INTERNET OF THINGS

UNIT I:

Introduction to Internet of Things: Introduction – Physical Design of IoT – Logical Design of IoT – IoT Enabling Technologies – IoT & Deployment Templates.

Domain Specific IoTs : Introduction – Home Automation – Cities – Environment – Energy – Retail – Logistics – Agriculture – Industry – Health & Life style.

UNIT II:

IoT and M2M: Introduction – M2M – Difference between IoT and M2M – SDN and NFV for IoT.

IoT System Management with NETCONF-YANG : Need for IoT System Management – Simple Network Management Protocol (SNMP) – Network Operator Requirements – NETCONF-YANG – IoT Systems Management with NETCONF-YANG.

UNIT III:

IoT Platforms Design Methodology: Introduction – IoT Design Methodology – Case Study on IoT System for Weather Monitoring – Motivation for using Python.

IoT Systems – Logical Design using Python: Introduction – Installing Python – Python Data types & Data Structures – Control Flow – Functions – Modules – Packages – File Handling – Date/Time Operations – Classes – Python packages of Interest for IoT.

UNIT IV:

IoT Physical Devices & Endpoints: What is an IoT Devices – Exemplary Device: Raspberry Pi – About the Board – Linux on Raspberry Pi - Raspberry Pi Interfaces – Programming Raspberry Pi with Python – Other IoT devices.

IoT Physical Servers & Cloud Offerings: Introduction to Cloud Storage Models & Communication APIs – WAMP – AutoBahn for IoT – Xively Cloud for IoT – Python Web application Framework-Django – Designing a REST ful Web API – Amazon Web Services for IoT – SkynetloT messaging platform.

UNIT V:

Case Studies Illustrating IoT Design: Introduction – Home Automation – Cities – Environment – Agriculture – Productivity applications.

Data Analytics for IoT: Introduction – Apache Hadoop – Using Hadoop MapReduce for Batch Data Analysis – Apache Oozier – Apache Spark – Apache Storm – Using Apache Storm for Real-time Data Analysis.

Text Book:

1. Internet of Things, Arshdeep Bahga, Vijay Medisetti, Universities Press(INDIA) Private Ltd., 2015.

UNIT – I

1. INTRODUCTION TO IOT

1. IoT definition & Characteristics of IoT
2. Physical Design of IoT
3. Logical Design of IoT
4. IoT Enabling Technologies
5. IoT Levels & Deployment Templates

1.1 INTRODUCTION

The **Internet of Things** (**IoT**) is the network of physical devices, vehicles, home appliances, and other items embedded with electronics, software, sensors, actuators, and connectivity which enables these things to connect and exchange data, creating opportunities for more direct integration of the physical world into computer-based systems, resulting in efficiency improvements, economic benefits, and reduced human exertions.

- The focus on IoT is the configuration,control and networking via internet of devices.
- These include devices such as thermostats, utility meters, a Bluetooth-connected headset and sensors.

Data:

Raw and unprocessed data.

Information:

It is inferred from data by filtering,processing,categorizing,condensing and contextualizing data.

Knowledge:

It is inferred from information by organizing and structuring information.

Applications of IoT:

- Home ,Cities
- Environment
- Energy
- Retail

- Logistics
- Agriculture
- Industry
- Health & Lifestyle

IoT DEFINITION

A dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual "things" have identities, physical attributes, and virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network, often communicate data associated with users and their environments.

Characteristics of IoT

• Dynamic & Self-Adapting
• Self-Configuring
• Interoperable Communication Protocols
• Unique Identity
• Integrated into Information Network

1.2 PHYSICAL DESIGN OF IoT

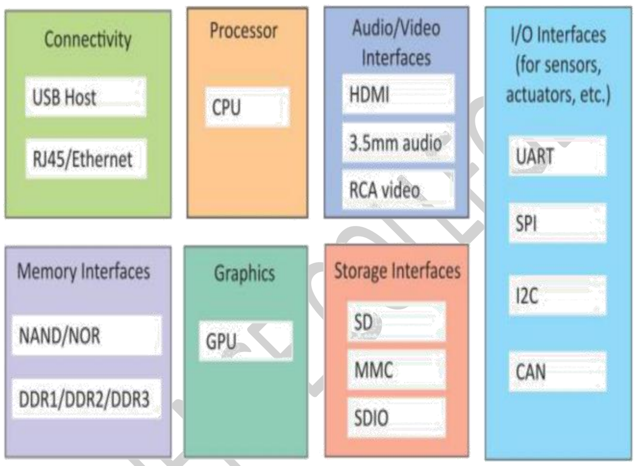
1.2.1 Things in IoT

• The "Things" in IoT usually refers to IoT devices which have unique identities and can perform remote sensing, actuating and monitoring capabilities.
• IoT devices can:
• Exchange data with other connected devices and applications (directly or indirectly), or
• Collect data from other devices and process the data locally or
• Send the data to centralized servers or cloud-based application back-ends for processing the data, or
• Perform some tasks locally and other tasks within the IoT infrastructure, based on temporal and space constraints.

Generic block diagram of an IoT Device

An IoT device may consist of several interfaces for connections to other devices, both wired and wireless.

- I/O interfaces for sensors
- Interfaces for Internet connectivity
- Memory and storage interfaces
- Audio/video interfaces.



1.2.2 IOT PROTOCOLS

Link Layer

- 802.3 – Ethernet
- 802.11 – WiFi
- 802.16 – WiMax
- 802.15.4 – LR-WPAN
- 2G/3G/4G

Network/Internet Layer

- IPv4
- IPv6
- 6LoWPAN

Transport Layer

- TCP
- UDP

Application Layer

- • HTTP
- • CoAP
- • WebSocket
- • MQTT
- • XMPP

• DDS

• AMQP

1.3.LOGICAL DESIGN OF IOT

- Logical design of an IoT system refers to an abstract representation of the entities and processes without going into the low-level specifics of the implementation.

1.3.1 IoT Functional Blocks

- An IoT system comprises of a number of functional blocks that provide the system the capabilities for identification, sensing, actuation, communication, and management.
  - Device
  - Communication
  - Services
  - Management
  - Security
  - Application

1.3.2 IoT communication model

- Request-Response communication model
- Publish-Subscribe communication model
- Push-Pull communication model
- Exclusive Pair communication model

1.3.3 IoT Communication APIs

- REST-based Communication APIs
- WebSocket-based Communication APIs
- Exclusive Pair communication model

1.4 IOT ENABLING TECHNOLOGIES

1.4.1 Wireless Sensor Networks

1.4.2 Cloud Computing

1.4.3 Bigdata Analytics

1.4.4 Communication Protocols

1.4.5 Embedded Systems

1.5 IOT LEVELS & DEPLOYMENT TEMPLATES

An IoT system comprises of the following components:

- **Device**: An IoT device allows identification, remote sensing, actuating and remote monitoring capabilities. You learned about various examples of IoT devices in section

- **Resource**: Resources are software components on the IoT device for accessing, processing, and storing sensor information, or controlling actuators connected to the device. Resources also include the software components that enable network access for the device.

- **Controller Service**: Controller service is a native service that runs on the device and interacts with the web services. Controller service sends data from the device to the web service and receives commands from the application (via web services) for controlling the device.
- **Database**: Database can be either local or in the cloud and stores the data generated by the IoT device.

- **Web Service**: Web services serve as a link between the IoT device, application, database and analysis components. Web service can be either implemented using HTTP and REST principles (REST service) or using WebSocket protocol (WebSocket service).

- **Analysis Component**: The Analysis Component is responsible for analyzing the IoT data and generate results in a form which are easy for the user to understand.

- **Application**: IoT applications provide an interface that the users can use to control and monitor various aspects of the IoT system. Applications also allow users to view the system status and view the processed data.
    - IoT Level-1
    - IoT Level-1
    - IoT Level-1
    - IoT Level-1
    - IoT Level-1
    - IoT Level-1

DOMAIN SPECIFIC IoTs

1.6 INTRODUCTION

The IoT Applications is a wide range of domains including,

- Home
- Cities
- Environment
- Energy Systems
- Retail
- Logistics
- Industry
- Agriculture
- Health & lifestyle

1.7 HOME AUTOMATION

- Smart Lighting
- Smart Appliances
- Intrusion Detection
- Smoke/Gas Detectors

1.8 CITIES

- Smart Parking
- Smart Lighting
- Smart Roads
- Structural health Monitoring

1.9 ENVIRONMENTS

- Weather Monitoring
- Air pollution Monitoring
- Noise Pollution Monitoring
- Forest Fire Detection
- River Flood Detection

1.10 ENERGY

- Smart Grid
- Renewable Energy Systems
- Prognostics

1.11 RETAIL

- Inventory Management
- Smart payments
- Smart Vending Machines

1.12 LOGISTICS

- Route Generation & Scheduling
- Fleet Tracking
- Shipment Monitoring
- Remote Vehicle Diagnostics

1.13 AGRICULTURE

- Smart Irrigation
- Green House Control

1.14 INDUSTRY

- Machine Diagnosis & Prognosis
- Indoor Air Quality Monitoring

1.15 HEALTH & LIFESTYLE

- Health & Fitness Monitoring
- Wearable Electronics

UNIT - II

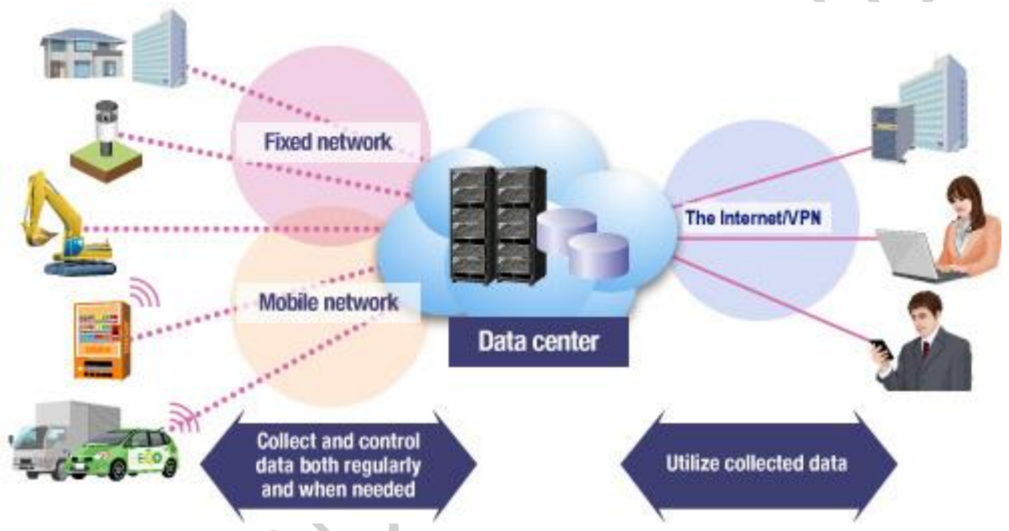
2.1 MACHINE-TO-MACHINE (M2M)

• Differences and Similarities between M2M and IoT

• SDN and NFV for IoT

Machine-to-Machine (M2M) refers to networking of machines (or devices) for the purpose of remote monitoring and control and data exchange.

An M2M area network comprises of machines (or M2M nodes) which have embedded hardware modules for sensing, actuation and communication.



• Various communication protocols can be used for M2M local area networks such as ZigBee, Bluetooh, ModBus, M-Bus, Wirless M-Bus, Power Line Communication (PLC), 6LoWPAN, IEEE 802.15.4, etc.

• The communication network provides connectivity to remote M2M area networks.

• The communication network can use either wired or wireless networks (IPbased).

• While the M2M area networks use either proprietary or non-IP based communication protocols, the communication network uses IP-based networks.

M2M gateway

• Since non-IP based protocols are used within M2M area networks, the M2M nodes within one network cannot communicate with nodes in an external network.

• To enable the communication between remote M2M area networks, M2M gateways are used.

2.2 DIFFERENCE BETWEEN IOT AND M2M

Communication Protocols

• M2M and IoT can differ in how the communication between the machines or devices happens.

• M2M uses either proprietary or non-IP based communication protocols for communication within the M2M area networks.

• Machines in M2M vs Things in IoT

• The "Things" in IoT refers to physical objects that have unique identifiers and can sense and communicate with their external environment (and user applications) or their internal physical states.

• M2M systems, in contrast to IoT, typically have homogeneous machine types within an M2M area network.

Hardware vs Software Emphasis

• While the emphasis of M2M is more on hardware with embedded modules, the emphasis of IoT is more on software.

• Data Collection & Analysis

• M2M data is collected in point solutions and often in on-premises storage infrastructure.

• In contrast to M2M, the data in IoT is collected in the cloud (can be public, private or hybrid cloud).

• Applications

• M2M data is collected in point solutions and can be accessed by on-premises applications such as diagnosis applications, service management applications, and on premisis enterprise applications.

• IoT data is collected in the cloud and can be accessed by cloud applications such as analytics applications, enterprise applications, remote diagnosis and management applications, etc.

2.3 SDN & NFV

2.3.1 SDN(Software-Defined Networking)

(SDN) is a networking architecture that separates the control plane from the data plane and centralizes the network controller.

• Software-based SDN controllers maintain a unified view of the network and make configuration, management and provisioning simpler.

• The underlying infrastructure in SDN uses simple packet forwarding hardware as opposed to specialized hardware in conventional networks.

Key elements of SDN

• Centralized Network Controller

• With decoupled control and data planes and centralized network controller, the network administrators can rapidly configure the network.

• Programmable Open APIs

• SDN architecture supports programmable open APIs for interface between the SDN application and control layers (Northbound interface).

• Standard Communication Interface (OpenFlow)

• SDN architecture uses a standard communication interface between the control and infrastructure layers (Southbound interface).

• OpenFlow, which is defined by the Open Networking Foundation (ONF) is the broadly accepted SDN protocol for the Southbound interface.

2.3.2 NFV

•Network Function Virtualization (NFV) is a technology that leverages virtualization to consolidate the heterogeneous network devices onto industry standard high volume servers, switches and storage.

• NFV is complementary to SDN as NFV can provide the infrastructure on which SDN can run.

Key elements of NFV

• Virtualized Network Function (VNF):

• VNF is a software implementation of a network function which is capable of running over the NFV Infrastructure (NFVI).

• NFV Infrastructure (NFVI):

• NFVI includes compute, network and storage resources that are virtualized.

• NFV Management and Orchestration:

• NFV Management and Orchestration focuses on all virtualization-specific management tasks and covers the orchestration and life-cycle management of physical and/or software resources that support the infrastructure virtualization, and the life-cycle management of VNFs.

NFV Use Case

• NFV can be used to virtualize the Home Gateway. The NFV infrastructure in the cloud hosts a virtualized Home Gateway. The virtualized gateway provides private IP addresses to the devices in the home. The virtualized gateway also connects to network services such as VoIP and IPTV.

2.4 IOT SYSTEM MANAGEMENT WITH NETCONF-YANG

Need for IoT Systems Management,

- Automating Configuration
- Monitoring Operational & Statistical Data
- Improved Reliability
- System Wide Configurations
- Multiple System Configurations
- Retrieving & Reusing Configurations

2.5 SIMPLE NETWORK MANAGEMENT PROTOCOL (SNMP)

• SNMP is a well-known and widely used network management protocol that allows monitoring and configuring network devices such as routers, switches, servers, printers, etc.

• SNMP component include

• Network Management Station (NMS)

• Managed Device

• Management Information Base (MIB)

• SNMP Agent that runs on the device Limitations of SNMP

• SNMP is stateless in nature and each SNMP request contains all the information to process the request. The application needs to be intelligent to manage the device.

• SNMP is a connectionless protocol which uses UDP as the transport protocol, making it unreliable as there was no support for acknowledgement of requests.

• MIBs often lack writable objects without which device configuration is not possible using SNMP.

• It is difficult to differentiate between configuration and state data in MIBs.

• Retrieving the current configuration from a device can be difficult with SNMP.

• Earlier versions of SNMP did not have strong security features.

2.6 NETWORK OPERATOR REQUIREMENTS

• Ease of use

• Distinction between configuration and state data

• Fetch configuration and state data separately

• Configuration of the network as a whole

• Configuration transactions across devices

• Configuration deltas

• Dump and restore configurations

• Configuration validation

• Configuration database schemas

• Comparing configurations

• Role-based access control

• Consistency of access control lists:

• Multiple configuration sets

• Support for both data-oriented and taskoriented access control.

2.7 NETCONF

Network Configuration Protocol (NETCONF) is a session-based network management protocol.NETCONF allows retrieving state or configuration data and manipulating configuration data on network devices.

• NETCONF works on SSH transport protocol.

• Transport layer provides end-to-end connectivity and ensure reliable delivery of messages.

• NETCONF uses XML-encoded Remote Procedure Calls (RPCs) for framing request and response messages.

The RPC layer provides mechanism for encoding of RPC calls and notifications. NETCONF provides various operations to retrieve and edit configuration data from network devices.

• The Content Layer consists of configuration and state data which is XML-encoded.

• The schema of the configuration and state data is defined in a data modeling language called YANG.

• NETCONF provides a clear separation of the configuration and state data.

The configuration data resides within a NETCONF configuration datastore on the server.YANG.

2.8 YANG

- YANG is a data modeling language used to model configuration and state data manipulated by the NETCONF protocol
- YANG modules contain the definitions of the configuration data, state data, RPC calls that can be issued and the format of the notifications.
- YANG modules defines the data exchanged between the NETCONF client and server.
- A module comprises of a number of 'leaf' nodes which are organized into a hierarchical tree structure.
- The 'leaf' nodes are specified using the 'leaf' or 'leaf-list' constructs.
- Leaf nodes are organized using 'container' or 'list' constructs.
- A YANG module can import definitions from other modules.
- Constraints can be defined on the data nodes, e.g. allowed values.
- YANG can model both configuration data and state data using the 'config' statement.

2.9 IOT SYSTEMS MANAGEMENT WITH NETCONF-YANG

- ➢ Management System
- ➢ Management API
- ➢ Transaction Manager
- ➢ Rollback Manager
- ➢ Data Model Manager
- ➢ Configuration Validator
- ➢ Configuration Database
- ➢ Configuration API
- ➢ Data Provider API

UNIT- III

IOT PLATFORM DESIGN METHODOLOGY

3.1 INTRODUCTION

IoT system comprise of multiple components and deployment tiers. IoT system levels described in 6 levels. Each level is suited for different applications and has different component.

3.2 IOT DESIGN METHODOLOGY – STEPS

Step 1: Purpose & Requirements Specification
Step 2: Process Specification
Step 3: Domain Model Specification
Step 4: Information Model Specification
Step 5: Service Specifications
Step 6: IoT Level Specification
Step 7: Functional View Specification
Step 8: Operational View Specification
Step 9: Device & Component Integration
Step 10: Application Development

3.3WHEATHER MONITORING

- Implementation: RESTful Web Services
- Implementation: Controller Native Service
- Implementation: Application
- Finally - Integrate the System
    - Setup the device
    - Deploy and run the REST and Native services
    - Deploy and run the Application
    - Setup the database

3.4 MOTIVATION FOR USING PYTHON

- Easy-to-learn read and maintain
- Python is a minimalistic language with relatively few keywords, uses English keywords and has fewer syntactical constructions as compared to other languages. Reading Python programs feels like English with pseudo-code like constructs. Python is easy to learn yet an extremely powerful language for a wide range of applications.
- Object and Procedure Oriented
- Python supports both procedure-oriented programming and object-oriented programming. Procedure oriented paradigm allows programs to be written around procedures or functions that allow reuse of code. Procedure oriented paradigm allows programs to be written around objects that include both data and functionality.
- Extendable
- Python is an extendable language and allows integration of low-level modules written in languages such as C/C++. This is useful when you want to speed up a critical portion of a program.
- Scalable
- Due to the minimalistic nature of Python, it provides a manageable structure for large programs.
- Portable
- Since Python is an interpreted language, programmers do not have to worry about compilation, linking and loading of programs. Python programs can be directly executed from source
- Broad Library Support
- Python has a broad library support and works on various platforms such as Windows, Linux, Mac, etc.

3.5 IOT SYSTEMS –LOGICAL DESIGN USING PYTHON

- Introduction to Python
- Installing Python
- Python Data Types & Data Structures
- Control Flow
- Functions
- Modules
- Packages

• File Input/Output

• Date/Time Operations

• Classes

3.6 PYTHON - INTRODUCTION

• Python is a general-purpose high level programming language and suitable for providing a solid foundation to the reader in the area of cloud computing.

• The main characteristics of Python are,

- Multi-paradigm programming language
- Python supports more than one programming paradigms including object-oriented programming and structured programming
- Interpreted Language
- Python is an interpreted language and does not require an explicit compilation step.
- The Python interpreter executes the program source code directly, statement by statement, as a processor or scripting engine does.
- Interactive Language
- Python provides an interactive mode in which the user can submit commands at the Python prompt and interact with the interpreter directly.

3.7 INSTALLING PYTHON – SETUP

Windows

• Python binaries for Windows can be downloaded from http://www.python.org/getit .

• For the examples and exercise in this book, you would require Python 2.7 which can be directly downloaded from: http://www.python.org/ftp/python/2.7.5/python-2.7.5.msi

• Once the python binary is installed you can run the python shell at the command prompt using

```
> python
```

• **Linux**

```
#Install Dependencies
sudo apt-get install build-essential
sudo apt-get install libreadline-gplv2-dev libncursesw5-dev libssl-dev libsqlite3-dev tk-dev libgdbm-dev libc6-dev libbz2-dev
#Download Python
wget http://python.org/ftp/python/2.7.5/Python-2.7.5.tgz
```

```
tar -xvf Python-2.7.5.tgz
cd Python-2.7.5
#Install Python
./configure
make
sudo make install
```

3.8 PYTHON DATA TYPES & DATA STRUCTURES

3.8.1 Numbers

Number data type is used to store numeric values. Numbers are immutable data types, therefore changing the value of a number data type results in a newly allocated object.

```
#Integer
>>>a=5
>>>type(a)
<type ’int’>

#Floating Point
>>>b=2.5
>>>type(b)
<type ’float’>

#Addition
>>>c=a+b
>>>c
7.5
>>>type(c)
<type ’float’>
```

3.8.2 Strings

A string is simply a list of characters in order. There are no limits to the number of characters you can have in a string.

```
#Create string
>>>s="Hello World!"
```

```
>>>type(s)
<type ’str’>
```

#String concatenation

```
>>>t="This is sample program."
>>>r = s+t
>>>r
’Hello World!This is sample program.’
```

#Get length of string

```
>>>len(s)
12
```

3.8.3 Lists

List a compound data type used to group together other values. List items need not all have the same type. A list contains items separated by commas and enclosed within square brackets.

#Create List

#Removing an item from a list

```
>>>fruits=[’apple’,’orange’,’banana’,’mango’]
>>>fruits.remove(’mango’)
>>>type(fruits)
>>>fruits
<type ’list’> [’apple’, ’orange’, ’banana’, ’pear’]
```

#Get Length of List

#Inserting an item to a list

```
>>>len(fruits)
>>>fruits.insert(1,’mango’)
4
>>>fruits
```

#Mixed data types in a list

```
>>>mixed=[’data’,5,100.1,8287398L]
>>>type(mixed)
<type ’list’>
```

```
>>>type(mixed[0])
<type 'str'>
>>>type(mixed[1])
<type 'int'>
>>>type(mixed[2])
<type 'float'>
>>>type(mixed[3])
<type 'long'>
```

3.8.4 Tuples

A tuple is a sequence data type that is similar to the list. A tuple consists of a number of values separated by commas and enclosed within parentheses. Unlike lists, the elements of tuples cannot be changed, so tuples can be thought of as read-only lists.

```
#Create a Tuple
>>>fruits=("apple","mango","banana","pineapple")
>>>fruits
('apple', 'mango', 'banana', 'pineapple')
>>>type(fruits)
<type 'tuple'>
```

3.8.5 Dictionaries

Dictionary is a mapping data type or a kind of hash table that maps keys to values. Keys in a dictionary can be of any data type, though numbers and strings are commonly used for keys. Values in a dictionary can be any data type or object.

```
#Create a dictionary
>>>student={'name':'Mary','id':'8776','major':'CS'}
>>>student
{'major': 'CS', 'name': 'Mary', 'id': '8776'}
>>>type(student)
<type 'dict'>
#Get length of a dictionary
>>>len(student)                    3
```

3.9 CONTROL FLOW

3.9.1 if statement

The if statement in Python is similar to the if statement in other languages.

```
>>>a = 25**5
>>>if a>10000:
print "More"
else:
print "Less"
More
```

3.9.2 for statement

- The for statement in Python iterates over items of any sequence (list, string, etc.) in the order in which they appear in the sequence.
- This behavior is different from the for statement in other languages such as C in which an initialization, incrementing and stopping criteria are provided.

```
#Looping over characters in a string
helloString = "Hello World"
for c in helloString:
print c
```

3.9.3 while statement

- The while statement in Python executes the statements within the while loop as long as the while condition is true.

3.9.4 range statement

- The range statement in Python generates a list of numbers in arithmetic progression.

```
#Generate a list of numbers from 0 – 9
>>>range (10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

3.9.5 pass statement

- The pass statement in Python is a null operation.
- The pass statement is used when a statement is required syntactically but you do not want any command or code to execute.

```
>fruits=['apple','orange','banana','mango']
>for item in fruits:
if item == "banana":
pass
else:
print item
apple
orange

mango
```

3.10 FUNCTIONS

- A function is a block of code that takes information in (in the form of parameters), does some computation, and returns a new piece of information based on the parameter information.
- A function in Python is a block of code that begins with the keyword def followed by the function name and parentheses. The function parameters are enclosed within the parenthesis.
- The code block within a function begins after a colon that comes after the parenthesis enclosing the parameters.
- The first statement of the function body can optionally be a documentation string or doc string.

3.11 MODULES

- Python allows organizing the program code into different modules which improves the code readability and management.
- A module is a Python file that defines some functionality in the form of functions or classes.
- Modules can be imported using the import keyword.
- Modules to be imported must be present in the search path.

3.12 PACKAGES

- Python package is hierarchical file structure that consists of modules and subpackages.
- Packages allow better organization of modules related to a single application environment.

3.13 FILE HANDLING

- Python allows reading and writing to files using the file object.
- The open(filename, mode) function is used to get a file object.
- The mode can be read (r), write (w), append (a), read and write (r+ or w+), read-binary (rb), write-binary (wb), etc.

# Example of reading a certain number of bytes

```
>>>fp = open('file.txt','r')
>>>fp.read(10)
'Python sup'

>>>fp.close()
```

# Example of getting the current position of read

```
>>>fp = open('file.txt','r')
>>>fp.read(10)
'Python sup'
>>>currentpos = fp.tell
>>>print currentpos
<built-in method tell of file object at 0x0000000002391390>
                        >>>fp.close()
```

3.14 DATE/TIME OPERATIONS

- Python provides several functions for date and time access and conversions.
- The date/time module allows manipulating date and time in several ways.
- The time module in Python provides various time-related functions.

# Examples of manipulating with date

```
>>>from datetime import date
>>>now = date.today()
>>>print "Date: " + now.strftime("%m-%d-%y")
Date: 07-24-13
>>>print "Day of Week: " + now.strftime("%A")
Day of Week: Wednesday
>>>print "Month: " + now.strftime("%B")
Month: July
>>>then = date(2013, 6, 7)
>>>timediff = now - then
>>>timediff.days
47
```

3.15 CLASSES

Python is an Object-Oriented Programming (OOP) language. Python provides all the standard features of Object Oriented Programming such as classes, class variables, class methods, inheritance, function overloading, and operator overloading.

Class

A class is simply a representation of a type of object and user-defined prototype for an object that is composed of three things: a name, attributes, and operations/ methods.

Instance/Object

Object is an instance of the data structure defined by a class.

Inheritance

Inheritance is the process of forming a new class from an existing class or base class.

Function overloading

Function overloading is a form of polymorphism that allows a function to have different meanings, depending on its context.

Operator overloading

Operator overloading is a form of polymorphism that allows assignment of more than one function to a particular operator.

Function overriding

Function overriding allows a child class to provide a specific implementation of a function that is already provided by the base class. Child class implementation of the overridden function has the same name, parameters and return type as the function in the base class.

Class Example

The variable studentCount is a class variable that is shared by all instances of the class Student and is accessed by Student.studentCount.

The variables name, id and grades are instance variables which are specific to each instance of the class.

3.16 CLASS INHERITANCE

In this example Shape is the base class and Circle is the derived class. The class Circle inherits the attributes of the Shape class.

The child class Circle overrides the methods and attributes of the base class (eg. draw() function defined in the base class Shape is overridden in child class Circle).

There is a special method by the name __init__() which is the class constructor. The class constructor initializes a new instance when it is created. The function __del__() is the class destructor.

```
# Examples of class inheritance
class Shape:
def __init__(self):
print "Base class constructor"
self.color = ’Green’
self.lineWeight = 10.0
def draw(self):
print "Draw - to be implemented"
def setColor(self, c):
self.color = c
```

UNIT - IV

IOT PHYSICAL DEVICES & ENDPOINTS

4.1 WHAT IS AN IOT DEVICE

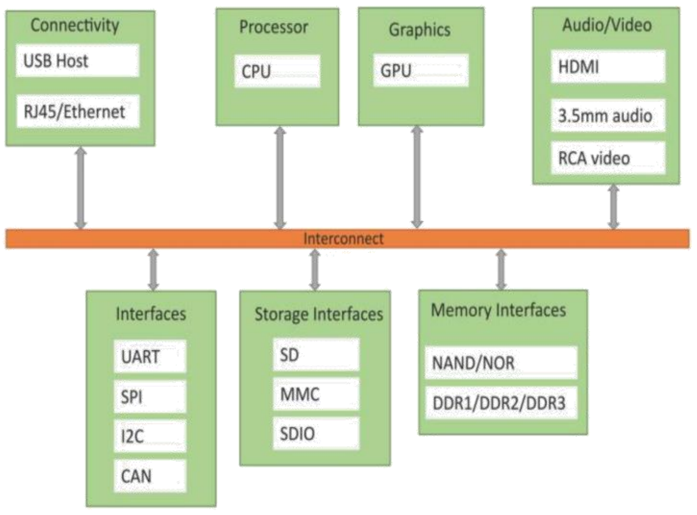
A "Thing" in Internet of Things (IoT) can be any object that has a unique identifier and which can send/receive data (including user data) over a network (e.g., smart phone, smart TV, computer, refrigerator, car, etc.).

IoT devices are connected to the Internet and send information about themselves or about their surroundings (e.g. information sensed by the connected sensors) over a network (to other devices or servers/storage) or allow actuation upon the physical entities/environment around them remotely.

IOT DEVICE EXAMPLES

- A home automation device that allows remotely monitoring the status of appliances and controlling the appliances.
- An industrial machine which sends information abouts its operation and health monitoring data to a server.
- A car which sends information about its location to a cloud-based service.
- A wireless-enabled wearable device that measures data about a person such as the number of steps walked and sends the data to a cloud-based service.

BASIC BUILDING BLOCKS OF AN IOT DEVICE



Sensing

Sensors can be either on-board the IoT device or attached to the device.

Actuation

IoT devices can have various types of actuators attached that allow taking actions upon the physical entities in the vicinity of the device.

Communication

Communication modules are responsible for sending collected data to other devices or cloud- based servers/storage and receiving data from other devices and commands from remote applications.

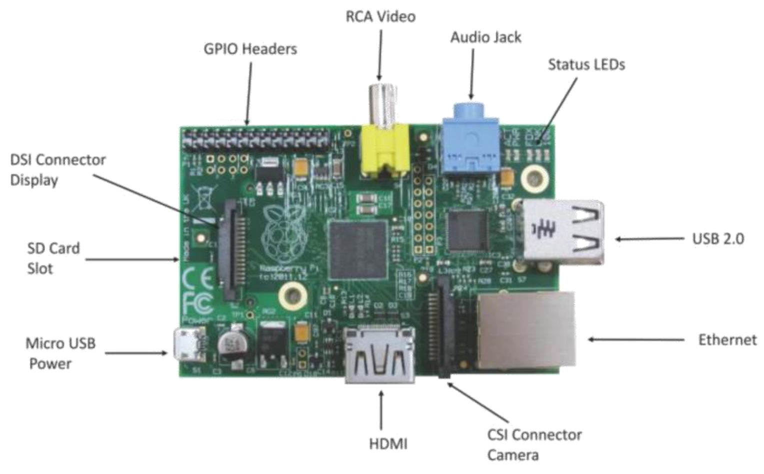
Analysis & Processing

Analysis and processing modules are responsible for making sense of the collected data.

4.2 EXEMPLARY DEVICE: RASPBERRY PI

- Raspberry Pi is a low-cost mini-computer with the physical size of a credit card.
- Raspberry Pi runs various flavors of Linux and can perform almost all tasks that a normal desktop computer can do.
- Raspberry Pi also allows interfacing sensors and actuators through the general purpose I/O pins.
- Since Raspberry Pi runs Linux operating system, it supports Python "out of the box".

4.3 RASPBERRY PI



4.4 LINUX ON RASPBERRY PI

Raspbian

Raspbian Linux is a Debian Wheezy port optimized for Raspberry Pi.

Arch

Arch is an Arch Linux port for AMD devices.

Pidora

Pidora Linux is a Fedora Linux optimized for Raspberry Pi.

RaspBMC

RaspBMC is an XBMC media-center distribution for Raspberry Pi.

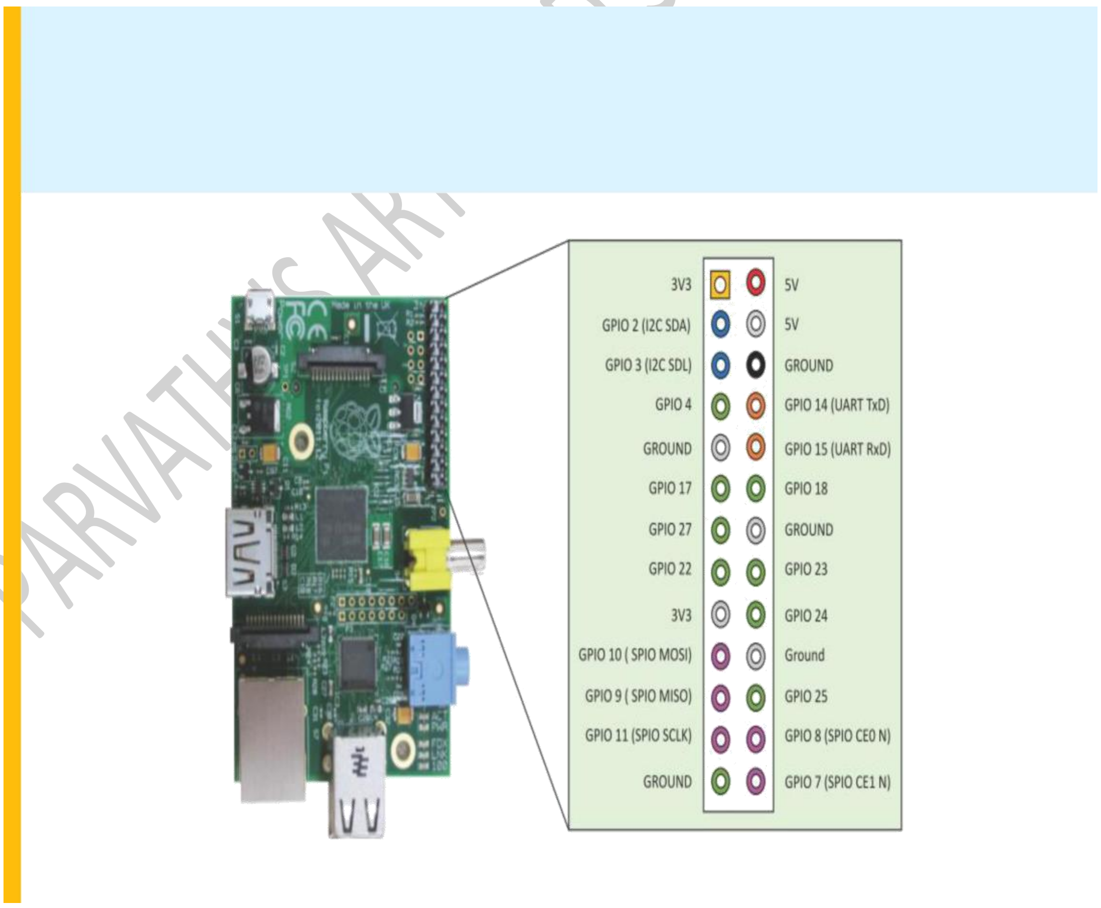
OpenELEC

OpenELEC is a fast and user-friendly XBMC media-center distribution.

RISC OS

RISC OS is a very fast and compact operating system.

Raspberry Pi GPIO



4.5 RASPBERRY PI INTERFACES

Serial

The serial interface on Raspberry Pi has receive (Rx) and transmit (Tx) pins for communication with serial peripherals.

SPI

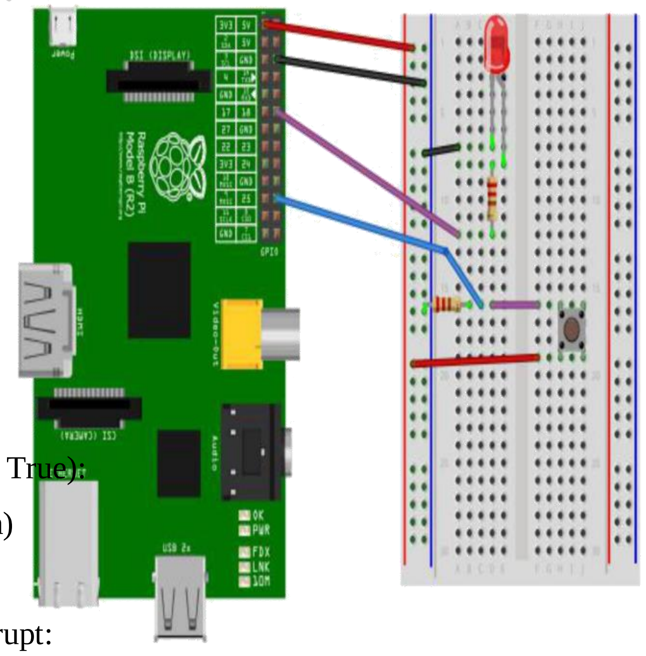
Serial Peripheral Interface (SPI) is a synchronous serial data protocol used for communicating with one or more peripheral devices.

I2C

The I2C interface pins on Raspberry Pi allow you to connect hardware modules. I2C interface allows synchronous data transfer with just two pins - SDA (data line) and SCL (clock line).

Raspberry Pi Example: Interfacing LED and switch with Raspberry Pi

```
from time import sleep
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
#Switch Pin
GPIO.setup(25, GPIO.IN)
#LED Pin
GPIO.setup(18, GPIO.OUT)
state=false
def toggleLED(pin):
        state = not state
        GPIO.output(pin, state)
while True:
        try:
                if (GPIO.input(25) == True):
                        toggleLED(pin)
                sleep(.01)
                except KeyboardInterrupt:
                        exit()
```



4.6 PROGRAMMIN RASBERRY PI WITH PYTHON

4.6.1 Controlling LED with Rasberry Pi

4.6.2 Interfacing an LED and Switch With Rasberry Pi

4.6.3 Interfacing a Light Sensor(LDR) and Switch With Rasberry Pi

4.7 OTHER DEVICES

- pcDuino
- BeagleBone Black
- Cubieboard

IOT PHYSICAL SERVERS & CLOUD OFFERINGS

4.8 INTRODUCTION TO CLOUD STORAGE MODELS & COMMUNICATION APIs

Cloud computing is a transformative computing that involves delivering applications and services over the Internet.

Cloud computing is a model for enabling ubiquitous,convenient, on-demand network access to a shared pool.

4.9 WAMP – AutoBahn for IOT

Web Application Messaging Protocol (WAMP) is a sub-protocol of Websocket which provides publish-subscribe and remote procedure call (RPC) messaging patterns.

1. Transport: Transport is channel that connects two peers.
2. Session: Session is a conversation between two peers that runs over a transport.

3. Client: Clients are peers that can have one or more roles. In publish-subscribe model client can have following roles:
   - Publisher: Publisher publishes events (including payload) to the topic maintained by the Broker.
   - Subscriber: Subscriber subscribes to the topics and receives the events including the payload.

In RPC model client can have following roles:
   - Caller: Caller issues calls to the remote procedures along with call arguments.
   - Callee: Callee executes the procedures to which the calls are issued by the caller and returns the results back to the caller.
4. Router: Routers are peers that perform generic call and event routing. In publish-subscribe model Router has the role of a Broker:
   - Broker: Broker acts as a router and routes messages published to a topic to all subscribers subscribed to the topic.

   In RPC model Router has the role of a Broker:
   - Dealer: Dealer acts a router and routes RPC calls from the Caller to the Callee and routes results from Callee to Caller.
5. Application Code: Application code runs on the Clients (Publisher, Subscriber, Callee or Caller).

4.10 XIVELY CLOUD FOR IoT

- It is a commercial platform-as-a-service that can be used for creating solutions for IoT.
- It comprises of a message bus for real-time message management and routing, data services for time series archiving, directory services that provides a search-able directory of objects.
- It have one or more channels. Each enables bi-directional.

4.11 PYTHON WEB APPLICATION FRAMEWORK – DJANGO

- Django is an open source web application framework for developing web applications in Python.
- A web application framework in general is a collection of solutions, packages and best practices that allows development of web applications and dynamic websites.

- Django is based on the Model-Template-View architecture and provides a separation of the data model from the business rules and the user interface.
- Django provides a unified API to a database backend.
- Thus web applications built with Django can work with different databases without requiring any code changes.
- With this fiexibility in web application design combined with the powerful capabilities of the Python language and the Python ecosystem, Django is best suited for cloud applications.
- Django consists of an object-relational mapper, a web templating system and a regular-expression-based URL dispatcher.

Django Architecture

- Django is Model-Template-View (MTV) framework.

- Model

The model acts as a definition of some stored data and handles the interactions with the database. In a web application, the data can be stored in a relational database, non-relational database, an XML file, etc. A Django model is a Python class that outlines the variables and methods for a particular type of data.

- Template

In a typical Django web application, the template is simply an HTML page with a few extra placeholders. Django's template language can be used to create various forms of text files (XML, email, CSS, Javascript, CSV, etc.)

- View

The view ties the model to the template. The view is where you write the code that actually generates the web pages. View determines what data is to be displayed, retrieves the data from the database and passes the data to the template.

4.12 DESIGNING A RESTFUL WEB API

The REST API allows to create, view, update and delete a collection of resources where each resource represents a sensor data reading from a weather monitoring station.

```
Pip install djangorestframework
Pip install markdown
Pip install django-filter
```

4.13 AMAZON WEB SERVICES FOR IOT

Amazon Ec2 – Python Example

Boto is a Python package that provides interfaces to Amazon Web Services (AWS). In this example, a connection to EC2 service is fi rst established by calling boto.ec2.connect_to_region.

The EC2 region, AWS access key and AWS secret key are passed to this function. After connecting to EC2 , a new instance is launched using the conn.run_instances function. The AMI-ID, instance type, EC2 key handle and security group are passed to this function.

Amazon AutoScaling – Python Example

- ➢ AutoScaling Service
  - A connection to AutoScaling service is first established by
  - calling boto.ec2.autoscale.connect_to_region function.

- ➢ Launch Configuration
  - After connecting to AutoScaling service, a new launch configuration is created by calling
  - conn.create_launch_con f iguration. Launch configuration contains instructions on how to launch new instances including the AMI-ID, instance type, security groups, etc.

- ➢ AutoScaling Group

After creating a launch configuration, it is then associated with a new AutoScaling group. AutoScaling group is created by calling conn.create_auto_scaling_group. The settings for AutoScaling group such as the maximum and minimum number of instances in the group, the launch configuration, availability zones, optional load balancer to use with the group, etc.

Amazon DynamoDB – Python Example

- ➢ In this example, a connection to DynamoDB service is first established by calling boto.dynamodb.connect_to_region.
- ➢ After connecting to DynamoDB service, a schema for the new table is created by calling conn.create_schema.
- ➢ The schema includes the hash key and range key names and types.

- A DynamoDB table is then created by calling conn.create_table function with the table schema, read units and write units as input parameters.

Python for MapReduce

- The example shows inverted index mapper program.
- The map function reads the data from the standard input (stdin) and splits the tab-limited data into document-ID and contents of the document.
- The map function emits key-value pairs where key is each word in the document and value is the document-ID.

Python Packages of Interest

- JSON

JavaScript Object Notation (JSON) is an easy to read and write data-interchange format. JSON is used as an alternative to XML and is is easy for machines to parse and generate. JSON is built on two structures - a collection of name-value pairs (e.g. a Python dictionary) and ordered lists of values (e.g.. a Python list).

- XML

XML (Extensible Markup Language) is a data format for structured document interchange. The Python minidom library provides a minimal implementation of the Document Object Model interface and has an API similar to that in other languages.

- HTTPLib & URLLib

HTTPLib2 and URLLib2 are Python libraries used in network/internet programming

- SMTPLib

Simple Mail Transfer Protocol (SMTP) is a protocol which handles sending email and routing e-mail between mail servers. The Python smtplib module provides an SMTP client session object that can be used to send email.

- NumPy

NumPy is a package for scientific computing in Python. NumPy provides support for large multi-dimensional arrays and matrices

4.14 SCIKIT-LEARN

Scikit-learn is an open source machine learning library for Python that provides implementations of various machine learning algorithms for classification, clustering, regression and dimension reduction problems.

UNIT – V

CASE STUDIES ILLUSTRATING IOT DESIGN

5.1 INTRODUCTION

The case study provides concrete implementations of several of these applications such as homes, cities, environments, retail, logistics, industries, agriculture and health.

5.2 HOME AUTOMATION

5.2.1 Smart Lighting

Smart Lighting is about intelligent and weather adaptive lighting in street, commercial and factory lighting.

The smart lighting market is primarily driven by the growing demand for energy efficient lighting system in order to reduce energy consumption cost. Development of sensor and wireless technology enable improved energy efficiency, process monitoring remote control, and worker safety and satisfaction. IoT solutions enable partially or fully autonomous lighting control based on daylight availability, time, facility activities, and other factors.

5.2.2 Home Intrusion Detection

IoT refers to the infrastructure of connected physical devices which is growing at a rapid rate as huge number of devices and objects are getting associated to the Internet. Home security is a very useful application of IoT and we are using it to create an inexpensive security system for homes as well as industrial use.

IoT or Internet Things refers to the network of connected physical objects that can communicate and exchange data among themselves without the need of any human intervention. It has been formally defined as an “Infrastructure of Information Society”, because IoT allows us to collect information from all kind of mediums such as humans, animals, vehicles, kitchen appliances. Thus any object in the physical world which can be provided with an IP address to enable data transmission over a network can be made part of IoT system by embedding them with electronic hardware such as sensors, software and networking gear. IoT is different than Internet as in a way it transcends Internet connectivity by enabling everyday objects that uses embedded circuits to interact and communicate with each other utilizing the current Internet infrastructure.

5.3 CITIES

5.3.1 Smart Parking

Smart Parking System is very important in aspect to avoid much human efforts, this system is completely based on a raspberry Pi Processor, and we are developing a web application in PHP.

whenever any person want to park his vehicle in park area then have to open his web application it will shows free parking slots, at that place vehicle identification is done by RFID sensors and id is generated for that user ,when a user park his vehicle on parking slot then IR sensor is detected and whenever the person leaves that slot that time for that id payable amount is generated ,whenever next time that person will park his vehicle on any slot that time only a amount is updated for that id.

It will be possible for 5 times,after that **time user have to pay a parking amount online** ,suppose any person not payed amount and if he will go for parking his vehicle that time the buzzer will generate a alarm and it sends a message to authorized person.in this payment for parking is also calculated by inbuilt timer of raspberry Pi.

5.4 ENVIRONMENT

5.4.1 Weather Monitoring System

REST-based Implementation

REST (Representational State Transfer protocol) is a scalable architecture that allows things to communicate over Hyper Text Transfer Protocol and is easily adaptable for IoT applications to provide a communication from things to a central web server.

Interoperability between computer systems on the internet is provided by the web services like REST or RESTful.

Web service is a service offered via World Wide Web by an electronic device to another electronic device, communicating with each other. In that web technology such as HTTP was originally designed for human to machine communication and now utilized for machine to machine communication.

HTTP (Hyper Text Transfer Protocol) is an application protocol used for data communication for the World Wide Web. HTTP is the protocol which is used to exchange or transfer hypertext. HTTP functions as request-response protocol in the client-server computing model. Here client is a web browser and application running on the computer or system is a server.

In this system Raspberry Pi itself act as a server. HTTP request is sent by the client to the server. Then response message is sent by the server to the client. In the response completion status of request and requested content is sent.

5.4.2 Weather Reporting Bot

weather parameter reporting over the internet. It allows the people to directly check the weather stats online without the need of a weather forecasting agency. System uses temperature, humidity as well as rain sensor to monitor weather and provide live reporting of the weather statistics.

The system constantly monitors temperature using temperature sensor, humidity using humidity sensor and also for rain. The system constantly transmits this data to the microcontroller, which now processes this data and keeps on transmitting it to the online web server over a wifi connection. This live updated to be viewed on the online server system. Also system allows user to set alerts for particular instances, the system provides alerts to user if the weather parameters cross those values. Thus the IOT based weather reporting system provides an efficient internet based weather reporting system for users.

5.5 AGRICULTURE

5.5.1 Smart Irrigation

Farming needs sustained irrigation that itself consumes the highest percentage of water resource in any area. For instance, in California alone, irrigation of various ornamental plants and crops consumes over 80 percent of water usage. As experts always perceived, a high percentage of this irrigation water is actually wasted due to lack of supervision and real-time monitoring. In many places like California drought is a recurring problem due to lack of control in the irrigation water.

To solve this problem smart irrigation systems powered by latest IoT technology can help conservation of water resources better by monitoring irrigation through remote sensing technologies. Smart connected sensors work with smart sprinklers leveraging is a great example of the application of IoT in agriculture. To equip the smart irrigation further smart sensors can be embedded in the farmland soil that can easily measure the level of moisture. The moisture measuring sensors of the farmland can further relay the information about moisture level to the smart sprinkler and it can start sprinkling the right amount of water upon the soil.

5.6 PRODUCTIVITY APPLICATIONS

5.6.1 IoT Printer

Most of the current desktop 3D printers are built based on open-source designs from online communities. The largest group of open-source 3D printers is the Self-Replicating Rapid Prototype (RepRap) 3D printers.

A RepRap 3D printer needs to connect to a computer or a microprocessor to feed G Code and provide interface for users to control the 3D printer. However, local computer is a relatively expensive solution comparing to the cost of a RepRap 3D printer; while the microprocessor has much less computing capability comparing to a normal computer, and cannot handle computing-intensive jobs like slicing 3D objects or generating G Code.

An alternate solution is to use the internet of things (IoT) application to control and monitor 3D printers. IoT is the network of physical devices, vehicles, buildings and other items, allowing objects to be sensed and controlled remotely across existing network. IoT and 3D printing are two important new technologies, which progressively impact a lot of areas of the industries and also our everyday life.

DATA ANALYTICS FOR IOT

5.7 APACHE HADOOP ECOSYSTEM

Apache Hadoop is an open source framework for distributed batch processing of big data.

Hadoop Ecosystem includes:

- Hadoop MapReduce
- HDFS
- YARN
- HBase
- Zookeeper
- Pig
- Hive
- Mahout
- Chukwa
- Cassandra
- Avro
- Oozie
- Flume

- Sqoop

- A Hadoop cluster comprises of a Master node, backup node and a number of slave nodes.
- The master node runs the NameNode and JobTracker processes and the slave nodes run the DataNode and TaskTracker components of Hadoop.
- The backup node runs the Secondary NameNode process.

NameNode

NameNode keeps the directory tree of all files in the file system, and tracks where across the cluster the file data is kept. It does not store the data of these files itself. Client applications talk to the NameNode whenever they wish to locate a file, or when they want to add/copy/move/delete a file.

Secondary NameNode

NameNode is a Single Point of Failure for the HDFS Cluster. An optional Secondary NameNode which is hosted on a separate machine creates checkpoints of the namespace.

JobTracker

The JobTracker is the service within Hadoop that distributes MapReduce tasks to specific nodes in the cluster, ideally the nodes that have the data, or at least are in the same rack.

TaskTracker

TaskTracker is a node in a Hadoop cluster that accepts Map, Reduce and Shuffie tasks from the JobTracker.

Each TaskTracker has a defined number of slots which indicate the number of tasks that it can accept.

DataNode

A DataNode stores data in an HDFS file system. A functional HDFS filesystem has more than one DataNode, with data replicated across them.DataNodes respond to requests from the NameNode for filesystem operations.

Client applications can talk directly to a DataNode, once the NameNode has provided the location of the data.Similarly, MapReduce operations assigned to TaskTracker instances near a DataNode, talk directly to the DataNode to access the files.

TaskTracker instances can be deployed on the same servers that host DataNode instances, so that MapReduce operations are performed close to the data.

5.8 USING HADOOP MAPREDUCE FOR BATCH DATA ANALYSIS

- MapReduce job consists of two phases:

Map: In the Map phase, data is read from a distributed file system and partitioned among a set of computing nodes in the cluster. The data is sent to the nodes as a set of key-value pairs. The Map tasks process the input records independently of each other and produce intermediate results as key-value pairs. The intermediate results are stored on the local disk of the node running the Map task.

Reduce: When all the Map tasks are completed, the Reduce phase begins in which the intermediate data with the same key is aggregated.

- Optional Combine Task

An optional Combine task can be used to perform data aggregation on the intermediate data of the same key for the output of the mapper before transferring the output to the Reduce task.

MapReduce Job Execution Workflow

- MapReduce job execution starts when the client applications submit jobs to the Job tracker.
- The JobTracker returns a JobID to the client application. The JobTracker talks to the NameNode to determine the location of the data.
- The JobTracker locates TaskTracker nodes with available slots at/or near the data.
- The TaskTrackers send out heartbeat messages to the JobTracker, usually every few minutes, to reassure the JobTracker that they are still alive. These messages also inform the JobTracker of the number of available slots, so the JobTracker can stay up to date with where in the cluster, new work can be delegated.
- The JobTracker submits the work to the TaskTracker nodes when they poll for tasks. To choose a task for a TaskTracker, the JobTracker uses various scheduling algorithms (default is FIFO).
- The TaskTracker nodes are monitored using the heartbeat signals that are sent by the TaskTrackers to JobTracker.
- The TaskTracker spawns a separate JVM process for each task so that any task failure does not bring down the TaskTracker.

- The TaskTracker monitors these spawned processes while capturing the output and exit codes. When the process finishes, successfully or not, the TaskTracker notifies the JobTracker. When the job is completed, the JobTracker updates its status.

MapReduce 2.0 - YARN

- In Hadoop 2.0 the original processing engine of Hadoop (MapReduce) has been separated from the resource management (which is now part of YARN).
- This makes YARN effectively an operating system for Hadoop that supports different processing engines on a Hadoop cluster such as MapReduce for batch processing, Apache Tez for interactive queries, Apache Storm for stream processing, etc.
- YARN architecture divides architecture divides the two major functions of the JobTracker - resource management and job life-cycle management - into separate components:
    - ResourceManager
    - ApplicationMaster.

YARN Components

Resource Manager (RM): RM manages the global assignment of compute resources to applications. RM consists of two main services:

Scheduler: Scheduler is a pluggable service that manages and enforces the resource scheduling policy in the cluster.

Applications Manager (AsM): AsM manages the running Application Masters in the cluster. AsM is responsible for starting application masters and for monitoring and restarting them on different nodes in case of failures.

Application Master (AM): A per-application AM manages the application's life cycle. AM is responsible for negotiating resources from the RM and working with the NMs to execute and monitor the tasks.

Node Manager (NM): A per-machine NM manages the user processes on that machine.

Containers: Container is a bundle of resources allocated by RM (memory, CPU, network, etc.). A container is a conceptual entity that grants an application the privilege to use a certain amount of resources on a given machine to run a component task.

Hadoop Schedulers

- Hadoop scheduler is a pluggable component that makes it open to support different scheduling algorithms.
- The default scheduler in Hadoop is FIFO.
- Two advanced schedulers are also available - the Fair Scheduler, developed at Facebook, and the Capacity Scheduler, developed at Yahoo.
- The pluggable scheduler framework provides the flexibility to support a variety of workloads with varying priority and performance constraints.
- Efficient job scheduling makes Hadoop a multi-tasking system that can process multiple data sets for multiple jobs for multiple users simultaneously.

5.9 APACHE OOZIE

Apache Oozie is the tool in which all sort of programs can be pipelined in a desired order to work in Hadoop's distributed environment. Oozie also provides a mechanism to run the job at a given schedule.

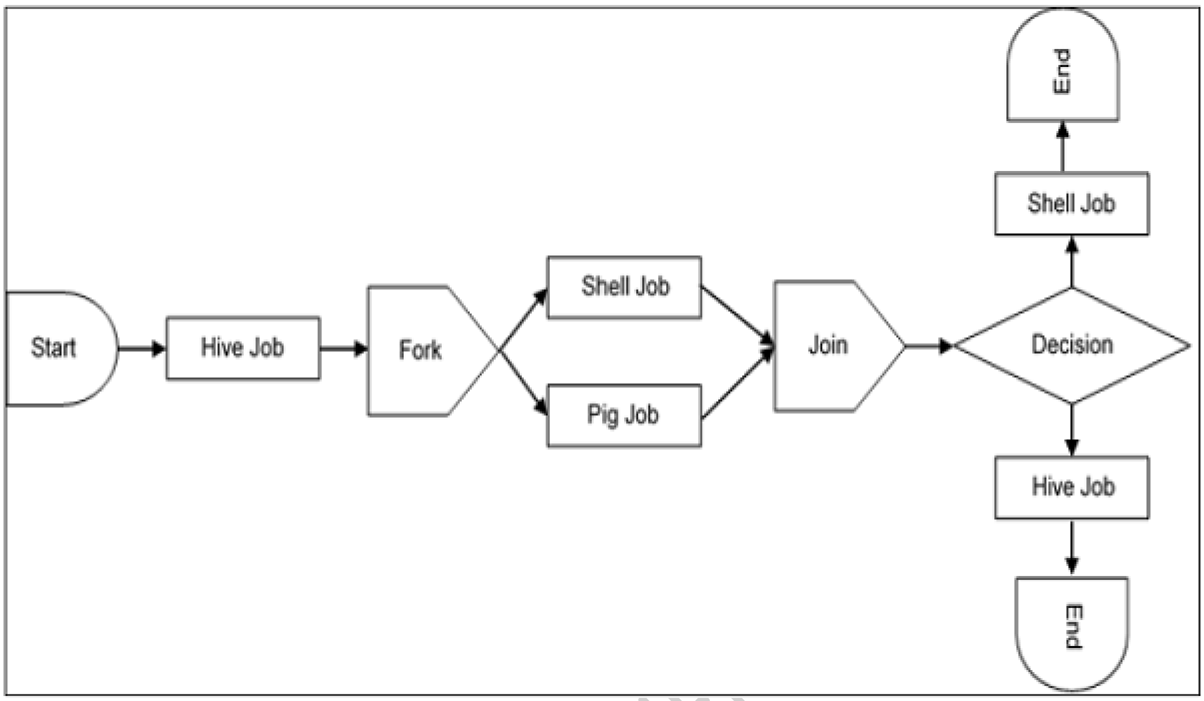
Apache Oozie is a scheduler system to run and manage Hadoop jobs in a distributed environment. It allows to combine multiple complex jobs to be run in a sequential order to achieve a bigger task. Within a sequence of task, two or more jobs can also be programmed to run parallel to each other.

One of the main advantages of Oozie is that it is tightly integrated with Hadoop stack supporting various Hadoop jobs like Hive, Pig, Sqoop as well as system-specific jobs like Java and Shell.

Oozie is an Open Source Java Web-Application available under Apache license 2.0. It is responsible for triggering the workflow actions, which in turn uses the Hadoop execution engine to actually execute the task. Hence, Oozie is able to leverage the existing Hadoop machinery for load balancing, fail-over, etc.

Following three types of jobs are common in Oozie −

- Oozie Workflow Jobs − These are represented as Directed Acyclic Graphs (DAGs) to specify a sequence of actions to be executed.
- Oozie Coordinator Jobs − These consist of workflow jobs triggered by time and data availability.
- Oozie Bundle − These can be referred to as a package of multiple coordinator and workflow jobs.

•


5.10 APACHE SPARK

Apache Spark is a lightning-fast cluster computing technology, designed for fast computation. It is based on Hadoop MapReduce and it extends the MapReduce model to efficiently use it for more types of computations, which includes interactive queries and stream processing. The main feature of Spark is its in-memory cluster computing that increases the processing speed of an application.

Spark is designed to cover a wide range of workloads such as batch applications, iterative algorithms, interactive queries and streaming. Apart from supporting all these workload in a respective system, it reduces the management burden of maintaining separate tools.

Features of Apache Spark-Apache Spark has following features.

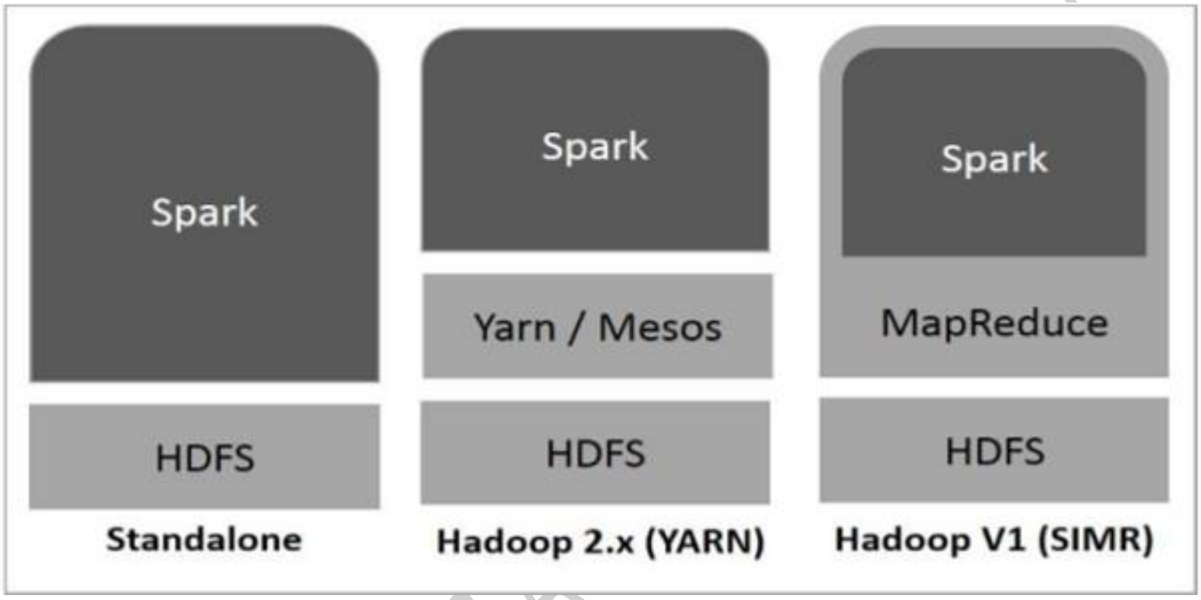
Speed − Spark helps to run an application in Hadoop cluster, up to 100 times faster in memory, and 10 times faster when running on disk. This is possible by reducing number of read/write operations to disk. It stores the intermediate processing data in memory.

Supports multiple languages − Spark provides built-in APIs in Java, Scala, or Python. Therefore, you can write applications in different languages. Spark comes up with 80 high-level operators for interactive querying.

Advanced Analytics − Spark not only supports 'Map' and 'reduce'. It also supports SQL queries, Streaming data, Machine learning (ML), and Graph algorithms.

Spark Built on Hadoop

The following diagram shows three ways of how Spark can be built with Hadoop components.



There are three ways of Spark deployment as explained below.

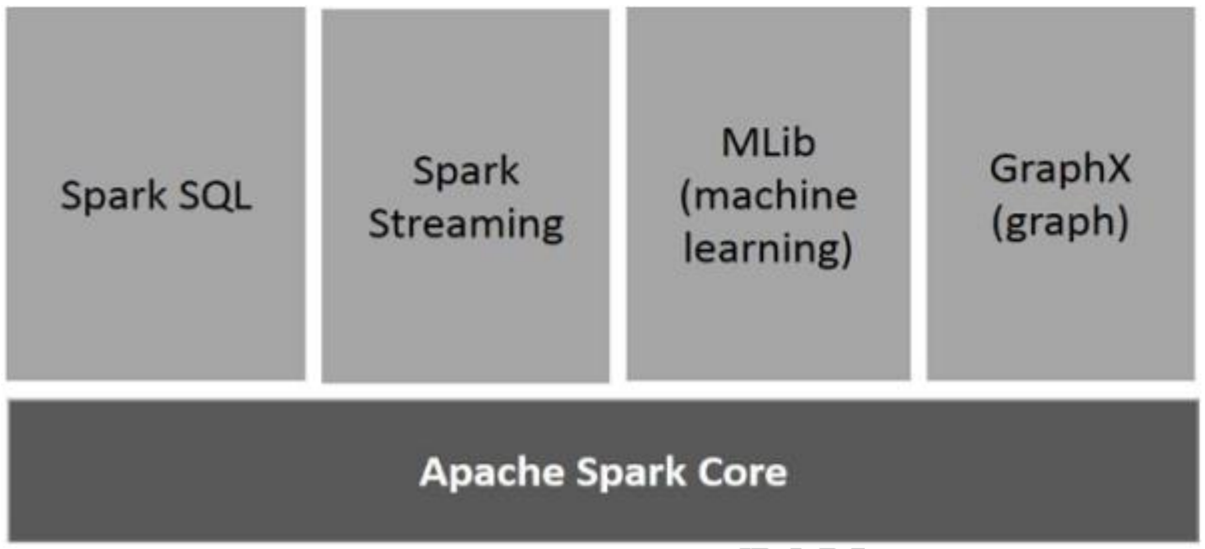
Standalone − Spark Standalone deployment means Spark occupies the place on top of HDFS(Hadoop Distributed File System) and space is allocated for HDFS, explicitly. Here, Spark and MapReduce will run side by side to cover all spark jobs on cluster.

Hadoop Yarn − Hadoop Yarn deployment means, simply, spark runs on Yarn without any pre-installation or root access required. It helps to integrate Spark into Hadoop ecosystem or Hadoop stack. It allows other components to run on top of stack.

Spark in MapReduce (SIMR) − Spark in MapReduce is used to launch spark job in addition to standalone deployment. With SIMR, user can start Spark and uses its shell without any administrative access.

Components of Spark

The following illustration depicts the different components of Spark.



Apache Spark Core

Spark Core is the underlying general execution engine for spark platform that all other functionality is built upon. It provides In-Memory computing and referencing datasets in external storage systems.

Spark SQL

Spark SQL is a component on top of Spark Core that introduces a new data abstraction called SchemaRDD, which provides support for structured and semi-structured data.

Spark Streaming

Spark Streaming leverages Spark Core's fast scheduling capability to perform streaming analytics. It ingests data in mini-batches and performs RDD (Resilient Distributed Datasets) transformations on those mini-batches of data.

MLlib (Machine Learning Library)

MLlib is a distributed machine learning framework above Spark because of the distributed memory-based Spark architecture. It is, according to benchmarks, done by the MLlib developers against the Alternating Least Squares (ALS) implementations. Spark MLlib is nine times as fast as the Hadoop disk-based version of Apache Mahout (before Mahout gained a Spark interface).

GraphX

GraphX is a distributed graph-processing framework on top of Spark. It provides an API for expressing graph computation that can model the user-defined graphs by using Pregel abstraction API. It also provides an optimized runtime for this abstraction.

5.11 APACHE STORM

Apache Storm is a distributed real-time big data-processing system. Storm is designed to process vast amount of data in a fault-tolerant and horizontal scalable method. It is a streaming data framework that has the capability of highest ingestion rates. Though Storm is stateless, it manages distributed environment and cluster state via Apache ZooKeeper. It is simple and you can execute all kinds of manipulations on real-time data in parallel.

Apache Storm is continuing to be a leader in real-time data analytics. Storm is easy to setup, operate and it guarantees that every message will be processed through the topology at least once.

Apache Storm Vs Hadoop

Basically Hadoop and Storm frameworks are used for analyzing big data. Both of them complement each other and differ in some aspects. Apache Storm does all the operations except persistency, while Hadoop is good at everything but lags in real-time computation. The following table compares the attributes of Storm and Hadoop.

Storm	Hadoop
Real-time stream processing	Batch processing
Stateless	Stateful
Master/Slave architecture with ZooKeeper based coordination. The master node is called as **nimbus** and slaves are **supervisors**.	Master-slave architecture with/without ZooKeeper based coordination. Master node is **job tracker** and slave node is **task tracker**.
A Storm streaming process can access tens of thousands messages per second on cluster.	Hadoop Distributed File System (HDFS) uses MapReduce framework to process vast amount of data that takes minutes or hours.

Storm topology runs until shutdown by the user or an unexpected unrecoverable failure.	MapReduce jobs are executed in a sequential order and completed eventually.
Both are distributed and fault-tolerant	
If nimbus / supervisor dies, restarting makes it continue from where it stopped, hence nothing gets affected.	If the JobTracker dies, all the running jobs are lost.

Use-Cases of Apache Storm

Apache Storm is very famous for real-time big data stream processing. For this reason, most of the companies are using Storm as an integral part of their system. Some notable examples are as follows −

Twitter − Twitter is using Apache Storm for its range of "Publisher Analytics products". "Publisher Analytics Products" process each and every tweets and clicks in the Twitter Platform. Apache Storm is deeply integrated with Twitter infrastructure.

NaviSite − NaviSite is using Storm for Event log monitoring/auditing system. Every logs generated in the system will go through the Storm. Storm will check the message against the configured set of regular expression and if there is a match, then that particular message will be saved to the database.

Wego − Wego is a travel metasearch engine located in Singapore. Travel related data comes from many sources all over the world with different timing. Storm helps Wego to search real-time data, resolves concurrency issues and find the best match for the end-user.

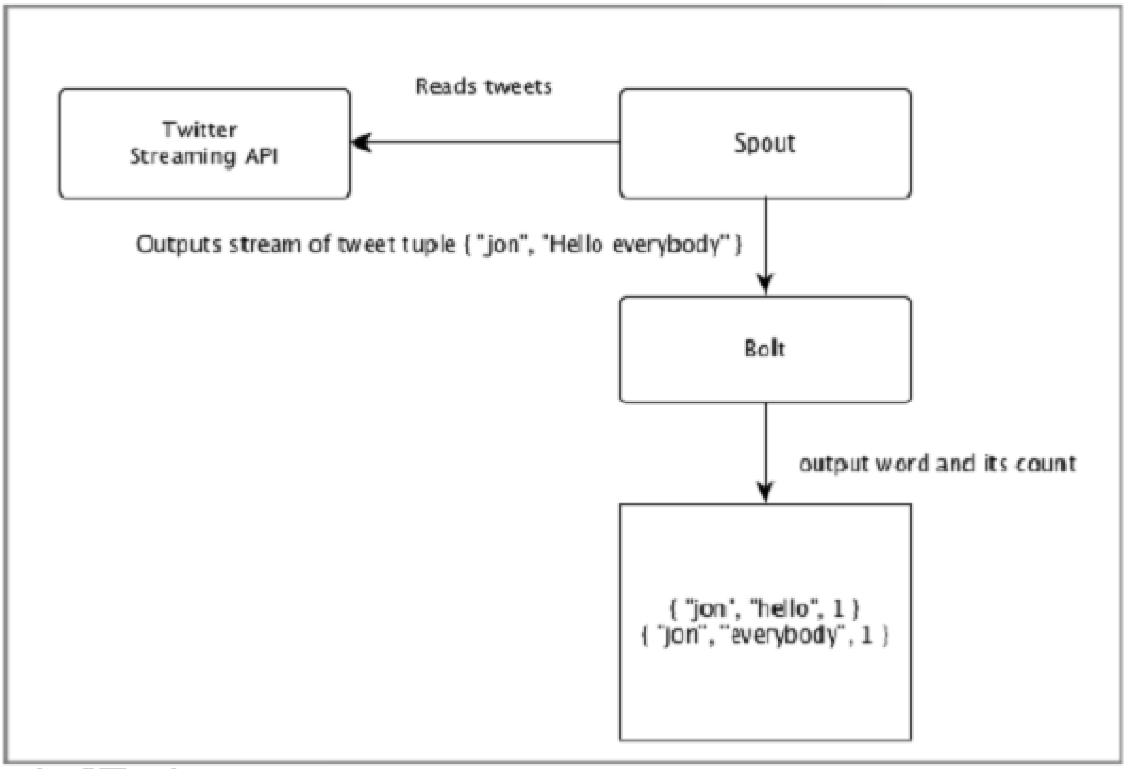
Apache Storm Benefits

- Here is a list of the benefits that Apache Storm offers −
- Storm is open source, robust, and user friendly. It could be utilized in small companies as well as large corporations.
- Storm is fault tolerant, flexible, reliable, and supports any programming language.
- Allows real-time stream processing.
- Storm is unbelievably fast because it has enormous power of processing the data.

- Storm can keep up the performance even under increasing load by adding resources linearly. It is highly scalable.
- Storm performs data refresh and end-to-end delivery response in seconds or minutes depends upon the problem. It has very low latency.
- Storm has operational intelligence.
- Storm provides guaranteed data processing even if any of the connected nodes in the cluster die or messages are lost.

5.12 USING APACHE STORM FOR REAL-TIME DATA ANALYSIS

Let's take a real-time example of "Twitter Analysis" and see how it can be modelled in Apache Storm. The following diagram depicts the structure.


Reads tweets
Twitter Streaming API
Spout
Outputs stream of tweet tuple { "jon", "Hello everybody" }
Bolt
output word and its count
{ "jon", "hello", 1 }
{ "jon", "everybody", 1 }

The input for the "Twitter Analysis" comes from Twitter Streaming API. Spout will read the tweets of the users using Twitter Streaming API and output as a stream of tuples. A single tuple from the spout will have a twitter username and a single tweet as comma separated values. Then, this steam of tuples will be forwarded to the Bolt and the Bolt will split the tweet into individual word, calculate the word count, and persist the information to a configured datasource. Now, we can easily get the result by querying the datasource.

- REST based approach
- WebSocket-based approach

THANK YOU