

Experiment no :1

Familiarization with Arduino/ Raspberry Pi and perform necessary software installation



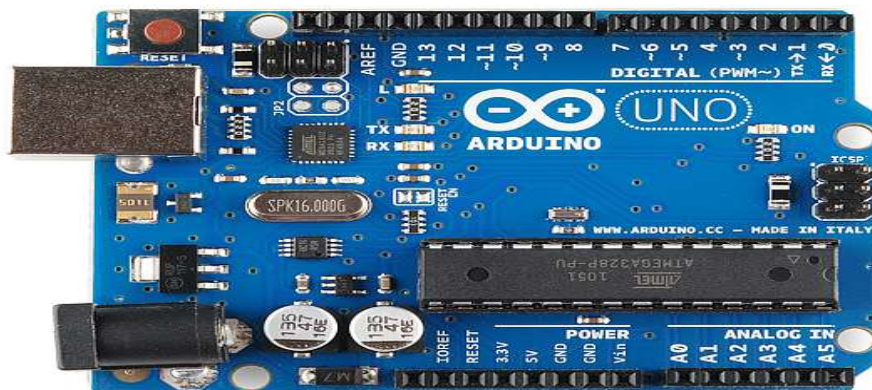
Arduino :-

- Arduino is microcontroller based .
- It is open sources electronic prototyping board .
- It is easy to used .

Major components:-

- Usb connector
- Power port
- Microcontroller
- Analog input pins
- Digital pins
- Reset switch
- Crystal oscillator
- USB interface chip
- TX RX LEDS

Diagram:-



1. USB connector :-

- It is a printer USB port .
- it is used to load a program.
- From Arduino IDE onto the Arduino board .
- It is also powered through this port .

2. Power port :-

- It can be powered through ac to dc.
- The power sources can be connected by plugging 2.1 mm center - positive plug .

3. 2.1 mm center -positive plug :-

- If the board is supplied with a higher voltage .
- There is voltage regulator that protects the board from burning out .

4. Microcontroller :-

- It is most prominent black rectangular chip with 28 pins.
- It is brain of arduino .
- It is microcontroller used on the UNO board is Atmega 328P by Atmel .



Component :-

- Flash memory :-
 - 32 KB
 - The program loaded from Arduino IDE is stored .
- RAM :-
 - 2 KB
 - This is a runtime memory .
- CPU :-
 - It control everything that goes on within the device.
 - It fetch the program instruction from flash memory and run them with the help of RAM.
- EEPROM :-
 - This is a type of non volatile memory .

5. Analog input pins :-

- The Arduino UNO board has 6 analog input pins .
- These pins can read the signal from an analog sensor .
- It is the system understanding .
- It only measures the voltage not current .
- It has very high internal resistance .

6. Digital pins :-

- These pins can be used either input and output pins .
- These pins act as a power supply source.
- Some of digital pins are labelled with tilde (~) symbol.

7. Reset switch :-

- It sends a logical pulse to the reset pin.
- It can be very useful . if your code doesn't repeat but you want to test it multiple times .

8. Crystal oscillator :-

- This is a quartz crystal oscillator which ticks 16 million times a second .
- The microcontroller performs one operation for eg : addition , subtraction etc.

9. USB interface chip :-

- Think of this as a signal translator .
- It converts signal in the USB level .

10. TX-RX leds :-

- TX- stands for transmit .
- RX- stands for receive .



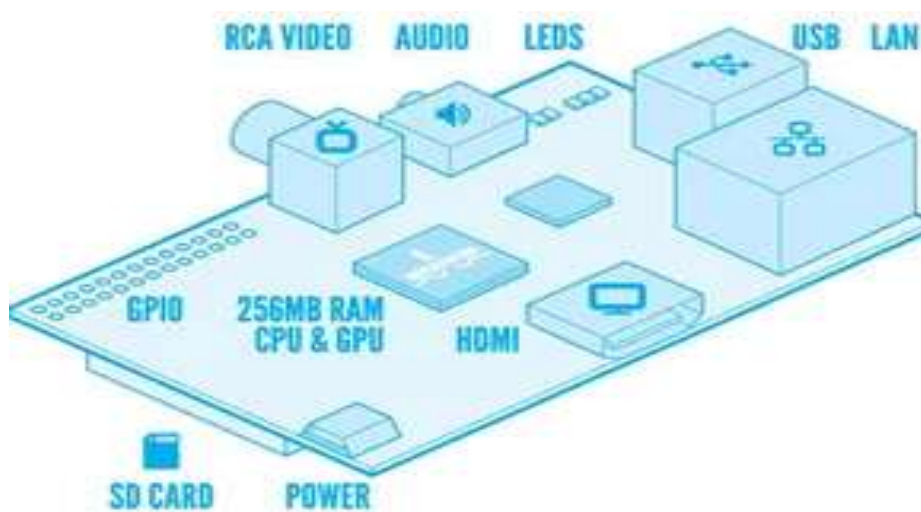
Raspberry PI:-

- The Raspberry Pi device looks like a motherboard. It is a cheap.
- Credit-card-sized device that uses a daily keyboard and mouse and joins to a TV or computer monitor.
- It is a thin weighable computer.
- Its low cost, modularity, and open architecture, it is now commonly used in many fields, such as for weather forecasting.

Major components :-

- ARM CPU/GPU
- GPIO
- RCA
- Audio out
- LEDs
- USB
- HDMI
- Power
- SD cardslot
- Ethernet

Diagram:-



1. ARM CPU/GPU:-

- This is a Broadcom BCM2835 System on a Chip (SoC) that's made up of an ARM central processing unit (CPU) and a Videocore 4 graphics processing unit (GPU). CPU handles all the computations that make a computer work and the GPU handles graphics output.

2. GPIO:-

- These are exposed general-purpose input/output connection points that will allow the real hardware hobbyists the opportunity to tinker.

3. RCA :-

- An RCA jack allows connection of analog TVs and other similar output devices.

4. Audio out :-

- This is a standard 3.55-millimeter jack for connection of audio output devices such as headphones or speakers. There is no audio in.

5. LEDs:-

- Light-emitting diodes, for all of your indicator light needs.

6. USB :-

- This is a common connection port for peripheral devices of all types (including your mouse and keyboard).

7. HDMI :-

- This connector allows you to hook up a high-definition television or other compatible device using an HDMI cable.

8. Power :-

- This is a 5v Micro USB power connector into which you can plug your compatible power supply.

9. SD cardslot :-

- This is a full-sized SD card slot. An SD card with an operating system (OS) installed is required for booting the device.

10. Ethernet :-

- This connector allows for wired network access and is only available on the Model B.



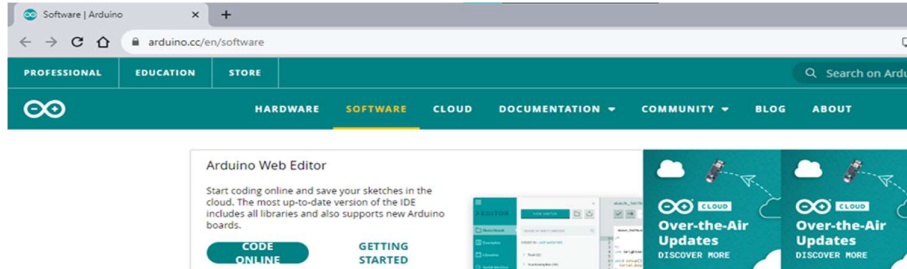
Compare of Arduino and Raspberry pi

Arduino	Raspberry pi
<ul style="list-style-type: none">○ The Arduino is defined as a simple microcontroller motherboard.	<ul style="list-style-type: none">○ The Raspberry Pi is defined as the mini computer.
<ul style="list-style-type: none">○ The Raspberry Pi is defined as the mini computer.	<ul style="list-style-type: none">○ It was developed to encourage basic learning for computer science students and other growing countries.
<ul style="list-style-type: none">○ Arduinos come with an 8-bit Microcontroller.	<ul style="list-style-type: none">○ Raspberry Pi has about 1GiB of RAM. Here, 1 GiB = $1024 \times 1024 \times 1024$ bytes = $(1024)^3$ bytes.
<ul style="list-style-type: none">○ It promotes C++ as the primary programming language.	<ul style="list-style-type: none">○ It promotes Scratch and Python as the chief programming language.
<ul style="list-style-type: none">○ it has no Operating system. The software platform requires Windows, Linux, and macOS operating system to run the program.	<ul style="list-style-type: none">○ The Operating system in all the files is saved in the SD card. For one Raspberry Pi, we can have multiple SD cards for the different operating system or file system.
<ul style="list-style-type: none">○ The connection to the Internet is quite complicated	<ul style="list-style-type: none">○ It can be easily connected to the Internet.
<ul style="list-style-type: none">○ The Analog to Digital converter is inbuilt in the Arduino. The Arduino UNO has 14 digital Input/Output pins, 6 analog pins, and power pins.	<ul style="list-style-type: none">○ The Analog to Digital converter is attached externally to the Raspberry Pi. The Raspberry Pi3 has 40 Input/Output pins on board.

Installation of Arduino

Step 1: Visit the Arduino Website's Downloads page

<https://www.arduino.cc/en/software>



Downloads

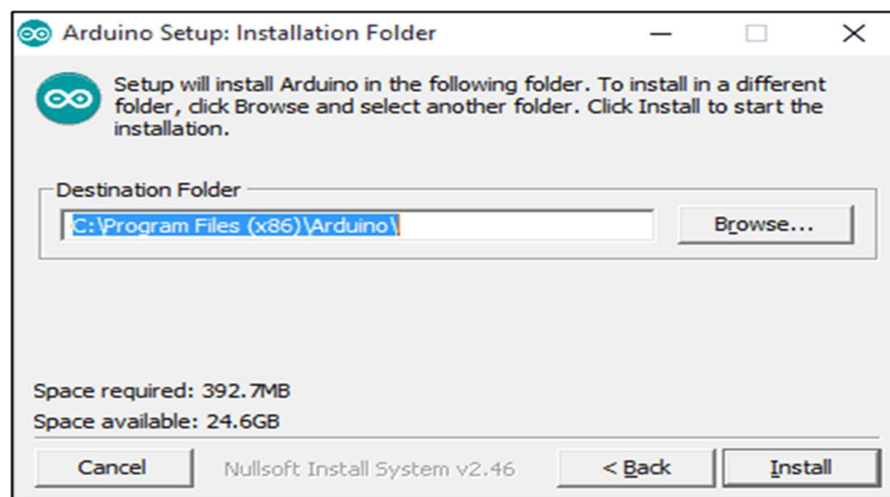


Step 2: Choose the version suitable for your operating system and click download

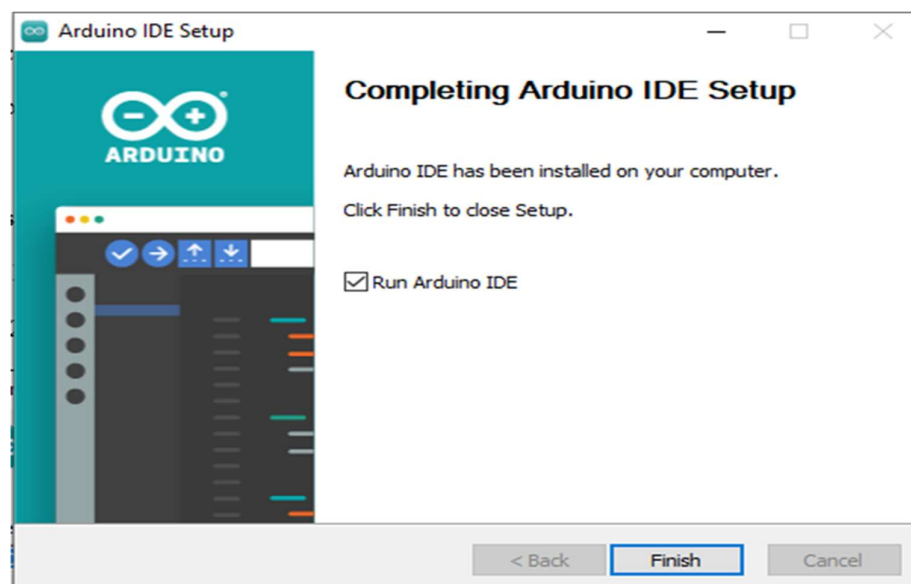
Downloads



Step 3: After Downloading open the installation setup and press install



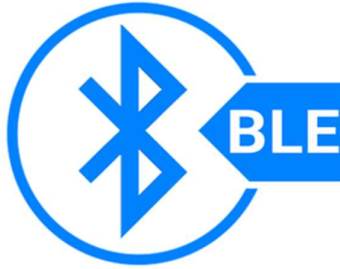
Step 4: Press Finish and here's your Arduino ready to launch.



Experiment no :2

To demonstrate the communication modules like BLE, WIFI, XBEE

 BLE:-



Bluetooth Low Energy (BLE), also known as Bluetooth Smart, is a wireless communication technology designed for short-range communication between devices with low power consumption. It is a subset of the classic Bluetooth technology and is widely used in various applications such as IoT (Internet of Things), wearable devices, healthcare, and home automation. Here are some key aspects of BLE:

1. Low Power Consumption: One of the primary advantages of BLE is its low power consumption. BLE devices can operate on small batteries for extended periods, making it ideal for battery-powered devices like fitness trackers and IoT sensors.
2. Short Range: BLE is designed for short-range communication, typically within a range of 10 meters (30 feet). This limited range helps reduce interference and power consumption.
3. Dual Mode: Many modern devices support both classic Bluetooth and BLE, allowing them to communicate with a wide range of devices, including older devices that use classic Bluetooth.
4. Central and Peripheral Roles: BLE devices can operate in two primary roles:
 5. Central Role: A central device (e.g., smartphone or tablet) can scan for and connect to peripheral devices.

6. Peripheral Role: A peripheral device (e.g., sensor or beacon) broadcasts data or services that central devices can discover and interact with.

7. Advertising and Scanning: BLE peripheral devices broadcast packets of data, which central devices can scan for. These packets are called advertisements and contain information about the peripheral's services and characteristics.

8. GATT (Generic Attribute Profile): BLE uses the GATT profile to define a standardized way for devices to communicate and exchange data. GATT organizes data into services and characteristics, making it easy to understand and interact with the data exposed by a BLE device.

9. Security: BLE provides several security features, including data encryption and authentication, to protect against eavesdropping and unauthorized access.

10. Profiles and Services: BLE devices use profiles and services to define their capabilities and functions. Common profiles include the Heart Rate Profile (HRP), Battery Service, and Environmental Sensing Profile (ESP), among others. These profiles define how data is structured and exchanged between devices.

WIFI



Wi-Fi, short for "Wireless Fidelity," is a popular wireless communication technology that allows electronic devices to connect to the internet and local area networks (LANs) without the need for physical cables. It has become an integral part of modern connectivity, enabling wireless internet access in homes, businesses, public spaces, and more. Here's a brief overview of Wi-Fi:

1. Wireless Connectivity: Wi-Fi enables devices such as smartphones, laptops, tablets, smart TVs, and IoT devices to connect to the internet and communicate with each other within a specific coverage area. This wireless connectivity eliminates the need for wired connections and allows for greater mobility and flexibility.

2.Standardization: Wi-Fi is based on a set of IEEE (Institute of Electrical and Electronics Engineers) standards, primarily the IEEE 802.11 family of standards. These standards define the protocols and technologies that govern Wi-Fi communication. Common Wi-Fi standards include 802.11a, 802.11b, 802.11g, 802.11n, 802.11ac, and 802.11ax (Wi-Fi 6).

3.Frequency Bands: Wi-Fi operates in the unlicensed radio frequency bands, primarily in the 2.4 GHz and 5 GHz bands. Different Wi-Fi standards use different frequency bands and channels within these bands to avoid interference and provide varying levels of performance and range.

4. SSID and Security: Wi-Fi networks are identified by a Service Set Identifier (SSID), which is a unique name that users can see when searching for available networks. To secure Wi-Fi networks, various encryption and authentication methods like WEP, WPA, and WPA2/WPA3 are used to protect data privacy and prevent unauthorized access.

5.Access Points (APs): Wi-Fi networks are typically set up using access points (routers or access point devices). Access points provide the connection between wireless devices and the wired network, allowing them to access the internet or local resources.

6.Range and Coverage: Wi-Fi signals have a limited range, which can vary depending on factors like frequency band, transmit power, and physical obstacles. Range extenders and mesh networks are commonly used to expand coverage in larger spaces.

7.speed and Throughput: Wi-Fi standards have evolved over the years to offer higher data transfer rates and improved performance. The latest standards, such as Wi-Fi 6 (802.11ax), offer faster speeds, increased capacity, and better performance in crowded environments.

8.Wi-Fi in Public Spaces: Wi-Fi is widely available in public places such as airports, cafes, hotels, and libraries. Users can often access these networks after agreeing to terms and conditions or by using a password.

9.Internet of Things (IoT): Wi-Fi is a common choice for connecting IoT devices to the internet and local networks due to its ubiquity and compatibility with various devices.

10.Emerging Technologies: As technology continues to advance, new Wi-Fi standards and technologies are being developed to meet the increasing demands for speed, reliability, and security. Wi-Fi 6E, for

example, extends into the 6 GHz band to provide additional spectrum and reduced interference.

XBEE



XBee is a brand of radio communication modules developed by Digi International (formerly MaxStream) for wireless communication in various applications. XBee modules are popular for their ease of use, reliability, and flexibility. Here's a brief overview of XBee modules:

1.Wireless Communication: XBee modules are designed for wireless communication in the Industrial, Scientific, and Medical (ISM) radio bands. They provide a convenient way to add wireless capabilities to electronic devices, allowing them to communicate over short to moderate distances.

2.Versatile Options: XBee modules come in various configurations and form factors, allowing users to choose the one that best suits their specific needs. These options include different frequency bands (2.4 GHz, 900 MHz, and 868 MHz), power levels, antenna options, and communication protocols.

3.Zigbee and DigiMesh Protocols: XBee modules support various communication protocols, with Zigbee and DigiMesh being the most common. Zigbee is a low-power, wireless mesh networking protocol used for applications like home automation and industrial control. DigiMesh is another mesh networking protocol designed for robust and reliable communication.

4.Point-to-Point and Mesh Networking: XBee modules can be configured for point-to-point communication, where two modules establish a direct connection, or for mesh networking, where multiple modules form a self-healing network. Mesh networks are resilient and can automatically route data through the network, making them suitable for applications with a dynamic topology.

5.Configuration and API Modes: XBee modules offer two primary modes of operation:

- AT Mode (Transparent Mode): In this mode, modules are configured using AT commands, and they transmit data transparently. It's like a simple serial cable replacement.

- API Mode (Application Programming Interface Mode): In API mode, modules allow for more advanced control and data framing, enabling developers to build custom communication protocols and manage complex data.

6.Range and Data Rate: The range and data rate of XBee modules vary depending on the specific model and configuration. Typically, XBee modules can provide communication ranges from a few hundred feet to several miles, with data rates ranging from a few kbps to over 1 Mbps.

7.Development Tools: Digi International provides development kits, software tools, and libraries to facilitate the integration of XBee modules into various projects. These tools make it easier for developers to configure and manage XBee modules.

8.IoT and Industrial Applications: XBee modules are commonly used in IoT (Internet of Things) applications, industrial automation, remote monitoring, sensor networks, and robotics, among others.

9.Integration with Microcontrollers: XBee modules can be interfaced with microcontrollers such as Arduino, Raspberry Pi, and others. This allows developers to create wireless-enabled projects with ease.

10.Security Features: XBee modules offer security features such as encryption and authentication to ensure the confidentiality and integrity of data transmitted over the wireless network.

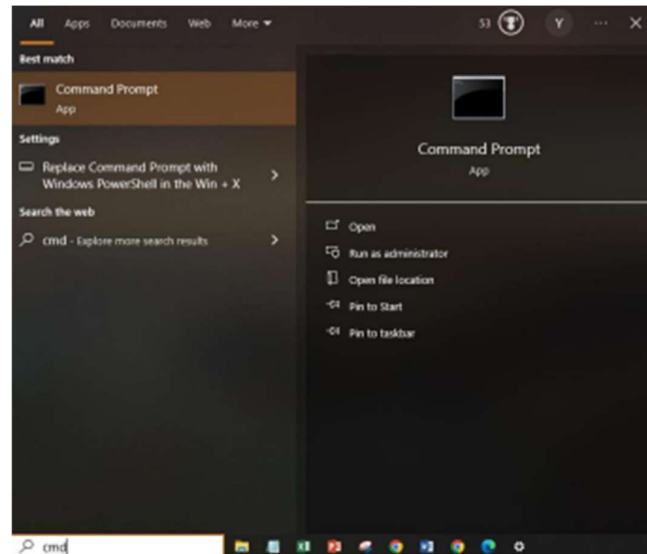
Experiment no :3

To find the IP address of Computer/ other devices

When your computer is connected to a network, it will be assigned an address on the network called an IP address. Scanning for IP address lets you have better control over your network. With 1-2 commands, you can quickly map out the devices in your network and the IP addresses that they are using.

Here are the simple steps to scan your network for IP address:

STEP 1: Click the Start icon, type command prompt or simply cmd into the search bar and press click the Command Prompt icon.



STEP 2: Now, Open the Command Prompt.

STEP 3: Type ipconfig /all and press Enter.

```

Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.

C:\Users\91750>ipconfig/all

Windows IP Configuration

Host Name . . . . . : DESKTOP-B0GG9DJ
Primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No

Ethernet adapter Ethernet:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :
Description . . . . . : Realtek PCIe GbE Family Controller
Physical Address. . . . . : 68-18-95-55-60-40
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Local Area Connection* 3:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :
Description . . . . . : Microsoft Wi-Fi Direct Virtual Adapter #3
Physical Address. . . . . : 14-85-7F-60-06-F4
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Local Area Connection* 12:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :

Connection-specific DNS Suffix . :
Description . . . . . : Microsoft Wi-Fi Direct Virtual Adapter #4
Physical Address. . . . . : 14-85-7F-60-06-F3
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . :
Description . . . . . : Intel(R) Wireless-AC 9462
Physical Address. . . . . : 14-85-7F-60-06-F3
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . : fe80::5161:6998:ca66:dc84%17(Preferred)
IPv4 Address. . . . . : 192.168.1.8(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : 09 October 2023 19:40:30
Lease Expires . . . . . : 10 October 2023 19:40:29
Default Gateway . . . . . : fe80::1%17
                          192.168.1.1
DHCP Server . . . . . : 192.168.1.1
DHCPv6 IAID . . . . . : 135562623
DHCPv6 Client DUID. . . . . : 00-01-00-01-28-C7-FC-A0-60-18-95-55-60-40
DNS Servers . . . . . : fe80::1%17
                          192.168.1.1
NetBIOS over Tcpip. . . . . : Enabled

Ethernet adapter Bluetooth Network Connection:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :
Description . . . . . : Bluetooth Device (Personal Area Network)
Physical Address. . . . . : 14-85-7F-60-06-F7
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes

```

STEP 4: The IP Address will display along with other LAN details.

Here, the address is IPv4 Address – 192.168.1.6

with Subnet Mask – 255.255.255.0

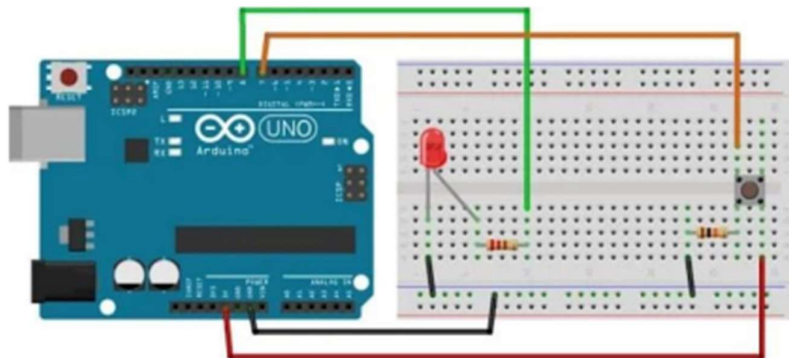
and DHCP Server – 192.168.1.1

Experiment no :4

To interface LED/ Buzzer with Arduino/ Raspberry Pi and write a program to turn ON/OFF LED for specific duration.

To perform this practical, we need these components –

- Arduino board Uno (preferably) but if you don't have the Uno board, you can easily adapt by finding the corresponding pins.
- Breadboard
- Push Button
- LED color – any
- 220 Ohm resistor for the LED. If you don't have this specific value, any resistor from 330 to 1k Ohm will do.
- 10k Ohm resistor for the push button. If you don't have, you can go until 20k-50k Ohm.
- A bunch of male to male wires (including if possible black, red, and other colors).



Step by step instructions to build the circuit:

1. Start by ensuring your Arduino is turned off and disconnected from any USB cables. Place a black wire between the blue section of the breadboard and one of the ground (GND) pins on the Arduino.
2. Take a look at the LED, and you'll notice that it has one leg shorter than the other. Connect the shorter leg to the ground (blue section) on the breadboard.
3. Now, connect the longer leg of the LED to a specific digital pin on the Arduino (for example, pin 8, but you can choose a different one). Insert a 220

Ohm resistor between the LED's longer leg and the digital pin. This resistor helps control the amount of current flowing through the LED.

4. Place the push button on the breadboard as shown in the picture.

5. Connect one leg of the push button to the ground on the breadboard. Insert a 10k Ohm resistor between this leg of the button and the ground. This resistor acts as a "pull-down" resistor, ensuring that the button's default state is LOW.

6. Use a red wire to connect another leg of the push button to the VCC (5V) on the breadboard.

7. Finally, connect one leg of the push button (the same side as the pull-down resistor) to a digital pin on the Arduino.

8. The goal is to control the LED: it should be off when the button is not pressed and on when the button is pressed.

CODE FOR THE CIRCUIT:

```
#define LED_PIN 8

#define BUTTON_PIN 5

void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUTTON_PIN, INPUT);
}

void loop() {
  if (digitalRead(BUTTON_PIN) == HIGH) {
    digitalWrite(LED_PIN, HIGH);}
  else {
    digitalWrite(LED_PIN, LOW);
  }
}
```

In the setup part of our program, we do two things:

- We tell the Arduino that we want to use the LED for showing things, so we set it as "OUTPUT."
- We tell the Arduino that we want to use the push button to get information, so we set it as "INPUT."

Now, in the main part of our program called "loop function", we do the following:

1. We check if the button is being pressed or not by reading its state with the help of `digitalRead()` function.
2. Since we added a special resistor, we know that when the button is not pressed, it gives us a signal called "LOW."
3. If the button is pressed (which means it's giving us a "HIGH" signal), we turn on the LED using the "HIGH" signal.
4. If the button is not pressed (giving us a "LOW" signal), we turn off the LED using the "LOW" signal

Experiment no :5

To interface DHT11/ DHT22 sensor with Arduino/ Raspberry Pi and write a program to print temperature and humidity readings.

The DHT11 measures relative humidity. Relative humidity is the amount of water vapor in air vs. the saturation point of water vapor in air. At the saturation point, water vapor starts to condense and accumulate on surfaces forming dew. The saturation point changes with air temperature. Cold air can hold less water vapor before it becomes saturated, and hot air can hold more water vapor before it becomes saturated. The formula to calculate relative humidity is:

$$RH = \left(\frac{\rho_w}{\rho_s} \right) \times 100\%$$

RH : Relative Humidity

ρ_w : Density of water vapor

ρ_s : Density of water vapor at saturation

Relative humidity is expressed as a percentage. At 100% RH, condensation occurs, and at 0% RH, the air is completely dry.

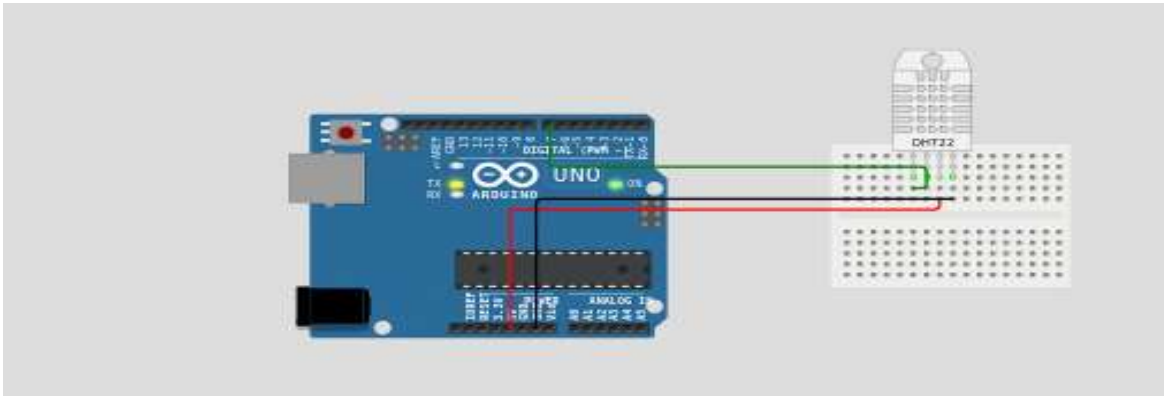
What is relative humidity?

When you look at the datasheet of the DHTxx sensors, you will see that they measure the relative humidity (RH) of the air and not the absolute humidity. But what's the difference? The absolute humidity is the amount of water vapor in the air (expressed in g/m³), regardless of temperature. The relative humidity does take temperature into account. Relative humidity is the ratio between the actual amount of water vapor present in the air and the maximum amount of water vapor that the air can hold at a given temperature. Warm air can hold more water than cold air. This means that for the same amount of water vapor in the air, the relative humidity in cool air will be higher than that in warm air. At 100 percent relative humidity, the air is saturated and is at its dewpoint.

Setup of DHT22 module on Arduino

If you are using an original DHT22 sensor then it will have a NTC thermistor and the sensor module inside it, The humidity sensing component consists of a moisture-holding substrate sandwiched in between two electrodes. When the substrate absorbs water content, the resistance between the two electrodes

decreases. The change in resistance between the two electrodes is proportional to the relative humidity. Higher relative humidity decreases the resistance between the electrodes, while lower relative humidity increases the resistance between the electrodes. This change in resistance is measured with the onboard MCU's ADC and the relative humidity is calculated. We can connect all the required wires to Arduino and write the code to get all the data out from the sensor. The following image shows the circuit diagram for interfacing the DHT22 sensor module with Arduino.



But before all of this, you'll need to install the DHTLib library. It's easy to install, just download DHTLib.zip file below and open up Arduino IDE. After this installation, upload this example program to the Arduino and open the serial monitor.

CODE:

```
#include "DHT.h"

#define DHTPIN 2 // what pin we're connected to

// Uncomment whatever type you're using!

#define DHTTYPE DHT11 // DHT 11
// #define DHTTYPE DHT22 // DHT 22 (AM2302)
// #define DHTTYPE DHT21 // DHT 21 (AM2301)

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

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

```

Serial.println("DHTxx test!");
dht.begin();
}

void loop() {
    // Wait a few seconds between measurements.
    delay(2000);

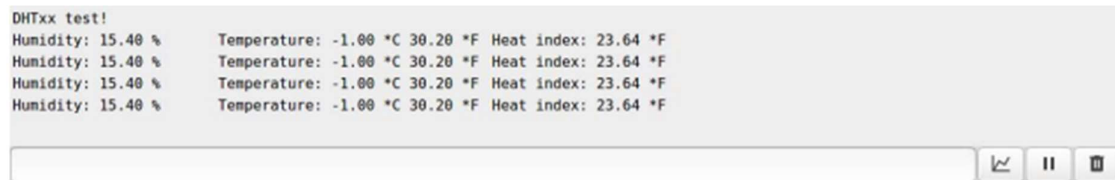
    // Reading temperature or humidity takes about 250 milliseconds!
    // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
    float h = dht.readHumidity();
    // Read temperature as Celsius
    float t = dht.readTemperature();
    // Read temperature as Fahrenheit
    float f = dht.readTemperature(true);
    // Check if any reads failed and exit early (to try again).
    if (isnan(h) || isnan(t) || isnan(f)) {
        Serial.println("Failed to read from DHT sensor!");
        return;
    }

    // Compute heat index
    // Must send in temp in Fahrenheit!
    float hi = dht.computeHeatIndex(f, h);
    Serial.print("Humidity: ");
    Serial.print(h);
    Serial.print(" %\t");
    Serial.print("Temperature: ");
    Serial.print(t);

```

```
Serial.print(" *C ");  
Serial.print(f);  
Serial.print(" *F\t");  
Serial.print("Heat index: ");  
Serial.print(hi);  
Serial.println(" *F");  
}
```

Output :



```
DHTxx test!  
Humidity: 15.40 %    Temperature: -1.00 *C 30.20 *F Heat index: 23.64 *F  
Humidity: 15.40 %    Temperature: -1.00 *C 30.20 *F Heat index: 23.64 *F  
Humidity: 15.40 %    Temperature: -1.00 *C 30.20 *F Heat index: 23.64 *F  
Humidity: 15.40 %    Temperature: -1.00 *C 30.20 *F Heat index: 23.64 *F
```

Experiment no :6

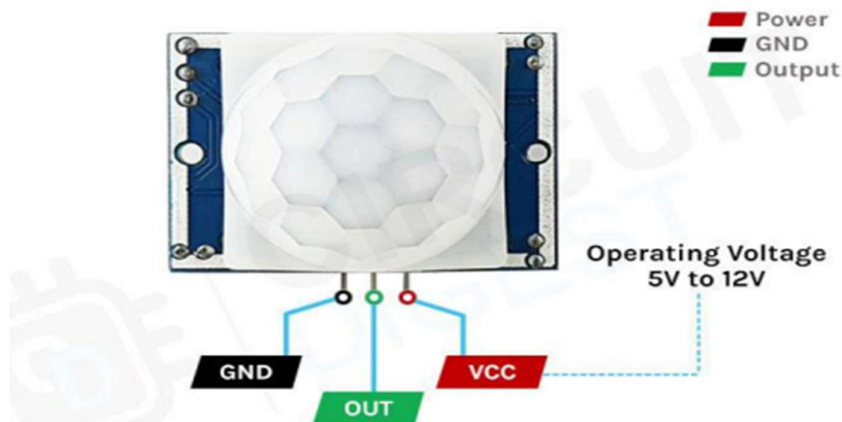
To interface PIR Sensor with Arduino/ Raspberry Pi and write a program to check the motion of PIR sensor.

The PIR motion sensor is ideal to detect movement. PIR stand for “Passive Infrared”. Basically, the PIR motion sensor measures infrared light from objects in its field of view. So, it can detect motion based on changes in infrared light in the environment. It is ideal to detect if a human has moved in or out of the sensor range.



Wiring the PIR motion sensor to an Arduino is pretty straightforward – the sensor has only 3 pins.

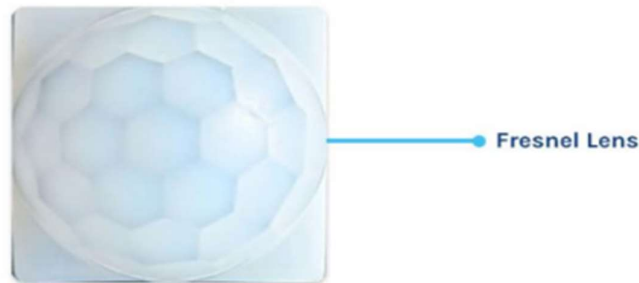
- GND – connect to ground
- OUT – connect to an Arduino digital pin
- 5V – connect to 5V



How does a PIR sensor work?

Every object with a temperature above absolute zero emits IR radiation, which is invisible to the human eye but carries information about an object's temperature and movement. The topmost part of the PIR sensor consists of a lens with concentric grooves carved into the plastic. These are called Fresnel lens.

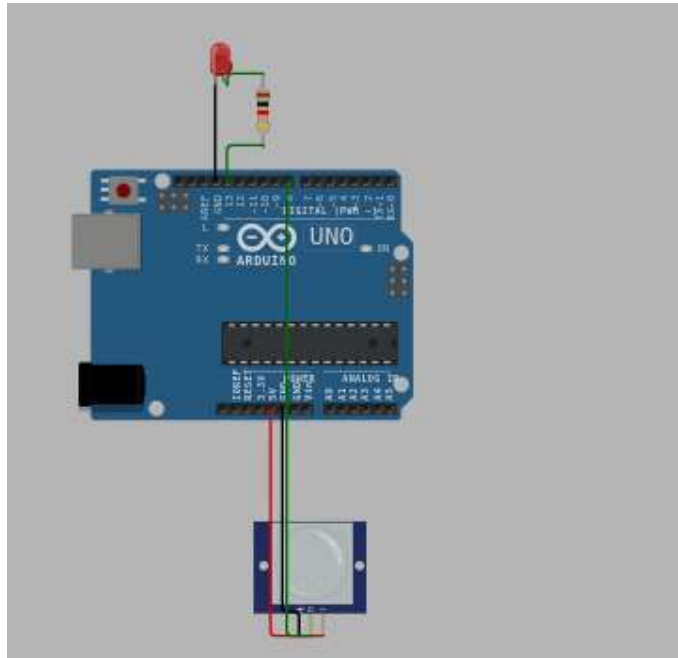
The contours on this plastic helps in gathering parallel light rays at a focal point, each one of them acting as individual refracting surfaces. These surfaces increase the range and field of view of PIR sensor, the lens is divided into several sections, each of them being a separate Fresnel lens.



The operation of the PIR Sensor unfolds in several stages. Upon powering up, the sensor initiates an initialization phase during which it calibrates itself to the ambient IR radiation conditions. This calibration helps the sensor adapt to the background temperature and minimize false triggers. Subsequently, the sensor continuously monitors its surroundings, comparing the detected IR radiation pattern with the previously calibrated background. When a warm object, such as a human or a pet, moves within the sensor's field of view, it causes a dynamic alteration in the IR radiation pattern received by the pyroelectric sensor. This change is promptly detected by the sensor.

The sensor's internal circuitry processes the signal output from the pyroelectric sensor. It includes signal amplification and filtering mechanisms to ensure accurate and reliable motion detection. The final output from the HC-SR501 PIR Sensor is typically a digital signal. This signal remains in a low state when no motion is detected. However, when motion occurs within the sensor's sensing range, the signal transitions to a high state. The duration for which the signal stays high is determined by the time-delay adjustment potentiometer on the sensor.

After this preset time, the signal returns to its low state, indicating the absence of motion.



Connect the PIR sensor's VCC (or +) pin to the Arduino's 5V output. Connect the GND (-) pin of the PIR sensor to any GND pin on the Arduino. Attach the OUT (or signal) pin of the PIR sensor module to a digital input pin on the Arduino (e.g., pin 5).

The following code gives the representation of how PIR sensor will work with Arduino Board.

CODE:

```
int led = 13; // the pin that the LED is attached to
int sensor = 2; // the pin that the sensor is attached to
int state = LOW; // by default, no motion detected
int val = 0; // variable to store the sensor status (value)
void setup() {
  pinMode(led, OUTPUT); // initialize LED as an output
  pinMode(sensor, INPUT); // initialize sensor as an input
  Serial.begin(9600); // initialize serial
}
void loop(){
  val = digitalRead(sensor); // read sensor value
  if (val == HIGH) { // check if the sensor is HIGH
    digitalWrite(led, HIGH); // turn LED ON
    delay(100); // delay 100 milliseconds
```

```
if (state == LOW) {  
  Serial.println("Motion detected!");  
  state = HIGH; // update variable state to HIGH  
}  
}  
else {  
  digitalWrite(led, LOW); // turn LED OFF  
  delay(200); // delay 200 milliseconds  
  
  if (state == HIGH){  
    Serial.println("Motion stopped!");  
    state = LOW; // update variable state to LOW  
  }}  
}}
```

Output :



A screenshot of a serial monitor window. The window has a light gray background and a dark gray border. The text "hello, I found you...hey...hey...hey..." is displayed in a monospaced font, repeated three times. At the bottom right of the window, there are three icons: a line graph, a play button, and a trash can.

```
hello, I found you...hey...hey...hey...  
hello, I found you...hey...hey...hey...  
hello, I found you...hey...hey...hey...
```

Experiment no :7

To interface PI Camera with Arduino/ Raspberry Pi and write a program to start the camera and to place the clicked pictures on the desktop

Interfacing a Pi Camera with an Arduino board is not a direct process since the Pi Camera uses the Raspberry Pi's Camera Module and communicates through the Raspberry Pi's GPIO pins. If you want to capture images using the Pi Camera and transfer them to your computer, you'll need to use a Raspberry Pi board.

Here's how you can achieve this using a Raspberry Pi and Python programming language:

Hardware Requirements:

- Raspberry Pi Board (e.g., Raspberry Pi 3/4)
- Pi Camera Module
- Arduino Board (for other purposes, if needed)
- Internet connection for Raspberry Pi

Software Requirements:

- Raspbian OS installed on Raspberry Pi
- Python 3 installed on Raspberry Pi

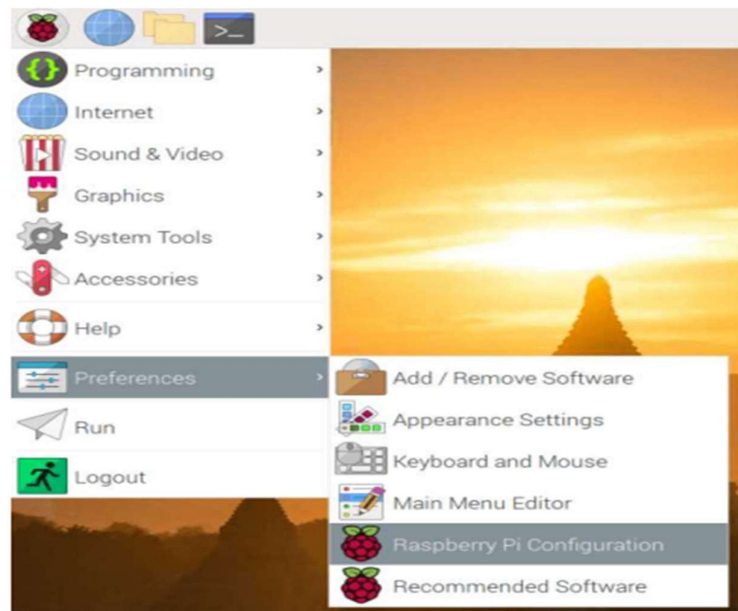
Setup and enable the Pi camera

First, make sure to shutdown and power off your Raspberry Pi. Localize the camera port (don't confuse it with the display port which has a similar connector). The camera port is between the HDMI ports and the jack port. You can see "CAMERA" written next to it.

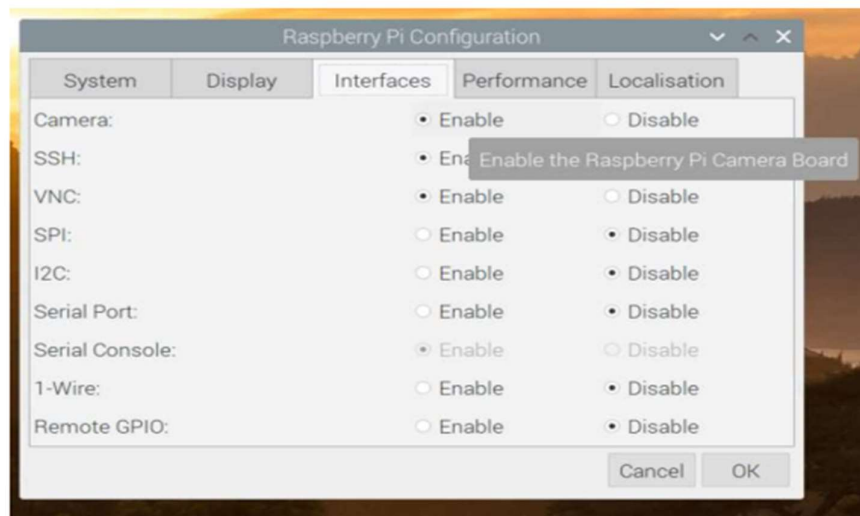


Once you've located the camera port, now make sure to plug the connector in the right way. The blue part should face the jack and USB ports.

Now, power on your Raspberry Pi. To open the settings for the Pi camera, click on the Raspberry Pi icon > "Preferences" > "Raspberry Pi Configuration".



Select the "Interfaces" tab and click on "Enable" next to "Camera:".



Then, click on OK and reboot your Pi so the change will be effective. Take a picture with the Raspberry Pi camera. Now that the camera is plugged and enabled, you can start to take pictures. Here we'll use the `raspistill` command in the terminal – already installed on the Raspberry Pi OS.

First, open a terminal. If you don't have the terminal icon on the top bar, click on the Raspberry

Pi icon > "Accessories" > "Terminal".

First picture with Raspberry Pi and raspistill To take a picture, you'll need to use the raspistill command and also provide one argument: the name of the file for the output, so raspistill can save the photo into that file.

1. Connect the Pi Camera:

- Make sure your Raspberry Pi is turned off.
- Connect the Pi Camera module to the Camera CSI port on the Raspberry Pi board.

2. Enable the Camera Interface:

- Boot up your Raspberry Pi and open the terminal.
- Run `sudo raspi-config` command.
- Navigate to "Interfacing Options" and enable the Camera interface.
- Reboot the Raspberry Pi.

3. Write Python Code to Capture and Transfer Images:

- Write a Python script to capture images using the Pi Camera and transfer them to your desktop computer.
- Here's an example Python code for capturing images and saving them on the Pi:

```
import os
import time
from picamera import PiCamera
# Initialize the PiCamera
camera = PiCamera()
# Set the resolution of the camera (optional)
camera.resolution = (640, 480)
# Path to save captured images
image_path = "/home/pi/Desktop/"
try:
    # Capture an image
    28
    timestamp = time.strftime("%Y%m%d_%H%M%S")
    image_filename = f"image_{timestamp}.jpg"
```

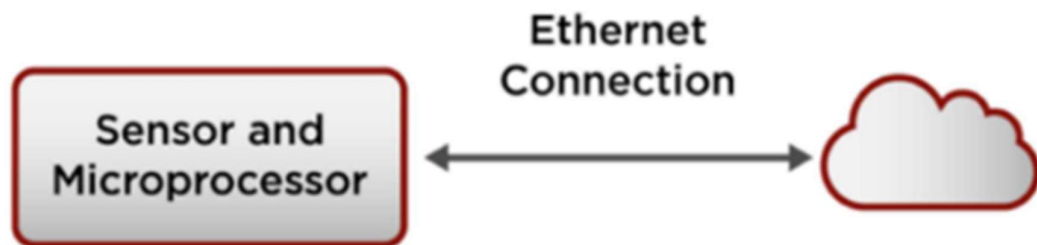
```
image_fullpath = os.path.join(image_path, image_filename)
# Capture the image
camera.capture(image_fullpath)
print(f"Image captured: {image_filename}")
except Exception as e:
    print(f"Error: {str(e)}")
finally:
    # Release the camera resources
    camera.close()
```

Experiment no :8
To transmit and access the sensed data to any cloud
Platform

Getting data from the sensor to the cloud

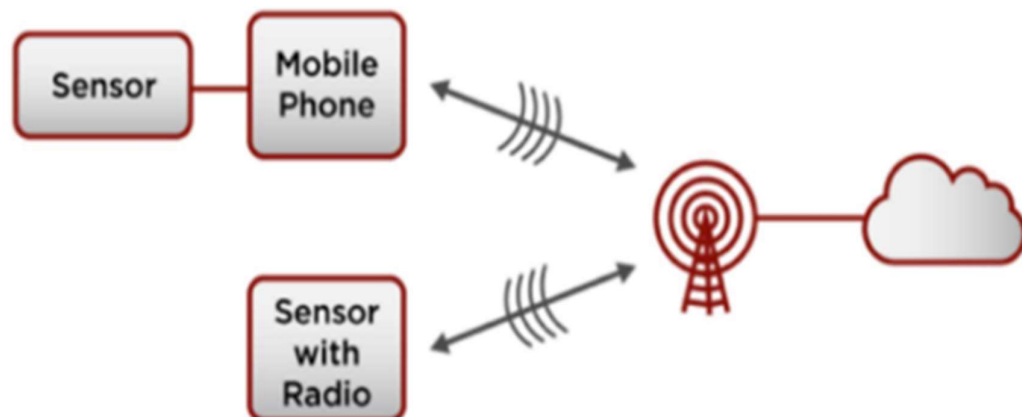
There are several ways to connect devices(any sensor or mobile) to the cloud. The evolution of various ways of sending data from a device to cloud started from 1970's. The evolution is still in process and we are approaching better ways of sending data from device to cloud. Given below are some methods of sending data from one device to cloud. The complexity of methodology adopted increases as you read.

Sensor To Cloud Over Ethernet



One of the simplest , rather evolved in the 1970s and 1980s, before the development of all the radio links. The Sensor includes a processor which is tough enough to configure the data uploading to cloud. The Ethernet connection would connect to wired Internet service. The problem: Some places don't have wired Internet. The processor could also have the ability to update or modify the functions of the sensor. There is no involvement of radio link.

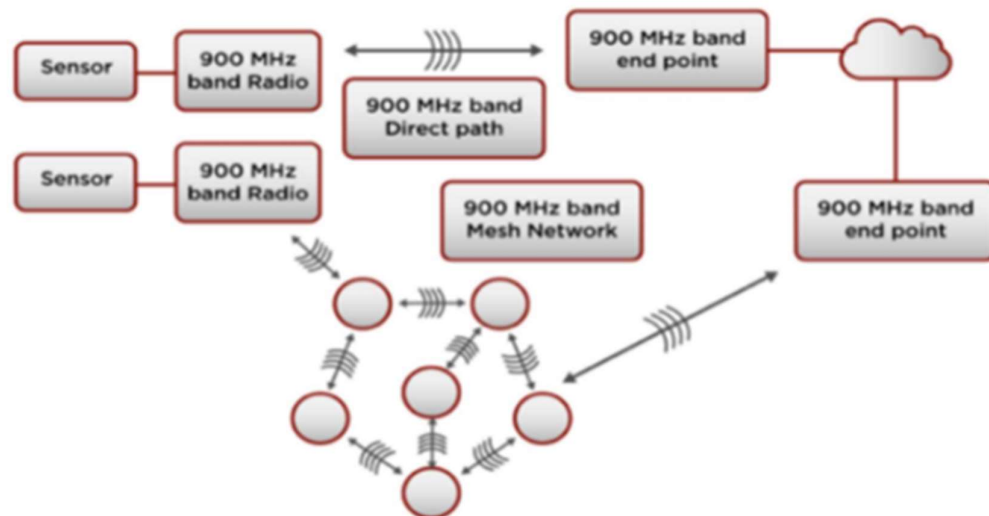
Sensor To Mobile-Phone Network To Cloud



The mobile phone network began to develop in the early 1980s. These early cellular networks were the first widely available radio link for connecting sensors to the cloud. The disadvantages are that

- The sensor still needs a wired connection to a mobile phone or needs an expensive custom radio in the mobile-phone band to connect to the phone tower.
- The uplink (sensor to the phone tower) radio transmitter needs a fair amount of power to reach the tower
- The user needs to pay the mobile network provider for usage.

Sensor To Long-Range Radio To The Cloud



Regulators established several license free radio bands as early as 1947. But these did not attract much interest until mobile phones really caught on in the late 1990s. IEEE 802.15.4 standard has frequencies of two bands at 902-928 MHz and 2400-2483 MHz. (There are other standards, such as Zigbee, in one or both of these bands.)

One configuration that uses these bands is a mesh network. It consists of many small, low-power radios connected to each other to relay data from remote sensors at the outer edges of an area to radios at a collection point. Each collection point has access to the cloud. This allows for wide-area usage by deploying sensors connected to very low-power radios.

Sensor To Wi-Fi Router To Cloud

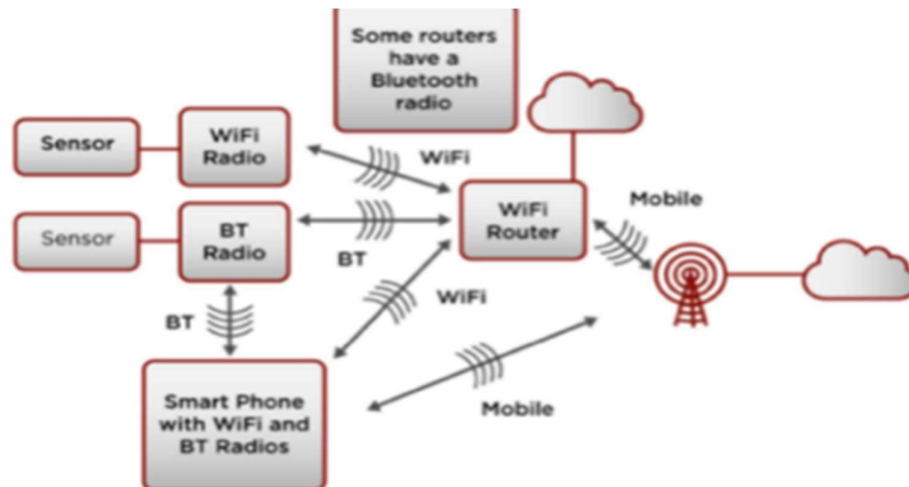


The 2400-2483 MHz band and another license-free band at 5130-5835 MHz were the original frequency bands of the 802.11 Wi-Fi Standard (created in 1997). They are used primarily for Wi-Fi access points, which is widely available in cities these days. The largest number of these routers are in homes, businesses, and public gathering places (coffee shops, malls, and airports).

Industry and infrastructure used a small number of more specialized routers. This is the most widely-used way today to connect mobile devices (laptops, tablets, smart phones) to the cloud. In fact, most applications in smartphones connect to the cloud primarily through a Wi-Fi router.

Shortly after Wi-Fi-capable smartphones became available, remote sensors that could connect directly to a Wi-Fi router also began to appear. Small sensors with low power Wi-Fi radio are placed within the range of wifi router. Internet connection is provided later.

Sensor To Mobile Phone To Cloud



The sensor just needs to connect to a mobile phone instead of connecting directly to a Wi-Fi router. The main reason for this is to allow the mobile-phone user to interact directly with the sensor before sending the information up to the cloud.

These applications are served by the Bluetooth standard, created in 1998. It was added to the 802.15.4 standard in 2003 but continues to maintain its own independent working group. It works in the same 2400-2483 MHz license-free band used by one of the Wi-Fi bands.

Recently, a version of Bluetooth called Bluetooth Low Energy (BLE) was introduced that draws very little power and is well-suited to simple sensors with low data rates or low on-off duty cycles.

This has led to a rapid increase in small sensors (like Fitbit fitness trackers and others like it) that connect to mobile phones, which in turn connect to a Wi-Fi router or a mobile network.

Another recent development is Wi-Fi routers that contain a Bluetooth radio as well as Wi-Fi. With these, BLE sensors can connect directly to the router and on to the cloud without passing through a mobile phone.

The figure above, with three radio links, looks complicated, but it shows that with the choice of radio standards and improvements in processing, there will be more and more configurations to enable the Internet of Things.

From Sensor to Cloud: A Plug and Play Approach Evolving

Today, more powerful, evolved gateways, can function either as dedicated devices or as a virtual part of a system. They play a new role in receiving, translating, processing and transmitting data as transparent information to the spectrum of cloud interfaces. So this is enabled by the new cloud API for IOT gateways. It is essentially a middle-ware and glue logic solution to enable simple orchestration of wired and wireless sensor networks as well as embedded system configurations. The cloud API provides application-ready software modules. They act as blueprints for original equipment manufacturers (OEMs) to develop their own applications. Therefore it helps in removing complexity and creating a smart path to connect all types of sensor networks to any cloud platform.

Amplifying the importance of gateways

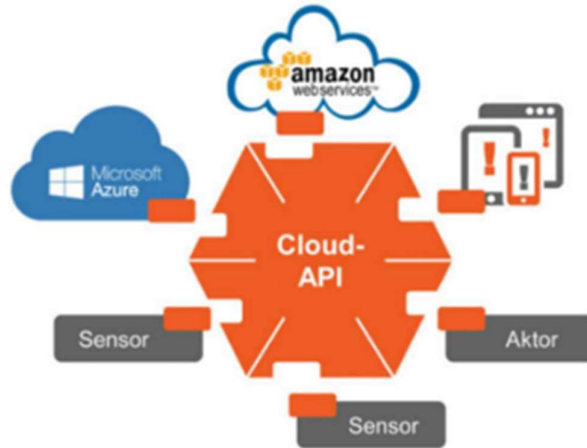


Fig. 1: The cloud API environment

Gateways are complex devices with excellent transcoding and decision-making capabilities. Using integrated logic, these collect, analyse and transcode sensor data. Later it determines whether it goes to the field, the cloud or perhaps another gateway. Their secure end-to-end encryption further allows them to structure and move data consistently. For example, enabling bidirectional communication with a specific cloud solution.