CMR
INSTITUTE OF
TECHNOLOGY

USN | | | | | | | | | | |

Internal Assesment Test –I, Answer key - January 2024									
Sub:	**Internet of Things**							**Code:**	**22MCA32**
Date:	**17/01/2024**	**Duration:**	90 mins	**Max Marks:**	50	**Sem:**	III	**Branch:**	MCA

	Answer Any 5 **QUESTIONS**	Marks	OBE	
			CO	RBT
1)	Explain IOT with its advantage and challenges. Explain the different industrial revolutions	10	CO1	L2
	OR			
2)	Explain with the neat diagram of one M2M IOT standardized architecture	10	CO1	L2
3)	Compare simplified, IoT world forum, ETSI standards IoT architectures	10	CO1	L3
	OR			
4)	Write a note on IOT impact on real world.	10	CO1	L2
5)	Describe IOT world forum Standardized architecture.	10	CO1	L2
	OR			
6)	Explain the simplified IoT architecture with a neat diagram.	10	CO1	L1
7)	With a neat diagram, Explain Raspberry Pi learning board with pin configuration	10	CO5	L3
	OR			
8)	Explain Raspberry pi programming language with any 5 programs of your choice	10	CO5	L3
9)	Explain Arduino UNO learning board with its technical specification	10	CO5	L3
	OR			
10)	Compare and contrast Raspberry pi and Arduino UNO boards	10	CO5	L2

1. **Explain IOT with its advantage and challenges. Explain the different industrial revolutions**

IoT is a technology transition in which devices will allow us to sense and control the physical world by making objects smarter and connecting them through an intelligent network.

Challenge	Description
Scale	While the scale of IT networks can be large, the scale of OT can be several orders of magnitude larger. For example, one large electrical utility in Asia recently began deploying IPv6-based smart meters on its electrical grid. While this utility company has tens of thousands of employees (which can be considered IP nodes in the network), the number of meters in the service area is tens of millions. This means the scale of the network the utility is managing has increased by more than 1,000-fold! Chapter 5, "IP as the IoT Network Layer," explores how new design approaches are being developed to scale IPv6 networks into the millions of devices.
Security	With more "things" becoming connected with other "things" and people, security is an increasingly complex issue for IoT. Your threat surface is now greatly expanded, and if a device gets hacked, its connectivity is a major concern. A compromised device can serve as a launching point to attack other devices and systems. IoT security is also pervasive across just about every facet of IoT. For more information on IoT security, see Chapter 8, "Securing IoT."
Privacy	As sensors become more prolific in our everyday lives, much of the data they gather will be specific to individuals and their activities. This data can range from health information to shopping patterns and transactions at a retail establishment. For businesses, this data has monetary value. Organizations are now discussing who owns this data and how individuals can control whether it is shared and with whom.
Big data and data analytics	IoT and its large number of sensors is going to trigger a deluge of data that must be handled. This data will provide critical information and insights if it can be processed in an efficient manner. The challenge, however, is evaluating massive amounts of data arriving from different sources in various forms and doing so in a timely manner. See Chapter 7 for more information on IoT and the challenges it faces from a big data perspective.
Interoperability	As with any other nascent technology, various protocols and architectures are jockeying for market share and standardization within IoT. Some of these protocols and architectures are based on proprietary elements, and others are open. Recent IoT standards are helping minimize this problem, but there are often various protocols and implementations available for IoT networks. The prominent protocols and architectures—especially open, standards-based implementations—are the subject of this book. For more information on IoT architectures, see Chapter 2, "IoT Network Architecture and Design." Chapter 4, "Connecting Smart Objects," Chapter 5, "IP as the IoT Network Layer," and Chapter 6, "Application Protocols for IoT," take a more in-depth look at the protocols that make up IoT.

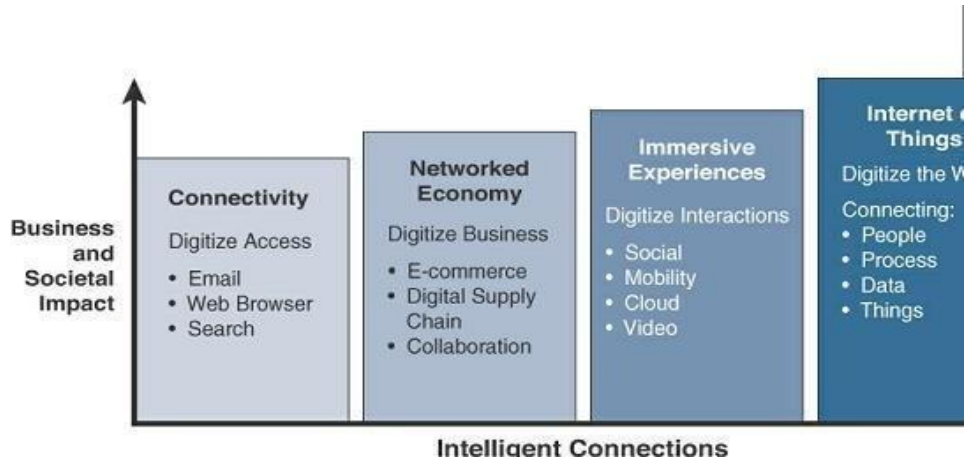


Table 1-1 *Evolutionary Phases of the Internet*

Internet Phase	Definition
Connectivity (Digitize access)	This phase connected people to email, web services, and search so that information is easily accessed.
Networked Economy (Digitize business)	This phase enabled e-commerce and supply chain enhancements along with collaborative engagement to drive increased efficiency in business processes.
Immersive Experiences (Digitize interactions)	This phase extended the Internet experience to encompass widespread video and social media while always being connected through mobility. More and more applications are moved into the cloud.
Internet of Things (Digitize the world)	This phase is adding connectivity to objects and machines in the world around us to enable new services and experiences. It is connecting the unconnected.

2. **Explain with the neat diagram of one M2M IOT standardized architecture**

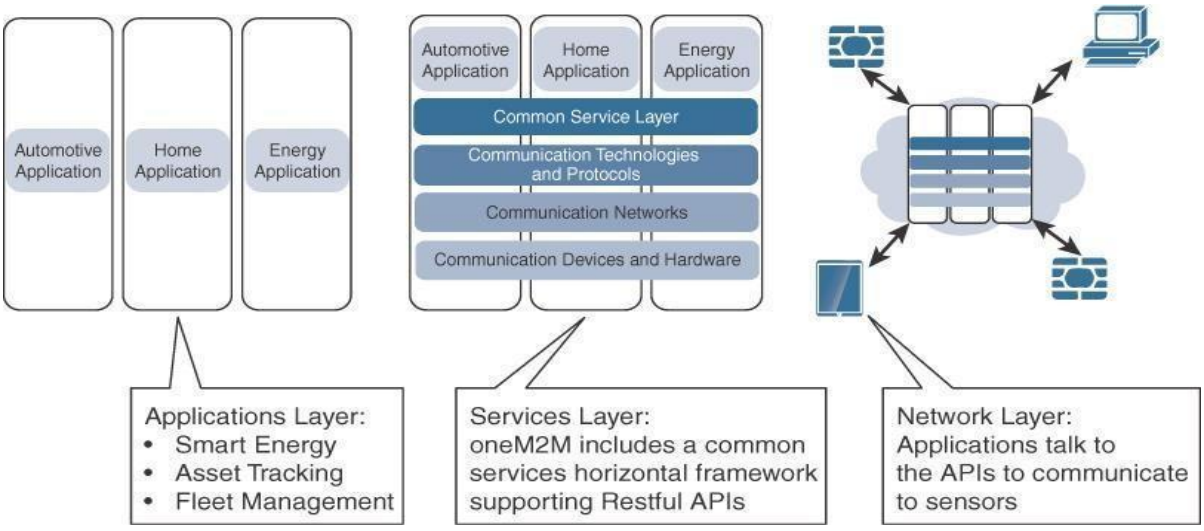
In an effort to standardize the rapidly growing field of machine-to-machine (M2M) communications, the European Telecommunications Standards Institute (ETSI) created the M2M Technical Committee in 2008. The goal of this committee was to create a common architecture that would help accelerate the adoption of M2M applications and devices. Over time, the scope has expanded to include the Internet of Things.

One of the greatest challenges in designing an IoT architecture is dealing with the heterogeneity of devices, software, and access methods. By developing a horizontal platform architecture, oneM2M is developing standards that allow interoperability at all levels of the IoT stack

- **Applications layer:** The oneM2M architecture gives major attention to connectivity between devices and their applications. This domain includes the application-layer protocols and attempts to standardize northbound API definitions for interaction with business intelligence (BI) systems. Applications tend to be industry-specific and have their own sets of data models, and thus they are shown as vertical entities.

- **Services layer:** This layer is shown as a horizontal framework across the vertical industry applications. At this layer, horizontal modules include the physical network that the IoT applications run on, the underlying management protocols, and the hardware. Examples include backhaul communications via cellular, MPLS networks, VPNs, and so on. Riding on top is the common services layer.

- **Network layer:** This is the communication domain for the IoT devices and endpoints. It includes the devices themselves and the communications network that links them. Embodiments of this communications infrastructure include wireless mesh technologies, such as IEEE 802.15.4, and wireless point-to-multipoint systems, such as IEEE 801.11ah.



3. **Compare simplified, IoT world forum, ETSI standards IoT architectures**

Aspect	**Simplified IoT Architecture**	**IoT World Forum**	**ETSI Standards for IoT Architectures**
Overview	Generic term for streamlined IoT architectures	Global platform for IoT discussions	European standards institute for IoT
Characteristics	Minimized complexity, cost-effective	Industry-wide discussions and collaboration	Standards development for IoT technologies
Components	Edge devices, communication protocols, gateway	Event or initiative for IoT stakeholders	Standards for communication, security, etc.
Scalability	Emphasis on scalability and adaptability	Varied discussions on trends and challenges	Aims for interoperability and standardization
Focus Areas	Basic features, ease of implementation	Broad topics including standards, security	Communication protocols, security, and more
Standards Development	May lack advanced features	Information exchange on IoT advancements	Active involvement in shaping IoT standards

Aspect	Simplified IoT Architecture	IoT World Forum	ETSI Standards for IoT Architectures
Key Areas	Streamlined components	Industry insights and real-world scenarios	Security, communication protocols, etc.
Interoperability	Emphasizes adaptability and scalability	Platform for discussions on interoperability	Standards to foster interoperability

4. **Write a note on IOT impact on real world.**

- **Connected roadways**
  Most connected roadways solutions focus on resolving today's transportation challenges.
  These challenges are:
  - Safety
  - Mobility
  - Environment

  These challenges (in connected roadways) will bring many benefits to society.
  These benefits include reduced traffic jams and urban congestion, decreased casualties and fatalities, increased response time for emergency vehicles, and reduced vehicle emissions.

- **Connected Factory**
  The main challenges facing manufacturing in a factory environment today include the following:
  - Accelerating new product and service introductions to meet customer and market opportunities
  - Increasing plant production, quality, and uptime while decreasing cost
  - Mitigating unplanned downtime (which wastes, on average, at least 5% of production)
  - Securing factories from cyber threats
  - Decreasing high cabling and re-cabling costs (up to 60% of deployment costs)
  - Improving worker productivity and safety

- **Smart connected building**
  Buildings have become increasingly complex intersections of structural, mechanical, electrical, and IT components.
  The function of a building
  - is to provide a work environment that keeps the workers comfortable, efficient, and safe.
  - Work areas need to be well lit and kept at a comfortable temperature.
  - To keep workers safe, the fire alarm and suppression system needs to be carefully managed, as do the door and physical security alarm systems.

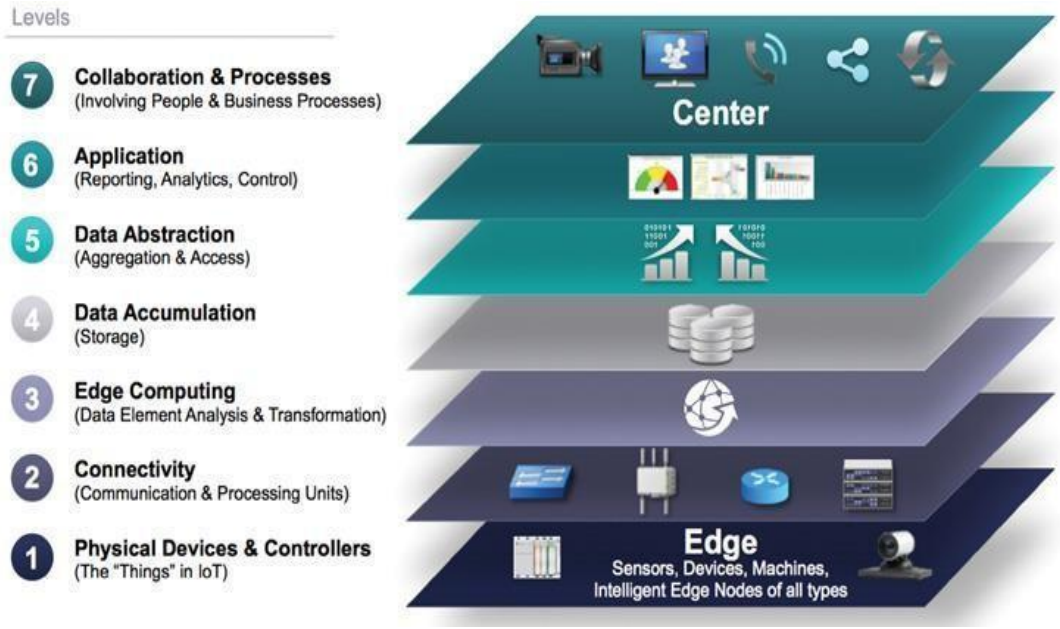
  Many buildings are beginning to deploy sensors throughout the building to detect occupancy.
  - These tend to be motion sensors or sensors tied to video cameras.

- Motion detection occupancy sensors work great if everyone is moving around in a crowded room and can automatically shut the lights off when everyone has left**.**

- **Smart creatures**
  Explanation on smart cockroach as a life savior.

5. **Describe IOT world forum Standardized architecture.**
   This publish a seven-layer IoT architectural reference model.

While various IoT reference models exist, the one put forth by the IoT World Forum offers a clean, simplified perspective on IoT and includes edge computing, data storage, and access. It provides a succinct way of visualizing IoT from a technical perspective. Each of the seven layers is broken down into specific functions, and Using this reference model, we are able to achieve the following:



1. Decompose the IoT problem into smaller parts
2. Identify different technologies at each layer and how they relate to one another
3. Define a system in which different parts can be provided by different vendors
4. Have a process of defining interfaces that leads to interoperability
5. Define a tiered security model that is enforced at the transition points between levels

Layer 1: Physical Devices and Controllers Layer

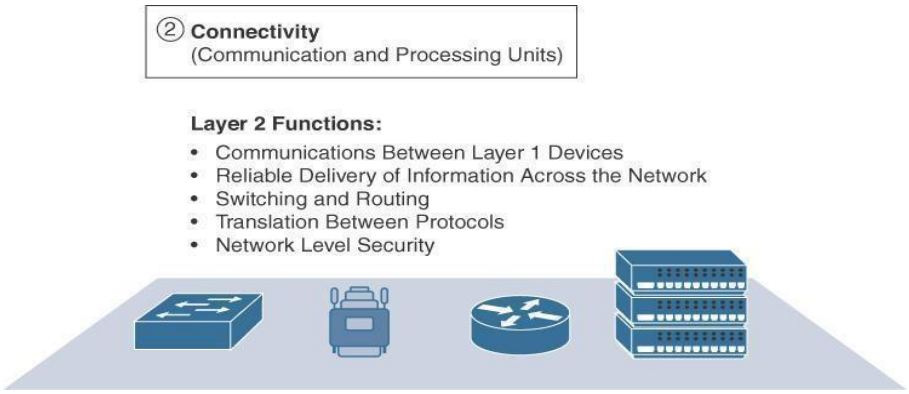
The first layer of the IoT Reference Model is the physical devices and controllers layer. This layer is home to the "things" in the Internet of Things, including the various endpoint devices and sensors that send and receive information. The size of these "things" can range from almost microscopic sensors to giant machines in a factory. Their primary function is generating data and being capable of being queried and/or controlled over a network.

Layer 2: Connectivity Layer

In the second layer of the IoT Reference Model, the focus is on connectivity.

The most important function of this IoT layer is the reliable and timely transmission of data. More specifically, this includes transmissions between Layer 1 devices and the network and between the network and information processing that occurs at Layer 3 (the edge computing layer).

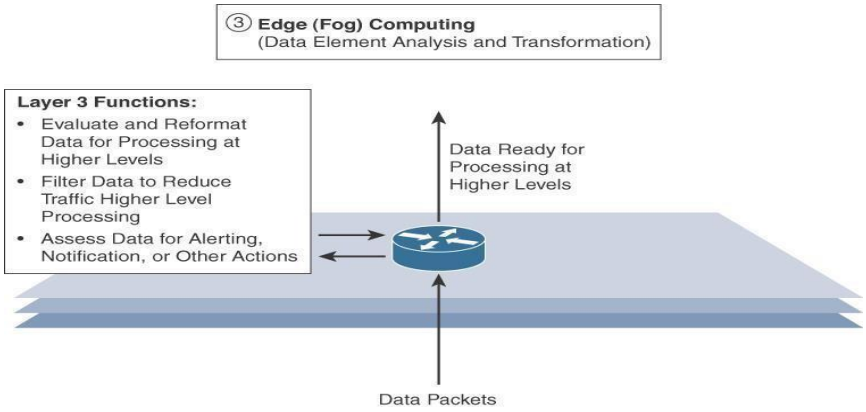
.



IoT Reference Model Connectivity Layer Functions

Layer 3: Edge Computing Layer

Edge computing is the role of Layer 3. Edge computing is often referred to as the "fog" layer and is discussed in the section "Fog Computing," later in this chapter. At this layer, the emphasis is on data reduction and converting network data flows into information that is ready for storage and processing by higher layers. One of the basic principles of this reference model is that information processing is initiated as early and as close to the edge of the network as possible



IoT Reference Model Layer 3 Functions

Another important function that occurs at Layer 3 is the evaluation of data to see if it can be filtered or aggregated before being sent to a higher layer. This also allows for data to be reformatted or decoded, making additional processing by other systems easier. Thus, a critical function is assessing the data to see if predefined thresholds are crossed and any action or alerts need to be sent.

Upper Layers: Layers 4–7

The upper layers deal with handling and processing the IoT data generated by the bottom layer. For the sake of completeness, Layers 4–7 of the IoT Reference Model are summarized in Table .

IoT Reference Model Layer	Functions
Layer 4: Data accumulation layer	Captures data and stores it so it is usable by applications when necessary. Converts event-based data to query-based processing.
Layer 5: Data abstraction layer	Reconciles multiple data formats and ensures consistent semantics from various sources. Confirms that the data set is complete and consolidates data into one place or multiple data stores using virtualization.
Layer 6: Applications layer	Interprets data using software applications. Applications may monitor, control, and provide reports based on the analysis of the data.
Layer 7: Collaboration and processes layer	Consumes and shares the application information. Collaborating on and communicating IoT information often requires multiple steps, and it is what makes IoT useful. This layer can change business processes and delivers the benefits of IoT.

6. Explain the simplified IoT architecture with a neat diagram.

A Simplified IoT Architecture

Although considerable differences exist between the aforementioned reference models, they each approach IoT from a layered perspective, allowing development of technology and standards somewhat independently at each level or domain. The commonality between these frameworks is that they all recognize the interconnection of the IoT endpoint devices to a network that transports the data where it is ultimately used by applications, whether at the data center, in the cloud, or at various management points throughout the stack.

It is not the intention of this book to promote or endorse any one specific IoT architectural framework. In fact, it can be noted that IoT architectures may differ somewhat depending on the industry use case or technology being deployed, and each has merit in solving the IoT heterogeneity problem discussed earlier. Thus, in this book we present an IoT framework that highlights the fundamental building blocks that are common to most IoT systems and which is intended to help you in designing an IoT network. This framework is presented as two parallel stacks: The IoT Data Management and Compute Stack and the Core IoT Functional Stack. Reducing the framework down to a pair of three-layer stacks in no way suggests that the model lacks the detail necessary to develop a sophisticated IoT strategy. Rather, the intention is to simplify the IoT architecture into its most basic building blocks and then to use it as a foundation to understand key design and deployment principles that are applied to industry-specific use cases. All the layers of more complex models are still covered, but they are grouped here in functional blocks that are easy to understand. Figure 2-6 illustrates the simplified IoT model presented in this book.

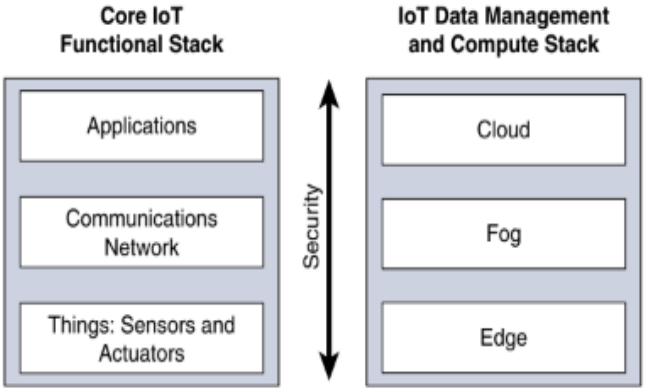


Figure 2-6 *Simplified IoT Architecture*

Nearly every published IoT model includes core layers similar to those shown on the left side of Figure 2-6, including "things," a communications network, and applications. However, unlike other models, the framework presented here separates the core IoT and data management into parallel and aligned stacks, allowing you to carefully examine the functions of both the network and the applications at each stage of a complex IoT system. This separation gives you better visibility into the functions of each layer.

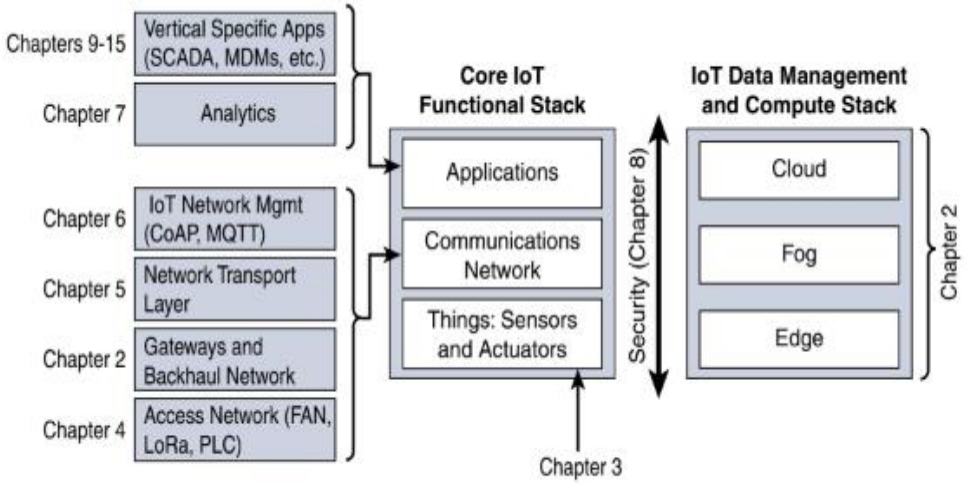


Figure 2-7 *Expanded View of the Simplified IoT Architecture*

The Core IoT Functional Stack

IoT networks are built around the concept of "things," or smart objects performing functions and delivering new connected services. These objects are "smart" because they use a combination of contextual information and configured goals to perform actions. These actions can be self-contained (that is, the smart object does not rely on external systems for its actions); however, in most cases, the "thing" interacts with an external system to report information that the smart object collects, to exchange with other objects, or to interact with a management platform. In this case, the management platform can be used to process data collected from the smart object and also guide the behavior of the smart object. From an architectural standpoint, several components have to work together for an IoT network to be operational:

- **"Things" layer:** At this layer, the physical devices need to fit the constraints of the environment in which they are deployed while still being able to provide the information needed.
- **Communications network layer:** When smart objects are not self-contained, they need to communicate with an external system. In many cases, this communication uses a wireless technology. This layer has four sublayers:
  - **Access network sublayer:** The last mile of the IoT network is the access network. This is typically made up of wireless technologies such as 802.11ah, 802.15.4g, and LoRa. The sensors connected to the access network may also be wired.
  - **Gateways and backhaul network sublayer:** A common communication system organizes multiple smart objects in a given area around a common gateway. The gateway communicates directly with the smart objects. The role of the gateway is to forward the collected information through a longer-range medium (called the backhaul) to a headend central station where the information is processed. This information exchange is a Layer 7 (application) function, which is the reason this object is called a gateway. On IP networks, this gateway also forwards packets from one IP network to another, and it therefore acts as a router.
  - **Network transport sublayer:** For communication to be successful, network and transport layer protocols such as IP and UDP must be implemented to support the variety of devices to connect and media to use.
  - **IoT network management sublayer:** Additional protocols must be in place to allow the headend applications to exchange data with the sensors. Examples include CoAP and MQTT.
- **Application and analytics layer:** At the upper layer, an application needs to process the collected data, not only to control the smart objects when necessary, but to make intelligent decision based on the information collected and, in turn, instruct the "things" or other systems to adapt to the analyzed conditions and change their behaviors or parameters.

IoT Data Management and Compute Stack

One of the key messages in the first two chapters of this book is that the massive scale of IoT networks is fundamentally driving new architectures. For instance, Figure 1-2 in Chapter 1 illustrates how the "things" connected to the Internet are continuing to grow exponentially, with a prediction by Cisco that by 2020 there will be more than 50 billion devices connected to some form of an IP network. Clearly, traditional IT networks are not prepared for this magnitude of network devices. However, beyond the network architecture itself, consider the data that is generated by these devices. If the number of devices is beyond conventional numbers, surely the data generated by these devices must also be of serious concern.

In fact, the data generated by IoT sensors is one of the single biggest challenges in building an IoT system. In the case of modern IT networks, the data sourced by a computer or server is typically generated by the client/server communications model, and it serves the needs of the application. In sensor networks, the vast majority of data generated is unstructured and of very little use on its own. For example, the majority of data generated by a smart meter is nothing more than polling data; the communications system simply determines whether a network connection to the meter is still active. This data on its own is of very little value. The real value of a smart meter is the metering data read by the meter management system (MMS). However, if you look at the raw polling data from a different perspective, the information can be very useful. For example, a utility may have millions of meters covering its entire service area. If whole sections of the smart grid start to show an interruption of connectivity to the meters, this data can be analyzed and combined with other sources of data, such as weather reports and electrical demand in the grid, to provide a complete picture of what is happening. This information can help determine whether the loss of connection to the meters is truly a loss of power or whether some other problem has developed in the grid. Moreover, analytics of this data can help the utility quickly determine the extent of the service outage and repair the disruption in a timely fashion.

In most cases, the processing location is outside the smart object. A natural location for this processing activity is the cloud. Smart objects need to connect to the cloud, and data processing is centralized. One advantage of this model is simplicity. Objects just need to connect to a central cloud application. That application has visibility over all the IoT nodes and can process all the analytics needed today and in the future.

However, this model also has limitations. As data volume, the variety of objects connecting to the network, and the need for more efficiency increase, new requirements appear, and those requirements tend to bring the need for data analysis closer to the IoT system. These new requirements include the following:

- **Minimizing latency:** Milliseconds matter for many types of industrial systems, such as when you are trying to prevent manufacturing line shutdowns or restore electrical service. Analyzing data close to the device that collected the data can make a difference between averting disaster and a cascading system failure.
- **Conserving network bandwidth:** Offshore oil rigs generate 500 GB of data weekly. Commercial jets generate 10 TB for every 30 minutes of flight. It is not practical to transport vast amounts of data from thousands or hundreds of thousands of edge devices to the cloud. Nor is it necessary because many critical analyses do not require cloud-scale processing and storage.
- **Increasing local efficiency:** Collecting and securing data across a wide geographic area with different environmental conditions may not be useful. The environmental conditions in one area will trigger a local response independent from the conditions of another site hundreds of miles away. Analyzing both areas in the same cloud system may not be necessary for immediate efficiency.

An important design consideration, therefore, is how to design an IoT network to manage this volume of data in an efficient way such that the data can be quickly analyzed and lead to business benefits. The volume of data generated by IoT devices can be so great that it can easily overrun the capabilities of the headend system in the data center or the cloud. For example, it has been observed that a moderately sized smart meter network of 1 million meters will generate close to 1 billion data points each day (including meter reads and other instrumentation data), resulting in 1 TB of data. For an IT organization that is not prepared to contend with this volume of data storage and real-time analysis, this creates a whole new challenge.

The volume of data also introduces questions about bandwidth management. As the massive amount of IoT data begins to funnel into the data center, does the network have the capacity to sustain this volume of traffic? Does the application server have the ability to ingest, store, and analyze the vast quantity of data that is coming in? This is sometimes referred to as the "impedance mismatch" of the data generated by the IoT system and the management application's ability to deal with that data.

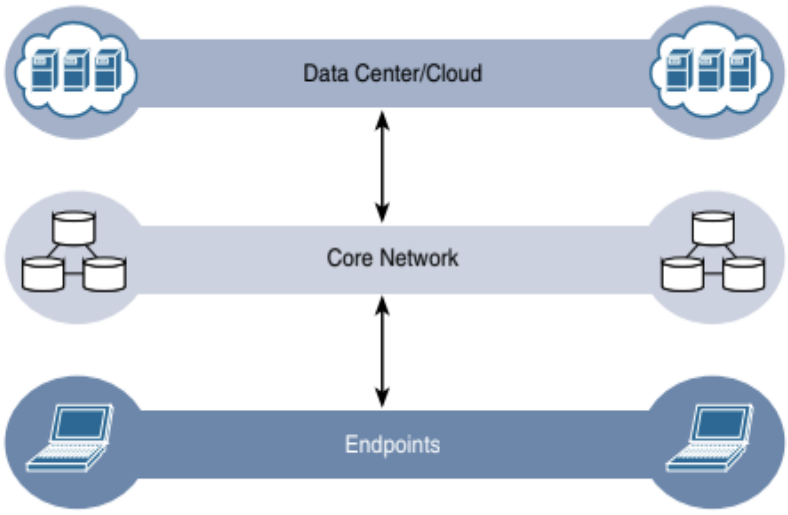


Figure 2-14 *The Traditional IT Cloud Computing Model*

Fog Computing

The solution to the challenges mentioned in the previous section is to distribute data management throughout the IoT system, as close to the edge of the IP network as possible. The best-known embodiment of edge services in IoT is fog computing. Any device with computing, storage, and network connectivity can be a fog node. Examples include industrial controllers, switches, routers, embedded servers, and IoT gateways. Analyzing IoT data close to where it is collected minimizes latency, offloads gigabytes of network traffic from the core network, and keeps sensitive data inside the local network.

An advantage of this structure is that the fog node allows intelligence gathering (such as analytics) and control from the closest possible point, and in doing so, it allows better performance over constrained networks. In one sense, this introduces a new layer to the traditional IT computing model, one that is often referred to as the "fog layer." Figure 2-15 shows the placement of the fog layer in the IoT Data Management and Compute Stack.

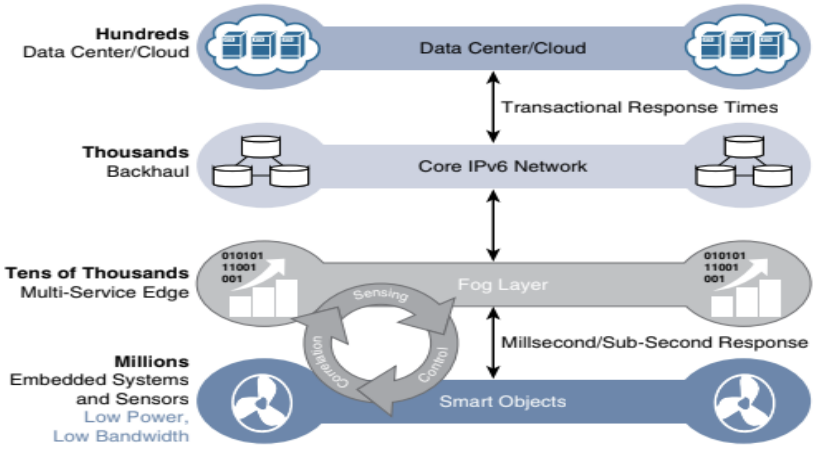


Figure 2-15 *The IoT Data Management and Compute Stack with Fog Computing*

Fog services are typically accomplished very close to the edge device, sitting as close to the IoT endpoints as possible. One significant advantage of this is that the fog node has contextual awareness of the sensors it is managing because of its geographic proximity to those sensors. For example, there might be a fog router on an oil derrick that is monitoring all the sensor activity at that location. Because the fog node is able to analyze

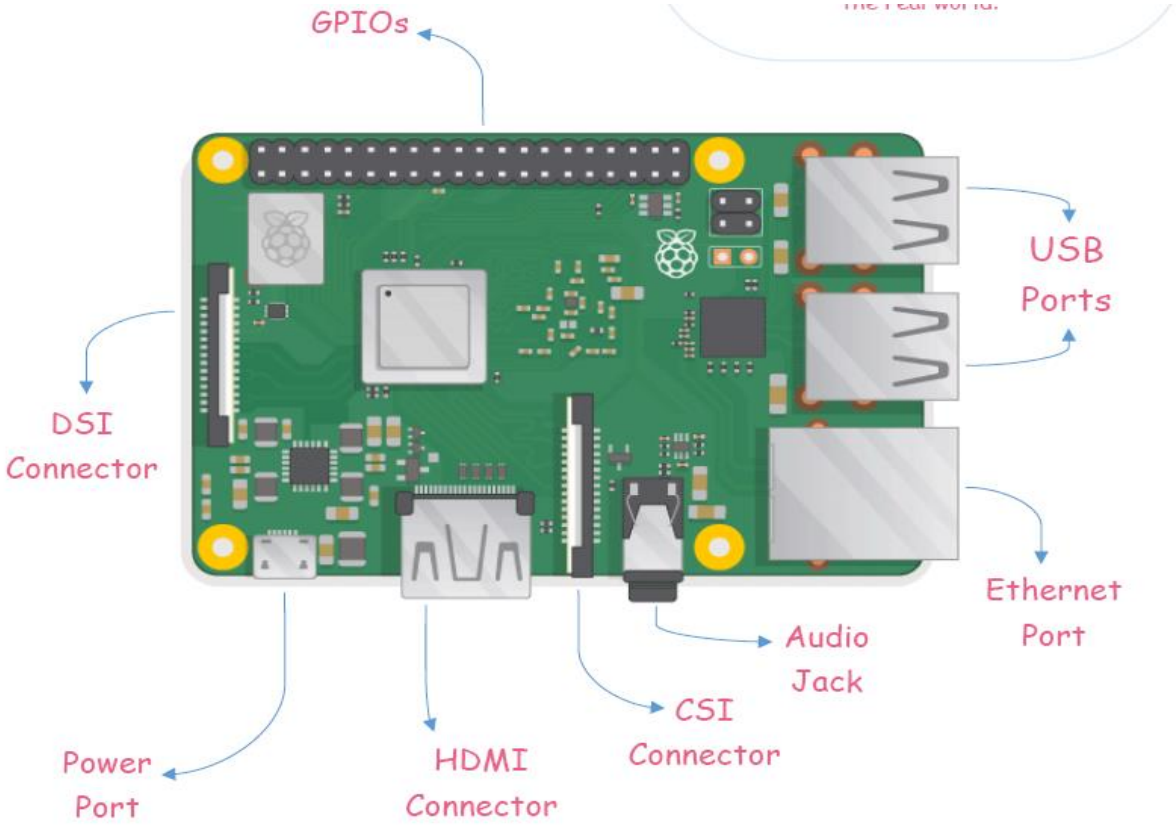
Edge Computing

Fog computing solutions are being adopted by many industries, and efforts to develop distributed applications and analytics tools are being introduced at an accelerating pace. The natural place for a fog node is in the network device that sits closest to the IoT endpoints, and these nodes are typically spread throughout an IoT network. However, in recent years, the concept of IoT computing has been pushed even further to the edge, and in some cases it now resides directly in the sensors and IoT devices.

Note Edge computing is also sometimes called "mist" computing. If clouds exist in the sky, and fog sits near the ground, then mist is what actually sits on the ground. Thus, the concept of mist is to extend fog to the furthest point possible, right into the IoT endpoint device itself.

IoT devices and sensors often have constrained resources, however, as compute capabilities increase. Some new classes of IoT endpoints have enough compute capabilities to perform at least low-level analytics and filtering to make basic decisions. For example, consider a water sensor on a fire hydrant. While a fog node sitting on an electrical pole in the distribution network may have an excellent view of all the fire hydrants in a local neighborhood, a node on each hydrant would have clear view of a water pressure drop on its own line and would be able to quickly generate an alert of a localized problem. The fog node, on the other hand, would have a wider view and would be able to ascertain whether the problem was more than just localized but was affecting the entire area. Another example is in the use of smart meters. Edge compute–capable meters are able to communicate with each other to share information on small subsets of the electrical distribution grid to monitor localized power quality and consumption, and they can inform a fog node of events that may pertain to only tiny sections of the grid. Models such as these help ensure the highest quality of power delivery to customers.

7. With a neat diagram, Explain Raspberry Pi learning board with pin configuration



- GPIOs- Connect devices to interact with the real world, for instance, sensors, LEDs, Motors etc.
- USB Port- to connect a mouse, a keyboard or other peripherals.
- Ethernet Port- to connect to the internet using an Ethernet cable.
- Audio Jack- to connect an audio device.
- CSI Connector- to connect a camera with a CSI(Camera Serial Interface) ribbon.
- HDMI Connector- to connect a monitor or TV.
- Power Port- to power up your Pi.
- DSI Connector- to connect DSI compatible Display.

8. Explain Raspberry pi programming language with any 5 programs of your choice

Run some python programs on Pi like:

a) Read your name and print Hello message with name

```
name = input('What is your
name?\n') print ('Hello %s.' %
name)
```

b) Read two numbers and print their sum, difference, product and division.

```
num1 = int(input("Enter First Number: "))
num2 = int(input("Enter Second Number:
"))
print("Enter which operation would you like to perform?")
ch = input("Enter any of these char for specific operation +,-,*,/: ")
result =
0 if ch
== '+':
result = num1 +
num2 elif ch == '-':
result = num1 -
num2 elif ch == '*':
result = num1 *
num2 elif ch == '/':
result = num1 /
num2 else:
print("Input character is not
recognized!") print(num1, ch , num2, ":",
result)
```

c) Word and character count of a given string.

```
word_count = 0
char_count = 0
usr_input = input("Enter a string :
") split_string = usr_input.split()
word_count = len(split_string)
```

```
for word in split_string:
char_count +=

len(word)
print("Total words : {}".format(word_count))
print("Total characters :
{}".format(char_count))
```

d) Area of a given shape (rectangle, triangle and circle) reading shape and appropriate values from standard input.

```
width = float(input('Please Enter the Width of a Rectangle: '))
height = float(input('Please Enter the Height of a Rectangle: '))
# calculate the
area Area = width
* height
# calculate the Perimeter
Perimeter = 2 * (width +
height)
print("\n Area of a Rectangle is: %.2f" %Area)
print(" Perimeter of Rectangle is: %.2f"
%Perimeter)
```

Python Program:

```
# Python Program to find the area of triangle
# Three sides of the triangle a, b and c are provided by the user
a = float(input('Enter first side: '))
b = float(input('Enter second
side: ')) c = float(input('Enter
third side: '))

# calculate the semi-
perimeter s = (a + b + c) /

2
# calculate the area
area = (s*(s-a)*(s-b)*(s-c)) ** 0.5
print('The area of the triangle is %0.2f' %area)
# Python Program to find Diameter, Circumference, and Area of a Circle
PI = 3.14
radius = float(input(' Please Enter the radius of a circle: '))
diameter = 2 * radius

circumference = 2 * PI *
radius area = PI * radius *
radius
print(" \n Diameter of a Circle = %.2f" %diameter)
print(" Circumference of a Circle = %.2f" %circumference)
print(" Area of a Circle = %.2f" %area)
```

Combined Python Program:

```
#Area
print("Select one of the following:")
print("1. Rectangle\n2. Triangle\n3.
Circle") s=input("Enter your choice: ")
if s=='1':
x=int(input("Enter
length:"))
y=int(input("Enter
breadth:"))
print("Area={}".format(x*
y))
elif s=='2':
x=int(input("Enter
```

```
base:"))
y=int(input("Enter
height:"))
print("Area={}".format(0.5*x*y))
elif s=='3':
x=int(input("Enter
radius:"))
print("Area={}".format(3.14*x*x))
else:
print("Enter a valid choice")
```

e) Print a name 'n' times, where name and n are read from standard input, using for and while loops.

Python Program (Using For Loop):

```
i=1
print ("enter the

name")
name=raw_input()
print ("enter the no of time" )
num=raw_input()
#print
(type(num))
num=int(num)
for i in
range(1,num+1):
print (i , name)
i=i+1
```

Python Program (Using While Loop):

```
print ("enter the name")
name=raw_input()
print ("enter the no of time" )
num=raw_input()
print
(type(num))
num=int(num
) i=1
while(i<=num
):
print
(name)
i=i+1
```

Python Program (Without Loop):

```
def name(n):
if n != 0:
name(n-1)
print("Name")

name(10)
```

f) Handle Divided by Zero Exception.

```
print ("enter two no n1 and n2")
n1=raw_input()
n2=raw_input()
n1=int(n1)
n2=int(n2)
try:
div=n1/n2
```

```
print
(div)
except ZeroDivisionError:
print ("zero division is
handled") print ("out of try
catch block ")
#DivideByZero
Exception
x=int(input("First
No:"))
y=int(input("Second
No:")) try:
print("x/y={}".format(x/y))
except Exception:
print("DivideByZero Exception")
```

g) Print current time for 10 times with an interval of 10 seconds.
Read a file line by line and print the word count of each line.

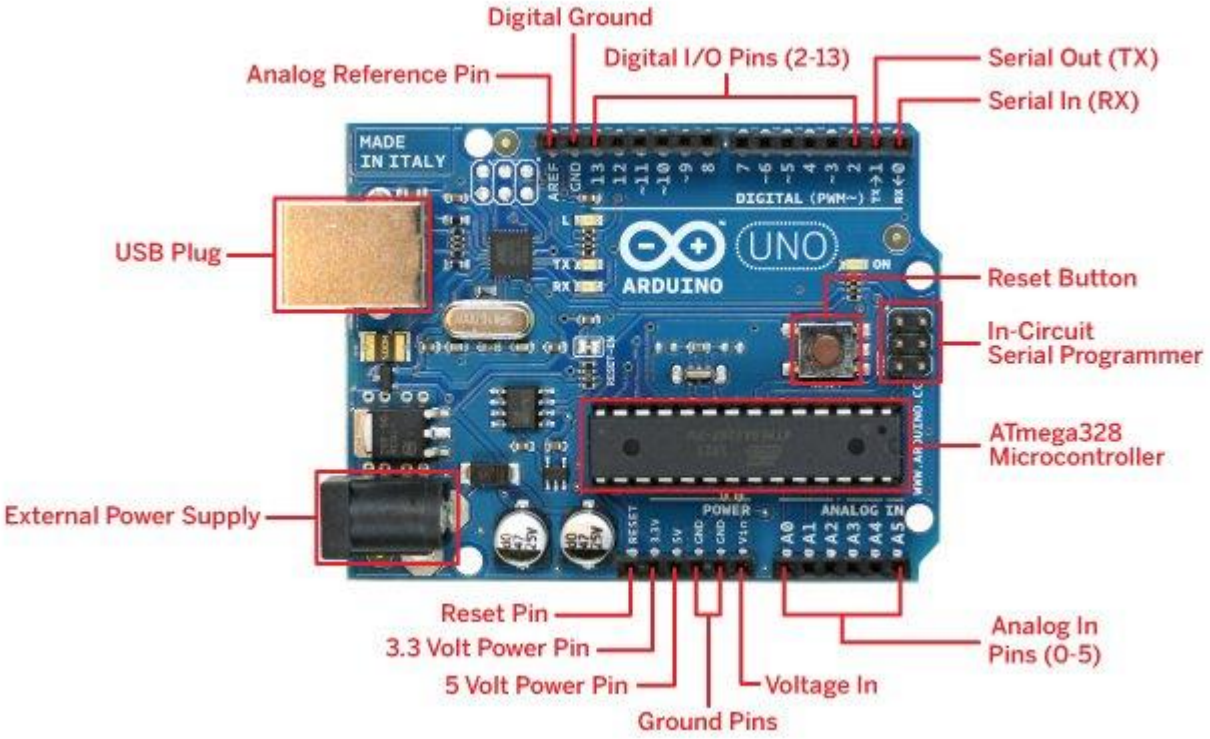
Python Program:

```
import time
for i in range(1,11):
zz=time.asctime(time.localtime(time.time()))
zz=zz[11:19]
print (zz)
print
(time.asctime(time.localtime(time.time())))
time.sleep(10)
```

Python Program:

```
#Current time 10
times import
datetime import
time
for i in range(0,10):
print(datetime.datetime.now().time())
time.sleep(10)
```

9. Explain Arduino UNO learning board with its technical specification



The Arduino Uno is a popular open-source microcontroller board that is widely used for prototyping and DIY electronics projects. Here are the technical specifications for the Arduino Uno:

Microcontroller: ATmega328P

Clock Speed: 16 MHz
Flash Memory: 32 KB (2 KB used by the bootloader)
SRAM: 2 KB
EEPROM: 1 KB
Operating Voltage: 5V

Input Voltage (recommended): 7-12V

Input Voltage (limits): 6-20V

Digital I/O Pins: 14 (of which 6 provide PWM output)

PWM Digital I/O Pins: 6

Analog Input Pins: 6

DC Current per I/O Pin: 20 mA

DC Current for 3.3V Pin: 50 mA

Voltage Regulator: AMS1117 5.0V

USB Interface: ATmega16U2

Communication:

Serial Communication: Yes (via USB and hardware UART)
I2C: Yes
SPI: Yes
Clock Source: 16 MHz Crystal Oscillator

Size: 68.6 mm x 53.4 mm

Weight: 25 g

LEDs:

13: Digital Pin 13 (default built-in LED)
TX, RX: Serial communication LEDs
Reset Button: Yes

Power Jack: 2.1mm center-positive

In-Circuit Serial Programming (ICSP) Header: Yes

Operating Temperature Range: -40°C to +85°C

USB Connector: Type-B

Programming: Via USB or ICSP

Bootloader: Yes (Optiboot)

Board Type: Digital

Compatible Shields: Yes (with the standard Arduino form factor)

Open-Source: Yes (Schematics and design files are available for free)

IDE Compatibility: Arduino IDE (and other compatible IDEs)

10. Compare and contrast Raspberry pi and Arduino UNO boards

Feature	Raspberry Pi	Arduino Uno
Microcontroller/Processor	Broadcom BCM2835 (varies by model)	ATMega328P
Architecture	ARM (varies by model)	AVR
Clock Speed	Varies (e.g., 1.2 GHz, 1.4 GHz)	16 MHz
Operating System Support	Linux-based (Raspbian, etc.)	None (Real-time or single-task OS)
GPIO Pins	Yes (40 GPIO pins on most models)	Yes (14 digital pins on Arduino Uno)
Analog Input Pins	Via GPIO pins (with analog-to-digital converters)	6 dedicated analog input pins
PWM Pins	Yes (software-based)	Yes (6 hardware PWM pins)
Digital I/O Pins	Yes	Yes
USB Ports	Yes (multiple USB ports)	Yes (1 USB port for programming)
Ethernet	Yes (on some models)	No
Wi-Fi	Yes (on some models)	No
Bluetooth	Yes (on some models)	No
Storage	MicroSD card	Flash memory (32 KB)
RAM	Varies (e.g., 1GB, 2GB, 4GB)	2 KB SRAM
Power Supply	Micro-USB or USB-C (varies by model)	7-12V DC (via power jack or USB)

Feature	Raspberry Pi	Arduino Uno
Operating Voltage	5V	5V
Programming Language	Multiple (Python, C++, etc.)	C/C++
Development Environment	Various (e.g., Thonny, VS Code)	Arduino IDE, PlatformIO
Use Case	General-purpose computing, multimedia applications, IoT projects	Embedded systems, robotics, simple control systems
Complexity	More complex due to OS and higher-level capabilities	Simpler, focused on real-time control