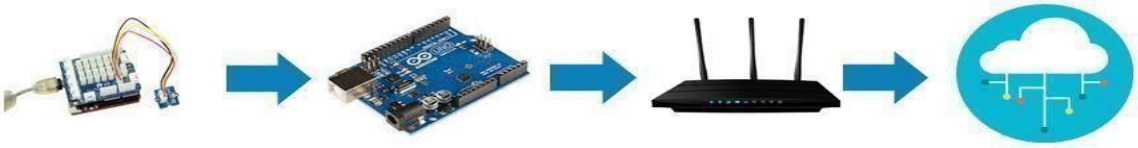
UNIT– 3

1. IOT architecture

State of the art

IoT architecture varies from solution to solution, based on the type of solution which we intend to build. IoT as a technology majorly consists of four main components, over which an architecture is framed.

1) Sensors
2) Devices
3) Gateway
4) Cloud



Stages of IoT Architecture

Stage 1: Sensors/actuators

Sensors collect data from the environmentor objectundermeasurementandturn itintousefuldata. Think ofthe specialized structures in your cellphone that detectthe directionalpullofgravity and the phone's relative position to the "thing" we call the earth and convert it into data that your phone can use to orient the device.

Actuators can also intervene to change the physical conditions that generate the data. An actuator might,forexample,shutoffapowersupply,adjustanairflowvalve,ormovearoboticgripper in an assembly process.

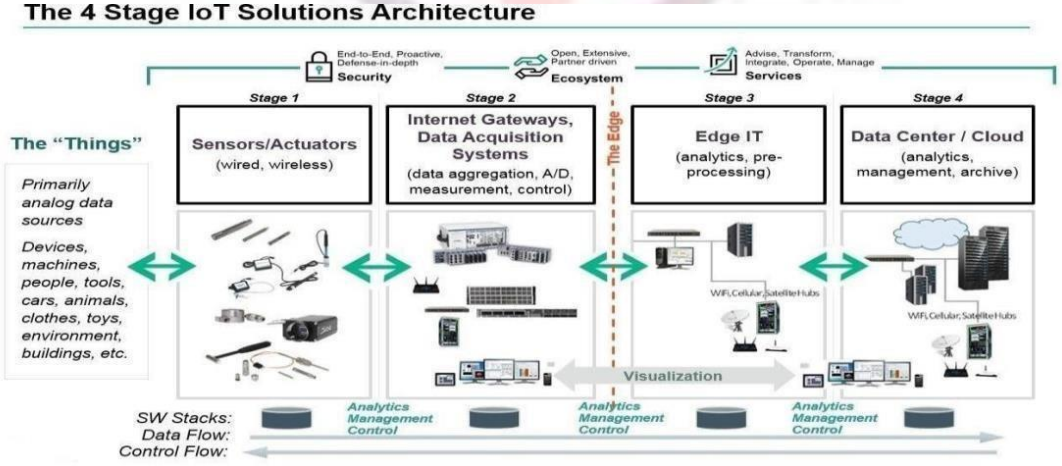
The sensing/actuating stage covers everything from legacy industrial devices to robotic camera systems,waterleveldetectors,airqualitysensors,accelerometers,andheartratemonitors.And thescopeoftheIoT isexpandingrapidly, thanks inparttolow-powerwirelesssensor network technologiesandPoweroverEthernet,whichenabledevicesonawiredLANtooperatewithout the need for an A/C power source.

Stage2:-

The Internet gateway

Thedatafromthesensorsstartsinanalogform.Thatdataneedstobeaggregatedandconverted into digital streams for further processing downstream. Data acquisition systems (DAS) perform these data aggregation and conversion functions. The DAS connects to the sensor network, aggregates outputs, and performs the analog-to-digital conversion. The Internet gateway receives the aggregated and digitized data and routes it over Wi-Fi, wired LANs, or the Internet, to Stage 3 systems for further processing. Stage 2 systems often sit in close proximity to the sensors and actuators.

For example, a pump might contain a half-dozen sensors and actuators that feed data into a data aggregationdevicethatalsodigitizesthedata.Thisdevicemightbephysicallyattachedtothepump. Anadjacent gatewaydevice or server would then process the data and forward it to the Stage 3 or Stage4systems.Intelligentgatewayscanbuildonadditional,basicgatewayfunctionalitybyadding such capabilities as analytics, malware protection, and data management services. These systems enable the analysis of data streams in realtime.



Stage 3:-Edge IT

Once IoT data has been digitized and aggregated, it'sreadyto cross into the realm ofIT. However, the data may require further processing before it enters the data center. This is where edge IT systems,whichperformmoreanalysis,comeintoplay.EdgeITprocessingsystemsmaybelocated inremoteofficesorother edgelocations, butgenerallythesesit inthefacilityor locationwherethe sensors reside closer to the sensors, such as in a wiring closet. Because IoT data can easily eat up network bandwidth and swamp your data center resources, it's best to have systems at the edge capable ofperforming analytics as a wayto lessenthe burdenoncoreIT infrastructure. You'd also face securityconcerns, storage issues, anddelaysprocessingthedata. Withastagedapproach, you can preprocess the data, generate meaningful results, and pass only those on. For example, rather than passing on raw vibration data for the pumps, you could

aggregateandconvertthedata,analyzeit, andsendonlyprojectionsastowhen each device will fail or need service.

Stage4:-

Thedatacenterandcloud

Data that needs more in-depth processing, and where feedback doesn't have to be immediate, gets forwarded to physical data center or cloud-based systems, where more powerful IT systems can analyze, manage, andsecurelystorethedata.Ittakes longer to getresultswhenyouwait untildata reachesStage4,butyoucanexecuteamorein-depthanalysis, aswellascombineyour sensor data with data fromother sources for deeperinsights. Stage 4 processingmaytake placeon-premises, in the cloud, or in a hybrid cloud system, but thetype ofprocessing executed in this stage remains the same, regardless of the platform.

Script.py

```
1. a=5
2. print(a,"isoftype",type(a))
3. a=2.0
4. print(a,"isoftype",type(a))
5. a=1+2j
6. print(a,"iscomplexnumber?",isinstance(1+2j,complex))
```

Integers can be ofany length, it is only limited by the memoryavailable. A floating point number isaccurateupto15decimalplaces. Integerand floatingpointsareseparatedbydecimalpoints. 1is integer, 1.0 is floating point number. Complex numbers are written in the form, x + yj, where x is the real part and y is the imaginary part. Here are some examples.

```
>>> a =1234567890123456789

>>>a

1234567890

123456789

>>> b=0.1234567890123456789

>>>b

0.12345678

901234568

>>>c=1+2j

>>>c(1+2j)
```

Python List

List is an ordered sequence of items. It is one of the most used data type in Python and is very flexible. All the items in a list do not need to be of the same type. Declaring a list is pretty straight forward. Items separated by commas are enclosed within brackets [ ].

```
>>>a=[1,2.2,
    'python']
```

We can use the slicing operator[]to extract an item orarange of items fromalist. Index starts form 0 in Python.

Script.py

```
1. a=[5,10,15,20,25,30,35,40]
2. #a[2] =15
3. print("a[2] =",a[2])
4. #a[0:3]=[5,10, 15]
5. print("a[0:3]=",a[0:3])
6. #a[5:]=[30,35, 40]
7. print("a[5:]=",a[5:])
```

Lists are mutable, meaning;value of elements of a list can be altered.

```
>>> a =[1,2,3]
>>>a[2]=4
```

Python Strings

String is sequence of Unicode characters. We can use single quotes or double quotes to represent strings. Multi-line strings can be denoted using triple quotes, ''' or """.

```
>>>s="Thisisastring"
>>>s='''amultiline
```

Likelistandtuple, slicingoperator[]canbeusedwithstring.Stringsareimmutable.
Script.py

```
a={5,2,3,1,4}
#printing set variable print("a = ",
a)#datatypeofvariableaprint(type(a))
```

We can perform set operations like union, intersection on two sets. Set have unique values. They eliminateduplicates.Since,setareunorderedcollection,indexinghasnomeaning.Hencetheslicing operator [] does not work. It is generally used when we have a huge amount of data. Dictionaries areoptimizedforretrievingdata.Wemustknowthekeytoretrievethevalue.InPython,dictionaries are defined within braces {} with each item being a pair in the form key: value. Key and value can be of any type.

```
>>>d={1:'value','key':2}

>>>type(d)

<class'dict'>
```

We use key to retrieve the respective value. But not the other way around.

Script.py

```
d = {1:'value','key':2}print(type(d))print("d[1] =",d[1]);
print("d['key']= ",d['key']);#Generateserrorprint("d[2]=",d[2]);
```

Python if Statement Flowchart

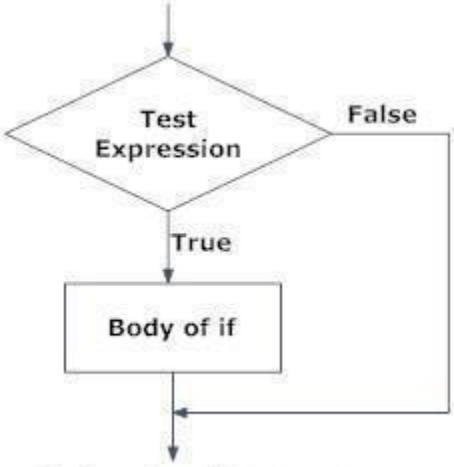


Fig: Operation of if statement

Example: Python if Statement

#If the number is positive, we print an appropriate message num=3 if num > 0:
print(num,"isapositive

number.") print("This isalwaysprinted.")
num = -1 if num > 0: print(num, "is a positive number.") print("This is also always printed.")

Python if..else Flow chart

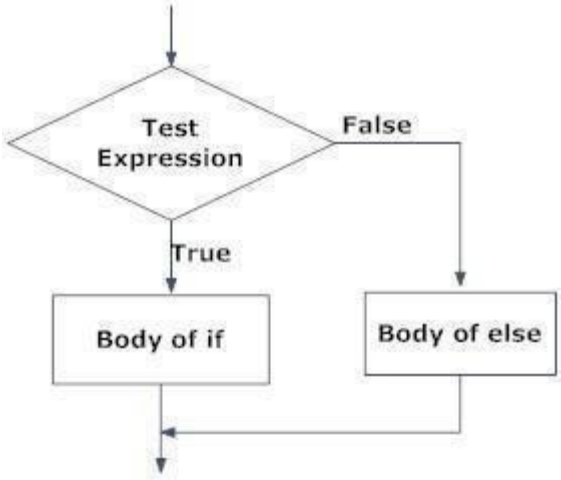


Fig: Operation of if...else statement

Example of if...else

#Program checks if the number is positive or negative#And displays an appropriate message
num = 3
#Trythesetwovariationsaswell.#num=-5 #
num = 0
ifnum>=0:print("PositiveorZero") else:
print("Negativenumber")
Intheaboveexample, when numis equalto3, the test expression is trueand body of if is executed and body of else is skipped.

Ifnumisequalto-5,the test expression is false and body of else is executed and body of if is skipped.

Ifnumisequalto0,thetest expressionistrueandbodyofif isexecutedandbodyof elseisskipped. Python if...elif...else Statement Syntax if test expression:
Bodyofif
eliftest expression:

Python File Methods Files

File is a named location on disk to store related information. It is used to permanently store data in a non-volatile memory (e.g. hard disk). Since, randomacces
s memory (RAM) is volatile which loses its data when computer isturnedoff, weusefilesfor futureuseofthedata.Whenwewanttoreadfromor writetoa filewe need to open it first. When we are done, it needs to be closed, so that resources that are tied with the file are freed. Hence, inPython, a file operation takes place inthe followingorder.

Open a file
Read or write(perform operation)
Close the file

How to open a file?

Python has a built-in function open()to open a file. This function returns a file object, also called a handle, as it is used to read or modify the file accordingly.

```
>>>f=open("test.txt")#open file incurrent directory
>>>f=open("C:/Python33/README.txt")#specifying full path
```

We can specify the mode while opening a file. In mode, we specify whether we want to read 'r', write 'w' or append 'a' to the file. We also specify if we want to open the file in text mode or binary mode. The default is reading in text mode. In this mode, we get strings when reading from the file. On the other hand, binary mode returns bytes and this is the mode to be used when dealing with non-text files like image or exefiles.

Python File Modes

Mode	**Description**
'r'	Open afileforreading. (default)
'w'	Openafileforwriting.Createsanew fileif itdoesnotexistortruncatesthefileifit exists.
'x'	Openafileforexclusivecreation.Ifthefilealreadyexists,theoperationfails.
'a'	Open for a pending at the end of the file without truncating it. Creates a new file if it does not exist.
't'	Open in text mode. (default)
'b'	Open in binary mode.
'+'	Open a file for updating(reading and writing)

```
f=open("test.txt")
#equivalent to'r' or'rt'
f=open("test.txt",'w')#write in text mode
f=open("img.bmp",'r+b')#read and write in binary mode
```

Unlikeotherlanguages,thecharacter'a'doesnotimplythenumber97untilitisencodedusingASCII(orother equivalent encodings). Moreover, the default encoding is platform dependent. In windows, it is 'cp1252' but 'utf-8' in Linux. So, we must not also rely on the default encoding or else our code will behave differently in different platforms. Hence, when working with files in text mode, it is highly recommended to specify the encoding type.

```
f=open("test.txt",mode='r',encoding='utf-8')
```

Python Program with Raspberry PI with focus of interfacing external gadgets,controlling output,readinginput from pins.

Light an LED through Python program.

Solution:

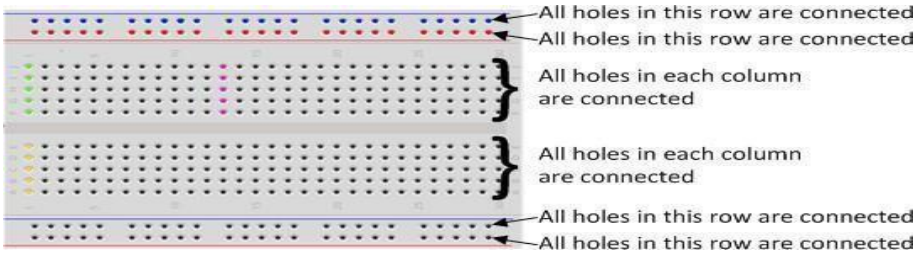
One of the biggest selling points of the RaspberryPiis its GPIO, or General Purpose Input/Out put ports. They are the little pins sticking out of the circuit board and allow you to plug various devices into your Raspberry Pi. With a little programming, you can then control themor detect what they are doing.

InthistutorialIamgoingtoshowyouhowtolightanLED.InadditiontoyourRaspberryPirunningRaspbian, what you will need is:

An LED
A330ohm resistor

The Bread board:

he breadboard is a wayof connecting electronic components to each other without having to solder them together. They are often used to test a circuit design before creating a Printed Circuit Board (PCB).



The LED:

When you pickup the LED,you will notice that one leg is longer than the other. The longer leg(known as the 'anode'), is always connected to the positive supply of the circuit. The shorter leg(known as the'cathode') is connected to the negative side of the power supply, known as 'ground'.

LEDstandsforLightEmittingDiode,andglowswhenelectricityispassedthroughit.

TheResistor:

You must ALWAYS useresistors toconnect LEDs uptotheGPIO pins of theRaspberryPi.The Raspberry Pi can only supply a small current (about 60mA). The LEDs will want to draw more, and if allowedtothey willburn out theRaspberryPi. Thereforeputtingtheresistors inthecircuit will ensure that only this small current will flow and theRaspberry Pi will not be damaged.

Jumper Wires:

Jumper wires are used on breadboards to 'jump' from one connection to another. The ones you will beusing in this circuit have different connectors on each end. The end with the 'pin' will goin to the Breadboard. The end with the piece of plastic with a hole in it will go onto the Raspberry Pi's GPIO pins.

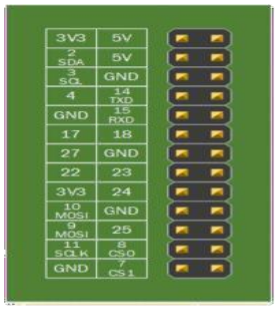
TheRaspberryPi'sGPIOPins:

GPIOstandsfor**GeneralPurposeInputOutput**.Itis awaytheRaspberryPicancontrol and monitor the outside world by being connected to electronic circuits. The Raspberry Pi is able to control LEDs, turning them on or off, or motors, or many other things. It is also able to detect whether a switch has been pressed, or temperature, or light. In the CamJam EduKit you will learn to control LEDs and a buzzer, and detect when a button has been pressed. The diagram below left shows the pin layout for a Raspberry Pi Models Aand B (Rev 2 -theoriginal Rev 1 Pi is slightly different), looking at theRaspberry Piwith thepins in the top right corner. The new 40 pin Raspberry Pi's shares exactly the same layout of pins for the top 13 rows of GPIOpins.

BuildingtheCircuit:

Thecircuitconsistsofapowersupply(theRaspberryPi),anLEDthatlightswhenthepowerisapplied,and

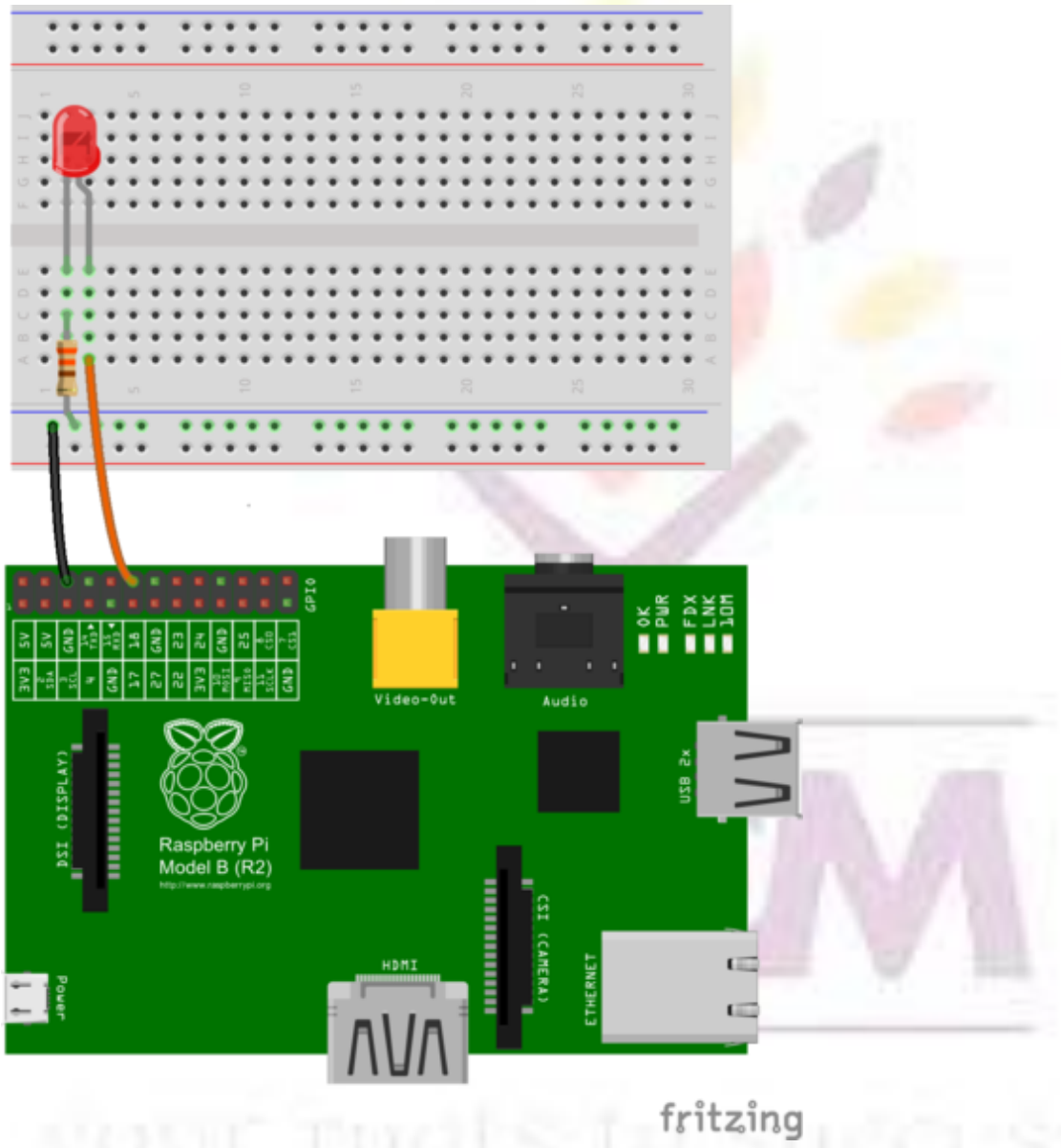
Models A & B

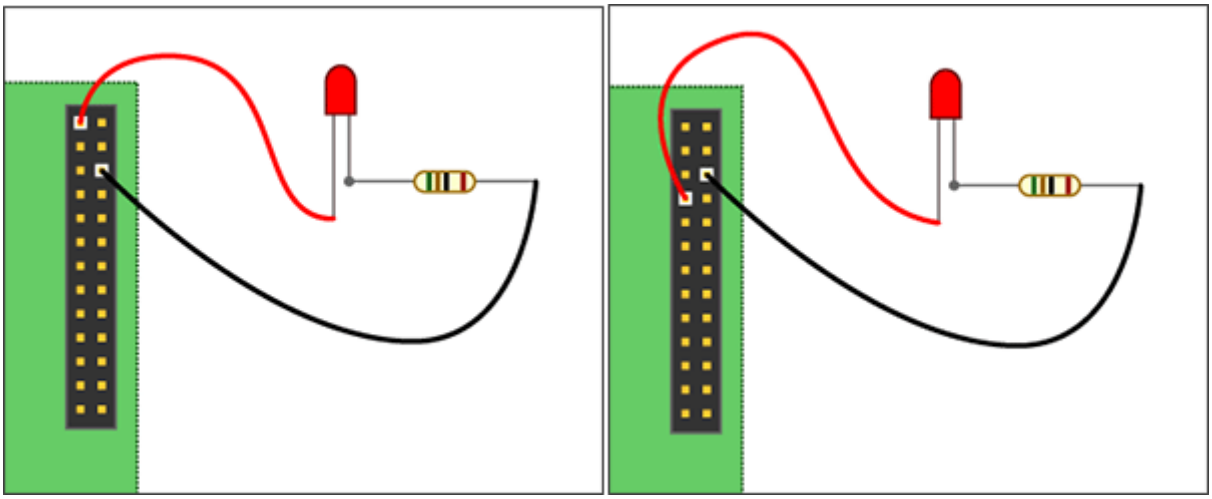


Models A+, B+ & Pi2

aresistortolimitthecurrentthatcanflowthroughthecircuit.

You will be using one of the ‘ground’ (GND) pins to act like the ‘negative’ or 0 volt ends of a battery. The ‘positive’ end of the battery will be provided by a GPIO pin. Here we will be using pin 18. When they are ‘takenhigh’,whichmeansitoutputs3.3volts,theLEDwilllight.Nowtakealookatthecircuitdiagrambelow. You should turn your Raspberry Pioff for the next bit, just in case you accidentally short something out.





print"LEDon"GPIO.output(18,GPIO.HIGH)ti
me.sleep(1)print "LED
off"GPIO.output(18,GPIO.LOW)
Onceyou havetypedallthecodeandcheckedit, saveandexitthetext editor with“Ctrl+x”then“y”then “enter”.

RunningtheCode:

Torunthiscodetype:
sudo pythonLED.py
YouwillseetheLEDturnonforasecondandthenturnoff.

Ifyourcodedoesnotrunandanerrorisreported,editthecodeagainusingnanoLED.py.