**GURU TEGH BAHADUR INSTITUTE OF TECHNOLOGY**


# INTRODUCTION TO INTERNET OF THINGS

**SUBJECT CODE: CIE-330P**

**SEMESTER – VI**

**Submitted to,**

**Ms. Shipra Raheja**

**Submitted by,**

**Name: Abhay Raj**
**Enrolment No: 00976803122**
**Branch: IT – 3 (FSD)**
**Batch: 2022-2026**

| S.No | Program | Date | Signature |
|---|---|---|---|
| 1. | To Study the Arduino UNO and installation of IDE software for Arduino. | | |
| 2. | Familiarisation with the concept of IoT and characteristics and the components to be used. | | |
| 3. | To demonstrate digital read/write function by blinking the built-in LED in the Arduino UNO. | | |
| 4. | To design and implement a traffic light system using an Arduino microcontroller that simulates real-world traffic signal operations. | | |
| 5. | To demonstrate analog read/write function by using potentiometer with the Arduino UNO. | | |
| 6. | To measure light intensity by using an LDR sensor with Arduino UNO. | | |
| 7. | To measure humidity and temperature by using a DHT11 sensor with Arduino UNO. | | |
| 8. | Design and implement a smoke level detector using MQ2 sensor using Arduino UNO. | | |
| 9. | To interface HC SR04 sensor with Arduino | | |
| 10. | To interface Bluetooth with Arduino and write a program to send sensor data to smartphone using Bluetooth. | | |
| 11. | Start Raspberry Pi and try various Linux commands in command terminal window : ls, cd, touch, mv, rm, man, mkdir, rmdir, tar, gzip, cat, more, less ,ps. sudo, cron, chgrp ping etc. | | |
| 12. | Run some python program on Pi like:<br>Read your name and print hello message with name<br>Read two number and print sum ,difference, multiplication and division.<br>Word and character count of a given string. | | |
| 13. | Run some python program on Pi like:<br>Print a name ‘n’ times where name and n are read from standard input using for and while loops, | | |

| | | | |
|---|---|---|---|
| | Handle Divided by Zero Exception.<br>Print current time for 10 times with an interval of 10 sec.<br>Read a file line and print the word count of each line. | | |
| 14. | Developing Projects to Solve Real-World Problems | | |

## Experiment – 1

## Aim: -

To Study the Arduino UNO and installation of IDE software for Arduino. Familiarisation with the concept of IoT and characteristics and the components to be used.

## THEORY:-

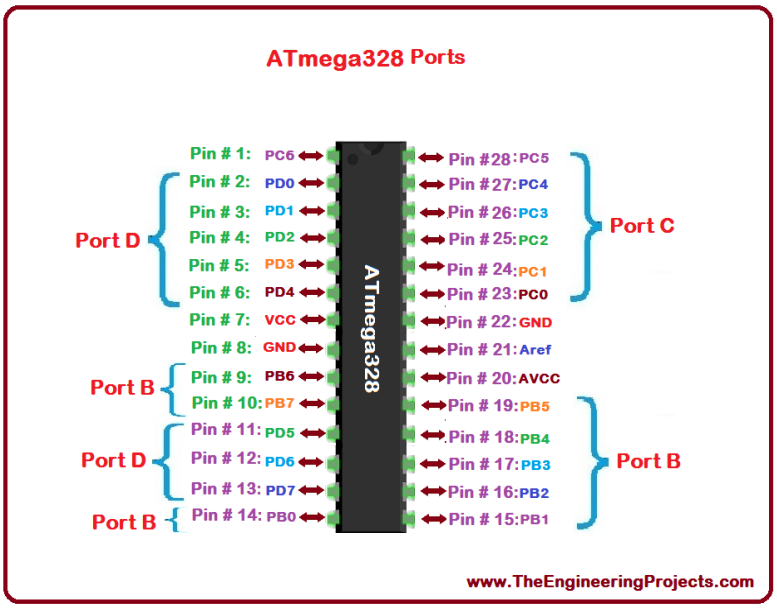
### Introduction to Arduino.

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message and turn it into an output activating a motor, turning on an LED, publishing something online.

### ATmega 328 AVR microprocessor

ATmega328 is an Advanced Virtual RISC (AVR) microcontroller. It supports 8-bit data processing. ATmega-328 has 32KB internal flash memory.

### ATmega328 Ports




ATmega328 has 1KB Electrically Erasable Programmable Read-Only Memory (EEPROM). This property shows if the electric supply supplied to the micro-controller is removed, even then it can store the data and can provide results after providing it with the electric supply. Moreover, ATmega-328 has 2KB Static Random Access Memory (SRAM). Other characteristics will be explained later. ATmega 328

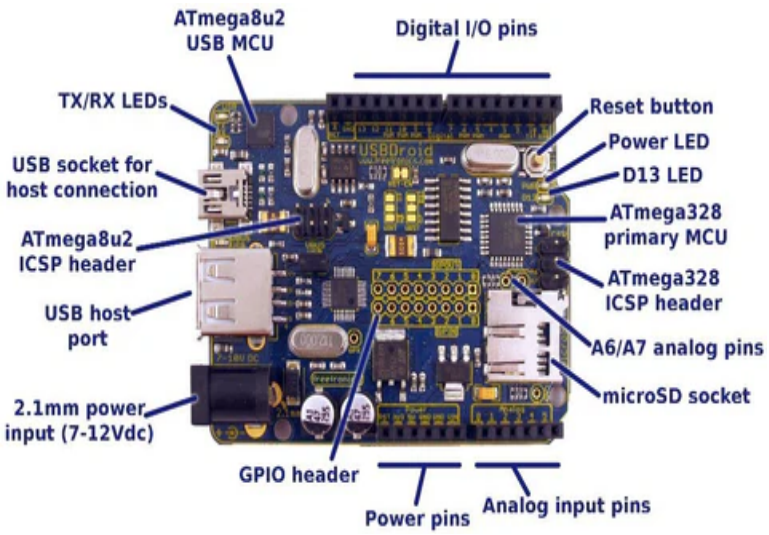
has several different features which make it the most popular device in today's market. These features consist of advanced RISC architecture, good performance, low power consumption, real timer counter having separate oscillator, 6 PWM pins, programmable Serial USART, programming lock for software security, throughput up to20 MIPS etc.

## Types of Arduino:-

### 1) Arduino Uno

The development of Arduino UNO board is considered as new compared to other Arduino boards.

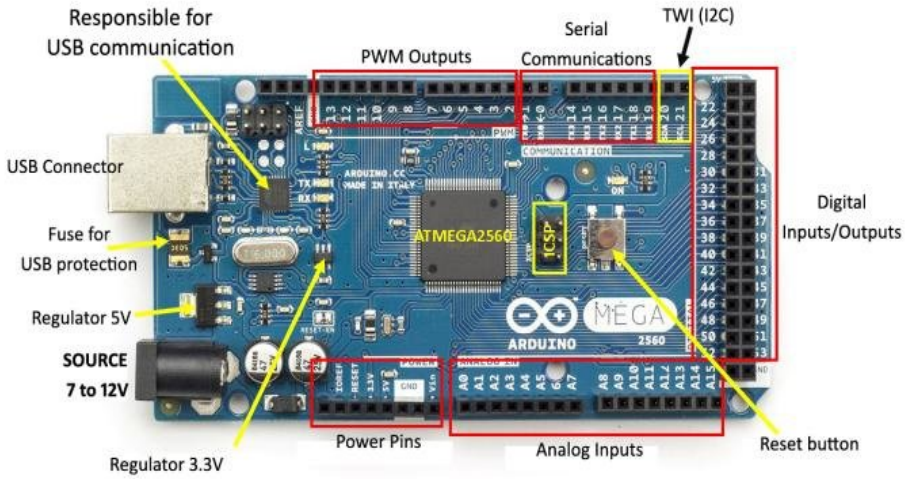
This board comes up with numerous features that helps the user to use this in their project. The Arduino UNO uses the Atmega16U2 microcontroller that helps to increase the transfer rate and contain large memory compared to other boards. No extra devices are needed for the Arduino UNO board like joystick, mouse, keyboard and many more. The Arduino UNO contain SCL and SDA pins and also have two additional pins fit near to RESET pin.




The board contains 14 digital input pins and output pins in which 6 pins are used as PWM, 6 pins as analog inputs, USB connection, reset button and one power jack. The Arduino UNO board can be attached to computer system buy USB port and also get power supply to board from computer system. The Arduino UNO contains flash memory of size 32 KB that is used to the data in it. The other feature of the Arduino UNO is compatibility with other shield and can be combined with other Arduino products.

## 2) Arduino Mega

This boards is considered as the microcontroller that uses the Atmega2560 in it. There are total 54 input pins and output pins in it in which 14 pins are of PWM output, 4 pins are of hardware port, 16 pins as analog inputs. The board also contain one USB connection, ICSP header, power jack and one REST pin

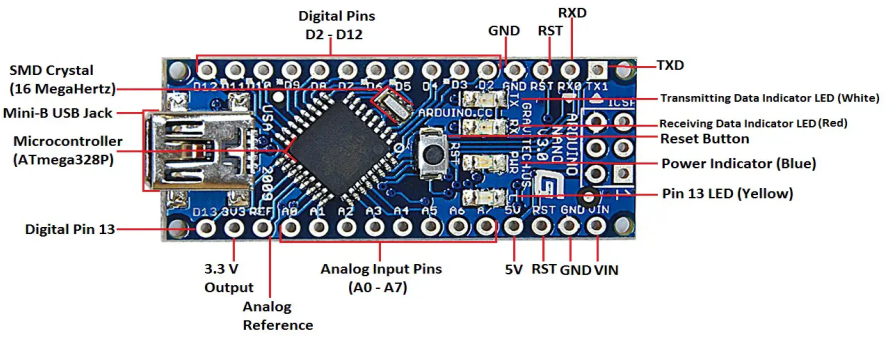



.

There are additional pins that act as crystal oscillator having frequency of 16 MHz. The board also has flash memory of 256KB size that uses to store the data in it. The Arduino Mega board can be attached to computer system via USB connection and power supply can be provided to board by using battery or AC to DC adapter. As the board has large number of pins fitted in it that make the board suitable for projects that requires more number of pins in it.

## 3) Arduino Nano

Arduino Nano are based on ATmega328P Microcontroller but Nano is significantly smaller in size compared to UNO. The Type-B USB connector from Arduino UNO is replaced with mini-B type connector. Also, there is no 2.1 mm DC jack to provide external power supply. Apart from that, the layout of Arduino Nano is very much self-explanatory.



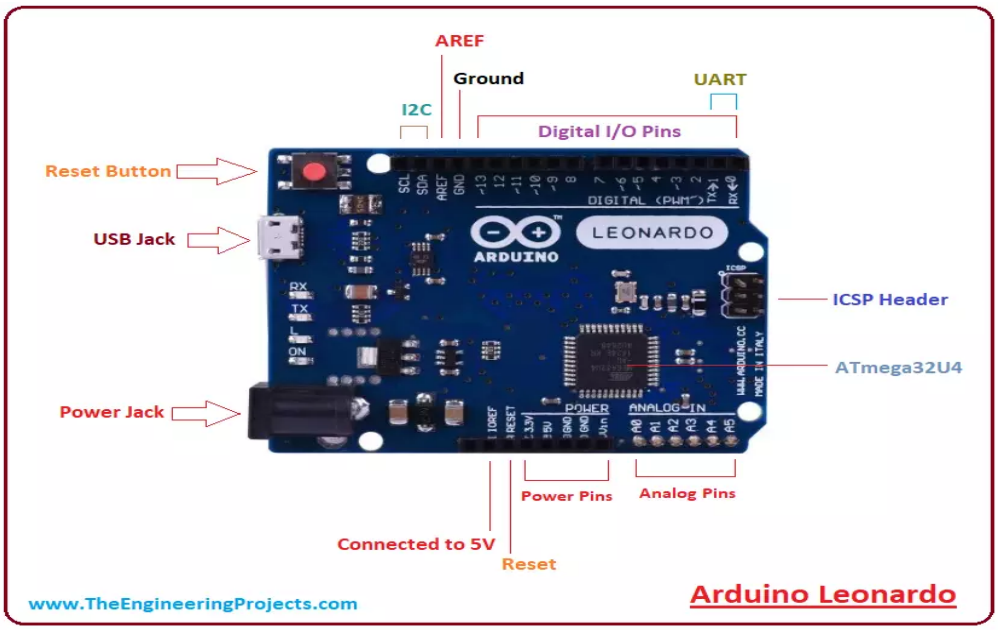

**Arduino Nano V3.0 Pinout**

www.CircuitsToday.com

Arduino Nano has 32 KB of Flash Memory,2 KB of SRAM,1 KB of EEPROM,2 KB of the Flash Memory is used by the bootloader code. 22 pins are associated with input and output. In that 14 pins (DO to D13) are true digital IO pins, which can be configured as per you application using pinMode(), digitalWrite() and digitalRead() functions. All these Digital 10 pins are capable of sourcing or sinking 40mA of current. An additional feature of the Digital 10 pins is the availability of internal pull-up resistor (which is not connected by default). The value of the internal pull-up resistor will be in the range of 20K to 50KO. There are also 8 Analog Input Pins (AO to A7). This is a couple more than Arduino UNO (which only has 6). All the analog. input pins provide a 10-bit resolution ADC feature, which can be read using analogRead() function.

**4) Arduino Leonardo**

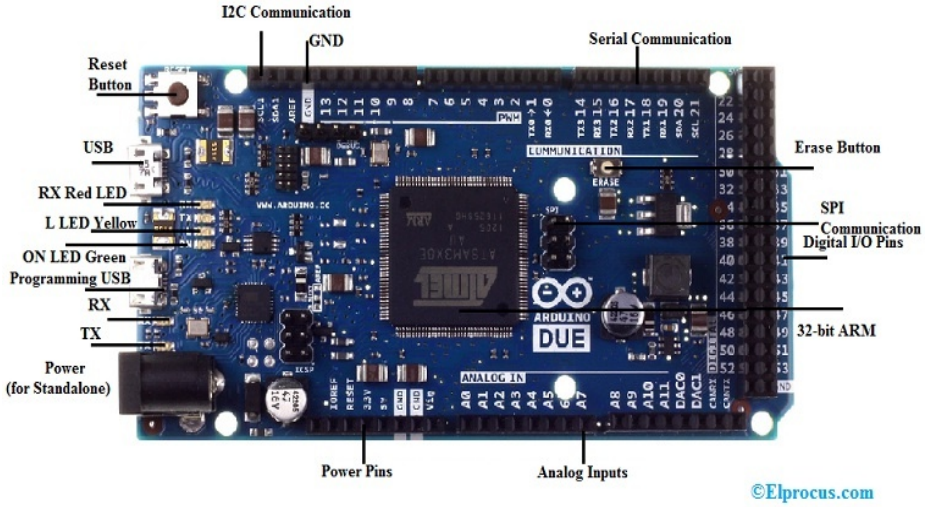
This board is considered as the microcontroller that uses the Atmega32u4 in it. There are total 20 digital input pins and output pins in it, in that 7 pins are used As PWM and 12 pins used as analog inputs. The board also contain one micro USB connection, power jack, and one RESET button fit in it. There are additional pins which act as crystal oscillator of frequency 16 MHz.




The Arduino Leonardo board can be attached to computer system via USB connection and power supply can be provided to board by using battery or AC to DC adapter. The microcontroller used by the Arduino Leonardo has in-built USB connection that removes the dependency of extra processor in it. As there is no additional USB connection in the board, it helps the board to act as mouse or keyboard for the computer system. The Arduino Leonardo is considered as cheapest Arduino boards compare to other Arduino products.

## 5) Arduino Due

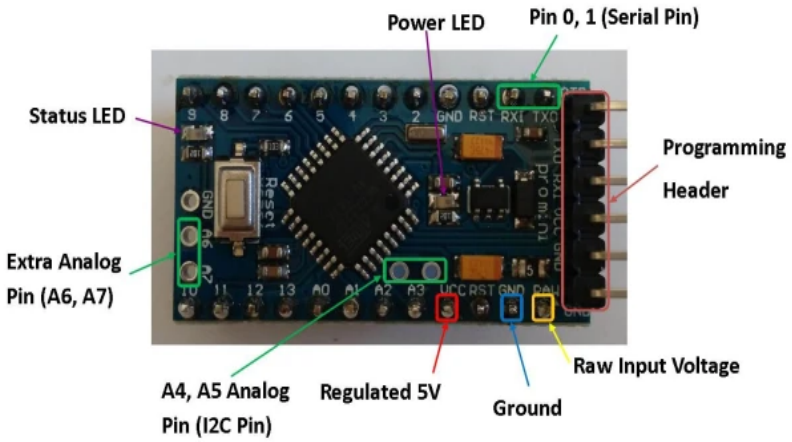
The Arduino Due is the first Arduino board based on a 32-bit ARM core microcontroller. With 54 digital input/output pins, 12 analog inputs, 2 DAC and 2 CAN it is the perfect board for powerful larger scale Arduino projects.




The Arduino Due is a microcontroller board based on the Atmel SAM3X8E ARM Cortex-M3 CPU. It is the first Arduino board based on a 32-bit ARM core microcontroller. It has 54 digital input/output pins (of which 12 can be used as PWM outputs), 12 analog inputs, 4 UARTS (hardware serial ports), a 84 MHz clock, an USB OTG capable connection, 2 DAC (digital to analog), 2 TWI, a power jack, an SPI header, a JTAG header, a reset button and an erase button.

## 6) Arduino Pro Mini

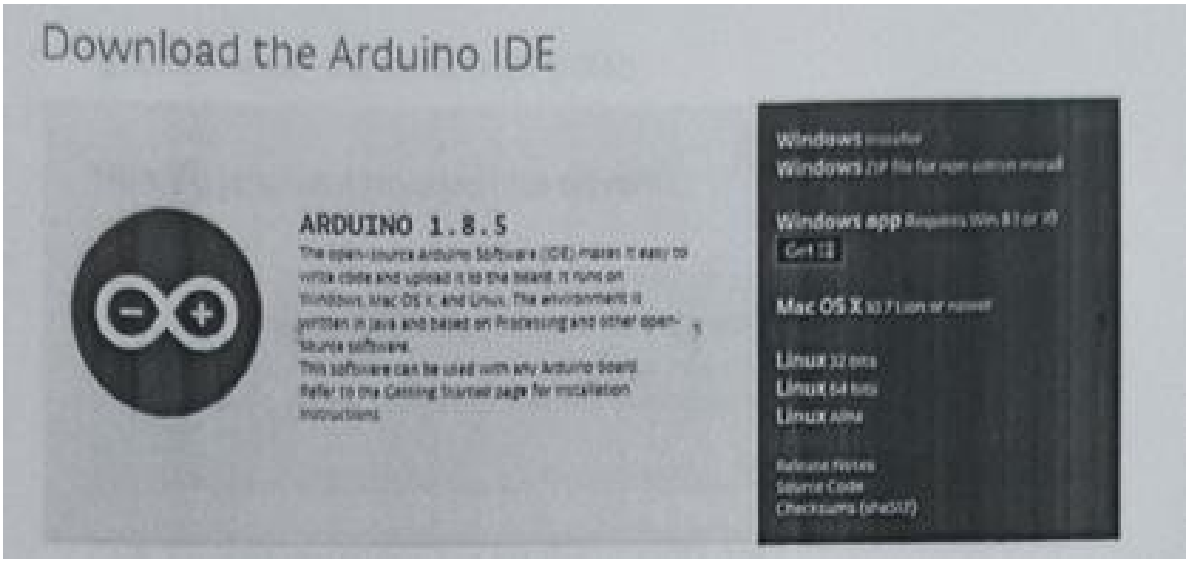
The Arduino Pro Mini is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, an on-board resonator, a reset button, and holes for mounting pin headers. A six pin header can be connected to an FTDI cable or Spark fun breakout board to provide USB power and communication to the board.



**Installation of Arduino**

**Step 1: Download and Install the IDE**

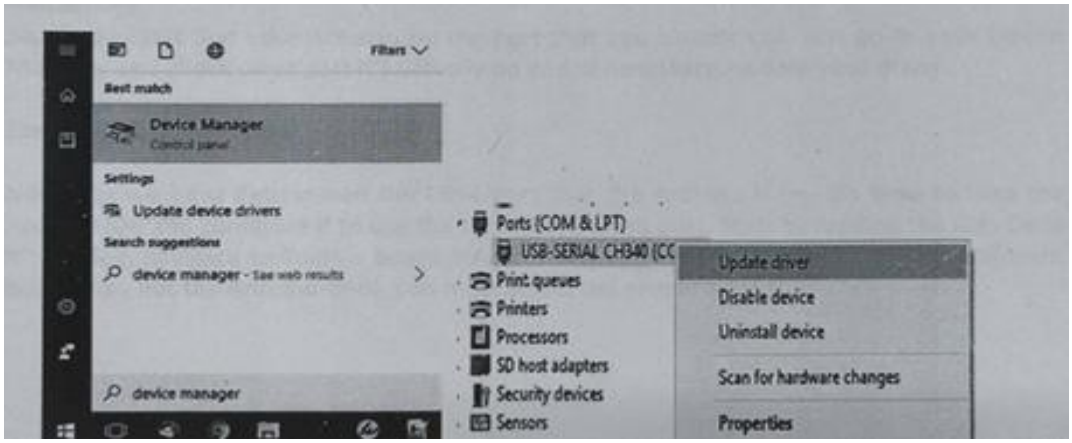
You can download the IDE from the official Arduino website. Since the Arduino uses a USB to serial converter (which allow it to communicate with the host computer), the Arduino board is compatible with most computers that have a USB port. Of course, you will need the IDE first. Luckily, the Arduino designers have released multiple versions of the IDE for different operating systems, including Windows, Mac, and Linux. In this tutorial, we will use Window 10, so ensure that you download the correct version of the IDE if you do not have Windows 10.




Once downloaded, install the IDE and ensure that you enable most (if not all) of the options, INCLUDING the drivers.

**Step 2: Get the Arduino COM Port Number**

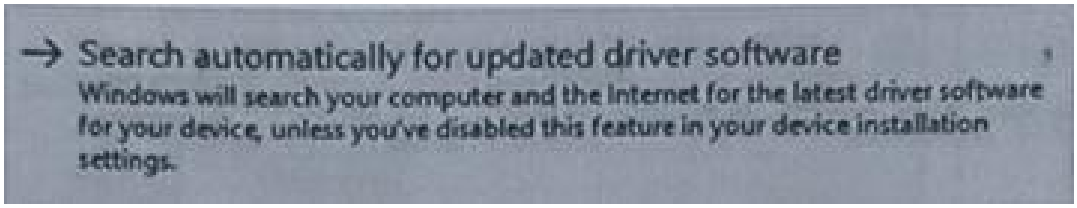
Next, you'll need to connect the Arduino Uno board to the computer. This is done via a USB B connection. Thanks to the wonderful world of USB, we do not need to provide power to the Arduino, as the USB provides 5V up to 2A. When the Arduino is connected, the operating system should recognize the board as a generic COM port (for example, my Arduino Uno uses a CH340G, which is an RS-232 serial to USB converter). Once it's recognized, we will need to find out what port number it has been assigned. The easiest way to do this is to type "device manager" into Windows Search and select Device Manager when it shows.




In the Device Manager window, look for a device under "Ports (COM & LPT)", and chances are the Arduino will be the only device on the list. In my Device Manager, the Arduino shows up as COM7 (I know this because CH340 is in the device name).

→ Update Drivers-USB-SERIAL CH340 (COMT)

**How do you want to search for drivers?**




→ Browse my computer for driver software

Locate and install driver software manually.

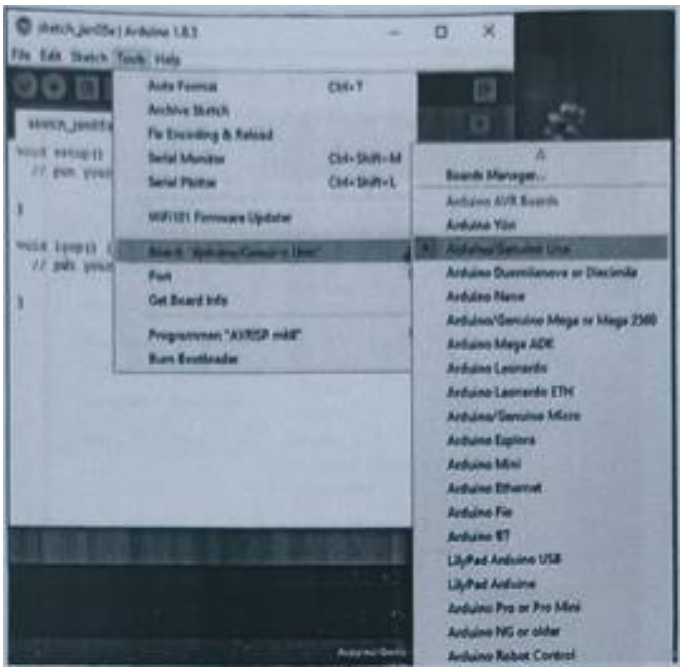
Be warned, the Arduino won't always be recognized automatically. If your Arduino is not recognized. then uninstall the driver, remove the Arduino, reinsert the Arduino, find the unrecognized device, right click "Update driver", and then click "Search automatically". This should fix 99 out of 100 problems.

Windows can be a real pain sometimes with COM ports, as it can magically change their numbers between connections. In other words, one day, your Arduino may be on port 7 (as shown here), but then on other days, Windows may shift it to a different port number. As I understand it, this happens when you connect other COM ports to your system (which I do frequently).

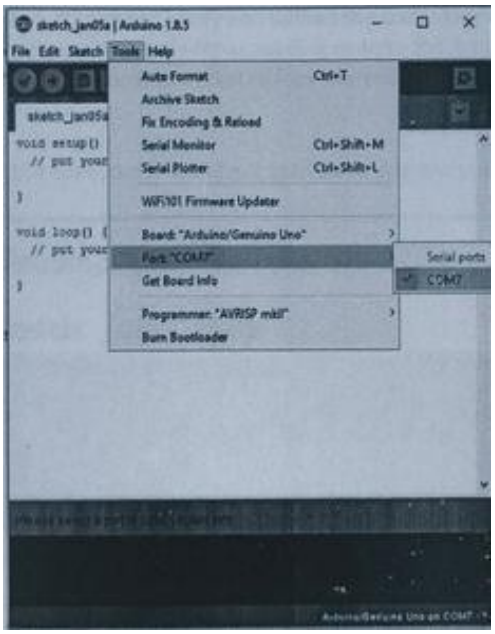
So, if you can't find your Arduino on the port that you usually use, just go to your Device Manager and check what port it's actually on and, if necessary, update your driver.

**Step 3: Configure the IDE**

Now that we have determined the COM port that the Arduino is on, it's time to load the Arduino IDE and configure it to use the same device and port. Start by loading the IDE. Once it's loaded, navigate to Tools > Board > Arduino Uno. However, if you are using a different board (i.e., not the Arduino Uno), you must select the proper board!

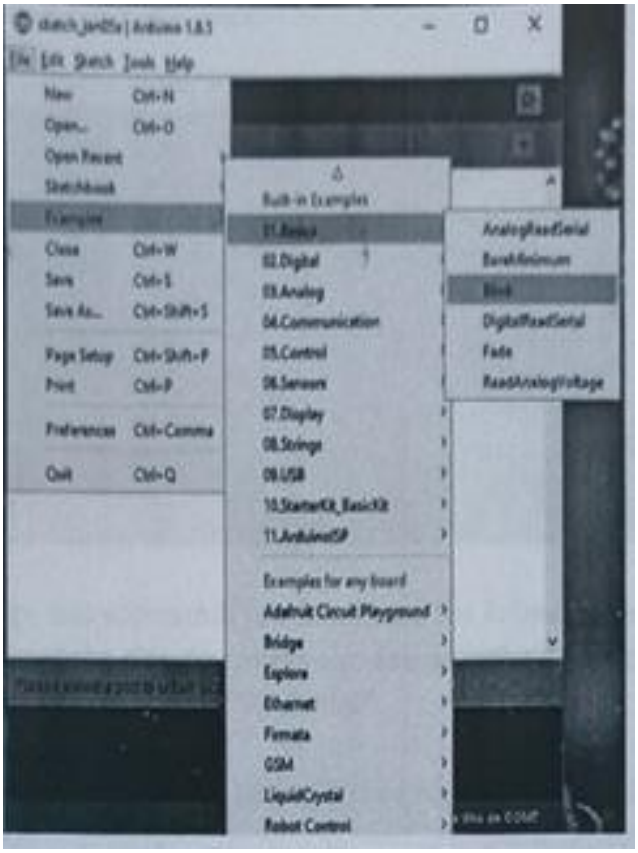



Next, you must tell the IDE which COM port the Arduino is on. To do this, navigate to Tools > Port > COM7. Obviously, if your Arduino is on a different port, select that port instead.



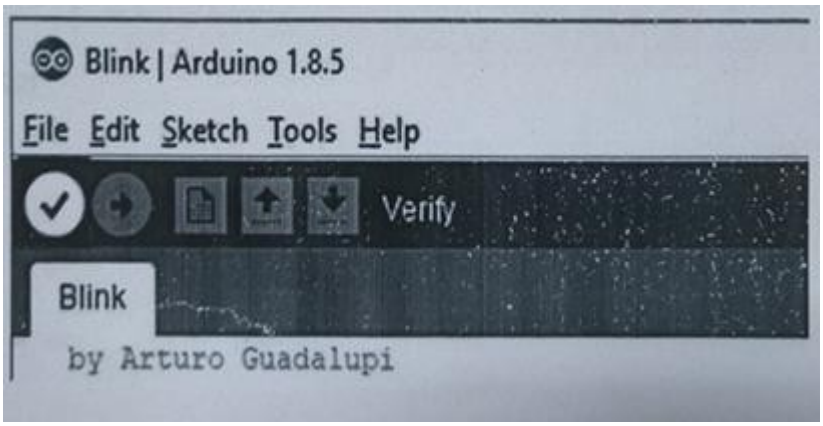

**Step 4: Loading a Basic Example**

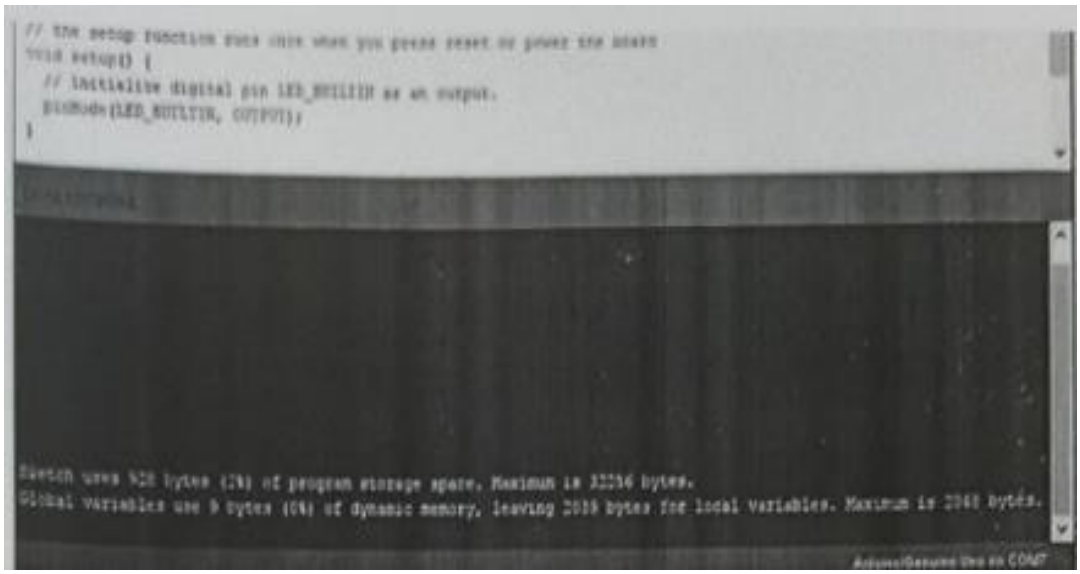
For the sake of simplicity, we will load an example project that the Arduino IDE comes with. This example will make the onboard LED blink for a second continuously. To load this example, click File > Examples > 01.Basics > Blink.



With the example loaded, it's time to verify and upload the code. The verify stage checks the code for errors, then compiles the ready-for-uploading code to the Arduino. The upload stage actually takes the binary data, which was created from the code, and uploads it to the Arduino via the serial port.

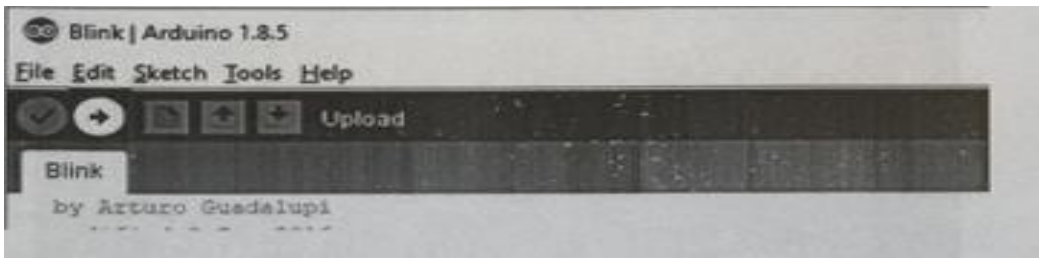
To verify and compile the code, press the check mark button in the upper left window.



If the compilation stage was successful, you should see the following message in the output window at the bottom of the IDE. You might also see a similar message-just it's one that does not have words like "ERROR" and "WARNING".

With the code compiled, you must now upload it the Arduino Uno. To do this, click the arrow next to the check mark.




CODE: -

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

void loop() {
  Serial.println("Hello, Arduino!");
  delay(1000);
}
```

Output:-

```
Hello, Arduino!
Hello, Arduino!
Hello, Arduino!
Hello, Arduino!
Hello, Arduino!
Hello, Arduino!
Hello, Arduino!
Hello, Arduino!
```

## Experiment - 2

**AIM**:-

Familiarisation with the concept of lot and characteristics and the components to be used.

**THEORY:-**

1. **Arduino Board**


The major components of Arduino UNO board are as follows:

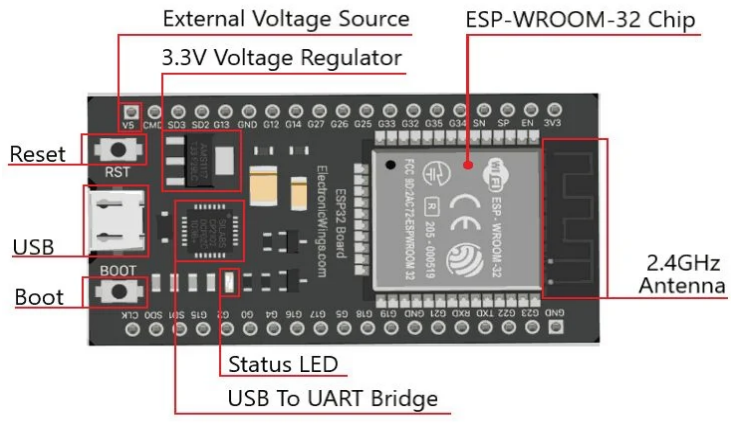
1. USB connector: This is a printer USB port used to load a program from the Arduino IDE onto the Arduino board. The board can also be powered through this port.
2. Power port: The Arduino board can be powered through an AC-to-DC adapter or a battery. The power source can be connected by plugging in a 2.1mm centre-positive plug into the power jack of the board.
3. Microcontroller: It is the most prominent black rectangular chip with 28 pins. Think of It as the brains of your Arduino. The microcontroller used on the UNO board is Atmega328P by Atmel (a major microcontroller manufacturer). Atmega328P is pre- programmed with bootloader. This allows you to directly upload a new Arduino program into the device, without using any external hardware programmer, making the Arduino UNO board easy to use.
4. Analog input pins: The Arduino UNO board has 6 analog input pins, labelled "Analog 0 to 5. These pins can read the signal from an analog sensor like a temperature sensor and convert it into a digital value so that the system understands. These pins just measure voltage and not the current because they have very high internal resistance. Hence, only a small amount of current flows through

these pins. Although these pins are labelled analog and are analog input by default, these pins can also be used for digital input or output.

5. Digital pins: You can find these pins labelled "Digital 0 to 13." These pins can be used as either input or output pins. When used as output, these pins act as a power supply source for the components connected to them. When used as input pins, they read the signals from the component connected to them. When digital pins are used as output pins, they supply 40 milliamps of current at 5 volts, which is more than enough to light an LED. These pins act as normal digital pins but can also be used for Pulse-Width Modulation (PWM), which simulates analog output like fading an LED in and out
6. ◦ Reset switch: When this switch is clicked, it sends a logical pulse to the reset pin of the Microcontroller, and now runs the program again from the start. This can be very useful if your code doesn't repeat, but you want to test it multiple times.
7. ◦ Crystal oscillator: This is a quartz crystal oscillator which ticks 16 million times a second. On each tick, the microcontroller performs one operation, for example, addition, subtraction, etc.
8. ◦ USB interface chip: Think of this as a signal translator. It converts signals in the USB level to a level that an Arduino UNO board understands.
9. ◦ TX RX LEDS: TX stands for transmit, and RX for receive. These are indicator LEDs which blink whenever the UNO board is transmitting or receiving data. Now that you have explored the Arduino UNO board, you have started your journey toward building your first lot prototype. In the next article, we will discuss Arduino programming and do a few experiments with Arduino and LEDs

**2. ESP-32**

The ESP32 family includes the chips ESP32-DOWDQ6 (and ESP32-DOWD), ESP32-D2WD, ESP32- SOWD, and the system in package (SIP) ESP32-PICO-D4. At its heart, there's a dual-core or single-core Tensilica Xtensa LX6 microprocessor with a clock rate of up to 240 MHz. ESP32 is highly integrated with built-in antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules. Engineered for mobile devices, wearable electronics, and IoT applications, ESP32 achieves ultra-low power consumption through power saving features including fine resolution clock gating, multiple power modes, and dynamic power scaling.



**Processors**:

Main processor: Tensilica Xtensa 32-bit LX6 microprocessor

Cores: 2 or 1 (depending on variation) All chips in the ESP32 series are dual-core except for ESP32

SOWD, which is single-core.

Clock frequency: up to 240 MHz 4) Performance: up to 600 DMIPS

Ultra-low power co-processor: allows you to do ADC conversions, computation, and level thresholds while in deep sleep.

**Wireless connectivity:**

1. **Wi-Fi**: 802.11 b/g/n/e/i (802.11n @ 2.4 GHz up to 150 Mbit/s)
2. **Bluetooth**: v4.2 BR/EDR and Bluetooth Low Energy (BLE)
3. **Memory**:

Internal memory: ROM: 448 KIB

For booting and core functions.

**SRAM**: 520 KIB

For data and instruction.

**Embedded flash:** Flash connected internally via 1016, 1017, SD_CMD, SD_CLK, SD_DATA_0 and SD_DATA_1 on ESP32-D2WD and ESP32-PICO-D4.

a) 0 MIB (ESP32-D0WDQ6, ESP32-DOWD, and ESP32-SOWD chips)

b) 2 MIB (ESP32-D2WD chip)

c) 4 MIB (ESP32-PICO-D4 SIP module)

1) **External flash & SRAM**: ESP32 supports up to four 16 MiB external QSPI flashes and SRAMs with hardware encryption based on AES to protect developers' programs and data. ESP32 can access the external QSPI flash and SRAM through high-speed caches.

a) Up to 16 MiB of external flash are memory-mapped onto the CPU code space, supporting 8-bit, 16-bit and 32-bit access. Code execution is supported. Prepared by-Neetu Settia (AP, ECE).

b) Up to 8 MiB of external flash/SRAM memory are mapped onto the CPU data space, supporting 8bit, 16-bit and 32-bit access. Data-read is supported on the flash and SRAM. Data-write is supported on the SRAM.

ESP32 chips with embedded flash do not support the address mapping between external flash and peripherals.

4) **Peripheral Input/output**; Rich peripheral interface with DMA that includes capacitive touch, ADCs (analog-to-digital converter), DACs (digital-to-analog converter), IC (Inter- Integrated Circuit), UART (universal asynchronous receiver/transmitter), CAN 2.0 (Controller Area Network), SPI (Serial Peripheral interface), I'S (Integrated Inter-IC Sound), RMII (Reduced Media-independent interface), PWM (pulse width modulation), and more.

5) **Security:**

a) IEEE 802.11 standard security features all supported, including WFA, WPA/WPA2 and WAPI b)

Secure boot

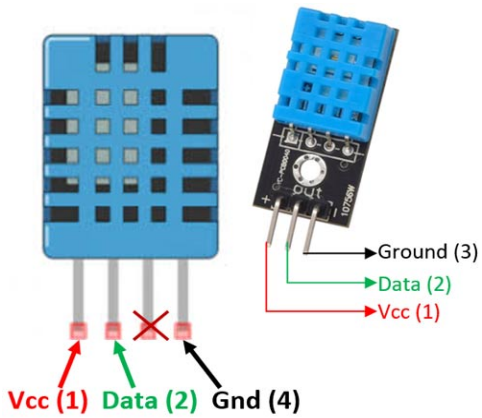
c) Flash encryption

d) 1024-bit OTP, up to 768-bit for customers

e) Cryptographic hardware acceleration: AES, SHA-2, RSA, elliptic curve cryptography (ECC), random number generator (RNG) 3) Sensors

The sensors are defined as a machine, module, or a device that detect changes in the environment. The sensors transfer those changes to the electronic devices in the form of a signal. The sensor is a device, which is made up of Single Crystal Silicon. It is considered as a widely used semiconductor material. It has superior mechanical stability, machinability, etc. It can also combine electronics and sensing elements on the same substrate.

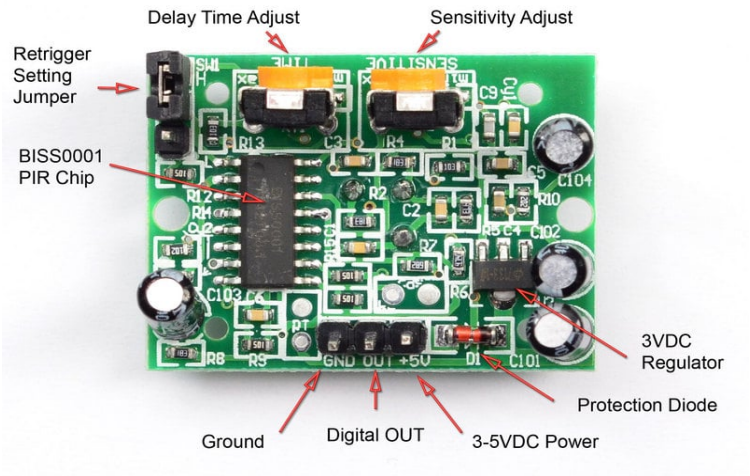
- **Temp and Humidity sensor**




The DHT11 is a commonly used Temperature and humidity sensor. The sensor comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data. The sensor is also factory calibrated and hence easy to interface with other microcontrollers. The sensor can measure temperature from 0°C to 50°C and humidity from 20% to 90% with an accuracy of ±1°C and ±1%. So if you are looki to measure in this range then this sensor might be the right choice for you. Prepared by-Neetu Settia (AP, ECE).

- **PIR sensor**

Passive infrared (PIR) sensors use a pair of pyroelectric sensors to detect heat energy in the surrounding environment. These two sensors sit beside each other, and when the signal differential between the two sensors changes (if a person enters the room, for example), the sensor will engage. That may mean it triggers an alarm, notifies authorities, or maybe turns on a floodlight. IR radiation focuses on each of the two pyroelectric sensors using a series of lenses constructed as the sensor's housing. These lenses widen the device's sensing area.



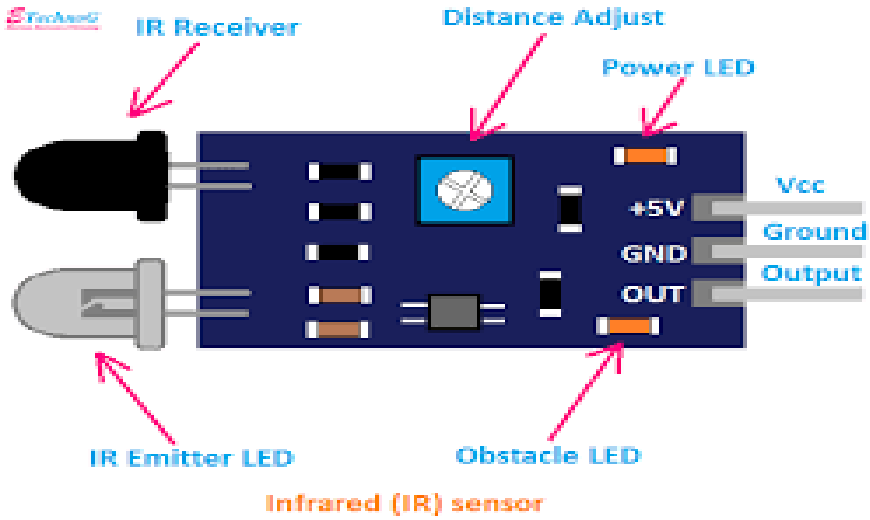

While the lens setup and sensor electronics are sophisticated technology, these units are easy to use in a practical application. You only need power and ground for the sensor to produce a discreet output that's strong enough for a microcontroller to use. Typical adjustments include adding potentiometers for sensitivity and tweaking how long a PIR stays engaged once it's triggered. You can also toggle the sensor between:

1. Staying on for a set amount of time after detecting movement.

2. Pulsing on and off in a "non-retriggering" mode.

- **IR Sensor**

IR technology is used in daily life and also in industries for different purposes. For example, TVs use an IR sensor to understand the signals which are transmitted from a remote control. The main benefits of iR sensors are low power usage, their simple design & their convenient features. IR signals are not noticeable by the human eye. The IR radiation in the electromagnetic spectrum can be found in the regions of the visible & microwave. Usually, the wavelengths of these waves range from 0.7 µm 5 to 1000µm. The IR spectrum can be divided into three regions like near-infrared, mid, and farinfrared. The near IR region's wavelength ranges from 0.75-3µm, the mid-infrared region's wavelength ranges from 3 to 6um & the far IR region's infrared radiation's wavelength is higher than 6um.




IR sensors are classified into different types depending on the applications. Some of the typical applications of different types of sensors. The speed sensor is used for synchronizing the speed of multiple motors. The temperature sensor is used for industrial temperature control. PIR sensor is used for an automatic door opening system and the Ultrasonic sensor is used for distance measurement.

- **Ultrasonic sensor**

HC-SR04 Ultrasonic (US) sensor is a 4-pin module, whose pin names are Vcc, Trigger, Echo and Ground respectively. This sensor is a very popular sensor used in many applications where measuring distance or sensing objects are required. The module has two eyes like projects in the front which forms the Ultrasonic transmitter and Receiver.

The Ultrasonic transmitter transmits an ultrasonic wave, this wave travels in air and when it gets objected by any material it gets reflected back toward the sensor this reflected wave is observed by the Ultrasonic receiver module.

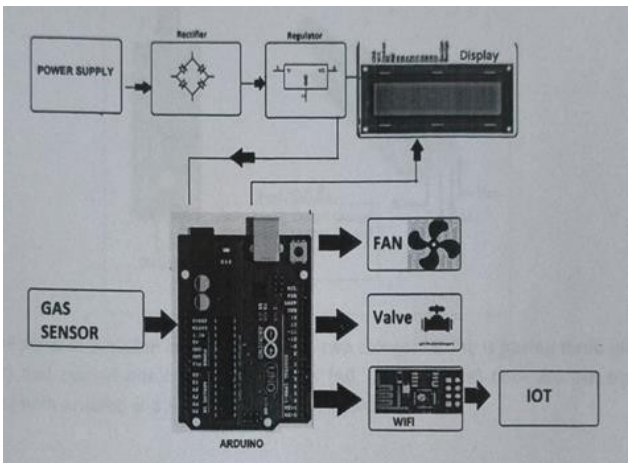
Now, to calculate the distance using the above formulae, we should know the Speed and time. Since we are using the Ultrasonic wave we know the universal speed of US wave at room.

conditions which is 330m/s. The circuitry inbuilt on the module will calculate the time taken for the US wave to come back and turns on the echo pin high for that same particular

amount of time, this way we can also know the time taken. Now simply calculate the distance using a microcontroller or microprocessor.

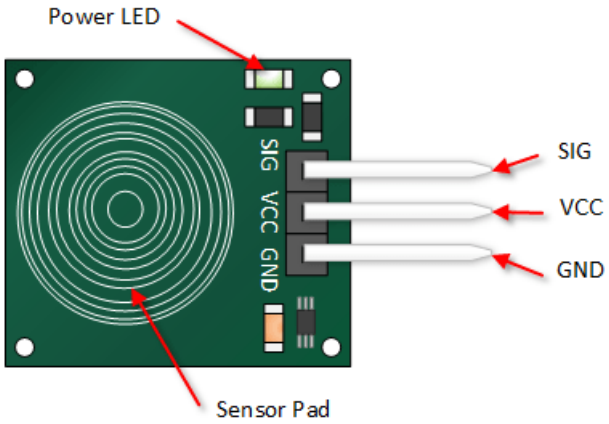
- **Gas Sensor**

A gas sensor is mainly used to detect multiple gasses in the atmosphere. It is a device that responds to the concentration of gasses (in parts per million or ppm) in the environment where it is installed. The device works by creating a potential difference by altering the material's internal sensor's resistance.




- **Touch sensor**

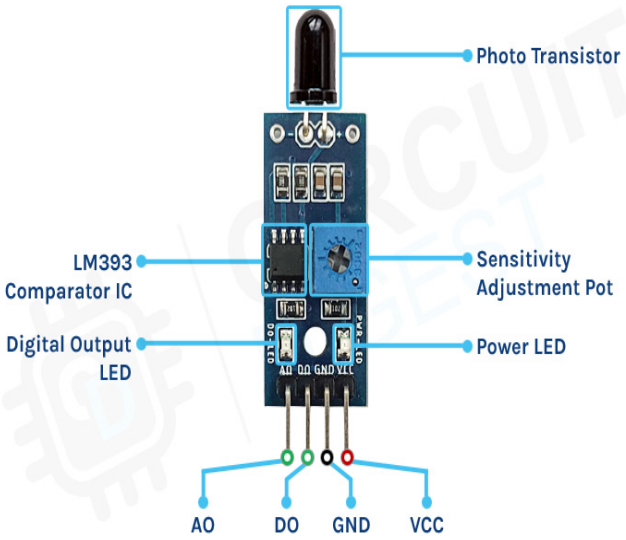
Touch sensors work similar to a switch. When they are subjected to touch, pressure or force they get activated and acts as a closed switch. When the pressure or contact is removed, they act as an open switch. Capacitive touch sensor contains two parallel conductors with an insulator between them.




When these conductor plates come in contact with cur fingers, our finger acts as a conductive object. Due to this, there will be an uncertain increase in the capacitance. A capacitance measuring circuit continuously measures the capacitance CO of the sensor. When this circuit detects a change in capacitance it generates a signal. The resistive touch sensors calculate the pressure applied on the surface to sense the touch. These sensors contain two conductive films coated with indium tin oxide, which is a good conductor of electricity, separated by a very small distance.

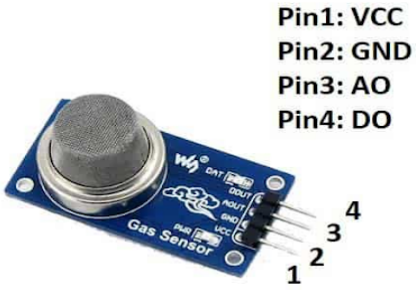
- **Flame OR sensor**

Flame sensors, also known as flame detectors or flame sensors, are devices designed to detect the presence of an open flame, fire, or other sources of intense heat and light. They are widely used in various applications, primarily for safety and control purposes. Flame sensors operate on the principle of detecting the specific wavelengths of light emitted by flames, and they play a crucial role in preventing fires and ensuring safety in many industries.



The flame sensors available in the market with two categories ona is having three pins (00 Gnd, Vcc) and second one is having four pins (AO, DO, Gnd, Vcc) both are can be easily interfaced with Arduino and Arduino compatible boards.

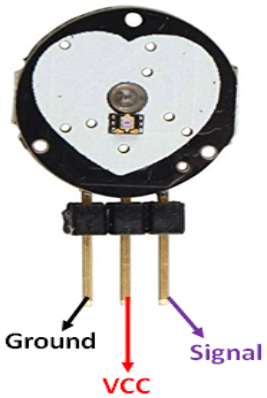
- **Smoke sensor**




Smoke sensors such as photoelectric or ionization detectors, operate based on distinct principles. Photoelectric sensors use a light source and a sensor to detect smoke particles scatter the light. When smoke is present, it scatters the light, triggering an alarm, Ionization detectors contain a small radioactive source that ionizes the air, creating a small electrical current. When smoke enters the chamber, it disrupts the current, causing the alarm to activate. Both types of sensors provide early fire detection by detecting smoke particles or changes in ionization due to combustion, thereby alerting occupants to potential fire hazards.

- **Heartbeat sensor**

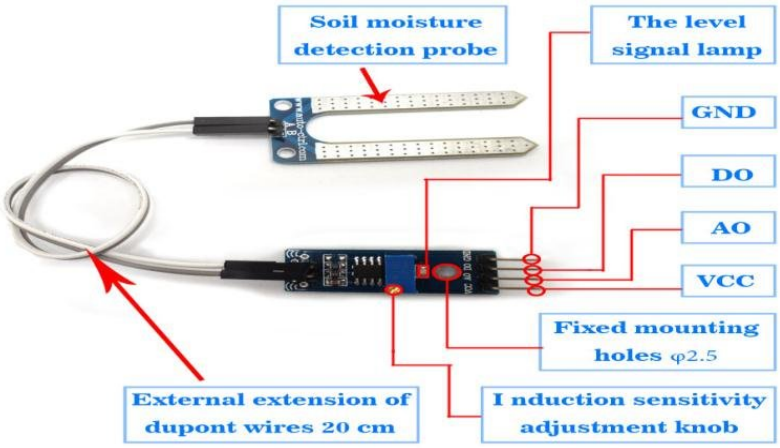
Heartbeat sensors, commonly found in medical devices like pulse oximeters, work on the principle of photoplethysmography (PPG), PPG measures blood volume changes in peripheral tissues as blood pulses through them. When light (typically an LED) is shone onto the skin, the absorption of this light varies with blood volume changes, as blood absorbs more light than surrounding tissue.



A photodetector measures the light that passes through or reflects off the skin, resulting in a waveform that corresponds to the heartbeat. By analysing the variations in this waveform, the sensor can calculate heart rate and oxygen saturation levels in the blood, among other vital signs.

- **Soil sensor**

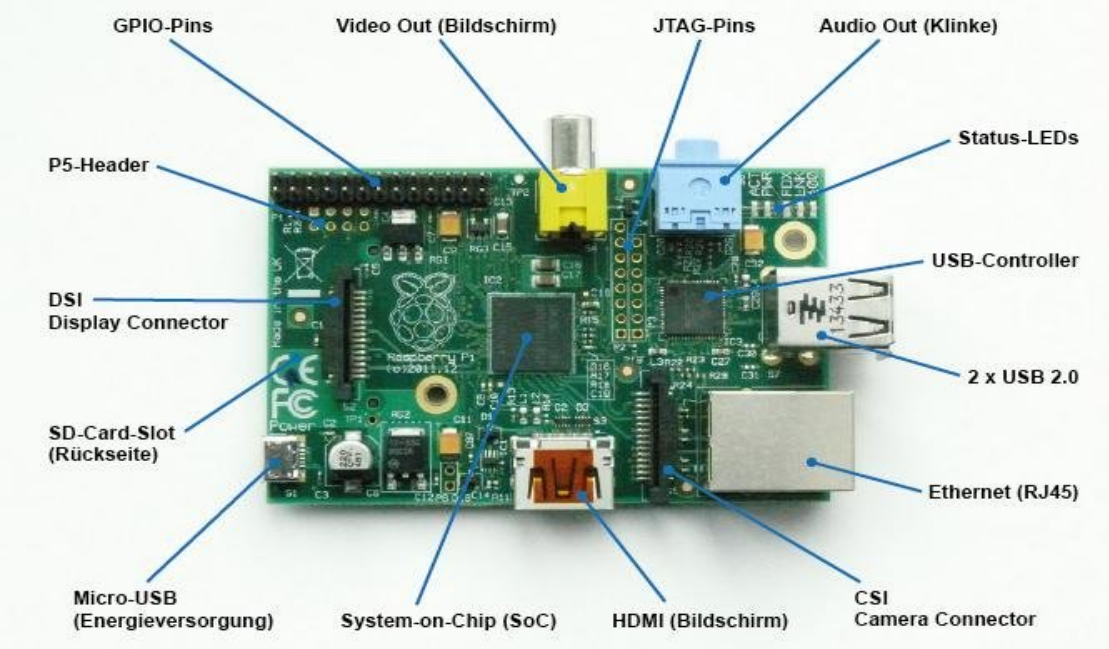
Soil sensors operate on the principle of measuring soil moisture content and sometimes other parameters like temperature and salinity. They typically use either capacitive or resistive sensing methods. Capacitive sensors measure changes in electrical capacitance as the soil moisture level changes, while resistive sensors gauge the electrical resistance between two electrodes in the soil, which varies with moisture levels. By analysing these electrical changes, the sensors provide data on soil moisture, enabling precise irrigation and agriculture management, as well as environmental monitoring. This information helps optimize water usage, increase crop yield, and prevent overwatering or underwatering of plants, ultimately conserving resources.




- **Raspberry pi**

The Raspberry Pi is a small, affordable, credit card-sized computer that has gained immense popularity for its versatility. It features a CPU, RAM, and GPU on a single board. Users can load various operating systems onto an SD card, transforming it into a full-fledged computer. It offers a range of ports, including HDMI, USB, and GPIO, making it suitable for diverse projects.

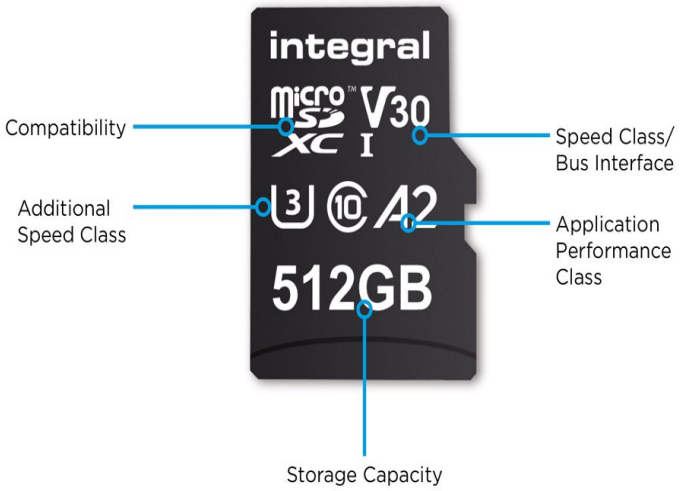
The Raspberry Pi is widely used for educational purposes, as a media centre, for loT applications, robotics, and more. Its low cost, compact size, and community support have made it a favourite among hobbyists, educators, and developers for innovative and practical computing projects.




The Raspberry Pi is a credit card-sized single-board computer designed for diverse applications. Its working principle involves a compact system-on-chip (SoC) architecture, featuring a CPU, RAM, GPU, and various interfaces. Upon power-up, the operating system. loads from an SD card, enabling users to run a wide range of software and programs. The GPIO (General Purpose Input/Output) pins facilitate hardware interaction, and it connects to peripherals through USB, HDMI, and Ethernet ports. Users can customize and expand capabilities, making it ideal for education, DIY projects, and even embedded systems, thanks to its affordability and versatility.

- **SD card**

An SD (Secure Digital) card is a compact, portable flash memory storage device used in various electronic devices, such as cameras, smartphones, and digital music players. These small, durable cards offer high data storage capacity and fast data transfer speeds, making them ideal for storing photos, videos, documents, and more. They come in different sizes, with microSD cards being the most common. SD cards are also used in embedded systems and single-board computers. Their reliability, versatility, and ease of use make them a popular choice for expanding device storage and transferring data between devices.




It works by using non-volatile memory to store digital data. Inside the SD card, a flash memory chip stores data as electrical charges. When data is written, these charges are trapped in memory cells, and when data is read, the charges are measured to retrieve the stored Information. SD cards have a controller that manages data storage and retrieval, file systems, and wear-levelling to ensure even usage of memory cells.

## Experiment - 3

**AIM**:-

To demonstrate digital read/write function by blinking the built-in LED in the Arduino UNO

**Theory:-**

To demonstrate the digital read and write functions on an Arduino UNO, we can create a simple project that blinks the built-in LED on and off. The built-in LED on the Arduino UNO is connected to digital pin 13, and we can use digital write functions to turn it on and off, while using a digital read function to check its current state.

**Objective:-**

The code will blink the built-in LED on pin 13 with a delay, turning it on and off repeatedly. We'll also read the LED's state to demonstrate the digitalRead() function.

**Code Explanation**

In this example:

- digitalWrite() is used to turn the LED on or off.
- digitalRead() is used to read the current state of the LED pin (whether it's HIGH or LOW)

**Code:-**

```
const int ledPin = 13;

void setup() {
pinMode(ledPin, OUTPUT);
Serial.begin(9600);
}

void loop() {
digitalWrite(ledPin, HIGH);
Serial.println("LED is ON");
delay(1000);

digitalWrite(ledPin, LOW);
Serial.println("LED is OFF");
delay(1000);
}
```

Output

```
LED is ON
LED is OFF
LED is ON
LED is OFF
LED is ON
LED is OFF
```

## Experiment – 4

**AIM:**

To design and implement a traffic light system using an Arduino microcontroller that simulates real- world traffic signal operations.

**COMPONENTS REQUIRED:**

- Arduino Uno
- Red, Yellow, and Green LEDs
- Resistors (220Ω)
- Breadboard
- Jumper Wires
- Power Supply (USB or 9VBattery )

**THEORY:**

Traffic lights are an essential part of traffic management systems, used to regulate vehicle and pedestrian movement. In this project, an Arduino microcontroller is programmed to control the sequence of Red, Yellow, and Green LEDs to simulate a traffic light system.

The LEDs will follow a predefined timing sequence:

- Green LED: ON for 5 seconds (vehicles move)
- Yellow LED: ON for 2 seconds (warning)
- Red LED: ON for 5 seconds (vehicles stop)

This cycle will repeat continuously to simulate real-

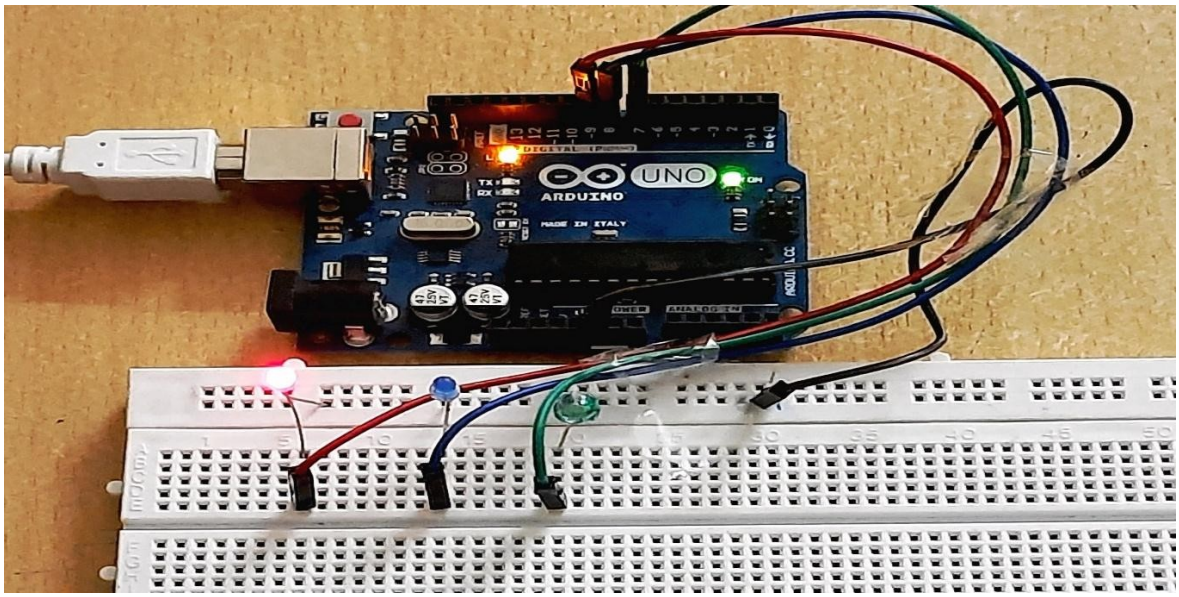
world traffic control.

**CIRCUIT CONNECTIONS:**

- Connect the Red LED to pin 9 of the Arduino with a 220Ω resistor.
- Connect the Yellow LED to pin 10 of the Arduino with a 220Ω resistor.
- Connect the Green LED to pin 11 of the Arduino with a 220Ω resistor.
- Connect the negative terminals of all LEDs to the ground (GND) on the

**CODE:**

```
int red = 9;
int yellow = 10;
int green = 11;

void setup()
{
pinMode(red, OUTPUT);
pinMode(yellow, OUTPUT);
pinMode(green, OUTPUT);
}

void loop()
{
digitalWrite(green, HIGH);
delay(5000);
digitalWrite(green, LOW);

digitalWrite(yellow, HIGH);
delay(2000);
digitalWrite(yellow, LOW);

digitalWrite(red, HIGH);
delay(5000);
digitalWrite(red, LOW);
}
```



## Experiment - 5

**Aim**:-

To demonstrate analog read/write function by using potentiometer with the Arduino UNO.

**Theory:-**

To demonstrate analog read and write functions with an Arduino UNO using a potentiometer, we can create a simple project where the potentiometer controls the brightness of an LED. The potentiometer will be connected to an analog input pin on the Arduino, and the LED will be
connected to a PWM-capable digital output pin. The potentiometer's value will determine the LED's brightness.

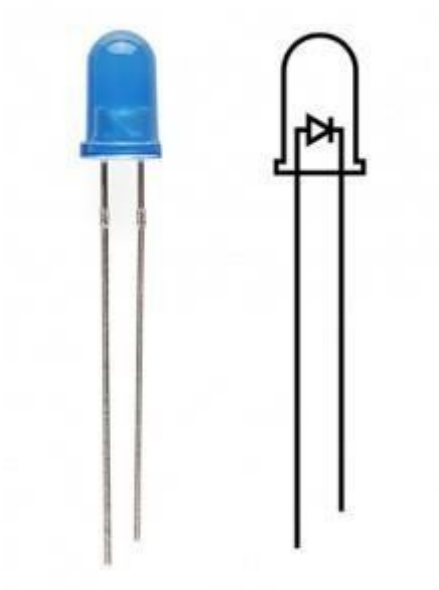
## Components Required

1. Arduino UNO
2. Potentiometer (10kΩ recommended)
3. LED
4. 220Ω resistor (to limit current to the LED)
5. Jumper wires
6. Breadboard

## Wiring Setup

1. **Potentiometer:**
    - Connect the **center pin** of the potentiometer to **A0** (analog input) on the Arduino.
    - Connect one side pin of the potentiometer to **5V**.
    - Connect the other side pin to **GND**.

2. **LED:**
    - Connect the **anode** (longer leg) of the LED to **digital pin G** (a PWM-capable pin).
    - Connect the **cathode** (shorter leg) of the LED to **GND** through a **220Ω resistor**.



**Code:**

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

void setup() {
pinMode(ledPin, OUTPUT);
}

void loop() {
int potValue = analogRead(potPin);
int ledBrightness = map(potValue, 0, 1023, 0, 255);
analogWrite(ledPin, ledBrightness);
delay(10);
}
```

**Explanation of the Code**

1. **analogRead(potPin)**: Reads the analog value (0–1023) from the potentiometer.
2. **map(potValue, 0, 1023, 0, 255)**: Maps the 10-bit potentiometer value to an 8-bit PWM range (0–255) for the LED brightness.
3. **analogWrite(ledPin, ledBrightness)**: Sets the LED brightness by writing the mapped PWM value to the LED pin.

**How It Works**

As you turn the potentiometer, the analog input from **A0** changes from 0 to 1023. The map function scales this value to a range of 0 to 255, which controls the LED's brightness from fully off
(0) to fully on (255). The LED brightness will increase as you turn the potentiometer up.

This setup demonstrates how analog reading from a potentiometer can control analog output using PWM on the Arduino.

## Experiment – 6

**AIM**:-

To measure light intensity using LDR with Arduino UNO.

**Apparatus**:-

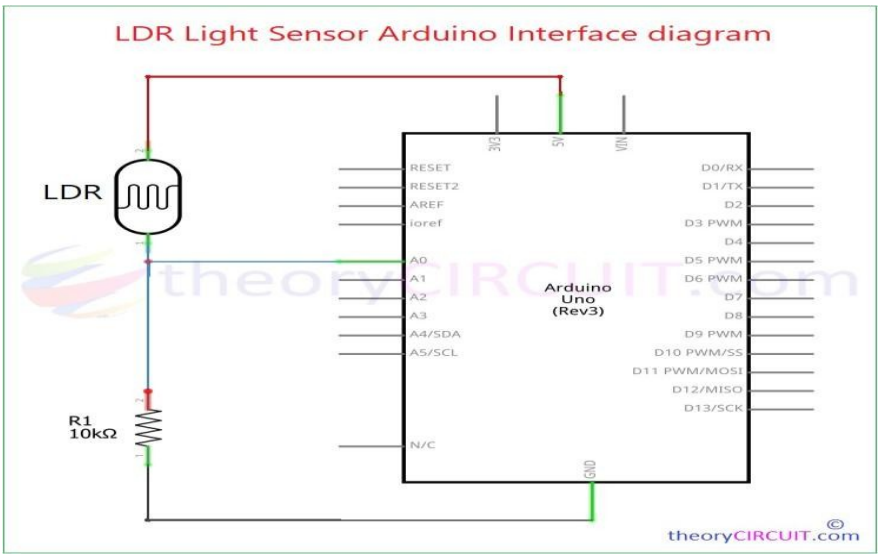
Arduino UNO Board, Resistors, Breadboard, LED, Connecting wires, LDR(Light dependent Resistor), IDE Software.

**THEORY:-**

Light Intensity is usually measured to control the switching on and off of the light in the Home Automation system. LDR sensor is the photoresistor that plays a major role in the Light Intensity Measurement circuit. Arduino is the brain behind smart lightning. LDR sensor full form is Light Dependent Resistor. The final step before deploying the smart lighting system is to fine-tune the Arduino code using the Arduino IDE, adjusting the thresholds for light intensity levels to ensure seamless and precise control over the LED brightness. Devices and circuits are usually designed and simulated on software before building the circuit itself.

This ensures circuit safety and makes it easy to build the circuit without worrying about any hazards from the design. Tinker Cad is an online simulation software for electronic circuits. Let us explore how to measure Light Intensity using LDR and Arduino.

**Circuit Diagram**



**PROCEDURE:**

1. Connect the Circuit:
   - Connect one leg of the LDR to the 5V pin on the Arduino.
   - Connect the other leg of the LDR to the AO analog input pin on the Arduino. Connect one leg of the resistor (10k) to the AO pin on the Arduino.
   - Connect the other leg of the resistor to the GND (ground) pin on the Arduino.
   - Connect the junction between the LDR and resistor to the AO pin.

2. Set Up Arduino IDE:

   - Make sure you have the Arduino IDE installed on your computer.
   - Connect your Arduino Uno to your computer using a USB cable.

3. Write and upload the Arduino Code:

   - Open the Arduino IDE.
   - Write or copy the provided Arduino code into the IDE. • Upload the code to your Arduino Uno

**Code:**

```
const int ldrPin = A0;
int ldrValue = 0;
float lightIntensity = 0.0;

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

void loop() {
ldrValue = analogRead(ldrPin);
lightIntensity = map(ldrValue, 0, 1023, 0, 100);

Serial.print("LDR Value: ");
Serial.print(ldrValue);
Serial.print(" Light Intensity: ");
Serial.print(lightIntensity);
Serial.println("%");

delay(500);
}
```

**RESULT**

As the light conditions change, the LDR readings and voltage will vary, resulting in different light intensity percentage values. Higher values indicate higher light intensity

## Experiment – 7

**AIM:-**

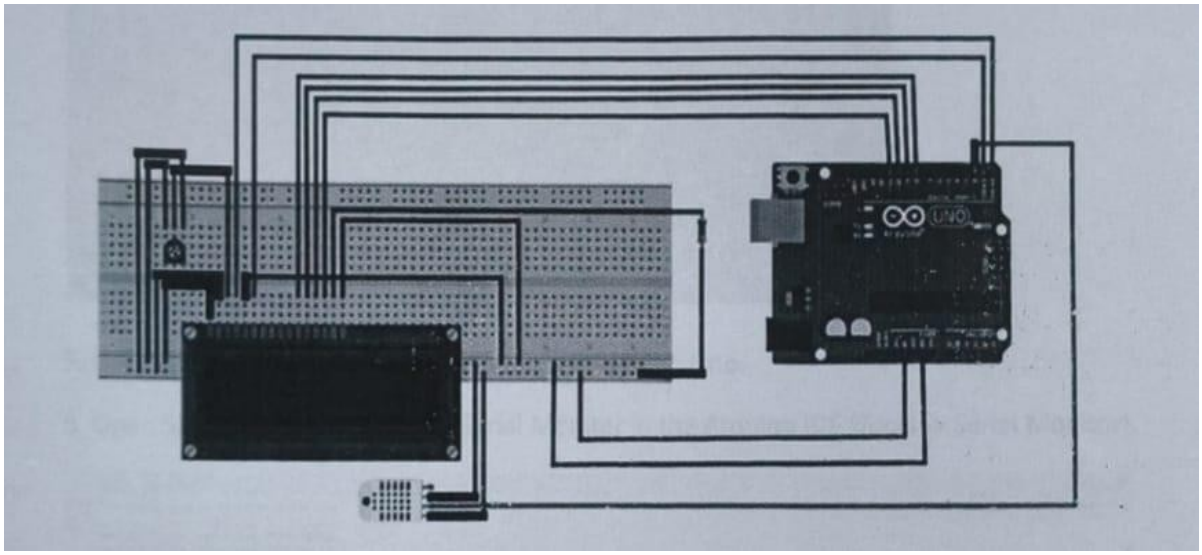
To measure humidity and temperature using DHT11 with Arduino UNO.

**Apparatus:-**

Arduino UNO Board, DHT11 (Temperature and humidity Sensor), Resistor, Potentiometer, Breadboard, Connecting Wires, IDE Software.

**THEORY:**

We need the temperature and humidity measurement to understand the environmental conditions for a given zone. We might use google or any whether predicting site for the same but often they do not show the real time data and most of the data is averaged out and often based on predicting algorithms. We might also need to measure temperature and humidity level of a closed chamber manufacturing process where controlling the temperature and humidity of a process is of utmost importance. In this project we will be using Arduino Uno and DHT11 Temperature and Humidity Sensor to measure the ambient temperature and humidity in the air and display the measured values on a 16x2 LCD screen.



**PROCEDURE:-**

1.Connect the circuit

- Connect the DHT11 sensor to the Arduino Uno using jumper wires.
- Connect the VCC pin of the DHT11 to the 5V pin on the Arduino.
- Connect the GND pin of the DHT11 to the GND pin on the Arduino.
- «Connect the Data pin of the DHT11 to a digital pin on the Arduino, e.g., D2.

2. Set Up Arduino IDE:

- Make sure you have the Arduino IDE installed in your computer.
- Connect your Arduino Uno to your computer using a USB cable.

3. Open the Arduino IDE

- Go to “sketch -> “Include Library” ->” Manage Libraries.....”
- In the Library Manager type “DTH” in the search bar.
- Find the “DTH sensor library” by Adafruit, and click “install”.

4. Write and upload the Arduino code:

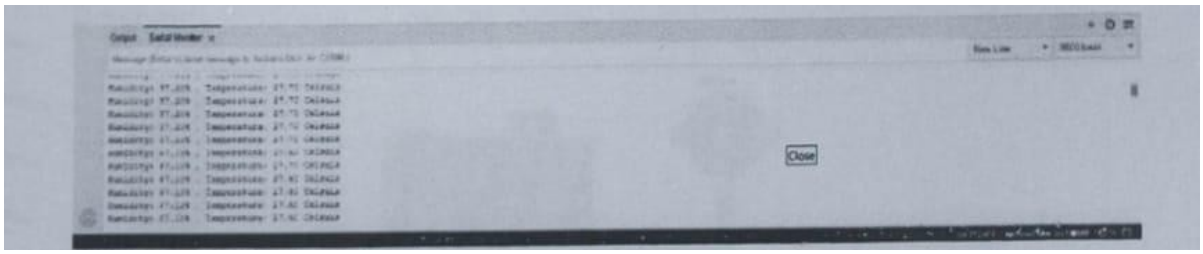
- Open a new sketch in the Arduino IDE.
- Write or copy the following Arduino code.

**Code:-**

```
#include <DHT.h>

#define DHTPIN 2 // Pin where the DHT11 is connected
#define DHTTYPE DHT11 // Define the type of DHT sensor

DHT dht(DHTPIN, DHTTYPE);

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

void loop() {
float humidity = dht.readHumidity(); // Read humidity
float temperature = dht.readTemperature(); // Read temperature
in Celsius
if (isnan(humidity) || isnan(temperature)) {
Serial.println("Failed to read from DHT sensor!");
return;
}
```

```
Serial.print("Humidity: ");
Serial.print(humidity);
Serial.print("% Temperature: ");
Serial.print(temperature);
Serial.println("°C");
delay(2000); // Wait for 2 seconds before next reading
}
```

5. Upload the code: Upload the code to your Arduino Uno.
6. Open Serial Monitor: Open the Serial Monitor in the Arduino IDE (Tools -> Serial Monitor).



**RESULT:-**

This system is compact to an extent and cost effective when compared to prices of instruments used to measure the environmental factor.

## Experiment – 8

**AIM:-**

Design and Implement a smoke level detector using Arduino IDE.

**THEORY:-**

The MQ2 sensor is one of the most widely used In the MQ sensor series. It ls a MOS (Metal Oxide Semicon ductor) sensor. Metal oxide sensors are also known as Chemireslstors because sensing is based on the change In resistance of the sensing material when exposed to gasses

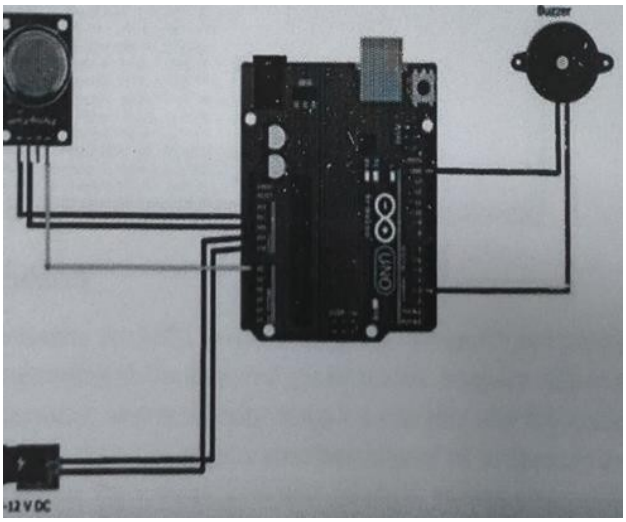

The MQ2 gas sensor operates on SV DC and consumes approximately 800mW. It can detect LPG, Smoke, Alcohol, Propane, Hydrogen, Methane and Carbon Monoxide concentrations ranging from 200 to 10000 ppm.

Note that the MQ2 gas sensor detects multiple gasses, but cannot identify them! That is normal; most gas sensors operate in this manner. Therefore, it is best suited for measuring changes in a known gas density rather than detecting which one is changing.

**Apparatus:-**

Arduino UNO, MQ2 gas sensor, Breadboard(optional),Jumper cables, USB cable, Buzzer, 7-12 V DC battery

**Diagram:-**



**Procedure:-**

1. Connect the VCC pin of the sensor to the SV of the Ardulno.
2. Connect the GND of the sensor to the GND of the Ardulno.
3. Connect the digital pin of the sensor DO to the digital pin number 8 of the Ardulno 4. Connect the analog pin of the sensor to the analog pin AO of the Arduino. Now connect the buzzer to the Arduino using D3 and GND pin

**Code:**

```
#define MQ2_SENSOR A0
#define BUZZER 8
#define THRESHOLD 300

void setup() {
Serial.begin(9600);
pinMode(BUZZER, OUTPUT);
}

void loop() {
int gasLevel = analogRead(MQ2_SENSOR);
Serial.print("Gas Level: ");
Serial.println(gasLevel);
```

```
if (gasLevel > THRESHOLD) {
Serial.println("Warning! High gas level detected!");
digitalWrite(BUZZER, HIGH);
}

else {
digitalWrite(BUZZER, LOW);
}
delay(1000);
}
```

**Result:-**

In conclusion, the MQ2 gas/smoke sensor demonstrated promising capabilities in detecting and responding to the targeted gas or smoke. Its quick response time, recovery characteristics, and sensitivity make it a valuable tool for applications requiring gas monitoring. However, careful consideration of its limitations and appropriate calibration are necessary for accurate and reliable results in diverse environmental conditions

## Experiment – 9

**Aim: -**

To interface HC SR04 sensor with Arduino.

**Theory:-**

Ultrasonic Sensor HC-SR04 is a sensor that can measure distance. It emits an ultrasound at 40 000 Hz (40kHz) 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.

The connection of Arduino and Ultrasonic Sensor HC-SR04


**Requirements:-**

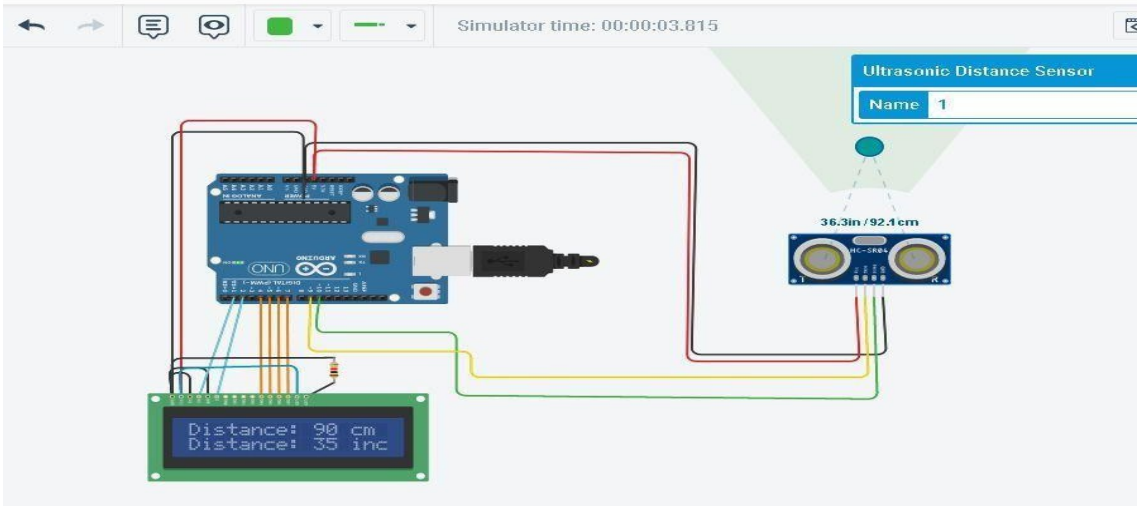
1. Arduino UNO R3 CH340 (you can use any Arduino Boards)
2. Ultrasonic Sensor HC-SR04
3. Male to Male Jumper Wires
4. Breadboard

**Procedure:-**

1. First do the wiring as shown in the circuit diagram.
2. Open Arduino IDE Software and write down your code, or download the code below and open it.

3. Choose your own Arduino board (in this case Arduino Uno), by selecting Tools > Board > Arduino/Geniuno Uno
4. Choose your COM Port (usually it appears only one existing port), Tools > Port > COM.. (If there are more than one ports, try it one by one)
5. Upload your code by pressing Ctrl + U or Sketch > Upload
6. To display the measurement data you can use Serial Monitor by pressing Ctrl + Shift + M (make sure that the baudrate speed is 9600)

**Circuit Diagram:**




**Code:**

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(1, 2, 4, 5, 6, 7);
const int trigPin = 9;
const int echoPin = 10;
long duration;
int distanceCm, distanceInch;

void setup() {
lcd.begin(16,2);
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
}

void loop() {
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distanceCm = duration * 0.034 / 2;
```

```
distanceInch = duration * 0.0133 / 2;
lcd.setCursor(0,0);
lcd.print("Distance: ");
lcd.print(distanceCm);
lcd.print(" cm");
delay(10);
lcd.setCursor(0,1);
lcd.print("Distance: ");
lcd.print(distanceInch);
lcd.print(" inch");
delay(10);
}
```

**Result:-**

The HC-SR04 Ultrasonic Sensor measures distance by emitting 40kHz ultrasound waves and detecting their reflection. It is connected to an Arduino UNO with TRIG (Pin 9) and ECHO (Pin 10). The sensor calculates distance using the time taken for the sound waves to return. The measured distance is displayed on an LCD screen in cm and inches.

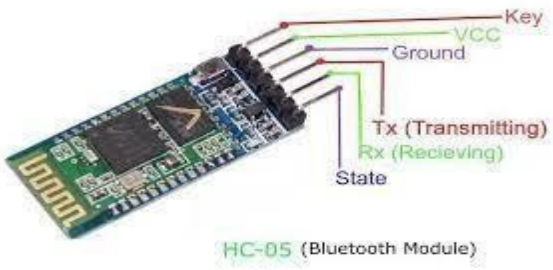
## Experiment – 10

**Aim: -**

To interface Bluetooth with Arduino and write a program to send sensor data to smartphone using Bluetooth.

**Theory:-**

Bluetooth is one of the most popular wireless communication technologies because of its low power consumption, low cost and a light stack but provides a good range. In this, data from a DHT-11 sensor is collected by an Arduino and then transmitted to a smartphone via Bluetooth.



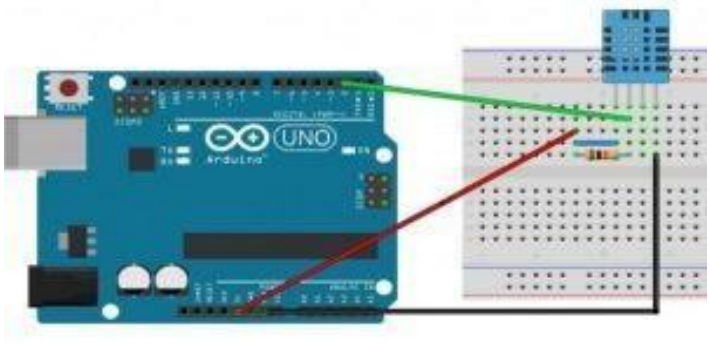

**Requirements:-**

1. Arduino Board and Arduino IDE
2. An Android Smartphone that has Bluetooth
3. HC-05 Bluetooth Module
4. Android Studio (To develop the required Android app) 5. USB cable for programming and powering the Arduino
5. DHT-11 temperature and humidity sensor.

**Working:**

To use the HC-05 Bluetooth module, simply connect the VCC to the 5V output on the Arduino, GND to Ground, RX to TX pin of the Arduino, and TX to RX pin of the Arduino. If the module is being used for the first time, you'll want to change the name, passcode etc. To do this the module should be set to command mode. Connect the Key pin to any pin on the Arduino and set it to high to allow the module to be programmed.

**Circuit Diagram:**




**Code:**

```
#include "DHT.h"

#define DHTPIN 2
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

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

void loop() {
char c;
if (Serial.available()) {
c = Serial.read();
if (c == 't') readSensor();
}
}

void readSensor() {
float h = dht.readHumidity();
float t = dht.readTemperature();
if (isnan(h) || isnan(t)) {
Serial.println("Failed to read from DHT sensor!");
return;
}
float hic = dht.computeHeatIndex(t, h, false);
Serial.print("Humidity: ");
Serial.print(h);
Serial.print(" %\t");
Serial.print("Temperature: ");
```

```
Serial.print(t);
Serial.print(" *C ");
Serial.print("Heat index: ");
Serial.print(hic);
Serial.println(" *C ");
}
```

**Result:-**

The Bluetooth module (HC-05/HC-06) is used to wirelessly transmit sensor data from an Arduino UNO to a smartphone. The DHT11 sensor measures temperature and humidity, and the Arduino collects this data. The Bluetooth module, connected via TX and RX pins, sends the sensor readings to a smartphone app like Bluetooth Terminal. The smartphone receives and displays the data in real time, allowing remote monitoring of environmental conditions.

**Experiment – 11**

**Aim: -**

Start Raspberry Pi and try various Linux commands in command terminal window : ls, cd, touch, mv, rm, man, mkdir, rmdir, tar, gzip, cat, more, less ,ps. sudo, cron, chgrp, ping etc.

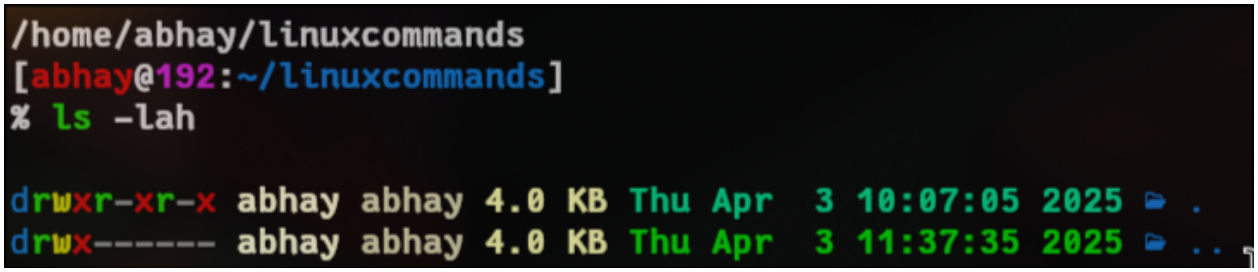
**Commands:-**

Step 1: Navigate to the Working Directory & Verify (`cd`, `pwd`)

cd ~/linuxcommands && pwd

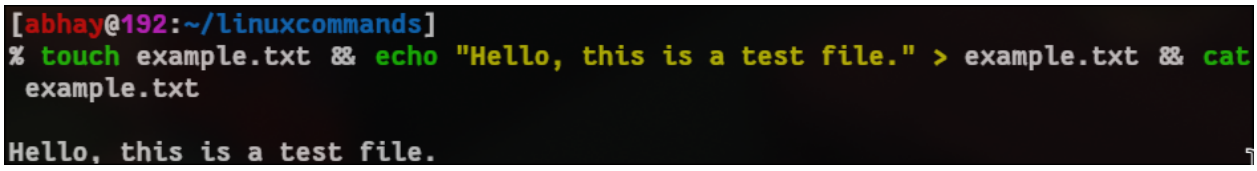

Step 2: List Files in the Directory (`ls`)

ls -lah



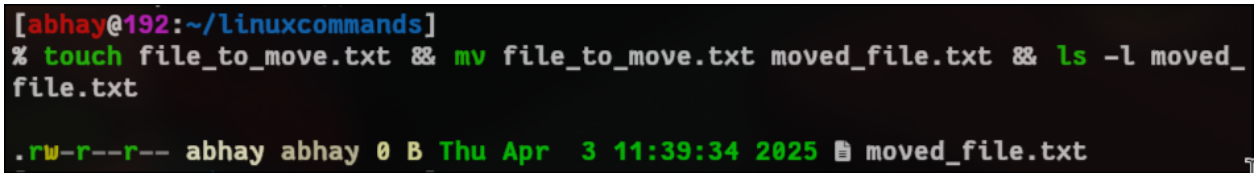

Step 3: Create a File & Display Contents (`touch`, `echo`, `cat`)

touch example.txt && echo "Hello, this is a test file." > example.txt && cat example.txt




Step 4: Create & Move a File (`touch`, `mv`, `ls`)

touch file_to_move.txt && mv file_to_move.txt moved_file.txt && ls -l moved_file.txt



Step 5: Create a Directory, Move File, List Files (`mkdir`, `mv`, `ls`)

mkdir testdir && touch testfile.txt && mv testfile.txt testdir/ && ls testdir/

```
[abhay@192:~/linuxcommands]
% mkdir testdir && touch testfile.txt && mv testfile.txt testdir/ && ls testdi
r/

testfile.txt
```

Step 6: Remove a File & Verify (`rm`, `ls`)

rm moved_file.txt && ls

```
[abhay@192:~/linuxcommands]
% rm moved_file.txt && ls

example.txt  testdir
```

Step 7: Display Manual Page (`man`)

man ls  # Press 'q' to exit

```
[abhay@192:~/linuxcommands]
% man ls   # Press 'q' to exit                                          1 ←

LS(1)                          User Commands
 LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor --sort
is  speci-
       fied.

       Mandatory  arguments  to  long options are mandatory for short options
       too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..
```

Step 8: Remove a Directory (`rmdir`, `mkdir`, `ls`)

mkdir tempdir && rmdir tempdir && ls

```
[abhay@192:~/linuxcommands]
% mkdir tempdir && rmdir tempdir && ls

example.txt  testdir
```

Step 9: Create & Archive a File (`tar`, `ls`)

touch archive_me.txt && tar -cvf archive.tar archive_me.txt && ls -l archive.tar

```
[abhay@192:~/linuxcommands]
% touch archive_me.txt && tar -cvf archive.tar archive_me.txt && ls -l archive
.tar

archive_me.txt
.rw-r--r-- abhay abhay 10 KB Thu Apr  3 11:43:11 2025  archive.tar
```

Step 10: Create, Compress & Verify (`gzip`, `ls`)

touch compress_me.txt && gzip compress_me.txt && ls -l compress_me.txt.gz

```
[abhay@192:~/linuxcommands]
% touch compress_me.txt && gzip compress_me.txt && ls -l compress_me.txt.gz

.rw-r--r-- abhay abhay 36 B Thu Apr  3 11:43:58 2025  compress_me.txt.gz
```

gunzip compress_me.txt.gz && ls -l compress_me.txt

```
[abhay@192:~/linuxcommands]
% gunzip compress_me.txt.gz && ls -l compress_me.txt

.rw-r--r-- abhay abhay 0 B Thu Apr  3 11:43:58 2025  compress_me.txt
```

Step 11: View File Content (`cat`, `more`, `less`)

echo "This is a sample text." > sample.txt && cat sample.txt && more sample.txt && less sample.txt

```
[abhay@192:~/linuxcommands]
% echo "This is a sample text." > sample.txt && cat sample.txt && more sample.
txt && less sample.txt

This is a sample text.
This is a sample text.
```

Step 12: List Running Processes (`ps`)

ps aux --sort=-%mem | head -10

```
[abhay@192:~/linuxcommands]
% ps aux --sort=-%mem | head -10

USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
abhay       8144  3.4  4.4 8616484 702672 tty2   Sl+  09:14   5:12 /usr/lib/li
breoffice/program/soffice.bin --writer
abhay       5015 12.0  3.6 12357952 580352 tty2  Sl+  09:11  18:27 /usr/lib/fi
refox/firefox https://
abhay       5282  3.0  3.4 3086052 534208 tty2   Sl+  09:11   4:43 /usr/lib/fi
refox/firefox -contentproc -isForBrowser -prefsHandle 0 -prefsLen 37984 -prefM
apHandle 1 -prefMapSize 265867 -jsInitHandle 2 -jsInitLen 247588 -parentBuildI
D 20250327171315 -sandboxReporter 3 -chrootClient 4 -ipcHandle 5 -initialChann
elId {46d092c3-4e45-49ea-8a52-a77e0cdf6663} -parentPid 5015 -crashReporter 6 -
greomni /usr/lib/firefox/omni.ja -appomni /usr/lib/firefox/browser/omni.ja -ap
pDir /usr/lib/firefox/browser 7 tab
tomcat8      1682  0.1  3.2 8342628 506232 ?      Sl   09:09   0:13 jsvc.exec -
```

Step 13: Execute a Command as Superuser (`sudo`)

sudo ls /root

```
[abhay@192:~/linuxcommands]
% sudo ls /root

Arch-Hyprland
```

Step 14: Change Group Ownership (`chgrp`, `ls -l`)

sudo groupadd testgroup && sudo chgrp testgroup example.txt && ls -l example.txt

```
[abhay@192:~/linuxcommands]
% sudo groupadd testgroup && sudo chgrp testgroup example.txt && ls -l example
.txt

.rw-r--r-- abhay testgroup 28 B Thu Apr  3 11:39:15 2025 example.txt
```

Step 16: Check Network Connection (`ping`)

ping -c 4 google.com

```
[abhay@192:~/linuxcommands]
% ping -c 4 google.com

PING google.com (2404:6800:4002:825::200e) 56 data bytes
64 bytes from e.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.5.2.8.0.2.0.0.4.0.0.8.6.4.0.4.2.
ip6.arpa (2404:6800:4002:825::200e): icmp_seq=1 ttl=118 time=9.47 ms
64 bytes from e.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.5.2.8.0.2.0.0.4.0.0.8.6.4.0.4.2.
ip6.arpa (2404:6800:4002:825::200e): icmp_seq=2 ttl=118 time=23.5 ms
64 bytes from e.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.5.2.8.0.2.0.0.4.0.0.8.6.4.0.4.2.
ip6.arpa (2404:6800:4002:825::200e): icmp_seq=3 ttl=118 time=23.6 ms
64 bytes from e.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.5.2.8.0.2.0.0.4.0.0.8.6.4.0.4.2.
ip6.arpa (2404:6800:4002:825::200e): icmp_seq=4 ttl=118 time=20.4 ms

--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 9.466/19.245/23.612/5.784 ms
```

## Experiment - 12

**Aim: -**

Run some python program on Pi like:

a) Read your name and print hello message with name

b) Read two number and print sum, difference, multiplication and division.

c) Word and character count of a given string

**Code:-**

```
name = input("Enter your name: ")
print(f"Hello, {name}!")

a, b = map(int, input("Enter two numbers: ").split())
print(f"Sum: {a + b}")
print(f"Difference: {a - b}")
print(f"Multiplication: {a * b}")
print(f"Division: {a / b if b != 0 else 'Undefined'}")

text = input("Enter a string: ")
print(f"Word count: {len(text.split())}")
print(f"Character count: {len(text)}")
```

**Output:-**

```
Enter your name: Abhay Raj
Hello, Abhay Raj!
Enter two numbers: 5 5
Sum: 10
Difference: 0
Multiplication: 25
Division: 1.0
Enter a string: Hello World
Word count: 2
Character count: 11
```

## Experiment – 13

**Aim: -**

Run some python program on Pi like:

a) Print a name ‘n’ times where name and n are read from standard input using for and while loops,
b) Handle Divided by Zero Exception.
c) Print current time for 10 times with an interval of 10 sec.
d) Read a file line and print the word count of each line.

**Code:-**

```
import time
name = input("Enter your name: ")
n = int(input("Enter the number of times: "))

for _ in range(n):
print(name)

i = 0
while i < n:
print(name)
i += 1

try:
a, b = map(int, input("Enter two numbers: ").split())
print(f"Result: {a / b}")
except ZeroDivisionError:
print("Error: Division by zero is not allowed.")

for _ in range(10):
print(time.strftime("%Y-%m-%d %H:%M:%S"))
time.sleep(10)

filename = input("Enter file name: ")
with open(filename, "r") as file:
for line in file:
print(f"Word count: {len(line.split())}")
```

**Output:-**

```
Enter your name: Abhay
Enter the number of times: 3
Abhay
Abhay
Abhay
Abhay
```

```
Abhay
Abhay
Enter two numbers: 2 2
Result: 1.0
2025-04-03 11:59:58
2025-04-03 12:00:08
2025-04-03 12:00:18
2025-04-03 12:00:28
2025-04-03 12:00:38
2025-04-03 12:00:48
2025-04-03 12:00:58
2025-04-03 12:01:08
2025-04-03 12:01:18
2025-04-03 12:01:28

Enter file name: data.py
Word count: 4
Word count: 3
Word count: 3
Word count: 7
Word count: 8
Word count: 9
Word count: 7
Word count: 7
Word count: 7
Word count: 7
Word count: 1
Word count: 0
Word count: 3
Word count: 9
Word count: 7
Word count: 1
Word count: 2
Word count: 3
Word count: 15
Word count: 11
Word count: 1
Word count: 2
Word count: 0
Word count: 8
Word count: 9
Word count: 10
```

```
Word count: 14
Word count: 14
Word count: 0
Word count: 6
Word count: 0
Word count: 9
Word count: 16
Word count: 5
```

## Experiment – 14

**Aim: -**

Developing Projects to Solve Real-World Problems:

To design a **Smart Water Level Indicator for Water Conservation** using Arduino that monitors water usage in a household and ensures that daily consumption does not exceed the UN-recommended limit of **40 liters per person per day**, while also preventing water wastage.

**Theory:-**

Water conservation is crucial for sustainable living. By monitoring water usage, individuals can be more mindful of their consumption. This project employs an **ultrasonic sensor** to measure the water level in a storage tank and calculates daily water usage. An **OLED display** shows real-time water levels and consumption, while a **buzzer** alerts the user when consumption exceeds the daily limit. This system helps in better water management and reduces unnecessary wastage.

**List of Components:-**

| Component | Specification | Quantity |
|---|---|---|
| Arduino UNO | ATmega328P | 1 |
| Ultrasonic Sensor | HC-SR04 | 1 |
| 0.96” Oled Display | 16x2 | 1 |
| Buzzer | 5V | 1 |
| LED | Red, Yellow, Green | 3 |
| Resistors | 220Ω | 3 |
| Jumper Wires | Male-Female | 10+ |
| Power Supply | 5V Adapter/Battery | 1 |

**Working:-**

1. **Water Level Measurement**

    - The **HC-SR04 ultrasonic sensor** is placed above the water tank.
    - It emits ultrasonic waves, which reflect back from the water surface.
    - The time taken for the echo to return is measured to calculate the water level.

2. **Usage Calculation**

    - The initial water level is recorded.
    - As water is used, the sensor detects the decrease in water level.
    - The difference between previous and current water levels is calculated and accumulated as **total daily usage**.

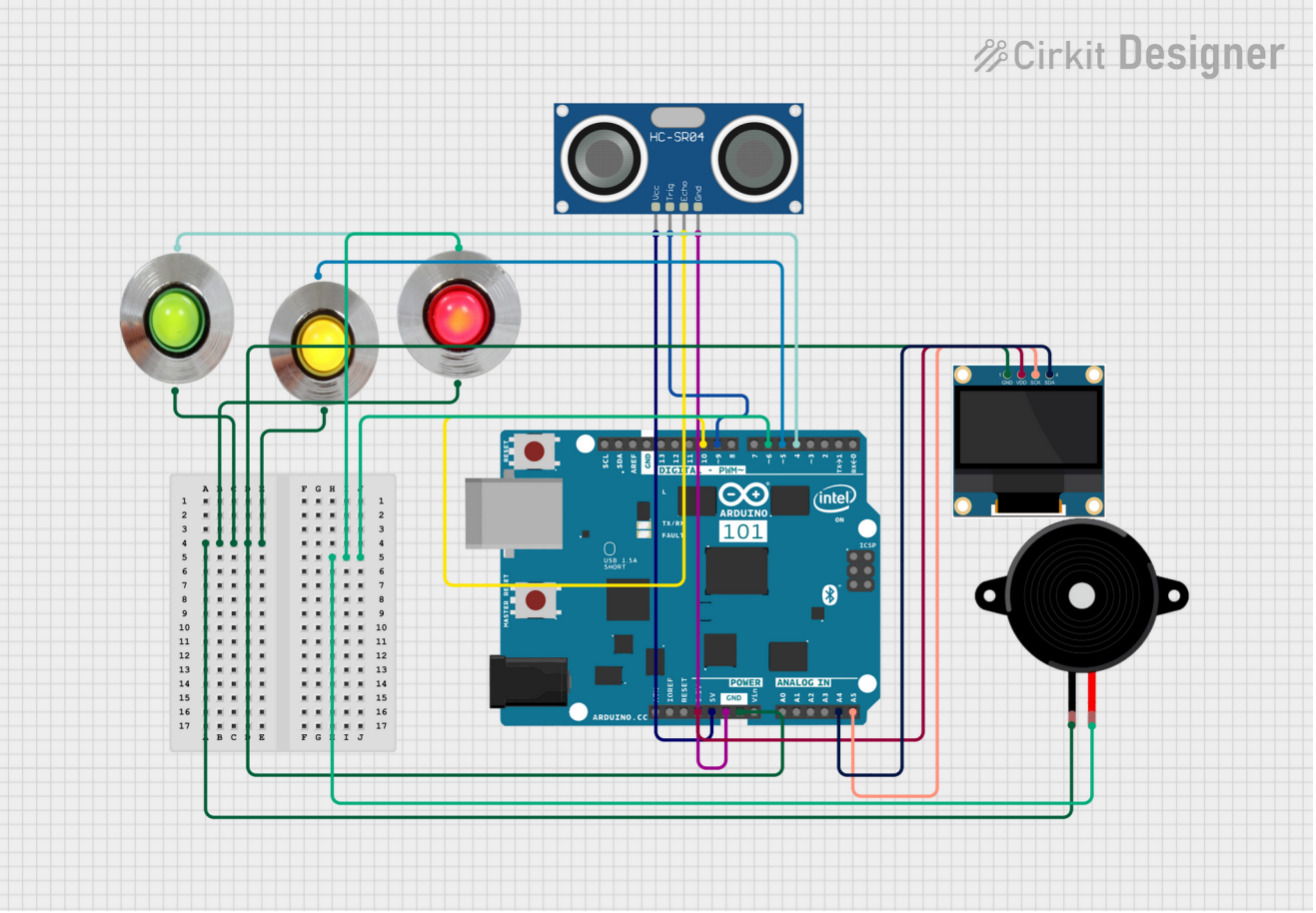
3. **Display & Alerts**

    - The **OLED display** continuously updates with the current water level and daily water usage.
    - If daily water usage exceeds **40 liters per person**, the **buzzer** is triggered as an alert.

4. **Water Conservation**

    - By monitoring consumption, users can adjust their water usage behavior.
    - The system helps in detecting water wastage and promoting efficient usage.

**Circuit Diagram:-**



**Code:-**

```
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_GFX.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

const int trigPin = 9;
const int echoPin = 10;
const int buzzer = 6;
const int dailyLimit = 40; // UN recommended daily limit per
person in liters
float tankCapacity = 500; // Tank capacity in liters
float previousLevel = 0;
float totalUsage = 0;

void setup() {
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
pinMode(buzzer, OUTPUT);
Serial.begin(9600);

if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
Serial.println(F("SSD1306 allocation failed"));
for (;;);
}
display.display();
delay(2000);
display.clearDisplay();
}

void loop() {
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
long duration = pulseIn(echoPin, HIGH);
float distance = duration * 0.034 / 2;
float waterLevel = tankCapacity - (distance * tankCapacity / 30);
float usage = previousLevel - waterLevel;
if (usage > 0) {
totalUsage += usage;
}
previousLevel = waterLevel;
```

```
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(SSD1306_WHITE);
display.setCursor(0, 0);
display.print("Water Level: ");
display.print(waterLevel);
display.println(" L");
display.print("Daily Usage: ");
display.print(totalUsage);
display.println(" L");
if (totalUsage > dailyLimit) {
display.println("Limit Exceeded!");
digitalWrite(buzzer, HIGH);
} else {
digitalWrite(buzzer, LOW);
}
display.display();
delay(5000);
}
```

**Result:-**

The **Smart Water Level Indicator** provides a practical solution for **water conservation** by monitoring and managing daily household water usage. The real-time display and alert system help prevent unnecessary water wastage while ensuring efficient resource management. By using this system, users can **stay within the recommended daily limit**, making a positive impact on sustainable water usage.