IOT FINAL QUESTION PAPER

Q1. Illustrate the IoT conceptual framework

Ans=>The **IoT (Internet of Things) conceptual framework** is composed of several key components that work together to enable smart systems and devices to communicate and exchange data over the internet.

1 **Devices/Things**: These are physical objects embedded with sensors, actuators, or other components that can collect data or perform actions. Examples include smart thermostats, wearable devices, and connected vehicles.

2. **Connectivity**: The devices communicate over various network protocols (e.g., Wi-Fi, Bluetooth, Zigbee, 5G) to transmit data to other devices or a central system.

3. **Edge Computing**: In some cases, processing of data can happen closer to the device at the edge of the network, reducing latency and bandwidth usage by performing preliminary data analysis locally.

4. **Data Processing**: This layer involves cloud platforms or data centers where the collected data is analyzed, processed, and stored for further use.

5. **Application Layer**: This layer encompasses the applications that utilize the processed data to deliver insights, automation, or control to the user. Examples include smart home systems, healthcare monitoring, or industrial automation.

6. **User Interface**: Users interact with IoT syst   ems through apps, dashboards, or voice commands to monitor, control, and receive notifications about the connected devices.

Q2. Discuss the LINUX significance in IoT deployment.

Ans=>1 **Open-source and Cost-effective**: Linux is open-source, which means it is free to use and modify. This makes it an affordable choice for IoT devices, which often have cost constraints.

2 **Flexibility and Customization**: Linux can be tailored to specific IoT device requirements, allowing developers to create lightweight, optimized systems that fit the performance and memory constraints of IoT devices.

3 **Wide Support and Community**: Linux has a large, active community, which provides a wealth of tools, libraries, and support. This makes it easier to develop, deploy, and troubleshoot IoT solutions.

4 **Security**: Linux offers robust security features, including user permissions and access controls, which are essential for protecting sensitive data in IoT systems.

5 **Scalability**: Linux supports a wide range of hardware platforms, from low-power microcontrollers to high-performance servers, making it scalable for diverse IoT applications.

Q3. Write a program using python for reading data from a sensor (simulated as a random number generator) and sending it to an IoT platform (simulated as printing the data).

Ans=>

```
import random

import time

def read_sensor_data():

  return random.uniform(20.0, 30.0)

def send_to_iot_platform(data):

  print(f"Sending data to IoT platform: {data}°C")

while True:

  sensor_data = read_sensor_data()  # Read sensor data

  send_to_iot_platform(sensor_data)  # Send the data

  time.sleep(2)  # Wait for 2 seconds before reading the data again
```

Q4. **Write Difference b/w   Ardiuno/ Raspberry Pi. Or explain or**

Explain the concepts involved in Raspberry Pi. Discuss in detail about Arduino with

neat sketch.

Ans=>

Feature	**Arduino**	**Raspberry** Pi
Definition	Arduino is an open-source electronics platform based on simple software and hardware for building interactive projects.	Raspberry Pi is a small, affordable computer used for various applications like coding, learning, and DIY projects.
Type	Microcontroller-based platform.	Single-board computer (SBC).
Processing Power	Low processing power, usually 8-bit or 16-bit microcontroller.	High processing power with ARM-based CPU (e.g., 64-bit).
Operating System	Does not use an operating system; it runs a simple program directly.	Runs a full operating system (e.g., Raspberry Pi OS).
Programming Language	Primarily C/C++ using Arduino IDE.	Can be programmed in various languages like Python, C++, Java.
Input/Output Pins	Has a limited number of digital and analog pins for interfacing with sensors and actuators.	Has more I/O options, but no direct analog pins (use GPIO pins with added components).
Connectivity	Limited connectivity options (USB, serial, I2C, SPI).	Full networking capabilities with Ethernet, Wi-Fi, Bluetooth, USB ports, HDMI.
Power Requirements	Low power consumption, typically powered via USB or batteries.	Higher power consumption, usually requires a 5V power supply or micro-USB.

Q5. **List the properties of constrained environments. Use examples of connected devices, such as streetlights, RFIDs, and ATMs with the Internet.**

Ans=>Constrained environments are characterized by devices with limited resources, such as low processing power, limited memory, and energy constraints.

1. Limited Processing Power

- Devices in constrained environments often use low-power processors designed for specific tasks.
- **Example**:
    - **Streetlights**: Smart streetlights may use microcontrollers with minimal computing power to process light sensor data and control lighting intensity.
    - **RFIDs**: Operate on tiny chips with minimal processing power to manage identification data.
    - **ATMs**: Use specialized processors optimized for secure transactions rather than general computing.

2. Low Energy Availability

- Constrained devices often run on batteries or low-power sources, necessitating energy-efficient operation.
- **Example**:
    - **Streetlights**: Solar-powered units store limited energy, requiring efficient use for sensors and communication.
    - **RFIDs**: Passive RFIDs draw energy from nearby RFID readers to operate, limiting their functionality.
    - **ATMs**: Must maintain operation during power outages using backup power systems, limiting non-essential processes.

3. Limited Memory

- Constrained devices have limited RAM and storage capacity, focusing only on essential data processing and storage.
- **Example**:
    - **Streetlights**: Only store operational settings and minimal sensor data.
    - **RFIDs**: Store unique identification data and perhaps a small amount of metadata.
    - **ATMs**: Retain transactional logs and operational firmware but not extensive databases.

4. **Cost Sensitivity**

- Devices in constrained environments are designed to be cost-effective for mass deployment.
- **Example**:

- **Streetlights**: Use low-cost sensors and controllers for widespread installation.
- **RFIDs**: Are manufactured at a low cost for scalability in supply chains.
- **ATMs**: Use optimized hardware to manage costs while ensuring reliability.

Q6. Explain about the privacy and vulnerabilities of IoT. What are the security requirements and threat analysis in IoT?

Ans=>**Privacy and Vulnerabilities in IoT:**

The **Internet of Things (IoT)** connects devices like smart appliances, sensors, and cameras to the internet. These devices often collect sensitive data, like personal details, health records, or location, which can lead to **privacy issues** if the data is not properly secured. Common **vulnerabilities** include weak passwords, unencrypted data, and insecure networks, which make IoT devices easy targets for hackers.

Security Requirements in IoT:

1. **Data Encryption:** Protect data during transmission and storage.
2. **Authentication:** Ensure only authorized users can access devices.
3. **Regular Updates:** Patch security flaws in IoT devices.
4. **Secure Network Communication:** Use strong protocols like HTTPS.

Threat Analysis in IoT:

IoT faces threats like:

1. **Hacking:** Unauthorized access to devices.
2. **Data Breaches:** Leaking sensitive data.
3. **Botnets:** Compromised IoT devices used in cyberattacks.
4. **Denial of Service (DoS):** Overloading devices to disrupt their functions.

Q7. Analyse in detail the architectural components of IOT and M2M architecture.

Ans=>M2M architecture focuses on direct communication between devices without significant human intervention. It typically involves the following **components**:

Devices:

- **Sensors:** These devices collect data from the physical world, such as temperature, humidity, pressure, or motion

Connectivity:

- **Networks:** M2M devices often use various network technologies, including cellular networks (2G, 3G, 4G, 5G), Wi-Fi, Bluetooth, Zigbee, and LoRaWAN , and transmit it to a central system or cloud platform.

Data Processing:

- **Data Collection:** Data is gathered from devices and transmitted to a central server or cloud platform. Collected data is stored for analysis and future reference. Data is processed to extract meaningful insights and trigger actions.

Cloud Servers:

The cloud is often used to store and analyze large volumes of data collected by IoT devices. It can also host applications and services that offer higher-level analysis, such as machine learning, big data processing, and real-time data streaming

Applications:

M2M applications are designed to automate tasks and improve efficiency, such as remote monitoring, predictive maintenance, and supply chain optimization.

IoT Architecture or List the major components of an IoT system. How do these components work

together Lo achieve the goals of IOT?

A typical IoT architecture consists of the following components:

1. **Devices:**
   - **Sensors and Actuators:** Similar to M2M, IoT devices include sensors and actuators to collect and control data.
2. **Connectivity:**
   - **Networks:** IoT devices utilize a wide range of network technologies, including cellular networks, Wi-Fi, Bluetooth, Zigbee, LoRaWAN, and satellite communication. Gateways aggregate data from multiple devices and transmit it to the cloud or other network systems.
3. **Data Processing:**
   - **Data Collection:** Data is collected from devices and transmitted to a cloud platform or data center. Data is stored in various formats, such as time-series databases, NoSQL databases, or data lakes.Data is processed using advanced analytics techniques, including machine learning and artificial intelligence, to extract valuable insights.
4. **Cloud Servers:**
   - The cloud is often used to store and analyze large volumes of data collected by IoT devices. It can also host applications and services that offer higher-level analysis, such as machine learning, big data processing, and real-time data streaming.
5. **Applications and Services:**
   - **IoT applications** are diverse and cover a wide range of domains, including smart homes, smart cities, healthcare, agriculture, and manufacturing.
   - **User Interfaces:** User interfaces enable users to interact with IoT devices and applications, such as mobile apps, web portals, and voice assistants.

NewPaper

Q1. Name the Need For sensors in IoT.

Ans=> 1 **Real-world data acquisition:** Sensors capture data from the physical environment, enabling IoT systems to react to changes in real-time.

2 **Remote monitoring and control:** Sensors allow remote monitoring of various parameters, enabling remote control and decision-making.

3 **Automation and efficiency:** Sensors can trigger automated actions based on predefined conditions, improving efficiency and reducing human intervention.

4 **Data-driven insights:** Sensor data can be analyzed to extract valuable insights and optimize processes.

5 **Enhanced user experience:** Sensors can personalize user experiences by adapting to their preferences and surroundings.

Q2. **Describe Logical design using python in detail**

Ans=> In the context of the Internet of Things (IoT), logical design refers to the structure and organization of data flow, device interactions, and system processes that enable communication, control, and data processing in an IoT system. The logical design primarily focuses on defining the architecture of the system, which consists of various IoT devices, sensors, actuators, and cloud services.

1.Sensor Data Collection: IoT devices collect data from sensors (e.g., temperature, humidity, motion sensors). Python can be used to interact with these sensors via libraries like RPi.GPIO for Raspberry Pi or pySerial for Arduino to gather real-time data.

2.Data Transmission: The collected data is transmitted to a central server or cloud platform. This can be done using protocols like MQTT, HTTP, or CoAP. Python's paho-mqtt library, for example, can be used to send data over MQTT.

3.Data Processing: On the server/cloud side, Python can process and analyze the collected data. For instance, the data could be stored in a database like MySQL, MongoDB, or sent for analysis using libraries like Pandas or NumPy.

4.Device Control: Based on the processed data, Python can also be used to send commands to actuators (e.g., turning on a fan or light). This is achieved through communication protocols like MQTT, HTTP, or direct GPIO control on devices.

Q3. **Explain the deployment and operational view, resources, services, virtual entities, users in an IoT system by considering a Parking lot example.**

In an IoT system for a parking lot, here's a breakdown of the key elements:

1. **Deployment and Operational View**: This represents how the IoT devices (like sensors, cameras, and gateways) are physically placed in the parking lot and how they interact to collect, process, and transmit data. For example, sensors are deployed in each parking spot to detect whether it's occupied or free.

2. **Resources**: These are the physical or virtual components required for the IoT system, such as parking sensors, cameras, edge devices, cloud storage, and servers. Resources also include network infrastructure for communication.
3. **Services**: These are the functionalities provided by the IoT system, like real-time parking space availability, automated payment systems, and notifications for users. They are delivered through software or cloud platforms.
4. **Virtual Entities**: These are digital representations of physical components in the IoT system. For instance, each parking spot can be a virtual entity, which gets updated in real-time based on sensor data (whether the spot is occupied or free).
5. **Users**: These are the people or systems interacting with the IoT system. In the parking lot example, users could include drivers searching for available parking spots, parking lot operators managing the lot, or even maintenance teams monitoring system health.

Q4.

a) Define clustering. Summarize the function of Action Prediction model. Identify the

purpose of Data Preprocessing.

b) When the data is called as Week Type Data? What is meant by predictive analysis?

List out the various phases of CRISP-DM model and explain each with diagram.

Ans=>

1.**Clustering** is the process of grouping a set of objects in such a way that objects in the same group (or cluster) are more similar to each other than to those in other groups. It is often used in data analysis to identify patterns and relationships within data.

2.The Action Prediction model is designed to predict future actions or behaviors based on historical data. It utilizes various algorithms and techniques to analyze past events and make informed predictions about what might happen next. This model is commonly applied in fields such as marketing, finance, and user behavior analysis to enhance decision-making processes.

Data Preprocessing serves several critical purposes in the context of machine learning and data analysis:

- **Data Cleaning**: It involves identifying and correcting errors or inconsistencies in the dataset, such as handling missing values, removing duplicates, and addressing outliers.
- **Data Transformation**: This includes scaling features, encoding categorical variables, and normalizing data to ensure that all features contribute equally to the analysis.
- **Improving Model Performance**: Proper preprocessing enhances the quality of the data, which directly impacts the performance and accuracy of machine learning models.
- **Facilitating Analysis**: By transforming raw data into a more suitable format, preprocessing makes it easier to extract meaningful insights and patterns from the data.
- **Ensuring Consistency**: It helps maintain a consistent format across the dataset, which is essential for effective analysis and model training.

B) Ans=>Week Type Data typically refers to data that is categorized based on the type of week, such as weekdays versus weekends, or specific weeks in a month or year. This classification can help in analyzing patterns or trends that vary by week type.

Predictive Analysis: Predictive analysis is a branch of data analytics that uses statistical algorithms, machine learning techniques, and historical data to make predictions about future events or trends. The goal is to identify patterns in the data that can help predict outcomes.

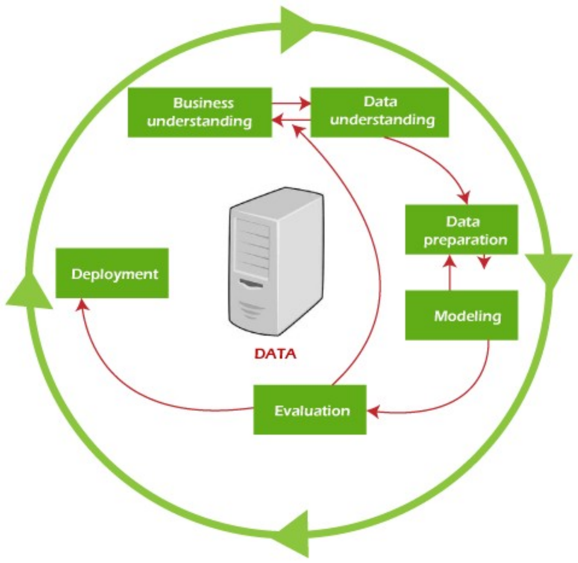
For example:

- **Sales Forecasting**: Predicting future sales based on past sales data
- **Customer Behavior**: Predicting which products a customer may purchase based on past behavior.
- **Demand Forecasting**: Predicting demand for products or services during certain periods, such as holidays.

CRISP-DM stands for the cross-industry standard process for data mining. The CRISP-DM methodology provides a structured approach to planning a data mining project. It is a robust and well-proven methodology. We do not claim any ownership over it.

various phases of CRISP-DM model

It includes descriptions of typical phases of a project, the tasks involved with each phase, and an explanation of the relationships between these tasks.



Phase 1: Business Understanding:

- This is the first phase where you define the objectives of the project. You must understand the problem you're trying to solve and translate it into a data mining problem. You also decide on the success criteria.

Example: In a retail company, the goal could be to predict customer churn (who will stop buying).

Phase 2: Data Understanding:

- This phase involves collecting and exploring data to get familiar with it. It includes understanding the dataset, identifying missing values, and detecting outliers.

Example: Collect data about customers, their purchases, and demographics.

Phase 3: Data Preparation:

- In this phase, you clean and prepare the data for analysis. This may involve dealing with missing data, removing duplicates, or transforming data into the required format.

Example: You might remove rows with missing customer age or format the data to work with machine learning models.

Phase 4: Modeling:

- Here, you apply different data mining techniques (such as classification, regression, or clustering) to build models based on the prepared data.

Example: You could use a decision tree algorithm to predict whether a customer will churn.

Phase 5: Evaluation:

- After building the model, you assess its performance and check if it meets the original business objectives. If not, you might need to adjust the model.

Example: You check if your model's accuracy is good enough to be used for real-world predictions.

Phase 6: Deployment:

- In this final phase, the model is deployed into production, where it can be used to make decisions or predictions. It could also involve creating reports or visualizations for the business.

Example: The churn prediction model is integrated into the company's system to notify when a customer is at risk of leaving.

Newpaper

Q1. **Compare and contrast Machine-to-Machine (M2M) communication with IoT**.

Ans=>

Definition	Direct communication between machines or devices without human intervention.	A network of interconnected devices that communicate over the internet.
Connectivity	Typically relies on point-to-point communication, often via cellular or wired networks.	Uses IP-based networks, including Wi-Fi, cellular, Bluetooth, and Ethernet.
Scope	Limited to specific devices or applications.	Broader scope, integrating diverse devices and applications.
Data Handling	Limited data storage and processing, often localized.	Extensive data collection, storage, and cloud-based processing.

Scalability	Less scalable due to reliance on fixed infrastructure.	Highly scalable, supporting millions of devices globally.
Cost	Higher setup and maintenance costs for specific use cases.	Cost-efficient due to shared infrastructure and cloud services.
Cloud Integration	Rarely integrated with cloud services.	Heavily relies on cloud platforms for data storage and analytics.
Application Example	Smart meters, ATM communication, industrial equipment monitoring.	Smart homes, wearable devices, autonomous vehicles, smart cities.
Human Interaction	Generally requires no human interaction.	Can involve humans for monitoring, control, and decision-making.

Q2. **Provide three real-world examples of IoT applications in different sectors. Or Present examples of how IoT is transforming various industries.**

Ans=> **1. Healthcare Sector:** IoT devices such as wearable health monitors (e.g., Fitbit, Apple Watch, or specialized medical devices) track vital signs like heart rate, blood pressure, and oxygen levels in real-time. Data is transmitted to healthcare providers for early diagnosis, chronic disease management, or emergency alerts, enhancing patient care while reducing hospital visits.

2. Agriculture Sector: IoT sensors monitor soil moisture, weather conditions, and crop health. Based on the collected data, these systems automate irrigation, ensuring crops receive optimal water levels. This reduces water waste and boosts crop yields, helping farmers maximize efficiency and sustainability.

3. Transportation Sector: IoT-enabled traffic lights and sensors collect real-time traffic data to optimize signal timings, reduce congestion, and provide alternate route suggestions. Connected vehicles also communicate with these systems for improved navigation and safety, leading to more efficient urban mobility.

4. Environmental Monitoring Sector: IoT sensors deployed in urban areas measure pollutants like CO2, PM2.5, and ozone. Real-time data helps governments and organizations develop strategies to combat air pollution and improve public health.

5. Logistics and Supply Chain Sector:
IoT-enabled GPS devices and sensors track vehicle locations, fuel consumption, and driver behavior. This ensures timely deliveries, reduces fuel costs, and enhances overall fleet efficiency.

Q3. **Set up an IOT device (e.g., Raspberry Pi with sensors).**

Ans=> **Components Needed**

1. **Raspberry Pi** (any model, but Raspberry Pi 3 or 4 is recommended)
2. **MicroSD Card** (at least 16GB, Class 10 recommended)
3. **Power Supply** (appropriate for your Raspberry Pi model)
4. **Sensors** (e.g., DHT11/DHT22 for temperature and humidity, HC-SR04 for distance measurement, etc.)

5. **Breadboard and Jumper Wires** (for prototyping)
6. **Internet Connection** (Wi-Fi or Ethernet)
7. **Optional: Case for Raspberry Pi**

Step 1: Set Up the Raspberry Pi

1. **Download Raspberry Pi OS:**
    - Go to the Raspberry Pi website and download the Raspberry Pi Imager.
    - Use the Imager to write the Raspberry Pi OS onto the microSD card.
2. **Insert the MicroSD Card:**
    - Once the OS is installed, insert the microSD card into your Raspberry Pi.
3. **Connect Peripherals:**
    - Connect a keyboard, mouse, and monitor to the Raspberry Pi.
4. **Power Up:**
    - Connect the power supply to the Raspberry Pi and turn it on.
5. **Initial Configuration:**
    - Follow the on-screen instructions to set up the Raspberry Pi (language, timezone, Wi-Fi, etc.).
    - Update the system using:

```
sudo apt update

sudo apt upgrade
```

Step 2: Connect Your Sensors

1. **Wiring:**
    - Connect your sensors to the Raspberry Pi GPIO pins using jumper wires. Refer to the datasheet for each sensor for the correct pin configuration.
    - Here's an example for the DHT11 sensor:
        - VCC to 3.3V
        - GND to Ground
        - Data pin to a GPIO pin (e.g., GPIO4)
2. **Breadboard Setup:**
    - Use a breadboard for easier connections if needed.

Step 3: Install Required Libraries

1. **Install Python and Pip (if not already installed):**

```
sudo apt install python3 python3-pip
```

2. **Install Sensor Libraries:**
    - For DHT sensors, you can install the Adafruit DHT library:

```
sudo pip3 install Adafruit-DHT
```

3. **Install GPIO Library:**

```
sudo apt install python3-rpi.gpio
```

Step 4: Write the Code

1. **Create a Python Script:**
    - Open a terminal and create a new Python file:

```
nano sensor_read.py
```

2. **Write the Code:**
    - Here is a simple example for reading temperature and humidity from a DHT11 sensor:

```
import Adafruit_DHT
import time
sensor = Adafruit_DHT.DHT11
pin = 4  # GPIO pin number
while True:
    humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
    # Check if the reading was successful
   if humidity is not None and temperature is not None:
      print(f'Temperature: {temperature}°C  Humidity: {humidity}%')
    else:
       print('Failed to retrieve data from sensor')
    time.sleep(2)
```

3. **Save and Exit:**
    - Press **CTRL + X**, then **Y**, then **Enter** to save and exit.

Step 5: Run Your Code

1. **Run the Script:**

```
python3 sensor_read.py
```

2. **Monitor Output:**
    - You should see temperature and humidity readings in the terminal.

Q4. **Develop a use and misuse case scenario for a smart home IoT system.**

Ans=> **Use Case Scenario: Smart Home IoT System**

A **use case** highlights a beneficial and intended interaction with the system.

Scenario:

A homeowner uses their smart home IoT system to control home appliances remotely via a mobile app. They turn on the air conditioner 30 minutes before arriving home to ensure the house is cool and comfortable. The system also uses motion sensors to detect if anyone is in a room and automatically adjusts the lighting and temperature, reducing energy consumption.

Misuse Case Scenario: Smart Home IoT System

A **misuse case** illustrates how the system can be exploited or used inappropriately.

Scenario:

An unauthorized person hacks into the IoT system due to weak password protection. They gain access to control the security cameras and locks, enabling a burglary while the homeowner is away.

Q5. Give two examples of smart devices and explain how they connect to and interact within the IoT network

Ans=>

1. Smart Apple Watch

- **Connection to IoT Network**: The Smart Apple Watch connects to the Internet via Wi-Fi or cellular networks, and it syncs with other Apple devices (iPhone, iPad, Mac) through Bluetooth. The watch can also connect to third-party applications via APIs, sharing data such as health metrics, location, and notifications.
- **Interaction within the IoT Network**: The Apple Watch interacts with other IoT devices like smart home appliances (e.g., controlling lights, thermostats), fitness trackers, or even health monitoring systems. It can send data such as heart rate, steps, and sleep patterns to cloud-based applications, which can then analyze and provide insights. It also receives notifications, calls, and messages, allowing users to interact with the watch seamlessly.

2. Smart Umbrella

- **Connection to IoT Network**: A Smart Umbrella connects to the IoT network via Bluetooth or Wi-Fi to a user's smartphone. It may include sensors like GPS, temperature, or humidity sensors to detect weather conditions.
- **Interaction within the IoT Network**: The umbrella interacts with weather apps or smart home systems to notify the user when it's about to rain. It may sync with a smartphone app that tracks its location and sends alerts if the umbrella is left behind. Additionally, some smart umbrellas can update weather data to cloud services, helping users plan their day based on real-time weather forecasts.

Newpaper

Q1. **what is vulnerabilities List common vulnerabilities associated with IoT devices.**

Ans=>IoT (Internet of Things) devices are often vulnerable to various security threats due to their interconnected nature, lack of robust security protocols, and diverse use cases. Here is a list of common vulnerabilities associated with IoT devices:

1. Weak Authentication and Authorization

- Many IoT devices use weak or hardcoded default passwords, making it easier for attackers to gain unauthorized access.

2. Insecure Communication

- IoT devices may transmit data over unencrypted communication channels, exposing sensitive information to eavesdropping or man-in-the-middle attacks.

3. Lack of Regular Software Updates

- Many IoT devices have limited or no mechanism to receive regular security patches and software updates, making them susceptible to known vulnerabilities.

4. Insecure Web Interfaces

- Some IoT devices have poorly designed web interfaces that are vulnerable to attacks like cross-site scripting (XSS), cross-site request forgery (CSRF), and SQL injection, allowing attackers to execute malicious code.

6. Lack of Privacy Protections

- Many IoT devices collect personal data but do not implement strong privacy protections or clear data usage policies, exposing users to potential privacy breaches.

7. Insecure APIs

- APIs used by IoT devices may lack proper authentication and security controls, making it easier for attackers to exploit vulnerabilities and gain unauthorized access to the device or its data.

Q2. **Compare and contrast wired and wireless communication technologies in the context of IoT/M2M Systems.**

Feature	**Wired Communication**	**Wireless Communication**
Connection Type	Physical cables (Ethernet, coaxial cables, fiber optic)	Radio waves (Wi-Fi, Bluetooth, Zigbee, LoRa, cellular, etc.)
Setup and Deployment	Requires physical installation and infrastructure	Easier deployment, no need for physical connections
Mobility	Static, not suitable for mobile devices	Highly flexible and supports mobile devices
Speed and Bandwidth	High-speed data transfer (Fiber Optic, Ethernet)	Varies; Wi-Fi offers high speed, while other wireless options may offer lower speed

Feature	**Wired Communication**	**Wireless Communication**
Reliability	More reliable and stable (less interference, physical connection)	Prone to interference, signal degradation, and range limitations
Power Consumption	Generally lower power consumption as devices are connected via cable	Wireless devices can consume more power, especially for long-range communication
Cost	Higher initial cost due to physical infrastructure	Lower initial cost for deployment but may have ongoing subscription or data costs
Security	Generally more secure due to physical connection	Security can be a concern due to potential vulnerabilities in wireless networks
Maintenance	Requires manual maintenance for cables and physical hardware	Easier to maintain; requires less physical intervention but may need software or network management
Range	Limited by cable length (e.g., Ethernet, fiber optic)	Varies based on the technology (Wi-Fi, Bluetooth have limited range, while LoRa can have longer range)
Scalability	Difficult to scale due to the need for additional cables and hardware	More scalable as additional devices can be added without the need for extensive cabling
Flexibility	Limited flexibility; requires re-wiring for changes in layout	Highly flexible; devices can be moved without the need to reconfigure infrastructure

Q3. **In what scenarios would you prefer one over the other for web connectivity? JSON vs. Tag Length Value (TLV)**

Ans=>**Prefer JSON when:**

- **Web APIs**: JSON is a standard for data exchange over the web due to its human-readable format and ease of integration with web technologies.

Prefer TLV when:

- **Performance-sensitive applications**: You need a more compact, efficient format, especially in low-level communication protocols (e.g., embedded systems, smart cards).

In conclusion, **JSON** is typically preferred for web connectivity and APIs due to its human-readable format, ease of use, and wide support. However, **TLV** is a better choice in systems that require compact, efficient, and binary data transmission, especially in low-level protocols or hardware-related applications.

Q4. **Name two common application layer protocols used on the internet. Explain the role of the Hypertext Transfer Protocol (HTTP) in the application layer.**

Ans=>Two common application layer protocols used on the internet are:

1. **Hypertext Transfer Protocol (HTTP)**: HTTP is the primary protocol used for transferring web pages and other resources on the World Wide Web. It defines the communication rules between a client (e.g., web browser) and a server. HTTP facilitates the request and response mechanism, allowing users to access websites, fetch data, and interact with online content.
2. **File Transfer Protocol (FTP)**: FTP is used to transfer files between a client and a server. It enables users to upload and download files over the internet. FTP is often used for managing files on remote servers, sharing large files, and accessing directories.

Role of HTTP in the Application Layer:

The **Hypertext Transfer Protocol (HTTP)** operates at the application layer in the OSI model, and its role is to enable communication between web browsers (clients) and web servers. HTTP governs how requests are made for web resources and how servers respond with those resources. It defines a client-server communication model where clients request web pages and servers provide the requested content. HTTP is stateless, meaning each request is independent, and no information about previous requests is retained. It uses methods like GET, POST, PUT, and DELETE to define actions to be performed on resources. Overall, HTTP is the fundamental protocol that supports the operation of the web by enabling the transfer of hypertext and multimedia content across the internet.

Q5. **Share a brief summary of one real-world case study where IoT technology has been**

applied in Smart Homes.

Ans=>A real-world case study where IoT technology has been applied in smart homes is the use of **Google Nest**. Google Nest is a smart home system that integrates various IoT devices to enhance convenience, security, and energy efficiency in homes.

Summary:

Google Nest includes devices like smart thermostats, cameras, doorbells, and smoke detectors, all connected to the internet. The **Nest Thermostat** allows homeowners to remotely control the temperature of their home via a mobile app or voice commands, optimizing energy consumption by learning the user's preferences and adjusting the heating/cooling accordingly. The **Nest Cam** and **Nest Hello** doorbell use IoT sensors to monitor the home for security threats, sending real-time alerts to users' smartphones if motion or unusual activity is detected. Additionally, the **Nest Protect** smoke detector provides early warnings and can be silenced remotely through the app.

Q6. **List the different libraries used in Python. Write the following to create a code on**

Arduino/Raspberry Pi:

a) To turn ON/OFF LED for specific duration(LED/ Buzzer).

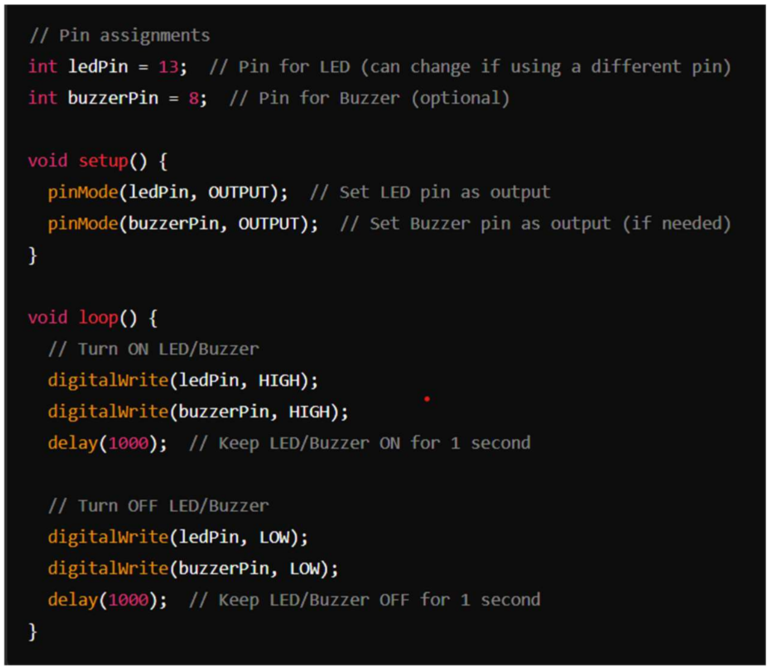
b) To print temperature and humidity readings (DHTII/ DHT22 sensor).

Ans=>**Common Python Libraries**

1. **NumPy**: For numerical operations and array handling.
2. **Pandas**: For data manipulation and analysis.
3. **Matplotlib**: For plotting and visualization.

4. **Requests**: For making HTTP requests.
5. **Flask**: For web development.
6. **TensorFlow/PyTorch**: For machine learning.
7. **OpenCV**: For computer vision.
8. **DHT**: For interfacing with DHT sensors.
9. **RPi.GPIO**: For controlling GPIO pins on Raspberry Pi.
10. **pySerial**: For serial communication.

    a) Code to Turn ON/OFF LED for Specific Duration (Arduino):



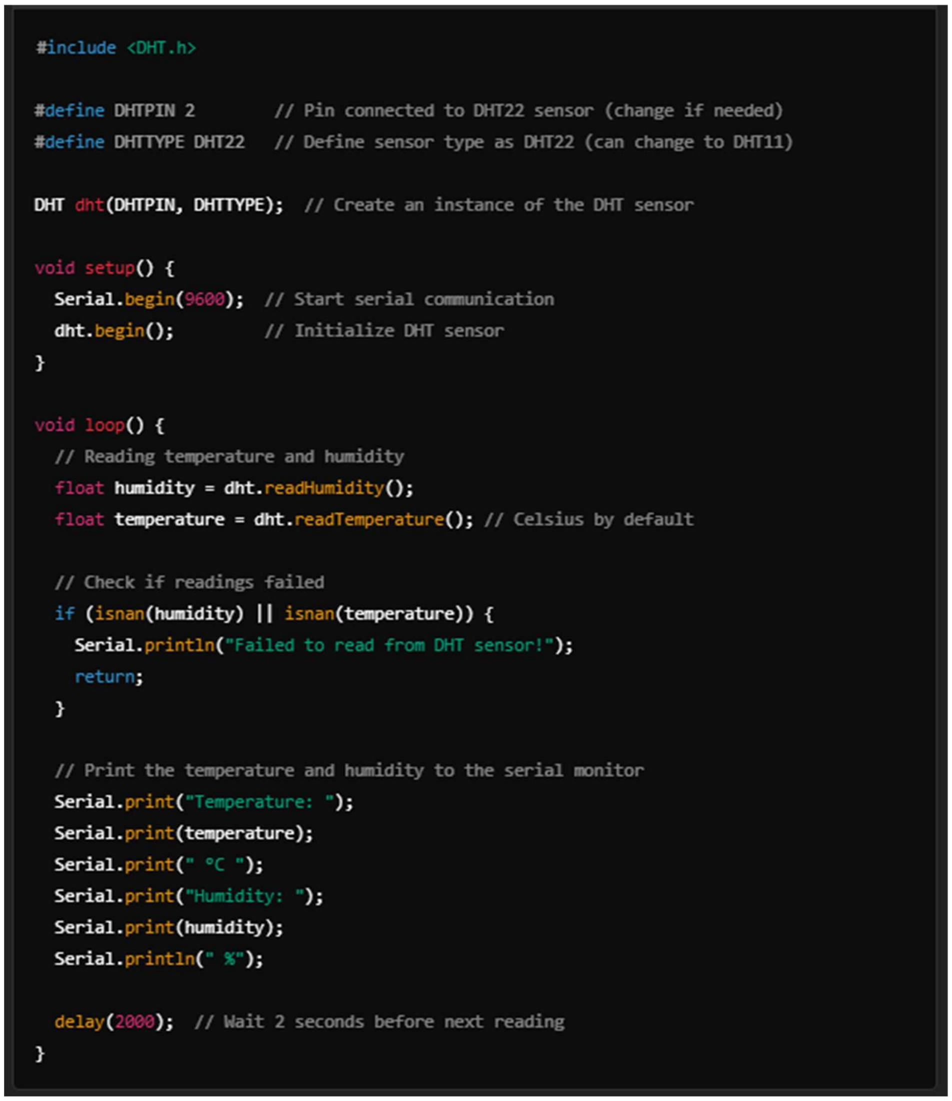
```
// Pin assignments
int ledPin = 13;  // Pin for LED (can change if using a different pin)
int buzzerPin = 8;  // Pin for Buzzer (optional)

void setup() {
  pinMode(ledPin, OUTPUT);  // Set LED pin as output
  pinMode(buzzerPin, OUTPUT);  // Set Buzzer pin as output (if needed)
}

void loop() {
  // Turn ON LED/Buzzer
  digitalWrite(ledPin, HIGH);
  digitalWrite(buzzerPin, HIGH);
  delay(1000);  // Keep LED/Buzzer ON for 1 second

  // Turn OFF LED/Buzzer
  digitalWrite(ledPin, LOW);
  digitalWrite(buzzerPin, LOW);
  delay(1000);  // Keep LED/Buzzer OFF for 1 second
}
```

b) Code to Print Temperature and Humidity Readings (DHT11/DHT22 Sensor):



```
#include <DHT.h>

#define DHTPIN 2        // Pin connected to DHT22 sensor (change if needed)
#define DHTTYPE DHT22   // Define sensor type as DHT22 (can change to DHT11)

DHT dht(DHTPIN, DHTTYPE);  // Create an instance of the DHT sensor

void setup() {
  Serial.begin(9600);  // Start serial communication
  dht.begin();         // Initialize DHT sensor
}

void loop() {
  // Reading temperature and humidity
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature(); // Celsius by default

  // Check if readings failed
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Print the temperature and humidity to the serial monitor
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print(" °C ");
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.println(" %");

  delay(2000);  // Wait 2 seconds before next reading
}
```