Internet of Things Laboratory

Submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

(Information Technology)

Submitted by:
Shivay Bhandari(D3 IT-B2)
CRN: 1921142
URN: 1905398

Submitted to:
Dr. Kamaljeet Kaur
Assistant Professor

Department of Information Technology,
Guru Nanak Dev Engineering College,
Ludhiana-141006

INDEX

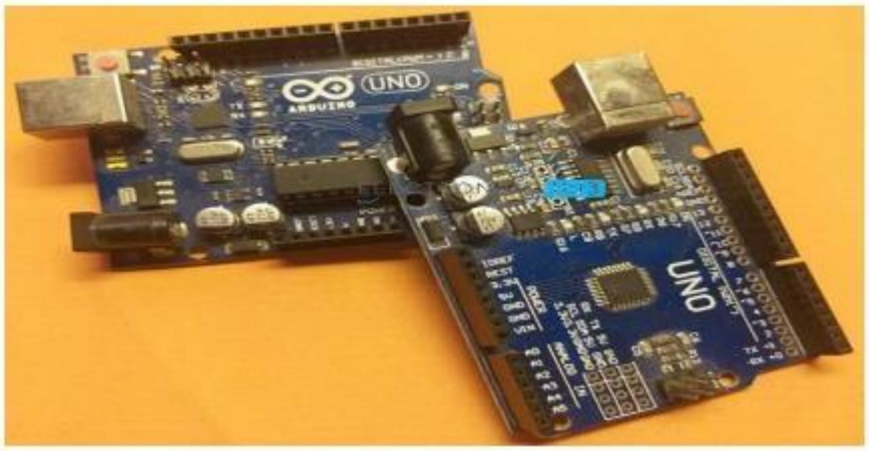
Practical 1

Familiarization with Arduino/ Raspberry Pi and perform necessary software installation.

Raspberry Pi and Arduino are two very popular boards among electronics DIY builders, hobbyists and even professionals. Raspberry Pi and Arduino are quite different boards. While Arduino is aimed at quick programming and circuit prototyping, Raspberry Pi acts as a learning tool for Computer Programming (but you can find Raspberry Pi is several DIY Projects as well). Each board has its own advantages and disadvantages.

Let us take a closer look at these two boards, understand the differences between Raspberry and Arduino and also build a comparison of Raspberry Pi vs Arduino in a tabular format. If you want to decide between the two, then it depends on the requirement of your project but we hope this article will helpful in understanding the differences between these two boards and helps you in selecting the right board for your next project.

Arduino:
Let us start with Arduino. Arduino was developed by Massimo Banzi Et Al. in Ivrea, Italy. Arduino is a simple electronics prototyping tool with open-source hardware and software. Arduino is essentially a Microcontroller development board using which you can Blink LEDs, accept inputs from Buttons, read data from Sensors, control Motors and many other "Microcontroller" related tasks.



The most popular Arduino board is the Arduino UNO, which is based on ATmega328P Microcontroller from Atmel (now Microchip). Coming to the software side of Arduino, all Arduino boards can be programmed in C and C++ programming languages using special software called Arduino IDE. The Arduino IDE consists of all the toolchains for editing source code, compiling and programming the Microcontroller on the Arduino board.
If you have previous experience with Microcontrollers like 8051, Atmel or PIC Microcontrollers, then you probably understand the lengthy process of developing applications using these microcontrollers. If you are not familiar, then let us see the process briefly.

First, you have to write the application software (the main source code) in a dedicated
IDE (like Keil, Atmel Studio or PIC’s MPLAB IDE). Then you have to compile the code and
generate the binary file in the form of a .hex file. Now using a special hardware called
“Programmer”, you have to upload the hex file to the target microcontroller using programmer
software.

Arduino simplified this process with plug-and-play style quick programming. Using a
single software (the Arduino IDE), you can write the code, compile it and upload it to the
Microcontroller. You also don’t need separate hardware for uploading the program. Simply plugin the
Arduino board to a Computer through USB Port, hit the upload button, et voila, the
Microcontroller on Arduino board is ready to do its tasks.
Another important thing about Arduino is it is open-source. This means the design files
and the source code for software and libraries are freely available. You can use the hardware
design files as a reference and essentially make your own Arduino board.

Raspberry Pi:
The Raspberry Pi was developed by Eben Upton at the University of Cambridge in the United
Kingdom with the aim of teaching and improving programming skills of students in developing
countries. While Arduino is a Microcontroller based development board, the Raspberry Pi is a
Microprocessor (usually an ARM Cortex A Series) based board that acts as a computer.
You can connect several peripherals like a Monitor (through HDMI or AV Port), Mouse
and Keyboard (through USB), connect to internet (through Ethernet or Wi-Fi), add a Camera
(through the dedicated Camera Interface), just like we do to our desktop computer.
Since the entire Computer (the Processor, RAM, Storage, Graphics, Connectors, etc.) is
sitting on a single Printed Circuit Board, the Raspberry Pi (and other similar boards) are called as
Single Board Computers or SBC

As Raspberry Pi is essentially a full computer, it can run an Operating System. The Raspberry Pi Foundation, the organization which is responsible for designing and developing Raspberry Pi SBC, also provides a Debian based Linux Distribution called the Raspberry Pi OS (previously known as the Raspbian OS).

Another important thing about Raspberry Pi is, as it is a Linux based Computer, you can develop software using several Programming Languages like C, C++, Python, Java, HTML, etc. Despite its original intentions, which is to promote programming (like Python and Scratch Programming Languages) in schools, the original Raspberry Pi SBC became extremely popular among DIY builders, hobbyists and enthusiasts for developing several applications like Robotics, Weather Stations, Camera based security systems etc.

Due to its success and popularity, the Raspberry Pi Foundation is continuously updating and releasing new versions of Raspberry Pi with the latest one being the Raspberry Pi 4 Model B. The hardware design files and the firmware of Raspberry Pi are not open-source.

Differences between Raspberry Pi and Arduino:
Both Arduino and Raspberry Pi are good teaching tools for students, beginners and hobbyists. Let us see some of the differences between Raspberry Pi and Arduino.

* The main difference between them is: Arduino is microcontroller board, while Raspberry Pi is a microprocessor based mini computer (SBC).

* The Microcontroller on the Arduino board contains the CPU, RAM and ROM. All the additional hardware on Arduino Board is for power supply, programming and IO Connectivity. Raspberry Pi SBC has all features of a computer with a processor, memory, storage, graphics driver, connectors on the board.

* Raspberry Pi needs an Operating System to run. Arduino doesn’t need any operating system. All you need is a binary of the compiled source code.

* Raspberry Pi comes with a fully functional operating system called Raspberry Pi OS (previously known as Raspbian OS). Although Pi can use different operating systems, Linux is preferred by Raspberry Pi Foundation. You can install Android, if you want.

Arduino does not have any operating system. You just need a firmware instructing the Microcontroller what task to do.

* The clock speed of Arduino is 16 MHz while the clock speed of Raspberry Pi is around 1.2 GHz.

* Raspberry Pi is good for developing software applications using Python, while Arduino is good for interfacing Sensors and controlling LEDs and Motors.

* This doesn’t mean we cannot connect sensors and LEDs to Raspberry Pi. To encourage learning programming by controlling hardware, the Raspberry Pi consists of a 40-pin GPIO, through which you can connect different electronic components like LEDs, Buttons, Sensors, Motors etc. On Arduino, the GPIO is called as Digital IO (for digital Input and Output) and Analog IN (for Analog Input).

* Using Arduino Shields, which plug into the Arduino Pin headers, you can add a dedicated feature or functionality like a Motor Driver, Ethernet Connection, SD Card Reader, WiFi, Touchscreens, cameras etc. to Arduino. While Raspberry Pi is a self-contained board, you can add external hardware like Touchscreen, GPS, RGB panels etc. to Raspberry Pi.

The Raspberry Pi Hardware Attached on Top or HAT Expansion Boards are inspired by Arduino Shields, using which you can add additional functionality to Raspberry Pi. They are connected to the GPIO Pins.

* The power requirements of Raspberry Pi and Arduino are completely different. Even though they both are powered by USB (micro-USB or USB Type C for Raspberry Pi and USB Type B for Arduino), Raspberry Pi needs more more current than Arduino. So, you need a power adapter for Raspberry Pi but you can power Arduino from the USB port of a Computer.

* Power interruption for Raspberry Pi may cause damage to the hardware, software or applications. In case of Arduino, if there is any power cut it again restarts. So, Raspberry Pi must be properly shutdown before disconnecting power.

* Arduino uses Arduino IDE for developing the code. While Raspberry Pi can use Python IDLE, Eclipse IDE or any other IDE that is supported by Linux. You can also program using the terminal itself with any text editor like Vim.

* Using the open-source hardware and software files of Arduino, you can essentially create your own Arduino board. This is not possible with Raspberry Pi as it is not open-source.

* The cost of original Arduino UNO is $23 but there are several clones of Arduino which are available for less than $4. Coming to Raspberry Pi, the original Raspberry Pi SBC was around $35, but the latest Raspberry Pi 4 Model B is available in different price points ($35, $55 or $75) depending on the memory configuration.

Practical 2

To demonstrate the communication modules like BLE, WIFI, XBEE:

Bluetooth Low-Energy (BLE)
Bluetooth Low-Energy is much more than just a low-energy version of Bluetooth Classic. In fact, it's applications are completely different than for normal Bluetooth.
Bluetooth LE is probably the most common type of wireless functionality for the products I help develop. It is designed to transmit/receive small amounts of data on a fairly infrequent basis, all while consuming extremely low amounts of power.
BLE has many applications but one of the most common is transmitting sensor data. A sensor device that measures the temperature once per minute, or a GPS device that records and transmits its location every 10 minutes, are a few examples.
In many cases, Bluetooth LE products are powered only from a small coin cell battery. If data is only sent infrequently, a BLE device running from a coin cell battery may have a battery life of a year or longer.
Bluetooth LE is widely supported by mobile phones and tablets making it an ideal solution for interfacing your product to a mobile app. It also supports a decent transfer speed of up to 1Mbps (classic Bluetooth can do up to 2-3 Mbps).

WiFi
WiFi Direct uses the same basic technology as traditional WiFi. It uses the same frequency and offers similar bandwidth and speed. But, it doesn't require an access point, allowing two devices to have a direct connection similar to Bluetooth.
The advantage of WiFi Direct over Bluetooth is mainly faster transfer speeds. In fact, WiFi Direct is over a hundred times faster than Bluetooth. That speed comes at a price though and that price is mainly higher power consumption.

XBee
The XBee Python Library provides the ability to communicate with remote nodes in the network, IoT devices and other interfaces of the local device. The communication between XBee devices in a network involves the transmission and reception of data.

- Send data
- Receive data

Send data
A data transmission operation sends data from your local (attached) XBee to a remote device on the network. The operation sends data in API frames. The XBee Python Library abstracts the process so you only have to specify the device to send data to and the data itself.

You can send data either using a unicast or a broadcast transmission. Unicast transmissions route data from one source device to one destination device, whereas broadcast transmissions are sent to all devices in the network.

Send data to one device

Unicast transmissions are sent from one source device to another destination device. The destination device could be an immediate neighbor of the source, or it could be several hops away.

Data transmission can be synchronous or asynchronous, depending on the method used.

Synchronous operation

This type of operation is blocking. This means the method waits until the transmit status response is received or the default timeout is reached.

The XBeeDevice class of the API provides the following method to perform a synchronous unicast transmission with a remote node of the network:

Practical 3

To find the IP address of Computer/ other devices

Windows 7 Instructions:

First, click on your Start Menu and type cmd in the search box and press enter.A black and white window will open where you will type ipconfig /all and press enter.There is a space between the command ipconfig and the switch of /all. Your ip address will be the IPv4 address.

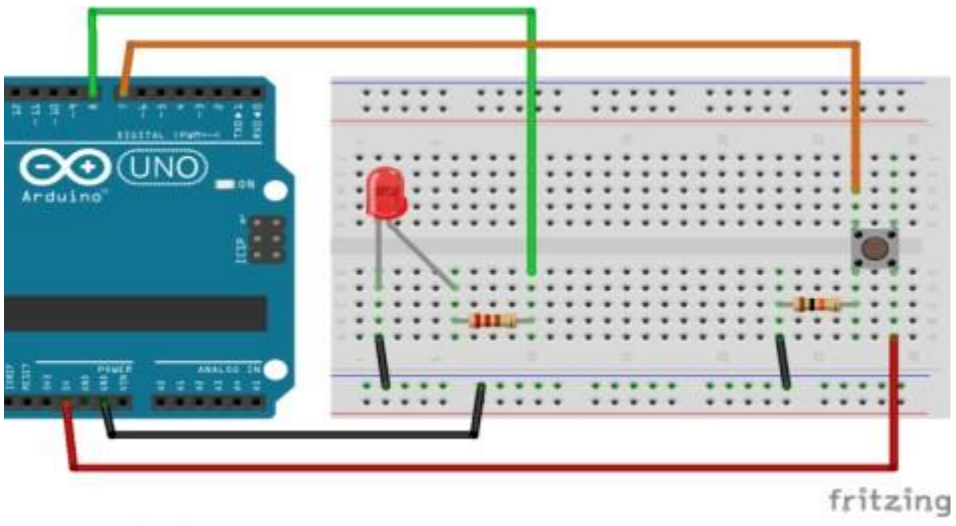
Windows 10 Instructions:
Right click the windows button in the lower left of your screen. Select command prompt from the list.A black and white window will open where you will type ipconfig /all and press enter.There is a space between the command ipconfig and the switch of /all.When a technician requests the IP address of your Ethernet card, the information that follows after the title of Ethernet Local Area Adapter gigabit connection. Your IP address will be the IPv4 address.The wireless card information will follow the title of Wireless Lan adapter connection. Any description that includes the word ‘Virtual’ is not the information that is needed for creating an internet connection for you.

Practical 4

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

Arduino circuit with an LED and a button
To build the circuit you will need those components:
* Arduino board (any board, if you don’t have Uno you can easily adapt by finding corresponding pins).
* Breadboard.
* LED – any color.
* Push button.
* 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:
* First, make sure to power off your Arduino – remove any USB cable.
* Plug a black wire between the blue line of the breadboard and a ground (GND) pin on the Arduino board.
* Plug the LED. You can notice that the LED has a leg shorter than the other. Plug this shorter leg to the ground (blue line here) of the circuit.
* Connect the longer leg of the LED to a digital pin (here pin no 8, you can change it). Add a 220 Ohm resistor in between to limit the current going through the LED.
* Add the push button to the breadboard, like in the picture.
* Connect one leg of the button to the ground, and put a 10k Ohm resistor in between. This resistor will act as a “pull down” resistor, which means that the default button’s state will be LOW.
* Add a red wire between another leg of the button and VCC (5V).
* Finally, connect a leg of the button (same side as the pull down resistor) to a digital pin.



Code:-

The code:

```
#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);
 }
}
```

Practical 5

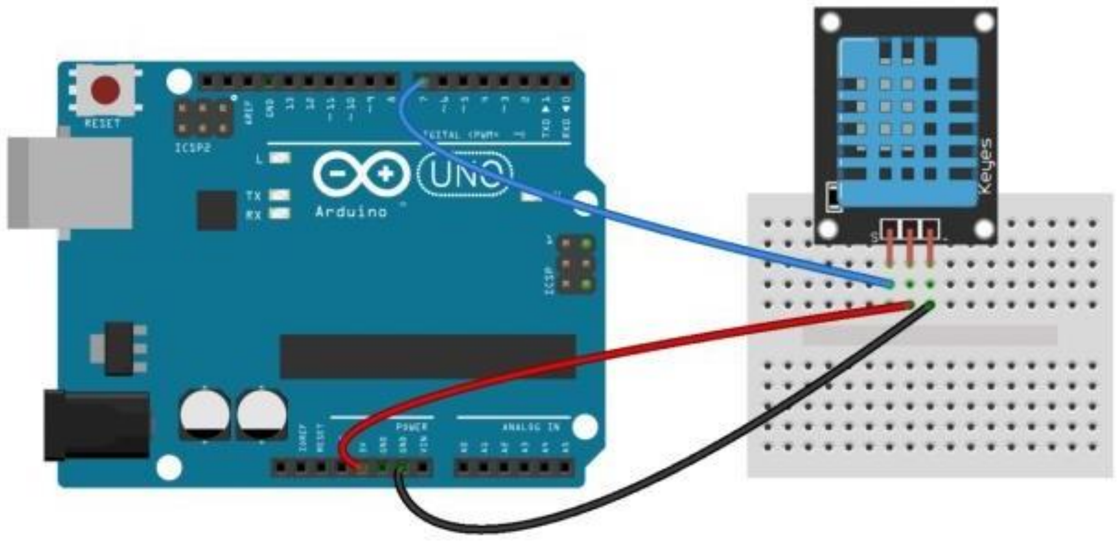
To interface DHT11/ DHT22 sensor with Arduino/ Raspberry Pi and write a program to print temperature and humidity readings.

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.

How the DHT11 Measures Humidity and Temperature:

The DHT11 detects water vapor by measuring the electrical resistance between two electrodes. The humidity sensing component is a moisture holding substrate with electrodes applied to the surface. When water vapor is absorbed by the substrate, ions are released by the substrate which increases the conductivity between the electrodes. 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.The DHT11 measures temperature with a surface mounted NTC temperature sensor (thermistor) built into the unit.An IC mounted on the back of the unit converts the resistance measurement to relative humidity. It also stores the calibration coefficients, and controls the data signal transmission between the DHT11 and the Arduino:The DHT11 uses just one signal wire to transmit data to the Arduino. Power comes from separate 5V and ground wires. A 10K Ohm pull-up resistor is needed between the signal line and 5V line to make sure the signal level stays high by default (see the datasheet for more info).There are two different versions of the DHT11 we might come across. One type has four pins, and the other type has three pins and is mounted to a small PCB. The PCB mounted version is nice because it includes a surface mounted 10K Ohm pull up resistor for the signal line. Here are the pin outs for both versions:

How to Set Up the DHT11 on an Arduino:



```
#include <dht.h>
dht DHT;
#define DHT11_PIN 7
void setup(){
Serial.begin(9600);
}
void loop(){
 int chk = DHT.read11(DHT11_PIN);
 Serial.print("Temperature = ");
 Serial.println(DHT.temperature);
 Serial.print("Humidity = ");
 Serial.println(DHT.humidity);
 delay(1000);
}
```

We should see the humidity and temperature readings displayed at one second intervals.

If you don’t want to use pin 7 for the data signal, you can change the pin number in line 5 where

Code:-

```
 #define DHT11_PIN 7.
```

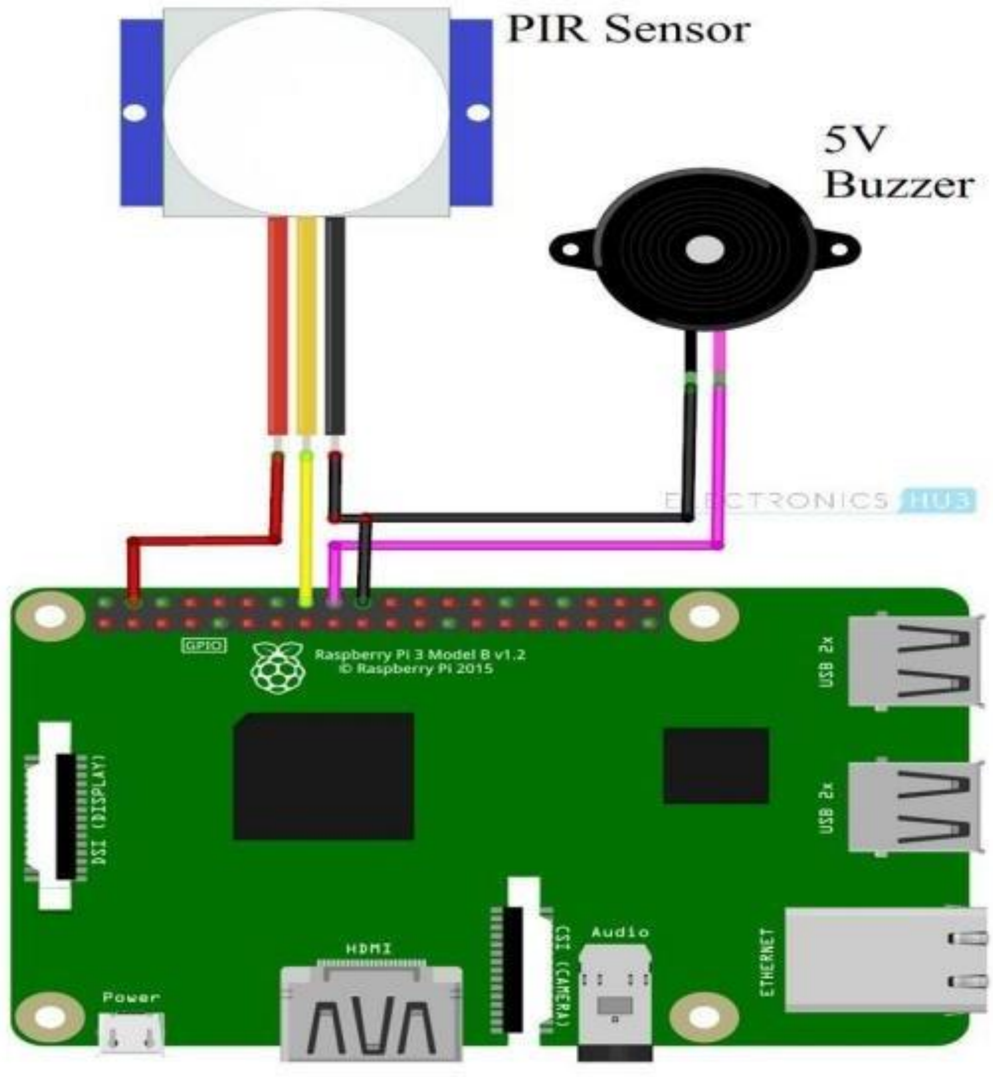
Display Humidity and Temperature on an LCD

```
#include <dht.h>
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
dht DHT;
#define DHT11_PIN 7
void setup(){
lcd.begin(16, 2);
}
void loop(){
 int chk = DHT.read11(DHT11_PIN);
 lcd.setCursor(0,0);
 lcd.print("Temp: ");
 lcd.print(DHT.temperature);
 lcd.print((char)223);
 lcd.print("C");
 lcd.setCursor(0,1);
 lcd.print("Humidity: ");
 lcd.print(DHT.humidity);
 lcd.print("%");
 delay(1000);
}
```

Practical 6

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

A PIR sensor can detect changes in the amount of infrared radiation impinging upon it, which varies depending on the temperature and surface characteristics of the objects in front of the sensor.When an object, such as a person, passes in front of the background, such as a wall, the temperature at that point in the sensor's field of view will rise from room temperature to body temperature, and then back again. The sensor converts the resulting change in the incoming infrared radiation into a change in the output voltage, and this triggers the detection. Objects of similar temperature but different surface characteristics may also have a different infrared emission pattern, and thus moving them with respect to the background may trigger the detector as well.
Circuit Diagram:
The following Fritzing based images shows all the connections with respect to the PIR Motion Sensor using Raspberry Pi.



Components Required:
* Raspberry Pi 3 Model B
* PIR Sensor
* 5V Buzzes
* Connecting Wires
* Mini Breadboard
* Power Supply
* Computer

Circuit Design:
Connect the VCC and GND pins of the PIR Motion Sensor to +5V and GND pins of the Raspberry Pi. Connect the DATA Pin of the PIR Sensor to GPIO23 i.e. Physical Pin 16 of the Raspberry Pi. A 5V Buzzer is connected to GPIO24 i.e. Physical Pin 18 of the Raspberry Pi. The other pin of the buzzer is connected to GND.

NOTE:
* I have directly connected the Buzzer to Raspberry Pi. But if you are not sure, connect it through an NPN Transistor.

* From the previous Raspberry Pi Projects, you already know that Raspberry Pi Input pins are 3.3V tolerant i.e. they work on 3.3V Logic.

* If you are wondering why I connected the output Data pin of the PIR Sensor directly to the Raspberry Pi, then you need to be confused as I have checked the output levels of the PIR Sensor on HIGH state and got a result of around 3.5V.

* You can also check for the same and then proceed with a level converter circuit (voltage divider) if it is required.

Code:
The Programming part of the project is implemented using Python. The following is the Python Script for the PIR Motion Sensor using Raspberry Pi.

```
import RPi.GPIO as GPIO
import time
sensor = 16
buzzer = 18
GPIO.setmode(GPIO.BOARD)
GPIO.setup(sensor,GPIO.IN)
GPIO.setup(buzzer,GPIO.OUT)
GPIO.output(buzzer,False)
print "Initialzing PIR Sensor......"
time.sleep(12)
print "PIR Ready.. "
print " "
```

```
try:
while True:
if GPIO.input(sensor):
GPIO.output(buzzer,True)
print "Motion Detected"
while GPIO.input(sensor):
time.sleep(0.2)
else:
GPIO.output(buzzer,False)
except KeyboardInterrupt:
GPIO.cleanup()
```

Working:

The working of the PIR Motion Sensor using Raspberry Pi is very simple. If the PIR Sensor detects any human movement, it raises its Data Pin to HIGH. Raspberry Pi upon detecting a HIGH on the corresponding input pin, will activate the Buzzer.

Applications

The applications of the PIR Motion Sensor using Raspberry Pi project have already been mentioned. Some of them are:

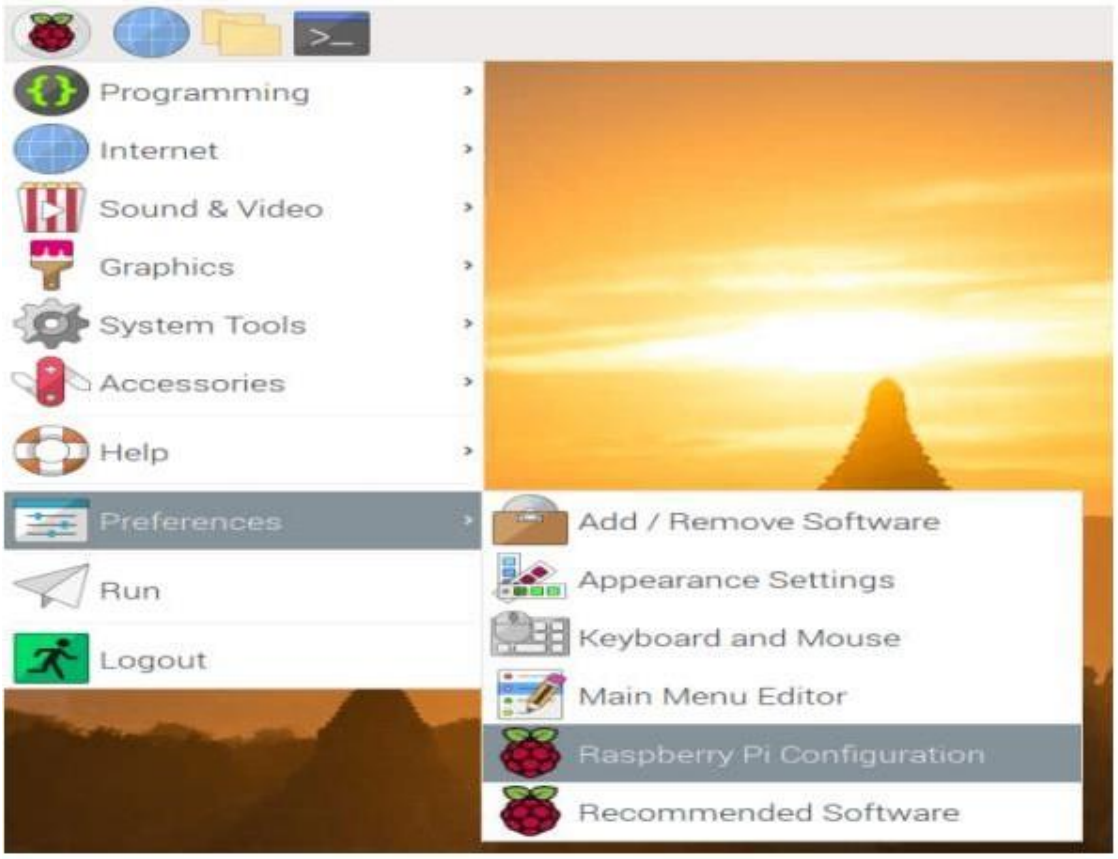
* Automatic Room Light
* Motion Detection
* Intruder Alert
* Automatic Door Opening
* Home Security System

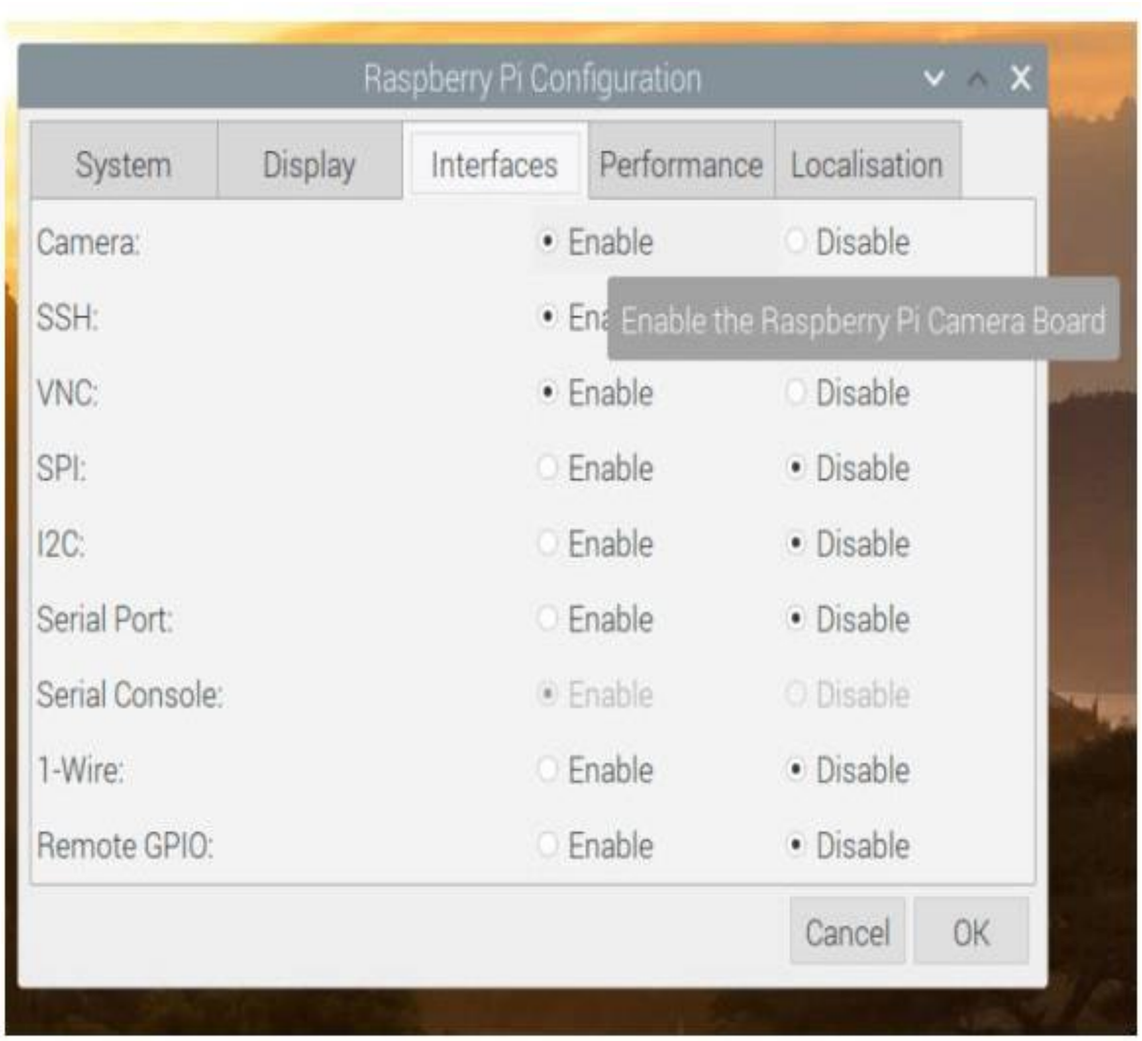
Practical 7

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.

The Raspberry Pi camera module is a great addition to your Pi. It will allow you to develop more advanced applications with vision. And if you thought that taking pictures with the Pi camera would be hard, here you'll see that it's just the opposite.

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 HDMIs 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".




Raspberry Pi Configuration

Camera:	[✓ Enable]	Disable
SSH:	[✓ Enable]	
VNC:	[✓ Enable]	Disable
SPI:	Enable	[✓ Disable]
I2C:	Enable	[✓ Disable]
Serial Port:	Enable	[✓ Disable]
Serial Console:	[✓ Enable]	Disable
1-Wire:	Enable	[✓ Disable]
Remote GPIO:	Enable	[✓ Disable]

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.Let's give it a try.

```
$ raspistill -o ~/Pictures/first_image.jpg
```

This command will take a few seconds to execute, because the camera functionality needs some time for initialization.
After writing
raspistill
, I have added
-o ~/Pictures/first_image.jpg
. What does it mean?
* -o is the option to specify an output file name. Here o is the abbreviation for output.
* ~/Pictures/first_image.jpg is the path + file name for saving the picture. "~" means that

we’ll start from the home directory (from the user you’re connected as), then we have “/Pictures/” to go in the Pictures directory of that user, and finally “first_image.jpg” is the file name. I have used a “.jpg” extension here, you can also use other common image extensions such as “.png”. Don’t forget the extension otherwise you might have trouble opening the file if you share it for example on a Google Drive or on Windows.Note: if you execute the same command again with the same file name, the previous file will be replaced. So, if you want to keep all pictures you take with the Pi camera, make sure to provide a different file name every time. You can also provide a relative path to save the picture from where you are in the terminal.

```
$ cd ~/Pictures/
$ raspistill –o second_image.jpg
```

This will also create a new picture file in your “~/Pictures/” directory.Open the image you’ve taken with raspistillNow that you’ve taken a photo, you might want to actually see that photo. There are 2 main ways of doing that.First, and this is the easiest way since you’re already in the terminal: use the xdg-open command, which is basically the same thing as double clicking on the file to execute it or open it.

```
$ xdg-open ~/Pictures/first_image.jpg
```

Or, if you’re already in the picture directory:

```
$ cd ~/Pictures/
$ xdg-open first_image.jpg
```

We should now be able to see the picture you’ve just taken with the Raspberry Pi camera!The second way to open the picture, is to open a file manager (click on it on the top bar), and use your mouse to find the file + double click on it.Change resolution and flip the picture: When you open an image in the image viewer, the resolution will be displayed on top. For example when I view the picture I see 3280 x 2464, which is quite big. Now let’s say you want to take pictures with a specific resolution: 1280 x 720. Also, depending on how you placed your camera, you may see that the image is upside down. It would be quite boring to have to resize and apply changes for all pictures you take. Instead, you can provide some options with the raspistill command, to directly take pictures with the correct configuration. No need to do any post-processing. And one more thing: by default the raspistill command will wait for 5 seconds before taking a photo. The camera needs about 2 seconds to initialize, so we could also reduce this timeout. Let’s try one command to change the resolution, flip the image vertically, and wait 2 seconds instead of 5.

```
$ raspistill -o ~/Pictures/new_image.jpg -w 1280 -h 720 -vf -t 2000
```

Let’s break this command down:

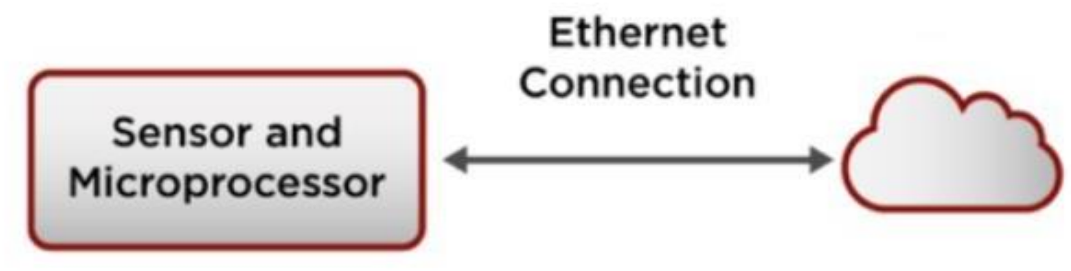
- Raspistill: the command to take a picture.
- -o ~/Pictures/new_image.jpg: the option to add an output file name, as previously seen.
- -w 1280 -h 720: to set the resolution you have to provide 2 options. –w for the width in
pixels, and –h for the height in pixels.
- -vf: this option will vertically flip the picture.
- -t 2000: this is the timeout option. Basically, once you execute the raspistill command in
the terminal, it will wait for x amount of milliseconds before taking the picture and exiting. By default the amount is 5 seconds, or 5000 milliseconds. Here we choose 2000, which means 2000ms or 2 seconds. Make sure to always provide duration in milliseconds for that option.

Practical 8

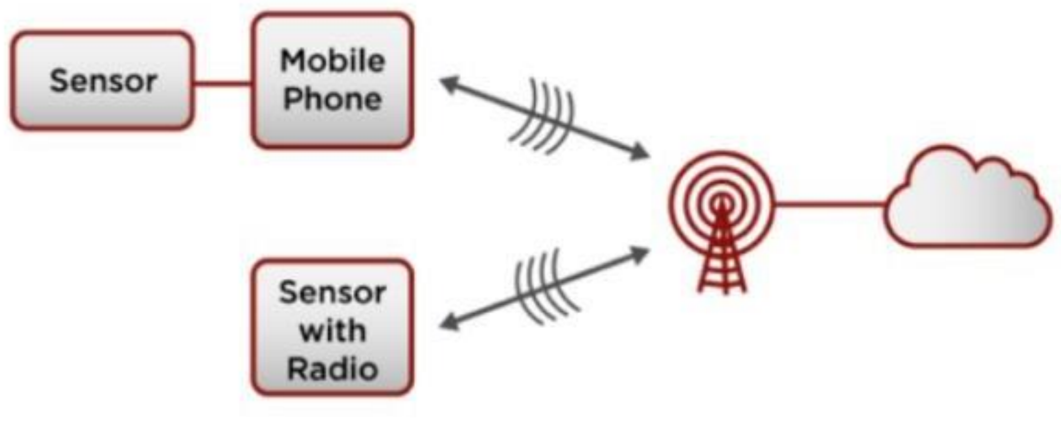
To transmit and access the sensed data to any cloud platform.

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.

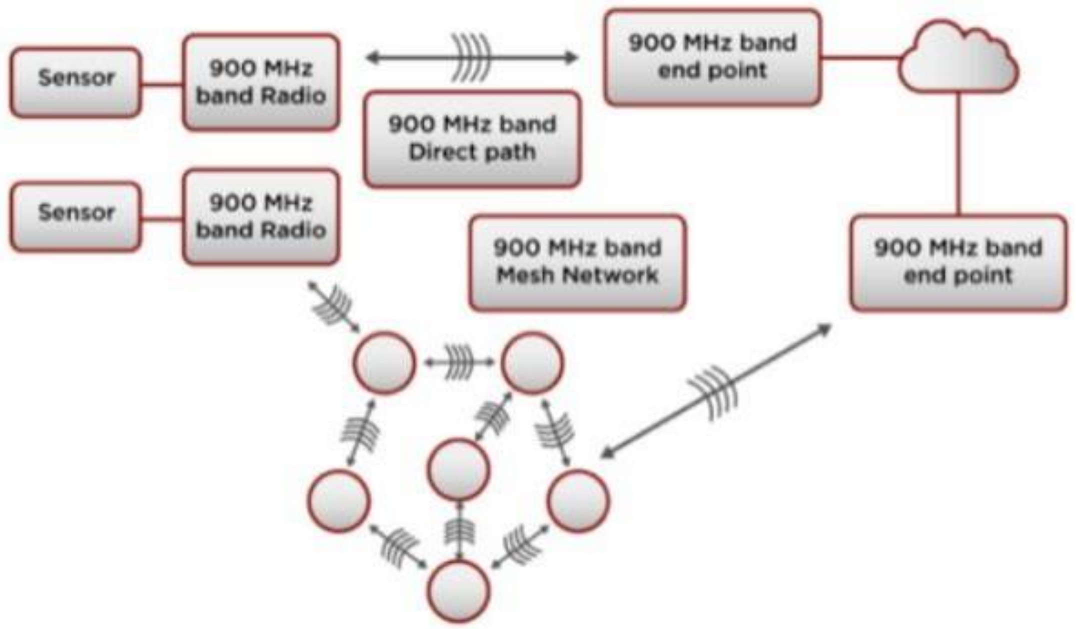
Sensor To Cloud Over Ethernet:
One of the simplest , rather evolved in the 1970s and 1980s, before the development of all the radio links. The Sensor includes a processor which is tough enough to configure the data uploading to cloud. The Ethernet connection would connect to wired Internet service. The 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:

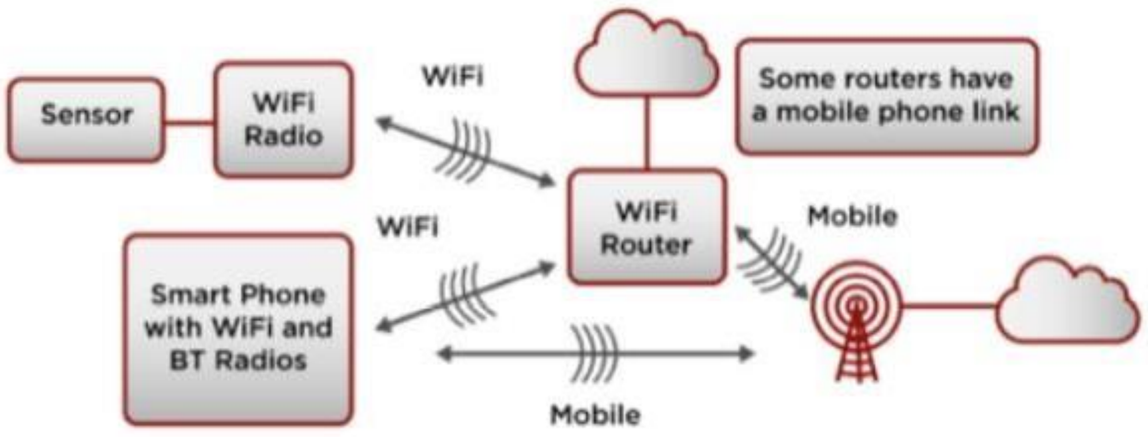


The mobile phone network began to develop in the early 1980s. These early cellular networks were the first widely available radio link for connecting sensors to the cloud. The disadvantages are that
* The sensor still needs a wired connection to a mobile phone or needs an expensive custom radio in the 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:



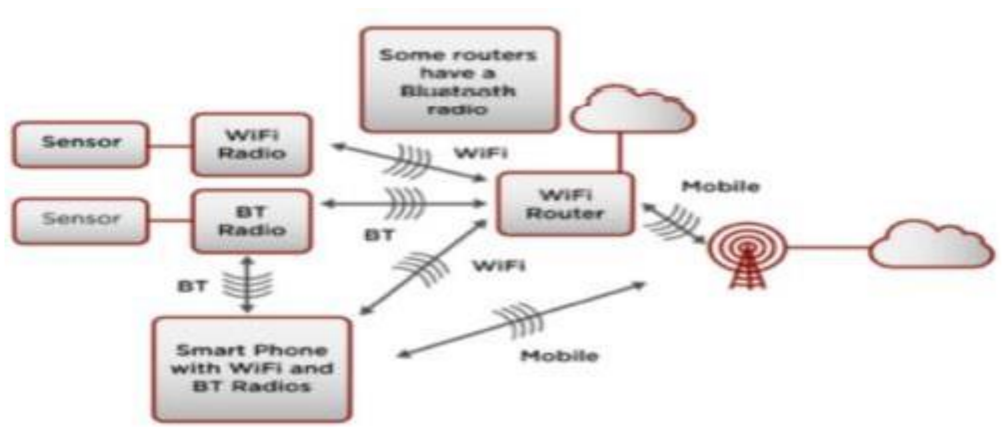
Regulators established several license free radio bands as early as 1947. But these did not attract much interest until mobile phones really caught on in the late 1990s. IEE 802.15.4 standard has frequencies of two bands at 902-928 MHz and 2400-2483 MHz.(There are other standards, such as Zigbee, in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


Sensor
WiFi Radio
WiFi
Some routers have a mobile phone link
WiFi
WiFi Router
Mobile
Smart Phone with WiFi and BT Radios
Mobile

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 WiFi 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 routor. Internet connection is provided later.

Sensor To Mobile Phone To Cloud:


Some routers have a Bluetooth radio
Sensor
WiFi Radio
WiFi
Sensor
BT Radio
BT
WiFi Router
Mobile
WiFi
BT
Smart Phone with WiFi and BT Radios
Mobile

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.From Sensor to Cloud: A Plug and Play Approach EvolvingToday, 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:

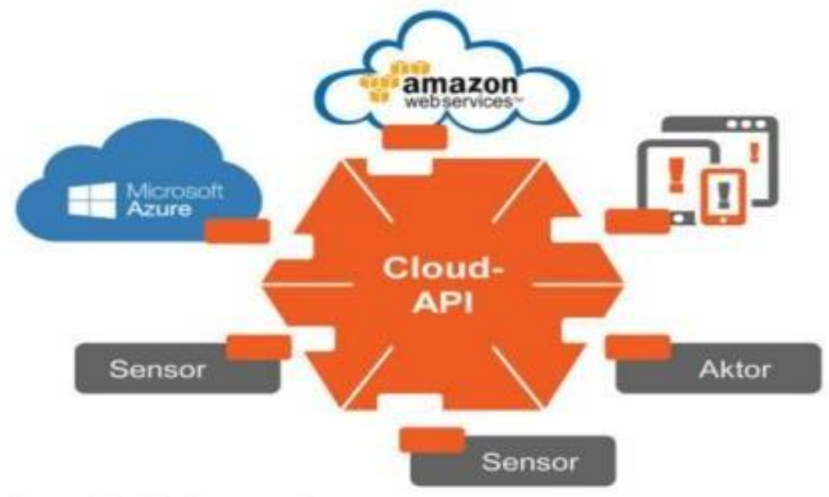


Fig. 1: The cloud API environment

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.Enabled with the new cloud API, the IoT gateway communicates locally with intelligent sensors. Now it is capable of processing and converting the received sensor data. Embedded driver modules (EDMs) interface with hardware and third-party expansion cards, providing the glue logic that translates received data into the semantics of the application-specific IoT gateway logic. This sensor engine, with EDM modules incorporated in its structure, was the first software components which was standardised as a cloud function module. Its critical value is in moving data from local sensors to a generic middleware, independent of protocols.