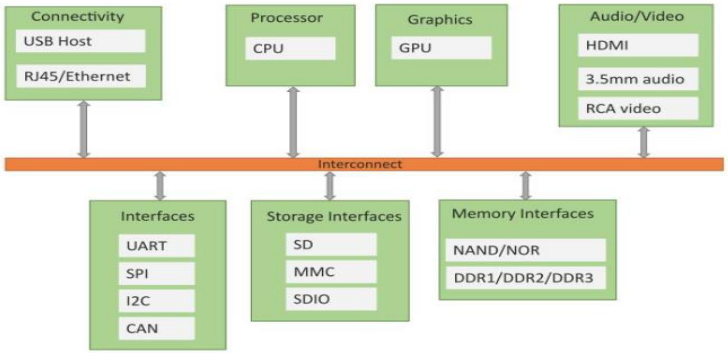
Module 5 (Developing IoT Systems)

IoT Logical Design using Python, IoT Physical Devices and Endpoints - Raspberry Pi interfaces, Programming Raspberry Pi using Python, Other IoT devices, IoT Physical devices and Cloud offerings, Cloud Storage Models, WAMP - Autobahn for IoT, Django, Designing RESTful Web API, Cloud Web Services for IoT.

1) What are the basic building blocks of an IoT Device?

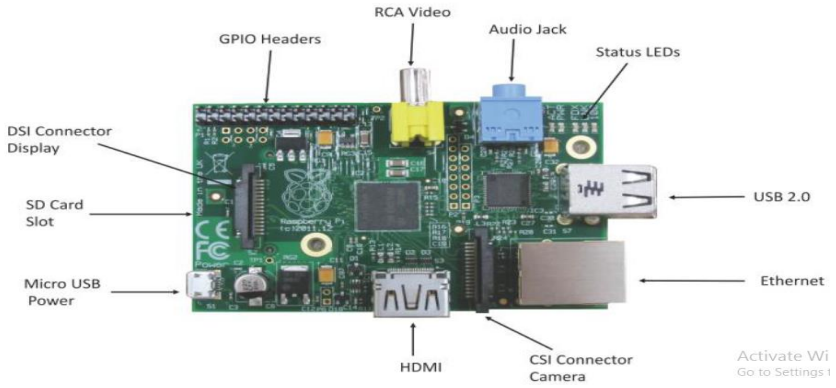
1. Sensing- Sensors can be either on-board the IoT device or attached to the device.
2. Actuation- IoT devices can have various types of actuators attached that allow taking actions upon the physical entities in the vicinity of the device.
3. Communication-Communication modules are responsible for sending collected data to other devices or cloud-based servers/storage and receiving data from other devices and commands from remote applications.
4. Analysis & Processing- Analysis and processing modules are responsible for making sense of the collected data.

Block diagram of an IoT Device



2)Write a note on Raspberry Pi?

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



1. Processor & RAM: Raspberry Pi is based on an ARM processor. The latest version of Raspberry Pi comes with 700 MHz Low Power ARM11 processor and 512 MB SDRAM.
2. USB Ports: Raspberry Pi comes with two USB 2.0 ports. The USB ports on Raspberry Pi can provide a current upto 100mA. For connecting devices that draw current more than 100mA, an external USB powered hub is required.
3. Ethernet Ports: Raspberry Pi comes with a standard RJ45 Ethernet port. You can connect an Ethernet cable or a USB Wifi adapter to provide Internet connectivity.
4. HDMI Output: The HDMI port on Raspberry Pi provides both video and audio output. You can connect the Raspberry Pi to a monitor using an HDMI cable.
5. Composite Video Output: Raspberry Pi comes with a composite video output with an RCA jack. The RCA jack can be used to connect old televisions that have an RCA input only.
6. Audio Output: Raspberry Pi has a 3.5mm audio output jack. This audio jack is used for providing audio output to old televisions along with the RCA jack for video.

7. GPIO Pins: Raspberry Pi comes with a number of general purpose input/ouput pins. There are four types of pins on Raspberry Pi - true GPIO pins, 12C interface pins, SPI interface pins and serial Rx and Tx pins.
8. Display Serial Interface (DSI): The DSI interface can be used to connect an LCD panel to Raspberry Pi.
9. Camera Serial Interface (CSI): The CSI interface can be used to connect a camera module to Raspberry Pi.
10. Status LEDs: Raspberry Pi has five status LEDs Table lists Raspberry Pi status LEDs and their functions.

Status LED	Function
ACT	SD Card access
PWR	3.3V power is present
FDX	Full duplex LAN connected
LNK	Link/Network activity
100	100Mbit LAN Connected

11. SD Card Slot: Raspberry Pi does not have a built in operating system and storage. You can plug-in an SD card loaded with a Linux image to the SD card slot.
12. Power Input: Raspberry Pi has a micro-USB connector for power input.

3) Explain Raspberry Pi Interfaces?

1. Serial- The serial interface on Raspberry Pi has receive (Rx) and transmit (Tx) pins for communication with serial peripherals.
2. SPI- Serial Peripheral Interface (SPI) is a synchronous serial data protocol used for communicating with one or more peripheral devices using a master slave architecture. SPI requires 4 wires:
    1) MOSI (Master Out Slave In)-Master line for sending data to the peripherals.
    2) MISO (Master In Slave Out)-Slave line for sending data to the master.
    3) SCK-clock generated by master to synchronise data transmission.
    4) CE0: To enable or disable slave device0
       CE1: To enable or disable slave device1
3. I2C- the I2C interface pins on Raspberry Pi allow you to connect hardware modules. I2C interface allows synchronous data transfer with just two pins - SDA (data line) and SCL (clock line).

4)List Frequently used Raspberry Pi Commands?

Command	Function
cd	Change directory
cat	Show file contents
ls	List files and folders
locate	Search for a file
lsusb	List USB devices
pwd	Print name of present working directory
mkdir	Make directory
mv	Move(rename)file
rm	Remove file
reboot	Reboot device
shutdown	Shutdown device
grep	Print lines matching a pattern
df	Report file system disk space usage
ifconfig	Configure a network interface

5) Programming Raspberry Pi with Python

1. Controlling LED with Raspberry Pi
2. Interfacing an LED and Switch with Raspberry Pi
3. Interfacing a light Sensor (LDR) with Raspberry Pi

5.1) Write a Python program for blinking LED?

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(18,GPIO.OUT)
While True:
    GPIO.output(18,True)
```

```
        time.sleep(1)
        GPIO.output(18,False)
        time.sleep(1)
```

5.2) Interface LED and switch with Raspberry Pi?

```
from time import sleep
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)

#Switch Pin
GPIO.setup(25, GPIO.IN)

# KLED Pin
GPIO.setup(18, GPIO.OUT)
state=false
def togglelED(pin):
        state = not state
        GPIO.output(pin, state)
while True:
            try:
                if (GPIO.input(25) == True):
                    toggleLED(pin)
                    sleep(.01)
            except KeyboardInterrupt:
                    exit()
```

5.3) Python program for switching LED/Light based on LDR Reading?

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
 ldr_threshold=1000
LDR_PIN=18
LIGHT_PIN=25
def readLDR(PIN):
        reading=0
        GPIO.setup(LIGHT_PIN, GPIO.0UT)
        GPIO.output(PIN, False)
        time.sleep(0.1)
        GPIO.setup(PIN, GPIO.IN)
        while (GPIO.input(PIN)==Fa1se):
                reading=reading+1
        return reading
        def switchOnLight(PIN):
                GPIO.setup(PIN, GPIO.OUT)
                GPIO.output(PIN, True)
        def switchOfLight(PIN):
                GPIO.setup(PIN, GPIO.OUT)
                GPIO.output(PIN, False)
        While True:
                ldr_reading=readLDR(LDR_PIN)
                if ldr_reading<ldr_threshold:
                        switchOnLight(LIGHT_PIN)
                else:
                        switchOffLight(LIGHT_PIN)
                time.sleep(1)
```

6) Compare Single board mini-computers?

OR

Compare Raspberry Pi with other IoT devices?

	Raspberry Pi	**pcDuino**	**BeagleBone Black**	**Cubieboard**
CPU	700MHz ARM11 Processsor	1GHZ ARM Cortex A8	1GHZ ARM Cortex A8	1GHZ ARM Cortex A7
GPU	Dual Core VideoCore IV Multimedia Co-Processor	Mali 400	PowerVR	Dual Core ARM Mali 400
Memory	512 MB	1GB	512 MB	1GB
Storage	-	2GB Flash	2GB flash	4GB NAND Flash

Networking	10/100M Ethernet	10/100M Ethernet	10/100M Ethernet	10/100M Ethernet
Input/Output	2USB,SD,MMC,SDIO ,Card slot	-	4+1 USB,MicroSD slot	2USB,MicroSD slot,SATA,IR Sensor
Interfaces	GPIO,SPI,I2C,Serial	GPIO,SPI,I2C,Serial, ADC,PWM	GPIO,SPI,I2C,Serial,CAN,MMC	SPI,I2C,ADC,CVBS,VGA, R-TP
OS	Raspbian,pidora,RISC OS,Arch	Ubuntu,Android	Ubuntu,Android, Angstrom LInux	Android,Official Linux distribution
Video	HDMI,Composite RCA	HDMI	HDMI	HDMI
Audio	3.5mm jack HDMI	HDMI	HDMI	HDMI
Power	5VDC/700mA	5V/2A	5VDC/460mA	5V/2A

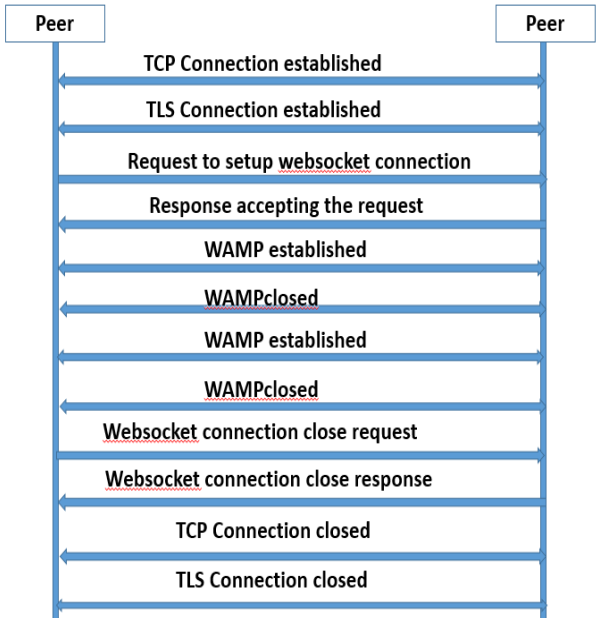
7) Explain the working of Web Application Messaging Protocol (WAMP)?

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

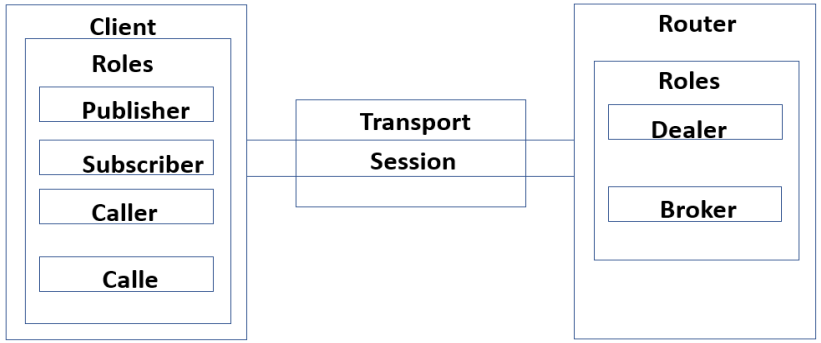
Key concepts of WAMP

1) **Transport:** Transport is a channel that connects two peers. The default transport for WAMP is websocket.
2) **Session:** Session is a conversation between two peers that runs over a transport.
3) **Client:** Clients are peers that can have one or more roles.
   - In the publish–subscribe model, the Client can have the following roles:
   a. Publisher: Publisher publishes events (including payload) to the topic maintained by the Broker.
   b. Subscriber: Subscriber subscribes to the topics and receives the events including the payload.
   - In the RPC model, the Client can have the following roles:
   a. Caller: Caller issues calls to the remote procedures along with call arguments.
   b. Callee: Callee executes the procedures to which the calls are issued by the Caller and returns the results to the Caller.
4) **Router:** Routers are peers that perform generic call and event routing.
   - In the publish–subscribe model, the Router has the role of a Broker.
   -Broker: Broker acts as a Router and routes messages published to a topic to all the subscribers subscribed to the topic.
   - In the RPC model, the Router has the role of a Dealer.
   – Dealer: Dealer acts a router and routes RPC calls from the Caller to the Callee and routes results from the Callee to the Caller.
5) **Application code**: Application code runs on the Clients (Publisher, Subscriber, Callee or Caller).

WAMP Peer to Peer interaction



WAMP session between client and router



- Figure shows a WAMP Session between Client and Router, established over a transport. Figure shows the WAMP Protocol interaction between peers.
- According to these figures, the Wamp transport used is web socket. WAMP sessions are established over web socket transport within the lifetime of web socket transport.
- The client (in the publisher role) Runs a WAMP application component that publishes a message to the router.
- The router (in the broker role) runs on the server and routes the message to the subscriber. It decouples the publisher from the subscribers.
- The communication between publisher- broker and broker to publisher happens over a WAMP web web-socket session.

Advantages

- This protocol combines two patterns (Pub-Sub & RPC), allowing it to be used for the entire messaging requirements of an application, reducing technology stack complexity.
- reduces networking overhead.

Disadvantages

- It is not an official standard.
- No higher-level software architectural styles like REST.
- Requires to deploy an additional component- the WAMP router.

8) Write a note on Xively Cloud for IoT?

- Xively is a commercial Platform-as-a-Service. Xively can be used for creating solutions for Internet of Things.
- With Xively cloud, IoT developers can focus on:
  - ❑ The front-end infrastructure
  - ❑ Devices for IoT
  - ❑ Management of back-end data collection infrastructure
- Xively Platform Comprises of:
  - Message bus for real time message management.
  - Data Services for time series archiving.
  - Directory services for device management

Advantages of Xivelv:

- Xively provides an extensive support for various languages and platforms.
- Xively libraries leverage standards-based API over HTTP, Sockets and MQTT for connecting IoT devices to the Xively Cloud.

9)Write a note on web application framework-Django?

- Django is an open-source web application framework for developing web applications and dynamic websites in Python.
- Django is based on the well-known Model-Template-View architecture and provides a separation of the data model from the business rules and the user interface.
- Django provides a unified API to a database backend. Therefore, web applications built with Django can work with different databases without requiring any code changes.
- With this flexibility in web application design combined with the powerful capabilities of the Python language and the Python ecosystem, Django is best suited for IoT applications.

Django Architecture

- Django is a Model-Template-View (MTV) framework wherein the roles of model, template, and view, respectively, are:
- **Model-** The model acts as a definition of some stored data and handles the interactions with the database. In a web application, the data can be stored in a relational database, non-relatioraal database, an XML file, etc.
- **Template-** In a typical Django web application, the template is simply an HTML page with a few extra; placeholders. Django's template language can be used to create various forms of text files. (XML, email, CSS, Javascript, etc.)
- **View-** The view ties the model to the template. The view is where you write the code that, generates the web pages. View determines what data is to be displayed, retrieves the data from the database and passes the data to the template.

10) Designing a RESTful web API?

1. **Define the resources:** Identify the resources that your API needs to expose. A resource is anything that can be accessed or manipulated through the API. Examples of resources might include users, products, or orders.
2. **Identify the operations:** For each resource, identify the operations that clients should be able to perform. Common operations include GET (retrieve a resource), POST (create a new resource), PUT (update an existing resource), and DELETE (delete a resource).
3. **Define the URL structure:** Define a URL structure that allows clients to access the resources and operations in your API. Use a consistent and intuitive URL structure that is easy to understand and remember.
4. **Define the request and response formats:** Define the format of requests and responses for each operation. Use a standard format such as JSON or XML to ensure that clients can easily parse and generate requests and responses.
5. **Define the authentication and authorization mechanism:** Define the authentication and authorization mechanism to ensure that only authorized clients can access your API.
6. **Define error handling:** Define the error handling mechanism to return appropriate error responses in case of errors.
7. **Define the API documentation:** Document the API using tools like Swagger or OpenAPI to make it easy for clients to understand and use your API.
8. **Test the API:** Test the API using a tool like Postman to ensure that it works as expected.

11) Discuss cloud web services for IoT?

1) **Amazon Web Services (AWS):**
   - AWS provides a range of services for IoT, including **AWS IoT Core,AWS IoT Analytics, and AWS Greengrass.**
   - These services allow users to securely connect, manage, and process data from IoT devices, and build scalable IoT applications.
2) **Microsoft Azure IoT:**
   - Azure IoT provides a suite of tools and services for building and managing IoT applications, including **Azure IoT Hub, Azure IoT Edge, and Azure Stream Analytics.**
   - These services allow users to connect, manage, and process data from IoT devices, and build intelligent and scalable IoT applications.
3) **Google Cloud IoT:**
   - Google Cloud IoT provides a range of tools and services for building and managing IoT applications, including **Google Cloud IoT Core, Google Cloud IoT Edge, and Google Cloud Pub/Sub.**
   - These services allow users to securely connect, manage, and process data from IoT devices, and build scalable and intelligent IoT applications.
4) **IBM Watson IoT:**
   - IBM Watson IoT provides a suite of services for building and managing IoT applications, including **Watson IoT Platform, Watson IoT Edge, and Watson IoT Analytics.**

- These services allow users to securely connect, manage, and analyze data from IoT devices, and build scalable and intelligent IoT applications.

5) Oracle IoT:

- Oracle IoT provides a range of tools and services for building and managing IoT applications, including **Oracle IoT Cloud Service, Oracle IoT Asset Monitoring Cloud, and Oracle IoT Fleet Monitoring Cloud.**
- These services allow users to securely connect, manage, and process data from IoT devices, and build scalable and intelligent IoT applications.

12) Explain Cloud storage models?

1. Object Storage:

- Object storage is a type of storage that stores data as objects rather than files or blocks. Objects contain data along with metadata that describes the data, such as its type, size, and creation date.
- Objects are typically accessed using HTTP RESTful APIs.
- Object storage is commonly used in cloud-based applications, backup and recovery systems, and content delivery networks.
- Advantages: Object storage is highly scalable, reliable, and offers high availability.
- Disadvantages: It is less performant than block storage and have slower access times due to the additional metadata.

2. Block Storage:

- Block storage is a type of storage that stores data in fixed-sized blocks, typically accessed using a Storage Area Network (SAN) or Network Attached Storage (NAS) protocol.
- Block storage is commonly used in enterprise applications, database systems, and virtualized environments.
- Advantages: Block storage is highly performant and offers low latency access times.
- Disadvantages: it is less scalable and less cost-effective than object storage.

3. File Storage:

- File storage is a type of storage that stores data as files, typically accessed using Network File System (NFS) or Server Message Block (SMB) protocols.
- It is commonly used in content management systems, file sharing platforms, and collaborative applications.
- Advantages: File storage is easy to use and provides shared access to files across multiple users or systems. It also supports features such as file locking, file permissions, and access control.
- Disadvantages: It have slower access times and lower scalability than object storage.

4. Archive Storage:

- Archive storage is a type of storage that is designed for long-term retention of data that is rarely accessed.
- Archive storage is ideal for storing data that must be retained for regulatory purposes, such as legal documents or financial records.
- Advantages: Archive storage provides cost-effective, long-term storage with low access costs.
- Disadvantages: It may have longer access times and lower availability than other types of storage.