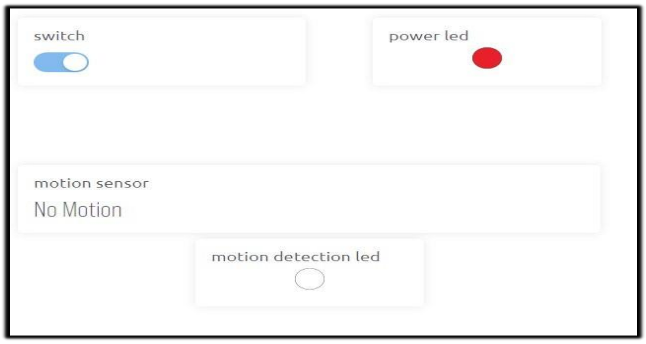
```
import  RPi.GPIO  as  GPIO
import  BlynkLib
import  time

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11,  GPIO.IN)
GPIO.setup(3,  GPIO.OUT)
 GPIO.setup(5,  GPIO.OUT)
token  =  "<Paste  the  Auth  Token  here>"
blynk  =  BlynkLib.Blynk(token,server="blynk.cloud")
 @blynk.VIRTUAL_WRITE(0)
```

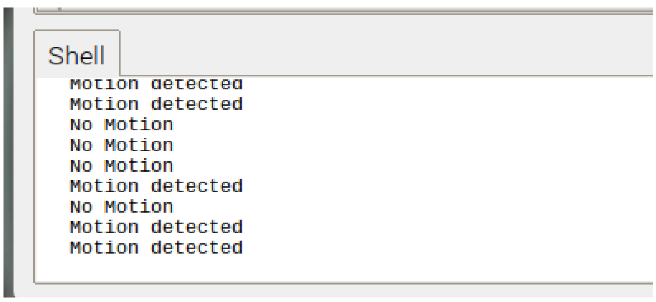
```
def Power_supply(value):
        while (int(value[0]) == 1):
                GPIO.output(5,1)        # Turn On Power LED
                blynk.virtual_write(2,1)
                i=GPIO.input(11)
                if i==0:     # When output from motion sensor is LOW
                        print("No Motion")
                        blynk.virtual_write(1,"No Motion")
                        GPIO.output(3, 0) # when motion is not detected ,Turn OFF LED
                        blynk.virtual_write(3,0)
                        time.sleep(0.1)
                elif i==1:   # When output from motion sensor is HIGH
                        print("Motion detected")
                        blynk.virtual_write(1,"Motion Detected")
                        GPIO.output(3, 1) # when motion is detected , Turn ON LED
                        blynk.virtual_write(3,1)
                        time.sleep(0.1)
                        blynk.sync_virtual(0)
                        break
                if(int(value[0]) == 0):
                        GPIO.output(5,0)
                        blynk.virtual_write(2,0)


while True:
        blynk.run()
```

Sample Input/Output:









RESULT:

Thus, the python code for motion detection was successfully executed and the outputs are shown in cloud system.