UNIT I

Syllabus : Embedded Systems : Application Domain and Characteristic of Embedded System, Real time systems and Real time scheduling, Processor basics and System-On-Chip, Introduction to ARM processor and its architecture. IoT : Definition and characteristics of IoT, Internet of Things : Vision, Emerging Trends, Economic Significance, Technical Building Blocks, Physical design of IoT, Things of IoT, IoT Protocols, Logical design of IoT, IoT functional blocks, IoT communication models, IoT Communication APIs, IoT enabling technologies, IoT levels and deployment templates, IoT Issues and Challenges, Applications.

UNIT II

Syllabus : Purpose and requirement specification, Process specification, Domain model specification, information model specification, Service specifications, IoT level specification, Functional view specification, Operational view specification, Device and component integration, Application development.

UNIT III

Syllabus : Horizontal, verticals and four pillars of IoT, M2M : The internet of devices, RFID: The internet of objects, WSN : The internet of transducer, SCADA: The internet of controllers, DCM: Device, Connect and Manage, Device : Things that talk, Connect: Pervasive Network, Manage : To create business values. IoT Physical Devices and Endpoints : Basic building blocks of and IoT device, Exemplary device: Raspberry Pi, Raspberry Pi interfaces, Programming Raspberry Pi with Python, Other IoT Devices.

UNIT IV

Syllabus : Protocol Standardization for IoT, Efforts, M2M and WSN Protocols, SCADA and RFID Protocols, Issues with IoT Standardization, Unified Data Standards, Protocols – IEEE 802.15.4, BACNet Protocol, Modbus, KNX, Zigbee Architecture, Network layer, APS layer.

IoT Security : Vulnerabilities of IoT, Security Requirements, Challenges for Secure IoT, Threat Modeling, Key elements of IoT Security : Identity establishment, Access control, Data and message security, Non-repudiation and availability, Security model for IoT.

UNIT V

Syllabus : Web of Things versus Internet of Things, Two Pillars of the Web, Architecture Standardization for WoT, Platform Middleware for WoT, Unified Multitier WoT Architecture, WoT Portals and Business Intelligence. Cloud of Things: Grid/SOA and Cloud Computing, Cloud Middleware, Cloud Standards – Cloud Providers and Systems, Mobile Cloud Computing, The Cloud of Things Architecture.

UNIT VI

Syllabus : Introduction to Cloud Storage Models, Communication API, WAMP : AutoBahn for IoT, Xively Cloud for IoT, Python Web Application Framework : Djanjo, Amazon Web Services for IoT, SkyNet IoT Messaging Platform. Case Studies : Home Intrusion Detection, Weather Monitoring System, Air Pollution Monitoring, Smart Irrigation.

❑❑❑

UNIT I

CHAPTER 1

Introduction to Embedded System and Internet of Things

Syllabus Topics

Embedded Systems : Application Domain and Characteristic of Embedded System, Real time systems and Real time scheduling, Processor basics and System-On-Chip, Introduction to ARM processor and its architecture. **IoT :** Definition and characteristics of IoT, Internet of Things : Vision, Emerging Trends, Economic Significance, Technical Building Blocks, Physical design of IoT, Things of IoT, IoT Protocols, Logical design of IoT, IoT functional blocks, IoT communication models, IoT Communication APIs, IoT enabling technologies, IoT levels and deployment templates, IoT Issues and Challenges, Applications.

Syllabus Topic : Embedded Systems

1.1 Introduction

1.1.1 Embedded Systems

Q. What is Embedded systems ?

- Embedded systems can be defined as an electronic device which is developed to perform specific set of tasks using limited set of hardware and small software system coordinating the hardware.
- Embedded system has a microcontroller or a microprocessor at its core and the task which needs to be performed by this system are inserted into the ROM of such devices. A personal computer also follows this definition but we don't consider PC as a whole embedded system because of number of tasks it can perform. We consider that embedded systems will only perform limited set of activities.
- We are surrounded by the embedded systems and its application. Let's see some day to day examples of embedded systems. Remote control, microwave, washing machine, elevator controller, small robotic applications, traffic signal controller, and the list can go on and on.

Syllabus Topic : Application Domain and Characteristic of Embedded System

1.1.2 Application Domain of Embedded System

Q. Explain applications of embedded systems.

As we have discussed embedded systems are integrated part of our day to day life activities. Let's see some more examples in various fields.

- **Electronic products :** Cameras, Music systems, TVs, DVD players, microwave, refrigerators, etc.
- **Household applications :** AC, Fire alarm system, intruder alert systems, etc.
- **Medical devices :** ECGs, EEGs, Scanners, Heart rate monitors, etc.
- **Industries :** Automations devices, controllers, alarms, etc.
- **Banking Sector :** ATMs, currency counter, checkers, etc.
- **Military Services :** Radios, military surveillance systems, etc.
- **Computer Devices :** Printers, scanners, mouse, webcams, etc.
- **Networking :** Routers, hubs, switches, etc.
- **Aviation Controllers :** Airplane controls, guidance systems, etc.
- This list is unending as the technology and user requirement grows day by day.

1.1.3 Desirable Characteristics of Embedded Systems

Q. Explain characteristics of embedded systems.

- It should effectively implement a small and limited set of functionalities.
- Most of the embedded systems has battery backup as continuous electricity supply might not be available. So power dissipation must be controlled.
- These systems must be reliable.
- Processing carried out in an embedded system must consider storage of data as it has limited storage capacity.
- Functionalities must be encapsulated from user alterations.
- Many of the embedded system have to handle real time tasks effectively.
- Embedded systems must not be bulky.
- It has limited set of external peripherals.

1.2 Real Time Systems and Real Time Scheduling

- In operating systems, program executing in the RAM is stated as a process. An Operating System consist of plenty of heavy processes and small tasks running in the backend to properly run our systems like Desktops, Laptops etc.
- Out of such tasks system can have tasks which are strictly bound to timings which are referred as real time tasks. This section contains details about types of real time tasks and its scheduling algorithms.

Syllabus Topic : Real Time Systems

1.2.1 Real Time Systems

Q. What is real time systems? Explain with examples.

- Each task in an Operating System is related with some time related parameters like arrival time, starting time, processing time (burst time), finishing time etc.
- Some of the tasks are also related with deadline time, may be starting or ending, which specifies that task must start before starting deadline and complete before finishing deadline.
- A task whose performance is judged based on these deadlines is termed as real time tasks.
- Output of real time tasks is considered correct or useful if it is generated within the time constraints.
- If a real time task fails to complete within the deadline it leads to system failure or reduced quality of output.
- Hardware systems that handle real time tasks are real time systems. Real time systems can have normal tasks along with real time tasks.
- It is not mandatory that all the tasks in real time systems should be real time tasks.
- General purpose operating systems used for basic interaction of hardware and software to manage the computer are not sufficient to handle real time tasks, so special operating systems are developed for handling real time tasks called Real Time Operating Systems(RTOS).

Examples of real time tasks

(i) Air Traffic Controls
(ii) Anti-missile systems in war planes
(iii) Airbags controllers
(iv) Life support systems at hospitals
(v) Telecommunications
(vi) Robotics
(vii) Real time database

- All above systems processes real time data and generates appropriate results for the same within the time bound.
- For example Anti-missile systems in war planes require launching heat flares as soon as a (heat sensing) missile is lunched on the plane to deviate the missile from it.
- If the data is not processed within time constraints human life loss is inevitable. Same can be said for the other examples above.

1.2.2 Types of Real Time Tasks

Q. What are the types of real time tasks? Explain in details.

- Real time tasks can be differentiated based on what happens to the expected output if the deadline of the task is missing. Real time tasks are categorized into three types viz. hard, soft and firm real time tasks.

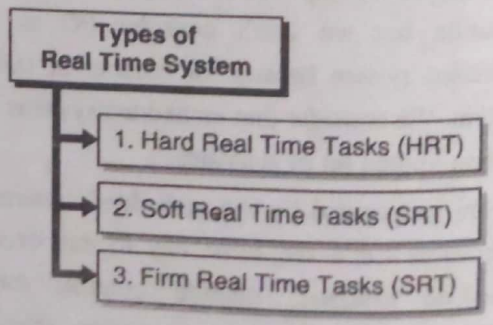

Fig. C1.1 : Types of Real Time System

- For the following discussion consider *'T'* stands for task, *'C'* stands for completion time, *'D'* stands for

deadline, 'Q' stands for quality/significance of the output and *'F'* stands for fine/penalty.

➜ **1. Hard Real Time Tasks (HRT)**

- In HRT it is compulsory for a task to be completed within the deadline, failing to do so results in disastrous effects most of the time and might lead to loss of human lives.
- If a HRT task misses the deadline i.e. **C > D** the output generated is completely useless. In other words value of the outcome is zero and the output is rejected completely.
- The system fails if HRT misses the deadline.

☞ **Example**

(i) Air Traffic Control System
(ii) Air bags controls in a car
(iii) Anti-Missile system

- Failing to achieve results in time bound results in loss of lives.
- Required Condition: $C \leq D$ then $Q = 1$;
- If $C > D$ then $Q = 0$;

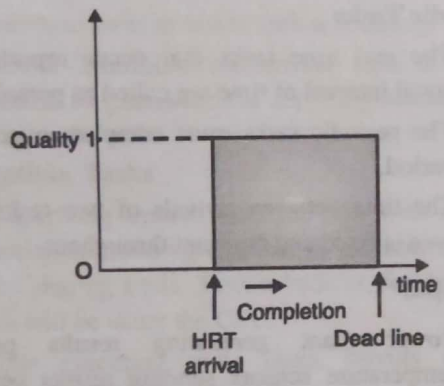

Fig. 1.2.1 : Hard Real Time tasks Execution

- As shown in the Fig. 1.2.1, the outcome of the HRT task is significant if and only if the deadline is followed or else (after the deadline) result is zero even if the output is achieved.

➜ **2. Soft Real Time Tasks (SRT)**

- In SRT tasks the deadline is followed but it is not mandatory.
- This means that even if the deadline is missed the output is not rejected completely.
- The task that misses the deadline is affected with some penalty or fine in the form of reduced quality of service.
- The penalty increases as the completion time crosses the deadline. So higher the *(C - D)* higher the penalty.
- As the completion time crosses the deadline the value of the output reduces until it reaches zero i.e. output is of no use.
- There are no catastrophic effects or loss of human lives associated in SRT on missing the deadline but quality of the system drops over time.
- Even if the deadline is missed the system keeps on working but the quality is lowered.

☞ **Examples**

(i) Online Gaming
(ii) Constantly updating online sites (eg: Scoreboard)

- Consider example of constantly updating online scoreboard, it is expected to update the scoreboard as soon as the actual game proceed, but even if the results are updated a bit late the scoreboard still shows valid score and are still useful.
- If $C \leq D$ then $Q = 1$;
- If $C > D$ then Q lowers and move till $Q = 0$;

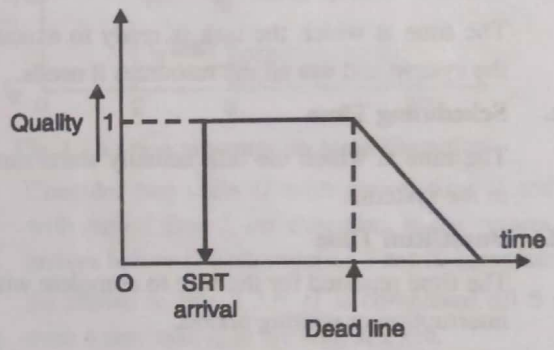

Fig. 1.2.2 : Soft Real Time tasks Execution

- As shown in the Fig. 1.2.2, the outcome of the SRT task is significant even if the deadline is missed and result is still acceptable after deadline till it becomes zero.

➜ **3. Firm Real Time Tasks (SRT)**

- FRT tasks can be treated as a combination of HRT and SRT.
- In FRT task, if the deadline is missed the value of the output is dropped to zero (as done in HRT) but the system is still working without any catastrophic effects (as done in SRT).
- If the deadline is missed the output is rejected completely, but the system is still running.
- The output is allowed to miss the deadline but not to regularly. If the system misses the deadline to commonly the system failure occurs.

☞ **Example**

Video processor: Consider a system processes video frame by frame and shows continuous video playback.

Missing 2 or 3 frames in between the complete processing doesn't cause any disruption in the video playback but missing too many frames leads to interrupted video.

1.2.3 Real Time Scheduling Terms

Q. Explain following terms regarding real time scheduling : Arrival time, Ready time, deadline, scheduling time, burst time, waiting time, completion time.

Q. What are periodic, aperiodic and sporadic tasks?

Before moving to actual scheduling algorithms for real time tasks let's first see some of the terminologies that are frequently used in scheduling.

1. **Some important definitions**

a. Arrival Time

The time at which the tasks arrive in the system but might not have all the resources it needs to execute.

b. Release / Ready Time

The time at which the task is ready to execute in the system and has all the resources it needs.

c. Scheduling Time

The time at which the task actually starts running in the system.

d. Burst/Run Time

The time required for the task to complete without interruption or waiting period.

e. Waiting Time

The time the task waits in the system after it arrives in the system.

f. Completion time

It is the moment at which the task finishes its execution. The total time the task takes to complete its execution along with waiting time. (Waiting time + Run Time)

g. Deadline

The time before which the task should begin or end is called as deadline of the task.

h. Tardiness

Tardiness term is used when the task misses the deadline. The amount of time by which the task misses its deadline is called as tardiness.

Tardiness = Completion Time - deadline

i. Laxity

Laxity term is used when the task follows the deadline. The maximum time the task can wait before successfully serving the deadline is called as laxity.

Laxity = Deadline – Completion Time

Consider following Example

– Two tasks A and B need to be scheduled by OS with following specifications. Try to make sure that all the tasks are completed before the deadline.

Task	Arrival Time	CPU Burst	Deadline
A	0	5	10
B	2	3	5

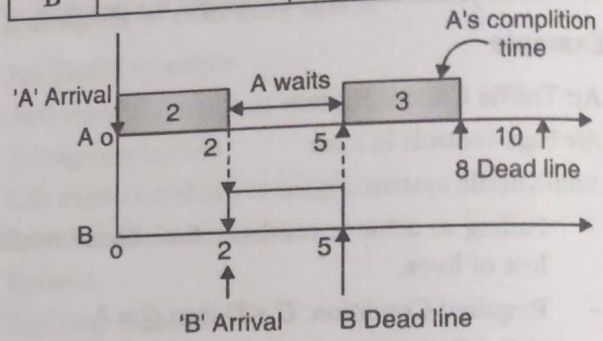

Fig. 1.2.3 : Task Scheduling in OS

2. **Periodic Tasks**

– The real time tasks that occur repeatedly after equal interval of time are called as periodic tasks.

– The periodic tasks must complete once per time period.

– The time between arrivals of two tasks of same type is fixed and constant throughout.

Example

– Power plant generating results per hour, temperature sensors sending results per minute, train leaving at 6 p.m. daily etc.

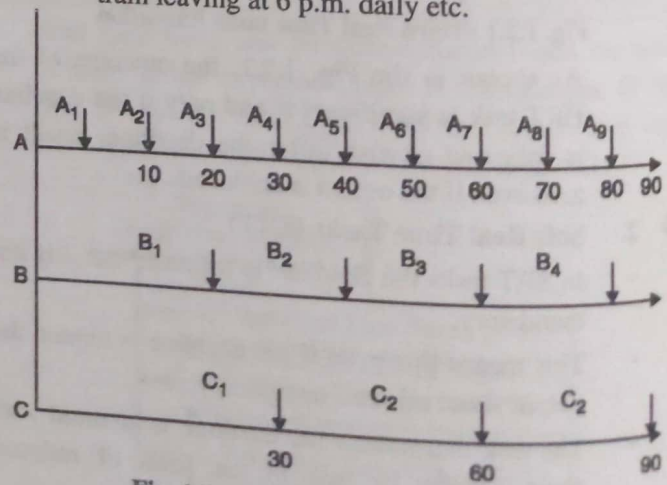

Fig. 1.2.4 : Periodic Tasks Execution

– As shown in Fig. 1.2.4 *A*, *B* and *C* are periodic tasks that occur after fixed interval of time,

Task	Time Period
A	10
B	20
C	30

- In case of periodic tasks the time period is the deadline for each instance.

3. Aperiodic Tasks

- The streams of tasks that occur with irregular arrival are termed as aperiodic tasks.
- The time between arrivals of two aperiodic tasks (interarrival time) is random and even can be zero.
- Two aperiodic tasks can arrive at same time interval, in such scenarios both the tasks cannot be scheduled at the same time.
- Because of these reasons the aperiodic tasks have soft deadlines.

4. Sporadic Tasks

- Aperiodic tasks with a hard deadline are termed as sporadic tasks.
- The time between arrivals of two sporadic tasks is never zero so as to assure task scheduling.
- Without minimum interarrival time it is not possible to guarantee the servicing/scheduling of all sporadic tasks.

5. Pre-emptible Tasks

- In operating system multiple task with different priority are running together at the same time on time sharing basis. Task scheduler decides which task will be using the CPU.
- While executing some tasks, usually a lower priority task is interrupted by a higher priority task and CPU is taken over by higher priority task even if the lower priority task is incomplete.
- Such tasks which can be interrupted by other tasks are called as pre-emptible tasks.

Task	Arrival time	Burst time	Prority
t_1	0	5	1
t_2	2	5	1

Lower number specifies higher priority

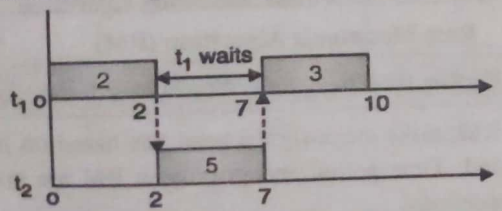

Fig. 1.2.5 : pre-emptible tasks Execution

- Consider two tasks *t1* with arrival time 0 and *t2* with arrival time 2, are executing in the system. *t1* arrives before *t2* both requires 5 sec to complete and *t2* has higher priority than *t1*.
- As shown in Fig. 1.2.5, *t1* is pre-empted at 2 sec before completion as *t2*, a task with higher priority arrives in the system.
- t1 waits for t2 to finish its execution, then continues from 7 sec to 10 sec.

6. Non Pre-emptible Tasks

- In operating system the tasks that cannot be interrupted until completion are called as non pre-emptible tasks.

Task	Arrival time	Burst time
t_1	0	5
t_2	2	5

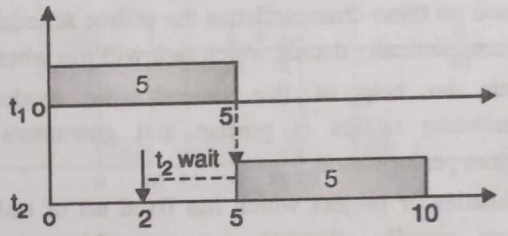

Fig. 1.2.6 : Non pre-emptible tasks Execution

- Consider two tasks *t1* with arrival time 0 and *t2* with arrival time 2, are executing in the system. *t1* arrives before *t2* both requires 5 sec to complete.
- As shown in Fig. 1.2.6, *t1* is completed till 5 sec even a new task *t2* is arriving at 2 sec.
- t2 waits for t1 to finish its execution, then continues from 5 sec to 10 sec.

Syllabus Topic : Real Time Scheduling

1.2.4 Real Time Scheduling

Q. Explain the concept of Real Time Scheduling Algorithms with example.

The operating system that handles the real time tasks are called as real time operating systems. The concept scheduling is well known in operating system, the part of OS that decides which task to run at what point of time in OS. General purpose OS have different scheduling algorithms like FCFS, SJF, SRT etc. But same might not be applicable in real time sense. So in the next section we will see two algorithms that are specifically designed for real time tasks.

Based on the characteristics of the real time system and available information about release time, burst time etc., real time scheduling can be categorized in two types :

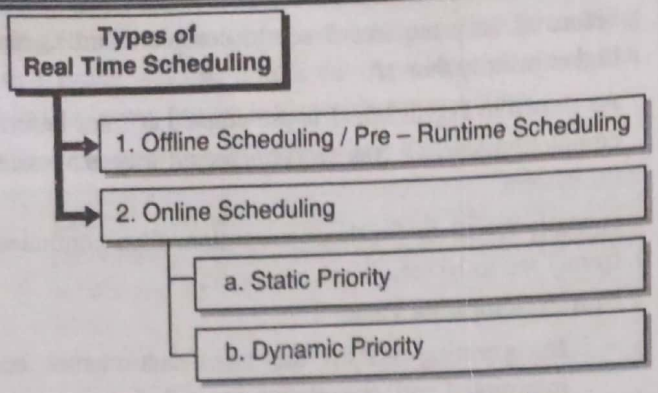

Fig. C1.2 : Types of Real Time Scheduling

➜ **1. Offline Scheduling / Pre - Runtime Scheduling**

- In case of offline scheduling all the time related parameters like release time, arrival time, deadlines, execution time etc., must be known for all the tasks in the system before system starts executing.
- Based on these characteristics the offline scheduler can deterministically decide which task will run when.
- With the help of this deterministic model, this scheduling tactics is precise and guarantees great system performance.
- Industries or project which has fixed set of tasks that never usually changes can use this scheduling mechanism.
- But this strategy is inflexible. Inflexible in the sense that no new addition can be done to the running set of tasks.
- Any new entry to the system means all the scheduling has to be done form the beginning including new and previous set of tasks.

➜ **2. Online Scheduling**

- If the number of real time tasks and some parameters related to that are not known beforehand, in such cases the offline scheduling fails. In this scenario online scheduling policies are used.
- Systems which handle such dynamic set of tasks can implement online scheduling.
- Because of the unpredictability of the tasks, these set of algorithms might not make the best use of all available resources.
- Online scheduling is categorized into two types viz. *static priority* and *dynamic priority*.

➜ **a. Static Priority**

- In case of static priority all the tasks coming in the system for scheduling are associated with priority value. Generally a higher priority task runs before lower priority task. This strategy can have pre-emptive and non pre-emptive approach.
- In non pre-emptive approach the highest priority task is run from the set until it has finished its execution.
- In pre-emptive approach the already running process can be interrupted if another higher priority task becomes ready and this task will overtake the execution.

➜ **b. Dynamic Priority**

- In this approach the priority of the task is decided at the run time. Because of this the algorithm implementation and computation involved is complex.
- There are many algorithms that follows dynamic priority scheduling. Normally it is categorized in two approaches viz. planning based approach and best effort approach.
- The planning based approach ensures that all the tasks that are accepted will be completed before its deadlines. Whenever a new task is entering the system feasibility of the task is checked at runtime and accepted only when it can be served.
- In case of best effort strategy, as the name suggests it tries to maximize the number of real time tasks scheduled. This approach ensures the completion of all the hard real time tasks and tries to optimize the soft real time tasks. It aborts any soft real time tasks which are missing the deadlines.

1.2.5 Real Time Scheduling Algorithms

Q. Explain different types of real time scheduling algorithms.

In this section we will study the two most commonly used real time scheduling algorithms.

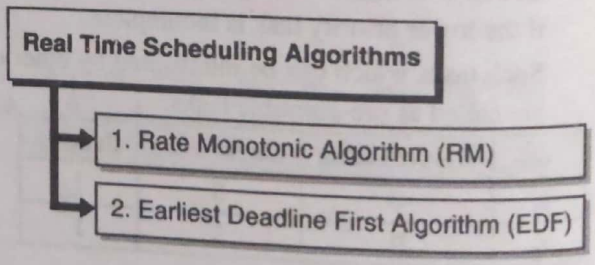

Fig. C1.3 : Real Time Scheduling Algorithms

➜ **1. Rate Monotonic Algorithm (RM)**

Q. Explain Rate Monotonic Algorithm with example.

- In RM, tasks are assigned priorities based on its time period. Time period and priority in RM are inversely proportional.
- Higher the time period of the task lower is the priority.

- This means that the process with lowest time period gets the highest priority for scheduling.
- RM is a pre-emptive static priority approach. So higher priority task will be scheduled interrupting the lower priority task.
- RM has a sufficient condition based on which one can determine whether the set of tasks can be scheduled with RM or not.

Equation

$$\sum_{i=1}^{n} \frac{C_i}{P_i} \leq n(2^{1/n} - 1)$$

n = Number of tasks to be scheduled

- In the above equation *C* represents burst time/execution time *P* represents time period of the task. LHS of the equation specifies total CPU utilization.
- If total CPU utilization is less than the RHS then the given set of tasks can be surely scheduled by RM following all the deadlines. Above equation is well known as utilization bound test.
- Again this is a sufficient condition but NOT necessary condition. This means that even if the condition is not being followed by the set of tasks there is still a possibility that set of tasks can be scheduled by RM. For this we have to solve the given set of tasks manually.
- In case of periodic tasks the time period is the deadline for each instance.

Let's see some Examples for RM

Ex. 1.2.1

Check whether the given set of tasks follows utilization bound test. Try to schedule following set of real time tasks using Rate monotonic Algorithm.

OR

Check whether following set of real time tasks are schedulable using Rate monotonic Algorithm or not. If not which task misses the deadline?

Assume arrival time for all the tasks is zero (0).

Task	Time Period	CPU Burst
T1	10	2
T2	15	4
T3	35	10

Soln. :

- First of all we will try to check the above set of periodic tasks by sufficient condition of RM i.e. utilization bound test.

$$\sum_{i=1}^{n} \frac{C_i}{P_i} \leq n(2^{1/n} - 1)$$

n = Number of tasks to be scheduled

- Hence for given set of tasks

  L.H.S. = $2/10 + 4/15 + 10/35 = 0.752$

  R.H.S. = $3(2^{1/3} - 1) = 0.779$

- From above calculation,

  L.H.S < R.H.S

- Utilization bound test is followed for RM. So given set is solvable using Rate Monotonic Algorithm.
- Now let's try to schedule the given set of task :

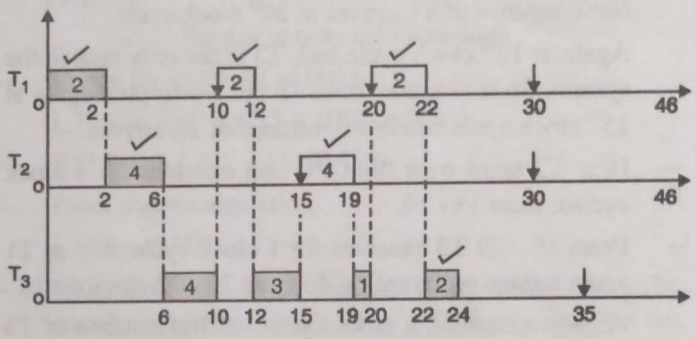

Fig. P. 1.2.1

System remains idle as no task is on the system from 24th clock cycle to 30th clock cycle when T1 and r2 arrives again.

Explanation

- In RM scheduling algorithm, highest priority is given to the task which has the lowest time period.
- Task having Lowest time period is given highest priority

So,

Task	Period	Priority
T1	High	1
T2	Medium	2
T3	Low	3

- As shown in Fig. P. 1.2.1 we have time instance on Y-axis and tasks shown on X-axis.
- For scheduling CPU checks how many tasks are in the system at given moment of time and schedules the one having highest priority.
- So as all the tasks have 0 as arrival time, first instance of T1, T2 and T3 are in the system at 0th clock cycle.
- T1 has highest priority and CPU burst of 2 so T1 is scheduled from 0 - 2 clock cycle. T1 has time period of 10 so it next instance of T1 arrives at (0 + 10) i.e. 10th clock cycle.

- Now at 2nd clock cycle T2 and T3 are in the system, so T2 is scheduled with burst of 4 from 2 – 6 clock cycle which has higher priority than T3. Next instance arrives at (0+15) i.e. 15th clock cycle.
- Now at 6th clock cycle only T3 is in the system so it is scheduled until any higher priority task appears in OS.
- So T3 is scheduled from 6 - 10 clock cycle. At 10th clock cycle next instance of T1 appears in the system. So at 10th clock cycle task T1 and T3 are in the system and as time period of T1 is less than T3 priority is given to T1 pre-empting execution of task T3.
- So from 10 - 12 clock cycle task T1 runs in the system. Next instance of T1 arrives at 20th clock cycle.
- Again at 12th clock cycle task T3 is the only task in the system. So it executes from 12 but interrupted again at 15th clock cycle where 2nd instance of T2 arrives.
- Here T2 takes over the CPU and executes for 4 clock cycles, from 15 - 19.
- From 19 - 20 T3 executes for 1 clock cycle only as T1 again arrives pre-empting T3 from 20 - 22 clock cycle.
- Finally remaining 2 clock cycles of first instance of T3 completes from 22 - 24 completing its execution.
- In this way till 24th clock cycle 3 instances of T1, 2 instances of T2 and a single instance of T3 has finished all of them following deadlines.
- Note that in case of periodic tasks time period is deadline itself.
- Further execution of the set of tasks can also be shown. But system will remain idle from 24 - 30 as no tasks are alive in the system within this time period. 4th instance of T1 and 3rd instance of T2 arrives at 30th clock cycle, where T1 will be scheduled.
- In this way rate monotonic algorithms work, giving highest priority to task having lowest time period.
- Make sure that at any point of time task with highest priority will be executing in the system.

Ex. 1.2.2

Check whether the given set of tasks follows utilization bound test. Try to schedule following set of real time tasks using Rate monotonic Algorithm.

OR

Check whether following set of real time tasks are schedulable using Rate monotonic Algorithm or not. If not which task misses the deadline?

Assume arrival time for all the tasks is zero (0).

Task	Time Period	CPU Burst
T1	10	2
T2	15	4
T3	25	10

Soln. :

- First of all we will try to check the above set of periodic tasks by sufficient condition of RM i.e. utilization bound test.

$$\sum_{i=1}^{n} \frac{C_i}{P_i} \leq n(2^{1/n} - 1)$$

n = Number of tasks to be scheduled

- Hence for given set of tasks

L.H.S. = $2/10 + 4/15 + 10/25 = 0.867$

R.H.S. = $3(2^{1/3} - 1) = 0.779$

- From above calculation,

L.H.S > R.H.S

- Utilization bound test fails for given set of tasks. As stated earlier this test is only sufficient condition for RM. So there are still chances that given set can be scheduled using Rate Monotonic Algorithm.
- Let's try to schedule the given set of task :

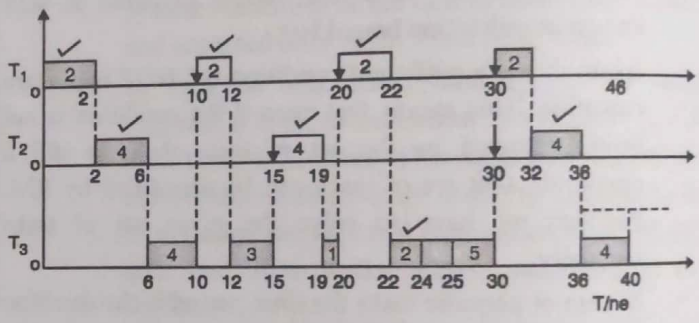

Fig. P. 1.2.2

Explanation

- In RM scheduling algorithm, highest priority is given to the task which has the lowest time period.

  Task having Lowest time period is given highest priority

  So,

Task	Period	Priority
T1	High	1
T2	Medium	2
T3	Low	3

- As shown in Fig. P. 1.2.2, we have time instance on Y-axis and tasks shown on X-axis.
- For scheduling CPU checks how many tasks are in the system at given moment of time and schedules the one having highest priority.
- So as all the tasks have 0 as arrival time, first instance of T1, T2 and T3 are in the system at 0th click cycle.
- T1 has highest priority and CPU burst of 2 so T1 is scheduled from 0 - 2 clock cycle. T1 has time period of

10 so it next instance of T1 arrives at (0 + 10) i.e. 10th clock cycle.

- Now at 2nd clock cycle T2 and T3 are in the system, so T2 is scheduled with burst of 4 from 2 - 6 clock cycle which has higher priority than T3. Next instance arrives at (0+15) i.e. 15th clock cycle.
- Now at 6th clock cycle only T3 is in the system so it is scheduled until any higher priority task appears in OS.
- So T3 is scheduled from 6 - 10 clock cycle. At 10th clock cycle next instance of T1 appears in the system. So at 10th clock cycle task T1 and T3 are in the system and as time period of T1 is less than T3 priority is given to T1 pre-empting execution of task T3.
- So from 10 - 12 clock cycle task T1 runs in the system. Next instance of T1 arrives at 20th clock cycle.
- Again at 12th clock cycle task T3 is the only task in the system. So it executes from 12 but interrupted again at 15th clock cycle where 2nd instance of T2 arrives.
- Here T2 takes over the CPU and executes for 4 clock cycles, from 15 - 19.
- From 19 - 20 T3 executes for 1 clock cycle only as T1 again arrives pre-empting T3 from 20 - 22 clock cycle.
- Finally remaining 2 clock cycles of first instance of T3 completes from 22 - 24 completing its execution.
- In this way till 24th clock cycle 3 instances of T1, 2 instances of T2 and a single instance of T3 has finished all of them following deadlines.
- Note that in case of periodic tasks time period is deadline itself.
- System will remain idle from 24 - 25 as no tasks are alive in the system within this time period. 4th instance of T1 and 3rd instance of T2 arrives at 30th clock cycle, where T1 will be scheduled.
- 2nd instance of T3 arrives at 25 so it is the only task alive at 25th clock cycle so it is scheduled for 5 clock cycles from 25 - 30 and at 30th clock cycle T1 takes over.
- In this way rate monotonic algorithms work for the given set of tasks even though the utilization test fails.

Ex. 1.2.3

Check whether the given set of tasks follows utilization bound test. Try to schedule following set of real time tasks using Rate monotonic Algorithm.

OR

Check whether following set of real time tasks are schedulable using Rate monotonic Algorithm or not. If not which task misses the deadline?

Assume arrival time for all the tasks is zero (0).

Task	Time Period	CPU Burst
T1	4	1
T2	6	2
T3	10	3

Soln. :

- First of all we will try to check the above set of periodic tasks by sufficient condition of RM i.e. utilization bound test.

$$\sum_{i=1}^{n} \frac{C_i}{P_i} \leq n(2^{1/n} - 1)$$

n = Number of tasks to be scheduled

- Hence for given set of tasks

L.H.S. = 2/10 + 4/15 + 10/25 = 0.883

R.H.S. = 3 ($2^{1/3}$ – 1) = 0.779

- From above calculation,

L.H.S > R.H.S

- Utilization bound test fails for given set of tasks. As stated earlier this test is only sufficient condition for RM. So there are still chances that given set can be scheduled using Rate Monotonic Algorithm.
- Let's try to schedule the given set of task:

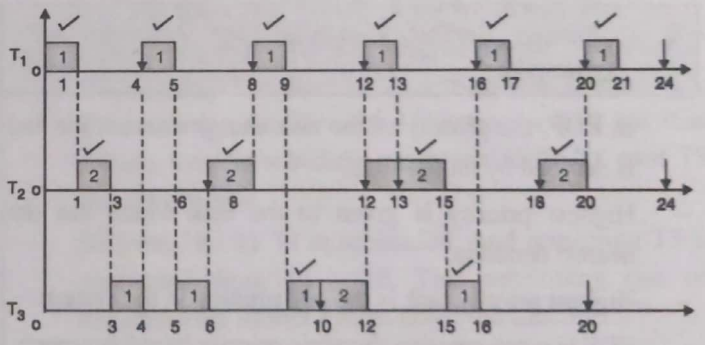

Fig. P. 1.2.3

Explanation

- In RM scheduling algorithm, highest priority is given to the task which has the lowest time period.

  Task having Lowest time period is given highest priority

  So,

Task		Period
T1	High	1
T2	Medium	2
T3	Low	3

- As shown in Fig. P. 1.2.3 we have time instance on Y-axis and tasks shown on X-axis.

- As all the tasks have 0 as arrival time, first instance of T1, T2 and T3 are in the system at 0^{th} clock cycle.
- T1 has highest priority and CPU burst of 1 so T1 is scheduled from 0 - 1 clock cycle. T1 has time period of 4 so it next instance of T1 arrives at (0 + 4) i.e. 4^{th} clock cycle.
- T2 is scheduled from 1 - 3 for 2 clock cycles, completing its first instance.
- T3 is scheduled for 1 clock cycle from 3 - 4 as higher priority instance of T1 arrives at 4^{th} clock cycle.
- T1 is scheduled from 4 - 5 clock cycle completing its second instance and giving the controls of CPU to T3 as it is the only task in the system.
- At 6^{th} clock cycle 2^{nd} instance of T2 arrives and is scheduled from 6 - 8 clock cycle.
- At 8^{th} clock cycle T1's 3^{rd} instance and T3's 1^{st} instance are in the system, and scheduler picks T1 as it has higher priority than T3.
- 3^{rd} instance of T1 completes from 8 - 9 and 1^{st} instance of T3 is completed finally from 9 - 10 clock cycle.
- Given set of tasks are scheduled further for RM as shown in figure.

2. Earliest Deadline First Algorithm (EDF)

Q. Explain Earliest Deadline First Algorithm with example.

- In EDF, the priority of the task changes at runtime and is decided by its deadline.
- Highest priority is given to the task which has the nearest deadline.
- Highest priority task is the one running in the system.
- EDF is a pre-emptive dynamic priority based approach.
- For a given set of tasks if RM fails, EDF can be used to scheduled such set of tasks.
- Periodic, aperiodic and sporadic set of tasks can use this algorithm as scheduling principal.
- Necessary condition for EDF is that the CPU utilization should be less than 1.
- As EDF follows dynamic approach while determining the priorities of the tasks, there is an overhead of EDF implementation and calculations performed in EDF. This is the drawback of EDF.

Let's see some examples

Ex. 1.2.4

Try to schedule following set of real time tasks using Earliest Deadline First Algorithm.

OR

Check whether following set of real time tasks are schedulable using Earliest Deadline First algorithm or not. If not which task misses the deadline?

Task	Arrival Time	CPU Burst	Deadline
T1	0	1	2
T2	0	2	5
T3	2	2	4
T4	3	2	10
T5	6	2	9

Soln. :

- Let's try to solve the above set of tasks:

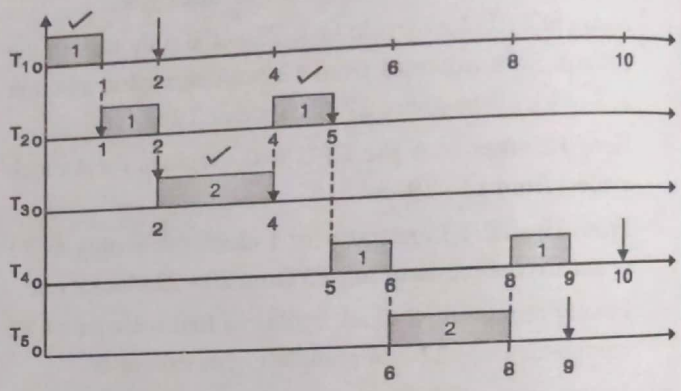

Fig. P. 1.2.4

Explanation

- As shown in Fig. P. 1.2.4 we have time instance on Y-axis and tasks shown on X-axis.
- We mark all the arrival time and deadlines for each and every task that are undergoing EDF scheduling.
- The main focus of EDF is to deadlines of tasks. Task having earliest deadline is given the highest priority rather than having lowest time period as in RM.
- At 0^{th} instance task T1 and T2 are in the system, priority is given to T1 as its deadline is earlier that T2.
- Schedule T1 form 0 - 1 for 1 clock cycle. T1 completes its execution.
- No new task is arriving in the system at 1^{st} clock cycle, only task in the system is T2 so it is scheduled form 1 - 2 for 1 clock cycle.
- At 2^{nd} clock cycle task T3 arrives with deadline 4. As $4 < 5$, it means T3 has earlier deadline than T2, so priority is given to T3 over T2.
- T3 executes for 2 clock cycles from 2 - 4. At 3^{rd} clock cycle new entry of task T4 is observed but its deadline is 10 so there is no urgency for T4.
- Schedule T2 from 4 - 5, and complete its execution. T4 is only task in the system so schedule it until new task

of higher priority i.e. earlier deadline arrives in the system.

- That happens at 6th clock cycle when T5 arrives with a deadline of 9 which is earlier than deadline for T4. So T5 is scheduled form 6 - 8 for 2 clock cycles and T4 from 8 - 9.
- In this way all the tasks follows its deadlines and EDF has run successfully.

Ex. 1.2.5

Try to schedule following set of periodic real time tasks using Earliest Deadline First Algorithm.

OR

Check whether following set of periodic real time tasks are schedulable using Earliest Deadline First algorithm or not. If not which task misses the deadline ?

Assume arrival time for all tasks as 0.

Task	Time Period	CPU Burst
T1	5	1
T2	15	4
T3	20	5

Soln. :

- Let's try to solve the above set of tasks:
- For periodic tasks time period acts as a deadline for separate instances of each task.
- So in given example considering arrival time for all tasks is 0, we have deadline of 5th clock cycle for T1, 15th clock cycle for T2 and 20th clock cycle for T3 but for first instance.
- For T1 2nd instance will have 10th clock cycle as deadline, 15th clock cycle for 3rd instance and so on.
- In the same way all the periodic tasks for scheduling will have their own deadlines independent of other tasks. Table P. 1.2.5 shows deadline for each instance of each task.

Table P. 1.2.5

Task ->	T1	T2	T3
1st Instance	5	15	20
2nd Instance	10	30	40
3rd Instance	15	45	60
4th Instance	20	60	80

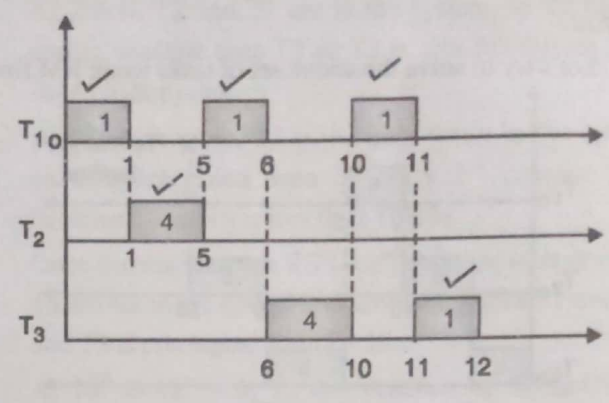

Fig. P. 1.2.5

Explanation

- As shown in Fig. P. 1.2.5 X-axis shows time and Y-axis shows tasks.
- At 0th instance all the tasks T1, T2 and T3 are available, so the task having earliest deadline is scheduled first i.e. T1 from 0 - 1 for 1 clock cycle.
- At 1 both T2 and T3 are in the system, so T2 having earlier deadline than T3 so T2 is scheduled from 1-5 for 4 clock cycles.
- At 5th clock cycle, T1's 2nd instance have deadline 10. Even if T3's 1st instance is yet to be scheduled scheduler picks task T1 (10) which has deadline earlier than 1st instance of T3 (20).
- So T1 is scheduled from 5 - 6 for 1 clock cycle.
- At 6th clock cycle T3 is scheduled from 6 - 10. At 10th clock cycle 3rd instance of T1 arrives with deadline 15. 15 is also the deadline for 1st instance of T2 but it is already met. So scheduler gives priority to T1 over T3 and schedules it.
- So from 10 - 11 T1 is scheduled. And remaining T3 is completed from 11 - 12. This scheduling can be continued for further events down the timeline.
- In this way all the tasks are meeting their deadlines at least once.

Ex. 1.2.6

Try to schedule following set of periodic real time tasks using Earliest Deadline First Algorithm.

OR

Check whether following set of periodic real time tasks are schedulable using Rate Monotonic algorithm or not. If not which task misses the deadline? Try to schedule the same using Earliest Deadline First algorithm and state the changes.

Assume arrival time for all tasks as 0.

Task	Time Period	CPU Burst
T1	25	6
T2	20	5
T3	15	5

1.3.2 System on Chip (SoC)

- As the name suggests, a SoC packs the whole system on a small chip.
- An integrated circuit that holds together several peripherals like memory (RAM/ROM), digital connections, analog connectivity, clock, GPIOs with a processor at its core is called as *System on Chip*.
- As shown in the Fig. 1.3.1 a SoC holds a processor/core that is the main part which handles the coordination of all the other components on the SoC and does the general instruction processing.
- It has a system memory which can be categorized in many types. Mostly embedded systems work with flash memory that handles code storage, RAM handles runtime data, ROM manages boot code which is the first of all to be executed when the system boots.
- SoC can have digital and analog peripherals as per the requirements.

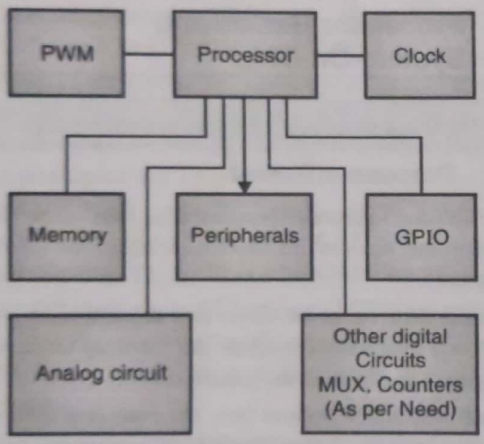

Fig. 1.3.2 : Basic SoC components

- Some of the digital peripherals are Timers, Pulse Width Modulator, Multiplexers, counters etc.
- Some of the analog peripherals are op-amps, ADCs, current voltage sources, etc.
- SoC also has General Purpose Input Output mechanisms in the form of pins so that external devices can be mounted on the SoC for application development.

Syllabus Topic : Introduction to ARM Processor and its Architecture

1.4 Introduction to ARM Processor and its Architecture

Q. Explain ARM SoC.

- ARM, which is an acronym for Advanced RISC Machine, is a popular 32 bit processor.
- Initially ARM was known as Acorn RISC machine as in 1985 Acorn Computer Ltd. developed a processor based on a design based on RISC principal published by few students in University of Berkley, USA.
- Based on this, Acorn developed their first ARM processor with less than 25000 transistors which operated at 6MHz in 1985.
- ARM 2 came up in 1987, after that ARM3, ARM4, ARM5 were developed but were not that popular.
- In between the timeline Acorn merged up with Apple Computers and VLSI Technology group in 1990 and renamed ARM as "Advanced RISC Machine".
- This group came up with ARM6, ARM7 and so on from 1991 onwards. This ARM design became very popular and was used in PDAs, IPods, Computer Hard Disks, etc.
- ARM6 introduced first embeddable processor in this series and ARM7 came up with multimedia support which made it a huge success till date.
- ARM7's feature made it so popular that as of 2011 over 90% of the market of handheld devices, mobile phones, small gaming devices etc. was being developed based on this 32 bit RISC system.

1.4.1 The ARM SoC

- As discussed above, a SoC packs the whole system on a small chip.
- The ARM SoC has ARM processor at its core processing unit which has all the power of computation, data movement, logical operations, control flow and decides how all the peripherals and other components are going to interact with each other.
- ARM in itself has a high computational capability, which can be used to a very high extent. With rich set of instructions and powerful register set makes ARM a microprocessor.
- The ARM core can be attached with basic peripherals and can be deployed to act as a Microcontroller Unit as well, where it performs the limited functionality. So ARM can also be treated as a microcontroller.
- The ARM as a company doesn't develop the entire chip in hardware format but it only distributes the design of the ARM processor to other company.
- Based on the license, the purchasing company develops their own ARM chips.
- The ARM distributes the design using Intellectual Property Rights (IPR) mechanisms. This IPR

distribution is done either in the form of Soft IPR or Hard IPR.

- In Hard IPR, the user gets the design as a black box format and no additional changes can be done to the given design by the user. In this only the layout of the design and necessary wirings is provided.
- In Soft IPR the user gets the design in the format of a Verilog code or Register Transfer Language (RTL) which describes the data flow at the register-transfer level which can be modified to a certain level by the user according to the need.
- When a fabricated chip has an ARM core and required peripherals to operate as a whole system it is termed as ARM SoC.

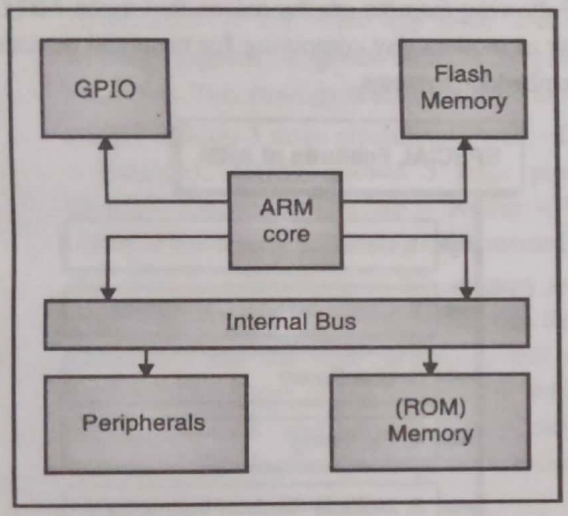

Fig.1.4.1 : ARM SoC

1.4.2 Advanced RISC Machine

ARM essentially follows the RISC architecture. But it also has a CISC property which we will see in further topics.

☞ Difference between RISC and CISC

Q. Differentiate between RISC and CISC.

RISC	CISC
Stands for Reduced Instruction Set Computing	Stands for Complex Instruction Set Computing
Main focus is on the software.	Main focus is on the hardware.
RISC have smaller set of instructions with few addressing modes.	CISC have larger set of instructions with many addressing modes.
RISC instructions are calculated in a single clock cycle.	Some CISC instructions are calculation requires multiple cycle.
Calculations are faster.	Calculations are slower.
Most instructions handle memory via registers.	Most instructions handle memory directly.
Instruction size is fixed.	Instruction size is variable.
Decoding the instructions is simpler.	Decoding the instructions is complex.
Execution time for a similar RICS program is less.	Execution time for a similar CICS program is more.
Disk space required for a similar program is less.	Disk space required for a similar program is more.

☞ Features that define RISC properties of ARM

(i) All instructions are of same size : 32 bit

(ii) Instructions is executed in one clock cycle

(iii) Only load and store instructions have direct access to the memory.

(iv) As number of transistors is less compared to the similar CISC architecture less hardware is needed.

(v) Less hardware requirement results in less die/chip size and power consumption is also lower.

1.4.3 ARM Features

Q. Describe advanced features of ARM.

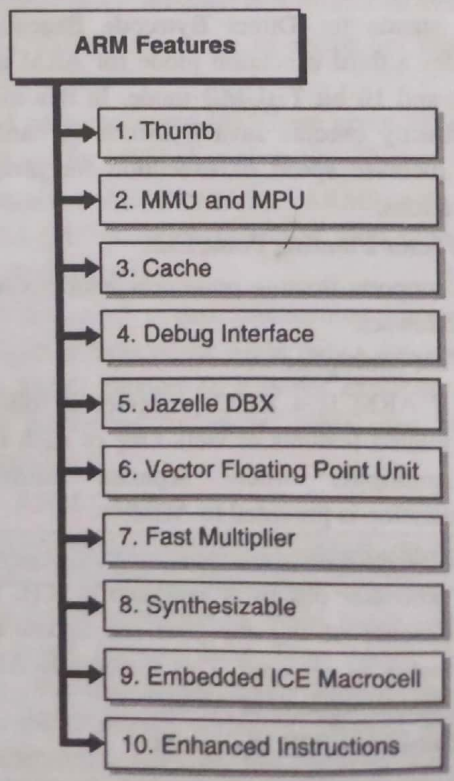

Fig. C1.4 : ARM Features

➜ **1. Thumb**

Basic ARM architecture works on 32 bit instruction set but ARM also provides a less powerful 16 bit instruction set. Those applications which don't need full powered 32 bit system of ARM can use this 16 bit THUMB mode. This increases the code density of the application.

➜ **2. MMU and MPU**

Memory Management Unit (MMU) and Memory Protection Unit (MPU) are the two features of ARM that handles memory related issues. MMU manages issues like which policy should be used paging or segmentation etc., while MPU manages memory out of bound or memory overflow errors. ARM gives us choice of which of these two or both should be implemented in ARM feature.

➜ **3. Cache**

ARM also supports cache memory for faster access of frequently used data. First ARM processor with cache was ARM3 with 4KB of on chip cache. ARM7 had 8 KB of cache which was later improvised.

➜ **4. Debug Interface**

ARM provides JTAG (Joint Test Action Group) support which expresses inbuilt functionality for debugging of external hardware that will be interfacing with ARM processor.

➜ **5. Jazelle DBX**

DBX stands for Direct Bytecode Execution. DBX provides a third execution mode for ARM along with 32 bit and 16 bit THUMB mode. In this mode ARM can directly execute Java Bytecode in hardware that helps increase speed of execution for java platform applications.

➜ **6. Vector Floating Point Unit**

ARM supports floating point computations with given instruction set.

➜ **7. Fast Multiplier**

Even if ARM is a RISC processor it follows some CISC related features as well. One of such features is Fast multiplier where separate hardware for multiplication is provided in ARM.

➜ **8. Synthesizable**

ARM processor design is available in RTL format as well. Because of this the user can update the given functionality as per need. This freedom in ARM is the synthesizable property of ARM.

➜ **9. Embedded ICE Macrocell**

Macrocell specifies a small prebuilt hardware that handles Macro level functionalities. Some ARM processors provide ICE i.e. In Circuit Emulator macrocells for adding breakpoints, setting watch points in the execution sequence of ARM instruction set.

This feature can be used for debugging by reading the status registers and control registers.

➜ **10. Enhanced Instructions**

Many of upcoming embedded systems works with complex DSP instructions. As per the requirement new DSP enhancements can be added to the ARM core seamlessly.

1.4.4 Features of ARM that makes it SPECIAL

Following features are the reason that made ARM very popular in modern day computing for handheld devices and most embedded systems.

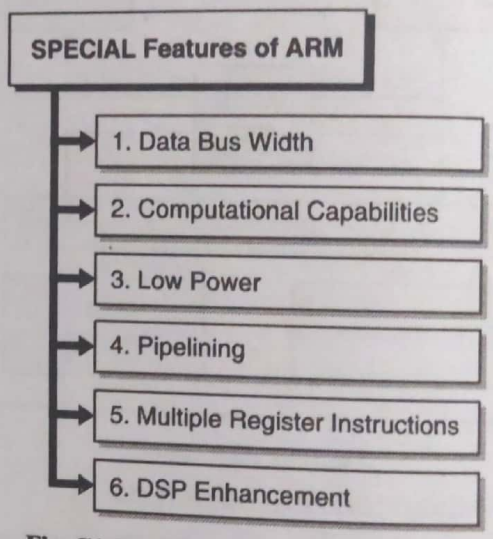

Fig. C1.5 : SPECIAL Features of ARM

➜ **1. Data Bus Width**

ARM has 32 bit bus width which means that ARM can process 32 bits of data for reading and writing purpose in one clock cycle. This capability makes the ARM application comparatively faster.

➜ **2. Computational Capabilities**

The design of the ARM processor is developed on RISC capabilities which make the execution of ARM application faster than its CISC counterpart. In addition to this some of the CISC functionality like fast multiplier is added to the ARM without hindering the speed of execution that much.

➜ **3. Low Power**

Because of the RISC design the hardware requirement is minimized which leads to lower power consumption for ARM based embedded systems. In addition to this working frequency for ARM varies between 60MHz to 1GHz. This low operating frequency of ARM makes it

a desirable microprocessor for power saver applications.

➜ **4. Pipelining**

- Pipelining is provided in a processor for faster execution of the instructions. In pipelining the operation of the instruction is divided into sub stages which are used to improve the performance of the processor. A simplest pipelining divides the instruction in "Fetch-Decode-Execute" stages.
- Because of this when one instruction is fetching the data the previous instruction which has been already fetched is decoding the data for other instruction and the instruction whose decoding is already done is in execution mode.
- So because of this there are three instructions working together at given moment of processor execution. This strategy is also available in ARM.
- ARM7 follows 3 stage pipelining (Fetch – Decode – Execute). ARM9 follows 5 stage pipelining (Fetch – Decode – Execute – Buffer – Write). ARM10 has 6 stage pipelining implemented in it.
- But there is a drawback to this strategy as when there is looping or branching in the code there is a possibility that the code execution will not go sequentially. Some of the instructions which are already fetched and decoded may never be executed. So higher penalty is associated as number of stages increases.

➜ **5. Multiple Register Instructions**

ARM processor follows ARM methodology so the instructions use registers as operands in instructions. This feature makes the execution of an instruction faster as there is no direct access to memory, which takes time. Apart from LOAD and STORE instructions all other instructions in ARM are register based instructions only.

➜ **6. DSP Enhancement**

Many of upcoming embedded systems works with complex DSP instructions. As per the requirement new DSP (Digital Signal Processing) enhancements can be added to the ARM core seamlessly. This feature deviates from the RISC methodology of ARM as such additional features require more hardware to the ARM which is a CISC philosophy. But wherever necessary such add-ons can be added to the ARM to make it application specific.

1.4.5 Naming Conventions for ARM

Q. Describe naming convention of ARM.

All the features of ARM are represented by the name and versions of the ARM processors. The developers and vendors use a specific naming convention that explains which features are available with the given ARM processor name.

ARM {x}{y}{z}{T}{D}{M}{I}{E}{J}{S}{F}

Symbol	Meaning
X	ARM Version Family (7,8,9,10,11...)
Y	MMU and/or MPU
Z	Cache Memory
T	Thumb Mode (16 bit)
D	JTAG Debug Interface
M	Fast Multiplier Availability
I	Embedded ICE Macrocell Availability
E	DSP Instruction Support (TDMI already assumed)
J	Jazelle DBX Support
S	Synthesizable Support
F	Vector Floating Point Unit Availability

☞ **Examples**

- **ARM7TDMI :** This naming specifies that ARM uses version 7 and has Thumb mode, JTAG debugger, Fast multiplier and Embedded ICE macrocell available.
- **ARM7TDMI- S :** Same as above but this version is synthesizable.
- ARM with cache and MMU are given the suffix as 26 or 36 and for MPU 46 is given.
- **Example :** ARM926EJ-S : ARM9 with cache and MMU, DSP processing enabled with support to DBX and synthesizable.
- Over the period of time this naming convention is also changed as most of the ARM now a day already comes with TDMI support so it is not explicitly mentioned. Also numbering for cache, MMU and MPU is also changed.

1.4.6 ARM Cortex

Q. What is ARM cortex series? Explain in details.

With the development of ARM series from ARM7 to ARM11 many features have been added to the ARM processor. But one of the major feature changes can be stated as the CORTEX series of ARMv7. ARM's cortex series differentiate between the types of applications the embedded system is going to implement.

1. Cortex A

A stands for Application domain. The high end application implemented with the modern embedded system uses this feature of ARM. Mostly used in modern mobile phones with android support and in video systems.

☞ **Example : ARMv7-A**

2. Cortex R

R stands for Real time domain. The high end application implemented for handling real time tasks and scheduling principals uses this feature of ARM. Mostly used in safety critical systems.

☞ **Example : ARMv7-R**

3. Cortex M

M stands for Microcontroller domain. When ARM is going to be implemented for limited kind of applications where the scope of functionalities is limited this profile is used.

☞ **Example : ARMv7-M**

1.4.7 ARM Programmer's Model

A programmer's model is while coding for ARM how a programmer views the processor. ARM provides a rich programming environment with its operating modes, register set and vast instruction set. Each instruction transforms the data in the specified registers which are visible in the current mode of operation.

1. Operating Modes

Q. Explain seven operating modes of ARM in short.

- ARM provides 7 operating modes for execution. Each of the operating modes has separate privileges. The least privileged mode is user mode but most of the programs execute in this mode.
- The most privileged mode is the system mode which is used for manipulating, controlling and monitoring the activities of the processor by the operating system. Rest of the modes are entered by the processor on occurrence of specific interrupts.

(i) User Mode

It is the least privileged mode where most of the user's program executes.

(ii) Fast Interrupt Request (FIQ) Mode

This mode is entered by the processor on occurrence of high priority interrupts. Tasks in this mode need to be served before tasks in IRQ.

(iii) Interrupt Request (IRQ) Mode

This mode is entered by the processor on occurrence of low priority interrupts. Tasks in this mode are served after tasks in FIQ.

(iv) Supervisor Mode

Supervisor mode is a privileged mode that makes sure that the user level program doesn't gain access to privileged modes by accident. The user level code needs to perform supervisor mode functions to get access to system mode. This mode defines a software interrupt function which makes sure user mode program doesn't directly gain access to other privileged modes.

(v) Abort Mode

When there is ambiguous memory access like accessing memory out of bound or trying to change data of unknown memory address memory violations occurs and system might halt. To avoid this scenarios memory violations are handled by this mode.

(vi) Undef Mode

When user level program tries to execute an instruction which is not defined in the instruction set of ARM this mode changes avoid the program failure and raises error by putting program in Undef mode.

(vii) System Mode

This is the highest privileged mode where the operating system manipulates and monitors the status of the ARM core with the help of status registers and control registers.

All the above operating modes in ARM have their own registers as well as shared registers. We will see the register sets available for each one of them in the next section.

2. Register Set

Q. Explain ARM register set.

- ARM provides 37 registers in all, each one of them are 32 bit long. All the 37 registers are distributed amongst 7 working modes of ARM, some of them are shared while some are independent.
- But while executing in any mode each mode works with 12 general purpose registers, 1 Link Register, 1 Stack pointer and 1 Program Counter with Current Program Status Register and Saved Program Status Register backing up the operating mode.
- As shown in Fig. 1.4.2, registers are distributed amongst operating modes.
- All modes have a common ***R15*** Program Counter (PC) which keeps track of the address of instruction which is going to be executing next.
- ***R13*** is Stack Pointer (SP) and ***R14*** is Link Register (LR). SP is used to point to top of the stack that stores data while making a function call and LR stores the address of the location to jump when interrupt is handled by the OS.

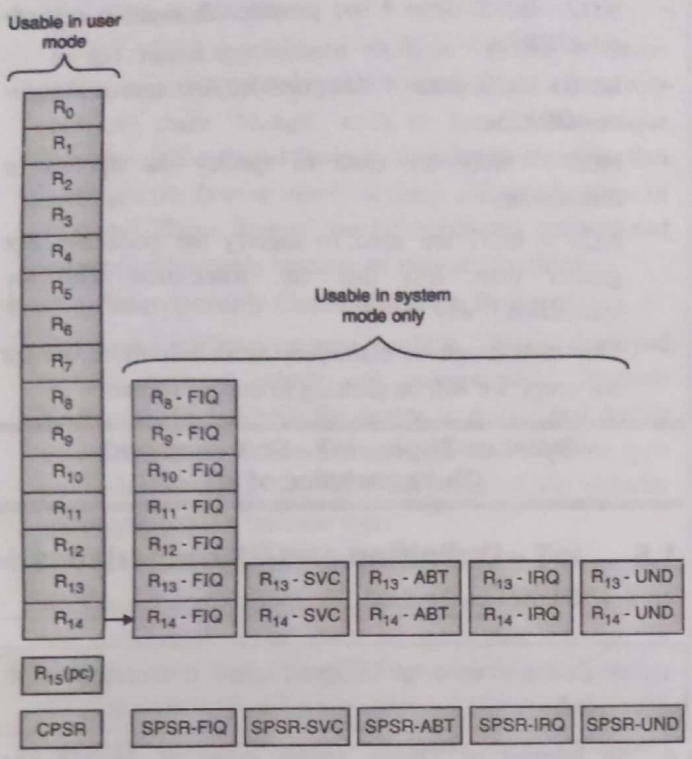

Fig. 1.4.2 : ARM Register set in various Operating Modes

- R13 and R14 are the same for User mode and System Mode for all other modes, ARM provides independent registers like R13_FIQ, R13_IRQ and so on for each mode.
- ARM also provides Current Program Status Register (CPSR) as the name suggests its stores current status of execution. We'll see CPSR in details in next section.
- To store CPSR while mode switching ARM provides Saved Program Status Register (SPSR) for each mode except for user and system mode.
- ***R0*** to ***R12*** are General Purpose Registers which are shared amongst all the seven working modes. These set of 12 registers are commonly used to store program data in all the modes except for FIQ.
- When the operating mode is switched for example from User mode to IRQ, contents of R0 to R12 is stored away and R0 to R12 registers are now used for IRQ execution mode.
- This happens for all modes IRQ to Abort, Abort to Undef and other possibilities as well but not for FIQ. What different happens in FIQ?
- As FIQ needs to serve the interrupts faster than IRQ or any other modes, ARM provides FIQ with independent 5 General Purpose Registers (R8-R12).
- Some time is needed while switching modes, so to avoid this time waste and serve the interrupt in faster way FIQ uses its independent R8 to R14 registers and only R0 to R7 registers are transferred for this execution.
- This independent set of registers saves time for FIQ which results in faster execution of interrupts in FIQ mode.

CPSR Format

Q. Describe bit configuration of CPSR of ARM.

- CPSR is a 32 bit register that keeps track of the current status of the program execution.
- Following is the bit format for CPSR

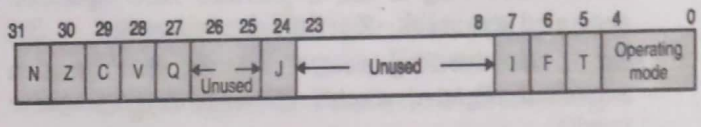

Fig. 1.4.3 : CPSR bit format

Bit Number	Code	Explanation
0 - 4	Mode	Operating mode of ARM
5	T	Thumb Bit
6	F	FIQ Bit
7	I	IRQ Bit
8 – 23, 25 - 26	Unused	For Future Purpose
24	J	Jazelle State
27	Q	Sticky Overflow Bit
28	V	Overflow Flag
29	C	Carry Flag
30	Z	Zero Flag
31	N	Negative Flag

- Bit Number 0 - 4 defines operating mode of ARM.

Bit Sequence [4:0]	Operating Mode
10000	User
10001	FIQ
10010	IRQ
10011	Supervisor
10111	Abort
11011	Undef
11111	System

- Bit 5 is used for defining Thumb Mode. If T = 0 thumb mode is active if T=1 ARM mode is active.
- Bit 6 define FIQ mode. If F=1 disables FIQ.
- Bit 7 define IRQ mode. If I=1 disables IRQ.

- Bit 8 - 23 and 25 - 26 are unused right now and can be used in future.
- Bit 24 defines jezelle state of ARM. If J = 1 then Jezelle state is on.
- Bit 28 is Overflow flag which is set if the previous ALU operation generated overflow in the sign bit.
- Bit 27 defines sticky overflow flag. It is same as Overflow flag which is once set stays the same until reset manually by the programmer.
- Bit 29 is Carry flag, which is set, if previous ALU operation generated carry.
- Bit 30 Zero Flag is set, if previous ALU operation generated zero result. (Z = 1 means zero result)
- Bit 31 Negative Flag is set, if previous ALU operation generated negative results. (N = 1 means negative result)

3. Instruction Set

Q. Explain instruction format of ARM.

- ARM provides a vast instruction set which can be categorized in following set of instructions:
  (i) Data Processing Instructions
  (ii) Branch Instructions
  (iii) Load / Store Instructions
  (iv) Status Register Access Instructions
  General format for ARM instruction is:
  Label <opcode> <opr1> <opr2> <opr3> ;Comment
- Most ARM Instructions uses three registers in an instruction in the operand fields.

Example

Addition:
ADD R3, R1, R2 ; R3 = R1 + R2
Addition with carry:
ADC R3, R1, R2 ; R3 = R1 + R2 + Carry
Subtraction
SUB R3, R1, R2 ; R3 = R1 + R2
Subtraction through carry:
SUBC R3, R1, R2 ; R3 = R1 + R2 +Carry

- ***<OPR1>*** is mostly the result/destination operand, and ***<OPR2>*** and ***<OPR3>*** are source registers.
- Bit format for a general Instruction in ARM:
- DIAGRAM for INSTRUCTION FORMAT

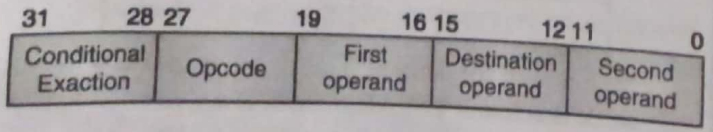

Fig. 1.4.4 : Generation ARTI instruction bit format

- As shown in Fig. 1.4.4, bit0 - bit11, these 12 bits provides second source register or ***<OPR3>***.
- bit12 - bit15, these 4 bits provides destination register or ***<OPR1>***.
- bit16 - bit19, these 4 bits provides first source register or ***<OPR2>***.
- bit20 – bit27 are used to specify the opcode or instruction.
- bit28 – bit31 are used to specify the condition like greater than, less that etc. associated with the instruction if any.
- This format can be elaborated in details more but for our scope we will be sticking to this bit format.

Syllabus Topic : IoT - Definition and Characteristics of IoT

1.5 IoT - Definition and Characteristics of IoT

Q. Define Internet of Things. Explain characteristics of IoT.

- In Internet of Things, things refers to devices like Bluetooth connected headsets, thermostats, utility meters temperature reader, sensors, actuators etc. which can sense some parameters. In IoT these "things" are connected together to generate some meaningful results. In IoT each of these "things" has independent identities and can be connected over a network for data sharing.

Definition

IoT is a global network of "things" i.e. physical and virtual devices having separate identity to each one of them, which can be connected via a vast network to share data and process it into meaningful information.

1.5.1 Characteristics of IoT

IoT based application and services can be distinguished from some well-defined characteristics.

These are as follows :

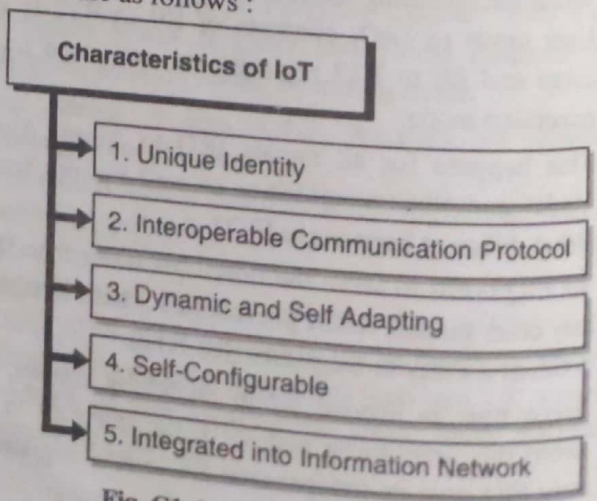

Fig. C1.6 : Characteristics of IoT

1. Unique Identity

- In IoT based applications multiple "things" of same utility are usually deployed, to coordinate effectively between these "things" each of these have unique identity. IoT systems have an intelligent interface that manages the system based on these unique identities of "things". These "things" can be monitored, updated and controlled remotely because of unique identities.

2. Interoperable Communication Protocol

- IoT based applications have multiple "things" deployed in the network which can communicate. Various protocols are available for device to device and device to infrastructure communication (devices of same type or different type). We will see some of the popular protocols used in the next topic.

3. Dynamic and Self Adapting

- IoT based applications and devices are made aware of the environment in such a way that change in parameters of their surroundings changes the context and the IoT applications can accommodate such changes based on user's context. Such intelligent systems can be designed and are called as self-adapting IoT applications.

Example

- A surveillance systems has cameras with capability of infra-red night mode and normal mode. Based on the surroundings, environment camera should be able to switch between normal mode and night mode.
- Doing so a central camera can also send notification to other cameras about doing the same. Changing the resolution based on light availability is one more dynamic behavior of cameras.

4. Self-Configurable

- Most of the small IoT devices such as sensors, actuators etc. once deployed has very less direct user interaction. Due to this many of such devices are capable of fetching the latest software updates, setup basic networking and status checks themselves or with very less user intervention.
- Such devices work together to generate results.

Example : Weather Monitoring System.

5. Integrated into Information Network

- IoT devices are typically configured in such a way that they can communicate between other devices in the IoT based environment to create an Information Network. IoT devices can describe themselves using the unique identity to other devices or to a network and are dynamically discovered in the network by other devices and/or network.

Example

- In weather monitoring system, multiple sensors can communicate to other sensors or a central node in a network. This capability makes the IoT applications "smart".
- This data from large number of devices are collectively merged together to generate meaningful information with the help of IoT infrastructure.

Syllabus Topic : IoT - Vision, Emerging Trends, Economic Significance, Technical Building Blocks

1.6 Internet of Things - Vision, Emerging Trends, Economic Significance, Technical Building Blocks

- In the following section we will see how the development in IoT has changed our day to day life. IoT vision tries to describe a short case study in IoT which covers different aspects of IoT. Emerging trends discusses about features of latest IoT based systems that make the development and deployment in IoT more noteworthy.
- Economic significance states the economic growth happening because of IoT in the world, and technical building blocks discusses the different aspects of IoT application development.

1.6.1 Internet of Things - Vision

Q. Explain vision of Internet of Things with a suitable case study.

- IoT refer to a vast network of smart devices that are connected to each other and constantly communicating over the network producing huge data that is processed into information. IoT contains smart devices referred as things which use various communication protocols like TCP, UDP, IPv4, IPv6, HTTP, CoAP and a lot more in different layer to communicate between each other. Along with these protocols technologies like RFID, NFC, ZigBee and Bluetooth makes significant mark in IoT based applications.
- A simple case study describes a vision of IoT. Following case study discusses a trip of Rajiv a businessman who uses his mobile phones and other "things" to stay notified with his schedule, surroundings and home even when he is out of town. On one of his business trip Rajiv needs to travel to Japan for the first time.
- Rajiv is travelling via plane and he gets flight notification two days prior to his travel stating his schedule. On the day of the journey, he gets the

notification stating his flight is on time. This is **notification service** by the travel agency.

- Once he enters the airport "things" at the airport notifies available services like routes to the departure gate, procedure for security check-up, available shops and food counters, current status of his flight etc. on his smartphone.
- This is **detection of public "things" and services** via notification. This is common to all passengers arriving at the airport and not specific to Rajiv. Rajiv checks the security check-up details and finds that it generally takes 20 minutes for clearance.
- This is **use of public services**. At the security counter Rajiv's phone beeps notifying delay in the flight due to bad weather and lunch coupons are made available from the flight. This notification is specific to Rajiv or passengers of that particular flight.
- This was possible because Rajiv's information was available to the XYZ airlines. These are **notification based on personal information**.
- Rajiv reaches Japan, totally unknown to him; he searches for the best route hotel his company booked for him. Maps are made available to him via route/map assistance at the airport.
- His device/smartphone accesses this information and selects the nearest bus stop. On arriving at the bus stop mark enters address of his hotel and finds he has 4 buses in next 5 minutes for his hotel. This is detection of different service options.
- Rajiv selects one option and payment request is popped on his smartphone which he approves. This is **access control and payment gateway permission** approved by Rajiv. On arriving at the hotel Rajiv issues a new device from local Internet Service Provider so he can access other "things". ISP provides the device by checking valid ID from Rajiv.
- This ID is linked with Rajiv's company which is also executing in Japan and is linked together with Indian database. This is **distributed policy management**. In a while a new message pop's up stating an old friend Prakash is in nearby area. This is an **update from location based services**. Same notification goes to Prakash's smartphone.
- Both of them meet. Now Rajiv want's to roam around the city for a while but Prakash is busy. So Prakash manages to rent out a car on his local insurance credentials for Rajiv. This is **policy negotiation**. Now Rajiv is driving a car based on Prakash's credentials. The car has a GPS sensor which keeps track of where the vehicle is moving.
- This can be considered as **data management by sensors**. Based on the distance driven by Rajiv bill amount will be calculated and deducted from Prakash's credit card as his details are used for renting the car. This is **context management**. At one point Rajiv's feels uneasy his body sensor shows his blood pressure has dropped suddenly.
- This is health **monitoring** by body sensors. Rajiv's device records this data without fail and forwarded to his family doctor. This is regular data synching and processing by sensors and smartphone application.
- Rajiv visits a local doctor and just transfers his health records to local doctors and from the medical history local doctors prescribe perfect medications for him. This is **data sharing and data processing based on previously stored** health records.
- Back at home Rajiv's mailbox is full and he receives a pop up showing mailbox is full of letters. This is **remote sensor monitoring**. His garden is also regularly watered while he is out of town via IoT based watering system which waters the plants if soil moisture level drops below certain threshold. This process can be **monitored remotely** all the way from Japan.
- Above case study is a short view of what a IoT based systems can do. Various steps and processes are involved in the implementation perspective of IoT based systems which will be covered in upcoming sections.

1.6.2 Internet of Things - Emerging Trends

Q. What are the emerging trends of Internet of Things ?

- As seen in previous section IoT vision can be illustrated with the help of "things" that has sensing, actuating, communication and computation capability. IoT is aiding many sector to automate the day to day work in a very fluent way. Some of the examples can be readily available media like audio, video images at fingertips, home automation, and health care systems so on. For any IoT based system there can be three basic stakeholders which are users, providers and society. Users are the ones who access the facilities of the IoT based systems via devices or softwares. Providers creates and maintains the IoT based systems and provides this utility to the users. Society which holds the sensors in some cases also provides the legal framework of how this monitoring and sensing should be done. These stakeholders hand in hand provides and uses IoT applications.

- Some of the emerging trends in development and deployment of IoT based applications are stated below:

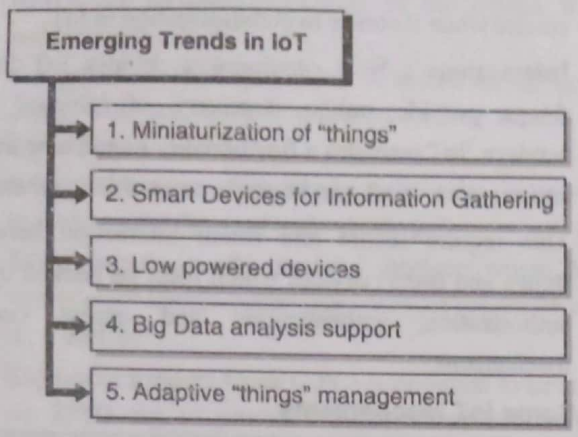

Fig.C1.7 : Emerging Trends in IoT

➜ **1. Miniaturization of "things"**

- Smart devices i.e. things with sensing, computing and communication capability in IoT are referred as "things", are getting smaller in size. Now a day due to advancement in VLSI techniques, microelectronics technologies it has become easier to create tiny things.
- This creates an opportunity to adjust these things on any physical gears as well as in a living body as well.

➜ **2. Smart Devices for Information Gathering**

- Users are actively using smart devices like mobile phones, tabs, laptops etc. to make use of IoT applications via apps or softwares.
- Having technologies like camera, Bluetooth, WiFi, NFC etc. enables these devices to sense and read data sensed by the things in IoT applications. For example, wearable gadgets senses various body functions which are displayed on mobile device, barcodes can be read by using phones' camera etc.

➜ **3. Low powered devices**

- IoT applications make use of things which are low powered devices so energy consumption is one of the main focus while using such devices.
- To provide necessary power to such devices solar cells, energy generation from shakes and wind can be used.

➜ **4. Big Data analysis support**

- Things generate data which can be stored and processed locally. But in some cases where data generated is huge popularly termed as Big Data methods are required to store and process such data.
- Availability of cloud and its fluent communication with IoT "things" provides support for Big Data.

➜ **5. Adaptive "things" management**

- Due to increasing demands and scalability point of view the "things" are increasing in number day by day. Billions of nodes will be working for various IoT bases systems one day. So for adding new devices and updating previous ones self-monitoring and self-configurable things are needed.
- Cooperative communication and concurrent processing along with efficient business logic is required to achieve such capability.

1.6.3 Internet of Things - Economic Significance

Q. Explain economic significance of Internet of Things.

Due to the usability and wide range of applications, IoT is becoming point of interest to Industries, Government and Individuals as well. Considering business point of view IoT can provide wide range of opportunities in retail marketing, logistics, manufacturing, energy field etc. Along with the business opportunities it also creates jobs that require skill and knowledge. In view of this IoT's economic significance can be shortened as follows:

(i) It will create new business opportunities with higher accuracy, efficiency, and mobility.

(ii) It will help automate tedious jobs with less or no supervision.

(iii) Improved quality of services which will facilitate easier and improved lifestyles.

(iv) Easier tracking and monitoring of products and companies resources.

(v) The business processes can be improved in terms of performance and scalability.

(vi) It will provide cost-effective and efficient solutions to the well known problems.

1.6.4 internet of Things - Technical Building Blocks

Q. Explain technical building blocks of Internet of Things.

- While developing any IoT based application developers have to focus on three layers which are things that specifies the hardware level, middleware that manages the connectivity so we'll call it the communication layer and Services and access to the services which is application layer. Let's discuss this in more detail:

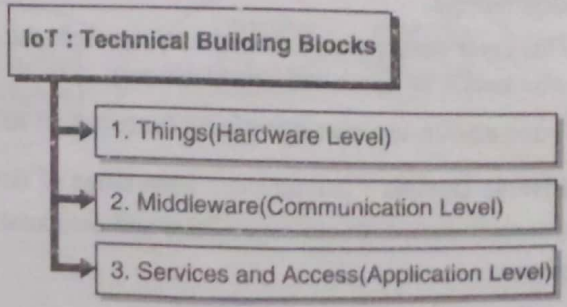

Fig. C1.8 : IoT : Technical Building Blocks

➜ **1. Things (Hardware Level)**

- Things in IoT are a device that has sensing, actuating, communication and computation capability.
- Things can be real or virtual nodes and has varied computational capability.
- Some things are able to process complex data while some are only able to sense a single entity.
- This level is used for sensing data, storing it, communicating and processing it whenever needed.
- One main feature of things is, each "thing" has a unique identity and an identifier that separate out individual thing in a wide spread network of devices. IPv4 and IPv6 are frequently used for this and other mechanisms are also available.

➜ **2. Middleware (Communication Level)**

- There are plenty of things deployed in an IoT based application. Some of them are of the same type while many of them are not alike.
- IoT application creates a network of such heterogeneous devices.
- For an IoT application to run effectively these devices need to communicate with each other.
- Device to device communication in homogeneous devices is easy as addressing is usually the same for homogeneous devices.
- While communicating between heterogeneous devices this level has to create business logic to map the addresses of heterogeneous devices so as they can send and receive data amongst them.
- Middleware has to take care of addressing, data formats while providing medium for storage and tools for performing computations.

➜ **3. Services and Access (Application Layer)**

- This layer manages the information coming from the middleware.
- The services provided by the IoT based systems are accessed by users via this level.
- The data gathered from middleware needs to be processed and made available to the user as per requirement.
- This layer provides a mechanism to represent the data, information in a user understandable way.

☞ **From above we can conclude features of IoT**

(i) **Diverse Devices** : IoT includes wide range of devices (sensors) with different capabilities and computational power.

(ii) **Unique Identity** : Each of the "things" has a unique identity and identifier. Digital identity of a device is crucial when it comes to communication in IoT.

(iii) **Interactions** : With omnipresent devices IoT based things provide public discovery of "things" and services. IoT provides a functionality to add new things and services which can be made accessible to others.

This require secure and fluent interaction between things and users' devices which must be backed up by authorization, authentication and access control mechanisms.

☞ **Some IoT Mechanisms**

Q. Explain following IoT mechanisms : RFID, EPCIS, ONS.

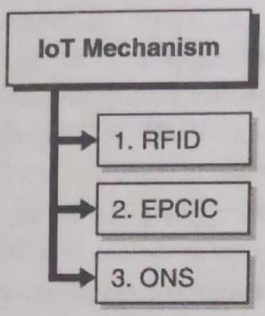

Fig. C1.9 : IoT Mechanism

➜ **1. RFID**

- Radio Frequency Identification is the wireless identification service which is used to bind device/"things" with unique serial number encoded within a tag.

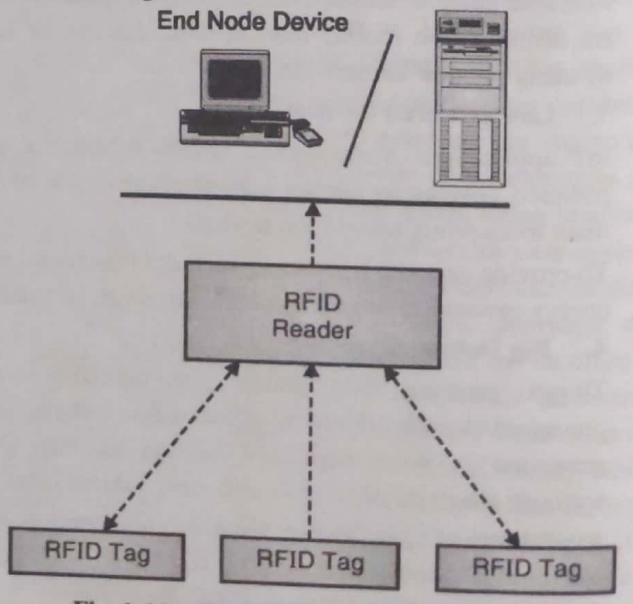

Fig. 1.6.1 : RFID Tags and Communication

- RFID tags can be recognized using RFID readers even without direct line of sight or direct contact.

- The communication between special tag and the reader happens via radio frequency.
- RFIDs are attached physically to the device which needs identification.
- Some additional data can also be stored on the device along with the serial number according to the size limitations.
- RFIDs can work on read only, read-write, read-write-re-write as per the need and encoding.
- RFIDs are frequently used for products where direct line of sight identification like barcodes cannot be used.

➜ 2. EPCIC

- Electronic Product Code (EPC) is designed to be stored on RFID tag to provide unique identification for a product.

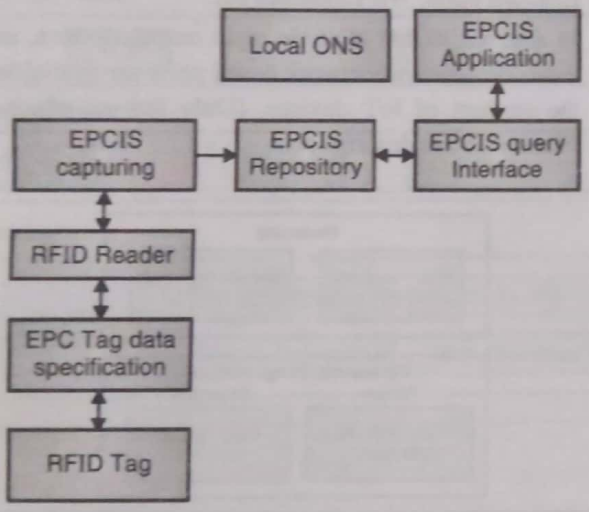

Fig. 1.6.2 : General working in EPCIS

- Electronic Product Code Information Service (EPCIS) is a network service that offers Electronic.
- Product Code (EPC) and RFID integration. **EPCIS capture interface** recognizes the RFID data and provides it to the repository and applications and other EPCIS nodes.
- **EPCIS repository** is a storage that provides raw RFID data and other information associated with it. **EPCIS query interface** gives user the application level access to the system.

➜ 3. ONS

- Object Name Service is a mechanism to find out information about a product and services from EPC. ONS is a request that returns the list of all network accessible services related to the requested EPC. ONS takes EPC as input parameter and provides address of EPCIS as output in the form of URL.
- ONS doesn't contain the actual data of EPC but the addresses of services where actual data resides. ONS acts as a directory that routes the EPC information from requester to destination node.
- ONS system has root ONS and local ONSs. Root ONS keeps the IP addresses of local ONSs and local ONSs keeps addresses of EPCIS. When the client sends the request with particular EPC as input query, root ONS returns the IP address of local ONS where the EPC can be found.
- In next step client send query to the local ONS which returns the IP address of EPCIS to the client. The working of ONS is similar to DNS in networking.

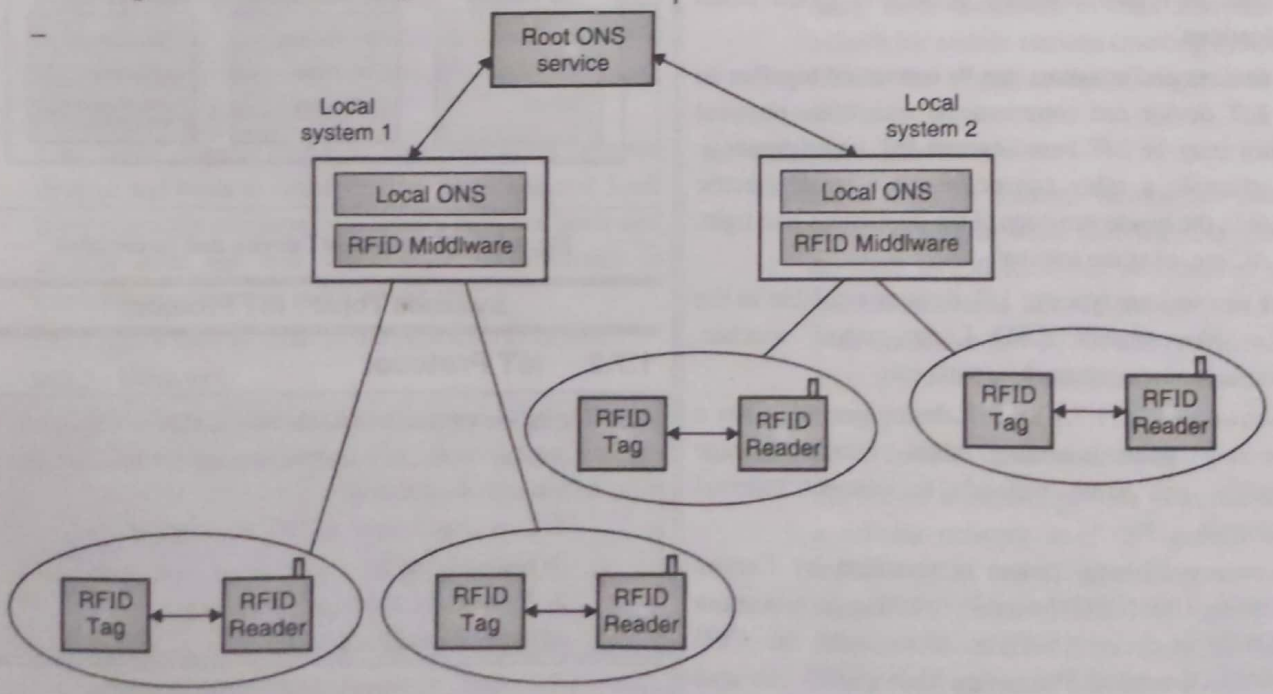

Fig. 1.6.3 : ONS and RFID Communication

Syllabus Topic : Physical Design of IoT

1.7 Physical Design of IoT

- Physical design of IoT focuses on how the IoT devices are built internally and how they communicate with each other. In this section we will discuss in details what is "thing" in IoT and what are the protocols used in various layers of networking.

Syllabus Topic : Things of IoT

1.7.1 Things of IoT / IoT Devices

Q. What is meant by "thing" in IoT ? Explain in details.

- As discussed earlier "things" in IoT refers to an IoT device with a unique identity that has a capability of sensing, monitoring and actuating.
- IoT devices can send and receive data directly or indirectly and also has a ability to process it to information.
- This processing can be done locally, remotely, using a centralized servers or can use a cloud based application platform for the same.
- IoT device can process the data as per its memory, processor, speed, communication etc. If processing requires higher specification IoT infrastructure can provide the same.
- IoT devices can pull together data from attached or on-board sensors viz. IR sensors, temperature, humidity sensors etc. and can process data locally or can forward the data to higher powered devices or cloud based applications.
- IoT devices and actuators can be connected together so that IoT device can communicate with other physical entities may be IoT based or non-IoT based systems. For example a relay connection to a small electric circuit in the house can control the appliances like light, fan, AC etc. over the internet.
- There are various types of IoT devices available in the market like sensors, LED Lights, smart watches, wearable sensors, automobile parts etc.
- As shown in Fig. 1.7.1 an IoT device generally has a processing power, sensing ability, some storage capability and some methods to connect external peripherals.
- The core processing power is provided by Central Processing Unit (CPU) generally working on maximum frequency of 1 to 1.5 GHz. Along with the CPU additional Graphical Processing Unit (GPU) can also be attached as per applications requirement.
- IoT based applications uses embedded devices that uses Flash type of memory, Embedded Multimedia Memory Cards (eMMC), Secondary Storage, NAND, NOR type of memory is given to IoT devices.
- IoT based devices needs to communicate and interact with external devices may be of same type or different type for this such devices provides connectivity such as Input / Output interfaces with General Purpose Input Output (GPIO) for connection of sensors, actuators etc.
- Universal Asynchronous Receiver-Transmitter (UART) for asynchronous serial communication to devices like GPS, Bluetooth etc.
- Audio Video interfaces with 3.5 mm jack port, HDMI support is also available.
- Network connectivity is provided to IoT devices using Ethernet ports, WiFi modules etc.
- In addition to this multiple input output devices, audio video devices and network based ports are available for the support of IoT devices. (Only few of which are listed in above description.)

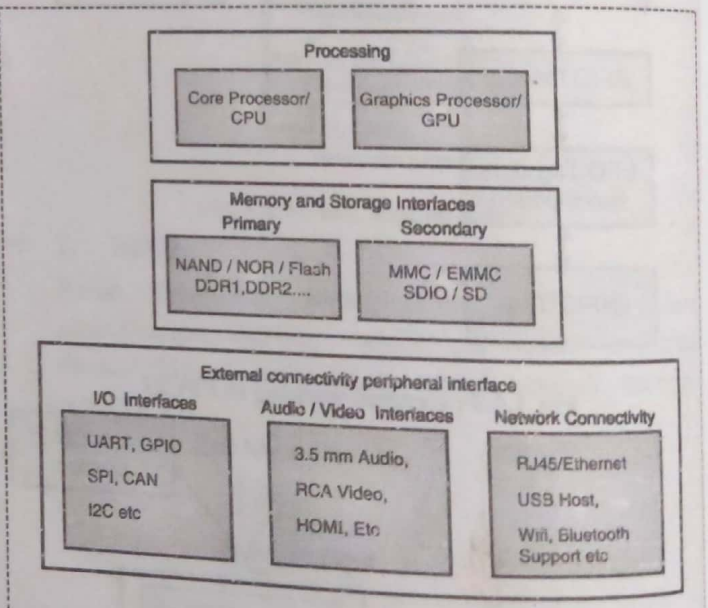

Fig. 1.7.1 : A general IoT device and its contains

Syllabus Topic : IoT Protocol

1.7.2 IoT Protocol

Q. Explain various protocols needed for IoT systems.

Q. Explain various protocols needed for IoT systems in the following layers:

1. Link Layer
2. Network Layer
3. Transport Layer
4. Application Layer

- As discussed earlier IoT devices communicate with each other and central nodes seamlessly throughout the internet network. For this communication to happen smoothly various protocols are available in various layers of Internet Protocol suit. Let's see some of the well-known protocols used by IoT based application in various layers like Link Layer, Network Layer, Transport Layer and Application Layer. We will discuss the protocols in short in the following section.

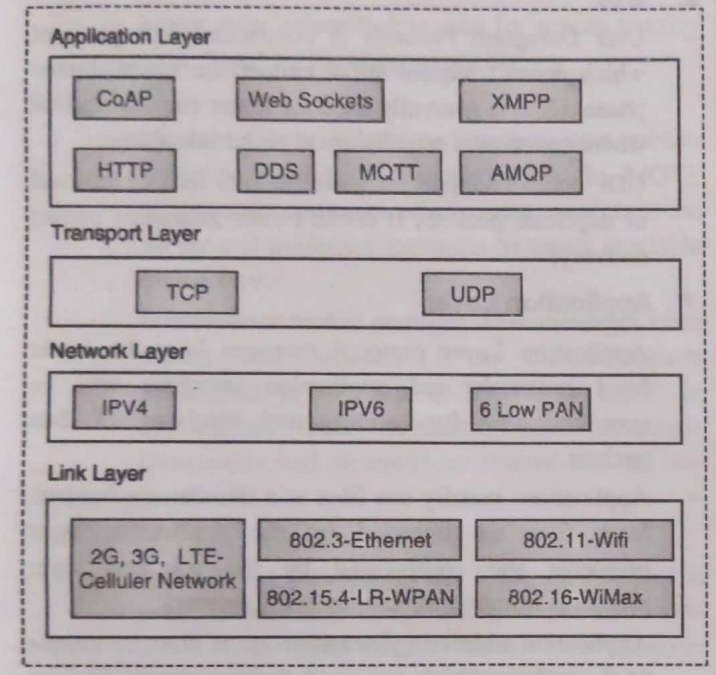

Fig. 1.7.2

Link Layer

- Link layer manages physical level of data communication in the network for example communication over copper wire, twisted pair cable, radio waves or coaxial cable etc.
- Link layer protocol manages communication between devices and hosts to which devices are connected. Link layer protocols communicate locally between hosts and devices over the link layer where connectivity is available via cables or radio waves etc.
- Let's take a look at some of the protocols in details:

1. 802.3 - Ethernet

- IEEE 802.3 contains various wired Ethernet based connection mediums. The shared medium in Ethernet can be performed using twisted pair cable, coaxial cable or optic fibers.
- Data sent by source can be received by all the connected devices based on its capabilities.
- Data rate of 10Mb/S to 40Gb/S and more can be achieved using these standards. Table 1.7.1 shows 802.3 protocol specifications:

Table 1.7.1

Protocol Standard	Protocol Use
802.3	10BASE5
802.3.i	10BASE-T
802.3.j	10BASE-F
802.3ae	10 Gbits/s over fiber network

2. 802.11 – WiFi

- IEEE 802.3 contains various wireless local area network based connection mediums (WLAN).
- Data rate of 1Mb/S to 6.75Gb/S can be achieved using these standards. Table 1.7.2 shows 802.3 protocol specifications :

Table 1.7.2

Protocol Standard	Protocol Operation Frequency Range
802.11a	5 GHz band
802.11g and 802.11g	2.4 GHz band
802.11n	2.4-5 GHz band
802.11ac	5 GHz band
802.11ad	60 GHz band

3. 802.16 – WiMax

- IEEE 802.16 contains various wireless broadband connection mediums with extensive use of Link Layer (WiMax). WiMAX operates similar to WiFi, but has higher speed, larger distances range and greater number of users.
- Data rate of 1.5 Mb/S to 1 Gb/S can be achieved using these standards. WiMax also archives 100 Mbit/S for mobile stations (moving systems) and 1 Gbit/S for fixed stations.(Larger compared to WiFi)

4. 802.15.4 – LR-WPAN

- IEEE 802.15.4 contains various low rate wireless personal area network (LR-WPAN). Data rate of 40 Kb/S to 250 Kb/S can be achieved using LR-WPAN standards. LR-WPAN are used for devices with limited or low power and provides low-speed and low-cost communication which consumes lesser power.

5. 2G/3G/4G Mobile Communication

- 2G/3G/4G standards provide communication over a cellular network level. 2nd generation i.e. 2G includes GSM and CDMA, 3rd generation i.e. 3G uses UMTS and CDMA-1000, while currently famous 4G uses LTE. Data rate of 9.6 Kb/S for 2G devices and up to 100 Mb/S for 4G devices can be achieved using these standards.

Network/Internet Layer

Network layer handles sending and receiving the message packets over a network from source to destination. This layer handles addressing and routing of packets over the network. Host addressing schemes is managed by IP addresses.

1. **IPv4**
   - Internet Protocol version 4 is the most used addressing scheme used for unique identification of the devices over the network.
   - IPv4 has 32 bit header length which provides 2^{32} addresses distributed in classes (A – E). IPv4 is a connection less protocol which doesn't guarantee packet delivery.
2. **IPv6**
   - Internet Protocol version 6 is the successor of IPv4. IPv6 has 128 bit header length which provides 2^{128} addresses which are far more than IPv4 in comparison.IPv6 provides location system and identification for computers in the network.
   - IPv4 and IPv6 are not interoperable by default but some development is in process where direct communication between IPv4 and IPv6 is possible via middle level logic.
3. **6LoWPAN**
   - IPv6 over Low power Personal Area Network is designed for lower powered devices where communication overhead should be less.
   - Devices with low processing and communication capabilities use this protocol in network layer. 6LoWPAN works over frequency range of 2.4GHz and has data transfer rate of 250 Kb/S. 6LoWPAN in network layer and 802.15.4 in link layer works together and describes compression mechanism for IPv6 packets for communication in LR-WPAN.

Transport Layer

- Transport layer manages end to end communication irrespective of the underlying network. Transport layer defines communication to be connection oriented or connectionless.
- Added features such as error correction, segmentation, congestion control and flow control are also managed by transport layer.

1. **TCP**
   - Transmission Control Protocol i.e. TCP is a connection oriented protocol used by web browsers, email programs, for file transfer etc. TCP ensures that the sent packets are received are the destination if not resending of packets is also provided.
   - TCP provides error detection meaning duplicate packets can be removed. Flow control manages sending data rate is not too higher than the receivers processing capacity. Congestion control avoids jamming of packets in the TCP based communication.
2. **UDP**
   - User Datagram Protocol is connection less protocol which doesn't require initial connection establishment phase. UDP is generally used for faster communication where connection establishment phase takes time.
   - UDP doesn't support retransmission of data or removal of duplicate packets. It doesn't even guarantee packet delivery.

Application Layer

- Application Layer protocol manages how the lower level protocols and application interface will be communicating for sending and receiving of data packets.
- Applications usually use files as a transfer mechanism. Such files are encoded by the application layer protocols and compressed by the transport layer protocols for communication in the network.
- Application addressing/identification is done by unique port numbers which are used for process to process communication. Let's see some well-known application layer protocols in short:

1. **HTTP**
   - Hyper Text Transfer Protocol is the base for WWW i.e. World Wide Web. HTTP protocol has commands which are used for data communication by applications. GET, PUT, POST, DELETE etc. are some of such commands.
   - HTTP is a stateless protocol and uses request response communication model. Universal Resource Identifiers i.e. URI's are used for maintaining HTTP resources generally over a server. Clients uses these URIs to access HTTP resources.
2. **CoAP**
   - Constrained Application Protocol i.e. CoAP used for machine to machine (M2M) communication where the applications are meant for small embedded devices, controlled environment and constrained networks.
   - Similar to HTTP, CoAP is also a web transfer protocol and follows request-response model. CoAP uses client server communication using

UDP based communication. Similar to HTTP, CoAP also provides commands like GET, PUT, POST, DELETE etc. and both are simply interoperable.

3. **WebSocket**
   - WebSocket is a TCP type of communication protocol which establishes a connection before transferring any data packets.
   - Once connection is made between client and server same connection is used for stream transfer until closed by the client explicitly.
4. **MQTT**
   - Message Queue Telemetry Transport uses publish subscribe model of communication. In MQTT client server communication, client connects to the server and publishes messages to topics available on the server.
   - Intermediate broker manages these messages from clients and forwards it to the consumers subscribed to the topics. MQTT is useful in applications and devices where resources (processing and memory) are limited and has low bandwidth.
5. **XMPP**
   - eXtensibleis Messaging and Presence Protocol is used for real time communication and also when XML data needs to be streamed in network. XMPP allows transmitting XML based data in small chunks throughout the network devices in the real time manner.
   - Client to sever and server to server communication is possible using XMPP. XMPP is a decentralized protocol which provides high powered application support such as voice/video calls, multi-party chats, gaming and messaging etc.
6. **DDS**
   - Data Distribution Service provides data centric device to device or machine to machine communication. DDS uses publish subscribe model of communication in which publishers i.e. device that generates data creates topics to which consumers i.e. devices that want to use this data can subscribe.
   - DDS has configurable reliability and also provides Quality of Service.
7. **AMQP**
   - Advanced Message Queuing Protocol is an open application layer protocol mostly used for business messaging. AMQP uses point to point and publish subscribes model of communication for routing and queuing of data.
   - An intermediate AMQP broker manages messages and its flow through the network. It receives messages from publisher and routes it to the subscribed consumer.
   - This is done using queuing technique. Published messages are copied into the queues. Messages can either be delivered by the brokers or consumers can pull them from message queues.

Syllabus Topic : Logical Design of IoT

1.8 Logical Design of IoT

- Logical design of IoT specifies entities and processes in abstraction without going into details.
- Following section describes a functional block of IoT which will help to understand how IoT based applications are implemented and communication models which will elaborate how the data is sent and receive in case of IoT based communication.

Syllabus Topic : IoT Functional Blocks

1.8.1 Internet of Things - Functional Blocks

Q. Explain Functional Blocks of IoT.

- IoT application consist of various functional blocks which provides the ability to sense data, identify devices, actuate, communicate in between and manage the resources. Fig. 1.8.1 describes the IoT functional blocks.

☞ **Device**

- Device or "thing" in IoT based systems provides functionalities such as sensing, actuating, communicating, monitoring and controlling. This is the actual hardware part of an IoT based system.
- An IoT system requires a business logic that manages the whole blocks. Following four blocks are decides core functioning of IoT system :

(1) Communication

Communication block handles communication between various devices and central nodes using various protocols discussed in Section 1.7.2.

(2) Services

An IoT system uses numerous services for device controlling, device detection, device monitoring, device controlling etc.

(3) Management

- An IoT system provides various functionalities like turning ON LEDs, buzzing alarms etc. which are activated on certain conditions.
- All such functions are governed via management block dictating which function should execute when.

(4) Security

Security manages authorization, authentication, integrity and data security in an IoT system.

Application

- An IoT system manages various "things" and notifies user about any conditions or generate results based on sensed data. Applications are needed to represent such data relevant to the user.
- Applications are also used to control and monitor such IoT systems. This is the front end part of an IoT application which is accessible to the user.

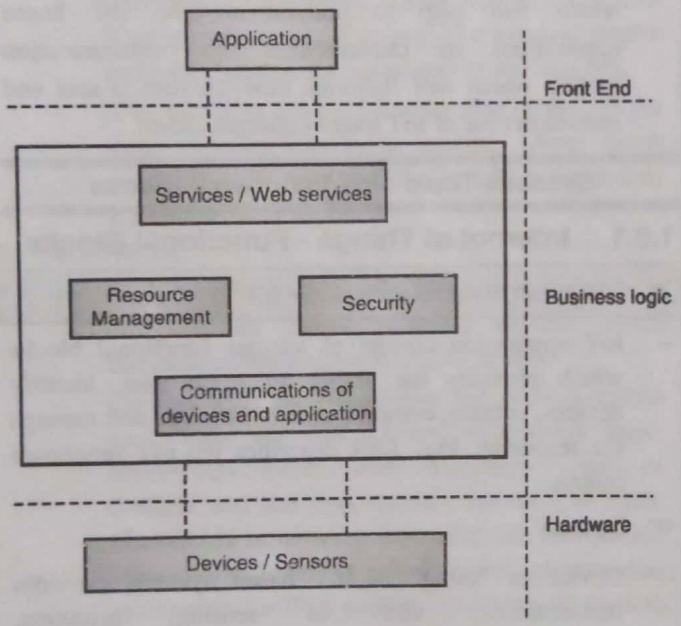

Fig. 1.8.1 : IoT Functional Block

Syllabus Topic : IoT Communication Models

1.8.2 Internet of Things : Communication Models

Q. Explain following communication models of IoT.
(1) Request – Response Model
(2) Push – Pull Model
(3) Publisher – Subscriber Model
(4) Exclusive Pair Model

Communication in IoT based application can be carried out in various ways. Following are some of the communication models discussed.

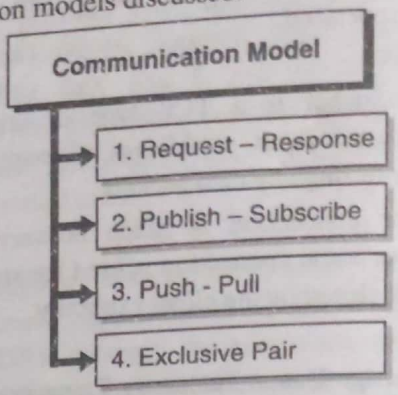

Fig. C1.10 : Communication Model

1. Request - Response

- In case of a Request Response model as the name suggest client sends request and server responds to the request.
- Each of such request - response is independent of other requests and responses. This makes it a stateless model of communication.
- On reception of the request from the client, server categorizes the type of request and determines what kind of data needs to be sent as a response.
- Server fetches this data from the storage (local or remote) and prepares a response with the fetched data.
- Finally server sends this to the client as a final response completing one request - response cycle.

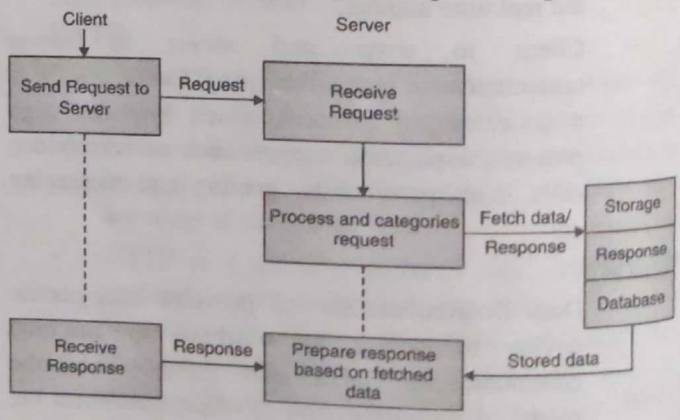

Fig. 1.8.2 : Request - Response Model

2. Publish - Subscribe

- Publish subscribe model has three key elements involved viz. publishers, brokers and consumers.
- Publishers are the source of data in communication. Brokers are intermediate entity that manages topics to which publishers send data.

- Consumers are the users which read the messages from the topics maintained by the brokers. Publishers are not aware of the consumers.
- On reception of the request from the client, server categorizes the type of request and determines what kind of data needs to be sent as a response.
- Once data is published to a topic, broker sends this message to all the consumers subscribed to the specific topic.

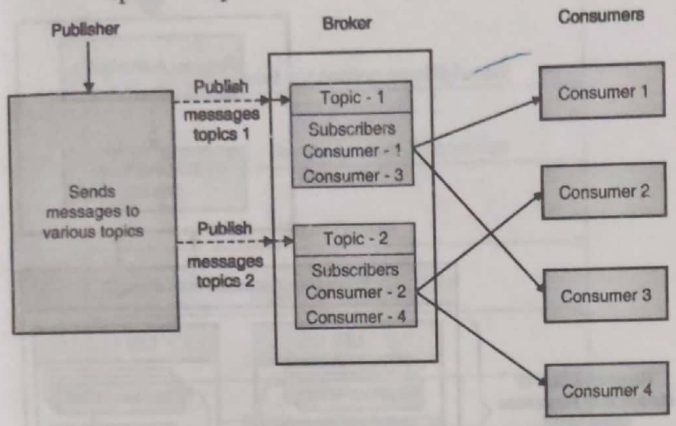

Fig. 1.8.3 : Publish - Subscribe Model

➜ 3. Push - Pull

- In Push - Pull communication model data producers push the data to the queues, and consumers pulls the data from the queues as per need.
- Queues are used to separate out single producer consumer communication. It is not necessary for the producers to be aware of the consumers.
- Queues also works as a buffer mechanism and a flow control mechanism when there is mismatch between the rate at which data is pushed by producers and the rate at which it is pulled by consumers.

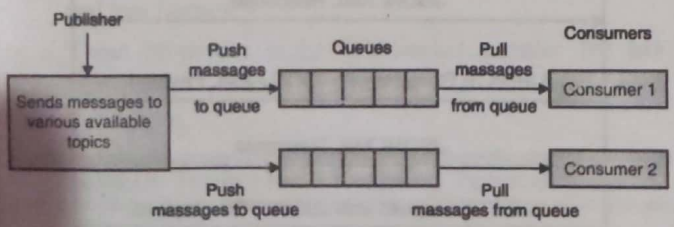

Fig. 1.8.4 : Push - Pull Model

➜ 4. Exclusive Pair

- In exclusive pair communication model a dedicated communication link is set between the client and server.
- This provides full duplex and bidirectional communication between the client and server.
- Once the connection is established it remains open until client send request to close it.
- Exclusive pair communication model is stateful type of communication and the server is aware of all the open connections.

Syllabus Topic : IOT Communication APIs

1.8.3 Internet of Things - Communication API

- In this section we study two specific communication APIs that are most frequently used in IoT based applications.
- The first one is REpresentational State Transfer which is referred as REST based API and the other is WebSocket based API.

☞ REST

Q. Explain REST communication API.

- REST is a set of architecture style used for designing network applications, web services or web APIs that are targets how server resources are addressed and transferred.
- REST follows request – response type of communication model.
- Most of the REST API works over HTTP by using the delivery methods that HTTP offers (GET, PUT, POST, DELETE etc.).

☞ REST based API follows following characteristics or constraints

- **Stateless :** In REST communication each request receives its own response and it is not related with any other request or reply.
- Because of this, each request must clearly describe its requirement and specify it in the request itself as no data from the previous context is available in communication.
- **Client - Server :** Client server communication distinguishes works of each side and separates it clearly.
- As Client side will not be bothered about storage and other server work and server side will not be bothered

about GUI development for the user and front end i.e. the client's work.

- **Cacheable :** Cache property defines whether the response for any request can be cached or not.
- If cache feature is ON then data in the response is cached in client and is readily available for reuse for next requests.
- The request needs to be implicitly or explicitly labeled as cacheable or non-cacheable.
- **Layered System :** Layered system defines the boundaries of the components within each specific layer only.
- For example a client as a component cannot tell whether it is connected to the end server or an intermediate node/server.
- **Uniform Interface :** This constraint specifies that the method of communication between client and server should be uniform throughout the communication.
- For example resources are identified by URIs in web based systems, which is uniform.
- The data returned by the server to the client has different format and is well understood by client.
- **Code on Demand :** This constraint provides the server facility of sending executable codes and/or scripts to the client.
- Although this constraint is optional and all the other are compulsory.
- A RESTful web service is an API implemented as a combination of HTTP and REST characteristics defined above.
- Fig. 1.8.6 shows client server communication using REST API and Fig. 1.8.7 shows request - response model using REST.
- As shown in Fig. 1.8.6, URIs are used to represent resources in a RESTful web service.
- Client tries to access these resources via URIs using commands like PUT, GET, POST, DELETE etc. defined by HTTP.
- In response to the request in RESTful web service, server responds with JSON object or XML file.

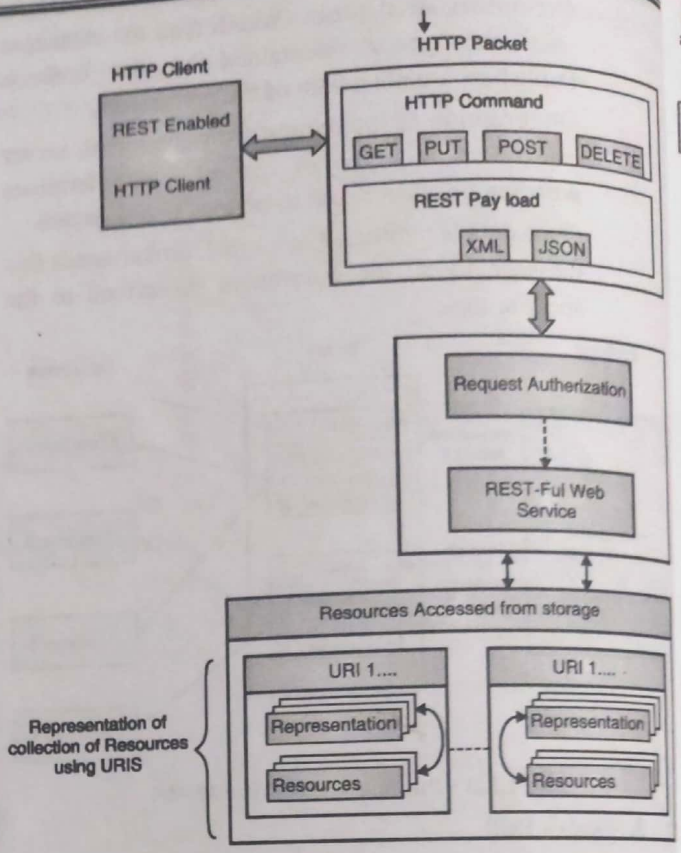

Fig. 1.8.5 : REST Communication Flow

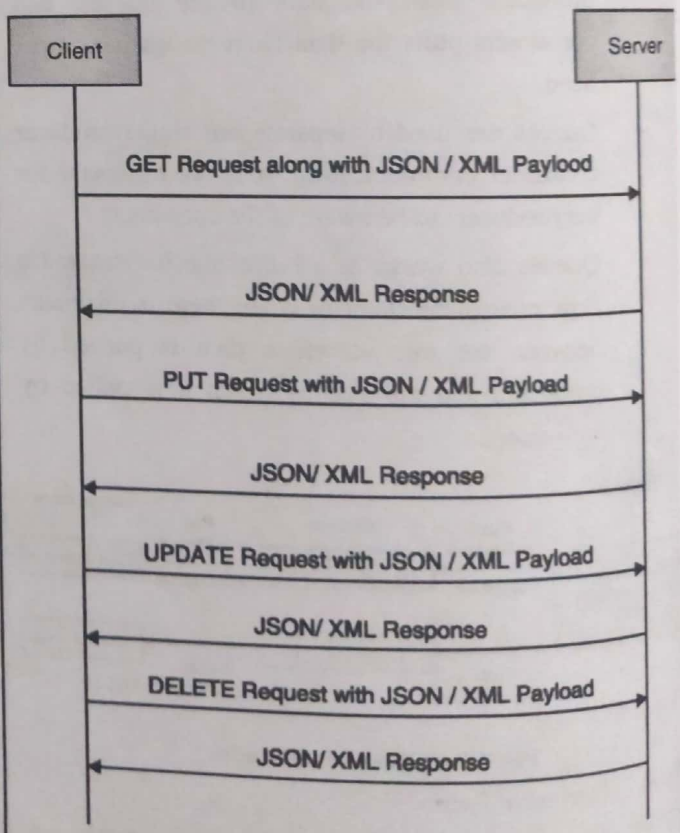

Fig. 1.8.6 : Client Server Communication Using REST

WebSocket

Q. Explain WebSocket communication API.

- It is a full duplex, bidirectional communication in client and server.
- Web Sockets follow exclusive pair type of communication model.

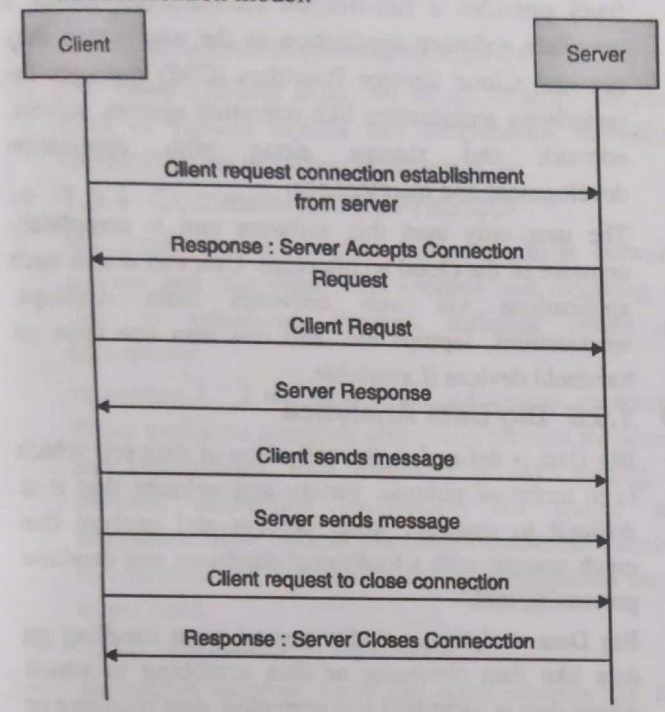

Fig. 1.8.7 : Client Server Communication Using WebSocket

- Web Socket communication starts with a connection establishment phase where a dedicated connection is created between client and server which is maintained until closed explicitly by client.
- This phase is also called as WebSocket handshake, once this is successful full duplex communication is followed between client and server.
- Because of this, it can be considered as stateful type.
- Connection setup is not required for each message because of which there is less overhead, lower traffic and less latency.
- These properties make WebSocket popular for IoT applications that require low latency and high throughput.

Syllabus Topic : IoT - Enabling Technologies

1.9 Internet of Things - Enabling Technologies

Q. Explain following technologies in relation with IoT.
1. Wireless Sensor Network
2. Cloud Computing
3. Big Data Analytics

- IoT based systems are backed up by plenty of technologies in the back end. Embedded Systems, Wireless Sensor Network, Communication Protocol, Cloud Computing, Big Data Analysis, Security protocols and architectures are few of them. These technologies handle hardware, software and business intelligence perspective of an IoT application. In the following section few are discussed.

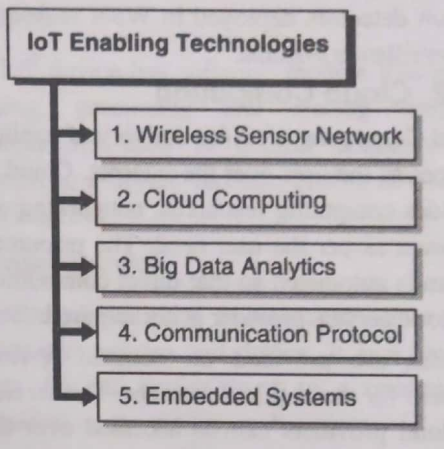

Fig. C1.11 : IoT Enabling Technologies

1.9.1 Wireless Sensor Network

- WSN consist of network of distributed sensors or collection of sensors that acts as nodes, that collects actual data in the environment. WSN also has routers and coordinators. Routers help transfer the collected data from the nodes to coordinators for further processing. Once this data is collected this data can further be transmitted to the Internet where coordinators act as gateway.
- WSN provides self-organizing and self-configurable network. And along with this the ability to provide environmental and physical condition monitoring using large number of low-powered and low-cost nodes makes WSN very useful. As WSN has large number of nodes, manual setup is very tedious task. Self-organizing capacity of WSN makes adding new nodes or removing old ones easier.
- ZigBee which is based on IEEE 802.15.4 is one of the most popular protocols used by WSN. ZigBee works at 2.4 GHz frequency and provides data rate up to 250 Kb/S. ZigBee provides operational range of 10m to 100m based on the physical / environmental conditions it is deployed under.

Some examples of WSN used in IoT systems are

(i) WSN is used to monitor soil moisture levels in different regions of land in agricultural IoT systems.

(ii) Indoor WSN checks air quality and levels of various gases in the room's environment in indoor case study of Home Management IoT based systems.

(iii) WSN helps collect data like temperature, humidity, pressure etc. in the atmosphere to correctly deduce weather conditions in weather monitoring system.

(iv) Building maintenance systems deploys WSN to check pressure, vibrations in buildings, bridges etc.

(v) Motion detectors deployed in WSN is frequently used in surveillance systems.

➜ 1.9.2 Cloud Computing

- Cloud Computing provides delivery of applications and services to the user over the internet. Cloud computing provides computing resources, networking and storage resources as per the user need. The resources delivery system is automated so that direct communication with the cloud service provider is not required.
- Services are provided on demand obviously with payment for the required services. The services offered by cloud providers can be accessed over the internet via. web applications using PCs, laptops or using apps on handheld devices.
- In cloud based environment the storage or computing services are on shared basis meaning the same hardware that is providing storage to one user is also providing storage resources to other users. This is called as multi-tenant aspect of cloud. Users access virtual resources on top of physical resources because of which this multi-tenant aspect is not felt by the user while cloud use.
- Cloud Computing is provided in different forms to the user:

(1) Infrastructure as a Service (IaaS)

- IaaS provides computing and storage resources to the user as per their requirement. Computing and storage environment is provided in the virtual manner to the user. This storage space and computing resources such as RAM, processor power etc. is managed by the user in the virtual space.
- The underlying implementation is managed by cloud service provider. User can install any OS and applications as per need on this virtual space provided on the cloud. User can start, stop, manage and configure this space accordingly and will be charged as per use.

(2) Platform as a Service (PaaS)

- In PaaS cloud service provides application development tools, application programming interface (APIs), software libraries and all other services needed for a complete application development.
- Cloud Service Providers (CSP) manages the underlying architecture like operating system, servers, network and storage. Users manage developing, deploying, configuring and maintaining the applications developed over the cloud.

(3) Software as a Service (SaaS)

- SaaS provides a full-fledged UI for a service or a complete software application to the user which they can use. Cloud Service Providers (CSP) manages the underlying architecture like operating system, servers, network and storage along with application development and maintenance.
- The user only uses this software and is completely unaware of the cloud architecture. User can access such applications via web browsers from desktops, workstations, laptops etc. and can also use apps on handheld devices if available.

➜ 1.9.3 Big Data Analytics

- Big Data is defined as the collection of data sets which is in terms of volume, variety and velocity that it is difficult to manage, store, process and analyze this much amount with a traditional databases and database processing tool.
- Big Data analytics includes several steps handling on data like data cleansing or data scrubbing in which wrong data is identified and corrected, data munging or data wrangling is the process in which raw data is converted into more understandable format, data processing uses the dataset and apply various algorithms to convert it into useful information and this information is represented by data visualization techniques.

☞ Characteristics of Big Data

(1) **Volume :** Big data is generally not threshold with particular amount of data but the vast sized data which is difficult to manage, store, process and analyze using traditional databases and database processing tool is categorized as big data. For example data generated through modern healthcare systems, insurance companies, IT industries etc. requires big data assistance for storage and processing such huge data.

(2) **Velocity :** Velocity is the characteristic which generates big data. Velocity refers to how fast data is generated and how frequently it changes.

(3) **Variety :** Variety refers to the different forms of data coming into for processing. Structured, unstructured data, text, image, audio, video etc formats have be managed by big data analytics.

Some examples of Big Data in IoT based systems

(i) Sensor data generated by weather monitoring system.

(ii) Health and body related data generated by IoT based wearable systems.

(iii) Retail outlet and inventory management IoT system generates vast data.

(iv) IoT based location monitoring systems for vehicles.

(v) IoT based systems deployed in an industry keeping track of various aspects like temperature, humidity, chemical levels in chemical industries etc.

1.9.4 Communication Protocol

- IoT based systems require communication between devices and machines. This sending and receiving of data is processed through different levels of networking.
- In section 1.7.2 we have discussed various protocols. These protocols specifies addressing formats for source and destinations, data encoding if any, routing of packets, connection mechanism for source and destination, flow control mechanisms, sequence control mechanisms etc. that makes the communication fluent as per need.

1.9.5 Embedded Systems

- Embedded systems form the basic devices for IoT based systems. We have discussed embedded systems in details in section 1.1.1. Embedded system has a microcontroller or a microprocessor at its core.
- Various kinds of components like memory based components for ROM, RAM along with networking (Ethernet, WiFi, Bluetooth), storage(SD, eMMC, Flash memory) and input/output (Display, Mouse, Keyboard) units are grouped together with the central core.
- To coordinate all these components embedded systems run Embedded Operating Systems which generally focuses on real time task management.

Syllabus Topic
IoT Levels and Deployment Templates

1.10 Internet of Things - Levels and Deployment Templates

Q. Differentiate between REST and WebSocket.

- IoT levels describe the deployment scope of an IoT application. Starting with lowest IoT level – 1 based application where a single node interacts locally to highest level – 6 based IoT application where multiple nodes independently coordinate data over the cloud.
- Before going into the details of IoT deployment templates let's review few terms used in IoT based applications and will be used in diagrammatic representation.

Device

- An IoT device has unique identity, sensing, actuating and monitoring capabilities. Section 1.7.1 discusses devices in details.

Resource

- Each IoT device has software module that is used for accessing, processing and storing sensor data. Resources are used to control actuators that are connected to a device and also enable devices to have network access.

Controller Service

- Controller service acts as an intermediate connector between web service and device. It is a local service running on the device used to communicate with external (no on the device) web service.
- The data collected by the device is sent by the controller service to the web service and receives the commands to handle the device sent by an application via a web service.

Database

- All the data generated by the IoT device is stored locally or on the cloud using a database. Whether to provide local or remote access depends upon application requirement.

Analysis Component

- Analysis component accesses the IoT device's data from database and convert it into useful information. With the help of various algorithms this module analyze the data, generates results and represents it in user understandable way.
- This analysis can be performed locally or on the cloud and similarly the results can be stored locally or on the cloud.

Application

- An application is a way users interact with IoT based systems. Applications provide an interface to access, monitor and control the IoT system via a GUI based software application, a web URI or even a mobile app.

Web Service

- Web service relates together all the IoT systems core components. Web service connects IoT device,

database, analysis component and application to form a complete IoT based system.

- Web services are used to write the core logic (Business Intelligence) of the system. Web services can be implemented using WebSocket or REST with HTTP. As seen in section 1.8.3 both the APIs are discussed in details, here we'll distinguish between them.

Comparison	REST	WebSocket
State	REST is stateless.	WebSocket is stateful.
Data Flow Direction	REST follows unidirectional data flow.	WebSocket follows bidirectional data flow.
Communication Model	REST use request – response communication model.	REST use full duplex (exclusive pair) communication model.
TCP connections	In REST, each HTTP request sets up new TCP connection.	In WebSocket, client – server communicates over a single TCP connection.
Header Overhead	In REST all requests are independent of each other and needs to be represented fully for which header is required. Due to this REST has header overhead associated with it.	WebSocket maintains a single connection for whole communication so no extra overhead is needed for each request.
Real Time applications	REST is not suitable for real time applications because of the overhead.	WebSocket is suitable for real time applications as there is lesser overhead comparatively.
Scalability	Scalability is comparatively easier in REST as each request is independent.	Scalability is comparatively tougher in WebSocket because of stateful nature.

1.10.1 IoT Level - 1

Q. Explain Level - 1 of IoT deployment template with example.

- IoT level - 1 based system has a single node or IoT device that performs sensing, actuation, data storage, performs data analysis and hosts the application.
- As shown in Fig. 1.10.1 a single node is responsible for sensing, storage and data analysis. IoT level - 1 based solution are used for low-cost low-complexity solutions in which data involved is less and data analysis and computational complexity is low.

IoT level - 1 based system example

- Consider home automation system where few home appliances and lights are controlled remotely via application.

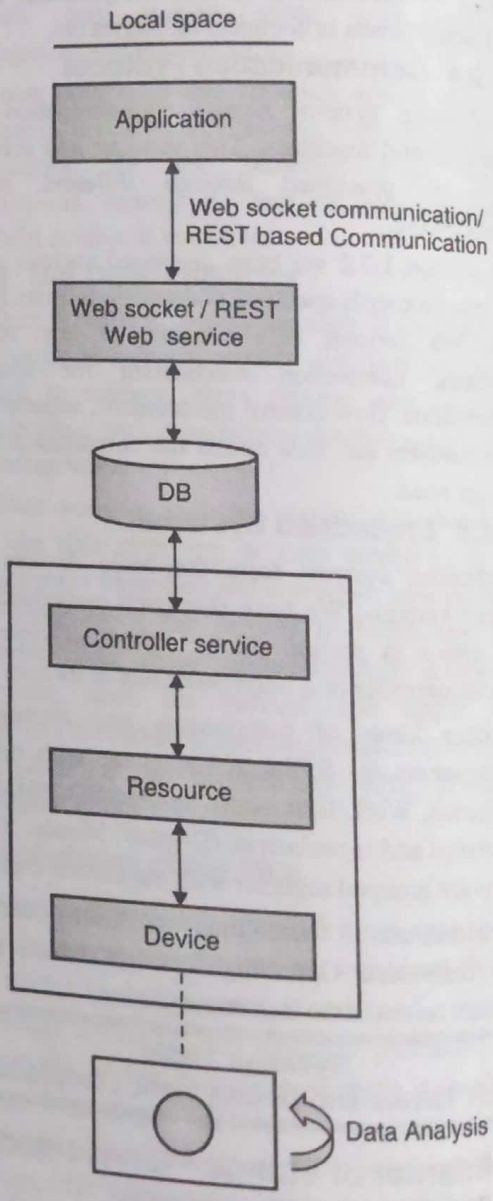

Fig. 1.10.1 : IoT Level - 1 System

- Monitoring node that performs sensing actuating sto data and performs analysis.
- A relay is connected in a system with single node **device** and lights and/or other home appliances.
- Status information of each connected home appliance maintained via device monitoring in a **local databa**

REST service is used to access and update the status of lights and appliances by changing the values in local database.

- The **controller service** continuously monitors the database and keeps track of the appliances and triggers the relay switches.
- The **application** which is deployed locally provides a GUI using which lights and other home appliances can be handled by the user.
- As the device is connected to the internet, application can remotely control the home appliances.

1.10.2 IoT Level - 2

Q. Explain Level - 2 of IoT deployment template with example.

- IoT level - 2 based systems has a single node or IoT device that performs sensing, actuation and performs data analysis.
- Data storage and hosting the application is usually done using cloud based services.
- As shown in Fig. 1.10.2 a single node is responsible for sensing and analysis while data storage and application hosting is done in the cloud environment.
- IoT level – 2 based systems can handle big data due to cloud support but data analysis and computational complexity is low as it is done locally.

IoT level – 2 based system example

- Consider smart irrigation system where a single node measures the soil moisture level and manages the irrigation system. If the moisture level falls below the specified threshold value the irrigation system is turned on.
- A single node **IoT device** senses the soil moisture and **controller service** monitors the moisture level and sends the data to the cloud. **REST** based service in the cloud receives this data and stores it in the **cloud database**, and retrieves it whenever needed by the application.
- The moisture levels are displayed in an **application** to the users which can be used for creating schedule for irrigation.

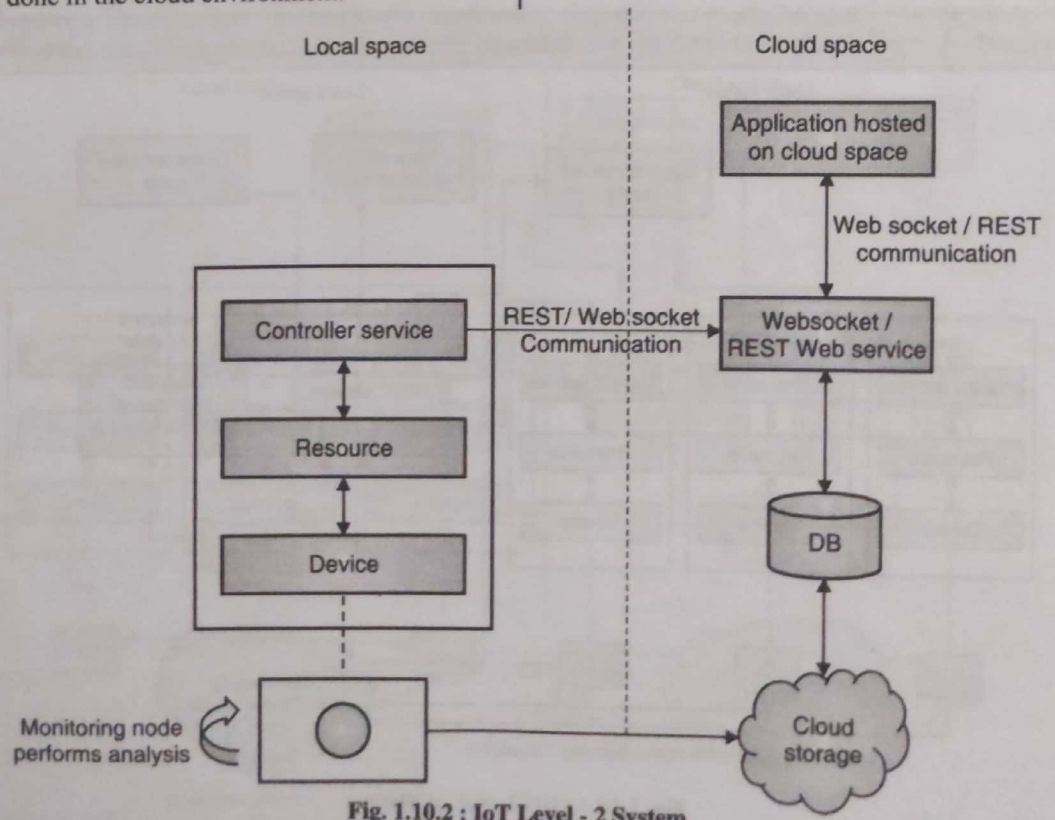

Fig. 1.10.2 : IoT Level - 2 System

1.10.3 IoT Level - 3

Q. Explain Level - 3 of IoT deployment template with example.

- IoT level - 3 based systems has a single node or IoT device that only performs sensing and/or actuation. Data storage, data analysis and hosting the application is done using cloud based services.
- As shown in Fig. 1.10.3 a single node is responsible for sensing data while data storage, data analysis and application hosting is done in the cloud environment. IoT level - 3 based systems can handle big data, extensive data analysis and higher computational complexity due to cloud support.

IoT level - 3 based system example

- For IoT level - 3 based system we'll consider example of package tracking in a delivery system. The system checks the vibration occurring to the package if the vibration levels are above the threshold alarm is raised.
- The **IoT device** has a gyroscope and accelerometer sensors to check the vibrations in the package. The **controller device** sends real time data to the cloud using **WebSocket** API which is useful in real-time application because of low overhead. **WebSocket** based service in the cloud receives this real-time data from **IoT device** and stores it in the **cloud database**, and retrieves it whenever needed for the analysis or by the application all of which exist in the cloud space.

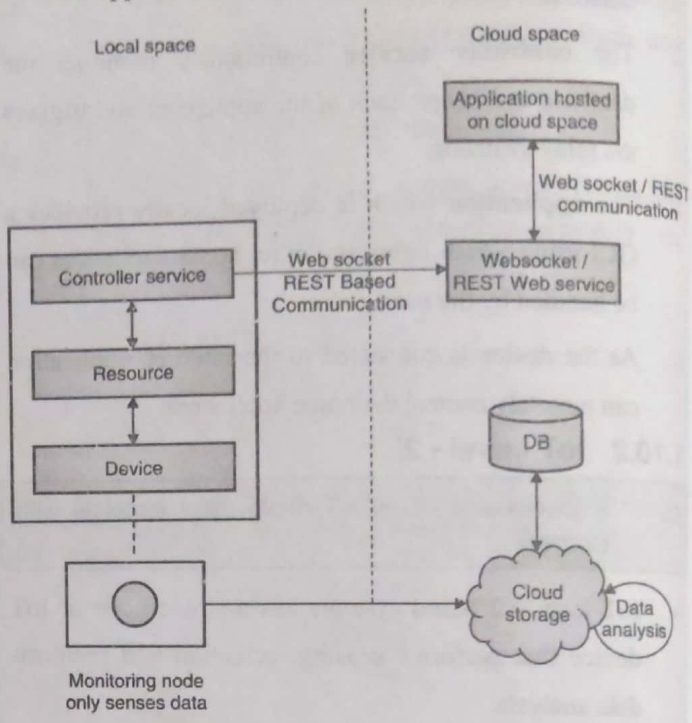

Fig. 1.10.3 : IoT Level - 3 System

1.10.4 IoT Level - 4

Q. Explain Level - 4 of IoT deployment template with example.

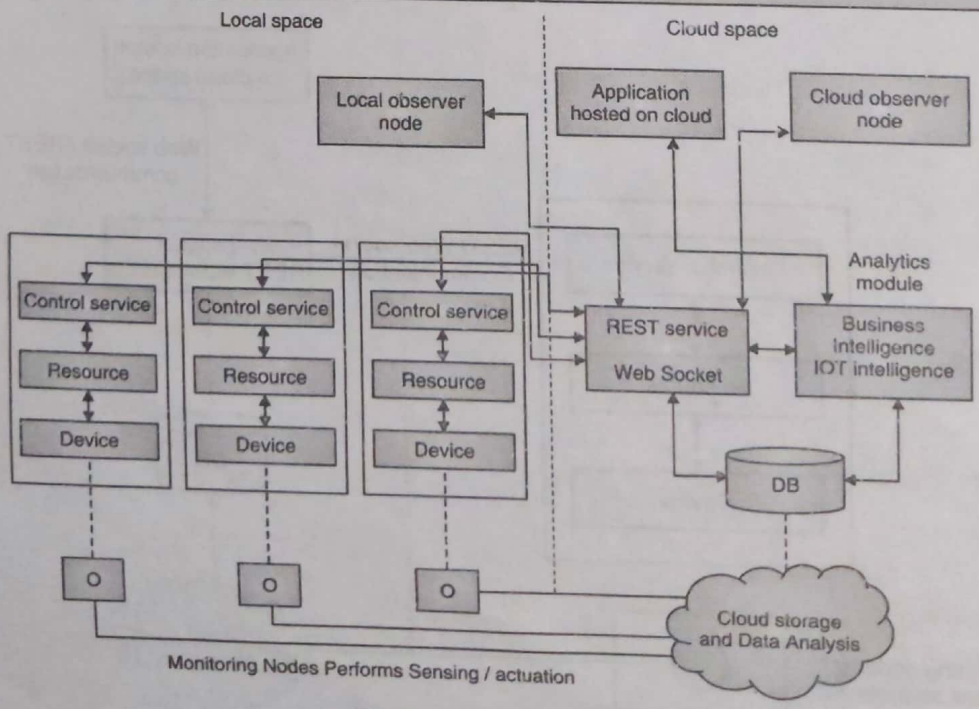

Fig. 1.10.4 : IoT Level - 4 System

- IoT level - 4 based systems contains multiple nodes or IoT devices that performs sensing and/or actuation. Data storage, data analysis and application hosting is done using cloud based services. Level - 4 also has additional local and cloud based observers.
- These observers can subscribe to IoT devices and cloud based services via REST or WebSocket based services. Observers can also fetch the information from the cloud and process the information for applications. But the observers never perform the controller functions.
- IoT level - 4 based systems are useful where multiple nodes are needed and big data can be generated with intensive data analysis capability which can be achieved by separate business intelligence logic written in cloud space.

IoT level - 4 based system example

- Consider noise monitoring IoT based system for level – 4 IoT deployments. In this system multiple nodes are dispatched in different locations for noise monitoring in that area.
- Nodes/Devices in this example have sound sensors in them. All the nodes/devices are independent of each other which means each node has its own controller service which sends data to the cloud for storage and analysis. To represent the analyzed data and result application is deployed on the cloud.

1.10.5 IoT Level - 5

Q. Explain Level - 5 of IoT deployment template with example.

- IoT level – 5 based systems contains multiple nodes or IoT devices that performs sensing and/or actuation and one coordinator node that collects data from other nodes and send it over the cloud for storage and analysis. Data storage, data analysis and application hosting is done over the cloud.
- Level - 5 also has additional local and cloud based observers same as level - 4. These observers can subscribe to and receive information from IoT devices or cloud. Observers can also process the information for applications requirement. But the observers never perform the controller functions.
- IoT level - 5 based systems are useful where multiple nodes are needed which produces big data and needs intensive data analysis capability with computational complexity which can be achieved by separate business intelligence logic written in cloud space..

IoT level - 5 based system example

- For IoT level - 5 consider Forest Fire Detection system. Nodes/ devices in this system have the capacity to sense temperature, humidity and CO2 levels in the surroundings. So the node/IoT device has temperature, humidity and CO2 sensors in it. The data sensed by these nodes is collected by the coordinator node and transferred to the cloud by the controller service on the coordinator. Coordinator node also acts as a gateway and provides Internet connectivity to the IoT based system. The data stored in the cloud is accessed by the analytics module to predict/generate results. The result is displayed to the user via application deployed over the cloud.

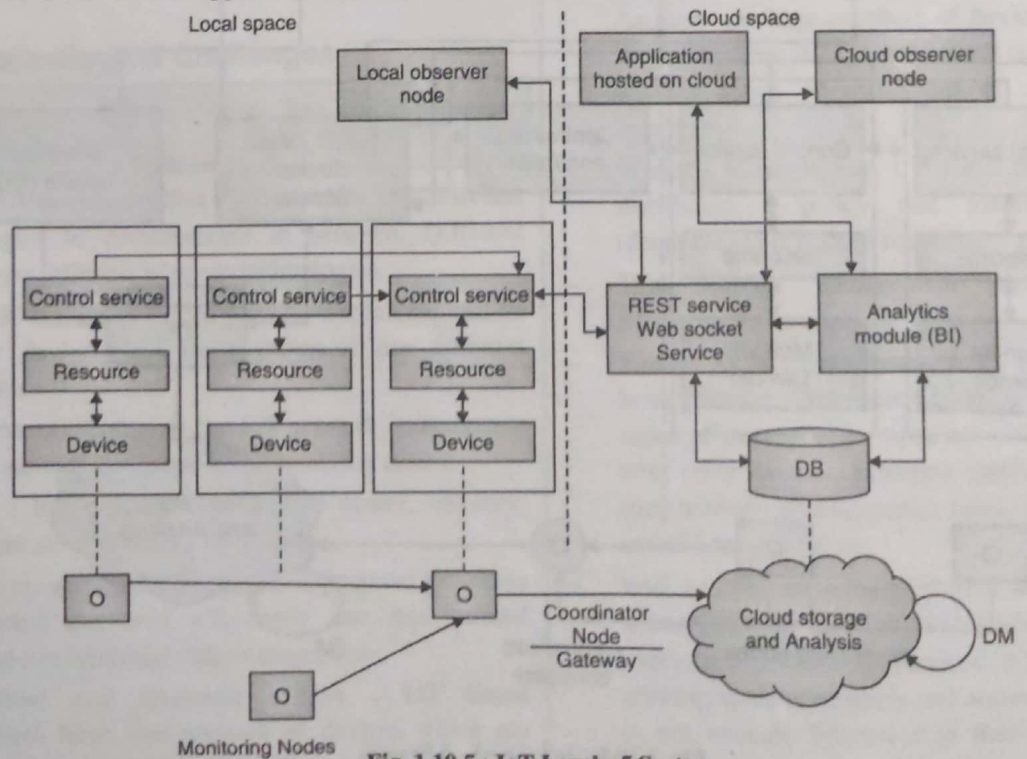

Fig. 1.10.5 : IoT Level - 5 System

1.10.6 IoT Level - 6

Q. Explain Level - 6 of IoT deployment template with example.

- IoT level - 6 based systems contains multiple nodes or IoT devices that performs sensing and/or actuation and sends data over the cloud for storage and analysis.
- Data storage, data analysis and application hosting is done over the cloud.
- IoT level - 6 has a centralized controller that can handle the ends nodes by sending control commands to the ends nodes. The centralized controller is aware of all the end nodes in the system.
- Level - 6 also has additional local and cloud based observers same as level - 5.
- These observers can subscribe to and receive information from IoT devices or cloud. Observers can also process the information for applications requirement.
- But the observers never perform the controller functions.
- IoT level - 6 based systems are useful where multiple nodes are needed which produces big data and needs intensive data analysis capability with computational complexity.

IoT level - 6 based system example

- For IoT level - 6 consider weather monitoring system which has multiple nodes/devices containing temperature, humidity, pressure etc. sensors in them.
- These nodes are deployed in different locations and sends data to the cloud based storage in real-time using WebSocket based API.
- The centralized controller monitors the nodes.
- Any update or modification in nodes is carried out via the centralized controller.
- The data stored in the cloud is accessed by the analytics module to predict/generate results.
- The result is displayed to the user via application deployed over the cloud.

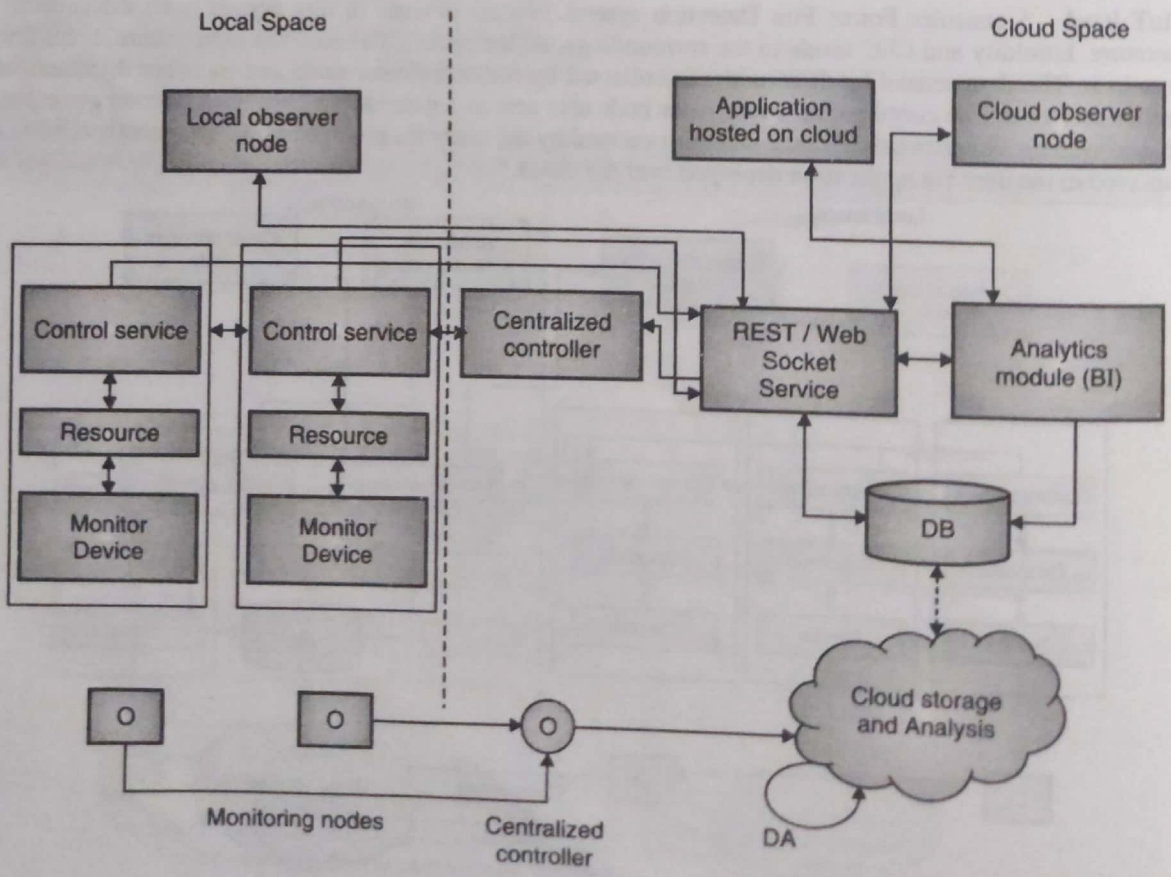

Fig. 1.10.6 : IoT Level - 6 System

Syllabus Topic : IoT Issues and Challenges

1.11 Internet of Things : Issues, Challenges and Applications

1.11.1 Design Issues

Q. Explain design issues in IoT based system development.

- **Low Memory :** "Things"/Devices in IoT have limited memory resource. (Eg: RFID tags, sensors, etc.)
- **Low Computational Power :** Objects have low processing and computational power using which small piece of data can be processed.
- **Low bitrate and throughput :** Data generated by the objects in IoT based application varies from application to application. So communicating the generated amount of data between such devices can be a challenge due to lower bitrate which results in lower throughput.
- **Low energy systems :** Object in an IoT application are working continuously so the energy consumption needs to be low and also the power supply to such devices needs to be properly maintained.
- **Bandwidth Requirements :** IoT application ranges from a small use case of home automation to a vast network monitoring in weather monitoring system.
- Depending upon the scope of application and scalability of IoT the bandwidth requirements will vary. This is one more challenge in communication point of view.

1.11.2 Technological Challenges

Q. Explain technological challenges in IoT based system development.

- **Wireless Communication :** IoT devices uses wireless technologies to communicate in between. Different devices use different wireless technologies.
- WiFi, GSM, Bluetooth, LR-WPAN, WiMax etc. can be used by devices. All these protocols use different bandwidth.
- While communicating in the IoT network convergence of these techniques causes interoperability issues.
- **Energy :** IoT devices have limited power, memory, computational and energy resources.
- To cope up with this limitation low powered protocols are needed that will effectively use this limited resources and minimizes the consumption.
- **Distributed and Dynamic nature :** IoT based applications have vast network of devices which are distributed geographically over a large span.
- These devices need to communicate with each other irrespective of their locations and they interact dynamically. It is a challenge to automatically detect these devices in the network and make them interoperable with existing system.
- **Identification:** IoT based systems has plenty of devices throughout the network and important feature of these devices is that it has a unique identity and an identifier that separate out individual thing in a wide spread network of devices.
- This addressing capability provides the means of communication endpoint to these devices. Various mechanisms are used for such identification like RFID, NFC, IPv4, IPv6, EPC Global etc. Maintaining these identities is a challenge in IoT.
- **Scalability:** Day by day the number of nodes in IoT based applications is increasing. Billions of nodes will be working for various IoT bases systems one day. So for seamless addition of new devices monitoring and configuring of things are needed.
- Based on the application the number of devices can vary, so IoT system should be able to handle these many devices at runtime. IPv6 provides one such approach as it provides large addressing space.

1.11.3 Security Challenges

Q. Explain security challenges in IoT based system development.

- **Privacy :** In the field of IoT where everything is connected to the internet privacy is a crucial issue.
- As there are large numbers of devices that can access the nodes in an IoT system which is available over the internet sense of ownership of a device is also important.
- **Identity Management :** As IoT based system is so distributed a policy that maintains the unique identification is needed for security purpose.
- This identity management protocols must be lightweight as well.
- **Trust:** As devices communicate with each other in homogeneous and heterogeneous environment the sense of trust is very important while sharing the data with other devices. Effective mechanisms to generate trust between sharing parties needs to be created before actual transfer of data.
- **End to end Security :** Internet host and device are connected over the World Wide Web where security is must. In a resource constrained IoT environment only cryptographic encryption and authentication of packets is not enough. Moreover to these available security

measures newer IoT based security techniques are required.

- **Security Solutions for attacks:** IoT nodes holds crucial data about users and environment, so external attacks are eminent on such nodes. Attacks like denial of service, snooping, flood attack, etc. can be made over the network or nodes by external mediums over the internet. Attack resistant and lightweight security protocols needs to be developed to avoid such attacks.
- **Authentication and access control:** Identity establishment between two data sharing parties is authentication. And access control refers to how much data is to be shared amongst the available resources.
- Authentication and access control together monitors the resource sharing between devices. Interoperability and backward compatibility is crucial in this technique.

Syllabus Topic : Applications

1.11.4 Application Domains

Q. Explain any four IoT based application domain.

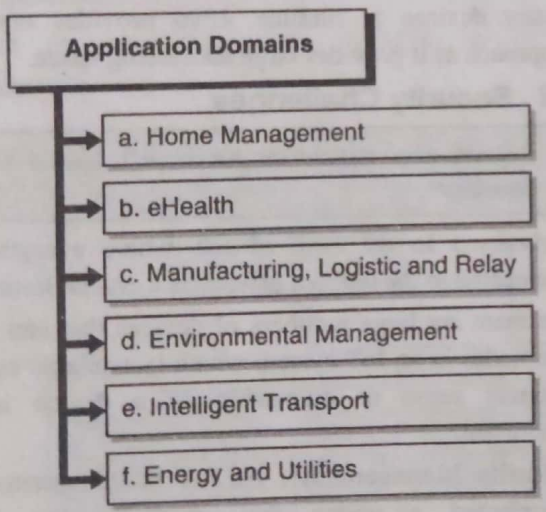

Fig. C1.12 : Application Domains

➜ **(a) Home Management**

- At home plenty of services can be deployed via various sensors. A simplest case study of monitoring lights and home appliances remotely using the mobile app.
- Intrusion detection system at the door to detect whether person is known or unknown, opening and closing doors remotely, watering the plants in the garden via remotely handled showers, etc.
- Security is crucial in case of home automation as attackers can intercept the system and get entry into the house.

➜ **(b) eHealth**

- eHealth IoT system tries to provide health services to a person regardless his/her geographical location. The patient's health is monitored and recorded using IoT based systems which can be used by a new doctor to know the patients' health history and suggest new medication accordingly.
- If person is having wearable health sensors which monitors various criteria like blood pressure, heart rate etc. such update can be tracked by doctor remotely and medications can be informed to the patient.
- Again data privacy becomes vital as personal and crucial information is available to the sensors and IoT based system.

➜ **(c) Manufacturing, Logistic and Relay**

- IoT can work wonders in Industries starting from planning phase till the complete supply chain management.
- Inventory management, goods authentication, RFID used for monitoring products, using location based services to track packages for delivery, real time stock monitoring, etc can be done in industries.

➜ **(d) Environmental Management**

- Environment monitoring and weather forecasting are very important in agricultural point of view. The vastly distributed nodes senses physical conditions environmental conditions, soil moisture levels, PH values etc., which can be used to generate a proper plan for harvesting.
- Such data can also be used for disaster management via cyclone, tsunami, and flood prediction.

➜ **(e) Intelligent Transport**

- GPS tracking enabled with vehicle to vehicle communication can create a smart system for transport. Users will be able to select shortest route to their destination.
- Notification of any road blocks or newly created rules can be prompted to the drivers. For government vehicle tracking system, traffic data collection and route management, new law or rules enforcement etc. can be performed using smart transport systems.

➜ **(f) Energy and Utilities**

- IoT can be used at consumers end as well as at supplier's end. Smart electricity grid to monitor and create scheduled power consumption charts, smart water transmission grid that controls systematic water flow to areas with suitable timings.
- At consumers' end smart electricity and water management (use cases of home automation) can reduce water and electricity wastage.

1.12 Exam Pack (Review Questions)

Syllabus Topic : Embedded Systems

Q. What is Embedded systems ? *(Refer section 1.1.1)*

Syllabus Topic : Application Domain and Characteristic of Embedded System

Q. Explain applications of embedded systems. *(Refer section 1.1.2)*

Q. Explain basic characteristics of embedded systems. *(Refer section 1.1.3)*

Syllabus Topic : Real Time Systems

Q. What is real time systems? Explain with examples. *(Refer section 1.2.1)*

Q. What are the types of real time tasks? Explain in details. *(Refer section 1.2.2)*

Q. Explain following terms regarding real time scheduling: Arrival time, Ready time, deadline, scheduling time, burst time, waiting time, completion time. *(Refer section 1.2.3)*

Q. What are periodic, aperiodic and sporadic tasks? *(Refer section 1.2.3)*

Syllabus Topic : Real Time Scheduling

Q. Explain the concept of Real Time Scheduling Algorithms with example. *(Refer section 1.2.4)*

Q. Explain different types of real time scheduling algorithms. *(Refer section 1.2.5)*

Q. Explain Rate Monotonic Algorithm with example. *(Refer section 1.2.5(1))*

Q. Explain Earliest Deadline First Algorithm with example. *(Refer section 1.2.5(2))*

Syllabus Topic : Processor Basics and System-On-Chip

Q. What is a processor? Explain system on chip. *(Refer section 1.3)*

Syllabus Topic : Introduction to ARM Processor and its Architecture

Q. Explain ARM SoC. *(Refer section 1.4)*

Q. Differentiate between RISC and CISC. *(Refer section 1.4.2)*

Q. Describe advanced features of ARM. *(Refer section 1.4.3)*

Q. Describe naming convention of ARM. *(Refer section 1.4.5)*

Q. What is ARM cortex series? Explain in details. *(Refer section 1.4.6)*

Q. Explain seven operating modes of ARM in short. *(Refer section 1.4.7(1))*

Q. Explain ARM register set. *(Refer section 1.4.7(2))*

Q. Describe bit configuration of CPSR of ARM. *(Refer section 1.4.7(2))*

Q. Explain instruction format of ARM. *(Refer section 1.4.7(3))*

Syllabus Topic : IoT - Definition and Characteristics of IoT

Q. Define Internet of Things. Explain characteristics of IoT. (Refer section 1.5)

Syllabus Topic : IoT - Vision, Emerging Trends, Economic Significance, Technical Building Blocks

Q. Explain vision of Internet of Things with a suitable case study.(Refer section 1.6.1)

Q. What are the emerging trends of Internet of Things ? *(Refer section 1.6.2)*

Q. Explain economic significance of Internet of Things. *(Refer section 1.6.3)*

Q. Explain technical building blocks of Internet of Things. *(Refer section 1.6.4)*

Q. Explain following IoT mechanisms : RFID, EPCIS, ONS. *(Refer section 1.6.4)*

Syllabus Topic : Things of IoT

Q. What is meant by "thing" in IoT ? Explain in details. *(Refer section 1.7.1)*

Syllabus Topic : IoT Protocol

Q. Explain various protocols needed for IoT systems. *(Refer section 1.7.2)*

Q. Explain various protocols needed for IoT systems in the following layers:

1. Link Layer
2. Network Layer
3. Transport Layer
4. Application Layer *(Refer section 1.7.2)*

Syllabus Topic : IOT Functional Blocks

Q. Explain Functional Blocks of IoT. *(Refer section 1.8.1)*

Syllabus Topic : IoT Communication Models

Q. Explain following communication models of IoT.

(1) Request – Response Model

(2) Push – Pull Model

(3) Publisher – Subscriber Model

(4) Exclusive Pair Model *(Refer section 1.8.2)*

☞ Syllabus Topic : IOT Communication APIs

Q. Explain REST communication API. *(Refer section 1.8.3)*

Q. Explain WebSocket communication API. *(Refer section 1.8.3)*

☞ Syllabus Topic : IoT : Enabling Technologies

Q. Explain following technologies in relation with IoT.

1. Wireless Sensor Network
2. Cloud Computing
3. Big Data Analytics *(Refer section 1.9)*

☞ Syllabus Topic : IoT Levels and Deployment Templates

Q. Differentiate between REST and WebSocket. *(Refer section 1.10)*

Q. Explain Level - 1 of IoT deployment template with example. *(Refer section 1.10.1)*

Q. Explain Level - 2 of IoT deployment template with example. *(Refer section 1.10.2)*

Q. Explain Level - 3 of IoT deployment template with example. *(Refer section 1.10.3)*

Q. Explain Level - 4 of IoT deployment template with example. *(Refer section 1.10.4)*

Q. Explain Level - 5 of IoT deployment template with example. *(Refer section 1.10.5)*

Q. Explain Level - 6 of IoT deployment template with example. *(Refer section 1.10.6)*

☞ Syllabus Topic : IoT Issues and Challenges

Q. Explain design issues in IoT based system development. *(Refer section 1.11.1)*

Q. Explain technological challenges in IoT based system development. *(Refer section 1.11.2)*

Q. Explain security challenges in IoT based system development. *(Refer section 1.11.3)*

☞ Syllabus Topic : Applications

Q. Explain any four IoT based application domain. *(Refer section 1.11.4)*

❑❑❑

UNIT II

CHAPTER 2

Embedded IoT Platform Design Methodology

Syllabus Topics

Purpose and requirement specification, Process specification, Domain model specification, information model specification, Service specifications, IoT level specification, Functional view specification, Operational view specification, Device and component integration, Application development.

2.1 Introduction

- In Chapter 1, we have studied six IoT system levels. Every IoT level starting from IoT level 1 to IoT level 6 consist of different components and deployment configurations.
- For different application different levels are suitable.
- Different components like IoT devices, network resources, application server, database server, web services, analytics components requires interaction between them in IoT system.
- So designing IoT system is challenging task. Because of various choices available for each component it is very difficult for designers to evaluate among available alternative.
- So IoT system designers consider specific product or service and design system with respect to that product or service.
- Disadvantage of above technique is it may lead to vendor lock-in which may be satisfactory to vendor but totally unacceptable to the customer.
- Another disadvantage is adding new feature or replacing certain service or product choice for component is very complex and sometime may require re-design of IoT system.
- In this unit, we will learn generic design methodology.
- That is this design methodology will be independent of specific service or product.
- This methodology is also independent of programming language.

Advantages of using such methodology

- Design, testing and maintenance time is reduced.
- Provide better interoperability.
- Complexity is reduced.

2.2 IoT Design Methodology

Q. Which are various steps in IoT design methodology ?

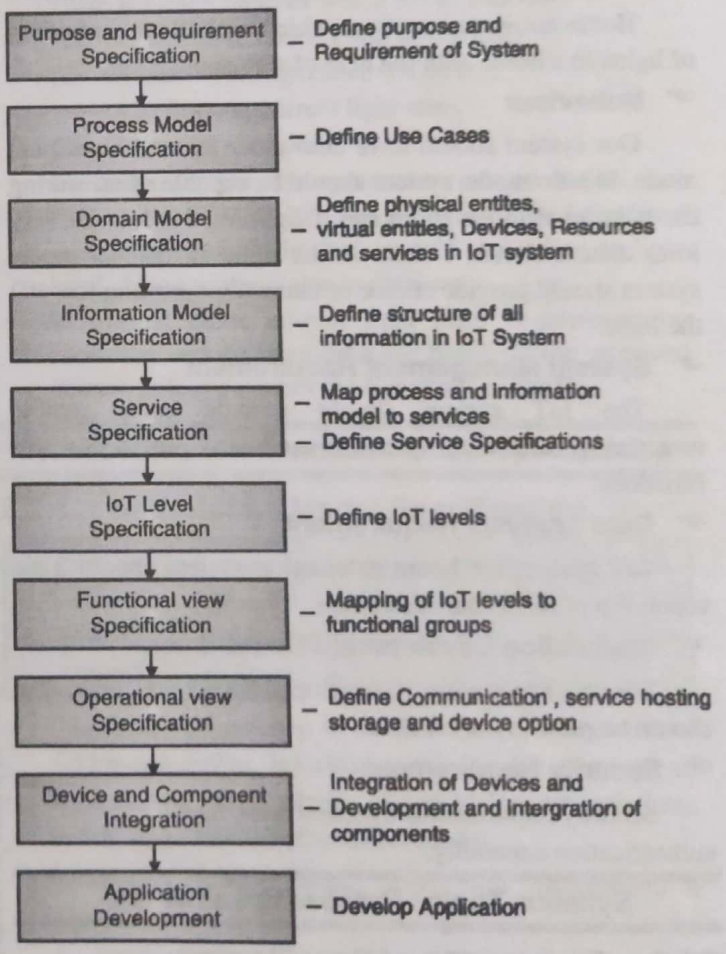

Fig. 2.2.1 : IoT system Design Methodology

Various steps involved in IoT system design methodology are shown in Fig. 2.2.1. To explain every step in IoT system methodology we will consider example of smart IoT based home automation system.

Syllabus Topic : Purpose and Requirement Specification

2.2.1 Purpose and Requirement Specification

Q. Explain purpose and requirement specification in IoT design methodology with example.

Q. Consider any IoT based application and write purpose and requirement.

- This is the first step in design methodology where we define the purpose and requirements of the system. Requirements include various requirements like data collection requirements, system management requirements, data analysis requirements, data privacy requirement, security requirements, user interface requirements.
- Considering the example of home automation system, purpose and requirement specification can be described as follows

Purpose

Home automation system which will allow controlling of lights in a home with the help of web application.

Behaviour

Our system should have auto mode as well as manual mode. In auto mode, system should be capable of measuring the light level in the room and if light level is low (or very low) then it should switch on the light. In manual mode, system should provide choice of manually operating (on/off) the light.

System Management Requirement

The IoT system should provide some remote monitoring functions. System should also provide control functions.

Data Analysis Requirement

IoT system for home automation system should have capability of local analysis of data.

Application Development Requirement

Though application is developed locally on device, it should be remotely accessible.

Security Requirement

Security requirements included basic user authentication capability.

Syllabus Topic : Process Specification

2.2.2 Process Model Specification

Q. Explain process model specification for home automation system.

Q. Consider any IoT based application and write process model specification for same application.

- This is the second step in IoT design methodology where we define process specifications. Process model specification for home automation system is shown in Fig. 2.2.2. In Fig. 2.2.2, we can see two modes of system auto mode and manual mode. If mode is auto mode, the system keeps track of light level. System changes state of light to on if light level is low (or very low).

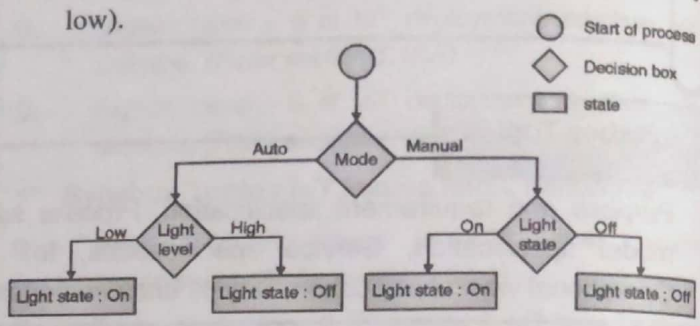

Fig. 2.2.2 : Home automation IoT system process specification

- System changes light state to off if light level is high (if there is no need of light).
- If mode is manual mode then system checks light level in the room set by the user. If light state is on (i.e. user set light state on) then system changes light state to "on". If light state is off (i.e. user set light state off) then system changes light state to "off".

Syllabus Topic : Domain Model Specification

2.2.3 Domain Model Specification

Q. Write domain model specification for smart home automation system.

- This is the third step in IoT system design methodology. In this step domain model is defined.

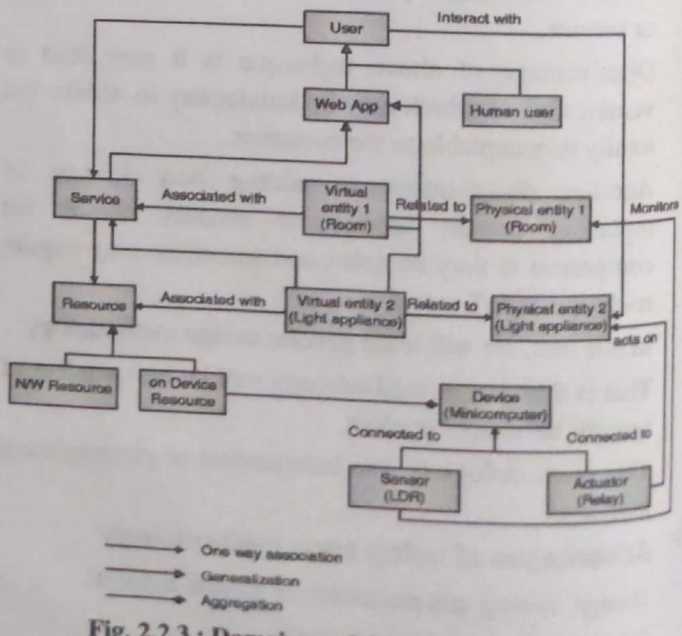

Fig. 2.2.3 : Domain model for IoT based home automation system

- The domain model consists of description about main concepts, entities and objects in the domain of system which is being designed.
- Domain model also defines various attributes of objects and relationships between those objects. Abstract representation of main concepts, entities and objects is provided by domain model. This abstract representation is independent of technology and platform to be used.
- With the help of domain model system designer can easily understand the IoT domain for the system which he wants to design.
- Fig. 2.2.3 shows domain model for our example of home automation system. Main concepts, entities and objects in the domain model consist of physical entity, virtual entity devices, resources and services.

Physical Entity

Q. Describe in short physical entity in domain model specification.

- Discrete and identifiable entities like room, light, an appliances are physical entities in the physical environment.
- Using sensors or by performing actuation on physical entities, IoT system gives information about various physical entities.
- In our example of home automation system two physical entities can be considered. First entity can be room of which light level is monitored and second entity can be light appliance which is to be controlled (on or off).

Virtual Entity

Q. Describe in short virtual entity in domain model specification.

- Representation of physical entities in digital word is called as virtual entity. There is a virtual entity for every physical entity.
- As there are two physical entities in our example of home automation system there will be two virtual entities one for room and other for light appliance.

Device

- Physical entities and virtual entities interact with each other through some medium which is provided by devices. Devices can be deployed near physical entities or attached to physical entities.
- Devices collects information about physical entities for example information can be collected using various sensors. Using actuators devices can perform actuation on physical entities. Devices can also used to identify physical entities using tags.
- In home automation system devices is minicomputer with light sensor and relay switch (actuator).

Resources

Q. Define "on-device Resources" and "Network Resources."

Resources are nothing but software components. Resources can be "on-device" resources or network-resources.

- **"On - device Resources":** They are hosted on device. On-device resources consist of software components that give information on physical entities to which they are attached. They also enable actuation on physical entities.
- **"Network Resources":** Network resources consist of software components that are available in network for example database.

In our example of home automation system operating system which runs on single board minicomputer is an 'on-device' resource.

Service

- Interface to interact with physical entity is provided by services. Resources hosted on a device or network resource are accessed by services in order to obtain information about physical entity and perform actuation on physical entities.
- Now consider our example of home automation system. There are three services which are as follows :
  (1) Service to set mode (auto or manual)
  (2) Service to set light state (on or off) or to find out current light state.
  (3) Controller service.

  Controller service runs as native service on device.
- In auto mode controller service checks the light level and switches the light 'on' or 'off' according to light level. Also updates status in database.
- In manual mode current light state is retrieved by controller service from database based on the retrieved current state if switches the light "on" or "off".

Syllabus Topic : Information Model Specification

2.2.4 Information Model Specification

Q. Explain information model application for smart home automation system.

Q. Explain information model specification for IoT based application.

- This is the fourth step in the IoT system design model where we define information model. Structure of all information like attributes of virtual Entities, relations, etc. is described in information model.
- It doesn't explain details about how information is stored or represented.
- In order to define the information model, we have to first list the virtual entities. After that by describing their attributes and relations information model adds more details to the virtual entity.

– In our example of home automation system there are two virtual entities. One is virtual entity for light appliance which has attribute light state. Another virtual entity is room which has attribute light level. Information model is shown in Fig. 2.2.4.

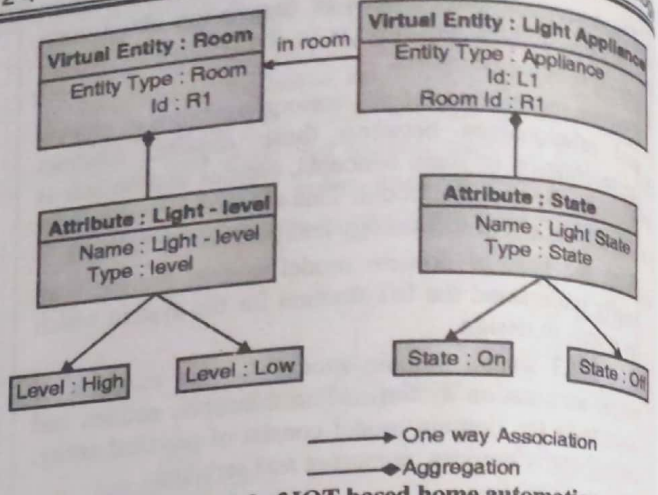

Fig. 2.2.4 : Information model of IOT based home automation system

Syllabus Topic : Service Specifications

2.2.5 Service Specifications

Q. Explain service specification step in IoT system design methodology.
Q. Describe various services in IoT system.
Q. Write service specification for smart home automation system.
Q. Write note on : (i) Mode service (ii) State service (iii) Controller service.

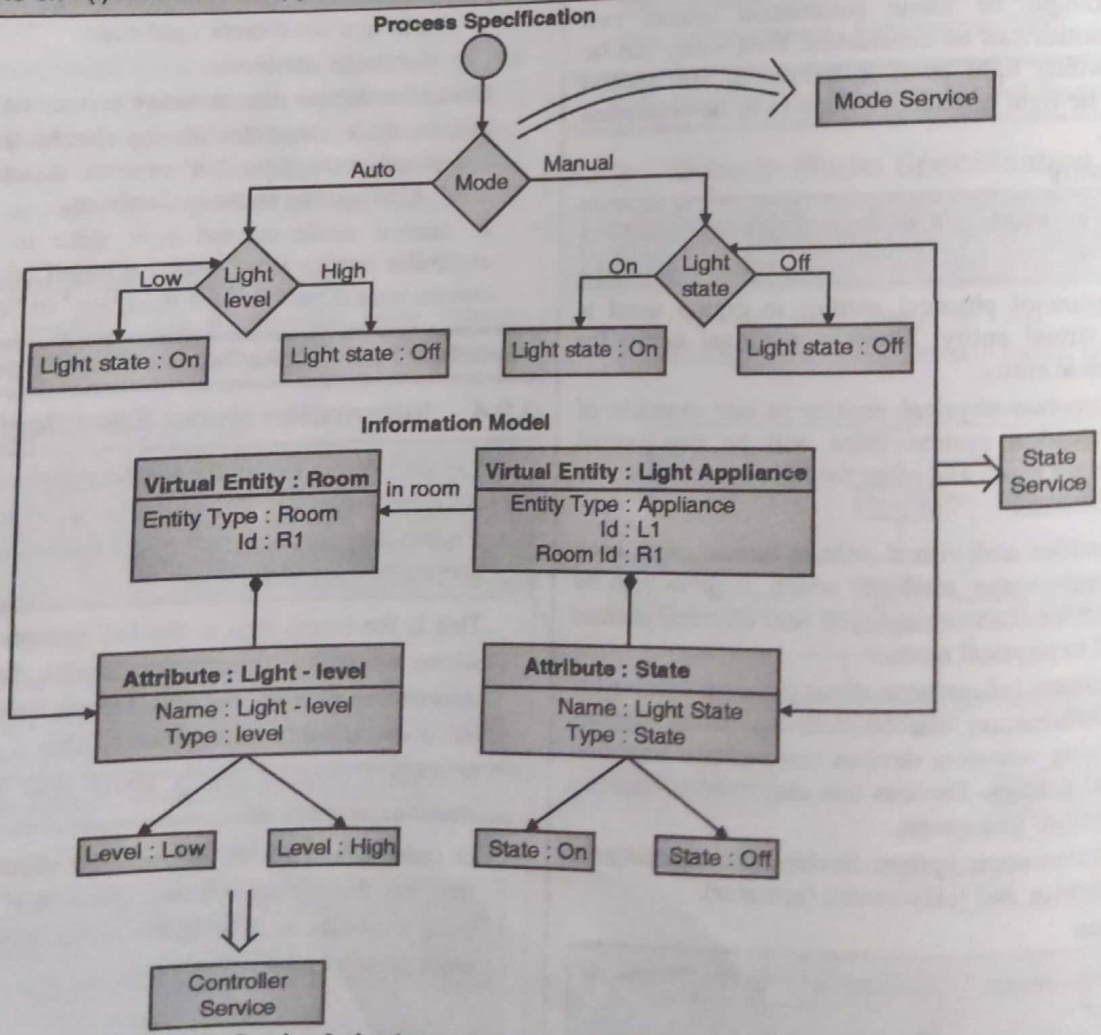

Fig. 2.2.5 : Service derivation from process specification and information model

- This is fifth step in design methodology where we define a service specification that is describing various services in IoT system.
- Service specification includes defining various services in the system, their service types, service inputs and outputs, service endpoints, schedules of services, service preconditions and service effects.
- For our example of home automation system we have made process specification and designed information model. Fig. 2.2.5 shows derivation of services from process specification and information model.
- Steps of deriving of services from specification and information model are as follows :
  - Identify states and attributes.
  - Define a service for each state and attribute.
  - That services can find out current values of state or attribute or can change that values.
- In our example of home automation mode service can retrieve current mode or can change mode from auto to manual or vice versa.
- In the same way state service can sets the light appliances to on or off or can retrieve its current state.
- Controller service can monitors the light level and can switch light on or off. Service can also update status to database.
- If mode is manual mode then controller service can retrieve current state from database and then can switch the light on or off.

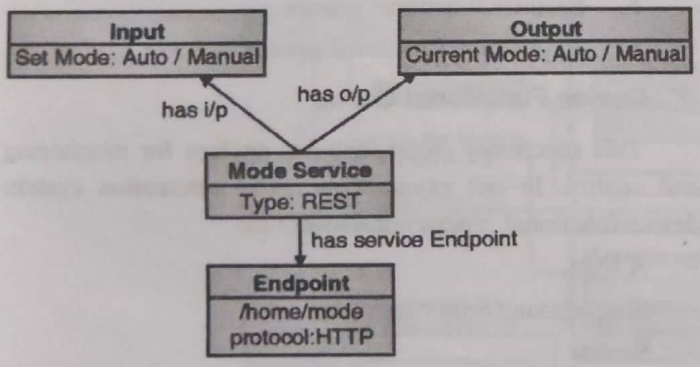

Fig. 2.2.6 : Mode - service specification

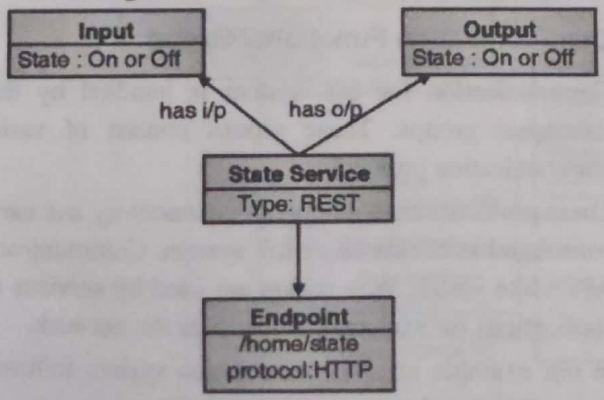

Fig. 2.2.7 : State Service Specification

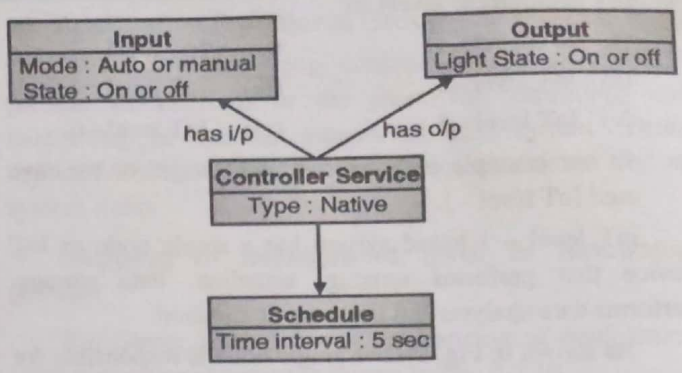

Fig. 2.2.8 : Controller service specification

- Fig. 2.2.6 shows specifications of mode service. Fig. 2.2.7 shows specifications of state service and Fig. 2.2.8 shows the specifications of controller service.

Some of the important details about mode service are:

- It is a RESTful web service.
- Service can set mode to auto or manual using PUT request.
- Service can retrieve current mode using GET request.
- Mode is updated to the database or can be retrieved from database by using mode service.

Some of the important details about state service are:

- It is also RESTful web service.
- Service can set state of light to on or off using PUT request.
- Service can retrieve current state using GET request.
- State is updated to the status database or can be retrieved from status database.

Details about controller service are as follows:

- Runs as a native service on the device.
- If mode is auto mode then this service checks the light level.
- According to light level service switches the light on or off and updates the status in the status database.
- If mode is manual mode then this service retrieves current state that is on or off from status database and switches the light appliance on or off.

Syllabus Topic : IoT Level Specification

2.2.6 IoT Level Specification

Q. State different IoT deployment levels. Explain IoT level specification for smart home automation system.

Q. Draw and explain deployment design for home automation.

- We have studied IoT deployment levels in unit – I.

- IoT deployment levels are
  (i) IoT level - 1. (ii) IoT level - 2.
  (iii) IoT level - 3. (iv) IoT level - 4.
  (v) IoT level - 5. (vi) IoT level - 6.
- In our example of home automation system we have used IoT level - 1.

IoT level – 1 based system has a single node or IoT device that performs sensing, actuation, data storage, performs data analysis and hosts the application.

As shown in Fig. 2.2.9 a single node is responsible for sensing, storage and data analysis.

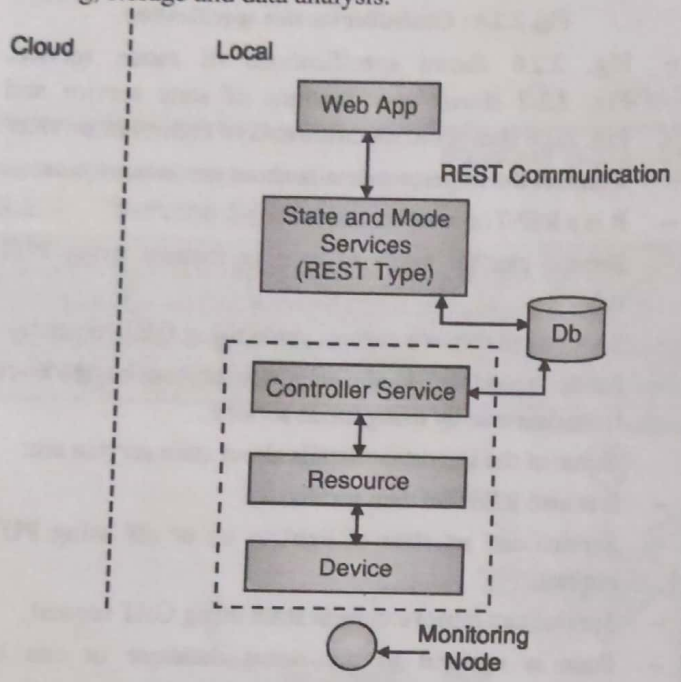

Fig. 2.2.9 : Deployment design of home automation system

IoT level – 1 based solution are used for low-cost low-complexity solutions in which data involved is less and data analysis and computational complexity is low.

- In our example of home automation system home appliances(lights) are controlled remotely via application.
- A relay is connected in a system with single node **IoT device** and lights and/or other home appliances.
- Status information of each connected home appliance is maintained via device monitoring in a **local database**.
- **REST service** is used to access and update the status of lights and appliances by changing the values in local database.
- The **controller service** continuously monitors the database and keeps track of the appliances and triggers the relay switches.
- The **application** which is deployed locally provides a GUI using which lights and other home appliances can be handled by the user.
- As the device is connected to the internet, application can remotely control the home appliances.
- Fig. 2.2.9 shows deployment design for home automation system.

Syllabus Topic : Functional View Specification

2.2.7 Functional View Specification

Q. Explain functional view specification step in IoT design methodology.
Q. Explain with diagram mapping of deployment level to functional group for home automation system.
Q. Write note on functional groups.

- This is the seventh step in IoT design mode methodology where we define functional view.
- Functions of IoT system are grouped into variou Functional Groups (FG). We define these functions o IoT system in functional view specification.
- Every FG that is functional group provides differen functionalities which are used for interacting with instances of concepts which are defined in Domain model or FG provides information related to these concepts.
- Functional Groups (FG) consists of :
  1. Device functional groups
  2. Communication functional groups
  3. Services functional groups
  4. Management functional groups
  5. Security functional groups
  6. Application functional groups

Device Functional Group

This functional group includes devices for monitoring and control. In our example of home automation system device functional groups consists of

- A light
- Single bound mini computer
- Sensor
- Actuator (relay switch)

Communication Functional Group

- Communication for IoT system is handled by these functional groups. These groups consist of various communication protocols.
- These protocols enable network connectivity and can be considered as backbone of IoT system. Communication API's like REST, Web socket are used by services and applications for exchanging data over the network.
- In our example of home automation system following protocols is used.

Link layer protocol : (802.11)
Network layer protocol : IPv4 / IPv6
Transport layer protocol : TCP
Application layer protocol : HTTP

- Communication API's are REST based in home automation system.

Service Functional Group

This functional group consists of following services

- Device monitoring services
- Device control services
- Device discovery services
- Data publishing services
- In our example of home automation system.
- There are two REST services.
- Mode service and state service and one native service that is controller service.

Management Functional Group

Functionalities which are required to configure and manage the IoT system are provided by management functional groups.

Security Functional Group

IoT system also requires security mechanisms security mechanisms like authentication, authorization and data security are provided by security functional group.

Application Functional Group

This functional group consists of applications that provide an interface to the users for controlling and monitoring of various aspects of IoT system. These applications allow users to view processed data as well as system status.

Mapping of deployment level to functional groups

For Smart home IoT system mapping of deployment level to functional groups is shown in Fig. 2.2.10.

- Devices which include sensors, actuators and communication devices are mapped to Device FG and device management (Management FG).
- Resources maps to Communication FG and Device FG.
- Controller Service maps to native Service that is to Services FG.
- Database maps to database management and database security that means database maps to Management FG and Security FG.
- REST web service maps to services FG.
- Application maps to web app, application server, database server that is to Application FG.
- Application also maps to application management and application security that means it maps to Management FG and Security FG.

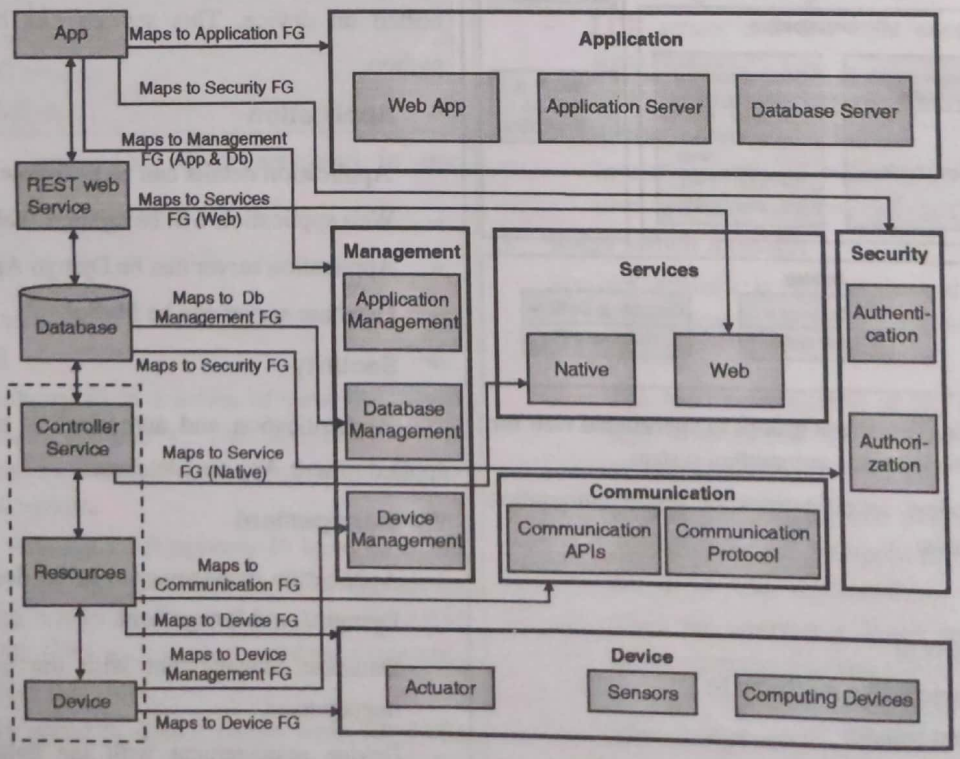

Fig. 2.2.10 : Mapping deployment level to functional groups for home automation IOT system

Syllabus Topic : Operational View Specification

2.2.8 Operational View Specification

Q. Explain operational view specification step from IoT design methodology.

Q. Explain operational view specification for IoT based home automation system.

This is the eighth steps in IoT design methodology where we define operational view specifications. Options and operations which are related to IoT system deployment are defined in this step.

These options are as follows:

- Device options
- Storage options
- Application hosting option
- Service hosting option

Mapping of functional groups to operational view for IoT based home automation system is shown in Fig. 2.2.11.

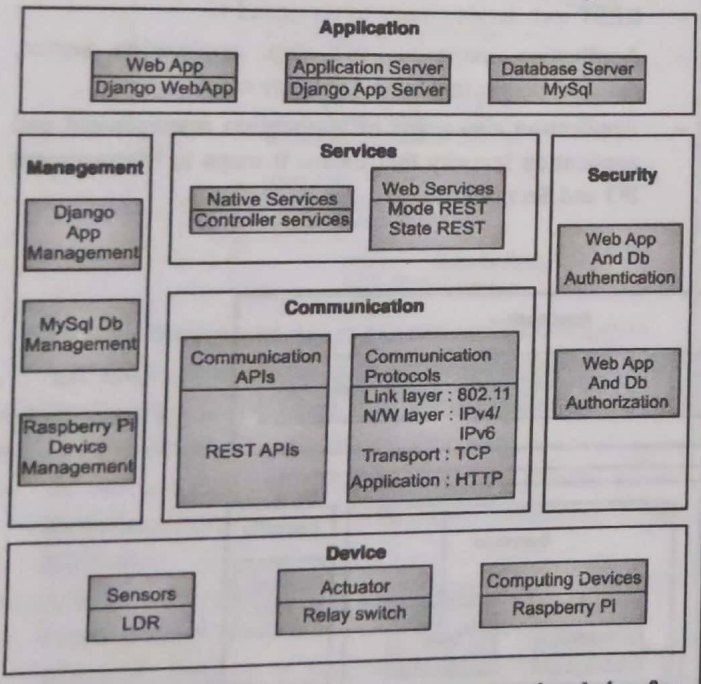

Fig. 2.2.11 : Mapping functional groups to operational view for IOT based home automation system

Operational view specifications of home automation system are as follows:

Devices

Devices consists of

- Computing devices like raspberry Pi
- Light dependent resistor
- Relay switch

Communication APIs

Communication APIs like REST APIs

Communication Protocols

Communication protocols are

- Link layer protocol - 802.11
- Network layer protocol – IPv4/ IPv6
- Transport layer protocol - TCP
- Application layer protocol – HTTP

Services

Services include mode service, state service and controller service.

Mode Service

It is RESTful web service which is hosted on a device. Service can be implemented using Django-REST framework.

State Service

It is RESTful web service which is hosted on a device. Service can be implanted using Django-REST framework.

Controller Service

Controller service is run as native service and it is hosted on device. This service can be implementing in python.

Application

Application details can be as follows

- Web application will be Django Web Application
- Application server can be Django App Server
- Database server can be MySql.

Security

Authentication and authorization mechanisms can be applied to web App and databases.

Management

- Application management can be done with the help of Django App Management.
- Database management with the help of Mysql DB management
- Device management with the help of Raspberry Pi device management.

Syllabus Topic : Device and Component Integration

2.2.9 Device and Component Integration

Q. Describe device and component integration for IoT based home automation system.

Q. Explain in short devices and components which can be used in IoT based home automation system.

This is ninth step in IoT design methodology where we can integrate devices and components. Schematic diagram of home automation system is shown in Fig. 2.2.12.

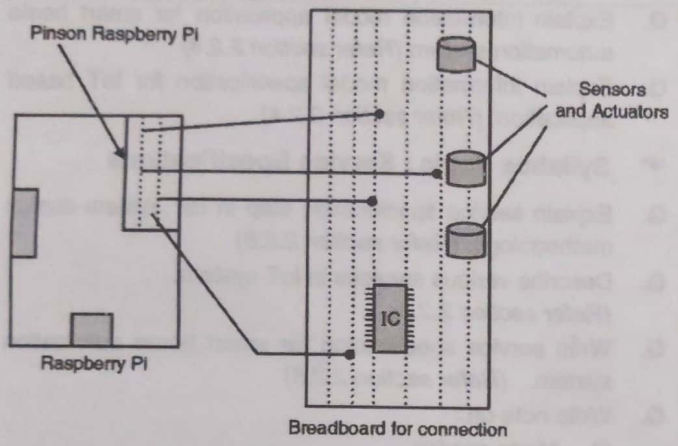

Fig. 2.2.12 : Home automation system with sensors and actuator integrated

Devices and components

Q. Write note on :
(i) Raspberry Pi
(ii) LDR sensors
(iii) Actuator

Devices and components which are used in our example are

1. Raspberry P_i
2. LDR sensor.
3. Relay switch actuator.

1. Raspberry Pi

- The Raspberry Pi is a series of powerful, small single-board computers. The Raspberry Pi is developed by Raspberry Pi Foundation in the United Kingdom.
- Various versions of Raspberry Pi have been out till date. All versions consist of a Broadcom system on a chip (SoC) with an integrated ARM compatible CPU and on-chip graphics processing unit (GPU).
- Processor speed of device ranges from 700 MHz to 1.2 GHz and memory range from 256 MB to 1 GB RAM.
- To store the operating system and program memory Secure Digital (SD) cards are used. Raspbian OS which is a Debian-based Linux operating system is recommended OS by Raspberry Pi Foundation. Some other third party operating systems like RISC OS Pi,Diet Pi, Kali Linux can also be run on Raspberry Pi.
- Fig. 2.2.13 shows Raspberry pi 3 model B.

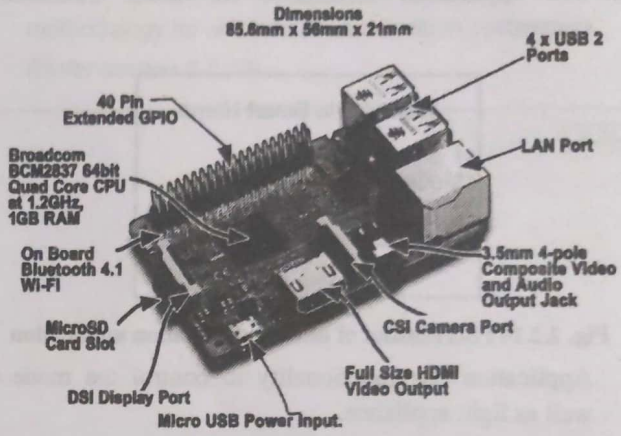

Fig. 2.2.13 : Raspberry pi 3 model B

2. LDR sensor

- LDR that is Light Dependent Resistor is a component which has variable resistance.
- Resistance changes depending on intensity of light falling upon it. This property makes LDR useful in various light sensing circuits.
- LDR sensor converts light energy into electrical signal output.
- LDR sensors generates output signal which measures intensity of light.
- In our example of smart home LDR sensors are used to measure intensity of light in a room.

3. Relay switch Actuator

- An actuator is the device which carries out the mechanical movements required for any physical process.
- The most basic control of any device is ability to turn it on or off. Best way to do it is by using switches.
- Relays are nothing but switches which can be turned on or off by application of a low voltage across the relay terminals.
- They are commonly found in automatic control applications as they can control equipment directly through electric signals and does not require physical operation.
- In our example of Smart home we are relay switch to turn light appliance on or off.

Syllabus Topic : Application Development

2.2.10 Application Development

Q. Explain application development step in IoT design methodology for smart home automation system.

- This is the last step in IoT design methodology. In this step we develop the IoT application. Fig. 2.2.14 shows web application screenshot for home automation system.

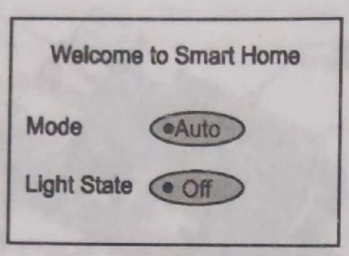

Fig. 2.2.14 : Screenshot of home - automation application

- Application has functionality to control the mode as well as light appliance.
- In auto mode based on light condition in room system will control light appliance (switch on or switch off).
- Light intensity is measured by LDR sensors which we have used. If light intensity is low (below specified limit) light appliance will turn on.
- If light intensity is high (more than specified limit) light appliance will turn off.
- If mode selected is manual then user can manually turn light appliance on or off.
- As the device is connected to the internet, application can remotely control the home appliances.

2.3 Exam Pack (Review Questions)

Q. Which are various steps in IoT design methodology ? *(Refer section 2.2)*

Syllabus Topic : Purpose and Requirement Specification

Q. Explain purpose and requirement specification in IoT design methodology with example. *(Refer section 2.2.1)*

Q. Consider any IoT based application and write purpose and requirement. *(Refer section 2.2.1)*

Syllabus Topic : Process Specification

Q. Explain process model specification for home automation system. *(Refer section 2.2.2)*

Q. Consider any IoT based application and write process model specification for same application. *(Refer section 2.2.2)*

Syllabus Topic : Domain Model Specification

Q. Write domain model specification for smart home automation system. *(Refer section 2.2.3)*

Q. Describe in short physical entity in domain model specification. *(Refer section 2.2.3)*

Q. Describe in short virtual entity in domain model specification. *(Refer section 2.2.3)*

Q. Define "on-device Resources" and "Network Resources." *(Refer section 2.2.3)*

Syllabus Topic : Information Model Specification

Q. Explain information model application for smart home automation system. *(Refer section 2.2.4)*

Q. Explain information model specification for IoT based application. *(Refer section 2.2.4)*

Syllabus Topic : Service Specifications

Q. Explain service specification step in IoT system design methodology. *(Refer section 2.2.5)*

Q. Describe various services in IoT system. *(Refer section 2.2.5)*

Q. Write service specification for smart home automation system. *(Refer section 2.2.5)*

Q. Write note on :
(i) Mode service
(ii) State service
(iii) Controller service *(Refer section 2.2.5)*

Syllabus Topic : IoT Level Specification

Q. State different IoT deployment levels. Explain IoT level specification for smart home automation system. *(Refer section 2.2.6)*

Q. Draw and explain deployment design for home automation. *(Refer section 2.2.6)*

Syllabus Topic : Functional View Specification

Q. Explain functional view specification step in IoT design methodology. *(Refer section 2.2.7)*

Q. Explain with diagram mapping of deployment level to functional group for home automation system. *(Refer section 2.2.7)*

Q. Write note on functional groups. *(Refer section 2.2.7)*

Syllabus Topic : Operational View Specification

Q. Explain operational view specification step from IoT design methodology. *(Refer section 2.2.8)*

Q. Explain operational view specification for IoT based home automation system. *(Refer section 2.2.8)*

Syllabus Topic : Device and Component Integration

Q. Describe device and component integration for IoT based home automation system.
(Refer section 2.2.9)

Q. Explain in short devices and components which can be used in IoT based home automation system.
(Refer section 2.2.9)

Q. Write note on :
(i) Raspberry Pi
(ii) LDR sensors
(iii) Actuator
(Refer section 2.2.9)

Syllabus Topic : Application Development

Q. Explain application development step in IoT design methodology for smart home automation system.
(Refer section 2.2.10)

❑❑❑

UNIT III

CHAPTER 3

Pillars of Embedded IoT and Physical Devices

Syllabus Topics

Horizontal, verticals and four pillars of IoT, M2M : The internet of devices, RFID : The internet of objects, WSN : The internet of transducer, SCADA : The internet of controllers, DCM : Device, Connect and Manage, Device : Things that talk, Connect: Pervasive Network, Manage : To create business values. IoT Physical Devices and Endpoints : Basic building blocks of an IoT device, Exemplary device : Raspberry Pi, Raspberry Pi interfaces, Programming Raspberry Pi with Python, Other IoT Devices.

Syllabus Topic
Horizontal, Verticals and Four Pillars of IoT

3.1 Horizontal, Vertical and Four Pillars of IoT

3.1.1 Horizontal and Vertical Applications in IoT

Q. What are horizontal and verticals of IoT applications ?

(A) Horizontal Applications

The applications based on IoT are spread across the globe widely providing solution to various problems. Many of such solutions are discussed in chapter 1's Applications topic. Here we'll try to figure out Horizontal, Vertical application development in IoT field.

- Horizontal applications Software Engineering terms are applications that provides solutions to many companies or users based on a common problem.
- Horizontal softwares are not business specific and can be used by multiple companies.
- It is easy to use and maintain.
- Some examples will be word processor, spreadsheets, financial software, web browsers, etc.
- Horizontal IoT model allows multiple providers to work together on a common interface. Horizontal approach assumes gateway and cloud services provide functionalities to be common, open and known to the developers.

Advantages

1 Robust,
2 Secure and high functionality oriented applications,
3 Higher possibility of more innovative application development,
4. Faster and less costly applications development.

Disadvantage

1. With increase in number of devices more cross linkage machine to machine communication is certain.
2. On contrary to horizontal applications, vertical applications are developed for a particular industry in mind. Vertical software is business specific.
3. Horizontal applications are more complex as compared to vertical applications.

- Examples are Enterprise Resource Planning i.e. ERP system, customer relationship management systems designed for an industry.

(B) Vertical Applications

- In **vertical IoT model**, IoT devices (nodes), gateways (coordinators) and cloud based services (we have seen in chapter 1) are provided and controlled by the same company.
- The vertical applications of IoT use the established technologies of the specific industry and commonly used features of network level for communication like wired and wireless communication, middleware architecture like three tired architecture to application level technologies like PaaS and service oriented architecture interface etc.

Advantages

1. Single point of contact,
2. No compatibility issues because other companies' devices are not involved.

Disadvantage

Completely dependent on a single vendor for modifications or upgrades.

- The common characteristic of both the models is to achieve the 5As and 3Is. That is 5As are anything, anywhere, anytime, anyway, anyhow and 3Is refer to instrumented, interconnected and intelligent.
- IoT based devices/objects refer to 3Is to provide services to the user in 5A manner.
- Information and Communication Technology (ICT) systems used to generate, gather and process data manually before the inception of IoT but IoT based systems enhanced this feature to be done automatically i.e. machine-generated data and cloud based storage and processing without human supervision.

3.1.2 Four Pillars of IoT

Q. Explain the four pillars of IoT in brief.

IoT based applications connect together smart devices and processes the data in the background. IoT applications are implemented in such a way that the underlying architecture and its working are unknown to the user. IoT rests upon four pillars that are used to connect devices using common infrastructure for communication, policies for gathering data and how it should be processed.

IoT gives these four pillars a unified way of representation. These four pillars of IoT are M2M (machine to machine), RFID (Radio Frequency Identification), WSN (Wireless Sensor Network) and SCADA (Supervisory Control and Data Acquisition). Let's see these four pillars in brief.

1. M2M

- M2M uses smart devices/objects to detect events and transfer it to other central devices (usually of higher processing capacity) which translate such sensed data to meaningful information.

Example

- WAN, GPRS, Cellular and Fixed Networks, etc.

2. RFID

RFID uses radio frequency to communicate between the tag attached on a device and RFID reader that identifies the unique RFID tag which can be used for identifying and tracking the implanted object.

Example

Radio waves, NFC, IC Cards, etc.

3. WSN

- WSN senses and gathers together data using sensors distributed spatially in the geographical region and collects it to a centralized location with the help of wired, wireless or sometimes hybrid network for processing.
- Usually WSN is used to detect physical and environmental changes like temperature, pressure, motion, heat etc.

Example

ZigBee, Bluetooth, Wireless Mesh Networks, etc.

4. SCADA

SCADA mainly focuses upon independent systems that work upon closed loop control systems that connect monitor and control equipment using a short range network inside a building or an industrial plant.

Example

BacNet, CanBus, Wired FieldBuses, etc.

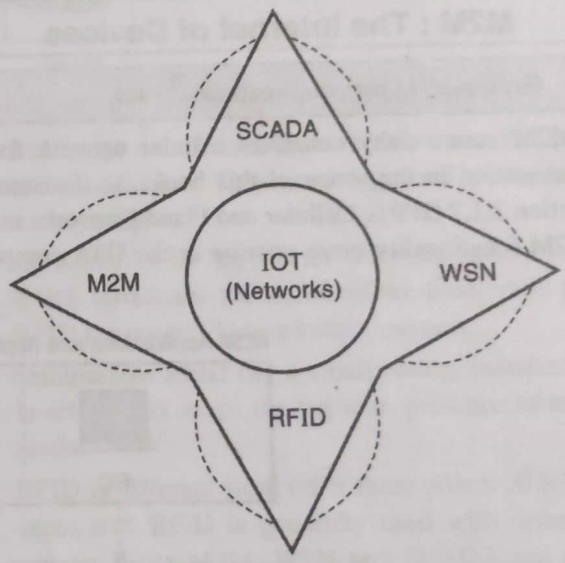

Fig. 3.1.1 : Four Pillars of IoT

- While using it may seem that each pillar is working as IoT but there are minute differences in the underlying architecture that separates the four pillar and makes IoT rest on top of the four pillars.
- The four pillars of IoT identify data gathering and communication terminology used in developing an IoT system. Communication in IoT applications can be for wired where devices are generally fixed / stationary or wireless where devices are movable. In addition to that it is categorized in short range communication and long range communication.

Difference between these four pillars

Q. Compare four pillars of IoT on the basis of communication Network.

- Table 3.1.1 will explain the network based difference between these four pillars.

Table 3.1.1 : Four Pillars of IoT and its relation with Network Communication

Communication Network → / Four Pillars of IoT ↓	Wired Connection		Wireless Connection	
	Short Range	Long Range	Short Range	Long Range
M2M	No	Some	Some	Yes
RFID	No	Some	Yes	Some
WSN	No	Some	Yes	Some
SCADA	Yes	Yes	Some	Some

- In the following section we'll discuss about the four pillars in details.

Syllabus Topic : M2M : The Internet of Devices

3.2 M2M : The Internet of Devices

Q. Explain M2M communication in details.

M2M connectivity considers cellular network for its communication in the scope of this book. As discussed in the section 3.1.2 GPRS, Cellular and Fixed Networks makes the M2M functionality more popular in the USA compared to IoT.

- M2M refers to products, services and device connectivity technology provided by cellular wireless networks which are operated by telecom companies.
- There are some consumer electronics services in place which follows M2M approach.
- M2M includes telematics (long distance transmission of data and information) and telemetry (reading and transmitting the sensed data from devices).
- As shown in figure, middleware placed on the server side is the backbone of M2M architecture.
- At the bottom of this architecture are the smart objects. These can be consumer electronic devices like fridge, AC, vending machines etc. or considerably small devices such as pet collar for pet tracker, personal medical devices, person tracker, etc. These objects can be accessed remotely using M2M framework.
- The next layer provides the communication and network solution for the smart devices deployed in the below layer. This layer consists of wired and/or wireless network adapters which are managed by gateway manager software. For accessing the smart devices remotely device specific protocols are used.
- The next layer is the brain of the whole M2M which is middleware. Middleware uses M2M specific communication protocols to collect data from the gateway via network and process it into useful information using business intelligence.

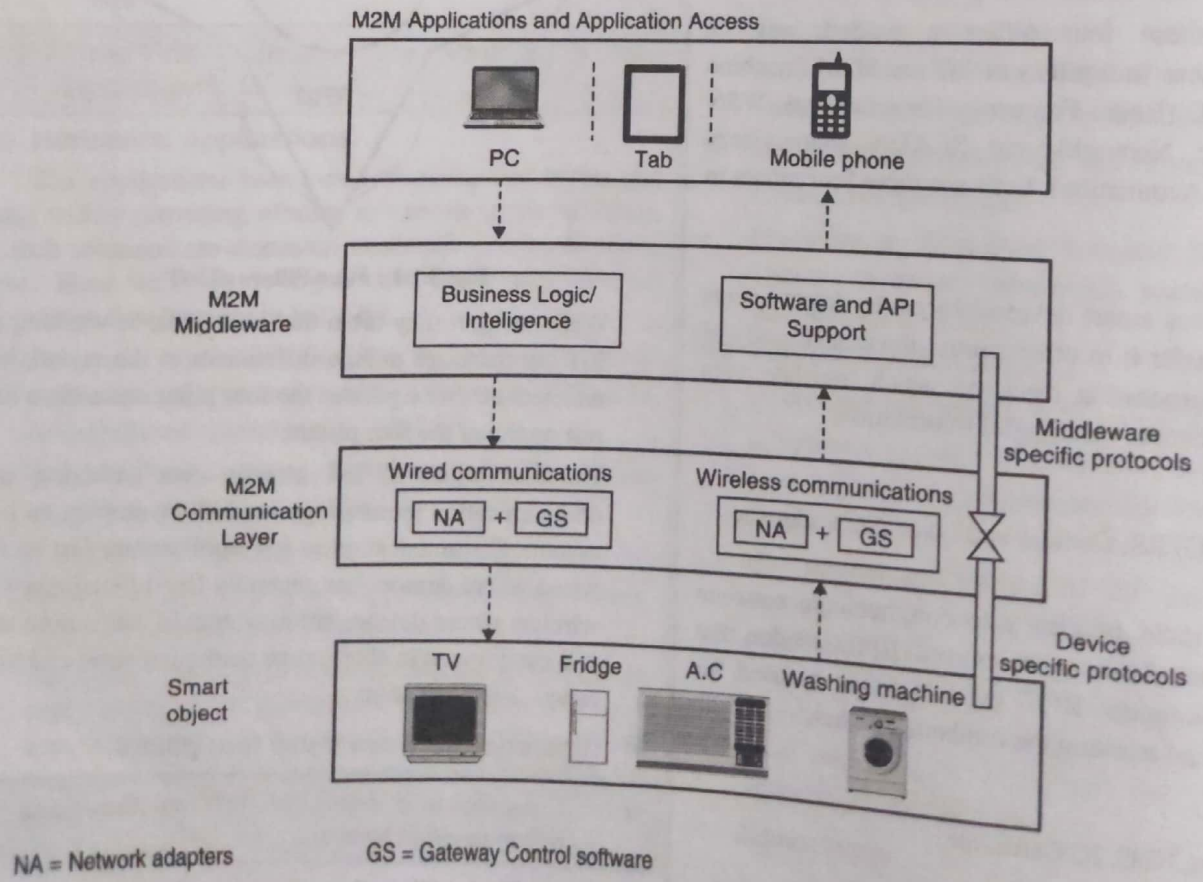

Fig. 3.2.1 : M2M Architecture

- Middleware also provides a platform to the developers for application development. It provides softwares and API support for application development.
- Because of middleware, developers don't need to worry about the data gathering and communication mechanism from smart devices to the middleware.
- The top layer which is the application layer is accessible to the user. It provides a platform to the users to access the smart devices remotely at touch of the fingertips.
- Mostly the applications developed are used to connect and communicate between devices personalized for specific verticals.

Examples of M2M applications (Applications for cellular M2M)

1. **Medical :** Wireless medical device can be used for remote patient monitoring.
2. **Transport :** Traffic control system can use devices to monitor traffic scenarios and regulate it efficiently.
3. **Industry :** Industrial automation of manufacturing process can help improve productivity and cost efficiency.
4. **Home automation :** Alarm and surveillance devices can keep track of home security in real time sense.
5. **Automobile :** Specially designed devices can be used for tracking and anti-theft alarm systems.
6. **Advertising :** Remote management of digital billboards can be updated remotely.

Syllabus Topic : RFID : The Internet of Objects

3.3 RFID: The Internet of Objects

Q. Explain RFID communication in details.

RFID uses radio frequency to communicate between the tag attached on a device and RFID reader that identifies the unique RFID tag which can be used for identifying and tracking the implanted object as we have already seen in section 3.1.2. Let's see in details how RFID is works internally and what the applications of it are.

- RFIDs are attached physically to the device which needs unique identification.
- RFID tags stores a serial number that identifies the object and some additional data can also be stored on the device along with the serial number according to the size limitations.
- It also has an antenna that is used to transmit this data from tag (on the device) to the reader.
- RFID provides low cost contact less identification of devices.
- When the device comes in range of the RFID reader equipment, readers read this data on the tag using the radio frequency even without the actual contact.
- As shown in Fig. 3.3.1, RFID systems contains a device implanted with RFID tag, a RFID reader, a middleware which can be used to specify business logic if any to process the data (for example payment of a vehicle on toll plaza) and the device that displays the processed RFID information which can be desktop laptops or an individual system.

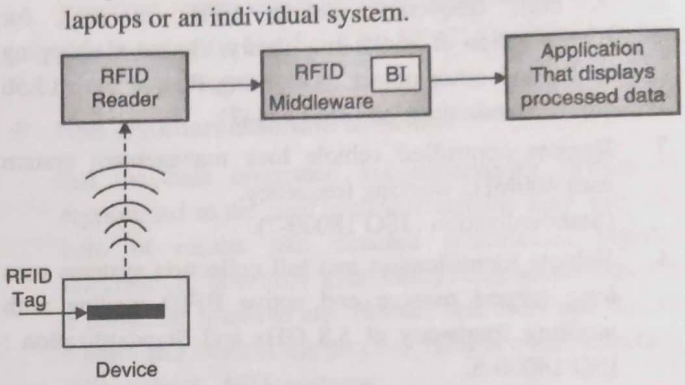

Fig. 3.3.1 : RFID Tag Communication

- RFID tags are of three types which are active, passive and semipassive. They are categorized on the basis of battery support.
- An active RFID tag has an on-board battery installed which broadcasts the signal all the time while passive RFID tag doesn't have a battery support.
- Semipassive RFID has a small battery installed which is active only when the tag is in presence of an RFID reader.
- RFID is different from other three pillars of IoT in the sense that RFID is generally used with unintelligent objects while M2M, WSN and SCADA are used for communication between intelligent objects.
- RFID technology is frequently used with RFID tags implanted on animals, cloths, cards, books etc.
- Before RFID barcodes and plain text were used for object and article identification.
- Universal Product Code (UPC) was used in USA and Canada for object identification and tracking.
- European Article Number (EAN), which was developed after UPC, was used in Europe and Ubiquitous ID (UID) was used in Japan for the same.

RFID concept is also used to develop new techniques like

1. Auto-ID is a mechanism that uses identification technology like RFID and manages automatic data capturing and storage of this information over the internet.

3. Electronic Product Code (EPC) is designed to be stored on RFID tag to provide unique identification for a product. (Refer section 1.6.4 chapter 1 for more detail)

Examples of RFID applications with standardization

1. A low frequency passive RFID is used for animal identification and tracking. (Working Frequency : 125 kHz, Standardization : ISO 18000-2)
2. A high frequency passive RFID is used for identification of books in a library, clothes at shopping malls and other objects. (Working Frequency : 13.56 MHz, Standardization : ISO 14443)
3. Remote controlled vehicle lock management system uses 400MHz working frequency. (Standardization : ISO 18000-7)
4. Vehicle identification and toll collection systems uses long ranged passive and active RFID reading with working frequency of 5.8 GHz and Standardization : ISO 18000-5.

Syllabus Topic : WSN : The Internet of Transducer

3.4 WSN : The Internet of Transducer

Q. Explain WSN communication in details.

Wireless Sensor Network senses and gathers together data using sensors distributed spatially in the geographical region and collects it to a centralized location with the help of wired, wireless or sometimes hybrid network for processing. Usually WSN is used to detect physical and environmental changes like temperature, pressure, motion, heat etc.

- WSN focuses on sensing and gathering data and routing it for further processing.
- The motive behind vast development in WSN was military and battlefield surveillance.
- WSN has multiple nodes ranging from few hundreds to thousands in number scattered throughout vast geographical region.
- Each of these nodes at least has a sensor attached to it obviously for the sensing purpose. Sometimes a single node has multiple sensors attached to it (temperature, pressure, motion, etc.).
- A node has an electronic circuit interface for the sensors inside, a microcontroller, a radio transceiver with antenna and battery as energy source.
- These node senses data and has a routing capability.
- This data is processed and provided to the end user in upper layers as shown in Fig. 3.4.1.
- Sensor nodes are usually connected to base station of sink that communicate between nodes and users.
- Intelligent logic can be provided here for processing or computing the data gathered by nodes.
- The processed data is converted into user representable information and provided to the end user via Internet connectivity.
- Routing protocols used in WSN are distributed and reactive which means nodes look for a route to transfer data only when some data is gathered or intermediate nodes have some data to be transmitted.

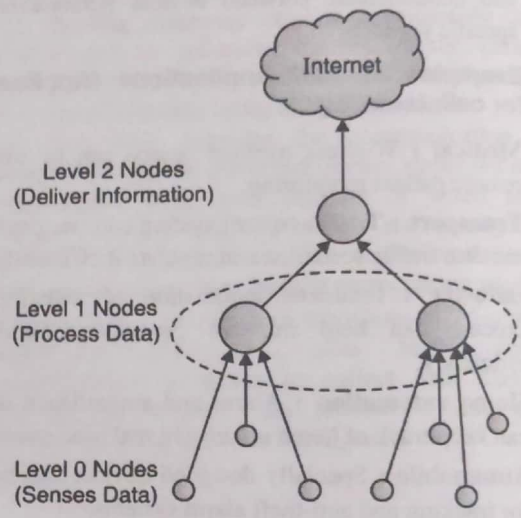

Fig. 3.4.1 : Simple WSN Architecture

- Ad hoc On demand Distance Vector (AODV) and Dynamic Source Routing (DSR) are used popularly for routing.
- The topology of WSN can be a simple star topology or multihop mesh network.
- The lifespan of a WSN depends upon the energy source of nodes as nodes are distributed over a large geographical area. So the algorithms and protocols used in WSN must be fault tolerant and robust to increase the lifespan of the node.
- Mobile Sensor Network (MSN) is a WSN in which nodes are mobile and can change location based on their own or due to environmental changes.
- The main difference between static WSN and MSN is the data routing mechanism.
- In static WSN the route is mostly predictable as the nodes in the system hardly changes but in MSN the number of nodes in the network at given point of time may not be the same later.
- Based on the number of nodes and their types the routing mechanism and routes can change.

- The development in WSN has led to the inception of WSAN i.e. wireless sensor and actuators network, which is capable of sensing environmental changes, processing this data and making decisions based on observations and performing appropriate actions.
- WSAN are used in battlefield surveillance, biological and chemical haphazard detection, climate control systems, nuclear power plants etc.
- Along with WSAN, one more extended version of WSN is becoming famous which is USN i.e. Ubiquitous Sensor Network.
- USN is considered as a network of smart devices/sensors which will become universal/global one day.

Examples of WSN

1. Forest fire detection
2. Weather monitoring systems
3. Manufacturing process control system in large industries
4. Hurricane, avalanche detection systems
5. Military surveillance etc.

Syllabus Topic
SCADA : The Internet of Controllers

3.5 SCADA : The Internet of Controllers

Q. Explain SCADA communication in details.
Q. Explain various components in SCADA communication architecture.

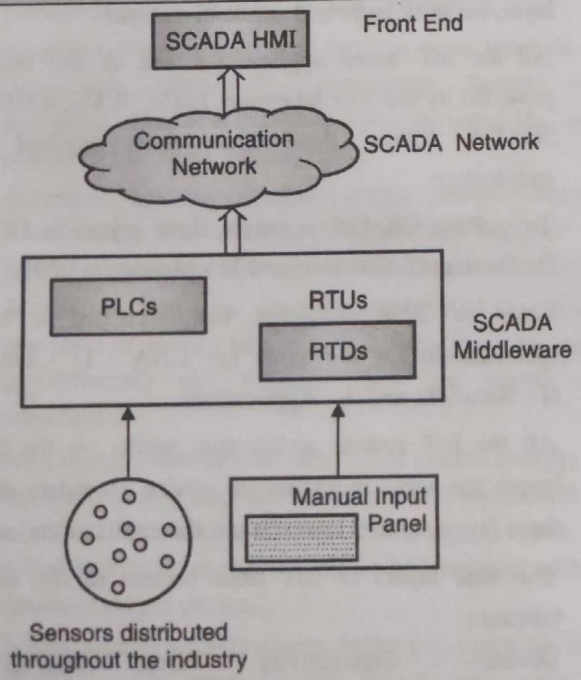

Fig. 3.5.1 : SCADA Architecture

SCADA mainly focuses upon independent systems that work upon closed loop control systems that connect monitor and control equipment using a short range network inside a building or an industrial plant.

- The foundation architecture of SCADA begins from PLCs and RTUs. PLCs and RTUs are the micro computing devices which communicate and coordinate with sensors and nodes.
- The SCADA software analyzes, distributes and generates results which are displayed over the HMIs.

As shown in the Fig. 3.5.1 the SCADA architecture has following key components:

HMI - Human Machine Interface

All the data processed via supervisory system is represented to the user in understandable way with the help of results like detailed schematics, logistic information, inventory availability, diagnostic data for a particular machine etc. Human operators use this to control and monitor the process. HMI is a user interface (UI) of the SCADA systems.

RTU - Remote Terminal Units

RTU receives the data from the sensors and this sensor signals are converted into digital data by RTU. This digital data is sent to the supervisory system for further processing. RTU are used for special type of sensors whose sensed data cannot be directly interpreted for processing. RTU provides an interface between SCADA and physical objects.

PLC - Programmable Logic Controller

PLCs are used alongside RTU to monitor and control the incoming data as they are flexible, configurable and economical compared to RTUs. PLC provide functionalities like motion control, relay control mostly functionalities providing local control.

Both RTU and PLC are electronic devices whose functionalities are alike but RTU is mainly focused towards specialized devices and objects (nodes) and PLCs are generalized.

One SCADA system can have multiple PLCs and RTUs that provides data control for SCADA systems.

DCS - Distributed Control Systems

DCS is a control system in which the control elements are distributed throughout the plant or industry. DCS has multiple local controllers distributes in various sections of the plant connected with high speed network. DCS is suited for large scale industries or manufacturing plants.

- The SCADA system is the middleware between the higher level large systems like ERP, Warehouse Management Systems (WMS), Enterprise Asset Management (EAM), etc. and lower layer RTU, PLC, DCS, Supervisory Information System (SIS) etc.
- A traditional SCADA based systems used to be client server architecture but now a days with modern IoT platforms SCADA works as a middleware oriented architecture.
- For example consider a SCADA system deployed in manufacturing plant. A notification (by SCADA middleware) pops on plant manager's Smartphone stating manufacturing in Production Line A is showing high error occurrences.
- Manager pauses the production, views the SCADA HMI to determine the cause of error. The manager reviews the data and figures out malfunctioning have occurred in machine 3.
- SCADA automates the complex industrial practices and processes where human control is not feasible.
- SCADA is being implemented as a core mechanism in many geographically separated services.
- Smart objects, strong communication network, improved intelligence (RTUs, PLCs) and add on availability of Internet and web technologies is making SCADA very popular in IoT.

Examples of SCADA

1. Manufacturing process in an industry
2. Industrial Control Systems
3. Electrical Power generation, transmission and distribution
4. Oil and Gas Industry

Syllabus Topic
DCM : Device, Connect and Manage

3.6 DCM : Device, Connect and Manage

Q. What is DCM with reference to IoT?
Q. What is DCM and DNA with reference to IoT?

- IoT based system has a long value chain that starts from the sensors, things, communication network, analysis platform like cloud and application. The value chain characterizes the service users, service providers which are the stakeholders. In the following section we'll discuss about the basic building block of IoT which is also called as the DNA of IoT systems.
- As discussed in first chapter section 1.8.1 the three basic building blocks are the sensors, objects, devices i.e. the hardware layer, the communication network layer which connects the devices and achieves the data transfer between devices and finally the application layer which represents the information processed in the underlying architecture.

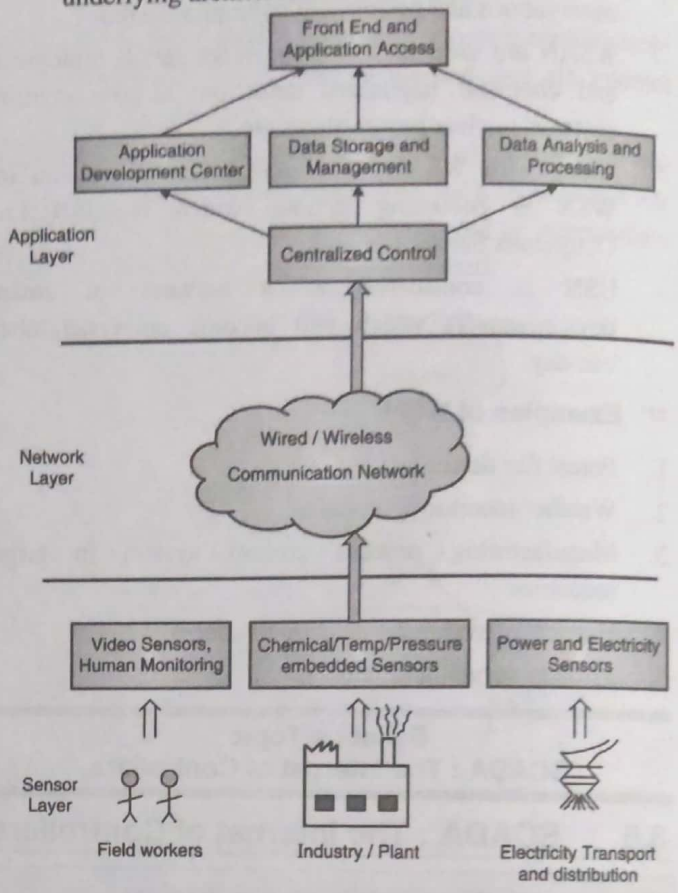

Fig. 3.6.1 : Three Layered IoT Architecture

- In this chapter the three layers are called as sensor layer, network layer and application layer.
- All the IoT based applications and all the services provided by the four layers i.e. M2M, WSN, RFID and SCADA follow this same three layered IoT architecture.
- Tong Fang Co. Ltd. acronym these layers as DCM : D - Device; C -Connect; and M - Manage in 2005.
- Later in 2008 Numerex Corp. created a better specification for the same i.e. DNA : D - Device, N - Networks and A - Applications.
- All the IoT system architecture works on the DNA layers but some industries or service providers divide these layers up to 9 layers as per the applications' need.
- The nine layers of IoT from bottom to top are as follows :

  Devices - Connectivity - Data Collection - Communication - Device Management - Data Rules - Administration - Applications - Integration.

M	– Data Management – Server Side/ Middleware platform – Vertical Applications	A
C	– Core Communication Network – Telecom Services – Pervasive Networks	N
D	– Sensors and Actuators – Smart Objects – Local Sensors in Network	D

Fig. 3.6.2 : DCM - DNA Mapping

Syllabus Topic : Device : Things that Talk

3.7 Device : Things that Talk

Q. What do you mean by "Things that talk" in IoT?

In the following section we'll discuss about the various devices available in the industry and its uses in details.

- As discussed earlier IoT devices can be categorized into two types one which have some form of intelligence inbuilt and those in which intelligence needs to be enabled.
- Devices like washing machine, fridge, AC etc. have some form of microcontroller logic inbuilt and can be used as a smart device.
- While for some devices or things like animals, books, furniture etc. techniques like RFID can be used to provide identification and more functionality.
- Sensors are the devices that perform input function or sense some change in surroundings. Sensors can recognize change in heat, pressure or force etc. and convert it into electrical signals.
- Actuators are the devices which perform output functionality like movement which can be used to control some external device attached to the actuators.
- Sensors and actuators are collectively used to convert one form of energy into the other form (may be same or different) and together they are known as transducers.
- For example, microphone takes voice (sound energy) as input converts it into electrical form and transfers it to amplifier which converts electrical signals back to sound and increase the volume to be produced over a speaker / output device.
- Sensors can be a RFID reader, M2M terminal node or a SCADA meter anything that senses data, send or receive data from same of different type of objects.
- That's why we refer devices as "things that talk".
- Sensors are small electrical devices which can be tracked or monitored, can be used to track objects once installed on them.
- Sensors can be installed on objects as small as a card or a smaller chip or as large as an aero plane.
- Sensors (fixed on devices) generally draws energy from battery power but alternate methods of energy harvesting where energy is derived from external sources like solar power, wind power, thermal energy, kinetic energy etc. is also possible.
- Sensors individually are considered as a part of IoT unless they are connected.
- Some sensors do not generate electrical signals like thermometer (mercury level). So selection of sensors depends on the requirements and the type of IoT application we are going to create.
- Large range of sensors are available for different types of measurements like pressure, temperature, etc.
- Table 3.7.1 shows some example of sensors and its uses.

Table 3.7.1

Sensor Type	Sensor Example
Temperature, Thermal, Heat	Bolometer, calorimeter, temperature sensor, thermocouple, etc.
Pressure	Barometer, boost gauge, pressure sensor, tactile sensor, etc.
Proximity, Presence	Alarm sensor, passive infrared sensor, proximity sensor, etc.
Optical, Light, imaging	Electro-optical sensor, infrared sensor, photoelectric sensor, etc.
Chemical	Oxygen sensor, carbon dioxide sensor, electrochemical as sensor, etc.
Automotive, Transportation	Crank sensor, vehicle speed sensor, wheel speed sensor, transmission fluid temperature sensor, etc.
Sound	Microphone, hydrophone, lace sensor, etc.
Flow, fluid	Flow sensors, gas meter, mass flow sensors
Navigation Sensors	Gyroscope, MHD sensor

Syllabus Topic : Connect : Pervasive Network

3.8 Connect : Pervasive Network

Q. What type of connections are available in IoT based systems ?

Q. Explain the term Connect in DCM in details.

Communication forms the backbone of the IoT based systems managing all the data transfer from device to device or device to a central collector. Communication technology is divided in two major types: wireless and wired (wireline). In this section communication networks is discussed in details.

- Both wired and wireless communications have categories like broadband or narrowband communication, short range or long ranged communication also packet or circuit switched communication.
- The connectivity domain allows wired as well as wireless broadband network.
- The communication network can use copper wires, optic fiber cables and air as a transmission medium to achieve connectivity.
- Data transfer is done in the form of small packets and are routed independently across the network.
- IPv4 is used to provide addressing mode for the devices in IoT systems. But the growth in devices is causing exhaustion in IPv4 (32 bit) addresses.
- To overcome this IPv6 (128 bit) is providing addressing support, but interoperability between IPv4 and IPv6 is not that simple.
- Wireless Communication is very important as three of the IoT's four pillars are based on wireless communication. (M2M,WSN,RFID)
- SCADA based short ranged wired fieldbus and long ranged TCP-IP is popular communication mechanism in industrial automation and building automation.
- China calls their communication network as "Triple Play", which focuses on three most used networks viz. Internet, Telecom Network and Cable TV broadcasting Network.

Let's discuss about the wired and wireless communications in details.

3.8.1 Wired Communication

Q. Explain Wired Communication term in detail (FieldBus).

- Wired communication for IoT based systems are mainly categorized into two types:
  1. Short ranged FieldBus based networks for most SCADA Applications
  2. IP based networks for some M2M and SCADA Applications

FieldBus

- The FieldBus is the family of industrial computer network protocol that allows real time distributed control of the nodes.
- FieldBus is made up of two words: Field and Bus. In this protocol standardization Field refers to limited geographical or contextual area in the industry where the IoT devices are distributed, and Bus refers to the electrical medium that carries data through it.
- FieldBus is standardized as IEC 61158 and IEC 61158 includes eight different protocol sets also known as type :
  - Type 1 : Foundation field bus H1
  - Type 2 : ControlNet
  - Type 3 : PROFIBUS
  - Type 4 : P-Net
  - Type 5 : Foundation field bus HSE (High Speed Ethernet)
  - Type 6 : SwiftNet
  - Type 7 : WorldFIP
  - Type 8 : Interbus
- Most of these uses twisted pair cable or fiber as a physical communication media, with various network topologies like Bus(PROFIBUS), Star(Foundation field bus H1), etc.
- Most of the Field Bus data transfer is done using small sized packets in a serial way.
- Field Bus provides bidirectional communication between the field instruments and HMI.
- Foundation FieldBus protocol was developed on the basis of OSI model, but it does not include all of the seven layers.
- Foundation field bus uses Physical Layer (OSI Layer 1), Data Link Layer (OSI Layer 2) and Application Layer (OSI Layer 7).
- Layer three to six are not described in the Field Bust standards.

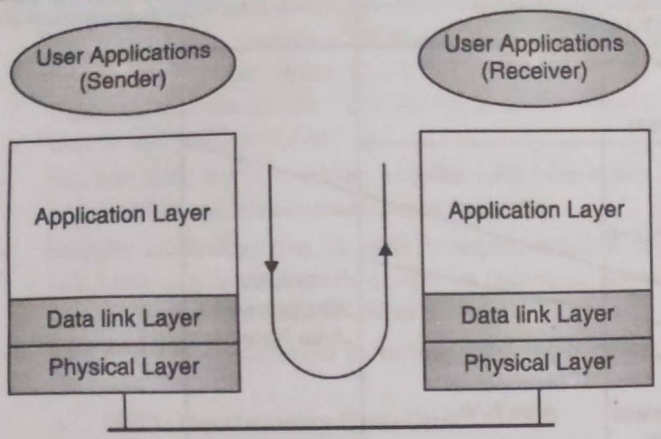

Fig. 3.8.1 : FieldBus 3 - Layered Structure

- General FieldBus Communication is shown in figure. FieldBus is mostly used for SCADA based applications.
- At the top we have SCADA/HMI that the operator can monitor, subsequently it is connected with middle layer PLCs using non time critical communications such as Ethernet.
- The bottom layer is the FieldBus that connects the middle layer PLCs to on field devices in the industrial environment such as sensors, actuators, valves, switches etc.
- The information in transferred over LAN or Internet based on the scale of application.

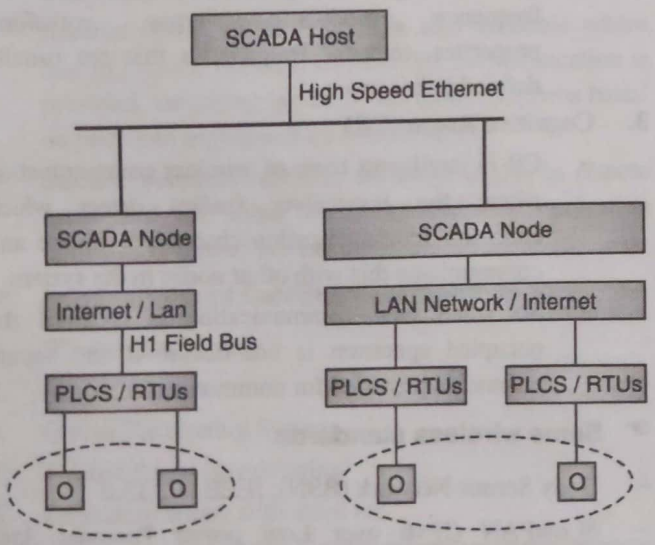

Fig. 3.8.2 : General FieldBus Communication

Some examples of FieldBus Standards

1. Automatic meter reading : Modbus
2. Building, Home automation : BACnet
3. Power System Automation : Profibus
4. Process Automation : Profibus, ProfiNet, Modbus

Internet Protocol Suit

- IP stack contains Physical Layer, Network Layer, Transport Layer and Application Layer.
- The Physical Layer manages the interconnection of nodes and hosts in the network. Protocols in this layer operate only on the links. Ethernet, LAN are some example of physical layer.
- The Network Layer (Internet Layer) manages sending of packets across multiple networks. Routing of packets from source node to destination node happens in the layer. Example: ICMP, IP, IGMP etc.
- The Transport Layer maintains end to end communication throughout the network. It also provides flow control, retransmission of packets, multiplexing etc. Example : TCP, UDP.
- The Application Layer manages application data exchange over the connection established by following layers.
- **Examples :** HTTP, FTP, POP3, SMTP, SNMP etc.

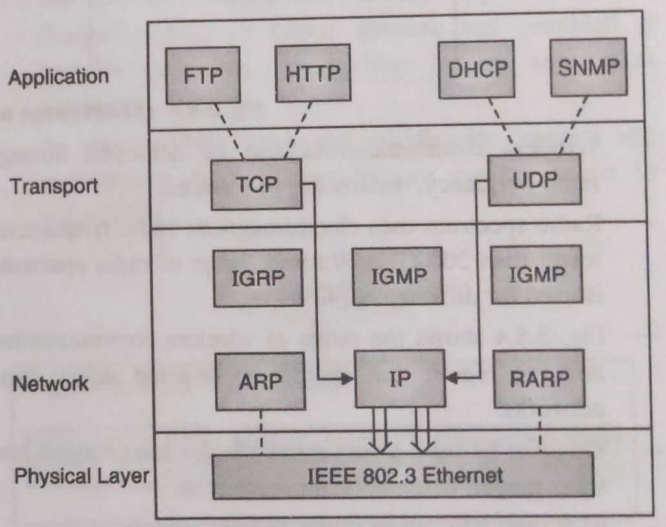

Fig. 3.8.3 : Internet Protocol Suit

3.8.2 Wireless Communication

Q. Explain Wireless Communication term in detail.

- Wireless communication for IoT based systems are mainly categorized into two types :
  1. Short ranged networks like RFID, Near Field Communication, WiMax, Wireless PAN, LAN, MAN etc.
  2. Long ranged networks like cellular network via GSM,CDMA etc., wireless WAN and satellite communication as well.
- WSN and RFID uses short ranged mesh network as its central communication technique.
- M2M uses long ranged cellular network as its core communication network.

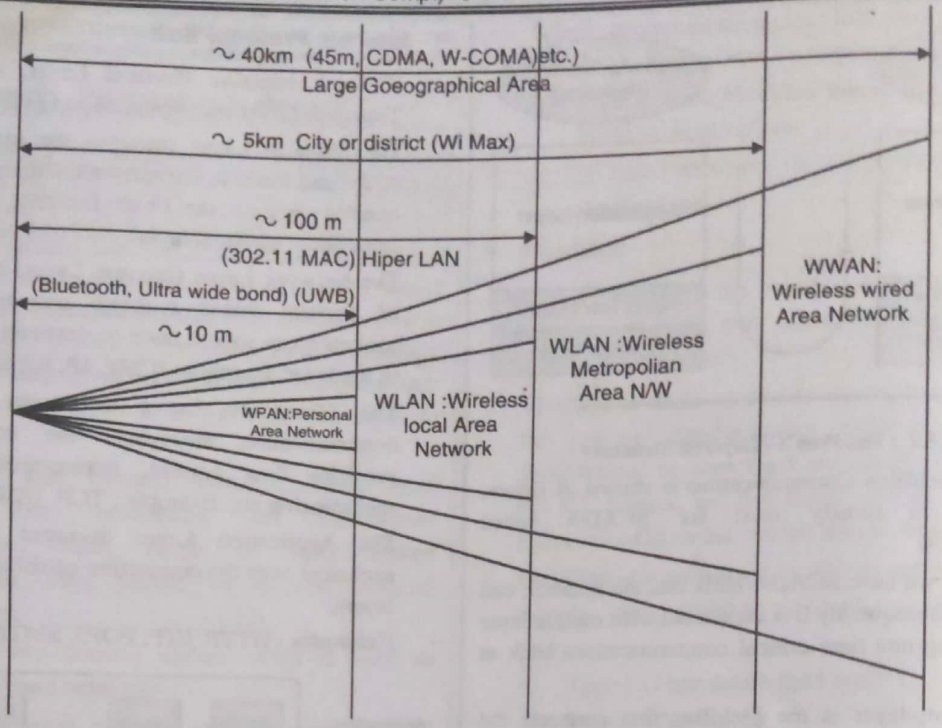

Fig. 3.8.4 : Short range and long range wireless networks

- Wireless communications can be achieved through radio frequency, microwave or infrared.
- Radio spectrum uses electromagnetic radio frequencies lower than 300 GHz. Various range of radio spectrum is used for different applications.
- Fig. 3.8.4 shows the range of wireless communication and the region that should be targeted using these networks.
- This can be used to set standards for long ranged and short ranged wireless communications.
- RFID and NFC come under WPAN region.

Wireless communication mechanisms

1. Ad hoc Sensor Network

- Ad hoc network provides connectivity to two or more mobile devices for a temporary basis without engaging vast infrastructure. Ad hoc network requires very less administration, low cost and can be deployed in remote geographical areas as well.
- Ad hoc based network can be integrated to a larger network like internet to increase the coverage area and scope of the ad hoc based systems.

2. Software Defined Radio (SDR)

- SDR is a full-fledged software implementation of the hardware component such as amplifier, filters, mixers, modulators/demodulators etc.
- SDR usually contains a personal computer and an analog to digital converters such as a sound card and a front end to access the functionalities. SDR provides a way to define the parameters like radio frequency, modulation types, waveform properties, transmit frequencies that are usually defined in hardware.

3. Cognitive Radio (CR)

- CR is intelligent form of wireless communication where the transceiver (radio) detect which spectrum (communication channel) is in use and communicate this with other nodes in the system.
- So when new communication is required the occupied spectrum is not disturbed and vacant channel is provided for communication.

Some wireless standards

- Body Sensor Network (BSN): IEEE 802.15.6
- 6LowPAN (IPv6 over Low power Personal Area Network) (Chapter 1 - Section 1.7.2)
- Digital Enhanced Cordless Communication (DECT)
- **HomeIR :** Wireless Infra-red Home networking
- **HomeRF :** Wireless radio frequency Home networking

3.8.3 Satellite IoT

Q. Explain satellite term in detail.

- Communication Satellite (COMSAT) is a specialized transponder in the space that transfers radio waves from one location to other location on th earth. This is also called as the bent pipe.
- Satellite industry is working together with the space organization and telecommunication industry.
- Satellite technology can be used in implementation of IoT based applications with a superior communication in remote areas compared to other networks.
- Satellites can be deployed in various orbit surrounding the earth :
  - GEO : Geostationary Earth Orbit
  - MEO : Medium Earth Orbit (GPS)
  - LEO : Low Earth Orbit
  - ELI : Elliptical Orbit
  - Molniya Orbit (High elliptical orbit)
- The two major challenges with satellite communication are speed and cost.
- Huge amount of one way data that can be transferred to a large antenna (Direct TV Broadcasting), but the two way communication with satellite and a small antenna is costly.
- So for two way communication cellular network provide better bandwidth and better costing solution than satellites.
- So satellite communication is effective for applications with small data and huge coverage.
- Another option of dual mode is also available where best of cellular network and satellite communication is provided, switching between two modes happens based on price and urgency of communication.
- Satellite communication is the most useful in remote areas and for ships where normal communication networks (cellular) are useless.

Applications of Satellite communication

1. Weather forecasting
2. Internet Access
3. Global Positioning System
4. TV and Radio broadcasting
5. Communications with ships etc.

Syllabus Topic : Manage : To Create Business Values

3.9 Manage : To Create Business Values

Q. Explain how business value can be created in Industry using in IoT based systems?

The previous two topics discusses how data is gathered and communicated throughout the IoT systems, in this section we will discuss how that data is managed and information can be used to provide opportunities in the business.

- Huge amount of data is generated by IoT based systems' devices but not all data is useful for processing, much of the data has low value or even considered as noise which must be avoided from the actual processing stages.
- High computing cloud based platforms are generally used to process this data into useful information.
- This processing is important in view of value creation and customer satisfaction, so IoT helps create this value to the industry.
- IoT based applications that can be linked to the existing platform of mobile operators and subscriptions provide a huge business opportunity to telecom operator.
- The use of smartphones running Apple's IOS and Google's Android based devices has provided a portable platform to various M2M application scenarios.
- As seen in Fig. 3.9.1, the middleware, control and automation services provide vast market scope to corporations.

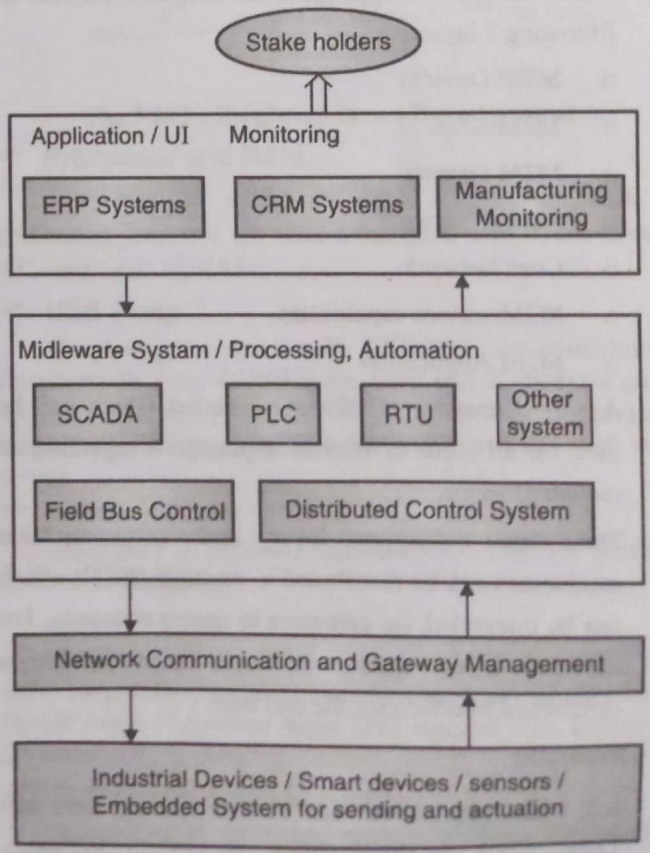

Fig. 3.9.1 : Industrial value chain

- The bottom layer manages smart devices for sensing data and actuation purpose. This layer includes smart devices, industrial devices etc.
- The next layer handles all the communication and exchanges data with middleware system which is responsible for processing and control of the information.
- Middleware can be built of SCADA, PLCs, RTUs, FieldBus, DCS etc.
- All these layers are working together to provide useful applications and value to the stakeholders.
- But these subsystems are not yet integrated fully, which is a major drawback of this scenario.
- For example, while automating an industry procedure, a factory database has to be manually fed to IT database, such task is tedious as the factory data will be vast.
- Complete combination of the on field objects integrating into a centralized SCADA system which provides support to the operator via suitable HMI. This is necessary for effective industry operation.
- Experts are required to provide solutions to such integration problems.
- For multinational companies or network operators for value creation IoT application development focuses on following 7 layers:
  - M2M Devices
  - M2M LAN
  - M2M Gateway
  - Access Network
  - Core Network
  - M2M service capabilities
  - M2M Application
- A single common platform is needed which can be used for all kinds of vertical applications regarding an industrial scope.
- The data collection layer and communication mechanism can be distributed to multiple WSNs which can be integrated via gateways to access networks, but the (top layer) application layer must provide a centralized interface for the end user.

Example

IoT based applications for Vehicle management can Lock/Unlock the doors, Track the vehicle, Remotely start/stop engines, Flash lights, Opens trunk etc.

Syllabus Topic : IoT Physical Devices and Endpoints - Basic Building Blocks of an IoT Device

3.10 IoT Physical Devices and Endpoints

Q. What is an IoT device ?

- As discussed in earlier chapter a "thing" or a device in IoT based system refers to any object that has a unique identity (and identifier) has ability to sense data (or provide method to sense data) and exchange this data over the network.
- IoT devices can connect to internet and send information about itself and surroundings over the network.
- Based on the data collected IoT systems also provide actuation capability to the physical entities in the environment, remotely.

Examples of IoT based systems

1. **Home automation System :** Remote access to the devices like lights, fan, fridge, etc.
2. **Industrial Automation :** A manufacturing line can send regular data about the quality and quantity of the products developed over the line.
3. **Vehicle Automation :** Location based services provides vehicle tracking and also remote vehicle access is also possible.
4. **Health Monitoring System :** Wearable devices can monitor health related parameters of a user and can inform the doctors about any haphazard.

In above examples additional use cases can be added as per requirement.

3.10.1 Basic Building Blocks of an IoT Device

Q. What are the basic building blocks of an IoT device ?

The basic IoT device must provide sensing and/or actuation, communication and processing capability using which a complete IoT system can be developed. Following are the key aspects which forms the building blocks of IoT device:

Sensing

An IoT device must be able to sense the surrounding data if not it must provide a way to integrate the sensor with the device. Sensing can collect data like temperature, pressure, light, chemical activity etc. This collected data can be communicated with other devices or cloud based services for storage or further processing.

Actuation

Actuators provide a way to take action in the physical world based on the decisions taken by the IoT based systems. For example a relay can be connected to an IoT device which can control turning the switch on/off based on application logic.

Communication

Communication module handles sending and receiving of data form device to device or device to cloud. Different communication protocols are used as seen in chapter 1.

Analysis and Processing

The collected data from devices are analyzed and processed into understandable information. This module specifies the logic to take actions based on the data gathered by the devices.

- The use of Single Board Computer (SBC) based IoT device is very popular these days.
- SBC like Raspberry Pi, Beagle Bone Black, Arduino etc. are used for development of IoT based systems.

Following figure show basic building blocks of an IoT device and SBC :

- A standard SBC includes a core CPU which performs the main computation task and an additional GPU as per need for extensive graphical processing.
- Primary and secondary storage blocks. Primary storage i.e. RAM manages the process execution while secondary storage handles the data storage.
- External peripheral connectivity i.e. input output peripherals and audio - video connectivity and networking support is also provided in SBC.

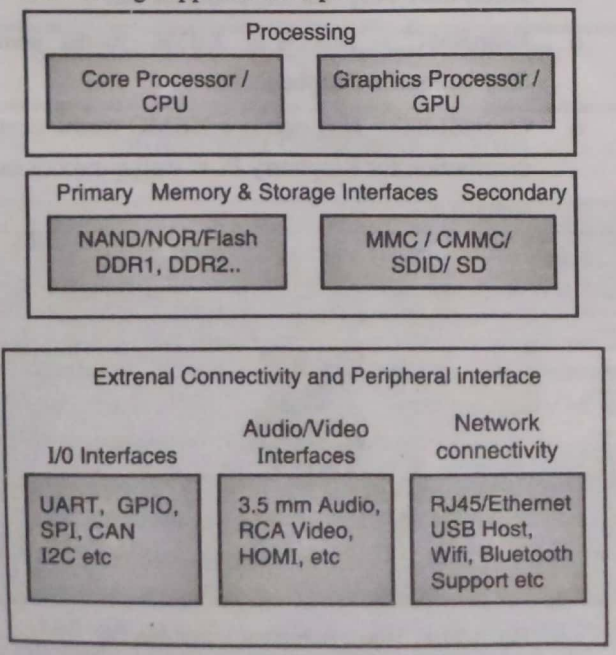

Fig. 3.10.1 : IoT Device Block Diagram

Syllabus Topic : Exemplary Device - Raspberry Pi

3.10.2 Raspberry Pi

Q. Explain Raspberry Pi in brief/ in details.
Q. Explain Pin Configuration of Raspberry Pi in brief.

Raspberry Pi is a mini computer having a size of a credit card. It functions like a normal computer does and needs to be connected with external input output devices (mouse, keyboard, display). Raspberry Pi runs Linux based operating system and has dynamic support to Python (Other programming languages as well as Linux supports other languages as well). Raspberry Pi allows connection of sensors and actuators by the means on board of general purpose input output pins. Raspberry Pi 3 is the latest version in the market. Let's see some details about Raspberry Pi :

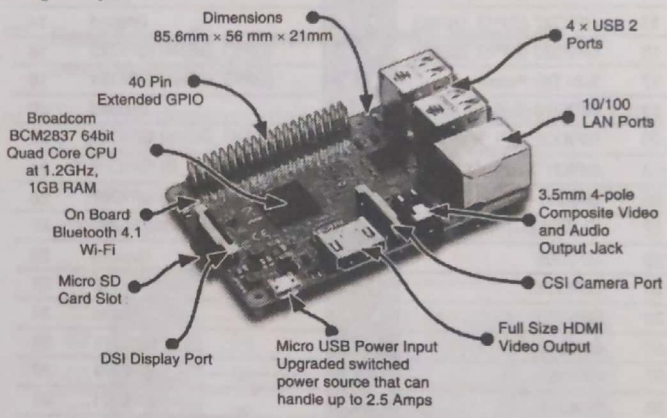

Fig. 3.10.2 : Raspberry Pi and Its components

Processor and RAM

Raspberry Pi works on ARM based processor, a processing power of 1.2 GHz 64bit quad core ARM Cortex A53 and 1 GB SDRAM.

USB Ports

Raspberry Pi comes with USB 2.0 port availability. Raspberry Pi 3 has 4 USB ports. USB hub is always a good option whenever number of USB ports requirement increases. USB port on Raspberry Pi provides 100 mA power

Ethernet Ports

Raspberry Pi provides standard RJ45 Ethernet port for network connectivity.

HDMI Output

Raspberry Pi has HDMI support for both audio and video output. Display device can be connected directly using HDMI cable if monitor have DVI support then HDMI to DVI converter is required.

Composite Video Support

Raspberry Pi provides composite video support via RCA (Radio Corporation of America) jack which can connect to both display standards PAL (Phase Alternation

by Line) and NTSC (National Television System Committee) video. Old television sets which have only RCA support can be directly connected to Raspberry Pi.

Audio Output

Raspberry Pi provides 3.5 mm audio jack support. The quality of 3.5 mm jack is inferior to HDMI support.

GPIO pins

Raspberry Pi has 40 pins onboard out of which four types of general purpose input output pins are available viz. true GPIO pins, SPI pins, I2C pins and Serial Rx and Tx pins. Fig. 3.10.3 shows pins distribution on Raspberry Pi 3.

Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5V	02
03	GPIO02 (SDA1, I2C)		DC Power 5V	04
05	GPIO03 (SCL1, I2C)		Ground	06
07	GPIO04 (GPIO, GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD(I2C) ID EEPROM		(I2C ID DEEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Fig. 3.10.3 : Raspberry Pi Pin Configuration

- Each GPIO pins is user configurable and the number associated with pin number is important in coding point of view. As seen in figure out of 40 pins pin number 12 is named as GPIO18 which means whenever we need to configure pin number 12 on Raspberry Pi, we have to refer it with number 18. This referencing is known as BCM Mode i.e. Broadcom SOC Channel. How this is done is explained in section 3.10.4.

Display Serial Interface (DSI)

DSI support is available in Raspberry Pi for LCD panel connection.

Camera Serial Interface (CSI)

CSI support is available in Raspberry Pi for camera module connection.

Status LED

Raspberry Pi (different models have different number of LEDs) has 5 types of LEDs on board. The functioning of these LEDs are as follows:

Status LED Name	Functionality
PWR	3.3V power supply
ACT	SD card access
LNK	Link or Network Activity
FDX	Full Duplex LAN Connectivity
100	100 Mbit LAN Connectivity

SD Card Slot

Raspberry Pi has SD card support which can be used for additional storage requirement and importantly booting a new operating system on Raspberry Pi is done via SD card support.

Power Input

Raspberry Pi has micro USB power adapter connection and 5V power supply is essential.

Operating System Support

- As discussed earlier Raspberry Pi supports various flavors of Linux based OS, let's take a look at some:
  - **Raspbian** : Raspbian is a Debian based operating system specially developed for Raspberry Pi. This OS is mostly recommended for use with Raspberry Pi.
  - **Arch** : Arch Linux flavor for AMD based devices.
  - **Pidora** : This is Fedora based OS developed for Raspberry Pi.
  - **RISC OS** : Very fast and compact OS.
  - **RaspBMC** : This is a XBMC media center distribution for Raspberry Pi.
  - **OpenELEC** : This one is a XBMC media center distribution for Raspberry Pi as well but more user friendly.

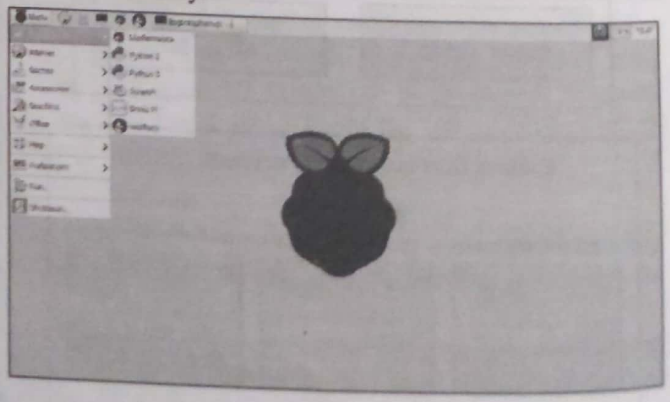

Fig. 3.10.4 : Home Screen of Raspbian OS

Syllabus Topic : Raspberry Pi Interfaces

3.10.3 Raspberry Pi Interfaces

Q. What kind of interfaces are available with Raspberry Pi ?

Raspberry Pi has 40 onboard pins and its distribution is discussed earlier. Let's see the use of serial, SPI and I2C interface available with Raspberry Pi.

Serial

Serial interface on Raspberry Pi can be used for serial communication with serial peripherals and provides Rx (receive) and Tx (Transmit) Pins.

I2C (Inter Integrated Circuit)

I2C is provides a way to connect hardware devices to Raspberry Pi module. I2C permits synchronous data transfer with the help of only two pins one for clock (SCL) and other for data (SDA).

SPI (Serial Peripheral Interface)

SPI provides synchronous serial data transfer with one or more peripheral devices connected. In SPI connection there is usually one master device and one or more slave devices. SPI uses five pins to provide such connectivity:

- MISO (Master In Slave Out) : Master line working for sending data.
- MOSI (Master Out Slave In) : Slave line working for sending data.
- SCK (Serial Clock) : Clock signals by master for synchronous data transfer.
- CE0 (Chip Enable 0): To enable or disable devices
- CE1 (Chip Enable 1): To enable or disable devices

Syllabus Topic : Programming Raspberry Pi with Python

3.10.4 Programming Raspberry Pi with Python

Q. Explain in brief about Python Coding for Raspberry Pi.

Q. Explain LED interfacing with Raspberry Pi. Handle the components using commands only.

Raspberry Pi has a support of many programming languages but we will be targeting Python because of its vast library and higher community support.

- As discussed previously Raspberry Pi provides GPIO pins which can be used for connecting input or output devices with the board.
- Each GPIO pins is user configurable and the number associated with pin number is important in coding point of view.
- This referencing is known as BCM Mode i.e. Broadcom SOC Channel referencing.

Example

Pin Number on Raspberry Pi	GPIO Number
12	18
16	23
18	24

- The GPIO numbers will be used while configuring Pin Number on Raspberry Pi. In BCM mode the numbering might change from RPi model to model.
- There is other mode available in Raspberry Pi i.e. BOARD Mode where the pin number can directly be used as GPIO number which is considerably simple.
- In BOARD mode the numbering generally doesn't change from RPi model to model.

Raspberry Pi connectivity with external component (LED)

- For this demonstration, we need a breadboard, LED light and few jumper wires and Raspberry Pi.
- A LED has two terminals one is to be connected to ground and the other to the GPIO pin.

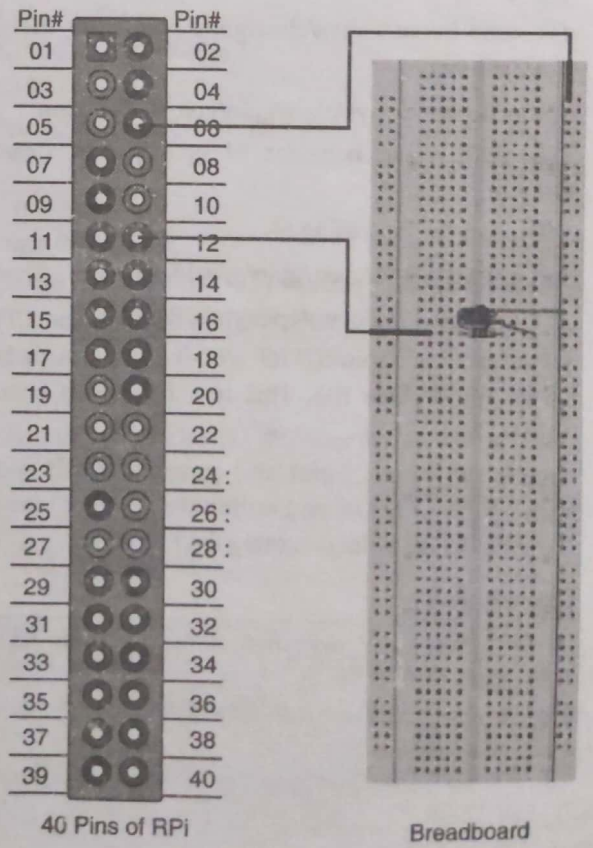

Fig. 3.10.5 : Breadboard Connectivity with Raspberry Pi

- Connect LED's longer leg on the breadboard and use the jumper wire to connect to the Ground (GND) pin number 6 on Raspberry Pi.
- Connect LED's shorter leg on the breadboard with a resistor (preferably 13K) and use a jumper wire to connect this to GPIO pin on the Raspberry Pi. In this example we have selected pin number 12 on Raspberry Pi (GPIO18).
- The final connection specifies that one end of LED is grounded and the other is provided with a GPIO connection.
- The final circuit looks like as in the Fig. 3.10.5.

Handling LED connected to Raspberry Pi using Console and commands

- Each GPIO pin is numbered as shown in the figure.
- Export pins that you want to use for interaction using following commands:
  - echo GPIO_PIN_NO > /sys/class/gpio/export
  - echo 18 > /sys/class/gpio/export
- This initializes the Raspberry Pi Pin 12 for using as input or output.
- Set directions for pins defining whether that pin is used for input or output:
  - echo in/out > /sys/class/gpio/ gpioGPIO_PIN_NO / direction
  - echo out > /sys/class/gpio/gpio30/direction
- This defines the behavior of gpio pin as Input or Output.
- Set values for pins using :
  - echo 1 > /sys/class/gpio/gpio18/value
  - echo 0 > /sys/class/gpio/gpio18/value
- "/sys/class/gpio/gpio<*GPIO_NO*>/value" is a path for GPIO pin's value file. This is a file which contains value 0 or 1.
- Setting the value of pins as 1 provides HIGH voltage turning the LED ON and setting the value of pins as 0 provides LOW voltage turning the LED OFF.

Sample Run

```
$ echo 18 > /sys/class/gpio/export # Initializes the RPi pin (GPIO number)
$ echo out > /sys/class/gpio/gpio30/direction # Set the device as output device
$ echo 1 > /sys/class/gpio/gpio18/value # Specifies HIGH_VOLTAGE
#Turns ON the LED
$ echo 0 > /sys/class/gpio/gpio18/value # Specifies LOW_VOLTAGE
#Turns OFF the LED
```

Handling LED connected to Raspberry Pi using Python Code

Q. Explain LED interfacing with Raspberry Pi with the help of Python Code.

- Python provides a package that is used for initialization of Raspberry Pi pins, specifies the type whether input or output and activates HIGH or LOW voltage in them. The name of that library is *RPi.GPIO*
- RPi.GPIO library has many functions that help in implementation of IoT applications.
- GPIO.setmode() : Specifies the mode of pin configuration to be BCM or BOARD.
- GPIO.setup() : Specifies the Raspberry Pi pin to be used as an input device or output device.
- GPIO.output() : Specifies HIGH or LOW voltage on parameter pin number.
- GPIO.input() : It has pin number as parameter and reads the status of the pin and returns True or False based on status of pin.
- GPIO.cleanup() : Cleans up all the pins,ports set in the code.
- Another function of time library is used that put the execution in wait state for prescribed time period.

Sample Code

```
import RPi.GPIO as GPIO #Library for Raspberry Pi GPIO pins
import time #Library for time functions
GPIO.setmode(GPIO.BCM) # BCM mode
GPIO.setup(18,GPIO.OUT) #Output device connected to GPIO18
while True:
GPIO.output(18,True) #Turns LED ON
time.sleep(1) #Sleeps for 1 sec
GPIO.output(18,False) #Turns LED OFF
time.sleep(1)
GPIO.cleanup()
```

Example

Q. Explain LED and switch interfacing with Raspberry Pi with the help of Python Code.

Python code for LED and Switch interfacing with Raspberry Pi.

LED should be turned ON when the switch is pressed. As seen in above example in the same way LED and Switch can be connected to Raspberry Pi.

- Switch is an input device and LED is output device.
- In the above connection add one switch on the breadboard and choose a GPIO pin on Raspberry Pi to be connected to the switch.

- For example assume pin number 16 – GPIO23 is connected to LED and pin number 18 – GPIO24 is connected to switch. Other connection like resistor and GND are assumed to be done.
- According to the logic GPIO23 is to be set as OUTPUT and GPIO24 is to be set as INPUT.
- So in switch is pressed LED should be turned ON otherwise it should be OFF.

Sample Code

```
import RPi.GPIO as GPIO #Library for Raspberry Pi GPIO pins
import time #Library for time functions
GPIO.setmode(GPIO.BCM) # BCM mode
GPIO.setup(23,GPIO.OUT) #LED connected to GPIO23
GPIO.setup(24,GPIO.IN) #SWITCH connected to GPIO24
while True:
if(GPIO.input(24)): # If GPIO24 is True i.e.HIGH Voltage at GPIO24
print "Port 24 is HIGH / 1 / True"
GPIO.output(23,True) #Turns ON LED
else:
print "Port 24 is LOW / 0 / False"
GPIO.output(23,False) #Turns OFF LED
time.sleep(1)
GPIO.cleanup()
```

Example

Q. Explain LED and LDR sensor interfacing with Raspberry Pi with the help of Python Code.

Python code for LED and LDR i.e. Light Sensor interfacing with Raspberry Pi.

Till now the connectivity of devices with Raspberry Pi is well explained. In this example we will add one new component to the circuit, a LDR sensor.

- Light Dependent Resistor is used to detect the light levels in the surroundings. In the dark LDR sensor has resistance in the range of mega ohms as the light level increases the resistance decreases (and reaches within few hundred ohms range).
- To measure up the resistance the circuit needs a capacitor, which charges up through the resistor. The capacitor charges when receiving power and discharges otherwise.
- So for this circuit we need LDR sensor, LED, capacitor and few jumper cables.
- Connect one end of the LDR to 3.3V on RPi i.e. pin number 1 and the other one to RPi Pin 16 i.e. GPIO23, 1µF capacitor is also attached between this LDR and GPIO23. Connect LED's one end to RPi Pin 18 i.e. GPIO24 and other to GND. GND connection for capacitor and LDR is assumed to be done.
- Set the LED as output device. We will set the LDR to output device to initialize it to 0 and the change it to input device for measuring Light (measuring resistance via capacitor).
- Function *readLDR()* returns count value in proportion to the light level sensed.
- After initialization, as the LDR starts sensing light, it starts charging the capacitor (as light increases resistance decreases) and in turn increasing count.
- The count increments until the input read HIGH. This is done when the capacitor charges greater than 1.4V.
- The count is directly proportional to the light level, greater the light in the surrounding greater the count.
- Smaller the LDR resistance more time is taken to charge the capacitor.
- Once counter is returned it can be compared with a threshold value indicating required light and if count is less than threshold value LED is turned ON.

Sample Code

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
ldr_thr=1000
#GPIO23 LDR RPi Pin 16
#GPIO24 LED RPi Pin 18
GPIO.setup(24,GPIO.OUT) #Setting LED as Output device

def readLDR():
count=0
GPIO.setup(23, GPIO.OUT) #Setting LDR as Output device
GPIO.output(23, GPIO.LOW) #Initializing to LOW/0/False
time.sleep(0.1)

GPIO.setup(23, GPIO.IN)  #Change the pin back to input
while (GPIO.input(23) == GPIO.LOW):
                          #Count until the pin goes high
    count += 1
return count

while True:               # The main loop
    current_lvl = readLDR()

    if current_lvl < ldr_thr:
        GPIO.output(24, GPIO.HIGH)
    else:
        GPIO.output(24, GPIO.LOW)
time.sleep(1)
```

Syllabus Topic : Beagle Board and Other IoT Devices

3.10.5 Other IoT Devices

Similar to Raspberry Pi there are other small board computer available in market which can also be deployed for implementation of IoT based systems. Let's take a look at some of them.

pcDuino

Q. Explain pcDuino IoT device in brief.

- pcDuino is a Arduino based small board computer system which comes with ARM Cortex A8 with 1GHz processing speed. pcDuino has a capacity to run Linux based OS like Ubuntu and Android (ICS).
- It has HDMI support which can be used for audio/video interfacing. It supports various programming languages such as C, C++, Java, Python etc.

Beagle Bone Black (BBB)

Q. Explain Beagle Bone Black IoT device in brief.

- Beagle Bone Black is similar to Raspberry Pi but is more powerful in many senses. It has more number of onboard GPIO pins and also provides on board storage capacity. Similar to Raspberry Pi, BBB supports HDMI, USB, SD card availability, Ethernet connectivity etc.
- BBB runs on 1 GHz with ARM Cortex A8 processor. It supports various programming languages such as C, C++, Java, Python etc. and can run Linux based operating systems and Android fluently.

Cubieboard

Q. Explain CubieBoard IoT device in brief.

- Cubieboard is another small board computer which runs on dual core ARM Cortex A7. It has HDMI port, Ethernet Connectivity, USB port, SD card slot and 96 extended pin support.
- Cubieboard supports SATA connectivity. It supports various programming languages such as C, C++, Java, Python etc. and can run Linux based operating systems and Android fluently.

Comparison between Raspberry Pi, pcDuino, BBB and Cubieborad

Q. Compare Raspberry Pi with other IoT device.

Table 3.10.1 : Compares above IoT devices various parameters

Feature	Raspberry Pi	pcDuino	BBB (Beagle Bone Black)	Cubieboard
CPU	1.2 GHz 64bit quad core ARM Cortex A53	1GHz ARM Cortex A8	1GHz ARM Cortex A8	1GHz Dual Core ARM Cortex A7
GPU	Video Core IV	Mali 400	PowerVR SGX530	Dual Core ARM Mali 400
Storage	NA	2GB Flash	4GB Flash	4GB NAND Flash
Memory	1GB	1GB	1GB	1GB
OS	Raspbian, pidora, Arch Linux, RISC OS	Ubuntu, Android ICS	Angstrom Linux, Debian, Android and other Linux based OS	Android and other Linux based OS
Input/Output	4 USB, SD Card slot, MMC	NA	4+1 USB, MicroSD card slot	2 USB, SD card, SATA
Networking	10/100M Ethernet	10/100M Ethernet	10/100M Ethernet	10/100M Ethernet
Audio	3.5mm jack, HDMI	HDMI	HDMI	HDMI
Video	Composite RCA, HDMI	HDMI	HDMI	HDMI

Feature	Raspberry Pi	pcDuino	BBB (Beagle Bone Black)	Cubieboard
Interfaces	GPIO, Serial, I2C, SPI	GPIO, Serial, I2C, SPI, PWM, ADC	69 pin GPIO, 4 Serial, I2C, SPI, CAN, MMC, JTAG	96 extended pin interface including Serial, I2C, SPI, ADC, RGB/LVDS, VGA,R-TP etc.
Power	5VDC	5V/2A	5VDC	5VDC

3.11 Exam Pack (Review Questions)

☞ **Syllabus Topic : Horizontal, Verticals and Four Pillars of IoT**

Q. What are horizontal and verticals of IoT applications ? *(Refer section 3.1.1)*

Q. Explain the four pillars of IoT in brief. *(Refer section 3.1.2)*

Q. Compare four pillars of IoT on the basis of communication Network. *(Refer section 3.1.2)*

☞ **Syllabus Topic : M2M - The Internet of Devices**

Q. Explain M2M communication in details. *(Refer section 3.2)*

☞ **Syllabus Topic : RFID - The Internet of Objects**

Q. Explain RFID communication in details. *(Refer section 3.3)*

☞ **Syllabus Topic : WSN - The Internet of Transducer**

Q. Explain WSN communication in details. *(Refer section 3.4)*

☞ **Syllabus Topic : SCADA - The internet of Controllers**

Q. Explain SCADA communication in details. *(Refer section 3.5)*

Q. Explain various components in SCADA communication architecture. *(Refer section 3.5)*

☞ **Syllabus Topic : DCM - Device, Connect and Manage**

Q. What is DCM with reference to IoT? *(Refer section 3.6)*

Q. What is DCM and DNA with reference to IoT? *(Refer section 3.6)*

☞ **Syllabus Topic : Device - Things that Talk**

Q. What do you mean by "Things that talk" in IoT? *(Refer section 3.7)*

☞ **Syllabus Topic : Connect - Pervasive Network**

Q. What type of connections are available in IoT based systems. *(Refer section 3.8)*

Q. Explain the term Connect in DCM in details. *(Refer section 3.8)*

Q. Explain Wired Communication term in detail (FieldBus). *(Refer section 3.8.1)*

Q. Explain Wireless Communication term in detail. *(Refer section 3.8.2)*

Q. Explain satellite term in detail. *(Refer section 3.8.3)*

☞ **Syllabus Topic : Manage - To Create Business Values**

Q. Explain how business value can be created in Industry using in IoT based systems? *(Refer section 3.9)*

☞ **Syllabus Topic : IoT Physical Devices and Endpoints - Basic Building Blocks of an IoT Device**

Q. What is an IoT device ? *(Refer section 3.10)*

Q. What are the basic building blocks of an IoT device ? *(Refer section 3.10.1)*

☞ **Syllabus Topic : Exemplary Device - Raspberry Pi**

Q. Explain Raspberry Pi in brief/ in details. *(Refer section 3.10.2)*

Q. Explain Pin Configuration of Raspberry Pi in brief. *(Refer section 3.10.2)*

☞ **Syllabus Topic : Raspberry Pi Interfaces**

Q. What kind of interfaces are available with Raspberry Pi ? *(Refer section 3.10.3)*

☞ **Syllabus Topic : Programming Raspberry Pi with Python**

Q. Explain in brief about Python Coding for Raspberry Pi. *(Refer section 3.10.4)*

Q. Explain LED interfacing with Raspberry Pi. Handle the components using commands only.
(Refer section 3.10.4)

Q. Explain LED interfacing with Raspberry Pi with the help of Python Code.
(Refer section 3.10.4)

Q. Explain LED and switch interfacing with Raspberry Pi with the help of Python Code.
(Refer section 3.10.4)

Q. Explain LED and LDR sensor interfacing with Raspberry Pi with the help of Python Code.
(Refer section 3.10.4)

Syllabus Topic : Beagle Board and Other Devices

Q. Explain pcDuino IoT device in brief.
(Refer section 3.10.5)

Q. Explain Beagle Bone Black IoT device in brief.
(Refer section 3.10.5)

Q. Explain CubieBoard IoT device in brief.
(Refer section 3.10.5)

Q. Compare Raspberry Pi with other IoT device.
(Refer section 3.10.5)

UNIT IV

CHAPTER 4

IoT Protocols and Security

Syllabus Topics

Protocol Standardization for IoT, Efforts, M2M and WSN Protocols, SCADA and RFID Protocols, Issues with IoT Standardization, Unified Data Standards, Protocols - IEEE 802.15.4, BACNet Protocol, Modbus, KNX, Zigbee Architecture, Network layer, APS layer.

IoT Security : Vulnerabilities of IoT, Security Requirements, Challenges for Secure IoT, Threat Modeling, Key elements of IoT Security : Identity establishment, Access control, Data and message security, Non-repudiation and availability, Security model for IoT.

Syllabus Topic : Protocol Standardization for IoT, Efforts

4.1 Protocol Standardization for IoT - Efforts

4.1.1 Protocol Standardization for IoT - Efforts

Q. What are some protocol standardization efforts taken for IoT?

Based on the four pillars of IoT studied in Chapter 3 we can conclude that IoT standardization is needed for data representation and protocols. In this section a slight focus is cast upon current efforts and work done in achieving this IoT standardization. Following are the major areas which needs to be concentrated

1. To create an architectural foundation for IoT, that will be interoperable with future internet.
2. Using existing technologies instead of creating new ones.
3. Proving usability of applications with the help of Use Cases.
4. Creating interdisciplinary solution which targets more deployment aspects that will generate strong stakeholders.
5. Uniting heterogeneous IoT technologies into a single IoT entity.

Keeping above points in mind two such noteworthy groups are making an effort towards standardization of IOT: Work Package (WP) Framework and Internet Protocol for Smart Objects (IPSO).

☞ Work Package (WP) Framework

- Work Package Framework defines the overall gross view of an IoT systems with respect to all stakeholders.
- Service Providers, Developers, Investors and Users can gain insightful knowledge from Work Package Framework.
- Work Package Framework can be described as shown in Fig. 4.1.1.
- As shown in Fig. 4.1.1, WP framework gives implementation standards for developers using Architectural Reference Model.
- Which Smart objects will deployed in the field, How they will be communicating i.e. protocols used and service identification and How thi· service will be integrating with future developments of IoT. All these things are described in Architectural Reference Model.
- Architectural Reference Model is generated by gathering requirements from stakeholders and are validated by the stakeholders time to time.
- WP also defines:
- Use Cases of the application which gives clear idea of available functionalities.
- Circulation of these services worldwide and its impact over the business.

- Maintenance strategy and coordination of applications after deployment.

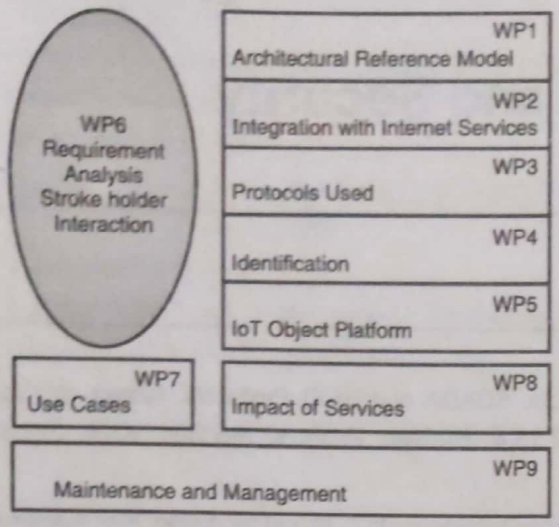

Fig. 4.1.1 : General Work Package Framework

Internet Protocol for Smart Objects (IPSO)

- IPSO defines an All IP approach for IoT standardization.
- IP defines a wide range of protocols that can be used to create highly scalable communication technology.
- IP provides an open, lightweight, stable, pervasive and manageable way to communicate between applications and devices.
- IPSO defines a smart object as:
  - An RFID Tag (Intelligent Device)
  - A Sensor: A device that measures physical change in analog or digital format.
  - An Actuator: A device that is able to alter or control physical devices in the field.
  - An Embedded Device: Small electric devices build to serve specific functionality.
  - Any complex combination of above devices.

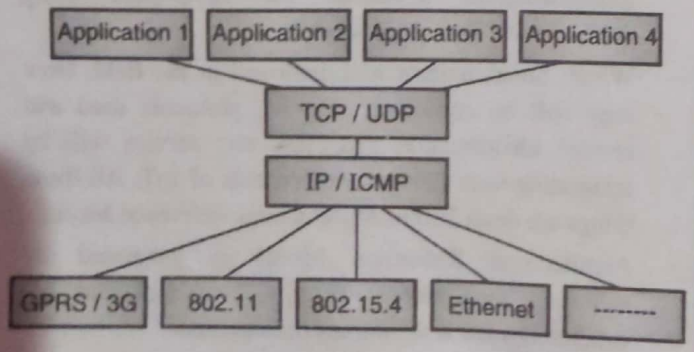

Fig. 4.1.2 : All IP Network Protocols

- As shown in Fig. 4.1.2 IPSO uses IP based network protocols to communicate between lower level devices and top layer applications.
- Mobile IP is one more approach by Internet Engineering Task Force (IETF) which manages the movement of the mobile devices across IPv4 type and IPv6 type networks.

Syllabus Topic : M2M and WSN Protocols

4.2 M2M and WSN Protocols

- As M2M applications and devices are getting popularity, more and more organization are shifting their focus towards M2M solution, there is greater need of standardization in the field. Most of the M2M solutions today are too specific and hence hard to fit under a common standardization and framework. This raises issues related interoperability and integration with existing and new solutions.
- M2M protocol and WSN protocol standardization is therefore too important for the growth of M2M and IoT industry. It will ensure the truly connected internet of things.
- With the combined effort of International Telecommunication Unit (ITU), ESTI's M2M technical committee, Global standards collaboration are working towards standardization of M2M and establish M2M task force. They have come up with a common conceptual framework for M2M application.

4.2.1 M2M Protocol Standardization Activities

Q. Explain M2M protocol standardization.

The M2M standard activities focus on the following points :

1. **Data transport protocol standards :** Focusing on the standardization of M2MXML, JSON (Java script object notation), WMMP, MDMP etc.
2. **M2M device protocol standards :** Extension of OMA DM such that it can be collaborated and supported M2M protocol management objects.
3. Standardization of the device management for M2M.
4. Standardization of the device gateways systems and configuration.
5. Standardizing the network APIs for to enhance M2M service capabilities.
6. Resolving IP addressing related issued for devices supporting IPv6.
7. Standardization related to remote provisioning and discovery.

4.2.2 WSN Protocol Standardization Activities

Q Explain WSN protocol standardization.

There are many standard bodies in area of wireless sensors network. The concentration of IEEE is on physical layer and MAC layer. Standardization of Layer 3 and above is handled by IETF.

IEEE 1452 is a standard for set of smart transducer interfaces which allows to develop network independent, common and open interface for connecting sensors and actuator. It can also be used for connecting microprocessors, instrumentation systems and field networks. The primary objective of this standard is to provide a standard and common interface for connecting different transducers (sensors and actuator).

Activities

It covers the following activities.

1. **1451.0-2007 :** Standardization of interface functions, communication protocols, and TEDS Formats
2. **1451.1-1999 :** Network Capable Application Processor Information Model
3. **1451.2-1997 :** Transducer to Microprocessor Communication Protocols & TEDS Formats
4. **1451.3-2003 :** Digital Communication & TEDS Formats for Distributed Multi-drop Systems
5. **1451.4-2004 :** Mixed-mode Communication Protocols & TEDS Formats
6. **1451.5-2007 :** Wireless Communication Protocols & TEDS Formats.
7. **1451.7-2010 :** Transducers to Radio Frequency Identification (RFID) Systems Communication Protocols and TEDS Formats.

Syllabus Topic : SCADA and RFID Protocols

4.3 SCADA and RFID Protocols

SCADA and RFID are the enabling technology (more specifically are the two pillars) for IoT. They serve as one of the pillars of IoT system. Let us discuss both the standards.

4.3.1 SCADA Standardization activities

Q. Explain SCADA protocol standardization.

- With more and more advanced commercial cloud application, SCADA is increasingly popular in IoT application.
- Std C37.1 is the IEEE standard for automation system and SCADA (Supervisory control and data acquisition).
- IEEE Std. C37.1 SCADA architecture is well suited and applicable to recent automation based industrial upgradations.
- For a power system, it can be observed that the processing is now distributed. Function and operations which had to be performed by control centred is now performed by systems called IEDs (Intelligent electronic devices). They can be IoT devices too.
- Many functions are now performed by IEDs, there are still utilities that need master station for the operation of power system.
- SCADA is also standardized in the sense that how different vertical standards (specific industry technology standards) interact with each other.
- The specification addresses all the aspect of SCADA system, from device specification to how different technology interacts with each other.

4.3.2 RFID Standardization Activities

Q. Explain RFID protocol standardization.

- Compared to other pillars of IoT, RFID data formats and protocols are well defined by EPCGlobal.
- As seen in Chapter 1, Electronic Product Code Information Service (EPCIC), Object Name Service (ONS), are some of the standardizations in RFID.
- Physical Markup Language (PML) is XML like language which defines its own tags to represent data related to individual Electronic Product Codes.
- The Application Level Events (ALE) standard, created by EPCGlobal, specifies an interface to write a business logic to specify behavior and functionalities of the software.
- One of such standardizations is being used in contactless payment system which uses RFID in its architecture.
- Contactless smart card payment system uses ISO/ISE 14443 standardizations which allow communication up to 10cm distance.
- Another such standardizations for contactless smart card payment system is ISO/ISE 15693 which allow communication up to 50cm distance.
- ISO 24769 and ISO 24770 provide Real Time Locating System management.

Syllabus Topic : Issues with IoT Standardization

4.4 Issues with IoT Standardization

Q. Explain the issues with IoT standardization.

IoT standardization provides a solid way to develop IoT based systems but it also limits the innovation and productivity in some cases where out of standard practice is utilized in market. Some of the challenges and issues face by standardization are listed below:

- Many standards for the internet are not adequate for supporting IoT efficiently, since IoT didn't exist when protocols were designed and implemented.
- There are no global validated standardization frameworks at the IoT stack level.
- Different organizations, forums, alliances and groups are working on their own with limited scope and focusing areas only they are comfortable with.
- For example EPCGlobal works only for RFID while 3GPP for cellular networks only.
- It is also found that two or more associations, groups working on the standardization of same concept or technical area are competing with each other rather than sharing information and findings.
- ICT standardization is highly decentralized activity. Now challenge is how these activities of heterogeneous standards can be coordinated.
- It is a challenge to develop a platform where various forums can collaborate to develop the standardization for IoT.
- It is essential to allow all interested stake holders to participate in the IoT standardization process towards and understand their requirements. Now issue is how this can be achieved.
- One of the possible solutions to this is to try and standardize the ubiquitous middleware and XML based representation for data exchange.
- This approach is being used by organizations like World Wide Web Consortium (W3C), Organization for the Advancement of Structured Information Standards (OASIS) etc.

Syllabus Topic : Unified Data Standards

4.5 Unified Data Standards

Q. Explain various data standards used in IoT data exchange.

Many IoT protocols for communications and data exchange are in use, many standardization approaches are being worked upon. Similarly the data and information roaming in IoT systems and WoT based applications is also key part as it must be understood by all the interested parties. Some of the unified data standards and issues are listed below.

- The Internet was originally Internet of multimedia and documents. So the WWW and Internet uses HTML and HTTP at its core to represent the data over the Internet.
- Most protocols uses XML based file representation which makes data exchange easier for heterogeneous techniques. So it will be helpful if XML based representation is integrated in OS itself.
- Resource Description Framework (RDF) is a work of W3C and was originally developed as a metadata model.
- RDF can be used for modeling of information that is deployed as a web resource, using various syntax formats. This model is useful for WoT applications.
- As seen in Chapter 1, SOAP and REST frameworks can be used to provide data exchange protocols for IoT applications.
- Electronic Data Interchange (EDI) is a technology that describes the format of electronic documents that can be shared over the Internet.
- EDI defines techniques for FTP, email, HTTP, SMTP etc.
- A combination of two famous techniques, EDI and XML gives a rise to a new technique called ebXML.
- ebXML is mostly used for e-commerce solutions. E-commerce software communication uses ebXML standards which specify rules of interaction between two parties.
- Some additional formats accepted by WoT are listed below :
  - CBRN, format for Chemical, Biological, Radiological and Nuclear data
  - EXDL, Emergency Data Exchange Language of OASIS
  - NGTP, Next-Generation Telematics Protocol
  - M2MXML, Machine-to-Machine XML
  - oBIX, open Building Information eXchange
  - WMMP, Wireless Machine Management Protocol of China Mobile and so on.

Syllabus Topic : Protocols - IEEE 802.15.4

4.6 Protocols - IEEE 802.15.4

4.6.1 IEEE 802.15.4

Q. Explain IEEE 802.15.4 protocols in details.

Institute of Electrical and Electronics Engineers (IEEE) group defined the 802 standard protocol family for physical and data link layer technologies. It explores Physical layer protocols and Media Access Control (MAC) protocols are specified in the IEEE 802.15.4 standards while Logical Link Control (LLC) layer is defined in IEEE 802.2 standards.

- IEEE 802.15.4 standard was first published in 2003 and later was modified in year 2006.
- IEEE 802.15.4 standard defines the Physical layer and MAC layer communication methods used to access the wired (Ethernet) and wireless network.
- As shown in Fig. 4.6.1, data frames are sent via Physical layer and MAC layer to LLC for formatting.
- The LLC IEEE 802.2 standard converts this data frames into defined frame formats which are understandable to the upper layers.

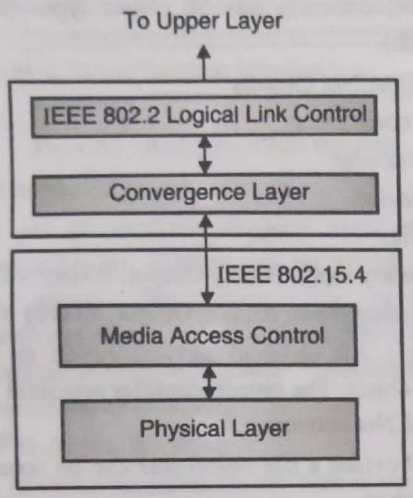

Fig. 4.6.1 : IEEE 802.15.4 Stack

IEEE 802.15.4 : Physical Layer

- Federal Communication Commission (FCC) manages the distribution of frequency band in USA.
- FCC has allocated frequencies for Industrial, Scientific and Medical Applications (ISM) and for transmission power below 1 Watt doesn't require any licensing.

FCC ISM Band Frequency Distribution

FCC Band	Frequency	Max Transmission Power
Industrial Band	902 MHz - 928 MHz	Below 1 W
Scientific Band	2.4 GHz - 2.48 GHz	Below 1 W
Medical Band	5.725 GHz - 5.85 GHz	Below 1 W

- 868 MHz - 868.6 MHz band is used in Europe which provides 20 kbps data rate.
- As seen in above table Industrial band uses 902 MHz – 928 MHz frequency range to provide data rate of 40 kbps using Direct Sequence Spread Spectrum Technique (DSSS).
- The Scientific Band uses 2.4 GHz – 2.48 GHz frequency band to provide data transfer rate of 250 kbps using DSSS.
- The frequency band usage is not universal. 868 MHz band is limited to Europe, 915 MHz band is limited to America while 2.4 GHz and 5.725 GHz band is used worldwide.
- Most applications uses 2.4 GHz frequency band for implementations.
- 802.15.4 describes the frequency band, modulation type (BPSK,QPSK etc.) and channel number in 32 bit format.
- 5 bits are used to define the page number and 27 bits are used to describe the channel specifications.
- Modulation types are used for encoding data in the phase of signal.
- Frequency band is divided into channel numbers each referring to specific part of the frequency band.

Example

- One example is shown in Table 4.6.1.

Table 4.6.1

Frequency Band	Modulation	Page Number	Channel Number and center frequency
2.4 GHz	0-QPSK	0	11:2405 MHz 12:2410 MHz . . . 26:2480 MHz
915 MHz	BPSK 0-QPSK	0 2	1: 906 MHz 2: 908 MHz . . . 10: 924 MHz

- Interested readers can read about modulation types and channel in details.

IEEE 802.15.4 : Media Access Control Layer

- 802.15.4 describes MAC in two parts one MAC layer responsible for data transfer also known as MAC Common Part Sub layer (MCPS) and second part responsible for MAC layer management also known as MAC Layer Management Entity (MLME).
- MLME configuration and state parameters for MAC layer.
- It contains following details :
  - Which addressing to use 64 bit IEEE address called as Extended Unique Identifier (EUI-64) or 16 bit short address for node having less computational capability.

- How many times to retry accessing the network if collision occurs. (Max 5 times)
- How much to wait for acknowledgement. (Max 120 units)
- How many times to resend the packet in case of acknowledgement not received? (0 to 7)

– In this way MAC layer handles frame validation, time slots for packets and manages delivery of packets.

– Following Fig. 4.6.2 shows 802.15.4 frame format.

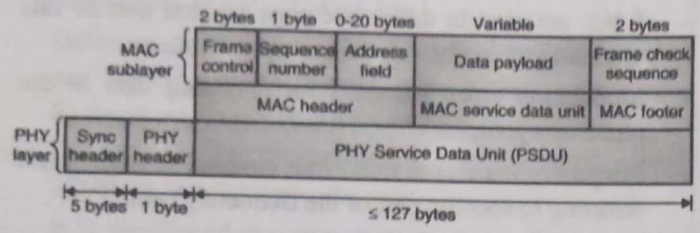

Fig. 4.6.2 : IEEE 802.15.4 Frame Format

– MAC layer frame resides just above PHY layer frame.

– Data Payload part contains the data to be transferred in the network layer.

– Sequence number and frame check sequence (FSC) manages the acknowledgement.

– PSDU can be in between 0 to 127 bytes in size with 5 bytes of synchronous header and 1 byte of physical Header.

Limitations of IEEE 802.15.4

– 802.15.4 has maximum packet size of 127 bytes. So application needs to take care of the packets that might exceed the limit.

– Limited bandwidth is also an issue as PHY layer provides 250kbps as seen previously.

– PHY layer packets waits for acknowledgments so packets cannot be sent continuously.

Uses of IEEE 802.15.4

– With limited operating power, IEEE 802.15.4 is suitable for IoT based application working with multiple sensor nodes.

– Low power requirements make it highly scalable where large number of nodes can be deployed together.

– Network maintenance is low cost and reliable.

Syllabus Topic : BACNet Protocol

4.6.2 BACNet Protocol

Q. Explain BACNet protocols in details.

BACNet stands for Building Automation and Control Networks. BACNet targets industrial automation and control mechanisms. Let's study BACNet in detail.

– BACNet is an Object Oriented protocol, which views its implementation in the form of objects.

– BACNet protocol is defined by three basic characteristics: Objects, Services and Properties.

BACNet Objects

– BACNet object is a logical representation of the usable entity which can be a physical device, analog input (temperature input), binary output (relay control) etc.

– BACNet currently has 30 object types (more under research).

– Some BACNet Objects
  - Access Door
  - Calendar
  - Device
  - File
  - Binary input, Binary Output, Binary Value
  - Analog input, Analog Output, Analog Value, etc.

– Device : All physical devices needs to implement device object. The object identifier separates devices on the BACNet network.

– File : Provides a file object that can be accessed using file services.

– Calendar object provides list of dates and access doors gives object emulating an actual physical door (open/close).

– Such objects are available in BACNet which can be effectively used for IoT based applications.

BACNet Properties

– The information about BACNet objects are defined by its properties.

– Each of the BACNet object must have following three properties :
  - object_identifier
  - object_name
  - object_type

– Objects may have properties as per its specifications and might not be found common in other objects.

BACNet Services

– Services are nothing but actions that can be performed with the help of BACNet object.

– Services such as read, write, input/output are available.

– Object that provides services is a server and the one that accesses the services is a client.

– An object can be a server, client or both based on the applications requirement.

☞ **Five classes of services provided by BACNet**

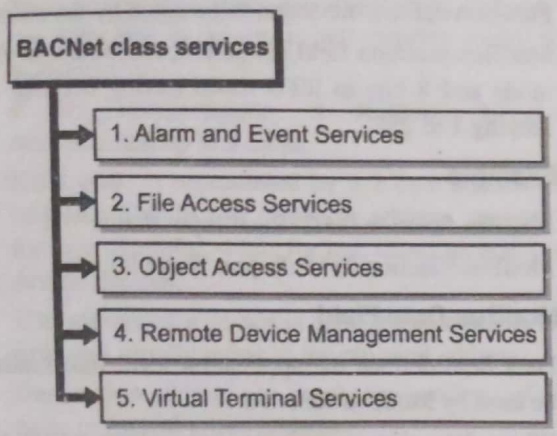

Fig. C4.1 : BACNety Class services

➜ **1. Alarm and Event Services**

These set of services provides event reporting to various activities. Alarms can be set for Change Of Value event. This class has services like: ***AcknowledgeAlarm, ConfirmedCOVNotification, UnconfirmedCOVNotification, GetAlarmSummary, GetEventInformation, etc***

➜ **2. File Access Services**

Read or Write operation on the files is performed with this service. Only one operation is done at a time.

➜ **3. Object Access Services**

Set of services providing access to the object with services like : **ReadProperty, WriteProperty, CreateObject, DeleteObject, ReadPropertyConditional, WritePropertyMultiple, etc.**

➜ **4. Remote Device Management Services**

Who-is/ I-am services are used to identify object over BACNet network and Remote Device Management Services can be used to **remotely** start/stop messages, device/object initialization, vendor specific commands.

➜ **5. Virtual Terminal Services**

Allows bidirectional exchange of character oriented data streams.

– **Other Examples of services**

1. ConfirmedTextMessage,
2. UnconfirmedTextMessage,
3. AtomicReadFile, AtomicWriteFile,
4. ReinitializeDevice, etc.

– **BACNet Object/Properties Example**

Name	Value
Object_Name	Space Temperature
Object_Type	Analog Input
Present_Value	75
High Limit	80
Low Limit	60

– Services can be used on condition if temperature is crossing the high or low value. Different services can be used base on application need.

Syllabus Topic : ModBus

4.6.3 ModBus

Q. Explain ModBus protocols in details.

– ModBus is a serial communication protocol developed by Modicon Inc. (now Schneider Electric) in 1979.

– ModBus is an application layer protocol which provides client server communication to the devices connected via buses and network.

– ModBus works on top of serial communication standards like RS 232, RS 442, RS 485.

– ModBus works with master slave model. One master device initiates the transaction by generating query and sent to an individual slave address or broadcasted in the network.

– The slave devices work according to the query or return data generated as a result of query back to the master device.

– ModBus has two transmission modes that defines framing and bit encoding for messages to be transmitted on the network.
  - o American Standard Code for Information Interchange (ASCII) Transmission Mode
  - o Remote Terminal Unit (RTU) Transmission Mode

– All nodes in the ModBus Network have to use the same transmission mode and serial parameters to function properly.

☞ **ASCII Transmission Mode**

– Coding System:
  - o One Hexadecimal character is contained in each ASCII character of message.
  - o ASCII character 0 – 9, A – F.

– Each ASCII character is sent separately as
  - o 1 start bit
  - o 7 data bits (LSB sent first)
  - o 1 bit for even/odd parity; no bit for no parity
  - o 1 stop bit if parity is used, 2 bits if no parity

– Longitude Redundancy Check (LRC) is used for error checking.

- One advantage of ASCII mode is that it allows time interval of one second between two characters without causing error.
- Greater time interval suggests that error might have occurred in transmission.
- Message starts with a ':' represented by ASCII 3A hex and CRLF defines carriage return line feed ASCII 0A and 0D to specify end of frame.

START	ADDRESS	FUNCTION	DATA	LRC CHECK	END
1 CHAR	2 CHARS	2 CHARS	n CHARS	2 CHARS	2 CHARS CRLF

Fig. 4.6.3 : ModBus ASCII Message Frame

RTU Transmission Mode

- Coding System
  - Two Hexadecimal characters are contained in each 8 bit field of message.
  - 8 bit binary with two Hexadecimal Character 0 – 9, A – F,
- Each ASCII character is sent separately as
  - 1 start bit
  - 8 data bits (LSB sent first)
  - 1 bit for even/odd parity; no bit for no parity
  - 1 stop bit if parity is used, 2 bits if no parity
- Cyclic Redundancy Check (CRC) is used for error checking.
- Each message is transmitted in continuous stream, time delay causes error.
- Greater character density of RTU mode allows better data throughput compared to ASCII mode.
- RTU doesn't have any specific characters to represent start and end of the frame as in ASCII.
- RTU uses T1-T2-T3-T4 to represents the baud rate being used in the network at start and end of the frame.

START	ADDRESS	FUNCTION	DATA	CRC CHECK	END
T1-T2-T3-T4	8 BITS	8 BITS	n x 8 BITS	16 BITS	T1-T2-T3-T4

Fig. 4.6.4 : ModBus RTU Message Frame

ModBus Address

- ModBus address begins with 8 bit address of target that ranges from decimal value 1 to 247.
- 0 is used for specifying broadcast address.
- Each query has address of specific slave defined in frame format for both modes.
- Address field contains 2 characters in ASCII mode and 8 bits in RTU mode.

ModBus Function

- Function defines the action to be taken by the message.
- ModBus function field contains 2 characters in ASCII mode and 8 bits in RTU mode having decimal value ranging 1 to 255.

Example

1. ModBus function 0x02 :Read Input Status
2. ModBus function 0x11 :Report Slave ID, etc.

ModBus Data Field

- Data field defines the application level information to be used by ModBus function.
- When data size is variable, byte count needs to be specified at the start of the data field.

Syllabus Topic : KNX

4.6.4 KNX

Q. Explain KNX protocols in details.

- KNX standardization (The Konnex Association) was formed by the merger of three European companies working for Smart Automation of homes and buildings.
- Batibus Club International using Batibus for building automation (France), The European Installation Bus Association using EIB system and European Home Automation System (Holland) using EHS system came together in 1999 to form KNX standardization which is compatible with Batibus, EIB and EHS systems.
- The main advantage of KNX is its backward compatibility. KNX is compatible with all of its previous versions and current version is 2.0 from August 2009.
- KNX defines communication specifications with various physical medium like:
  - Twisted Pair Wiring
  - Radio
  - Infrared
  - Ethernet
  - Power line Networking
- For routing of messages, KNX uses telegrams throughout the network of KNX nodes. The format is shown in Fig. 4.6.5

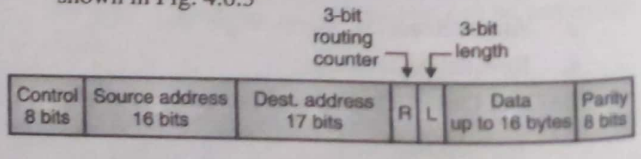

Fig. 4.6.5 : KNX telegram Format

- Telegrams are transmitted on the physical layer octet by octet.

- Each telegram is acknowledged by the receiver when the telegram is received and sender waits for the acknowledged.
- If no acknowledged is received telegram is retransmitted up to 3 times.
- KNX node is represented by a 2 byte address (source address). The 16 bits of source address contains 4 bits for area identifier, 4 bits for line identifier and 4 bits for device number.
- The address of a device is configured at the installation time and usually refers to the physical address.
- Destination address can be physical address of a single node or logical group address.
- Single KNX device is addressed by physical address and a set of nodes can be represented by a logical address of 16 bits.
- 17^{th} bit is used to represent whether the destination address is physical address of a single node (17^{th} bit = 0) or logical group address (17^{th} bit = 1).
- Control bits are used to represent priority level (system priority, alarm priority, high priority, low priority), transmission type (normal or repeat) and the same 8 bits can also be used for representing acknowledgement type ACK,NAK or BUSY.
- 3 bit routing counter define hop count and is usually limited to 6 hops.
- 8 bit parity is used to secure KNX telegram by parity check of previous telegram.

Functional Block

- KNX devices have a functional block specifying the functionalities of that device. Functional Blocks are logical grouping of input, output and parameters needed for performing a task.
- **Example: Functional Block: Sun Blind Actuator Basic**
- List of Inputs: Move Up/Down, Step Up/Down, Wind Alarm, Set Absolute Position etc.
- List of Outputs: Info Move Up/Down, Current Absolute Position, etc.
- List of Parameters: Pause Time, Move Up/Down Time, etc.

Device Configuration

- KNX devices can be working in following modes:
- S-Mode is the system mode. The configuration and management of KNX device is done in this mode from a PC.
- E-Mode is the easy mode. This mode is used to configure the network without the use of PC. An embedded master controller can be attached to the KNX node that detects and connects to other devices one by one.
- There is one more mode in which the devices are preconfigured with logical tags. Logical tags in this mode define an additional frame format along with the standard frame format. Tags can also be designed with geographic specifications.

Syllabus Topic : Zigbee Architecture, Network Layer, APS Layer

4.6.5 Zigbee

Q. Explain ZigBee protocols in details.

Zigbee Architecture

- Zigbee resides on top of the PHY and MAC layer defined in 802.15.4. PHY and MAC layer provides the functionality of OSI physical and link layers.
- Fig. 4.6.6 shows Zigbee Architecture:

Application Support (APS) Sublayer

- APS handles 64 bit IEEE to 16 bit Zigbee address mapping.
- APS routes the network layer message to suitable application object. Each application object is identified by endpoint ID.
- APS maintains a local binding table that holds record of remote nodes and endpoints registered to receive message from local endpoint.
- APS also handles acknowledgements, retry sending messages if not received and data duplication.

Zigbee Device Object (ZDO)

- ZDO is a special application running on Endpoint 0 and manages the state of Zigbee Node.
- ZDO initializes the APS, NWK and Security Service Provider.
- ZDO provides interfacing between application objects, Zigbee Device Profile and APS.
- ZDO manages security policies and Security configuration of a device provided by Security Service Provider.
- ZDO gathers configuration information from endpoint application and provides device and service discovery.

Zigbee Cluster Library

- ZCL was added late in Zigbee architecture and defines a library of interface specifications like commands and attributes used for application profiles.
- ZCL also provides functionalities for network interface, group formation and management.

- ZCL acts as a common mechanism for application development even with the ongoing development of Zigbee.

Zigbee Application Framework

- This provides API for Zigbee application development and each Zigbee application is appointed an endpoint staring from 1.

Types of Zigbee Node

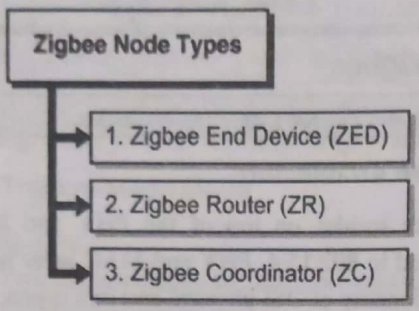

Fig. C4.2 : Zigbee Node Types

1. Zigbee End Device (ZED)

- ZED works with 802.15.4 providing reduced functioning device.
- These nodes are battery operated and are not always listening or sensing.
- ZED has to connect to a network via Zigbee Router which acts as its parent.

2. Zigbee Router (ZR)

- ZR works with 802.15.4 and provides fully functioning device (FFD).
- ZR are permanently listening and are able to route packets once it joins the existing Zigbee network.

3. Zigbee Coordinator (ZC)

- ZC works with 802.15.4 and provides fully functioning device (FFD).
- ZC can create a network and can also act as a 802.15.4 based Personal Area Network (PAN) coordinator.
- ZC can form a network and can also connect to the existing network and are able to routes packets, in such cases ZC become ZR.
- Zigbee nodes are not mutually exclusive which means a ZED can also act as ZR and even as ZC.

Zigbee Network Layer (NWK)

- NWK provides multihop routing of data packets in mesh network which is not available in 802.15.4.
- Zigbee uses short address allocation to nodes ranging from 0x0000 to 0xFFF7.
- Zigbee coordinator node uses short address 0x0000.
- Zigbee supports two address allocation modes:
- Stack profile 0x01 in which address is allocated based on the position of the node in the mesh/tree topology. Each potential parent node is associated with sub block of network addresses (among which address is allocated to the requesting node).
- Stack profile 0x02 uses random address allocation mechanism and any address conflicts are detected and resolves by Zigbee (which requires additional computation).

Zigbee Network Layer Frame Format

Field	Field Size in Octets(8 bits)
Frame Control	2
Destination Address	2 or 8
Source Address	2 or 8
Radius	1
Sequence Number	1
Payload	Variable

Octets : 2	2	2	1	1	Variable
Frame control	Destination Address	Source Address	Radius[a]	Sequence Number[b]	Frame Payload
	Routing Fields				
NWK Header					NWK Payload

Fig. 4.6.6 : Zigbee NWK Frame Format

- Frame Control bits are used to define various parameters like:
  - Communication type – unicast / multicast / broadcast
  - Security enabled / disabled
  - Route discovery enabled / disabled
  - Source IEEE address is specified or not
  - Destination IEEE address is specified or not.
- Address field can be 2 or 8 octets to handle 16 bit Zigbee address or 64 bit IEEE address.
- Radius defines maximum number of hops allowed for the packet.
- Sequence Number is the packet counter.
- Payload carries APS and NWK layer commands if any.

Zigbee APS Layer

- APS manages and provides support for local applications and also provides mechanisms for developing and interconnecting Zigbee applications.
- A Zigbee Device can implement multiple applications on the same 802.15.4 address.
- So to differentiate between applications on the same address Zigbee uses endpoints to define application. It is same as port number in TCP/IP network.
- An endpoint is described by simple descriptor. A simple descriptor has endpoint number, Application profile ID, Application device ID, Application Version ID, List of input clusters and List of output clusters.
- The simple descriptor of an endpoint can be retrieved by any node in the network using simple descriptor request command (***Simple_Desc_Req***).

Zigbee APS Layer Frame Format

Field	Field Size in Octets(8 bits)
Frame Control	1
Destination Endpoint	1
Cluster Identifier	1
Application Profile Identifier	2
Source Endpoint	1
Counter	1
Payload	Variable up to 80 bytes

- Frame Control bits are used to define various parameters like:
  - Frame Type - Data / Command / ACK
  - Delivery Mode - unicast / multicast / broadcast
  - Security enabled / disabled

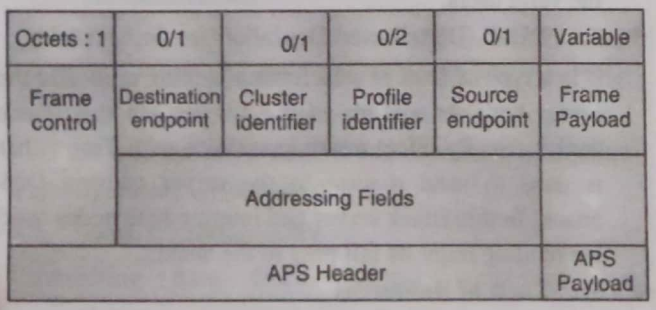

Fig. 4.6.7 : APS Frame Format

- Source endpoint is 8 bit long and specifies the originator of the frame.
- Destination Endpoint is 8 bit long and specifies the recipient of the frame.
- Cluster Identifier is 8 bit in length and is used to identify the cluster that is used in the binding operation of Zigbee coordinator (bind request function is used to form a cluster using Zigbee coordinator ZDO). This field is present for data frames but not for command frames.
- Application Profile Identifier occupies 2 octets and specifies the Zigbee profile for which frame is intended. This filed is used in Data or ACK frames only.
- Payload contains information about individual frames.

Syllabus Topic : IoT Security - Vulnerabilities of IoT

4.7 IoT Security

IoT connects all the devices in the surroundings and provide a unique solution with the help of ubiquitous devices to the user. As the usability of IoT is increasing every day the possibility of threats is growing the same.

- Security issues like addressing, authentication and underlying embedded security needs to be focused.
- IoT devices in house such as AC, refrigerators, washing machines etc. are accessible over the internet very poorly maintained in aspects of security.
- Privacy of these devices and security over the data it transmits is very crucial.
- IoT devices such as sensors and RFID do not have any access control mechanisms associated with them, due to which these devices can gather information from each other without difficulty.
- To avoid this scenario authentication and authorization schemes are essential.
- Because of these issues maintaining security in IoT is must.

4.7.1 Vulnerabilities of IoT

Q. Explain lifecycle of an IoT device.
Q. Discuss some vulnerabilities in IoT.

The IoT devices goes through various stages in its life cycle right from manufacturing of the device to the removal of the device from the IoT based systems. Throughout these stages various vulnerabilities might occur in the device and IoT based application. Let's take a look at some issues at various stages in life cycle of IoT based devices.

- The lifecycle of the device starts from the manufacturing of the device.
- Once the device is manufactured it has to be booted up so that it could start its sensing and/or actuation and other capabilities. So these devices are made functional using bootstrapping. **Trust based bootstrapping** is the security based issue in this stage.

- There are multiple vendors available in the market so in the next step of installation and starting up the system might have **interoperability** issues.
- While installation maintaining **unique identity and secret keys** is also important.
- This setup and installed devices stays under the control of resource owner and is altered only for maintenance or software upgradation.
- While upgradation or maintenance re-bootstrapping might be necessary and these steps are continued till the devices are taken out from the IoT system.
- Distributed nature of IoT devices and dynamic network topology makes IoT based systems more susceptible for attacks.
- The movement of IoT devices and weak physical security are more reasons for vulnerabilities.

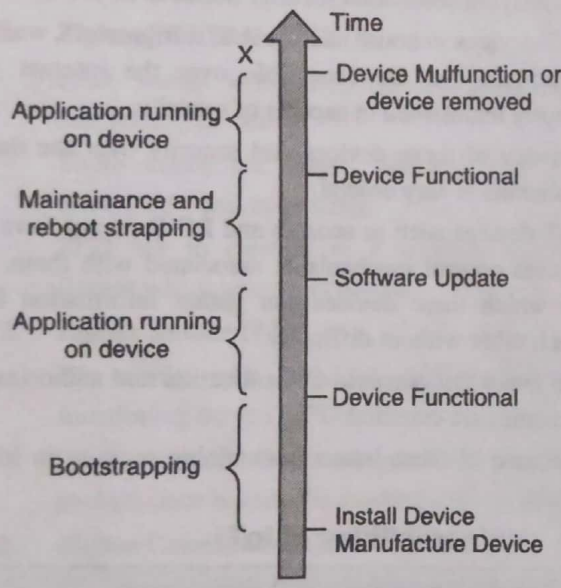

Fig. 4.7.1 : IoT device Lifecycle

- Attacks can be categorized into two types active attacks and passive attacks.
- Active attacks are in which the attacker tries to manipulate, interrupt or modify the valid data of the system.
- Some examples of active attacks are denial of service attack in which legit users are not allowed to access the services due to request overload by attackers.
- Identity theft, unauthorized access to services, information disclosure or invalid modification of data etc.
- In passive attacks the attacker only spies over or monitors the data transmission over the network. Attackers never change the actual data in passive attacks.

☞ **Some of the IoT based vulnerabilities**

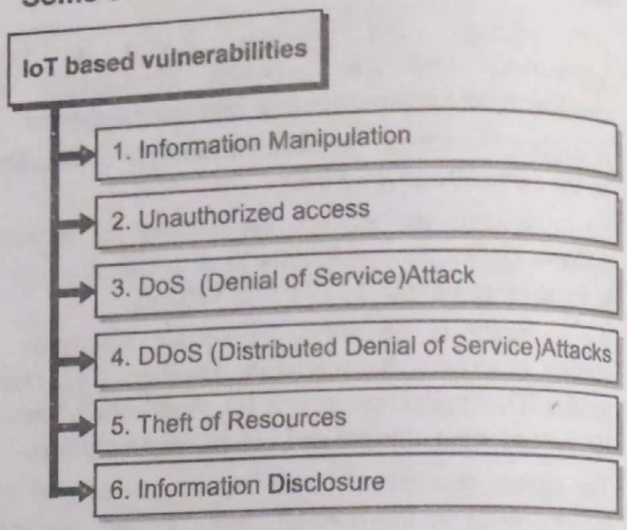

Fig. C4.3 : IoT based Vulnerabilities

➜ **1. Information Manipulation**

In this vulnerability attacker tries to change the legit data. This can be avoided secure data exchange design and robust credential and access rights should be provided.

➜ **2. Unauthorized access**

In this attack access to the resources is granted to the unknown or illegitimate user who may try to manipulate data. This can be done if attacker get hold of the authorization techniques used in the system. This can be avoided by identity based verification of users.

➜ **3. DoS (Denial of Service)Attack**

In this attack, attackers send so many requests to the server that it becomes flooded and valid request by the authorized user cannot be serviced by the server. This attack mainly focuses on avoiding access of services to the valid users.

➜ **4. DDoS (Distributed Denial of Service)Attacks**

It is a type of DoS in which the attacking node and the server both are victims of the attacker. In this attack hacker usually infect multiple systems with Trojan that is used to send request to the server causing DoS attack. In this attack server and intermediate nodes used for sending requests fall prey to the hacker.

➜ **5. Theft of Resources**

Huge data is transferred in the IoT based systems from device to device and over the server. This data is susceptible to attacks if the data transfer is not in check. This may lead to man in the middle attacks.

➜ **6. Information Disclosure**

In IoT based systems there are various possibilities of storage of data like on the device, in a local database, at a centralized node or over the cloud. These resource locations must be secured from attackers. Context

based access control to the resources can be implemented to avoid this.

Syllabus Topic : Security Requirements

4.7.2 Security Requirements

Q. List out the security requirements for IoT base systems.

Let take a look at security requirements of an IoT based system :

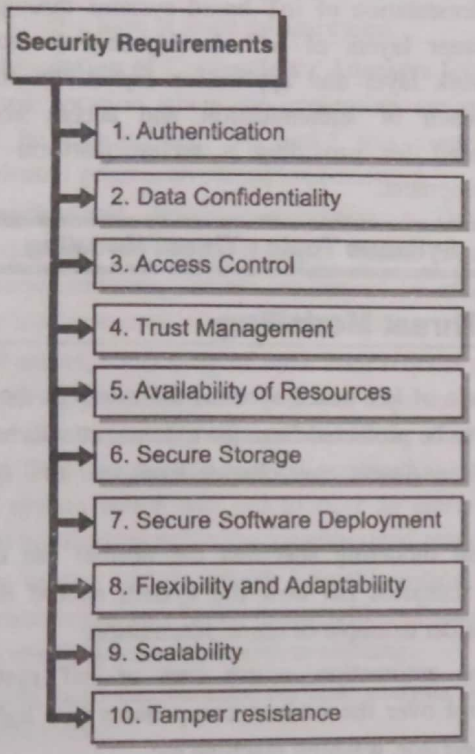

Fig. C4.4 : Security Requirements

➜ **1. Authentication**

Building identity between the two communicating devices is authentication. This is must before any kind of data exchange between devices. Lightweight and robust mechanism is needed due to variety of devices and dynamic network topology.

➜ **2. Data Confidentiality**

Protecting data from external tampering and unauthorized disclosure is important. Simple and efficient key exchange mechanism is needed.

➜ **3. Access Control**

IoT includes a vast network of scattered devices each of them having some data.

In such a huge data pool access control specifies the boundary of data to the devices. What data is accessible to which device is specified in access control.

➜ **4. Trust Management**

Due to the roaming nature of devices trust management and trust based access control is necessary.

➜ **5. Availability of Resources**

The resource in the IoT based systems must always be accessible to the legitimate user on the basis of the access control terms. DoS attacks must be prevented and access control mechanisms needs to be robust.

➜ **6. Secure Storage**

IoT based devices works with crucial data (health monitoring, home intrusion detection etc.) because of which the confidentiality and integrity of the data is important.

➜ **7. Secure Software Deployment**

Secure and well tested code must be deployed over the IoT based devices so as to prevent attacks from unknown software execution.

➜ **8. Flexibility and Adaptability**

IoT devices move around in the physical environment where it needs to interact with devices of different type over various network. Security is of prime importance in such unknown environment. So devices must be adaptable to the new environment they are moving into.

➜ **9. Scalability**

Scalability is compulsory in IoT infrastructure where the devices are added or removed from the system dynamically or even without any user control in some cases like weather monitoring system. Interoperability with other devices based on the network protocols and communication mechanisms is important. IoT solution must be developed keeping scalability in mind.

➜ **10. Tamper resistance**

IoT devices are susceptible to physical damage and intentional alteration. The device must maintain the security aspects even if it is lost into attackers' hands.

Syllabus Topic : Challenges for Secure IoT

4.7.3 Challenges for Secure IoT

Q. What are the challenges in creating secure IoT based system?

To achieve the above security requirements in the sensor and device field is a tough job. Maintaining privacy of data and things in IoT is challenge in itself.

Let's take a look at some more challenges for securing IoT based systems :

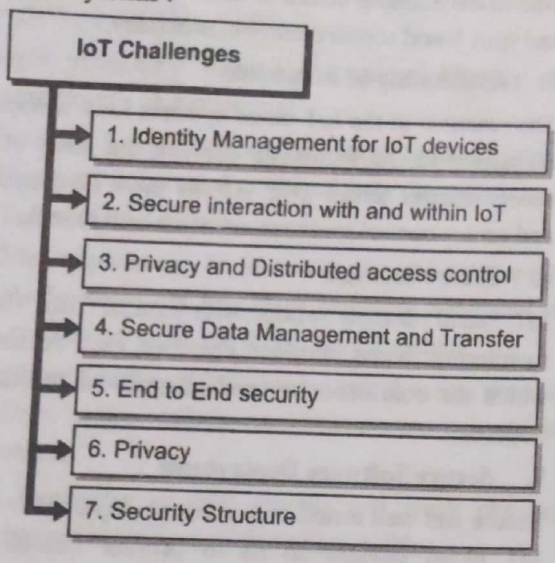

Fig. C4.5 : IoT Challenges

➜ **1. Identity Management for IoT devices**

Each IoT device needs a unique identity and identifier so various identity management concepts are required which should be known to the interacting devices. Trust management and building circle of trust is also an issue while communicating with other devices. Authentication services depends upon identity management.

➜ **2. Secure interaction with and within IoT**

Physical and virtual movement of devices needs to be managed, reliable computation, dynamic networking, and discovery of services to other devices is necessary.

➜ **3. Privacy and Distributed access control**

Access control is a crucial issue in IoT as seen earlier to achieve this identity of devices should be exchanged dynamically. Along with that credential management of devices which provide single step bootstrapping for device.

➜ **4. Secure Data Management and Transfer**

Secure storage management, separate data auditing policies for devices on user level is necessary in IoT. Along with that data exchange over the network should be secure and context driven data exchange policies needs to be implemented for secure data exchange.

5. End to End security

Device to device and service endpoint to endpoint security is a challenge in IoT based systems. Cryptographic encryption and authentication is one way to achieve this but more measures are required for robust end to end communication.

➜ **6. Privacy**

In the vast network of IoT, devices communicate with each other frequently over the internet, so privacy is a crucial element in security aspect. Over the vast communication network, internet, heterogeneous networks and devices privacy is a huge concern. Ownership of the devices is also a part of privacy.

➜ **7. Security Structure**

It is a challenging task to provide a secure structure for implementation of IoT based systems throughout the different layers of IoT development (sensor layer, network layer and application layer). An integrated approach of authentication and access control is required for providing a secure platform for IoT development.

Syllabus Topic : Threat Modeling

4.8 Threat Modelling

In case of IoT based systems the assets in the systems that need to be protected from the external attacks have to be defined first. Assets may change from one IoT system to other.

Threat modeling specifies the normal use case and negative scenarios i.e. how the system should not work. Let's consider example of home automation.

- Home automation, a use case of IoT system, has control over the various components like lights, fans, refrigerator, washing machine etc.
- In this use case owner and residents' information is stored in the system for accessing various components like providing entry to the house.
- Asset in this systems are, data typed by the user, data stored in user's devices, data stored at in the database and all the data transmitted over the network. **(Assets Identification)**
- Attacker tries to target these assets for gaining crucial information which can be used for malicious activities like unauthorized access to home. **(Possible Attacks)**
- Passkey (secure key) which is used to grant entry to the user in home must be protected via authorization, authentication and integrity of the data must be maintained. **(Security Measures)**

4.8.1 Use Cases and Misuse Cases

Q. What are some misuse cases in IoT security?

- Use cases define the normal behavior of the system while misuse cases define the malicious behavior when system is under attacks.
- In home automation system granting access to the owner **(actor)** via valid credentials is a use case while granting unauthorized access is a misuse case. In misuse case attacker is the actor who achieves the user's credentials **(asset)** and gains access to the house.

Some misuse cases

Let's take a look at some misuse cases :

1. **Manipulation of Credentials :** Attackers focus on the storage location where the credentials are stored and can be manipulated. To avoid this context based credential generation can be implemented.
2. **Unauthorized data transmission :** Data traffic monitoring is must in IoT based systems as unauthorized data can be transferred by attacker that may lead to expose private information.
3. **DoS attack :** Flooding of open door request may deny owner entry to the house. Multiple requests may block the valid request by the owner. If the door remains open DoS attack can focus on notification manager of IoT system which will fail to raise an alarm for door open notification providing unauthorized entry.
4. **Man in the middle attack :** In this misuse case the attacker might note down the activities of residents that may result in future intrusion or stalking.

- Misuse cases can be used for risk analysis of the IoT based systems.
- Using these misuse cases threat analysis can be done and necessary design changes can be incorporated in IoT system.

4.8.2 Activity Modeling of Threats

- Activity modeling of threats reveals stepwise actions taken when the attacks are happening on the system.
- Developers can figure out exactly where to define the security measures if we know the exact actions happening in the attack.
- Attackers might have defined various ways to attack the system which can be represented as a tree where the root node specifies the attacks end goal and intermediate nodes defines the steps to reach the goal.
- We have seen types of attacks to be active and passive in the same way attackers can also be distinguished:
- **Strong Active:** These types of active attackers can compromise intermediate source and destination nodes.
- **Strong Passive:** These types of passive eavesdropper can gain control over the complete transmission path.
- **Weak Passive:** These types of passive eavesdropper have limited capacity to watch over the transmission and cannot gain control over the complete transmission path.
- Man in the middle attack is possible at the time of key exchange for authentication as authenticating devices do not have knowledge about each other before transfer of data. While exchanging device identities man in the middle attack may lead to identity theft (of devices).
- Nodes in IoT based systems usually have limited computational and storage capacities because of which these nodes are susceptible to DoS attack.
- Another form of man in the middle attack is replay attack where the attacker sniffs the legit key sender is transferring and attacker resends the same data. So attacker sends the valid request or key to which receiver send the response or entry into the system thinking that actual sender is making the request.

4.8.3 IoT Security Tomography

Q. Explain various possible attacks in different layers of IoT.

- In this section we will study various attacks possible at various communication layers of transmission. Table 4.8.1 describes various attack possibilities for the IoT device.

Table 4.8.1

Layer	Attacks
Transport Layer	Send wrong data, Inject incorrect control packets
Network Layer	Routing Loop, Wormhole attack, Network Partitioning
MAC Layer	Spoofing, Buffer Overflow, Eavesdropping, **OS level threats**
RF Layer	Complete Jamming, Eavesdropping **Hardware / Sensor level threat,**

Transport Layer Threats

- TCP and UDP communication protocols are used in this layer which can be tampered with.
- TCP connection is targeted by the user and in connection less UDP attacker tries to increase the data loss.
- **Sending wrong data** triggers error correction policy in TCP based communication wasting CPU power on the receivers end.

- **Injecting incorrect packets** creates interference between communications. This also leads to de-synching data at receivers end and might result in connection failure.

Network Layer Threats

- In Network layer attacks the attacker tries to manipulate the routing of packets.
- Attacker sends new packets in the network or modifies existing packets in such a way that packets follow cyclic routes never reaching the destination. This is called as **routing loop** attacks.
- **Network partitioning** attack focuses on dividing the existing network which disrupts the overall communication. Attackers focus on boundary nodes or nodes which have low energy/power that will be depleted soon. Taking out such nodes from the whole network partitions the network in disjoint sets.
- **Wormhole attack** is generated by one or two nodes together. In this attack two nodes create a communication tunnel in which a packet is transmitted and replayed locally. If only one node is used in wormhole attack it creates fake neighboring nodes that do the same thing. This either results in packet dropping or selective packet forwarding which detains packets from reaching destination.

MAC Layer Threats

- Unauthorized devices can participate in the communication by **spoofing** their identities as if authorized devices are communicating.
- **Eavesdropping** attack is possible in broadcast communication medium.
- The node which is in physical range of the sender can receive the messages and can read the communication happening due to broadcast method.

RF (Radio Frequency) Layer Threats:

- **Eavesdropping** attack is possible in broadcast RF communication medium. The node which is in physical range of the sender can receive the messages and can read the RF communication happening due to broadcast method.
- **Replay attacks** are possible in RF communication that leads to dropping the packets or damaging the state of packet (data alteration).
- Increase in the noise level leads to blocking the communication. If noise level keeps on increasing it eventually jams the entire communication.
- Other possible attacks in IoT based system targets **Operating System, Sensor Level** and **Hardware Level :**
- **Buffer overflow** attack tries to overload the device by that. Due to the data overflowing binary code might get altered resulting in ambiguous behavior. This is **Operating System** level attacks.
- Sensor nodes that can be physically accessed by the attackers can be manipulated by altering the code directly. **Direct reprogramming** can be done in ROM level.
- **Hardware level** changes can also be made if devices can be directly / physically accessed by attackers.

Syllabus Topic : Key elements of IoT Security - Identity Establishment

4.9 Key elements of IoT Security

- Attackers target the weakness in implementation of the security policies that can be used to breach into the IoT environment.
- IoT devices are smart objects that communicate with each other, such devices should have privacy of data, secure authentication model, data ownership must be specified and trust must be established while communicating with each other and over the cloud.
- To achieve all this following key points must be focused to achieve IoT security:

4.9.1 Identity establishment

Q. Explain Identity Establishment issue with respect to IoT Security.

- Building secure identity between the two communicating entities for data transfer is called as identity establishment or authentication.
- An entity in the communication can be a single device, a single user, set of users or can also be an organization.
- Identity establishment confirms the source of the electronic data or documents that is going to be transmitted over the network.
- In communication network a service provider's services are consumed by the user, so in a typical communication developer has to consider user identity (users) and service identity (service provider). In IoT environment we also have device identity.
- Ensuring identity of the device confirms the trust in the communication.
- For authentication private key cryptography and public key cryptography means can be used.
- Private Key cryptography needs higher storage requirement to store keys and also require high processing power, which makes this inefficient for low powered IoT devices.

- Public Key cryptography doesn't need to distribute keys to devices before starting the communication and also has lower memory requirement thus overcoming the drawbacks of private key cryptography for IoT devices.

Syllabus Topic : Access Control

4.9.2 Access Control

Q. Explain Access Control issue with respect to IoT Security.

- Access control mechanism specifies which data is accessible to which entity in the network. It basically defines boundary of data access for the devices.
- Access control avoids the unauthorized access to the resources.
- Access to each device is provided based on their authentication. Device trying to access resource is authenticated and then given access to the resources based on their access level.
- As IoT provides dynamic and ad-hoc nature of communication with distributed devices, authentication and access control mechanism is important for IoT applications.
- Access control mechanisms are implemented using Access Control Matrix (ACM).
- ACM column defines resources to be accessed and ACM row defines users or whoever wants to access the resources.
- Based on ACM, Access Control List (ACL) and capability based models are developed.
- In ACL each resource is associated with permissions (read, write, modify, etc.), while capability based models uses key based approach which defines authentication and access rights associated with the key.

Syllabus Topic : Data and Message Security

4.9.3 Data and Message Security

Q. Explain Data and Message Security issue with respect to IoT Security.

- Data and message security is concerned with three sore issues: source of data should be authentic, detection of any modification in data and data must be private.
- The data can either be residing on a local and/or cloud storage or in transmission phase over the network.
- Source of data (origin) should be authentic and location of the device generating data must be kept secured from the attackers.
- Any modification, insertion or duplication of data must be avoided to maintain the integrity of the data.

Syllabus Topic : Non-Repudiation and Availability

4.9.4 Non-Repudiation and Aavailability

Q. Explain Non Repudiation and Availability issue with respect to IoT Security.

- Repudiation means rejecting the fact that one entity has sent or received the message.
- So because of non-repudiation, when a message is sent the receiver can prove that which sender has sent the specific message. And the sender can prove receiver has received the message.(Eg: Blue ticks on WhatsApp)
- This is very important in case of security of IoT devices for pinpointing attack origin.
- Availability defines that the resources in the network must be accessible to the authenticated devices on demand.
- Availability focuses on keeping all the hardware backing up the resources (eg: server, power backup etc.) well maintained and up to date.
- It also tries to avoid DoS attacks and system bottleneck that minimizes availability.

Syllabus Topic : Security Model for IoT

4.9.5 Security Model for IoT

Q. Explain security model for IoT.

- Security model for IoT represents the security features that should be followed by an IoT application.
- All the security features learned in above section are combined in this single model.
- The security model of IoT can be represented by a cube with three dimensions representing
  1. **security** – authorization,
  2. **trust** – repudiation and
  3. **privacy** – respondent.
- The intersection defines the specific characteristics of the IoT security model.
- As shown in Fig. 4.9.1, security of the IoT based application focuses on Authorization, Identification and Authentication, Confidentiality, Integrity, Non-repudiation and Availability.
- Privacy focuses on Owner's privacy, user's privacy, Ethics of communication, Laws concerned and accused's privacy.

– While trust focuses on Beliefs, credentials, delegation (allocations), recommendation and repudiation.

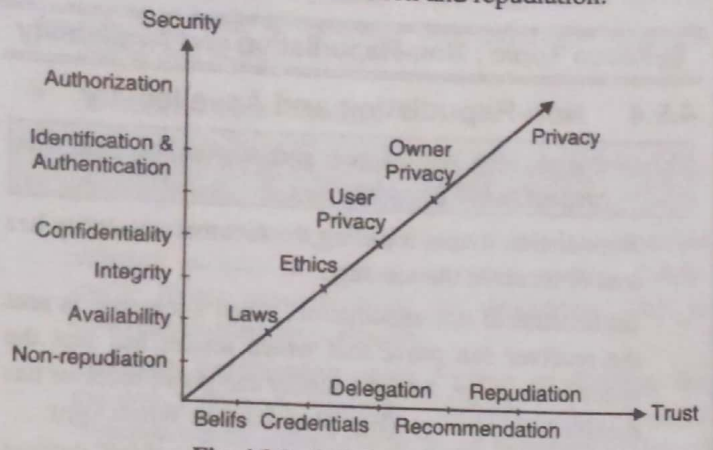

Fig. 4.9.1 : IoT Security Model

4.10 Exam Pack (Review Questions)

☞ **Syllabus Topic : Protocol Standardization for IoT, Efforts**

Q. What are some protocol standardization efforts taken for IoT? *(Refer section 4.1.1)*

☞ **Syllabus Topic : M2M and WSN Protocols**

Q. Explain M2M protocol standardization. *(Refer section 4.2.1)*

Q Explain WSN protocol standardization. *(Refer section 4.2.2)*

☞ **Syllabus Topic : SCADA and RFID Protocols**

Q. Explain SCADA protocol standardization. *(Refer section 4.3.1)*

Q. Explain RFID protocol standardization. *(Refer section 4.3.2)*

☞ **Syllabus Topic : Issues with IoT Standardization**

Q. Explain the issues with IoT standardization. *(Refer section 4.4)*

☞ **Syllabus Topic : Unified Data Standards**

Q. Explain various data standards used in IoT data exchange. *(Refer section 4.5)*

☞ **Syllabus Topic : Protocols - IEEE 802.15.4**

Q. Explain IEEE 802.15.4 protocols in details. *(Refer section 4.6.1)*

☞ **Syllabus Topic : BACNet Protocol**

Q. Explain BACNet protocols in details. *(Refer section 4.6.2)*

☞ **Syllabus Topic : ModBus**

Q. Explain ModBus protocols in details. (Refer section 4.6.3)

☞ **Syllabus Topic : KNX**

Q. Explain KNX protocols in details. *(Refer section 4.6.4)*

☞ **Syllabus Topic : Zigbee Architecture, Network Layer, APS Layer**

Q. Explain ZigBee protocols in details. *(Refer section 4.6.5)*

☞ **Syllabus Topic : IoT Security - Vulnerabilities of IoT**

Q. Explain lifecycle of an IoT device. *(Refer section 4.7.1)*

Q. Discuss some vulnerabilities in IoT. *(Refer section 4.7.1)*

☞ **Syllabus Topic : Security Requirements**

Q. List out the security requirements for IoT base systems. *(Refer section 4.7.2)*

☞ **Syllabus Topic : Challenges for Secure IoT**

Q. What are the challenges in creating secure IoT based system? *(Refer section 4.7.3)*

☞ **Syllabus Topic : Threat Modeling**

Q. What are some misuse cases in IoT security? *(Refer section 4.8.1)*

Q. Explain various possible attacks in different layers of IoT. *(Refer section 4.8.3)*

☞ **Syllabus Topic : Key elements of IoT Security - Identity Establishment**

Q. Explain Identity Establishment issue with respect to IoT Security. *(Refer section 4.9.1)*

☞ **Syllabus Topic : Access Control**

Q. Explain Access Control issue with respect to IoT Security. *(Refer section 4.9.2)*

☞ **Syllabus Topic : Data and Message Security**

Q. Explain Data and Message Security issue with respect to IoT Security. *(Refer section 4.9.3)*

☞ **Syllabus Topic : Non-Repudiation and Availability**

Q. Explain Non Repudiation and Availability issue with respect to IoT Security. *(Refer section 4.9.4)*

☞ **Syllabus Topic : Security Model for IoT**

Q. Explain security model for IoT. *(Refer section 4.9.5)*

□□□

UNIT V

CHAPTER 5

Web of Things and Cloud of Things

Syllabus Topics

Web of Things versus Internet of Things, Two Pillars of the Web, Architecture Standardization for WoT, Platform Middleware for WoT, Unified Multitier WoT Architecture, WoT Portals and Business Intelligence. Cloud of Things : Grid/SOA and Cloud Computing, Cloud Middleware, Cloud Standards - Cloud Providers and Systems, Mobile Cloud Computing, The Cloud of Things Architecture.

Syllabus Topic
Web of Things versus Internet of Things

5.1 Web of Things versus Internet of Things

Q. What is Web of Things (WoT)?

Now a day Internet and World Wide Web (WWW) terms are used as if they represent the same thing. But Internet existed long before WWW. The minute differences between these two terminologies are explained in below section.

- Internet terminology is used to represent the vast network of interconnected computers distributed all over the world.
- Internet refers to the actual connection (wired, wireless) or a path from one computer / resource to the other.
- While WWW is the method to access the Internet with the help of Hyper Text Transfer Protocol (HTTP) using address / links like *www.somthing.extension*.
- HTTP is one of the many protocols available to access resources available on the Internet.
- As shown in Fig. 5.1.1 Internet can be considered as a huge container containing parts like WWW, Email, FTP, P2P, Chat, Telnet, etc.

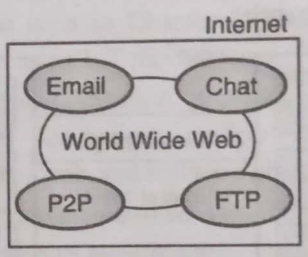

Fig. 5.1.1 : Internet and it's part

- Internet and WWW are treated as if they are interchangeable but it's not so, WWW can be considered as the most popular method in Internet.
- To give a general example, consider Internet is a Restaurant then WWW is the most popular Dish of that Restaurant.
- In the same sense Web of Things (WoT) is considered as the next step of IoT an application oriented view to the IoT.
- WoT focuses on data integration and interaction of objects.
- WoT provides techniques, software architecture styles and programming approaches to allow the real world objects or "things" to become a part of WWW.
- In short Web of Things provides a way to implement Internet of Things by integrating the Web (Application Layer) and Internet (Network Layer).

☞ **Example**

1. Arduino, Raspberry Pi, Beagle Bone Black etc. boards provide a way to realize IoT based applications.

2. Japan Geiger Map is a project that displays live radiation counter across Japan.
3. AgSphere is a platform that provides web based solutions for agricultural products.

Syllabus Topic : Two Pillars of the Web

5.2 Two Pillars of the Web

Q. What are the two pillars of the Web? Explain in brief.

The technologies like HTML, HTTP and URL in coordination with the TCP/IP protocol suit provided strong base to the Internet using which beneficial applications like browser were created. These applications were supported by the application servers which spread these applications to the world. These two techniques form the recognized two pillars of the web. Let's discuss this in details.

- As discussed earlier Web of Things specifies application view to the IoT.
- The overall architecture of WoT can be defined by two pillars of the web.

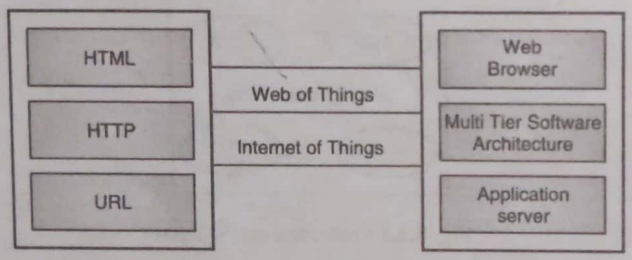

Fig. 5.2.1 : Two Pillars of the Web

First pillar consists

1. Hyper Text Markup Language (HTML)
2. Hyper Text Transfer Protocol (HTTP)
3. Uniform Resource Locator (URL)

- First pillar techniques provides a way to describe a resource or service, how to locate it and how to distribute / transfer it over the Web.
- Applications provide a way to access the resources provided by various service providers. This is nothing but a front end to the services.
- Hyper Text Markup Language (HTML) is used to create resources and applications which can be accessed over the Internet. It is used to create electronic documents which can be viewed and shared via Internet.
- HTTP is a technology that is used by the WWW for defining how to format and transmit the messages and how the web servers and browsers should work in response to various commands.
- Resources are located with the help of Uniform Resource Locator (URL) over the internet.

Second pillar consists

1. Web Browsers
2. Multi-Tiered Software Architectures
3. Application Servers

- Second pillar techniques provide a way to access the resources or services via URLs, how to design the Web Application and how to run and maintain the web application.
- Web browsers provide a way to access the resources defined in the first pillar techniques using URLs.
- Browsers takes users to a location where they can utilize the services defined with the help of the first pillar.
- Application servers are the software framework that provides an environment where the applications can run.
- Application server offers facilities like APIs, runtime libraries, programming languages, database connectivity, which makes the application development and administration easier.
- The multi-tiered architecture design for web applications separates the implementation perspective and makes the data management easier.
- A general three tiered architecture defines a front end that is accessible to the end user, a middleware or business logic that handles the overall functionalities of the application and the backend database server which stores and manages the incoming and outgoing data.

Syllabus Topic
Architecture Standardization for WoT

5.3 Architecture Standardization for WoT

This section describes the various frameworks and standards used in four pillars of IoT i.e. M2M, RFID, SCADA and WSN with respect to WoT. In second section below Multi-tier WoT Architecture and some Business Intelligence implemented worldwide are discussed.

Syllabus Topic : Platform Middleware for WoT

5.3.1 Platform Middleware for WoT

Middleware of WoT applications and IoT as well focuses upon communication interface and application's

business logic in depth. So middleware for WoT helps bring the IoT applications to the WWW.

- WoT / IoT tries to connect everyday objects like domestic appliances, sensors, actuators and some other embedded systems and provide a unified application.
- To realize this application sense of IoT, WoT uses object oriented approach to view everything as objects.
- Many such techniques like Object Oriented Design, JavaScript Object Notation (JSON), Common Object Request Broker Architecture (CORBA), Document Object Model (DOM) etc. are famous in WoT.

M2M Middleware Standards

Q. Explain M2M Middleware Standards in brief.

As seen in Chapter 3, M2M is the automated data exchange between machines may be real or virtual (instance of a software), without or minimal human intervention.

The key elements of M2M Standardization are as follows :

1. **M2M Device**

   M2M devices are able to sense data and transfer it to expected device regularly or on receiving a request.

2. **M2M Area Network**

   This layer provides connectivity between M2M devices and M2M Gateway. Examples of this layer are Short Ranged Devices, Bluetooth, KNX, ModBus etc.

3. **M2M Gateway**

   M2M Gateway makes sure that the M2M devices are interconnected with each other and coordinators via available communication networks.

4. **M2M Communication Network**

   A Communication network provides a platform for connecting the M2M Application servers and M2M gateway. Examples are WiMax, W-LAN, Satellite etc.

5. **M2M Application Sever**

   - Application server manages the data processing through various applications and how it is used for business logics.
   - M2M Middleware handles three core layers which are M2M Gateway, Communication Network and Application Server.

WSN Middleware Standards

Q. Explain WSN Middleware Standards in brief.

Wireless Sensor Network (WSN) Standardization approach is done by Ubiquitous Sensor Networks (USN) Standardization which includes following components.

1. **USN Application Platform**

   This layer provides a technical framework for application development and service delivery.

2. **USN Middleware**

   This layer provides functionalities for sensor connectivity, network management, sensor data mining and event management.

3. **Network Infrastructures**

   USN makes use of existing networks instead of creating a new one from scratch.

4. **USN Gateways**

   Gateways are used to connect sensor network with other networks to increase the reach of devices.

5. **Sensor Networks**

   This is the core network of sensors interconnected via IP based addressing directly or using gateways to connect indirectly.

SCADA Middleware Standards

Q. Explain SCADA Middleware Standards in brief.

- SCADA as seen in Chapter 3, focuses on industrial automation using IoT mechanisms.
- SCADA Standardization provides a framework for information exchange model, device interfaces, services, protocols and software object modeling.
- SCADA middleware platform focuses on following points in industry automation.
  1. **Interaction :** This step manages the interactive interfacing between Users / Human Controllers and Smart Objects distributed in the industry.
  2. **Communication Mechanisms :** The information exchange between different smart objects distributed in the industry.
  3. **Context :** Context in SCADA defines the conditions that form the setting for a particular event in an industry. It manages data distribution and processing according to the context.
  4. **Secure Distributed Storage :** This layer manages the secure storage of the information generated using smart objects in the industry.
  5. **Additional Tools :** Additional tools are also available using which operators or editors can perform manual work on the SCADA systems.

RFID Middleware Standards

Q. Explain RFID Middleware Standards in brief.

- As discussed in Chapter 3, RFID provides unique identification techniques for the "things" which can be recognized using RFID reader.

- RFID is the most complete and standardized approach in all of the four pillars of IoT.
- As discussed in Chapter 3, EPC Global describes the standards for RFID implementation.
- RFID middleware specifies the business logic that is used to filter the data and extract the useful information from the huge data.

1. **Manage Devices**

   In this layer middleware have to manage devices with RFID tags and the RFID readers that are used to detect RFID. In this layer middleware can check the performance of RFID reader and detect new ways to improve it.

2. **Collect and Integrate Data**

   All the RFID tags read by the RFID readers is directly processed by the middleware. So collecting RFID data and arranging it as per need is middleware's work.

3. **Structure and Filter Data**

   Based on the identification, large data about the product (on which RFID tag is embedded) is available. Middleware decide how this data is to be categorized and which data is to be used as per the functionalities. It also removes redundant and repetitive data.

4. **Tag ID Association**

   In case of RFID, middleware can allocate and manage tag ID numbers to the RFID, where thousands and more RFID tags are in use. Tag ID management is very useful in making sense of devices and RFID to the developers. Middleware can be used to allocate tag IDs only to relevant RFID reads to make the application more understandable.

- Example of RFID middleware working standards is explained by Cross UBiQuitous Platform (CUBIQ) a project in Japan.
- RFID tag information is recorded by mobile terminals with RFID readers and it also stores the location based data.
- The mobile terminals are connected via a CUNIQ architecture which can share data of other RFID devices read by other terminals.
- This data can be searched to locate the specific RFID tag using this architecture.

Syllabus Topic : Unified Multitier WoT Architecture

5.3.2 Unified Multitier WoT Architecture

Q. Explain unified multitier WoT Architecture in details.

Based on the middleware perspective studied in last section IoT based application realization can be done with the help of WoT. The following framework satisfies the middleware requirement for WoT. The Fig. 5.3.1 shows middleware approach for IoT which can also be considered for WoT as WoT is nothing but application overview of IoT.

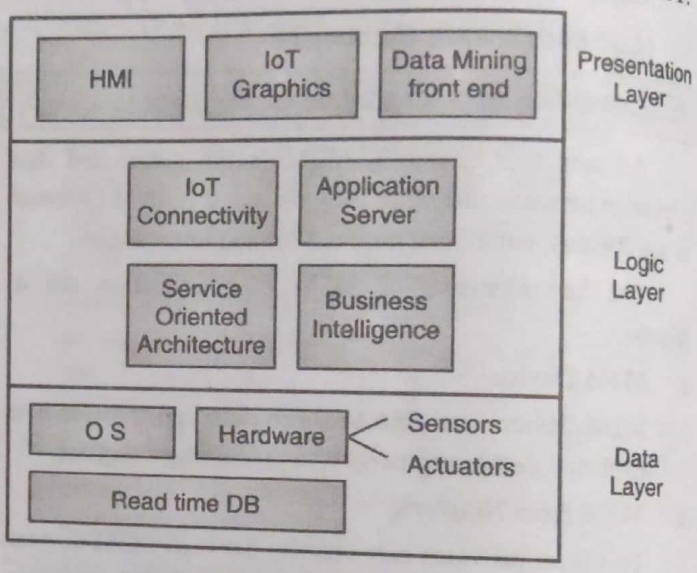

Fig. 5.3.1 : Multitier IoT Middleware

- Multitier IoT Middleware adds the IoT based connectivity and service oriented layer in the existing three tiered architecture.
- As shown in Fig. 5.3.1 the basic three tiered architecture consists of the data layer, logic layer (middleware) and presentation layer (User Interface).

☞ **Data Layer**

- The data tier handles all the data generation and storage in WoT.
- This layer handles data gathered from numerous distributed sensors and provides a mechanism to store it securely.
- This layer needs operating system that regulates the flow of data and other system activities.
- Along with the usual database (DBMS), WoT and IoT systems might also need **Real time Databases** based on the application requirement.

☞ **Logic Layer (Middleware)**

- Logic Layer defines middleware and the application logic of the whole IoT and WoT system. This layer regulates the working of the core working of the application.

- For IoT and WoT based systems IoT based connectivity and service oriented architecture is add on compared to the usual three tired architecture.
- The smart objects and device distributed in the field for implementing IoT based systems i.e. WoT needs a communicating mechanisms that is used to exchange this data over the network.
- This is implemented using IoT based networking techniques like M2M gateways, FieldBus, WSN, SCADA etc.
- Service oriented architecture defines how the data will be represented to different layers. This layer specifies the data representation to the applications, the content management and business oriented workflow of the system.
- The basic middleware techniques like application server utility using IIS, .NET framework is also implemented in logical layer.

Presentation Layer

- Presentation layer defines the User Interface of the WoT to the outside world.
- This layer provides a mechanism to access the information and services provided by the bottom two layers.
- IoT based systems and WoT has IoT based graphical representation that provides functionalities like report generating, data mining, HMI Tools in SCADA, Decision support system etc.

Syllabus Topic
WoT Portals and Business Intelligence

5.3.3 WoT Portals and Business Intelligence

This section briefly discusses how the data access is done by the users in WoT and how Business Intelligence (BI) is effective in WoT.

- A web portal provides a point of access to the WWW using URLs and addresses.
- These links, webpages and web portals gives combined information from the diverse data sources in IoT environment.
- Web portals have portlets, an application in web portals which receives requests and returns appropriate information.
- As discussed in Chapter 3, web portals are of two types horizontal and vertical.
- Horizontal portals cover many areas while vertical web portals focus only on one functional area.
- Java Specification Request (JSR168) standard was developed in around 2001 which revolutionized web portals.
- JSR168 allows various web portals to communicate between each other even with cross platform vendor solutions.

Some famous examples of web portals for IoT

Q. Give examples of some WoT Portals.

1. **Pachube (now xively.com)**
   This platform provides a IoT solution as per the user requirement where you can connect various devices and handle them online easily.
2. **Microsoft Hohm / Google Power Meter**
   Hohm by Microsoft and Power Meter by Google are IoT based solutions where user can monitor their household power usage and create plans on how to reduce energy wastage.
3. **Sun SPOT (Small Program Object Technology)**
   It is a WSN developed by Sun Microsystems provided a way to program the IoT needs using Java programming language.
4. **Sensor Map Microsoft**
   Sensor Map was a Microsoft project users can publish their live data collected by the sensors or other smart devices.

Business Intelligence (BI)

Q. Explain Business Intelligence associated with WoT.

- The huge data collected by large number of sensors, is available in raw format. To make sense of this data the WoT middleware defines a business logic which converts this data into meaningful information. Such logic is also known as BI of the WoT.
- Data mining is a well-known mechanism to implement BI.
- Data mining refers to finding patterns and establishing relationship between the huge dataset.
- Decision Support Systems is another mechanism of implementing BI.
- In Decision Support Systems provides the data analysis and simulations as per need which makes taking decisions easier for the system or users.
- BI provides historic, present and futuristic views to the system by using the available data.

Common uses of BI

1. Data report generation
2. Data analytics
3. Business Performance Monitoring and management
4. Benchmarking
5. Data Mining
6. Predictive Analytics and so on.

Syllabus Topic : Cloud of Things - Grid/SOA and Cloud Computing

5.4 Cloud of Things

As seen in the Chapter 1 – IoT levels and deployment templates, cloud play very important role IoT application development. Cloud of things tries to integrate IoT with cloud computing to provide higher scalability and improved performance of IoT.

5.4.1 Grid/SOA and Cloud Computing

Q. Explain Cloud Computing as integration to Grid computing.

Just like the IoT, Cloud computing also relies on the communication network for providing various services.

- Cloud Computing is derived from grid computing, where both technologies tries to use computing resources from multiple locations.
- Both the approaches provide a distributed and parallel programming environment.
- Parallel Virtual Machine (PVM), Message Passing Interface (MPI) and job scheduler plays important (for resource management) role in building the middleware for grid computing and cloud computing.
- One key feature of cloud computing and grid computing is that both techniques provide a single system image (parallel virtual working station) and hides the essential implementation based on clustering technologies.
- Example: Amazon Web Services which provide a combined user interface using web services over Internet.
- Cloud Computing also supports virtualization creates virtual versions of the resources like servers, storage, operating systems etc.
- Virtualization is done in two types, Single System Virtualization (SSV) and Multi System Virtualization (MSV).
- SSV technique provides one to many virtualization. Multiple operating systems are simulated on the same computer hardware. This provides increase in efficiency and resource utilization.
- VMWare is one such example and is very helpful for Cloud Computing providing various OS support in the same environment.
- MSV technique provides many to one virtualization. MSV provides a collective performance of many systems to a single node.
- MSV provides a centralized service providing node that collects large number of application services and provides high performance computing.
- Many to one virtualization form the base of Cloud Computing.
- SSV can be used with lower level nodes of MSV but is not compulsory.
- The resources described by cloud (storage and computing) are delivered to the end user by using SOA (REST based web services, SOAP etc.) using Internet.
- Service Oriented Architecture (SOA) provides a software design that helps communicate between application components using communication protocols on the network.
- SOA can be used in all the layers i.e. IaaS, PaaS and SaaS of cloud computing.
- Thus cloud computing integrates grid computing and SOA to provide services and solutions to the end users over the internet.
- Fig. 5.4.1 shows Cloud as integration of grid and SOA.

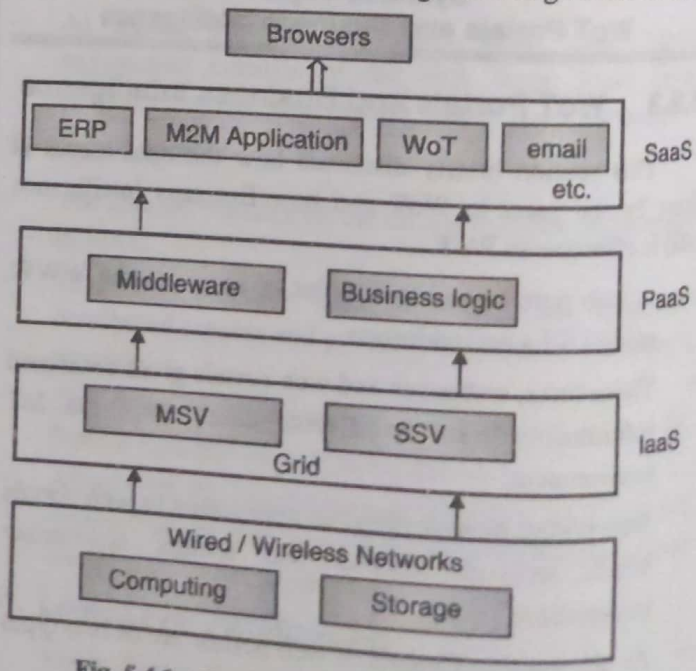

Fig. 5.4.1 : Cloud as integration of grid and SOA

- The basic resources of cloud i.e. storage and computing is provided using distributed Grid Computing

connected using wired or wireless communication standards.

- Cloud infrastructure is built using MSV and SSV which uses SOA to provide the resources to the end users using standalone softwares, applications or browsers.

Syllabus Topic : Cloud Middleware

5.4.2 Cloud Middleware

Q. Explain Cloud Middleware Architecture.

Just like the IoT and WoT seen earlier, Cloud Computing also has a multitier architecture with strong middleware perspective. Fig. 5.4.2 shows cloud middleware architecture.

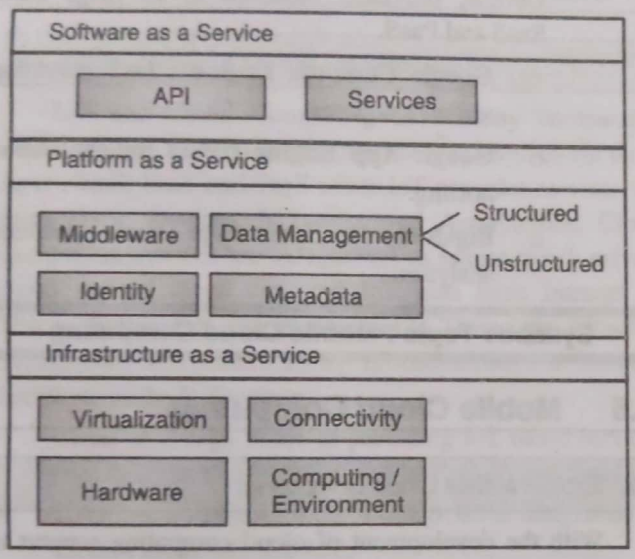

Fig. 5.4.2 : Cloud Middleware

- Cloud provides its services in three categories IaaS, PaaS and SaaS, and Cloud Middleware must focus on these categories in its architecture.
- SaaS utilizes the functionalities of IaaS Middleware and PaaS Middleware to provide Software as a Service (applications).
- At lower level the middleware helps reduce the SSV virtualization overhead of virtualization.
- IaaS middleware include components like network management, system management, configuration of systems and virtualization etc. providing functionalities for hardware and infrastructure of Cloud.
- IaaS middleware includes all Grid management, SSV and MSV cluster management and scheduler as well.
- PaaS middleware handles identity and access management of computing devices, data management which include managing metadata and categorizing structured and unstructured data on the Cloud.
- PaaS includes all business level solution and services that are required for SaaS.
- Thus Cloud Middleware consists of two middleware layers for IaaS and PaaS.
- SaaS uses IaaS and PaaS middleware functionalities to provide its services and doesn't have SaaS middleware separately.

Syllabus Topic : Cloud Standards

5.4.3 Cloud Standards

Q. Explain Cloud Standards.

- Cloud Computing provides a model for global, convenient and on demand use to the resources and services shared over the "cloud". Cloud Model consists of following key features :
  - **Service Models** : IaaS, PaaS and SaaS.
  - **Four Deployment Models :** Private Cloud, Public Cloud, Community Cloud and Hybrid Cloud
  - **Five essential characteristics :** on-demand self-service, Broad Network Access, Resource Pooling, Rapid Elasticity and Measured Services
- Cloud standards work on these features and create a standard way to provide above services.

Following list describes some of the cloud standardization organizations and some work done by them.

- **National Institute of Standards and Technology (NIST) :** Defined Cloud Computing
- **Distributed Management Task Force (DMTF) :** Cloud Management Initiative, Open Virtualization Format, Open Cloud Standards Incubator etc.
- **European Telecommunications Standards Institute :** Defines cloud definition considering Europe standards and standards for Information and Communications Technologies (ICT), including fixed, mobile, radio, broadcast and Internet technologies.
- **Cloud Management Working Group (CMWG) :** Cloud Infrastructure Management Interface (CIMI), Architecture for Managing Clouds, defines cloud service lifecycle, etc.
- **Standards Acceleration to Jumpstart Adoption of Cloud Computing (SAJACC) :** Defines 25 use cases for Cloud Computing
- **Open Cloud Consortium :** Open Cloud Testbeds, Open Science Data Cloud, benchmarks, reference implementation.
- Many such standardization bodies are working for betterment of Cloud society.

Syllabus Topic : Cloud Providers and Systems

5.4.4 Cloud Providers and Systems

Q. Give example of some cloud service providers in IaaS, PaaS and SaaS.

Just like the IoT, Cloud Computing has great market opportunity. SaaS provides the most market revenue out of all cloud services in IaaS, PaaS and SaaS.

- SaaS provides solutions that can be used by end-users directly making it popular and has higher value generation possibility comparatively.
- IaaS helps reduce the cost of infrastructure requirement to the organization, by providing cloud infrastructure for organizations.
- PaaS middleware is sold independently and is considered as SaaS when only a part of PaaS is hosted as a service.
- Many cloud service providers are available and many of them are working for cloud development providing Open Source application support.
- Some of the service providers are listed below :

Open Source IaaS and PaaS projects

1. **OpenStack :** A set of software tools for building and managing cloud computing platforms for public and private clouds.
2. **OpenNebula :** OpenNebula is a cloud computing platform for managing heterogeneous distributed data center infrastructures.
3. **RedHat Cloud :** builds and manage an open, private IaaS cloud at a much lower cost than alternative solutions.
4. **Other products are :** AppScale,Traffic Server, Cloudera (Hadoop), Puppet, Scalr, Eucalyptus, Amanda/Zmanda, cloud.com, Cloud Stack, TPlatform, Enomaly, Joyent, Globus Nimbus, XCP, Reservoir, and so on.

Open Source SaaS Projects

1. **Zoho :** Suite of online productivity tools and SaaS applications.
2. **Pentaho :** Pentaho's open architecture and standards provide solution for embedding into cloud-based applications, supporting with existing enterprise architectures.
3. **SourceTap :** SourceTap gives CRM application and provides highly flexible tool that is helpful to needs of sales managers and the sales representative.
4. **Other products are :** Phreebooks, Palo BI Suite, Knowledge Tree, Jaspersoft, SugarCRM, OpenKM, Processmaker, Alfresco, SourceTap, Open ERP, Openbravo, Collabtive, , Feng Office, TimeTrex, Zimbra, OpenProj, Orange HRM, JStock, GlobaSight, Compiere, openSIS and so on.
5. **Amazon Web Services :** Provides highly scalable and services for website hosting, storage, high performance computing, e-commerce solutions, etc.
6. **Microsoft Azure Cloud Services :** Cloud services by Microsoft providing vast range of applications like virtual machines, web apps, developers tools.
7. **Google Cloud Platform :** Cloud services by Google, providing solutions in all range IaaS, SaaS and PaaS.
   - **Google Compute Engine :** IaaS providing virtual machines
   - **Google App Engine :** PaaS for application hosting.
   - **BigQuery :** SaaS large scale database analytics.

Syllabus Topic : Mobile Cloud Computing

5.5 Mobile Cloud Computing

Q. Explain Mobile Cloud Computing.

With the development of cloud computing support to desktop computing and large growing market of handheld devices tech giants like Apple, Google Microsoft are taking the next step providing cloud computing services at users fingertips that is the smartphones.

- Mobile Cloud Computing (MCC) provides the functionality of a standard cloud in the small handheld devices like smartphones.
- The availability of internet to the mobile devices provides the functionality of IoT to the smartphones making it a "thing".
- Machine to Machine communication model and sensor capabilities are two main characteristics of mobile cloud computing.
- Mobile cloud collects data in the surrounding and location based data and uses the cloud services for storage, processing and display.
- Thus mobile computing, cloud computing and IoT are all part of a big picture providing IoT support to the user's smartphone via cloud services.

- Many apple and android apps and widgets are providing such cloud support.
- Apple's iCloud services allows user to store data like images, music files, documents etc. on a remote server which can be downloaded by same user's devices like iPhones, iPads, iPods and personal computers running Mac or Windows based OS.
- Windows Live Mesh allows files and folders synchronization on two machines running Windows and Mac OS computers. Windows Live Mesh is a free to use, Internet based service.

Syllabus Topic : The Cloud of Things Architecture

5.6 The Cloud of Things Architecture

Q. Explain Cloud of Things Architecture.

IoT and Cloud Computing have many comparable characteristics. Cloud Computing provides services via three layer : IaaS, PaaS and SaaS while IoT provides services via three layer: Devices, Networks and Applications. Cloud Computing services are categorized as public cloud, private cloud, hybrid cloud etc. and IoT also have Intranet of Things, Extranet of Things, Internet of Things etc. Cloud of Things tries to represent all the functionalities and working backbone of the IoT system.

- Cloud of Things refers to providing IoT based services over the cloud with the help of IaaS, PaaS and SaaS.
- To define it shortly, cloud of things is the integration of Cloud Computing and IoT.
- The Cloud of Things architecture is shown in the Fig. 5.6.1.

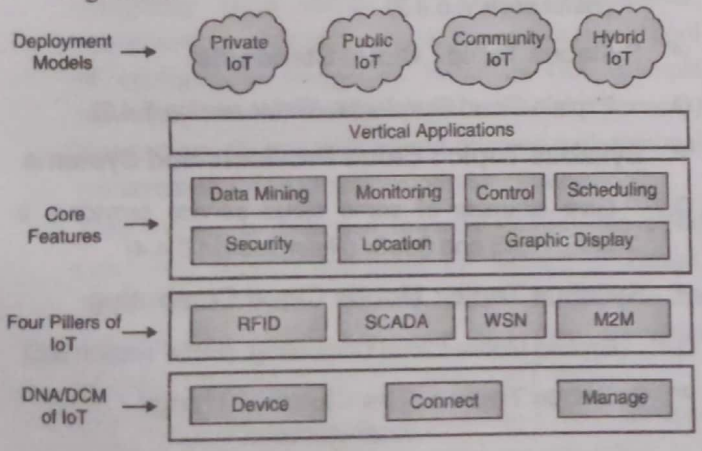

Fig. 5.6.1 : Cloud of Things Architecture

- The base layer defines the DNA of IoT which forms the foundation of the Cloud of Things.

Device

Devices are the smart things like WSN sensors, SCADA Actuators M2M terminals etc. and non-smart things like tables, chairs, and animals etc. which are given a sense of intelligence using RFIDs. These are the end nodes that actually gather data.

Connect

Connect phase provides the communication mechanisms for Cloud of things. This includes wired and wireless communication spread over short and long distance.

Manage

- This phase manages the application middleware and cloud computing that forms the backend of the Cloud of Things.
- The DNA of IoT is supported by the four pillars of IoT i.e. M2M, RFID, SCADA and WSN. All these techniques are covered in full in Chapter 3.

M2M

M2M is the area of interest for telecom companies. IP based identification for the mobile devices give M2M a huge strength in IoT based system.

RFID

RFID gives dumb things identity in IoT based system making them traceable. RFID provides much necessary identity management to devices which are not smart like table, animals, etc.

WSN

WSN provide the much needed network for the sensors and actuators deployed in the IoT system. Some WSN systems can also work as a full-fledged service providing entity sometimes.

SCADA

- SCADA provides IT controlled smart systems for industrial automation, smart grids, buildings etc.
- The layer above it specifies the core functionalities required for the Cloud of Things for Ubiquitous Connectivity.

Data Mining

All the data gathered via sensors needs to be refined in to useful information which is done using Data Mining Techniques and also provide decision support system.

Monitoring and Controlling

Monitoring and controlling are the fundamental functionalities of Cloud of Things. These features provide monitoring over the environment and controlling to some extent.

Scheduling and Dispatching

Time and event based job handling and scheduling of such jobs happening in the background of IoT based system.

- Security, Alter generation, Location based services, Maintenance and Graphic Display are some other core services of Cloud of Things.
- Using all these layers Vertical applications can be created and are provided to the end users using Public Cloud Services i.e. Public IoT, Private Cloud Services i.e. Private IoT, Hybrid Cloud Services i.e. Hybrid IoT, etc.

Private IoT

IoT system operated exclusively for a single organization is called as Private IoT. It is managed by the organization or third party manager.

Public IoT

A public IoT system provides its services to the general public and other industries and is managed by a single organization or group of organizations. Public IoT service provider sells these services for a some cost.

Community IoT

Community IoT is an integrated system shared by multiple organizations supporting common community concerns like social missions, security requirements, social policies etc. It is usually managed by multiple organizations keeping social welfare in focus.

Hybrid IoT

This IoT system is integration of two or more of above IoT systems. Private, community or public IoT are bound together with technologies that handles the application portability in between these various IoT services.

- All these applications and services are powered by the web technologies (HTML, CSS etc) and cloud technologies (IaaS, PaaS and SaaS) that gives the sense of Cloud of Things to IoT.

5.7 Exam Pack (Review Questions)

Syllabus Topic : Web of Things versus Internet of Things

Q. What is Web of Things (WoT)? *(Refer section 5.1)*

Syllabus Topic : Two Pillars of the Web

Q. What are the two pillars of the Web? Explain in brief. *(Refer section 5.2)*

Syllabus Topic : Platform Middleware for WoT

Q. Explain M2M Middleware Standards in brief. *(Refer section 5.3.1)*

Q. Explain WSN Middleware Standards in brief. *(Refer section 5.3.1)*

Q. Explain SCADA Middleware Standards in brief. *(Refer section 5.3.1)*

Q. Explain RFID Middleware Standards in brief. *(Refer section 5.3.1)*

Syllabus Topic : Unified Multitier WoT Architecture

Q. Explain unified multitier WoT Architecture in details. *(Refer section 5.3.2)*

Syllabus Topic : WoT Portals and Business Intelligence

Q. Give examples of some WoT Portals. *(Refer section 5.3.3)*

Q. Explain Business Intelligence associated with WoT. *(Refer section 5.3.3)*

Syllabus Topic : Cloud of Things - Grid/SOA and Cloud Computing

Q. Explain Cloud Computing as integration to Grid computing. *(Refer section 5.4.1)*

Syllabus Topic : Cloud Middleware

Q. Explain Cloud Middleware Architecture. *(Refer section 5.4.2)*

Syllabus Topic : Cloud Standards

Q. Explain Cloud Standards. *(Refer section 5.4.3)*

Syllabus Topic : Cloud Providers and Systems

Q. Give example of some cloud service providers in IaaS, PaaS and SaaS. *(Refer section 5.4.4)*

Syllabus Topic : Mobile Cloud Computing

Q. Explain Mobile Cloud Computing. *(Refer section 5.5)*

Syllabus Topic : The Cloud of Things Architecture

Q. Explain Cloud of Things Architecture. *(Refer section 5.6)*

UNIT VI

CHAPTER 6

IoT Physical Servers, Cloud Offerings and IoT Case Studies

Syllabus Topics

Introduction to Cloud Storage Models, Communication API, WAMP : AutoBahn for IoT, Xively Cloud for IoT, Python Web Application Framework : Djanjo, Amazon Web Services for IoT, SkyNet IoT Messaging Platform. Case Studies : Home Intrusion Detection, Weather Monitoring System, Air Pollution Monitoring, Smart Irrigation.

Syllabus Topic : Introduction to Cloud Storage Models

6.1 Introduction to Cloud Storage Models

Q. Give a brief overview of cloud computing and IoT cloud storage models.

- Cloud computing has revolutionized the way data and services are exchanged over the Internet. Applications, data and services are delivered over the network using the paradigm called cloud computing. NIST has specified the definition of cloud computing as " Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (for example networks, servers, storage, applications and services) that can rapidly provisioned and released with minimal management effort or service provider interaction".
- This chapter is dedicated towards learning the cloud computing benefits and usage for Internet of Things. There are few very popular and useful models are available for cloud computing and they are quite effective and customized for modern IoT needs.
- One of the best example is Amazon's web services which has dedicated service model for supporting IoT applications. Another model is available as Platform as a Service called Xively cloud for IoT.
- There are frameworks available to support application development for such platforms. One popular framework is Python Web Application Framework (Django).
- In this chapter, we are going to discuss all these models in detail in subsequent section.
- These models are supported by effective communication protocols and communication APIs. An overview of communication APIs is presented in next section.

Syllabus Topic : Communication API

6.2 Communication API

Q. Give a brief overview of communication API.

- The cloud models for IoT applications are heavily relied on communication APIs.
- These communication APIs can either be separately implemented from third party organization or natively by the cloud service providers.
- Communication APIs facilitate data transfer, control information transfer from application to cloud, one service to another.
- Few of the communication APIs exists in the form of communication protocol and they support publisher and subscriber models. They often used remote procedure call (RPC) messaging pattern. For example WAMP (Web Application Messaging Protocol) A sub protocol of web socket based on Publish-subscribe and RPC.
- These communication APIs supports the service invocation, data transfer, control information transfer, publish and subscription information for different services.

- The communication APIs specify the service invocation structure, URL patterns, response formats etc.
- There is not a single standard communication API available for cloud storage and IoT applications. However RESTful APIs are popular and effective for implementing communication in cloud models.
- We will study in subsequent section about how to design a RESTful web API. Django web framework will be used to implement the communication API.

Syllabus Topic : WAMP - AutoBahn for IoT

6.3 WAMP : AutoBahn for IoT

Q. Describe the IoT messaging mechanism called WAMP (AutoBahn for IoT).

- WAMP stands for Web Application Messaging Protocol. It is a communication protocol which can be used in cloud storage models for IoT. It can also be used other IoT service communication.
- WAMP is a sub protocol of Web socket. Web socket is a publish-subscribe based protocol well suited for IoT applications. It uses the remote procedure call messaging pattern (RPC).
- IoT systems require the application to be distributed on multiple nodes. WAMP protocol enables such distributed application architecture.
- There are several key concepts that must be understood while discussing WAMP. Following are the key terminologies along with description.

Transport

The channel between the two communication nodes is called the transport. Web Socket is the transport between two nodes or peers. WAMP does not necessarily need Web Socket as its transport. Other protocol channel can also serve the transport for WAMP which are bidirectional and message based.

Session

Once a transport is formed between the two nodes or peers, the session is required for communication. Session is built upon transport and runs over it. Informally, session is nothing but the conversation between two nodes or peers.

Clients

At any instance of conversation, the communicating nodes or peers called the clients. Clients can have many roles, but for this model of communication client can be publisher or subscriber. The roles are explained below:

1. **Publisher**
   Broker managed and stores the topic and the client as publisher publishes the event to the topic. The event may include many other information that is sent as payload.
2. **Subscriber**
   - The topic published by the publisher is subscriber by the clients called the subscriber. They also receives the pre-specified information for which the subscriber has been promised. The information is received as payload.
   - Additionally in remote procedure call model clients can take the role of caller and callee. Caller calls the remote procedure and passes the necessary information to execute the remote procedure. Callee executes the remote procedure.

Router

Router are special nodes or peers in the cloud communication model. The router play role of broker in publish-subscribe model while in RPC it plays the role of dealer. The two concepts are mentioned below.

Broker

When the router acts as broker, the primary responsibility of it is to route the published information to all valid subscribers. The valid subscribers are all those subscribers which are subscribe to the topic.

Dealer

In the RPC model, the broker acts like a dealer. So the responsibility of dealer is to channelize the calls from caller to callee. Also to route the response from caller to callee.

Application Code

- The client side logic to perform the required operation is encapsulated in application code.
- The application code runs nodes or peers. So the publisher, subscriber, brokers are eligible to run application code.
- A simple illustration of communication model on cloud for IoT can be presented by arrangement of client, transport, session and router. One such session between the client and router is shown in Fig. 6.3.1.
- The Fig. 6.3.1 shows the different roles of the client and different roles of the routers. The session is established over the transport.
- The session can be between publisher and subscriber routed via the broker or the session can be established between the caller and callee routed via the dealer.

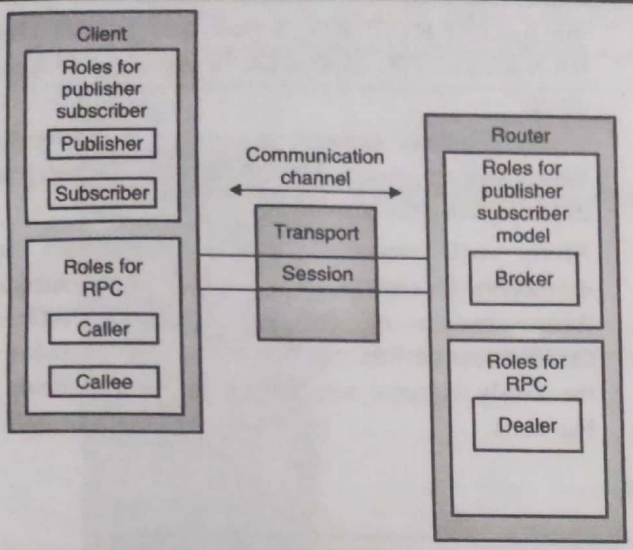

Fig. 6.3.1 : WAMP communication model : A session between Router and Client

- The interaction of nodes or peers implemented by WAMP protocol is shown in Fig. 6.3.2. The transport used in WAMP is the WebSocket.

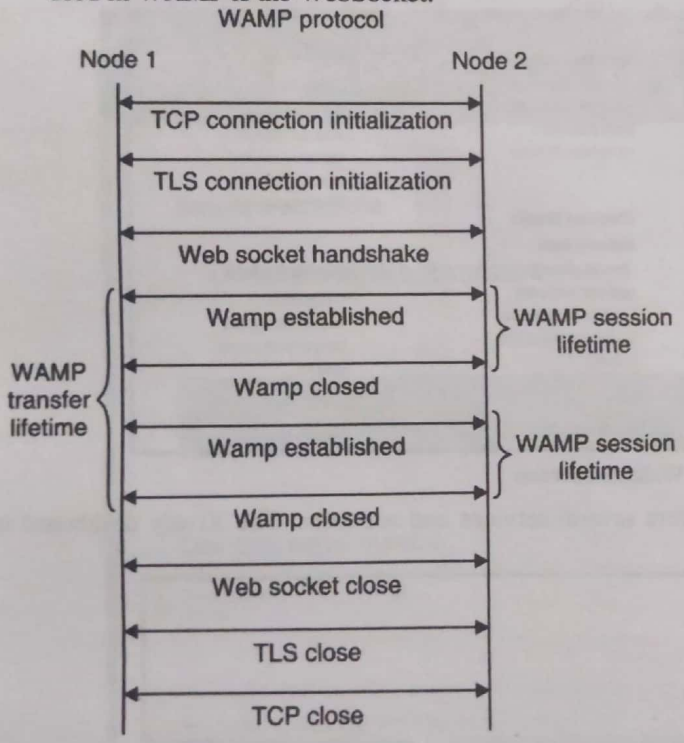

Fig. 6.3.2 : WAMP Protocol

- WAMP sessions exist for the entire lifetime of the Web Socket transport. The WAMP protocol between the two peers is shown in Fig. 6.3.2.

6.3.1 The AutoBahn Framework

Q. Describe the AutoBahn framework in brief.

- The open source implementation of the WAMP protocol and Web Socket is AutoBahn framework.
- In AutoBahn framework, the WAMP application component is deployed on client. The client as publisher publishes the events and messages to router.
- The router runs on server and as discussed earlier plays the role of broker. It routs the published messages to subscribers.
- The AutoBahn WAMP deployment is same as shown in Fig. 6.3.1. It shows the communication between different components of AutoBahn framework arrangement.
- The communication in the arrangement shown in Fig. 6.3.1 is between publisher, subscriber and router.
- This communication is used to take place over WAMP WebSocket session.

6.3.2 The Implementation and Installation Steps of the AutoBahn

Q. Write the AutoBahn installation and setup steps.

- The AutoBahn can be implemented for WAMP publisher and subscriber model. The commands to install the Autobahn-python are

(i) sudo apt-get install python-twisted python-dev
(ii) sudo apt-get install python-pip
(iii) sudo pip install-upgrade twisted
(iv) sudo pip install-upgrade autobahn

- The above commands will install and setup the AutoBahn on your Linux system.
- Once the AutoBahn is installed, Autobahn-Python framework implementation can be cloned from any available resource pool like Git hub. Following is the command for clone the AutoBahnPython from Git.

```
git clone https://github.com/tavendo/AutobahnPython.git
```

- Once the AutoBahn installation and framework setup is done, the publisher, subscriber and router on server can be implemented as per applications requirement.
- Interested readers can get many official and unofficial documents related to autobahn WAMP publisher subscriber and broker implementation. One such resource link is provided below.

http://autobahn.readthedocs.io/en/latest/

Syllabus Topic : Xively Cloud for IoT

6.4 Xively Cloud for IoT

- Many Internet of Things application are developed and deployed on Xively cloud. Xively cloud is a commercial platform available as PaaS (Platform as a service).

- IoT application developer writes the frontend application for IoT as per need and the backend services are implemented and managed by Xively cloud.
- The backend infrastructure provided by Xively is mainly the data collection, management and distribution infrastructure.
- Xively provides many more backend infrastructure facilities like real time message management, data service for time series archiving, directory services in the form of searchable directory for devices and services etc.
- There are multiple language and platform support from Xively. Xively supports and implements in its library the standard HTTP API, Sockets and MQTT. Using these standard API devices can be connected to Xively cloud.
- Xively popularly exposes its services in the form of Xively Python library. Multiple aspects of Xively and its library is explored in this section.
- Xively web interface allows programmers and developers to register in it for developers account. After creation of developers account, different development devices can be created. The snapshot of the Xively welcome screen for developers is shown in Fig. 6.4.1.

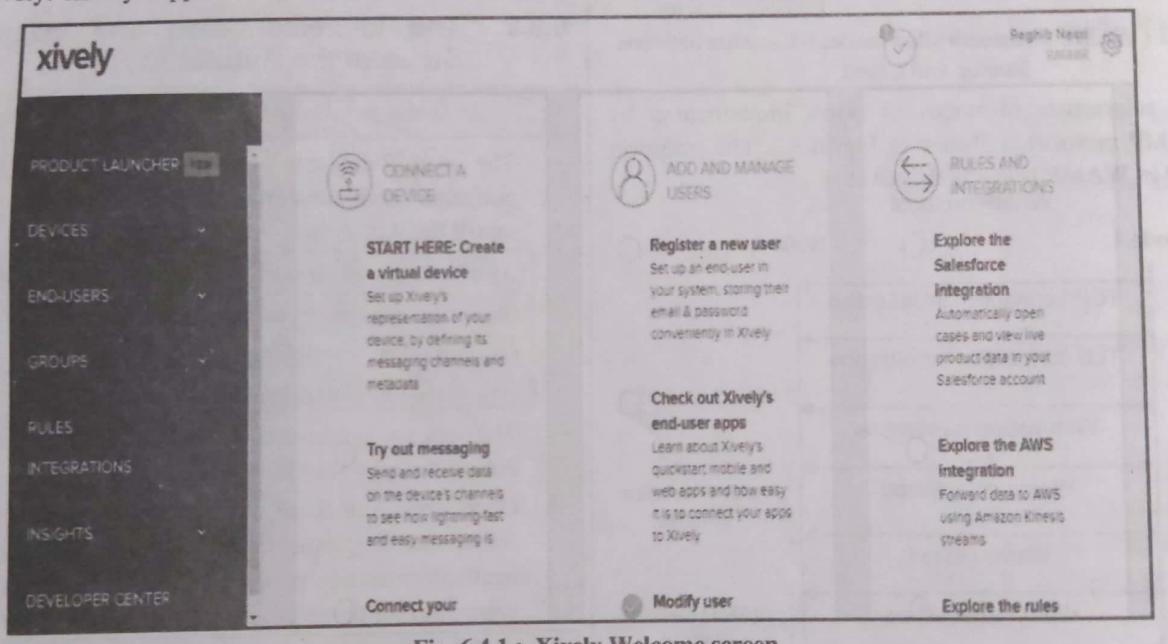

Fig. 6.4.1 : Xively Welcome screen

- Xively web interface offers a Xively dashboard which offers several services and activities. The Xively dashboard is shown in Fig. 6.4.2.

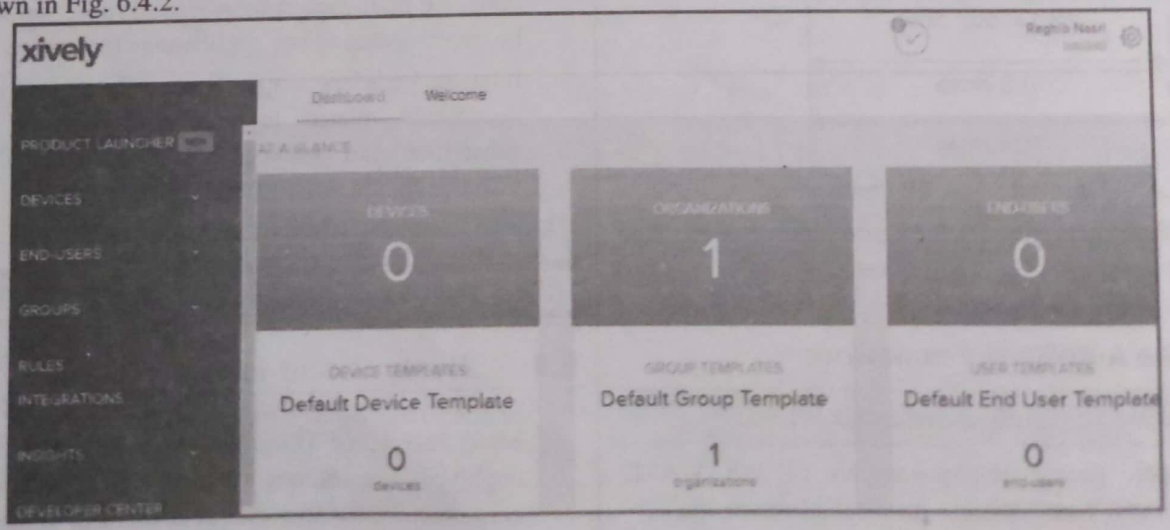

Fig. 6.4.2 : Xively Dashboard

- The standard way Xively offers to create a device is create the device template first. The following screenshot (Fig. 6.4.3) provides the glimpse of device template creation.

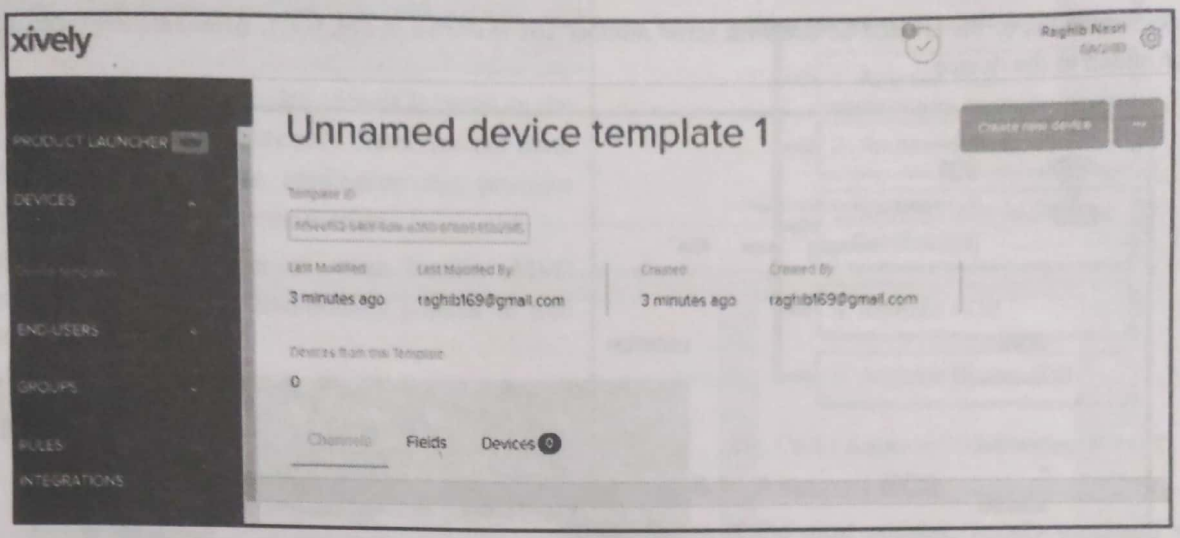

Fig. 6.4.3 : Xively Dashboard : Creating a new device template

- After creation of device template, different development devices can be created. The snapshot of the Xively dashboard for creating new device is shown in Fig. 6.4.4.

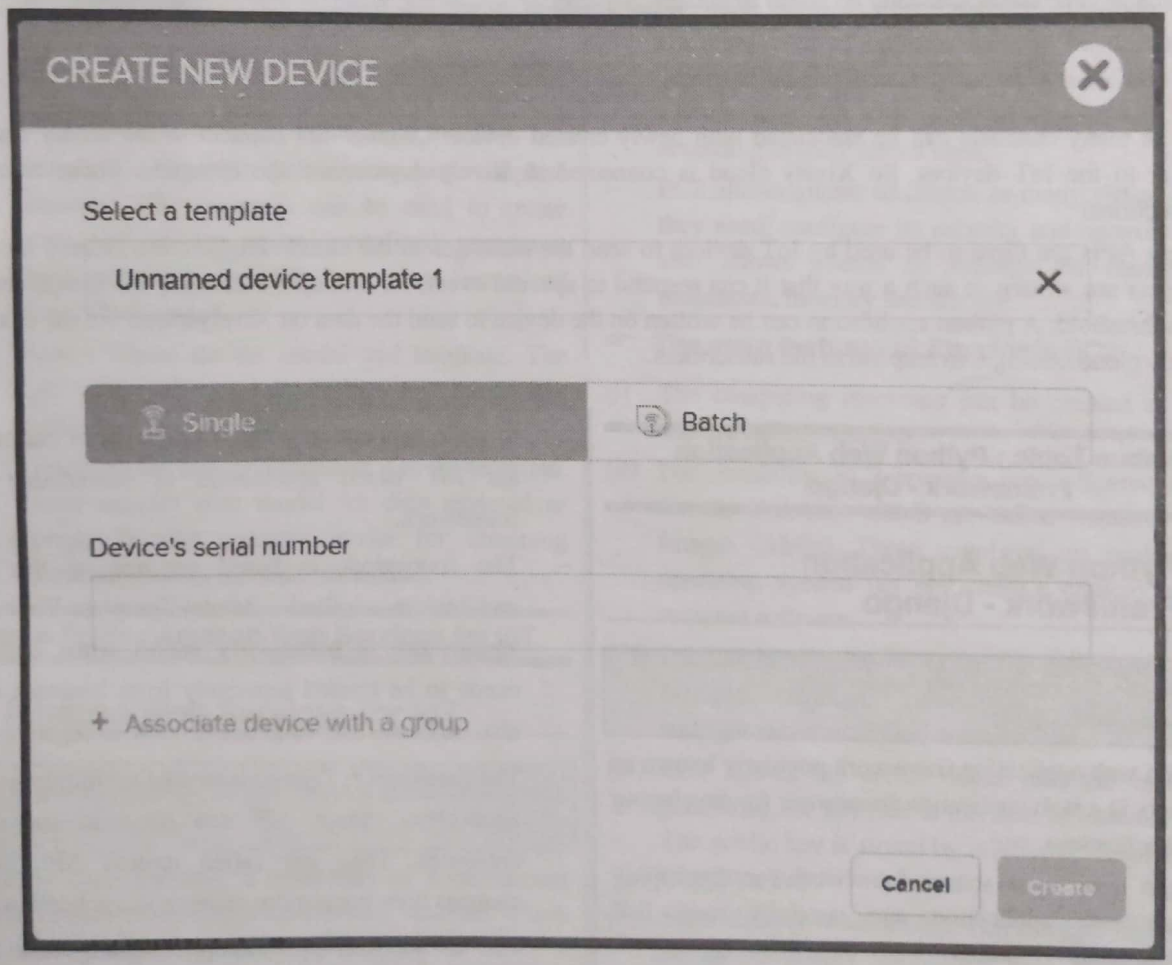

Fig. 6.4.4 : Xively Dashboard: Creating a new device

- After creation of device, Xively generates the Feed-ID and API key which are later used to connect with the device. Feed-ID for each device is unique within the application.
- Feed-ID specifies the data stream associated with device. It also associates meta-data with it.

- The permissions on the device is implemented with the help of API keys. The default associated permission are read, update, create, delete.
- The device details of the created device with serial number 101 is shown in Fig. 6.4.5. After creation more information can be added to the device.

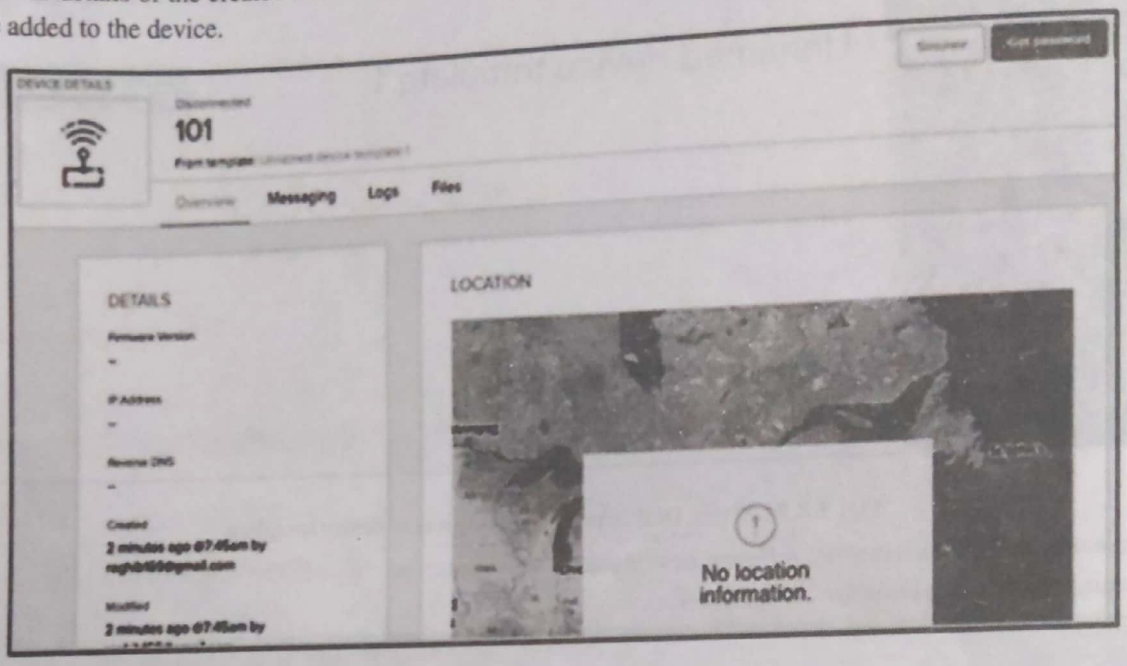

Fig. 6.4.5 : Xively Dashboard : Created device details

- One or many channels can be associated with newly created device. Channel can connect to the broker and via the broker to the IoT devices. So Xively cloud is connected to Xively devices via the channels. These channels are bidirectional.
- Xively APIs are there to be used by IoT devices to send the messages to the cloud. Triggers are created for channel. Triggers are written in such a way that it can respond to specific event, for example exceeding the temperature value from threshold. A python application can be written on the device to send the data on Xively cloud via the channel and Xively cloud conFig.d to respond to the subscribers.

Syllabus Topic : Python Web Application Framework - Django

6.5 Python Web Application Framework - Django

Q. Explain the Python Web Application Framework-Django.

- Python web application framework popularly known as Django, is a web application framework for developing IoT applications.
- Django is an open source framework for developing python web application and specially used IoT application development.
- Using this framework developers can develop dynamic python based web sites.
- Django encapsulates all the patterns and best practices for effective websites. Developers may find it very easy to write application portals. The level of customization for IoT based application is remarkable in the framework.
- The framework is based on one of the popular architecture called Model-Template-View. This architecture is particularly useful when data model needs to be treated separately from business logic. It also treats the user interface or view as separate unit.
- The databases is Django is treated uniformly across the application. Same API can be used for different databases. They are called unified API. Database changes does not require changes in application code.
- The entire web application designing task can be deeply coupled with Python ecosystem with the help Django framework. It well suits for IoT applications. The architecture of Django is discussed briefly in next section.

6.5.1 Django Architecture

Q. Explain the model-template-view architecture of Django.

- As stated earlier, the Django framework is based on the model-template-view architecture. These are the three logical components of the application that provides high level of abstraction for each other.
- This is somewhat similar to more familiar MVC (Model View Controller) traditionally popular in web application development.
- The responsibilities of these components of the model-template-view (MTV) in Django is stated below:

(1) **Model :** The interaction of the application with databases is handled through model. Model regulates all the flow of data. Data can be stored in relational or non-relational database, XML files etc. Model defines rules related to data storage and retrieval.

It also acts as the definition of some of the stored data. In Django, dedicated python class encapsulates the required logic of model. It defines the variables and methods for the data.

(2) **Template :** Django provides special template language. This language can be used to create forms and classes using predefined templates. These templates are further to create forms and other HTML pages.

(3) **View :** Views tie the model and template. The logic written in views determines the type of data required by the application.

It retrieves the data and transfers it to the template. Views interact with model for data retrieval or storage. It also consults model for obtaining information about the data (metadata).

Syllabus Topic : Amazon Web Services for IoT

6.6 Amazon Web Services for IoT

Q. Explain the different cloud based services offered by Amazon for IoT.

- Amazon Web Services, a subsidiary of Amazon.com offers several cloud based on demand platform which can be used for IoT application development.
- Few of the most noticeable and talked about services are EC2 (Elastic Compute Cloud) S3 (Simple Storage Services). We will discussed most popular and useful service in this section.

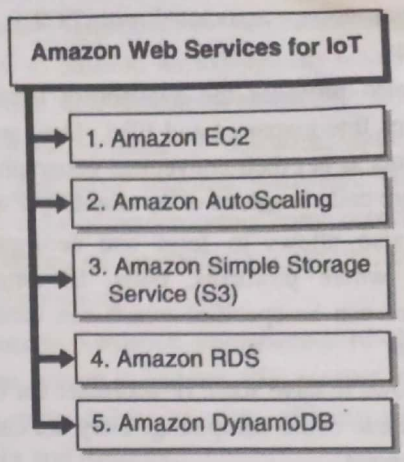

Fig. C6.1 : Amazon Web Services for IoT

➜ 6.6.1 Amazon EC2

- One of the most popular, widely used and wonderful service of Amazon is EC2 (Elastic Compute Cloud). It provides scalable computing capacity to its users. Clients may scale up or scale down the computing resources based on their changing requirements.
- Different kind of applications can be developed and deployed on EC2 faster than any other cloud services. Clients also do not need to invest on hardware according to their varied need.
- EC2 allows clients to launch as many virtual server as they need, configure its security and networking. EC2 also allows clients to manage its resources like processors, memory and storage.

☞ The main features of Amazon's EC2

(i) The computing machines can be created as required and are called instances.

(ii) For simplicity, EC2 provides preconfigured templates for the instances which are called Amazon Machine Images (AMIs). These templates are packaged with operating system (as per client's choice) and other required software.

(iii) Possible configurations of EC2 is determined by CPU, memory, storage, networking capability. These configurations are called *instance types*.

(iv) EC2 allows the secure login into the machine by generating the key pair at the time of instance creation. The public key is stored by AWS and clients maintains its private key.

(v) Each instance is associated with storage volumes where all the temporary application data are stored. These storage are of the EC2 are called **instance store volumes**. Once the instance get deleted or stopped, these volumes are also get deleted.

(vi) All the persistent data for the EC2 is stored using Amazon's Elastic Block Storage (EBS) and the

persistent storage associated with EC2 is called **EBS volumes**.

(vii) AWS also provides the availability location for the resources line instances and EBS. They are chosen by the clients as per their convenient geographic locations. These are called **regions** or **Availability Zones**.

(viii) EC2 also allows to form and configure **security groups** where protocols, ports, IP ranges for the machines can be specifies which can access the EC2 instance.

(ix) EC2 allows to have static IP addresses for dynamically configurable could computing. They are called **Elastic IP addresses.**

(x) EC2 also allows to create tags and associate it with EC2 instances. These tags the metadata for the created instances.

(xi) AWS allows **Virtual Private Clouds** to be created for Amazon EC2 instances. Using these virtual private clouds, client can connect EC2 instances with their own network.

- EC2 allows GUI for configuring and launching EC2 instances. However you can write scripts to create and launch EC2 instance. Interested readers can refer many online openly available python scripts to launch Amazon EC2 instance.

➜ 6.6.2 Amazon AutoScaling

- Amazon autoscaling allows very flexible upgrade and downgrade options to the clients of the EC2.
- Clients of the EC2 using autoscaling feature can dynamically scale the capacity of EC2 instance up or down. The scaling criteria is also determined by the clients of EC2.
- Autoscaling ensures the applications running on EC2 instances will remain available to its users in any conditions.
- Also the number of EC2 instances can be increased when the demand is on the peak. Different resource can be configured to be auto scaled.
- Autoscaling option is proved quite useful for IoT application where on specific time bulk of data is generated and need to be processed.
- Clients can write simple scripts to define their autoscaling criteria. Interested users can find variety of such python scripts available online for defining their application specific autoscaling conditions.

➜ 6.6.3 Amazon Simple Storage Service(S3)

- Second most popular service of Amazon AWS after EC2 is S3 (Amazon Simple Storage Service).
- This is fast, efficient cloud based storage service. This is the entire infrastructure for storage offered to its user for fast storage and retrieval of large amount of data.
- Data can be audio and video too. It supports streaming of data directly in the browser's media player or it can be downloaded. Once data is stored, S3 offers handy URLs to access the stored data.
- The Amazon S3 is affordable, scalable, fast, and quite easy to be used. It can easily be interfaces with other application storage and retrieval.
- S3 can be quite useful for fast storage of IoT application data. This can be generated data from the sensors.
- A simple web service interface is offered to S3 clients. Using this interface, client can easily interact with offered services.
- Most importantly, Amazon offers S3's application programming interface for the developers to integrate the S3 services with their web application. IoT developers can also use the same API for their remote IoT applications.
- Though the simple GUI web interface is available to create S3 buckets, users can write their own scripts to create S3 bucket and upload files and raw data into it. Interested readers can go online to refer such openly available python scripts.

➜ 6.6.4 Amazon RDS

- Amazon offers the convenient and efficient relational database solution called the RDS (Amazon Relational Database Solution).
- RDS has many features and most remarkable is easy to setup, scale and interface with other application.
- RDS is available on cloud, so accessible from anywhere from multiple application. Data sharing among the distributed IoT application can easily be achieved using the RDS.
- Amazon RDS is cost effective solution for the database. It is has resizable capacity and free the user from time consuming administrative task.
- High performance, fast retrieval, easy access, high security are few other key features of the RDS.
- Amazon RDS does not restrict user from using any specific kind of database. Client can opt from six available database engines : Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle, and Microsoft SQL Server.

➜ 6.6.5 Amazon DynamoDB

- Amazon offers the NoSQL alternative the traditional relational database solution as DynamoDB.

- This is fully managed, scalable and high performance database solution.
- This is quite useful as IoT applications data solution. DynamoDB allows IoT application to transfer bulk of unorganized data for storage and faster retrieval.
- DynamoDB can manage high number of request for data storage and retrieval. This feature is quite useful when distributed IoT application access the database simultaneously from different locations.

Syllabus Topic : SkyNet IoT Messaging Platform

6.7 SkyNet IoT Messaging Platform

Q. Explain the SkyNet IoT messaging platform.

- One of the popular messaging platform for IoT application is SkyNet messaging platform. It is an instant messaging platform which supports multiple API.
- SkyNet support the APIs like HTTP, REST and the Web Socket. The platform enables the devices to send quick messages reliably.
- SkyNet messaging platform manages the entire messaging active between the nodes. Nodes or devices can register itself to SkyNet messaging platform to send or receive the messages. Nodes can even avail more messaging services from SkyNet.
- The candidate for sending the messages via the SkyNet can virtually be any IoT component. It can be any device, sensor, server, appliance, drone etc.
- Device are assign a unique id called the UUID upon registration of SkyNet messaging platform. The device or node is also provided with secret access token.
- Device can also subscribe to other devices to receive the messages of interest. The messaging framework characteristic which allows the messages to be published and subscribed makes the SkyNet messaging framework well suited for IoT application.
- Instant messaging over RESTful API also makes the messaging platform useful for IoT application. In most IoT messaging scenario, messages need to be reliably and securely delivered to the registered devices. SkyNet helps in delivering small messages to the instantly to the subscribed devices.

Syllabus Topic : Case Study - Home Intrusion Detection

6.8 Case Study - Home Intrusion Detection

Q. Provide an IoT solution for home intrusion detection system.

- The primary objective of home intrusion detection system is to develop an IoT system which can detect the intrusion with the help of sensors.
- The system includes a software system, sensor, network and other required hardware. There can be multiple ways of implementing home detection system. It will depend on feasibility, availability, environment, financial resources.
- The primary hardware components of the intrusion detection system is camera, PIR sensors, door sensors and an embedded board with sufficient resources like processor and memory.
- These sensors need to be properly physically placed so that it can detect intrusion effectively. The sensors and cameras are interfaced with the embedded board which is connected to the Internet. User interfacing device like mobile app or alarming system should be connected to the network.
- Alerts related to the intrusion can be sent to the user's app (app notification), SMS, or email. Most preferable way can be alarming system within the home premises or nearest security station.

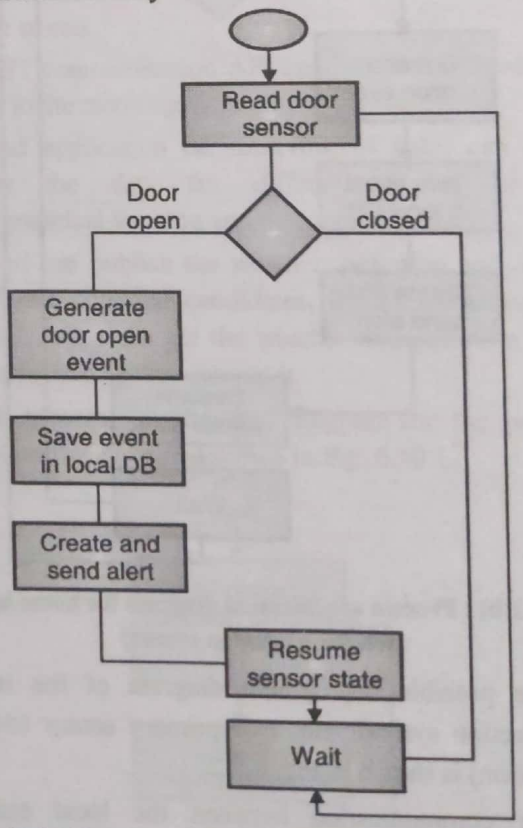

Fig. 6.8.1(a) : Process specification diagram for home intrusion detection (door sensor)

- More additional features can be to the basic intrusion detection system like capturing the image and sending it to the email attachment. Now a day's even instant

messaging app can receive images from other automated systems. This can be quite effective solution for notifying the intrusion detection.

- On possible process specification diagram of the home intrusion detection system is shown in Fig. 6.8.1(a).
- This is for the door sensor which will simply detect the door is open generate an alert by creating an event called "door opened" event.
- Let us have a look on the process flow diagram.
- Similarly for motion detection, motion detection sensor is read by the application, event is generate and alert is sent to the desired format to the required registered devices.
- The process specification diagram is shown in Fig. 6.8.1 (b).

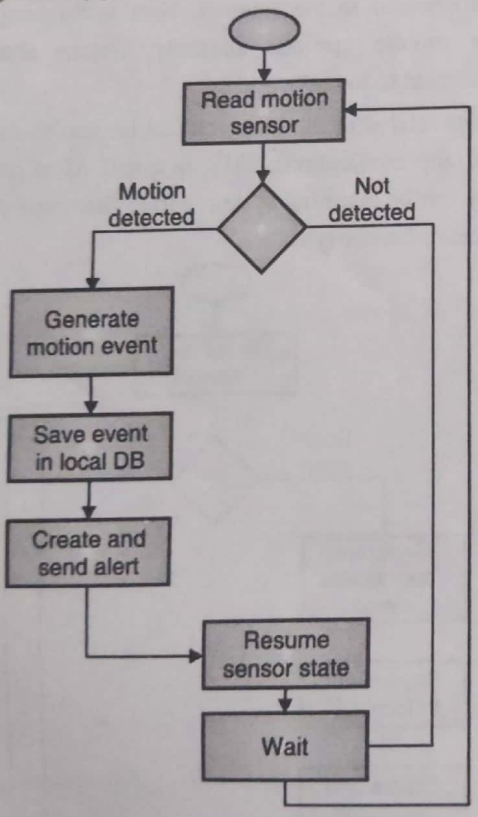

Fig. 6.8.1(b) : Process specification diagram for home intrusion detection (motion sensor)

- The possible deployment diagram of the intrusion detection system with two primary sensor (door and motion) is shown in Fig. 6.8.2.
- The communication between the local embedded system and cloud application is assumed to be a REST communication.
- However different implementation of intrusion detection system may opt for different communication mechanism and protocol.

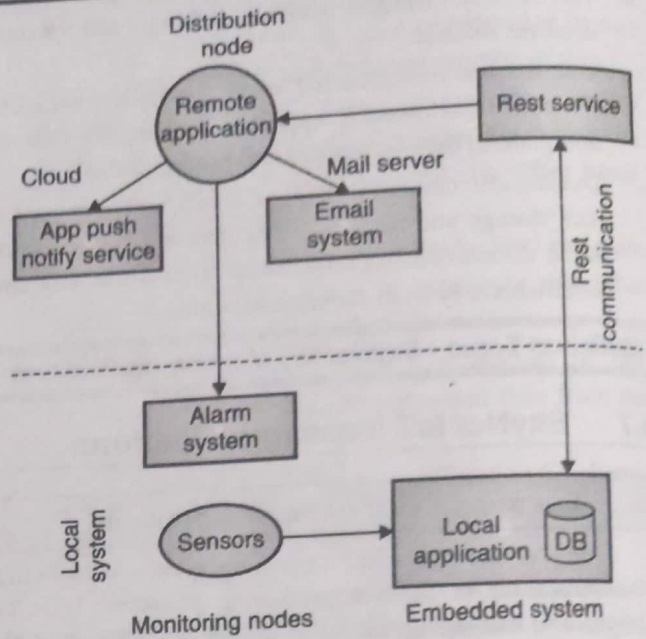

Fig. 6.8.2 : Deployment diagram of the home intrusion detection system

Syllabus Topic : Case Study - Weather Monitoring System

6.9 Case Study - Weather Monitoring System

Q. Provide an IoT solution for weather monitoring system.

- The primary objective of weather monitoring system is to develop an IoT system which can collect data from different environmental and weather conditions.
- This system may include many kind of sensors for monitoring different weather conditions like temperature sensor, humidity sensor, pressure sensor, moisture sensor, light sensor etc.
- These sensors acts as multiple end nodes of data collection for the IoT system.
- The data is collected by the local embedded system which is connected with the sensor directly or indirectly. The local system can apply some pre-processing on the collected data. It can apply certain rules under which data need be validated and qualified.
- REST communication API can be used to transfer the data to the cloud application.
- Cloud application on receiving of data, can further filter the data for different application area.
- Cloud can publish the weather data for different potential candidates. Registered subscribers, clients, apps can get the weather information as they require it.

- The process specification diagram for the weather monitoring system is shown in Fig. 6.9.1.

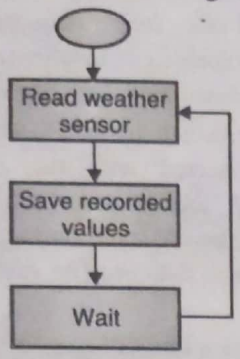

Fig. 6.9.1 : Process specification diagram for Weather Monitoring System

- The possible deployment diagram of the of the weather monitoring system is shown in Fig. 6.9.2.

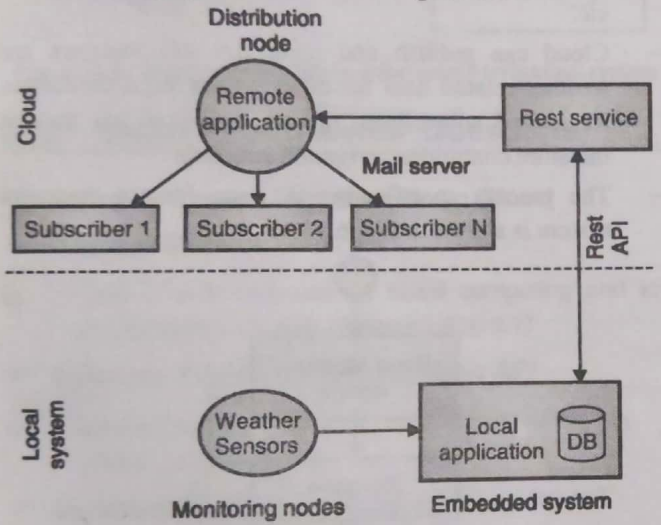

Fig. 6.9.2 : Process specification diagram for Weather Monitoring System

- The nodes are the sensors that collects weather conditions data and send it to the locally connected embedded system. Required preprocessing is applied and data is transferred to the cloud using REST API. Cloud service distributed the weather information to registered subscribers.

Syllabus Topic : Case Study - Air Pollution Monitoring

6.10 Case Study - Air Pollution Monitoring

Q. Provide an IoT solution for air pollution monitoring.

- The primary objective of air pollution monitoring system is to develop an IoT system which can collect data from different environmental condition and provide information about the air pollution.
- The system design and steps of implementation is quite similar weather monitoring system. The applications and subscribers of IoT may differ.
- This system is intended to detect harmful gases emitted by factories, vehicles and other sources.
- This system may include many kind of sensors for monitoring different environmental conditions and gases. The sensors in such IoT systems mainly belong to the category of gaseous and meteorological sensors.
- These sensors acts as multiple end nodes of data collection for the IoT system.
- The system design and deployment design of the pollution monitoring sensors are almost similar to weather monitoring system (discussed in previous case study). Only few additional gaseous and meteorological sensors added to the system.
- The IoT system consists of multiple nodes located in different pollution probable areas, conditions and locations. The sensors can be CO and NO_2 sensors.
- The data is collected by the local embedded system which is connected with the sensor directly or indirectly. The local system can apply some pre-processing on the collected data. It can apply certain rules under which data need be validated and qualified to be stored.
- REST communication API can be used to transfer the data to the cloud application.
- Cloud application on receiving of data, can further filter the data for different application area, geographical location and pollution sources.
- Cloud can publish the weather, pollution and data for different potential candidates. Registered subscribers, clients, apps can get the weather and pollution related information as they require it.
- The process specification diagram for the pollution monitoring system is shown in Fig. 6.10.1.

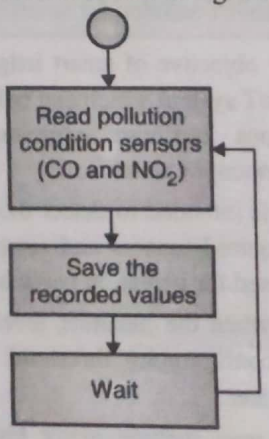

Fig. 6.10.1 : Process specification diagram for pollution monitoring system

- The possible deployment diagram of the of the pollution monitoring system is shown in Fig. 6.10.2.

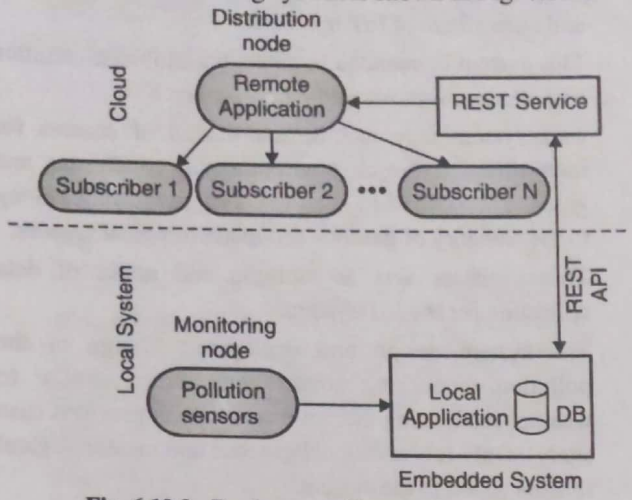

Fig. 6.10.2 : Deployment diagram for pollution monitoring system

- The end nodes in local the local system domain at different locations can be a mini computer like Beagle bone black or Raspberry Pi, and sensors could be MICS-2710 NO_2 and MICS-5525 CO sensors.
- The nodes are the sensors that collects pollution conditions data and send it to the locally connected embedded system.
- Required preprocessing is applied and data is transferred to the cloud using REST API. Cloud service distributed the pollution related information to registered subscribers.

Syllabus Topic : Case Study - Smart Irrigation System

6.11 Case Study - Smart Irrigation System

Q. Provide an IoT solution for smart irrigation system.

- The primary objective of smart irrigation system is to develop an IoT system which can collect data related to soil conditions and can automate the process of irrigation (amount of water flow).
- This system is intended to detect moisture level in soil using the required sensors and control the amount of water to be used for irrigation using automated valves.
- In general , when the moisture level goes below to a predefined (configurable) threshold value, it releases the flow of water
- This system may include many kind of sensors for monitoring soil and air moisture conditions.
- These sensors acts as multiple end nodes of data collection for the IoT system.
- Data required for smart irrigation from historical records and incidents can also be collected from cloud and can be analysed for scheduling of irrigation.
- The data is collected by the local embedded system which is connected with the sensor directly or indirectly. The local system can apply some pre-processing on the collected data. It can apply certain rules under which data need be validated and qualified to be stored.
- REST communication API can be used to transfer the data to the cloud application.
- Cloud application on receiving of data, can further filter the data for different application area like irrigation scheduling, water release and control systems etc.
- Cloud can publish and store the soil moisture and weather related data for different potential candidates. Registered subscribers, clients, apps can get the soil moisture conditions, irrigation schedule
- The process specification diagram for the irrigation system is shown in Fig. 6.11.1.

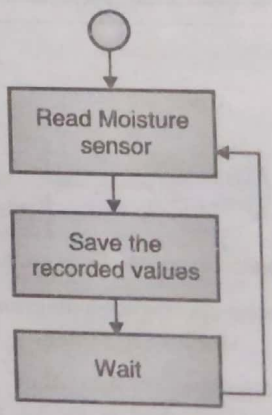

Fig. 6.11.1 : Process specification diagram for smart irrigation system

- The possible deployment diagram of the of the smart irrigation system is shown in Fig. 6.11.2.
- The end nodes in local system domain at different locations can be a mini computer like Beagle bone black or Raspberry Pi, and sensors could be soil moisture sensors.
- The nodes are the sensors that soil moisture data and send it to the locally connected embedded system. Required preprocessing is applied and data is transferred to the cloud using REST API.
- Cloud service distributed the moisture related information to registered subscribers. A water releasing

system checks the threshold value releases the water accordingly.

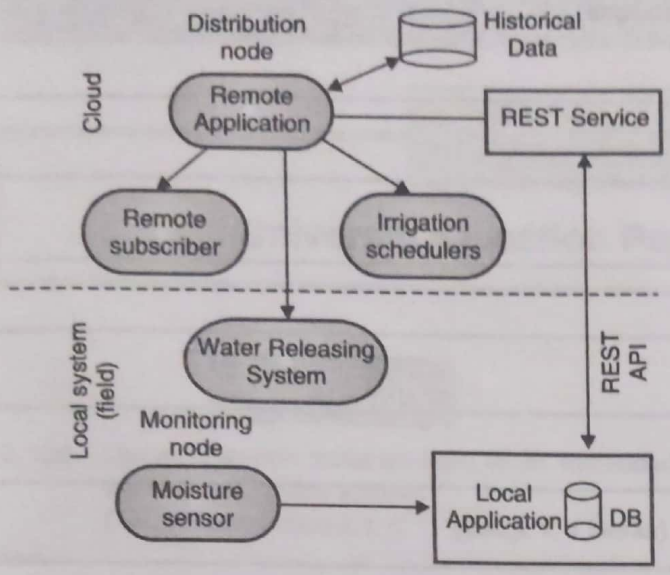

Fig. 6.11.2 : Deployment diagram for smart irrigation system

6.12 Exam Pack (Review Questions)

☞ **Syllabus Topic : Introduction to Cloud Storage Models**

Q. Give a brief overview of cloud computing and IoT cloud storage models. *(Refer section 6.1)*

☞ **Syllabus Topic : Communication API**

Q. Give a brief overview of communication API. *(Refer section 6.2)*

☞ **Syllabus Topic : WAMP - AutoBahn for IoT**

Q. Describe the IoT messaging mechanism called WAMP (AutoBahn for IoT). *(Refer section 6.3)*

Q. Describe the AutoBahn framework in brief. *(Refer section 6.3.1)*

Q. Write the AutoBahn installation and setup steps. *(Refer section 6.3.2)*

☞ **Syllabus Topic : Python Web Application Framework - Django**

Q. Explain the Python Web Application Framework-Django. *(Refer section 6.5)*

Q. Explain the model-template-view architecture of Django. *(Refer section 6.5.1)*

☞ **Syllabus Topic : Amazon Web Services for IoT**

Q. Explain the different cloud based services offered by Amazon for IoT. *(Refer section 6.6)*

☞ **Syllabus Topic : SkyNet IoT Messaging Platform**

Q. Explain the SkyNet IoT messaging platform. *(Refer section 6.7)*

☞ **Syllabus Topic : Case Study- Home Intrusion Detection**

Q. Provide an IoT solution for home intrusion detection system. *(Refer section 6.8)*

☞ **Syllabus Topic : Case Study - Weather Monitoring System**

Q. Provide an IoT solution for weather monitoring system. *(Refer section 6.9)*

☞ **Syllabus Topic : Case Study - Air Pollution Monitoring**

Q. Provide an IoT solution for air pollution monitoring. *(Refer section 6.10)*

☞ **Syllabus Topic : Case Study - Smart Irrigation**

Q. Provide an IoT solution for smart irrigation system. *(Refer section 6.11)*

❑❑❑