6LoWPAN Overview

Sure! Here's your **Introduction to IoT (Parts I, II, III)** content formatted in clear, structured **notes-style** just like the previous batch:

Introduction to IoT – Part I

Definition of IoT

- Involves connecting devices, machines, tools wirelessly to the internet.
- A network of physical objects embedded with technology to communicate and interact with internal states or external environments.

Scale of IoT

- Currently: 9+ billion connected devices.
- Projected: Over 20 billion in near future.

Key IoT Enablers

- Low-power embedded systems
- Cloud computing
- Big data
- Machine learning
- Networking
- RFID, nanotech, sensors, smart networks

Origin of the Term

- Popularized by **ITU Internet Report 2005**.

machine to machine

- Discussed extending M2M connectivity to everyday household devices.

Characteristics of IoT

- Efficient, scalable, and associated architecture
- Unique naming and addressing

- Abundant sleeping nodes
- Support for mobile and non-IP devices
- Intermittent connectivity

Intermittent connectivity refers to an Internet connection that continuously disconnects and reconnects, slows down, or becomes unstable at random intervals

IoT Market Applications

- **Business/Manufacturing:** Real-time analytics, robotics
- **Healthcare**: Portable monitors, e-records
- **Retail**: Inventory tracking, mobile purchasing
- **Security**: Biometric locks, remote sensors

Evolution of Connected Devices

- ATM (1974), WWW (1991)
- Smart meters, locks, vehicles, healthcare
- Smart Cities, Smart Dust

Modern IoT Applications

- Smart Parking
- Smart Grid
- Waste Management
- Forest Fire Detection
- Air Pollution Monitoring

Relationship with M2M, CPS, and WoT

- **M2M**: Communication between machines/devices, part of IoT.
- **CPS**: Cyber-Physical Systems—tight integration of computation with physical processes.
- **WoT**: Web-of-Things—uses web standards (e.g., REST APIs) to integrate IoT with the Web.

Introduction to IoT – Part II

Address Crunch in IoT

- Massive device growth (20–50 billion by 2018) leads to IP address shortage.
- Integration of legacy and new systems complicates address management.

IoT Network Topologies

- **IoT LAN**: Local comms, may not need internet access.
- **IoT WAN**: Connects LANs over geographic/organizational areas via the Internet.
- **IoT Node**: Connects with other nodes inside LAN.
- **IoT Gateway**: Router for LAN-to-WAN/Internet, forwards at IP layer.
- **IoT Proxy**: Active application-layer bridge between IoT nodes and external networks.

Addressing in IoT

- Use **local addresses** within gateway domains to conserve addresses.
- Gateways are assigned **unique network prefixes** by routers.

Impact of Mobility

- Changing WAN addresses doesn't affect LAN-level addresses when using **ULA (Unique Local Addressing)**.

Gateways vs. Proxies

- **Gateways**: Bridge between local devices and Internet.
- **Proxies**: Handle communication and processing for locally addressed nodes.

IPv4 vs IPv6

Feature	IPv4	IPv6
Address Length	32 bits	128 bits
Notation	Dotted Decimal	Hexadecimal
Allocation	DHCP	SLAAC / DHCPv6
Security	Optional IPSec	Mandatory IPSec
Header	Variable, Complex	Fixed, Simpler

IoT Deployment Challenges

- No global IPv6 transition plan.
- Interim solutions: NAT64, 6to4 Tunneling, Application-layer proxies.

Multi-homing

- Node/network connected to multiple networks to increase availability and reliability.

Introduction to IoT – Part III: Sensing & Actuation

Sensor

- Detects environmental/state changes and converts them into signals (input device).
- Part of a **transducer** when paired with an actuator.

Transducer

- Converts energy from one form to another.
- Includes both sensors and actuators.

Sensor Characteristics

- Sensitive to the desired property only
- Minimal influence on the measured property
- Resistant to interference

Sensor Resolution

- Smallest detectable change in measurement.
- High resolution → better precision (not necessarily accuracy).

Sensor Classifications

- **By Output**:
  - Analog (continuous)
  - Digital (discrete)
- **By Data Type**:
  - Scalar (e.g., temperature)

- Vector/Multimedia (e.g., acceleration)

Examples of Sensor Types

Property	Sensor Type
Light	LDR, Photodiode
Temperature	Thermocouple, Thermistor
Force	Strain gauge
Position	Potentiometer, Encoders
Speed	Doppler-based sensors
Sound	Carbon Microphone
Chemical	Liquid/Gas Chemical Sensors

Sensorial Deviations (Errors)

- **FSR**: Full Scale Range
- **Sensitivity Error**, **Offset/Bias Error**, **Non-linearity**
- **Drift**, **Noise**, **Hysteresis**, **Quantization Error**
- **Dynamic Error / Aliasing**, **Cross-Sensitivity**

Actuator

- Converts energy into mechanical motion (output device).
- Receives control signals and performs physical actions.

Types of Actuators

- **Hydraulic**: Fluid-based (high force)
- **Pneumatic**: Air-based
- **Electric**: Motors, solenoids
- **Thermal/Magnetic**: Shape memory alloys
- **Mechanical:** Rotary to linear converters

Soft & Smart Actuators

- Designed for delicate operations.
- **SMPs (Shape Memory Polymers)**, **LAPs (Light Activated Polymers)** respond to stimuli like heat or light.

Basic IoT System Components

- **Device (Thing)**: Sensor/Actuator
- **Local Network**: Connectivity layer
- **Internet**: Communication backbone
- **Backend Services**: Data storage and processing
- **Applications**: User-facing interfaces

Functional Components

- Interaction and communication modules
- Data processing and analytics
- Web and application service integration
- User interfaces and dashboards

IoT Categories

- **Industrial IoT (IIoT):** Large-scale, IP-network integrated.
- **Consumer IoT**: Home/retail use, often local networks (e.g., Bluetooth, Wi-Fi).

Associated Technologies

- Big Data, Cloud, Smart Grid, M2M, CPS, WoT

Challenges

- **Interfacing**
- **Interoperability**
- **Data Storage**
- **Security**
- **Scalability**
- **Energy Efficiency**

Sure! Here's a well-formatted and organized version of your notes for easier readability and quick reference:

Assignment 2

6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks)

- **Definition**: Enables small, low-power devices to communicate wirelessly using IPv6.
- **Purpose**: Facilitates IoT device integration with the Internet.
- **Standardization**: Defined by IETF (RFC 4919, RFC 5933).
- **IEEE 802.15.4 Support**: Supports 128-bit IPv6 addresses over IEEE 802.15.4 radios.
- **Header Compression**: Uses compression and translation to handle IPv6 headers efficiently.
- **Packet Handling**: IPv6 packets are compressed and adapted to IEEE 802.15.4 frame format.
- **Applications**: IoT, Smart Grid, M2M.
- **Addressing**:
  - 64-bit Extended (Globally Unique)
  - 16-bit Short (PAN-specific)
- **Multicast**: Handled as link-layer broadcast (802.15.4 doesn’t support native multicast).
- **Packet Format Headers**:
  - *Dispatch Header*: Communication start and next header identification.
  - *Mesh Header*: For intra-PAN multi-hop routing.
  - *Fragmentation Header*: Supports large IPv6 packets.
- **Routing**:
  - *Mesh Routing*: Within PAN.
  - *Inter-domain Routing*: Between PAN and IPv6.
- **Protocols**:
  - *LOADng*: AODV-based; destination-only reply to RREQ.
  - *RPL*: Distance Vector for lossy networks, supports proactive/reactive behaviors.

RFID (Radio-Frequency Identification)

- **Definition**: Uses radio waves to read data stored on tags.
- **Components**:
  - RFID Tag (IC + Antenna)
  - RFID Reader
  - Antenna
- **Types of Tags**:
  - *Passive*: Powered by reader signal.
  - *Active*: Has its own power source.
- **Working Principle**: AIDC technology using radio waves.
- **Advantages**: No line-of-sight needed (unlike barcodes).
- **Applications**: Inventory, asset tracking, access control, supply chain, anti-counterfeit, etc.

MQTT (Message Queue Telemetry Transport)

- **Definition**: Lightweight, publish-subscribe messaging protocol over TCP/IP (ISO/IEC PRF 20922).
- **History**: Developed by IBM (1999), standardized by OASIS (2013).
- **Design Goal**: Efficient connectivity for low-resource devices and networks.
- **Architecture**: Event-driven (pub/sub), with a central *broker*.
- **Key Components**:
  - *Publishers* (e.g., sensors)
  - *Subscribers* (applications)
  - *Broker*: Routes messages by topic
- **Methods**: Connect, Disconnect, Subscribe, Unsubscribe, Publish.
- **Topics**: Hierarchical (e.g., `home/livingroom/temp`)

    - `+`: single level wildcard
    - `#`: multi-level wildcard
- **Applications**: Facebook Messenger, AWS IoT, Azure IoT, Adafruit IO.

SMQTT (Secure MQTT)

- **Definition**: MQTT with lightweight attribute-based encryption.
- **Key Advantage**: Broadcast encryption—one message for multiple recipients.
- **Phases**:
    - *Setup*: Key registration
    - *Encryption/Decryption*: With a master key
    - *Publish*: Broker handles encryption
- **Goal**: Enhance MQTT security.
- **Note**: Encryption standards are not yet fixed.

CoAP (Constrained Application Protocol)

- **Purpose**: Lightweight web protocol for constrained devices and networks.
- **Communication Model**: Request-response over **UDP**.
- **RESTful Design**: Supports HTTP-like methods (GET, PUT, POST, DELETE).
- **Structure**:
    - *Messaging Layer*: Handles reliability
    - *Request/Response Layer*: Handles communication
- **Messaging Types**:
    - *Confirmable (CON)*: Reliable
    - *Non-confirmable (NON)*: Unreliable
    - *Piggyback*: Response in ACK
    - *Separate*: Response sent later

- **Features**: Minimal overhead, discovery, simple caching, subscription mechanism.
- **Applications**: Smart energy, building automation, IoT.

XMPP (Extensible Messaging and Presence Protocol)

- **Type**: XML-based messaging protocol for real-time structured data exchange.
- **Standard**: Open and extensible.
- **Architecture**: Client-server; can be decentralized.
- **Key Features**:
  - Service discovery
  - Real-time messaging
  - Peer-to-peer capabilities
- **Technologies**:
  - *Jingle*: Multimedia signaling
  - *PubSub*: Event updates
  - *BOSH*: HTTP binding
- **Weaknesses**: No QoS, base64 encoding needed for binary, text-based overhead.
- **Use Cases**: VoIP, file sharing, gaming, smart grid, social networking.

AMQP (Advanced Message Queuing Protocol)

- **Definition**: Binary protocol for business message exchange (ISO/IEC 19464).
- **Purpose**: Secure, reliable, and interoperable messaging.
- **Core Components**:
  - *Exchange*: Routes messages
  - *Queue*: Message storage
  - *Bindings*: Routing rules
- **Exchange Types**: Direct, Fan-out, Topic, Header.

- **Message Delivery Guarantees**:
  - At-most-once
  - At-least-once
  - Exactly-once
- **Frame Types**: 9 control frames including Open, Attach, Transfer, Close, etc.
- **Applications**: Task delegation, offline communication, monitoring, distributed systems.

IEEE 802.15.4

- **Definition**: Standard for low-rate WPANs.
- **Layers**: PHY + MAC + (LLC & SSCS for upper layer interaction).
- **Frequency**: ISM band.
- **Modulation**:
  - *BPSK*: Low data rate
  - *O-QPSK*: High data rate
- **Channel Access**: CSMA/CA.
- **Power Efficiency**: Very low power, suitable for battery operation.
- **Range**: 10m–75m (up to 1000m LOS).
- **Topologies**: Star, Mesh.
- **Network Types**:
  - *Beacon-enabled*: Uses superframe and beacon sync
  - *Non-beacon-enabled*: Slotted CSMA/CA
- **Device Types**:
  - *FFD*: Full Function Device
  - *RFD*: Reduced Function Device
- **Frame Types**: Beacon, Data, MAC command, Acknowledgment.

Zigbee

- **Built On**: IEEE 802.15.4 (adds network and application layers).
- **Focus**: Mesh networking for WPANs.
- **Security**: Provides authentication and encryption.
- **Topologies**: Star, Tree, Mesh.
- **Device Roles**:
  - *ZigBee Coordinator (ZC)*: Forms and manages network
  - *ZigBee Router (ZR)*: Forwards data, runs applications
  - *ZigBee End Device (ZED)*: Minimal function, low power
- **Routing Protocol**: AODV-based for dynamic path discovery.
- **Applications**: Home automation, energy monitoring, lighting control, healthcare, telecom.

Let me know if you'd like this turned into a printable or presentation format!

fri iot tandon

Here’s a well-structured and cleanly formatted version of your **Week 3 Notes**:

Week 3 Notes – Connectivity Technologies & Sensor Networks

Connectivity Technologies – Part III

HART & WirelessHART

HART (Highway Addressable Remote Transducer) is a communication protocol that allows two-way digital communication between smart field devices and control systems,

- **WirelessHART**: An advanced version of the HART Protocol for smart field device networking.
- **Physical Layer**:
  - Based on **IEEE 802.15.4**.
  - Uses **time-synchronized channel hopping** with super-frames (10ms timeslots).
  - Implements **channel blacklisting** to avoid interference.
- **Network & Transport Layers**:
  - Enable **mesh networking**.
  - Devices forward packets and maintain a **network graph**.
- **Application Layer**:
  - Command-response messaging, consistent with wired HART.
- **Congestion Control**:
  - Operates in **2.4 GHz ISM band**, excluding channel 26.
- **Network Manager**:
  - Manages routing, timing, node access, and security.
- **WirelessHART vs. ZigBee**:
  - Channel hopping: *per message* (WirelessHART) vs. *per network* (ZigBee).

- MAC Layer: *TDMA* (WirelessHART) vs. *CSMA/CD* (ZigBee).

NFC (Near Field Communication)

- **Derived from**: RFID.
- **Types**: Type A, Type B, **FeliCa** (common in Japan).
- **Device Types**:
  - *Passive*: Can only transmit stored data (e.g., NFC tags).
  - *Active*: Can transmit and receive (e.g., smartphones).
- **Working Principle**: **Magnetic induction**.
- **Specifications**:
  - Frequency: **13.56 MHz**.
  - Data Rates: **106, 212, 424 Kbps**.
  - Range: **< 20 cm**.
  - Storage: **96–512 bytes**.
- **Modes of Operation**:
  - Peer-to-Peer
  - Read/Write
  - Card Emulation
- **Applications**:
  - Mobile payments, parcel tracking, ads, smart toys, home automation.

Connectivity Technologies – Part IV

Bluetooth

- **Purpose**: Short-range wireless cable replacement; secure ad-hoc connections.
- **Device Classes**:
  - Class 3: 1 m

- Class 2: 10 m (common)
- Class 1: 100 m

- **Connection Setup**:
  - **Inquiry**, **Paging**, **Connection**
- **Power Saving Modes**:
  - *Sniff*: Intermittent listening
  - *Hold*: Sleep for defined time
  - *Park*: Inactive until reactivated
- **Protocol Stack**:
  - **Baseband**: Manages channel access, error correction, security
  - **L2CAP**: Multiplexing, segmentation
  - **RFCOMM**: Serial port emulation
- **Network**: **Piconet**
  - One *master*, up to *7 slaves*
  - Communication: *Master* ↔ *Slave*, no direct slave-to-slave

---

Connectivity Technologies – Part V

Z-Wave

- **Purpose**: RF-based home automation.
- **Frequencies**: Region-specific (e.g., 908.42 MHz – US, 868.42 MHz – EU).
- **Topology**: **Mesh**, supports up to **232 nodes**.
- **Modulation**: **GFSK**
- **Channel Encoding**: **Manchester**
- **Network Management**:
  - One primary controller.
  - Unique **Home ID** for network, **Node ID** for each device.
- **Self-Healing Mesh**: Routes around obstacles.

- **Z-Wave vs. ZigBee**:
  - Z-Wave: User-friendly, secure, slightly expensive.
  - ZigBee: Ultra low-power, customizable, tech-savvy users.

ISA 100.11A

- **Use Case**: Industrial automation.
- **Topologies**: Star, tree, mesh.
- **Supports**: Ethernet, fieldbuses, radio links.
- **Key Features**:
  - Multi-protocol support, error detection, TDMA, QoS.
  - **Security**: Dual-layer encryption (data link & transport), key distribution via security manager.
- **Usage Classes**: Based on criticality (Safety → Monitoring).

Sensor Networks – Part I

Wireless Sensor Networks (WSNs)

- **Composition**: Many sensor nodes deployed to monitor environmental factors.
- **Data Relay**: Uses **multi-hop paths** to transmit data to a sink.
- **Node Characteristics**:
  - Compact, battery-powered, multifunctional.
  - Short-range, often run **TinyOS**.
- **Constraints**:
  - Small size, low power, autonomy, low cost.
- **Applications**:
  - Monitoring: Temp, humidity, light, pressure, etc.
- **Detection Scenarios**:

    - Single/Multi Source vs. Single/Multi Object
- **Challenges**:
    - Energy management, Security threats (interference, eavesdropping).
- **Sensor Web**:
    - Interfaces to access/control sensors online.
    - Includes: SOS, SPS, SAS, WNS, SensorML, O&M.

Sensor Networks – Part II

Node Behavior in WSNs

- **Types**:
    - *Normal*
    - *Misbehaving*:
        - Unintentional: Failed/Badly failed
        - Intentional: Selfish/Malicious
- **Restoration Techniques**:
    - **CoRD**, **CoRAD**

Topology & Management

- **Event-Aware Topology**:
    - Dynamic response to events (location, area, duration).
- **INTSEM (Information-Theoretic Self Management)**:
    - Controls transmission via sleep cycles for energy efficiency.

Social Sensing & Rare Events

- **Challenge**: Monitoring rare events with standard methods (e.g., SMAC, PW-MAC).

- **Solution: PDC (Probabilistic Duty Cycle)** using social media insights to adjust node activity.

Applications

Mines:

- Fire Monitoring and Alarms (FMA) using real-time sensor feedback.

Healthcare:

- **WBANs (Wireless Body Area Networks):**
  - Continuous monitoring of vitals, elderly care.
  - **Cloud-Assisted WBANs**:
    - Post-disaster fairness in data aggregation using social choice.
    - Uses pseudo-clusters and mobile aggregation centers.
  - **Payload Tuning**:
    - Adjusts for patient-specific traits using **FIS** and **MDP**.
  - **PATS (Priority-based Time-slot Allocation Scheme):**
    - Evolutionary game theory approach to prioritize medical emergencies.

~~Let me know if you'd like this in a downloadable PDF, formatted as flashcards, or turned into a revision summary!~~

~~Sure! Here's a well-organized, formatted version of your **Week 4, Week 5, and Week 6 Notes** for easy reading and review:~~

Week 4 Notes

Sensor Networks – Part III

- Target Tracking:

    - Push-based, poll-based, and guided formulations.
    - Cluster, tree, and face structures respectively.
- **WSNs in Agriculture**:
    - Used for agricultural intrusion detection.
    - Distributed topology management (coverage, connectivity, lifetime).
    - Coalition Formation Games for WMSNs.

AID (Agriculture Information Device), a set of sensor nodes are deployed over an agricultural field to:

- **3D Localization in UWSNs**:
    - Silent, energy-efficient localization using 3 surface anchor nodes.
    - Iterative, mobility-aware approach.
- **Self-Organizing Virtual Architecture (Tic-tac-toe-arch)**:
    - Calculates node connectivity durations.
    - Dynamic virtual topology formation.

The objective of coverage in WSN is to use minimum number of sensors and maximize the network lifetime.

Sensor Networks – Part IV

- **WSN Coverage**:
    - Monitors area of interest, ensures connectivity.
    - Event-driven (e.g., forest fire) or on-demand (e.g., inventory) reporting.
    - Goal: Fewer sensors, longer network lifetime.
- **Coverage Algorithms**:
    - Centralized (global map), distributed (neighbor-based), localized (subset of nodes).
    - Sensor deployment: Deterministic or random.
- **Coverage Types in Static WSNs**:
    - **Area Coverage**: Random and connected random.
    - **Point Coverage**
    - **Barrier Coverage**: Weak and strong types.
- **Coverage Maintenance**:
    - Ensure all 'crossings' (intersections of sensing areas) are covered.

- **Optimal Geographical Density Control (OGDC) Algorithm:**
  - Nodes compute deviations (distance, angle) from desired config.
  - Optimal node remains active; others sleep to conserve energy.

Sensor Networks – Part V

If there is a failure in the stationary sensor network then it is likely that the point of failure can partition the network into two or more fragments.

- **Stationary WSNs**:
  - Static, easy deployment and optimized placement.
  - Risk of partitioning on node failure.
- **Mobile WSNs (MWSNs)**:
  - Like MANETs with self-CHOP properties (Configure, Heal, Optimize, Protect).
  - Components: Mobile nodes, sinks, data mules.
  - Applications: Marine life monitoring, water quality.
- **Participatory Sensing**:
  - Human-carried devices for distributed sensing.
  - Enables data sharing and authenticity verification.

Human Centric Sensing

Energy of devices (battery life) and participant selection (choosing reliable human users/devices) are major challenges in Human-Centric Sensing systems.

- **Flying Ad Hoc Networks (FANETs)**:
  - Gateway selection by stable node in sub-area.
  - Not ideal for time-critical relaying.
- **Machine-to-Machine (M2M) Communication:**
  - Numerous low-cost, low-energy nodes.
  - Automatic communication, minimal human interaction.
  - **Low-end Nodes**: Cheap, static, limited capabilities, no IP, dense deployment.

Week 5 Notes

Device Interoperability

User Interoperability: This refers to the interoperability problem between a user and a device.

Device Interoperability: This refers to the interoperability problem between two different devices

- Describes actions during new device connection and control/monitoring phases.

the ability of computer systems or software to exchange and make use of information.

Introduction to Arduino Programming – Part I

- **Arduino Overview**:
  - Open-source hardware (board) + software (IDE).
  - Accepts analog/digital input, gives output without extra loaders.
- **Arduino Boards**:
  - Based on ATMEGA328, ATMEGA32u4, ATMEGA2560, AT91SAM3X8E.
- **Arduino IDE**:
  - C/C++-based.
  - Key operations: New sketch, open, examples, verify, upload, save, serial monitor.
- **Sketch Structure**:
  - `setup()` – initialization.
  - `loop()` – continuous execution.
- **Data Types**:
  - Includes `int`, `char`, `boolean`, `byte`, `float`, `double`, `String`, etc.
- **Function Libraries**:
  - `pinMode()`, `digitalWrite()`, `analogRead()`
  - `delay(ms)`, character functions like `isdigit()`, `isalpha()`
- **Example – Blink**:
  - LED blinks using `digitalWrite()` and `delay()`.

Introduction to Arduino Programming – Part II

- **Operators**: Arithmetic, comparison, boolean, bitwise, compound.
- **Control Statements**: `if`, `else`, `switch`, ternary operator.
- **Loops**: `for`, `while`, `do...while`, nested/infinite loops.
- **Arrays**: Declared with/without size/values, multi-dimensional supported.
- **String Handling**:

        - Char array or String object.
        - Methods: `replace()`, `toUpperCase()`, `length()`
- **Math Library**: `min()`, `max()`, etc.
- **Random Numbers**:
    - `randomSeed()`, `random(min, max)`
- **Interrupts**:
    - External events.
    - Functions: `digitalPinToInterrupt()`, `attachInterrupt()`
- **Example – Traffic Control**:
    - LEDs simulate traffic lights using `digitalWrite()` and `delay()`.

Use of different programming languages such as JavaScript, Python, JAVA, and others is an example of heterogeneity in IoT. This brings in the need for interoperability

---

Integration of Sensors & Actuators with Arduino

- **Part I – DHT Sensor**:
    - Reads temperature/humidity.
    - Uses `DHT.h`, `DHT.read11()`
- **Part II – Actuators**:
    - Convert energy to motion.
    - Types: Servo, stepper, solenoid, relay, AC motors.
- **Servo Motor Interfacing**:
    - Control pin to digital pin.
    - Use `Servo` library: `attach()`, `write()`, `read()`, etc.

Week 6 Notes

✓

Introduction to Python Programming – Part I

- **Why Python?**
    - Simple, readable, versatile, supports hardware interfacing, open-source.

- **IDEs**: PyCharm, Spyder.
- **Basics**:
  - `print()` function.
  - Indentation defines blocks.
- **Data Types**: Numbers, Strings, Lists, Tuples, Dictionaries.
- **Control Statements**: `if`, `elif`, `else`, `while`, `for`, `break`, `continue`.
- **Functions**:
  - Use `def`, can return values or multiple values.
- **Variable Scope**:
  - Global vs Local.
- **Modules**:
  - Use `import`, `from ... import ...`.
  - Example: `random` module to generate integers.
- **Prime Check Program**: Demonstrates user input, loops, and conditions.

Introduction to Python Programming – Part II

- **File Operations**:
  - Open: `open(filename, mode)` with `r`, `w`, `a`, `r+`.
  - Read: `read()`, Write: `write()`, Close: `close()`
  - `with open(...) as file:` for best practice.
  - CSV Handling: `csv.reader()`, `csv.writer()`
- **Image Operations**:
  - Use `Pillow (PIL)` – `pip install pillow`.

Introduction to Raspberry Pi – Part I

- **Architecture**: CPU/GPU, RAM, USB, Ethernet, GPIO.

- **Setup Requirements**:
  - HDMI, monitor, keyboard, mouse, 5V adapter, microSD card with OS.
- **GPIO**:
  - Digital input/output functionality.
  - Configurable via diagrams.
- **Initial Setup**:
  - `sudo raspi-config` for filesystem and settings.
- **Languages Supported**:
  - Python, C/C++, Java, Scratch, Ruby.
- **Applications**:
  - Media center, home automation, bots, VPNs, lightweight servers.

Raspberry Pi – Part II + Sensor & Actuator Integration

- **Blinking LED Example**:
  - Use `RPi.GPIO` and `time.sleep()` to control LED state.
- **Temperature Dependent Auto Cooling System**:
  - **DHT Sensor**: Connected to GPIO, read using `Adafruit_DHT`.
  - **Relay Control**: Activates mini-fan when temperature exceeds threshold.

In a temperature-controlled fan system using a relay, the fand should turn on when the surrounding temperature exceeds a predefined threshold.

Let me know if you want these compiled into a PDF or formatted for print!

fri iot tandon

Here’s your content formatted into clean, exam-ready **notes** style for Weeks 7 and 8:

✅ Week 7: Software-Defined Networking (SDN)

🔹 Traditional vs SDN Architecture

- **Traditional Networks:** Distributed control (e.g., **OSPF**) on each device.
- **SDN:** Separation of control plane (decision-making) and data plane (packet forwarding).

🔹 SDN Architecture Components

- **Controller:** Centralized decision-making entity.
- **Switches:** Simple forwarding devices.
- **APIs:** Facilitate communication between controller ↔ switches, controller ↔ apps.

🔹 Flow Table and Rule Placement

- **Flow Table:** Contains flow-rules in switches.
- **PACKET-IN:** Triggered when no matching flow-rule found.
- **Controller installs rule** via flow-mod messages.

🔹 Challenges in Rule Placement

- Limited **TCAM** memory in switches.
- **Delays** in packet-in handling and rule installation.

🔹 Objectives

- Efficient **TCAM** usage.
- Minimize **PACKET-IN** events.

🔹 Controller Placement

- Can be **Local** or **Remote (Cloud-based)**.

🔹 Tools and Controllers

- **Mininet:** Virtual SDN network emulator.
- **Popular Controllers:** Pox, Nox, FloodLight, OpenDayLight, ONOS.

🔹 SDN in IoT

- Use-cases:
  - **Rule-placement & traffic engineering** (backbone networks).
  - **Flow classification & security** (data centers).

🔹 Software-Defined Wireless Sensor Networks (Soft-WSN)

- Applies SDN principles to **WSNs**.
- Features:
  - Real-time reprogramming.
  - Dynamic path updates.

📊 Experimental Benefits of Soft-WSN

- ↑ **Packet delivery ratio**
- ↓ **Replicated packets**
- ↑ **Control messages** (due to PACKET-IN)

🔹 SDN-WISE

- A **software-defined WSN** platform.
- Sensor nodes use flow tables.
- Programming support via APIs (any language).

✅ Week 8: Cloud Computing

🔹 Definition (NIST)

> "A model enabling convenient, on-demand access to a shared pool of configurable computing resources."

🔹 Essential Characteristics

1. **On-demand self-service**
2. **Broad network access**
3. **Resource pooling**
4. **Rapid elasticity**
5. **Measured service**
6. **Availability & reliability**
7. **Performance & optimization**

🔹 Cloud Service Models (NIST Visual Model)

- **IaaS (Infrastructure-as-a-Service):** Users manage OS/apps (e.g., AWS EC2).
- **PaaS (Platform-as-a-Service):** Users develop/run apps (e.g., Google App Engine).
- **SaaS (Software-as-a-Service):** Full apps delivered online (e.g., Google Workspace).

🔹 Cloud Deployment Models

- **Private Cloud:** Exclusive to one organization.
- **Community Cloud:** Shared by a specific group.
- **Public Cloud:** Open to general public.
- **Hybrid Cloud:** Combination of above.

🔹 SLAs (Service Level Agreements)

- Define non-functional guarantees: **Availability, Performance, Response time**, etc.

🔹 Accounting & Billing

- **Service accounting:** Tracks usage.
- **Billing:** Applies pricing models to usage data.

🔹 Economics of Scaling

- **Cap-Ex free computing**
- On-demand scalability
- Rapid deployment
- Cost-efficiency

🔹 Data Management in Cloud

- Choose **DBMS** based on deployment type.
- **DBaaS** examples: **Amazon RDS, Azure SQL, Google Cloud SQL**

🔹 Trust & Risk in Cloud

- **Trust mechanisms:** SLAs, audits, ratings, self-assessment tools.
- **Risk Assessment:** Formal/informal, qualitative/quantitative analysis.

🔹 Cloud Simulators

- **CloudSim, CloudAnalyst:** For simulating and evaluating cloud environments.

🔹 Amazon EC2

- Offers **instances** with customizable configurations.
- Supports multiple **OS**, **EBS/S3 storage**, **auto-scaling**, **security groups**.

Let me know if you'd like this as a PDF or formatted for Anki flashcards too!

Here's a clean, easy-to-learn **notes-style format** for Weeks 9 and 10, perfect for MCQ exam prep:

✅ Week 9: Sensor-Cloud and Fog Computing

🔹 WSNs Recap

- **Components:** Sensing unit, Processing unit, Communication unit
- **Features:** Short range, rely on **multi-hop** communication
- **Applications:** Target tracking, Healthcare, Smart cities

🔹 Cloud Computing Recap

- **Services:** SaaS, PaaS, IaaS
- **Benefits:** Elasticity, Pay-per-use, Self-service

🔹 Sensor-Cloud

- **Goal:** Virtualization of **physical sensor nodes**
- **Key Concepts:**
  - **Virtual Sensors (VSs)** are composed from real sensors
  - **CoV-I:** Same region
  - **CoV-II:** Multiple regions
- **Management Issues:**
  - Optimal **VS composition**
  - **Data caching**
  - **Pricing strategy**:
    - **pH (hardware)** – Usage of physical sensors
    - **pI (infrastructure)** – Usage of cloud infrastructure

The optimal composition of Virtual sensor nodes is a management issue in sensor-cloud. (

Internal Cache (IC) and External Cache (EC) are two different types of caching

🔹 Fog Computing

- **Why Needed:** Cloud can't handle IoT's volume, latency, and bandwidth needs
- **Definition:** Layer between IoT devices and the cloud
- **Fog Nodes:** Routers, switches, embedded servers
  - Provide **storage, compute, and network** services

⏱ Time Sensitivity-Based Handling

Data Type	Action
Very time-sensitive	Processed at nearest fog node
Less time-sensitive	Processed at aggregate node
Non-time-sensitive	Sent to cloud for storage and analysis

✅ Advantages

- ↓ Latency → Faster decisions
- ↑ Privacy → Local data analysis
- ↑ Business agility

- 💡 **Use Cases:**
  - Real-time health monitoring
  - Smart energy systems
  - Rail & pipeline monitoring
  - Wind turbine optimization

TCP/IP works best with localized data, which is not present in V2X environments.

This restricts the use of TCP/IP for V2X communication.

⚠️ **Challenges**

- ↑ Power usage (extra nodes)
- Ensuring **data security** across distributed fog
- Maintaining **reliability**

Link durations are short due to the highly dynamic nature of VANETs.

✅ Week 10: Smart Parking & Intelligent Connected Vehicles (ICVs)

🔹 Smart Parking

📡 **Information Collection Methods**

- Sensors
- Parking meters
- Sensor networks
- Crowd sensing

CCN (Content Centric Networking) is derived from Information Centric Networking (ICN)

The phases of ICV development:
Phase 1: Based on 2G,
Phase 2: Based on 4G LTE,
Phase 3: Vehicles connected to cloud

🧠 **System Deployment**

- Software system
- E-parking & guidance
- Data analytics for optimization

ai can bridge Decision making gaps between sensors and actuators

📈 **Decision Layer**

- **Monitors** parking conditions
- **Controls** sensors

HAN standards

Physical and MAC layers are defined by IEEE802.15.4. Network layer
and Application layers are defined by Zigbee.

🔹 Intelligent Connected Vehicles (ICVs)

📶 Spectrum Allocation

- **75 MHz (5850–5925 MHz)** reserved for ICVs by US DoT & FCC

📘 IEEE WAVE Standards (DSRC)

Standard	Focus
IEEE 1609.2	Security services (messages & apps)
IEEE 1609.3	Networking services
IEEE 1609.4	Multi-channel operations
IEEE P1609.11	Over-the-air data exchange for ITS

Let me know if you want these notes exported as a PDF or formatted for flashcards!

fri iot tandon

Here's a simplified, **MCQ-friendly** version of the **Week 11 and 12 notes**, formatted for **easy learning and quick revision**:

📘 Week 11: IIoT and Smart Grid/Home ✓

🏭 Industrial Internet of Things (IIoT)

- Connects **hardware and software.**
- Goals: **Remote access**, **end-to-end security**, **cloud integration**, **big data use**, **smart machines.**

⚙️ IIoT Requirements

- **Connectivity** (hardware/software)
- **Cloud platform**
- **App development support**
- **Big data analytics**

🛠️ Design Considerations

- **Energy**: Battery life
- **Latency**: Delay in data transfer
- **Throughput**: Max data transfer rate
- **Scalability**: No. of devices supported
- **Topology**: Device communication structure
- **Security**: Reliability and safety

🛎️ IIoT Services

- **Primary services**: Core node tasks
- **Secondary services**: Supportive functions

🚚 IIoT Applications

- **Transportation & Logistics**: **Barcodes, RFID**, real-time object tracking

- **Mining: Air quality, gas detection, Wi-Fi, RFID**

⚡ Smart Grid

- Features:
  - **Real-time monitoring**
  - **Smart appliance control**
  - **Building automation**
  - **Real-time pricing**
  - **Renewable energy integration**
  - **Distribution automation**

🏠 Smart Home in Smart Grid

- **Smart Meters**: Interface with provider, automated info transfer, cost reduction
- **Smart Appliances**: Respond to provider signals, can be overridden
- **Home Energy Generation**: **Solar**, **wind**, **hydropower**, **fuel cells**
- **Home Energy Management System**: Schedules appliances for off-peak use

🔁 Consumer Engagement

- **Net Metering**: Paid more during on-peak energy supply

🔌 Smart Grid Operation

- Uses **PMUs** for monitoring transmission systems
- PMUs send data to **SCADA**
- Grid features **self-healing**, **power rerouting**, and **oscillation damping**

🚗 Plug-in Electric Vehicles (PEVs)

- Charged during **off-peak hours**
- Can **supply power** back during peak time

📘 Week 12: Data Handling and Analytics in IoT

📦 Data Handling

- Focuses on **secure data storage, archiving, and disposal**
- Applies to **electronic and non-electronic** data

📊 Data Analytics

- Examines datasets to **derive insights**
- Supports **business decisions**, **scientific verification**

📈 Types of Data Analysis

📎 Qualitative

- Descriptive (text, video, interviews)
- Data grouped by **themes**

ANOVA Assumptions

Normal distribution

Equal variances (homogeneity)

Independent samples

🔢 Quantitative

- **Numerical data** (mean, median, std. dev.)
- Techniques:
  - **Correlation** (Pearson's *r*)
  - **Regression**
  - **Statistical Significance**
  - **Margin of Error** = Critical Value × Std. Deviation

Examples of dispersion measures include Range and Variance

☁️ Data Handling Technologies

- **Cloud Computing**: On-demand, scalable services
  - Models: **IaaS**, **PaaS**, **SaaS**
- **IoT**: Physical objects connected to the internet

🔄 Flow of Data

1. **Generation**
2. **Acquisition** (collection, transport, pre-process)
3. **Storage**
4. **Analysis**

🗂️ Data Sources

- **Enterprise**, **IoT**, **Bio-medical**, **Others** (e.g., astronomy)

🎛️ Data Acquisition

- **Log files** (activity)
- **Sensory data** (temp, sound, etc.)

📚 Big Data Characteristics

- **Volume**: Huge data (e.g., 140M tweets/day)
- **Variety**: Text, images, video, GPS, sensor data

🐘 Hadoop Components

- **NameNode**: Filesystem metadata, in-memory maps
- **DataNode**: Stores data, communicates for balancing/replication
- **Job Tracker**: Manages user jobs
- **Task Trackers**: Execute tasks on nodes

🏥 IoT in Healthcare

- **Telemedicine**: Remote patient monitoring
- **Emergency Response**: Faster care during complications

- **Digital Medical Records**: Cloud storage
- **AmbuSens:** Monitors heart rate, ECG, temp using **WBAN (Bluetooth)**

⚙️ **AmbuSens Focus:**

- **Power efficiency**
- **Data-rate tuning**
- **Filtering & noise removal**

👩‍⚕️ **Post-Disaster Care:**

- Uses **cloud-WBAN**
- Includes **social choice models**, **dynamic gateways**

📶 **WBAN Scheduling:**

- **Priority-based slots** for critical health data
- Fitness = **Criticality + Energy + History**

🧍‍♂️ Human Activity Recognition

- Detects activities like **running, jumping, gestures**
- Sensors: **Cameras, smartphones, fitness bands**
- Uses **ML/DL** for data analysis (on-device or network-based)

Let me know if you'd like a **summary sheet**, **flashcards**, or **MCQs** made from these!

Sensor Networks Coverage Notes

Here’s the content formatted as structured notes, classified into **Week 4**, **Week 5**, and **Week 6**:

Week 4: Sensor Networks – Part III

Target Tracking

- Nodes compute the target's position and notify the sink node periodically in a **push-based formulation** (uses cluster structure).
- In a **poll-based formulation**, nodes register the target’s presence and send reports only when queried (uses tree structure).

Coverage

- Purpose: To collect relevant data for processing or reporting.
- Types of reporting:
  - **Event-driven:** e.g., forest fire monitoring.
  - **On-demand:** e.g., inventory control systems.
- Objective: Use minimum sensors and maximize network lifetime.
- Algorithms:
  - **Centralized:** Global map computed at a central point.
  - **Distributed:** Nodes compute positions using neighbor communication.
  - **Localized:** Only a subset of nodes participates in sensing, communication, and computation.
- Deployment: **Deterministic vs Random**.
- Considerations: **Sensing and communication ranges**.
- Static WSN coverage classifications:
  - **Area coverage**
  - **Point coverage**
  - **Barrier coverage**

Area Coverage

- Examples: **Energy-efficient random coverage**, **Connected random coverage**.
- A network is connected if any active node can communicate with another active node.

Barrier Coverage

- Types: **Weak coverage**, **Strong coverage**.

Coverage Maintenance

- A region R is covered if:
  - Crossings exist in R.
  - Every crossing is covered.
- **Crossings**: Intersection points between disk boundaries or between disk and region boundary.
- A crossing is covered if within at least one node's coverage disk.

Mobile Wireless Sensor Networks (MWSN)

- Intersection of WSNs and MANETs.
- Should follow **self-CHOP**:
  Self-Configure, Self-Heal, Self-Optimize, Self-Protect.
- WSNs have densely deployed sensor nodes that collaborate to measure environmental conditions.

Participatory Sensing

- Proposed by Burke et al. (2006).
- Distributed sensing by devices carried by humans.
- Goal: Data collection **and** knowledge sharing.
- Provides:
  - **Quantitative data** (e.g., CO_2 levels)
  - **Authenticity endorsement** via geo-tags and timestamps.

Gateway Selection in FANETs

- Begins with selecting the **most stable node** in a sub-area.

- Then, **partition parameters** are optimized based on topology.
- Metrics are optimized over iterations to reach an optimal state.

Week 5: Introduction to Arduino Programming

Device Interoperability

- System detects new devices, identifies capabilities, installs drivers/configs.
- Devices are controlled via command signals; status and sensor data are received.

Arduino IDE Overview

- Used to write and upload code to Arduino boards.
- Key functions:
  - **Verify**: Check for compilation errors.
  - **Upload**: Flash code to board (TX–RX LEDs flash).
  - **New**: Create new sketch.
  - **Open**: Open existing sketch (e.g., File → Examples).
  - **Save**: Save current sketch.
  - **Serial Monitor**: View printed data from the board.

Sketch Structure

- **setup()**: Runs once at start. Initializes I/O and pin modes.
- **loop()**: Runs continuously. Executes repetitive tasks.

Supported Datatypes

- Includes: `Void`, `Long`, `Int`, `Char`, `Boolean`, `Unsigned char`, `Byte`, `Unsigned int`, `Word`, `Unsigned long`, `Float`, `Double`, `Array`, `String` (char array and object), `Short`.

Example - Blink

- Connect board to PC.
- Select correct port and board type.
- Use:

  - `digitalWrite(pin, value)` – Set pin to HIGH/LOW.
  - `delay(ms)` – Pause for given milliseconds.
- Verify and upload code.

Operators in Arduino

- **Arithmetic:** `=`, `+`, `-`, `*`, `/`, `%`
- **Comparison:** `==`, `!=`, `<`, `>`, `<=`, `>=`
- **Boolean:** `&&`, `||`, `!`
- **Bitwise:** `&`, `|`, `^`, `~`, `<<`, `>>`
- **Compound:** `++`, `--`, `+=`, `-=`, `*=`, `/=`, `%=`, `|=`, `&=`

Control Statements

- **If**: Executes code if condition is true.
- **If...Else**: Executes different code blocks based on condition.
- **If...Elseif...Else**: Checks multiple conditions sequentially.
- **Switch Case**: Executes code based on variable's value.

Additional Topics Mentioned (Details Not Provided)

- **Loops**, **Arrays**, **String**, **Math Library**, **Random Number**, **Interrupts**, **Example Program**

Week 6: Introduction to Python Programming

Python IDE

- Free, open-source IDEs for coding and module/library integration.
- Available for Windows, Linux, Mac.
- Examples: **Spyder**, **PyCharm**.

Starting with Python

- Print using interpreter prompt:

```python
```

```
>>> print "Hi, Welcome to python!"
```

- Python uses **rigid indentation** for blocks:

```python
if True:
    print "Correct"
else:
    print "Error"
```

Data-types in Python

- **Numbers**: Integers, floats, etc.
  Example: `x, y, z = 10, 10.2, " Python "`
- **String**: Sequence of characters.
  Indexing: `print x` → `T`
  Slicing: `print x[2:4]` → `is`
- **List**: Ordered, mutable. `x = [10, 10.2, 'python']`
- **Tuple**: Ordered, immutable.
- **Dictionary**: Key-value pairs. `d = {1:'item', 'k':2}`

Controlling Statements

- **if, elif, else**
- **while** loops with indented block after condition.

Functions in Python

- Define using `def`:

```python
def example(str):
    print (str + "!")
example("Hi")  # Output: Hi!
```

- Return multiple values:

```python
```

```
def greater(x, y):
    if x > y: return x, y
    else: return y, x
val = greater(10, 100)
print(val)  # Output: (100, 10)
```

- Functions as objects:

```python
def add(a, b): return a + b
print(add(4, 6))
c = add(4, 6)
print c
```

File Read/Write Operations

- Steps: **Open**, **Read/Write**, **Close**
- Modes:
  - `'r'` – Read
  - `'w'` – Write (overwrites)
  - `'a'` – Append
  - `'r+'` – Read + Write
- Read:

```python
file = open('data.txt', 'r')
file.read()
```

- Write:

```python
file = open('data.txt', 'w')
file.write('writing to the file')
```

- Close:

```python
```

```
file.close()
```

- With block:

```python
with open("data.txt", "w") as file:
    file.write("writing to the text file")
```

CSV Files

- Use `csv` module.
- Functions: `csv.reader`, `csv.writer`.

Image Read/Write Operations

- Use **PIL (Pillow)** library.
- Install: `sudo pip install pillow`
- PIL for Python 2.7; Pillow supports Python 3.x.

Additional Note

- Python is one of the default installed languages on Raspberry Pi, indicating its importance in IoT.

Let me know if you want this in PDF or DOCX format too!

DLNA stands for Digital Living Network Alliance

DSRC stands for Dedicated Short Range Communications

WAVE stands for Wireless Access in Vehicular Environments.

DAU stands for Data Aggregator Unit

MDMS stands for Meter Data Management System.

IIoT is a network of physical objects, systems, platforms and applications