

UNIT 1: Basic Electronics, Sensors, and Measurement

Q1. What are different types of sensors in terms of the energy converted and the type of output produced? Explain each type with proper examples.

Sensors are devices that sense physical quantities such as temperature, pressure, light, or motion and convert them into electrical signals. Sensors are mainly classified based on **energy conversion** and **type of output produced**.

1. Classification Based on Energy Conversion

a) Active Sensors

Active sensors generate electrical signals by themselves without requiring an external power source. The output is usually in the form of voltage or current.

Examples:

- **Thermocouple:** Produces voltage due to temperature difference.
- **Piezoelectric sensor:** Generates voltage when mechanical pressure is applied.
- **Photovoltaic cell:** Converts light energy into electrical energy.

Applications: Temperature sensing, vibration monitoring, energy harvesting.

b) Passive Sensors

Passive sensors do not generate electrical signals on their own. They require an external power supply and work by changing resistance, capacitance, or inductance.

Examples:

- **Thermistor:** Resistance changes with temperature.
- **LDR:** Resistance changes with light intensity.
- **Potentiometer:** Resistance varies with position.

Applications: Light measurement, displacement sensing, temperature measurement.

2. Classification Based on Output Signal

a) Analog Sensors

Produce continuous output signals proportional to the measured quantity.

Examples: LM35 temperature sensor, potentiometer.

b) Digital Sensors

Produce discrete digital outputs (0 or 1).

Examples: Ultrasonic sensor, DHT sensor.

Q2. Classify five common electronic components used in IoT (e.g., resistors, capacitors, diodes, transistors, ICs) and explain their functions.

Electronic components form the foundation of IoT hardware systems.

1. Resistor

- Limits current flow in circuits.
- Protects components such as LEDs.
- Used in voltage divider circuits.

2. Capacitor

- Stores electrical energy.
- Smooths voltage and filters noise.
- Used in power supply circuits.

3. Diode

- Allows current flow in only one direction.
- Used for rectification and circuit protection.

4. Transistor

- Acts as a switch or amplifier.
- Controls motors, relays, and high-power devices.

5. Integrated Circuit (IC)

- Contains multiple electronic components on a single chip.
 - Performs complex tasks such as processing, timing, and memory storage.
-

Q3. What is Ohm's Law? Explain the relationship between voltage and current in an electric circuit.

Ohm's Law states that the current flowing through a conductor is directly proportional to the voltage applied across it, provided temperature remains constant.

$$V = I \times R$$

Where:

- **V** = Voltage (Volts)
- **I** = Current (Amperes)
- **R** = Resistance (Ohms)

Explanation

- Increasing voltage increases current.
- Increasing resistance decreases current.

Example

For a $10\ \Omega$ resistor connected to 5 V:

$$I = \frac{5}{10} = 0.5A$$

Ohm's Law is essential for designing safe electronic and IoT circuits.

Q4. Why is it necessary to use a proper resistor to control an LED with Arduino?

Arduino digital pins provide fixed voltage (5V or 3.3V). LEDs require limited current to operate safely.

Importance of Resistor

- Limits current through the LED.
- Prevents LED damage.
- Protects Arduino GPIO pins from overcurrent.

Conclusion

Without a resistor, excessive current may flow, damaging the LED or microcontroller. Hence, resistors are essential for safe LED operation.

Q5. Describe series and parallel configuration of resistors. Explain how to calculate resultant resistance in each case.

Series Configuration

- Resistors connected end-to-end.
- Same current flows through all resistors.

$$R_T = R_1 + R_2 + R_3$$

Use: Current limiting applications.

Parallel Configuration

- Resistors connected across the same voltage points.
- Voltage across each resistor is equal.

$$\frac{1}{R_T} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}$$

Use: Reliability and current sharing.

Q6. Describe series and parallel configuration of batteries and determine resultant voltage in each case.

Series Battery Connection

- Positive terminal connected to negative terminal.
- Voltages add up.

Example: Two 1.5V batteries → 3V output

Parallel Battery Connection

- Positive terminals connected together.
- Voltage remains the same.
- Current capacity increases.

Example: Two 1.5V batteries → 1.5V output with longer backup.

Q7. For a resistor network consisting of $10\ \Omega$, $20\ \Omega$, and $30\ \Omega$ resistors in series, calculate the total resistance and current when connected across a 12 V battery.

Given:

- $R_1 = 10\ \Omega$, $R_2 = 20\ \Omega$, $R_3 = 30\ \Omega$
- Voltage = 12 V

Total Resistance

$$R_T = 10 + 20 + 30 = 60\Omega$$

Current

$$I = \frac{12}{60} = 0.2A$$

Answer:

- Total resistance = **$60\ \Omega$**
 - Current = **0.2 A**
-

Q8. Explain the voltage divider circuit. Describe its applications in sensor interfacing.

A voltage divider circuit divides an input voltage into a smaller output voltage using two resistors in series.

$$V_{out} = V_{in} \times \frac{R_2}{R_1 + R_2}$$

Applications

- Interfacing LDR and thermistors.
 - Reducing high voltage for Arduino ADC.
 - Reference voltage generation.
-

Q9. Differentiate between digital sensors and analog sensors with examples.

Feature	Analog Sensors	Digital Sensors
Output	Continuous	Discrete
Noise sensitivity	High	Low
Interfacing	Needs ADC	Direct
Examples	LM35, LDR	Ultrasonic, DHT

Q10. Why is LM35 considered a smart temperature sensor? What advantages does it provide compared to a thermistor?

LM35 is a semiconductor temperature sensor with built-in signal conditioning.

Advantages

- Linear output (10 mV/°C)
- High accuracy
- No calibration required
- Easy Arduino interfacing

Compared to Thermistor

- Better linearity
 - Stable output
 - Less complex circuitry
-

Q11. Explain working principles and output characteristics of: Ultrasonic sensor, Potentiometer, LM35 sensor, DHT temperature-humidity sensor.

Ultrasonic Sensor

- Uses ultrasonic sound waves.
- Measures distance using echo time.
- Digital output.

Potentiometer

- Variable resistor.
- Produces analog voltage.

LM35 Sensor

- Semiconductor temperature sensor.
- Linear analog output.

DHT Sensor

- Measures temperature and humidity.
- Digital output using single-wire protocol.

UNIT 2: Microcontrollers, Arduino/ESP32, Interfacing and Actuators

Q1. How do you interface ultrasonic sensor, potentiometer, LM35, and DHT sensor with Arduino? Describe using block diagrams and appropriate conversion/mapping equations.

Interfacing sensors with Arduino involves connecting sensor outputs to appropriate Arduino pins and processing the signals using software.

1. Ultrasonic Sensor (HC-SR04)

Connections:

- VCC → 5V
- GND → GND
- TRIG → Digital Output pin
- ECHO → Digital Input pin

Working:

- Arduino sends trigger pulse.
- Sensor emits ultrasonic waves.
- Echo time is measured to calculate distance.

Equation:

Distance

$$= \frac{\text{Time} \times \text{Speed}}{2}$$

f S

2. Potentiometer

Connections:

- One end → 5V
- Other end → GND
- Middle pin → Analog input (A0)

Working:

- Acts as voltage divider.
- Produces analog voltage (0–5V).

Mapping Equation:

Mapped V

$$= \frac{ADC\ V}{1023} \times Output\ R$$

alue

alue

ange

3. LM35 Temperature Sensor

Connections:

- VCC → 5V
- GND → GND
- Output → Analog pin (A1)

Conversion Equation:

$$Temperature(^{\circ}C) = ADC\ V \times \frac{5}{1023} \times 100$$

4. DHT Sensor

Connections:

- VCC → 5V
- GND → GND
- Data → Digital pin

Working:

- Measures temperature and humidity.
- Sends digital data using single-wire protocol.

Q2. How do you interface a servo motor and LED with Arduino? Explain their working using block diagrams.

1. LED Interfacing

Connections:

- Digital pin → Resistor → LED → GND

Working:

- HIGH → LED ON

- LOW → LED OFF

Purpose: Status indication, alerts.

2. Servo Motor Interfacing

Connections:

- Red → 5V
- Brown/Black → GND
- Yellow → PWM pin

Working:

- Controlled using PWM signals.
- Pulse width defines angle (0°–180°).

Library Used: Servo.h

Q3. What are the working principles and use cases of digital and analog pins in Arduino? Explain with examples of interfaced sensors.

Analog Pins

- Read continuous voltage (0–5V).
- Use ADC (10-bit → 0–1023).

Examples:

- LM35 temperature sensor
 - Potentiometer
 - LDR
-

Digital Pins

- Operate in HIGH or LOW states.
- Used for control and digital communication.

Examples:

- Ultrasonic sensor (TRIG/ECHO)
- DHT sensor
- LED, Relay

Q4. Compare Arduino and ESP32 on the basis of features, capabilities and example use cases.

Feature	Arduino UNO ESP32	
Clock speed	16 MHz	Up to 240 MHz
Wi-Fi	No	Yes
Bluetooth	No	Yes
GPIO pins	14	34
Operating voltage	5V	3.3V
Use case	Simple IoT	Advanced IoT

Conclusion: ESP32 is better for wireless and cloud-based IoT applications.

Q5. When is a relay used in an embedded system?

A relay is used when:

- Low-voltage microcontroller controls high-voltage devices.
- Electrical isolation is required.

Applications:

- Switching AC appliances
- Motor control
- Smart home automation

Q6. Can you directly run a DC motor from an Arduino digital output pin? Justify your answer.

No, you cannot.

Reasons:

- Arduino pins supply limited current (~40 mA).
- DC motors require high current.
- Risk of damaging Arduino.

Solution:

- Use relay, transistor, or motor driver (L293D).

Q7. Interface a servo motor with Arduino and explain how angle control is achieved.

Interfacing

- Control wire connected to PWM pin.
- Powered using 5V and GND.

Angle Control

- PWM signal with varying pulse width.
- Typical range: 1 ms → 0°, 2 ms → 180°.

Command Used:

```
servo.write(angle);
```

Q8. Explain the PWM concept used to control servo motors and LED brightness.

PWM (Pulse Width Modulation) controls power by varying ON/OFF time.

PWM in LED

- Higher duty cycle → brighter LED.
- Lower duty cycle → dim LED.

PWM in Servo Motor

- Duty cycle defines angular position.
- Precise control of movement.

Advantages of PWM

- Energy efficient
- Smooth control
- Widely used in IoT actuators

UNIT 3: IoT, IIoT, M2M Concepts and Applications

Q1. Differentiate between IoT, IIoT, and M2M in terms of areas of application, deployment, and development.

IoT, IIoT, and M2M are related concepts but differ in scope, complexity, and application domains.

Aspect	IoT	IIoT	M2M
Full form	Internet of Things	Industrial Internet of Things	Machine to Machine
Application	Consumer & commercial	Industrial & manufacturing	Device-to-device
Deployment	Cloud-based	Edge + Cloud	Mostly local
Intelligence	High (analytics, AI)	Very high (automation)	Low
Human interaction	Moderate	Minimal	None
Examples	Smart home	Smart factory	Smart meters

Conclusion: IIoT is a specialized industrial form of IoT, while M2M is basic device communication.

Q2. Discuss advantages of IoT systems in agriculture, transportation, manufacturing, and home automation.

1. Agriculture

- Smart irrigation systems reduce water wastage.
 - Soil moisture sensors improve crop yield.
 - Weather monitoring enables precision farming.
-

2. Transportation

- GPS tracking improves fleet management.
 - Traffic monitoring reduces congestion.
 - Predictive maintenance enhances vehicle safety.
-

3. Manufacturing

- Enables smart factories (IIoT).
- Predictive maintenance reduces downtime.
- Real-time monitoring improves productivity.

4. Home Automation

- Remote control of appliances.
- Energy efficiency.
- Enhanced safety and comfort.

Conclusion: IoT improves efficiency, automation, and decision-making.

Q3. Design an IoT ecosystem for irrigation, transportation, and home automation. Explain required devices, connectivity, cloud, dashboard, and analytics.

A. Smart Irrigation System

- **Devices:** Soil moisture sensor, ESP32, relay, water pump
 - **Connectivity:** Wi-Fi / LoRa
 - **Cloud:** AWS IoT / Azure
 - **Dashboard:** Moisture level, pump status
 - **Analytics:** Water usage optimization
-

B. Smart Transportation System

- **Devices:** GPS, speed sensors, ESP32
 - **Connectivity:** Cellular / Wi-Fi
 - **Cloud:** Fleet management cloud
 - **Dashboard:** Vehicle tracking
 - **Analytics:** Route optimization
-

C. Home Automation System

- **Devices:** Sensors, relays, Arduino/ESP32
 - **Connectivity:** Wi-Fi / Zigbee
 - **Cloud:** MQTT broker
 - **Dashboard:** Mobile app
 - **Analytics:** Energy consumption
-

Q4. Is cloud service preferred for each designed IoT ecosystem? Justify your answer.

Yes, cloud services are preferred due to:

- Scalability
- Centralized data storage
- Analytics and AI integration
- Remote access

However, **edge processing** may be used where latency or connectivity is critical (e.g., irrigation).

Q5. Which deployment model would you select for those systems and why?

System	Deployment Model	Reason
Irrigation	Hybrid	Local control + cloud analytics
Transportation	Public cloud	Global access
Home automation	Private/Hybrid	Data privacy

Q6. A consumer AC company wants to integrate IoT features into new versions of their products. Propose a system and describe benefits customers would experience.

Proposed System

- Temperature & humidity sensors
- ESP32 with Wi-Fi
- Mobile app & cloud dashboard

Benefits

- Remote control
 - Energy optimization
 - Predictive maintenance
 - Personalized comfort
-

Q7. Recommend an IoT system design for smart agriculture, listing major components and communication flow.

Major Components

- Sensors (soil moisture, temperature)
- ESP32 microcontroller

- Wi-Fi/LoRa communication
- Cloud server
- Dashboard & analytics

Communication Flow

Sensor → Microcontroller → Cloud → Dashboard → Actuator control

Conclusion: Enables precision agriculture and efficient resource use.

UNIT 4: Communication Technologies and Protocols for IoT

Q1. Compare Zigbee, Wi-Fi and Bluetooth with scenario-based analysis, such as battery-operated sensor node design.

IoT systems use different wireless technologies depending on power, range, and data requirements.

Feature	Zigbee	Wi-Fi	Bluetooth
Power consumption	Very low	High	Low
Range	Medium (10–100 m)	High	Short
Data rate	Low	Very high	Medium
Topology	Mesh	Star	Star
Battery use	Excellent	Poor	Good
Example	Smart lighting	IP cameras	Wearables

Conclusion: Zigbee is best for battery-powered sensor nodes due to low power consumption.

Q2. Compare Modbus, MQTT, CoAP, HTTP, REST and raw socket communication, providing appropriate use-case examples.

Protocol	Model	Overhead Use Case	
Modbus	Master-Slave	Low	Industrial PLCs
MQTT	Publish-Subscribe	Very low	IoT telemetry
CoAP	Request-Response	Low	Constrained devices
HTTP	Request-Response	High	Web services
REST	Stateless	Medium	APIs
Raw Socket	Custom	Lowest	Real-time systems

Conclusion: MQTT and CoAP are preferred for IoT due to low overhead.

Q3. Explain MQTT protocol and its role in IoT systems. Describe its working principle.

MQTT (Message Queuing Telemetry Transport) is a lightweight messaging protocol designed for IoT.

Role in IoT

- Efficient data transfer
- Low bandwidth usage

- Reliable communication

Working Principle

- Uses **Publish-Subscribe** model
- Components:
 - Publisher
 - Broker
 - Subscriber
- Messages sent using **topics**

Example: Sensor publishes data → broker → subscriber receives.

Q4. How does MQTT decouple publishers, brokers and subscribers?

- Publishers do not know subscribers.
- Subscribers do not know publishers.
- Broker manages message delivery.

Advantages

- Scalability
 - Flexibility
 - Loose coupling
-

Q5. What is the significance of a topic in MQTT?

- Topic is a hierarchical string used to categorize messages.
- Determines message routing.

Example:

home/livingroom/temperature

Significance

- Organized data flow
 - Easy subscription control
-

Q6. Design an example MQTT topic hierarchy for home automation.

Example hierarchy:

home/

```
|--- livingroom/  
|   |--- temperature  
|   |--- light  
|--- bedroom/  
|   |--- fan  
|   |--- humidity  
--- kitchen/  
    |--- gas  
    |--- smoke
```

Q7. Write the topic expression that allows a client to subscribe to all topics within a designed home automation system.

home/#

- # is a multi-level wildcard.
 - Subscribes to all subtopics under home.
-

Q8. In different layers of IoT systems, which application-layer protocols would you prefer and why?

Layer	Preferred Protocol	Reason
Device-to-Gateway	MQTT	Low power
Gateway-to-Cloud	HTTP/REST	Compatibility
Cloud APIs	REST	Scalability
Constrained devices	CoAP	Lightweight

Q9. In which situations should you avoid using MQTT, CoAP, HTTP or REST? Explain with reasons.

- **Avoid MQTT:** Real-time hard latency systems.
 - **Avoid CoAP:** Large data transfers.
 - **Avoid HTTP:** Battery-powered devices.
 - **Avoid REST:** Event-driven communication.
-

Q10. Compare MQTT and CoAP in terms of reliability, overhead, communication model and use cases.

Feature	MQTT	CoAP
Model	Publish-Subscribe	Request-Response
Reliability	QoS levels	Confirmable messages
Overhead	Very low	Low
Transport	TCP	UDP
Use case	Cloud IoT	Constrained devices

Q11. Create an MQTT topic structure for a smart college system covering classrooms, labs, sensors and devices.

Example structure:

```
college/
  └── classroom/
    |   └── room1/temperature
    |   └── room1/light
  └── lab/
    |   └── lab1/equipment
  └── security/
    |   └── camera/status
```

UNIT 5: Cloud Computing, Virtualization, Availability and Databases

Q1. What is cloud computing? Explain different cloud service models with examples.

Cloud computing is the on-demand delivery of computing resources such as servers, storage, databases, networking, and software over the internet.

Cloud Service Models

1. Infrastructure as a Service (IaaS)

- Provides virtualized hardware resources.
- User manages OS and applications.

Examples: AWS EC2, Microsoft Azure VM

2. Platform as a Service (PaaS)

- Provides platform for application development.
- User focuses on code.

Examples: Google App Engine, AWS Elastic Beanstalk

3. Software as a Service (SaaS)

- Provides complete applications over the internet.

Examples: Google Drive, Gmail, IoT dashboards

Q2. Explain cloud deployment models and suggest suitable models for IoT systems.

1. Public Cloud

- Shared infrastructure.
- Cost-effective and scalable.

Example: AWS, Azure

2. Private Cloud

- Dedicated infrastructure.
- High security.

Used in: Healthcare, defense IoT

3. Hybrid Cloud

- Combination of public and private.
- Flexible and secure.

Best for: Large-scale IoT systems

4. Community Cloud

- Shared among organizations with similar needs.
-

Q3. Explain virtualization and its importance in cloud-based IoT systems.

Virtualization is the creation of virtual resources such as servers, storage, or networks from physical hardware.

Importance

- Efficient resource utilization
- Scalability
- Fault isolation
- Cost reduction

In IoT: Enables multiple applications to run on shared cloud infrastructure.

Q4. What is availability in cloud computing? How can high availability be achieved in IoT systems?

Availability refers to the percentage of time a system remains operational.

High Availability Techniques

- Load balancing
- Redundancy
- Auto-scaling
- Backup and disaster recovery

Importance in IoT: Ensures continuous data collection and monitoring.

Q5. Differentiate between scalability and elasticity in cloud computing.

Feature Scalability Elasticity

Definition Handle growth Automatic adjustment

Feature	Scalability	Elasticity
Time	Long-term	Real-time
Example	Adding servers Auto-scaling	

Q6. What is a database? Compare SQL and NoSQL databases in the context of IoT.

A database is an organized collection of data.

SQL vs NoSQL

Feature	SQL	NoSQL
Structure	Fixed schema	Flexible
Scalability	Vertical	Horizontal
IoT data	Less suitable	Highly suitable
Examples	MySQL, MongoDB, Cassandra	

Q7. Why are NoSQL databases preferred over SQL databases for IoT applications?

Reasons

- Handles large-scale sensor data
- Schema flexibility
- High write throughput
- Horizontal scalability

Conclusion: NoSQL suits real-time IoT data ingestion.

Q8. Explain time-series data and its importance in IoT analytics.

Time-series data is data collected at regular time intervals.

Importance

- Trend analysis
- Predictive maintenance
- Real-time monitoring

Examples: Temperature logs, energy usage data

Q9. Explain the role of cloud dashboards and analytics in IoT systems.

Cloud Dashboards

- Visualize real-time data
- Enable remote monitoring

Analytics

- Detect patterns
- Predict failures
- Optimize performance

Conclusion: Dashboards convert raw IoT data into actionable insights.

Q10. Explain the data flow in a cloud-based IoT system from sensors to analytics.

Data Flow

Sensor → Microcontroller → Gateway → Cloud → Database → Analytics → Dashboard → Actuator

Benefits

- Centralized control
- Real-time insights
- Automation

Q1. Explain the role of Artificial Intelligence (AI) in IoT systems. How does AI improve decision-making?

Artificial Intelligence (AI) enables IoT systems to analyze large volumes of sensor data and make intelligent decisions without human intervention.

Role of AI in IoT

- Pattern recognition in sensor data
- Predictive maintenance
- Automation and optimization
- Anomaly detection

Decision-Making Improvement

- Real-time analysis
- Adaptive responses
- Reduced human dependency

Example: Smart thermostats automatically adjust temperature based on user behavior.

Q2. Differentiate between Artificial Intelligence, Machine Learning, and Deep Learning with examples.

Feature	AI	ML	Deep Learning
Definition	Broad concept	Subset of AI	Subset of ML
Learning	Rule-based	Data-driven	Neural networks
Complexity	Low–High	Medium	Very High
Example	Expert systems	Recommendation systems	Image recognition

Q3. Explain supervised, unsupervised and reinforcement learning with IoT-based examples.

1. Supervised Learning

- Uses labeled data.
- Predicts known outcomes.

Example: Predicting equipment failure using sensor data.

2. Unsupervised Learning

- Uses unlabeled data.

- Discovers hidden patterns.

Example: Clustering energy consumption data.

3. Reinforcement Learning

- Learns through rewards and penalties.

Example: Smart traffic signal control.

Q4. Explain the concept of Edge AI and compare it with Cloud AI.

Edge AI

- AI processing at device or gateway.
- Low latency.
- Reduced bandwidth usage.

Cloud AI

- Processing at centralized cloud.
- High computational power.
- Higher latency.

Feature	Edge AI	Cloud AI
Latency	Very low	High
Connectivity	Not always required	Required
Power	Limited	High

Q5. Why is Edge AI preferred in real-time IoT applications?

Reasons

- Immediate decision-making
- Reduced network dependency
- Enhanced data privacy
- Lower bandwidth consumption

Applications: Autonomous vehicles, industrial automation.

Q6. Explain migration in IoT systems. Why is migration required?

Migration refers to moving IoT applications, data, or services from one platform to another.

Reasons for Migration

- Scalability needs
 - Cost optimization
 - Technology upgrades
 - Vendor independence
-

Q7. Explain different types of migration in cloud-based IoT systems.

1. Device Migration

- Moving devices to new platforms.

2. Data Migration

- Transferring historical IoT data.

3. Application Migration

- Shifting applications between clouds.

4. Cloud-to-Cloud Migration

- AWS to Azure or vice versa.
-

Q8. What challenges are faced during IoT migration? Explain with examples.

Challenges

- Data loss risk
- Compatibility issues
- Downtime
- Security concerns

Example: Migrating healthcare IoT data requires strict compliance.

Q9. Explain the role of Machine Learning in predictive maintenance with an IoT example.

Role

- Analyzes historical sensor data.
- Predicts failures before they occur.

Example:

Sensors → ML model → Failure prediction → Maintenance scheduling

Benefits

- Reduced downtime
 - Cost savings
 - Increased equipment lifespan
-

Q10. Explain how AI-enabled IoT systems improve smart city applications.

Smart City Improvements

- Traffic management
- Smart energy grids
- Waste management
- Public safety

Conclusion: AI enhances efficiency, sustainability, and quality of life in smart cities.