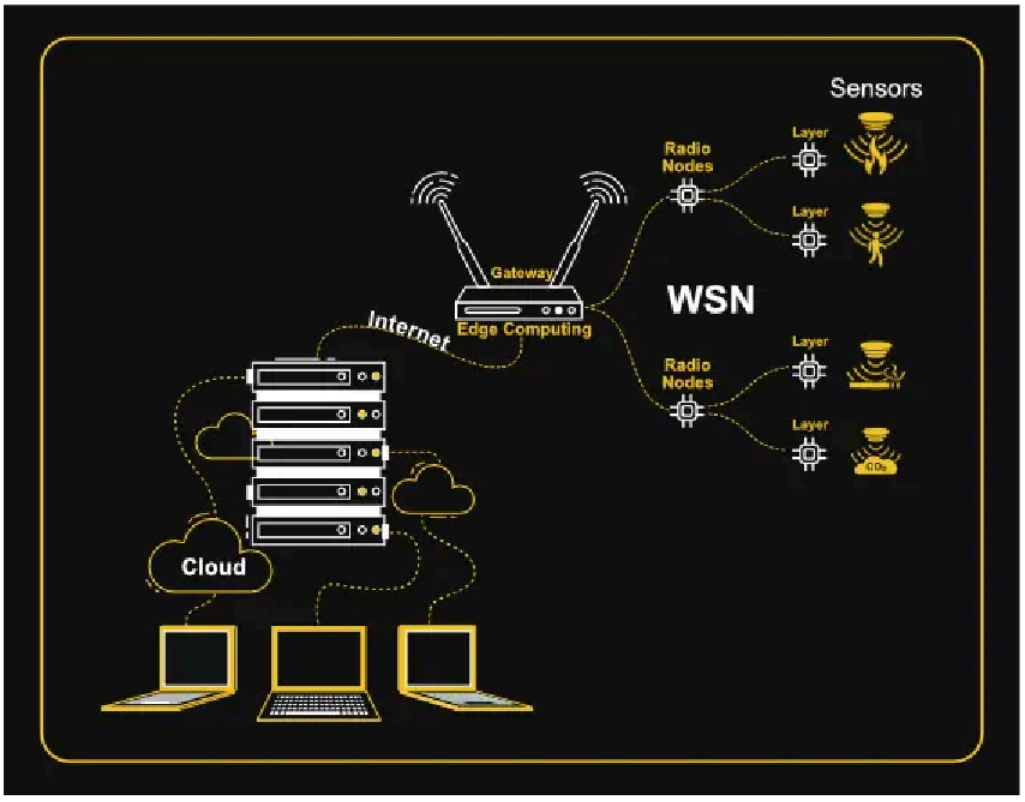
UNIT - 2 IOT Notes

UNIT 2: IoT Networking, Platforms, Sensors, and Actuators

1. Wireless Sensor Network (WSN) [PYQ Q5a, Q5b (June 2024)]

- **Definition:** A network of interconnected sensor nodes that communicate wirelessly to collectively gather and transmit data from their surrounding environment.
- **Node Capabilities:** Sensor nodes are equipped with sensors, processing capabilities, and wireless communication modules to monitor, collect, and transmit data from different points in a physical space.



- **Components of WSN:**
  - **Sensors:** Capture environmental variables for data acquisition; sensor signals are converted into electrical signals.
  - **Radio Nodes:** Receive data from sensors and send it to the WLAN access point. Consist of a microcontroller, transceiver, external memory, and power source.
  - **WLAN Access Point:** Receives data sent by radio nodes wirelessly (generally through the internet).
  - **Evaluation Software:** Processes data received by the WLAN Access Point for presentation and further processing (analysis, storage, mining).
- **Advantages of WSN:**

    - **Low cost:** Small, inexpensive sensors are easy to deploy, making them a cost-effective solution.
    - **Wireless communication:** Eliminates the need for wired connections, which can be costly and difficult to install; enables flexible deployment and reconfiguration.
    - **Energy efficiency:** WSNs use low-power devices and protocols to conserve energy, enabling long-term operation without frequent battery replacements.
    - **Scalability:** WSNs can be scaled up or down easily by adding or removing sensors.
    - **Real-time monitoring:** WSNs enable real-time monitoring of physical phenomena, providing timely information for decision-making.
- **Disadvantages of WSN:**
    - **Limited range:** The range of wireless communication can be a challenge for large-scale deployments or in environments with obstacles obstructing radio signals.
    - **Limited processing power:** WSNs use low-power devices, which may have limited processing power and memory, making complex computations difficult.
    - **Data security:** WSNs are vulnerable to security threats like eavesdropping, tampering, and denial of service attacks.
    - **Interference:** Wireless communication can be susceptible to interference from other wireless devices or radio signals.
    - **Deployment challenges:** Deploying WSNs can be challenging due to proper sensor placement, power management, and network configuration needs.
- **WSN Protocols (Layers - from PYQ Q5a June 2024):** [PYQ]
    - **Data-Link Layer Protocols:** Govern how data is framed, medium access control (e.g., MAC protocols like CSMA/CA, S-MAC, T-MAC), error detection/correction. IEEE 802.15.4 is a common standard.
    - **Network Layer Protocols:** Responsible for routing data packets from source to destination nodes. Examples: AODV, DSR, RPL (Routing Protocol for Low-Power and Lossy Networks - common in IoT/WSNs), or specialized WSN routing protocols (e.g., LEACH, TEEN).
    - **Security Protocols (in WSNs):** Mechanisms to ensure confidentiality, integrity, and authenticity. Examples: TinySec, SPINS, Link Layer Security in IEEE 802.15.4.
    - **Application Layer Protocols (in WSNs):** Define how application-specific data is exchanged. Can be custom, or lightweight protocols like CoAP, MQTT adapted for WSN constraints.
- **Node Behaviours in WSN (from PYQ Q5b June 2024):** [PYQ]
    - **Sensing Node:** Primary role is to sense physical parameters from the environment and collect data.
    - **Relay/Router Node:** Forwards data packets received from other nodes towards the sink or gateway. Crucial in multi-hop WSNs to extend network range.
    - **Sink Node (or Base Station/Gateway):** Collects data from all (or a subset of) sensor nodes in the WSN. Often acts as an interface to an external network (e.g., internet) for data transmission

to a server or cloud for further processing.

- **Coordinator Node:** (In some WSN architectures like Zigbee/IEEE 802.15.4 PANs) Responsible for network formation (initiating the network), management (assigning addresses, maintaining routing tables), and synchronization.
- **Sleeping/Active Node:** Nodes frequently alternate between low-power sleep states and active states (sensing, processing, communicating) to conserve energy, especially in battery-powered nodes. Duty cycling is a common behavior.
- **Data Aggregator Node:** May perform in-network processing, such as filtering redundant data or summarizing (aggregating) data from multiple nearby nodes before forwarding, to reduce communication overhead and save energy.
- **Cluster Head Node:** (In clustered WSN architectures like LEACH) A node elected to manage a cluster of sensor nodes, collecting data from its cluster members and transmitting aggregated data to the sink.

2. Participatory Sensing (PS)

- **Definition:** An approach to data collection and interpretation where individuals, acting alone or in groups, use their personal mobile devices (e.g., smartphones) and web services to explore and report interesting aspects of their worlds systematically, ranging from health to culture.
- **Core Idea:** Individuals and communities leverage mobile phones and cloud services to collect and analyze systematic data, contributing to discovery and forming a body of knowledge.
- **Phases of PS Process:**

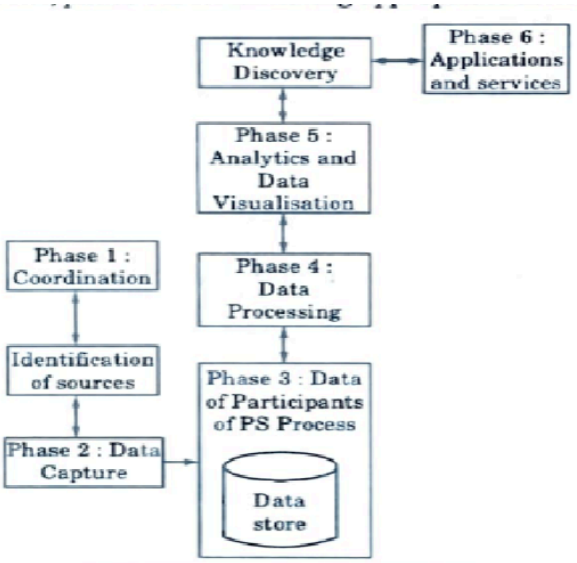


Fig. 2.13.1. Phases of PS process.

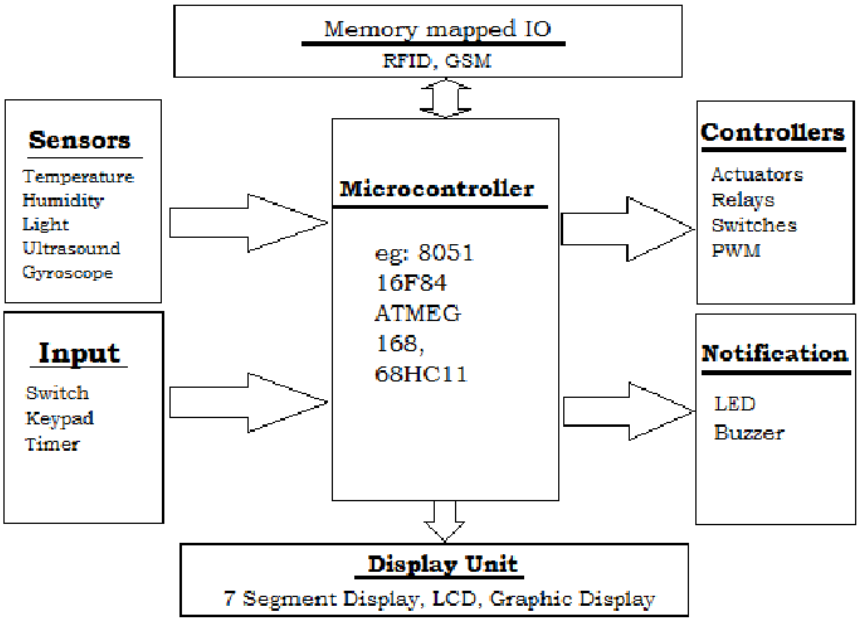
  - **Phase 1: Coordination:** Participants organize after identifying sources (or are recruited by an entity like city authorities). Objectives of the sensing campaign are communicated.
  - **Phase 2: Data Capture:** Participants use mobile phone applications or custom-designed applications to capture desired sensing modalities over a predetermined time.
  - **Phase 3: Data Transfer & Storage:** Collected data is transferred via phone connectivity options to data collection systems and stored on Internet servers (private or public).

- **Phase 4: Data Processing:** Data undergoes pre-processing to preserve the privacy of data collectors, and access control rules are added so data can be accessed only by authorized individuals/services.
- **Phase 5: Analytics and Data Visualization (Knowledge Discovery):** Collected data is analyzed using relevant tools, aggregated, correlated to detect patterns, and visualized for better understanding by the target group.
- **Phase 6: Action:** Appropriate actions may be taken by individuals or city authorities based on the insights and knowledge discovered.

- **Applications of PS:**
  - Retrieving information about weather, environment information, pollution levels.
  - Information for waste management, road faults.
  - Monitoring health of individuals and groups of people.
  - Assessing traffic congestion, urban mobility.
- **Challenges of PS:**
  - **Security:** Protecting the data and the devices involved.
  - **Privacy:** Ensuring the anonymity and confidentiality of participants and their data.
  - **Reputation:** Establishing trust in the data collected and the participants.
  - **Incentives:** Providing effective motivation for individuals to participate.

3. Embedded Devices (System) in IoT [PYQ Q1d (June 2024), Q2b (Dec 2024)]

- **Definition:** Objects that build a unique computing system, specifically designed to perform dedicated functions, often with real-time computing constraints. These systems may or may not connect to the Internet. They are essential for building IoT projects.



- **Components of Embedded Systems (Que 2.15):**

    - **1. Hardware:** Microcontroller (MCU) or Microprocessor (MPU), Memory (RAM, ROM, Flash), Sensors, Actuators, Input devices (switches, keypad, touchscreens), Output devices (LEDs, LCDs, motors, buzzers), Communication interfaces (UART, SPI, I2C, Ethernet, Wi-Fi, Bluetooth).
    - **2. Software:** Application-specific code (firmware) that defines the system's behavior and performs the dedicated tasks. Consists of instructions, commands, and data.
    - **3. Firmware:** Software programmed into non-volatile memory (e.g., ROM, Flash) of the embedded device. It often includes the bootloader, device drivers, and the main application logic.
- **Basic Concept of Embedded System (from PYQ Que 2.15):**
    - Embedding function software into computing hardware to enable a system function for specific dedicated applications.
    - A device embeds software into its computing and communication hardware, and the device functions for these applications.
    - **Key Software/Firmware Components (from Que 2.15):**
        - **Embedded Software:** The set of instructions, commands, and data that a computing and communicating device needs to operate.
        - **Bootloader:** A program that runs at the start of a computing device (e.g., MCU). It initiates the loading of system software (like an OS or the main application) when power is switched on and after the power-on-self-test (POST) completes. It may also facilitate the use of system hardware and networking capabilities for updates or diagnostics.
        - **Operating System (OS):** (Not always present, especially in simpler embedded systems). Facilitates the use of system hardware and networking capabilities. It manages memory allocation to different processes and, through prioritizing processes, enables the use of network and device hardware functions and the execution of software components.
        - **Real-Time Operating System (RTOS):** An OS designed to enable real-time execution of processes on computing and communication hardware. RTOS uses prioritization and priority allocation concepts to ensure timely execution of critical processes.
        - **Integrated Development Environment (IDE):** [PYQ Q1d (June 2024)] A set of software components and modules (tools) that provide the software and hardware environment for developing, debugging, and prototyping embedded applications. An IDE enables code development on a computer and facilitates the transfer (downloading/flashing) of compiled code to be executed on the target hardware platform. It often includes compilers, debuggers, and simulators, and enables software that communicates with internet web servers or cloud servers.
        - **Simulator:** Software that enables development and testing of embedded software on a computer without requiring the actual hardware. Prototyping hardware can then be connected for embedding the tested software and further validation.
        - **APIs (Application Programming Interfaces):** Software consists of device APIs and device interfaces for communication over the network and through communication circuits/ports.

This can also include middleware, which provides services beyond those offered by the OS to applications.

    - **Device Interfaces:** A connectivity interface consisting of communication APIs, physical device interfaces (ports, buses), and processing units.

- **Examples of Embedded Systems:** Digital watches, Washing Machines, Toys, Televisions, Digital phones, Laser Printers, Cameras, Industrial machines, Electronic Calculators, Automobiles, Medical Equipment.
- **Application of Embedded System:** Home appliances, Transportation, Health care, Business sector & offices, Defense sector, Aerospace, Agricultural Sector.
- **Characteristics of an Embedded System:**
    - **Performs specific task:** Designed to perform one or a few dedicated functions or tasks.
    - **Low Cost:** The price of an embedded system is often relatively low, especially for mass-produced devices.
    - **Time Specific (Real-time constraints):** Often performs tasks within a certain, strict time frame; critical for control systems.
    - **Low Power:** Embedded Systems often need to operate on limited power sources (like batteries) and are designed for low power consumption.
    - **High Efficiency:** The efficiency level of embedded systems is generally high due to specialization.
    - **Minimal User interface:** These systems often require less user interface (or no UI) and are designed to be easy to use or operate autonomously.
    - **Less Human intervention:** Embedded systems typically require no human intervention or very little once configured.
    - **Highly Stable:** Embedded systems usually do not change frequently; they are mostly fixed, maintaining operational stability.
    - **High Reliability:** Embedded systems are generally reliable and perform their tasks consistently well.
    - **Use microprocessors or microcontrollers:** Embedded systems use microprocessors or microcontrollers to design and utilize limited memory.
    - **Manufacturable:** The majority of embedded systems are compact and affordable to manufacture, based on their size and the low complexity of their hardware.
- **Advantages of Embedded System:**
    - Small size.
    - Enhanced real-time performance.
    - Easily customizable for a specific application.
- **Disadvantages of Embedded System:**
    - High development cost (for initial design and tooling).

- Time-consuming design process.
- As it is application-specific, there might be a less broad market available for a particular design.

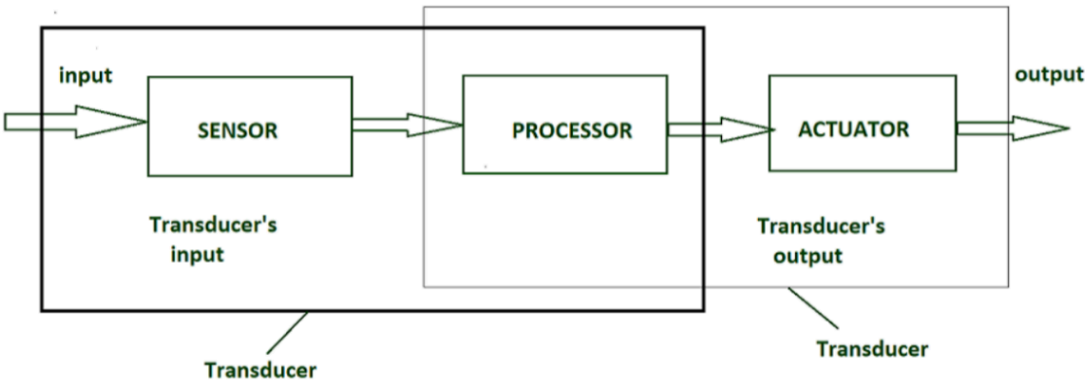
4. Sensors (Detailed from "Sensors_Ppt" - Doc 6, 7, 8, 9) [PYQ Q1c, Q1e (June 2024)]

- **Introduction to Sensors:**
  - Crucial components in IoT, providing data for intelligent interaction with surroundings.
  - Detects changes in the environment and provides output (analog or digital signals).
  - Converts physical phenomena (temperature, pressure, motion) into measurable electrical signals.
  - Enable technological systems to "see," "hear," "feel," "smell," or "taste."
  - **Sensor (Characteristic):** Characteristic of a device/material to detect a particular physical quantity.
  - **Output:** Signal converted to human-readable form.
  - **Function:** Senses/feels physical changes in response to stimuli.
  - **Role in IoT:** Bridge physical and logical worlds, enabling data collection, analysis, and monitoring.

    - A Sensor is a characteristic of any device or material to detect the presence of a particular physical quantity.
    - The output of the sensor is a signal, which is converted to human readable form.

- **Sensors in Laymen Terms:**
  - Most systems use something to sense their physical environment.
  - Human body example: nose, eyes, ears, skin, tongue sense light, sound, heat, taste.
  - Artificial sensors sense parameters like light, sound, vibration, temperature.
- **Transducer vs. Sensor:** [PYQ Q1c (June 2024)]
  - **Transducer:** Converts energy from one form to another. Converts a physical parameter into a corresponding electrical/electronic signal.
    - *Example:* Microphone converts sound to electrical signal. Based on conservation of energy.

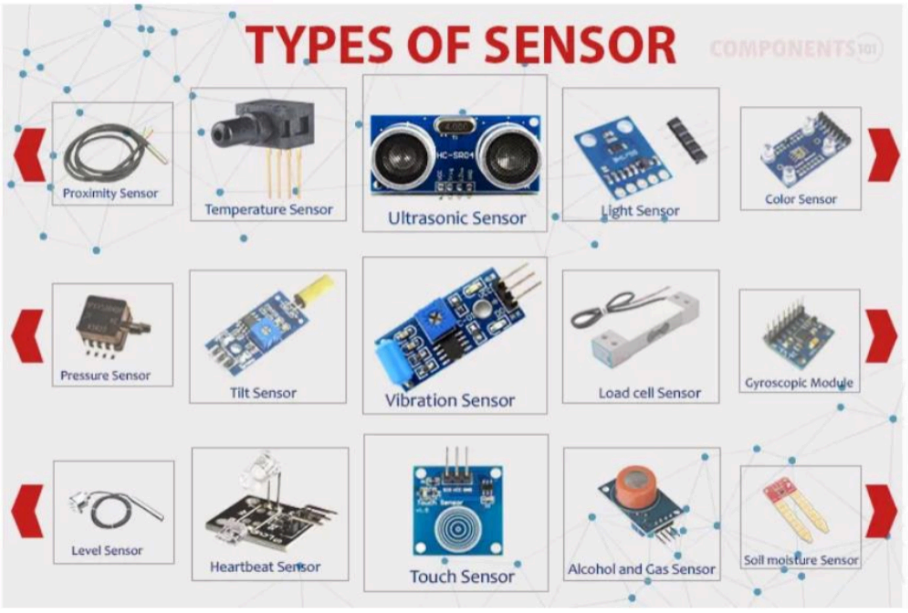
      - **Transducer** : It converts the signal from one physical form to another physical form. it is also called energy converter. For example, microphone converts sound to electrical signal . It is based on the principle of conservation of energy.

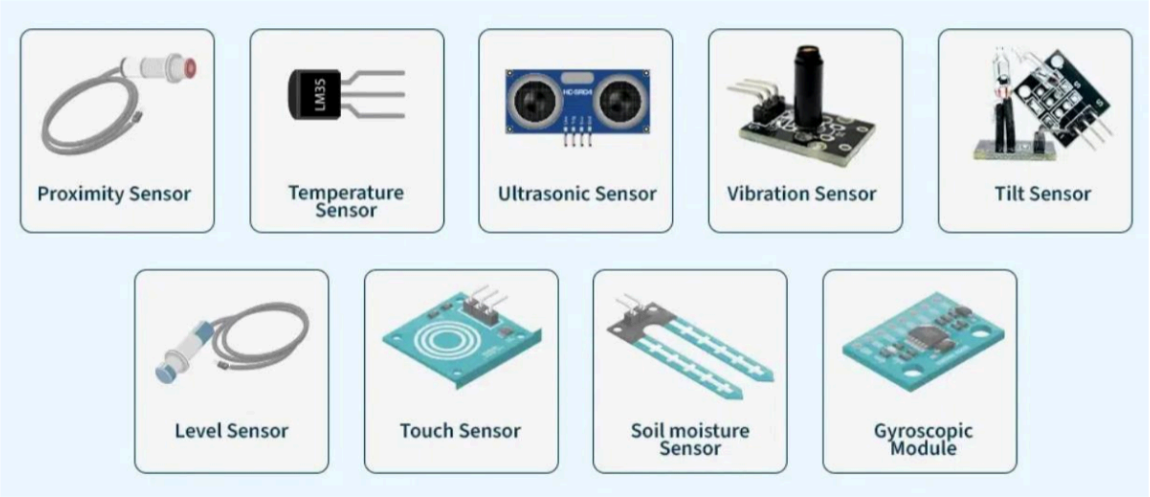


- **Distinction (from Page 62, Doc 6):**
  - Sensors detect changes and provide feedback; transducers convert physical quantities into electrical signals (or vice-versa).
  - Sensors often require power for operation; some transducers can function without external power sources (e.g., piezoelectric).

- **Key Characteristics of Sensors (from Page 63, Doc 6):**
  - **Accuracy:** Degree to which sensor's measurement corresponds to the actual value.
  - **Sensitivity:** Ability of the sensor to detect small changes in the measured quantity.
  - **Range:** Span of values over which the sensor can accurately measure.
  - **Resolution:** Smallest change in the measured quantity that the sensor can detect.
  - **Response Time:** Time it takes for the sensor to respond to a change in the measured quantity.
  - **Linearity:** Degree to which the sensor's output is directly proportional to the measured quantity.

- **Types of Sensors (General Overview & Applications from Pages 64-73, Doc 6):**





- **1. Temperature Sensors:** Measure temperature changes. *Apps:* Thermostats, weather monitoring, industrial processes, medical devices. (LM35 analog, DS1621 digital).
- **2. Pressure Sensors:** Measure pressure of gases or liquids. *Apps:* Weather forecasting, automotive (tire pressure), industrial machinery.
- **3. Proximity Sensors:** Detect presence/absence or distance of an object. *Apps:* Mobile phones, parking sensors, industrial automation.
- **4. Accelerometers:** Measure acceleration forces (and indirectly velocity, displacement, vibration). *Apps:* Smartphones (screen orientation), fitness trackers, vehicle dynamics, car electronics, ships, agricultural machines.
- **5. Light Sensors:** Detect light intensity. *Apps:* Smartphones (auto brightness), outdoor lighting, photography.
- **6. Motion Sensors:** Detect movement. *Apps:* Security systems, automatic doors, gaming consoles.
- **7. Humidity Sensors:** Measure moisture level in the air. *Apps:* HVAC systems, weather stations, agricultural monitoring.
- **8. Gas Sensors:** Detect presence/concentration of gases. *Apps:* Air quality monitoring, industrial safety, medical diagnostics, atmospheric analysis.
- **9. Magnetic Sensors:** Detect magnetic fields. *Apps:* Compass navigation, industrial applications, consumer electronics.
- **10. Sound Sensors:** Detect sound levels or specific audio frequencies. *Apps:* Voice recognition, hearing aids, security alarms.
- **11. Alcohol Sensors:** Detect alcohol. *Apps:* Breathalyzer devices.
- **12. Radiation Sensors:** Detect alpha, beta, gamma particles. *Apps:* Surveys, sample counting.
- **13. Image Sensors:** For distance measurement, pattern matching, color checking, 3D imaging, video, security, medical vision.
- **14. Position Sensors:** Sense positions of valves, doors, throttles. *Apps:* Control applications (string-pot).
- **15. Torque Sensors:** Measure rotating torque and speed of rotation.

- **16. Optical Sensors (Photosensors):** Detect light waves across spectrum (UV, visible, IR). *Apps:* Smartphones, robotics, Blu-ray players.
- **17. Touch Sensors:** Detect physical contact. *Apps:* Trackpads, touchscreens, elevators, robotics, soap dispensers.

- **Sensor Classification (from Pages 77-80, Doc 6):**
  - **1. Based on Power Requirement:**
    - **Active Sensors:** Require an external excitation signal or power source to work.
    - **Passive Sensors:** Do not require any external power source; directly generate output response.
  - **2. Based on Means of Detection:**
    - Sensors can use electrical, biological, chemical, or radioactive detection methods.
  - **3. Based on the Conversion Phenomenon:** (Input to output conversion type)
    - **Photoelectric:** Changes light to electrical signals.
    - **Thermoelectric:** Changes temperature difference to electrical voltage.
    - **Electrochemical:** Changes chemical reactions to electrical signals.
    - **Electromagnetic:** Changes magnetic fields to electrical signals.
    - **Thermoptic:** Changes temperature changes to electrical signals.
  - **4. Based on Output Type:**
    - **Analog Sensors:** Produce an output signal (voltage, current, resistance) proportional to the measured quantity.
    - **Digital Sensors:** Provide discrete or digital data as output (usually binary).

Feature	Digital Sensors	Analog Sensors
Signal Type	Digital (discrete)	Analog (continuous)
Signal Output	Discrete digital signal (usually binary)	Continuous analog voltage or current
Accuracy	Higher accuracy, less susceptible to noise	Generally less accurate due to noise and drift
Complexity	More complex due to signal processing	Simpler design and circuitry
Interference	Less prone to electromagnetic interference (EMI)	More prone to EMI

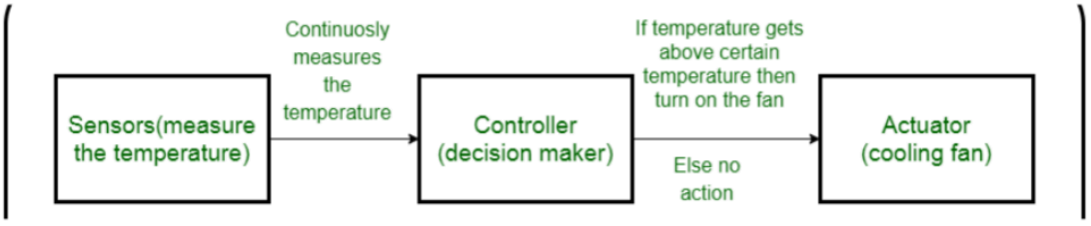
- **How do Sensors work? (from Pages 81-82, Doc 6):**
  - React to changing physical conditions by changing their electrical properties.
  - Often rely on electronic systems for analyzing, capturing, relaying environment information.
  - Convert stimuli (sound, motion, heat, light) into electrical signals.
  - Signals passed through an interface, converted to binary code for computer processing.

    - Many act as switches, controlling electric charge flow.
    - Components (transistors, diodes, ICs) use semiconducting material as switches.
    - Often use radiation (laser, light, IR, ultrasonic) to detect changes/objects.
        - **Active Sensor:** Emits radiation (from an energy source) and detects reflected radiation.
        - **Passive Sensor:** Detects radiation emitted by target objects (thermal IR, heat) or reflected from external sources (Sun).
- **Sensor Dynamics and Specifications (from Pages 83-86, Doc 6):**
    - **1. Sensor Dynamics:** Behavior and response characteristics of a sensor over time.
        - **Key aspects:**
            - **Response Time:** Time to reach stable reading after a parameter change.
            - **Sensitivity:** Degree of output change in response to input parameter change.
            - **Linearity:** Ideal linear relationship between input and output.
            - **Hysteresis:** Difference in output for increasing vs. decreasing parameter.
            - **Range:** Span of values a sensor can measure.
            - **Resolution:** Smallest detectable change in the measured parameter.
    - **2. Sensor Specifications:** Detailed information about capabilities and limitations.
        - **Common specifications:**
            - **Accuracy:** How close measurements are to the true value.
            - **Precision:** Consistency of measurements over repeated trials.
            - **Drift:** Change in sensor output over time (due to temperature, aging, etc.).
            - **Power Consumption:** Amount of power required to operate.
            - **Operating Temperature:** Range of temperatures for effective operation.
            - **Calibration:** Process of setting/correcting output to match a known standard.
            - **Noise:** Unwanted variations in output not related to the measured parameter.
            - **Environmental Resistance:** Ability to withstand moisture, dust, chemicals.
- **Smart Sensor vs. Sensor Node:** [PYQ Q1e (June 2024)]
    - **Sensor Node:** A fundamental component of a WSN. Typically consists of sensor(s), MCU, transceiver, power source. Primary role: collect and transmit data.
    - **Smart Sensor:** A sensor with integrated electronics for signal conditioning, data conversion (A/D), some processing/decision-making (calibration, self-diagnosis), and communication interface.
    - **Difference:**
        - A sensor node *contains* sensors; it's a networked entity. A smart sensor *is* a sensor with enhanced capabilities.

        - Smart sensors offer more processed/refined data, reducing load on the node's MCU or gateway.
        - Sensor node focuses on networking; smart sensor on the sensing element's capability.
        - A gateway can add "smart" processing if the sensor itself isn't inherently smart.

5. Actuators (Detailed from "Actuators_Ppt" - Doc 7, 8, 9) [PYQ Q1c (June 2024), Q2b (Dec 2024)]

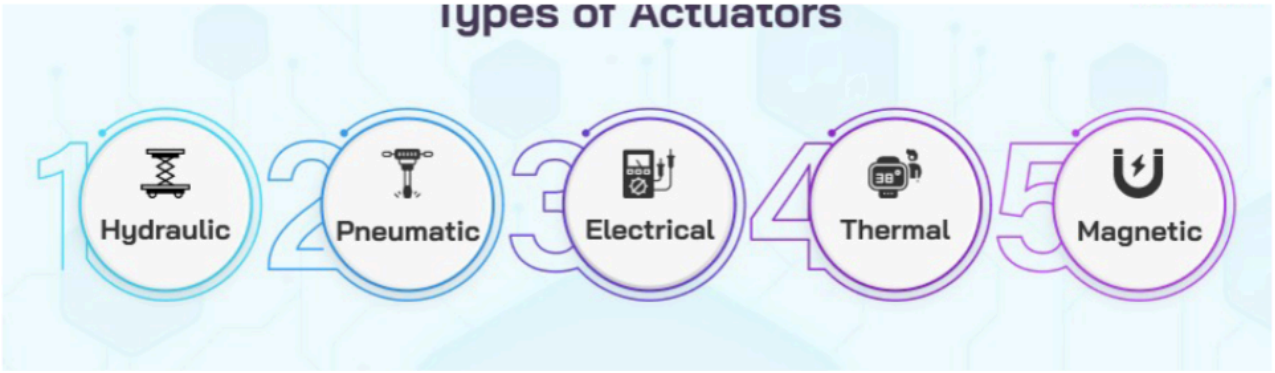
- **Brief Glimpse of Actuators:**
    - An IoT device = Physical object ("thing") + Controller ("brain") + Sensors + Actuators + Networks (Internet).
    - Machine component or system that moves or controls the mechanism of a system.
    - Control signals generated (based on sensor data) for actuators to perform actions.
- **What are Actuators?**
    - Device that changes electrical signals into mechanical work (movement or change in surroundings).
        - *Example:* Fan lowers temperature, servo motor changes position.
    - Connected to a system's output; receives electrical signal as input, produces mechanical movement as output.
    - Receives input/instruction from a system or signal conditioning unit and outputs to the environment.
    - Actuator is dependent on sensor data. Sensor sends data to signal conditioning unit, which commands actuator.
        - *Example:* Temperature control system: sensor manages temp; if limit surpassed, actuator (fan) instructed.



- **Actuators (General Definition from Page 27, Doc 7):**
    - Converts energy into motion.
    - Receives a control signal and uses it to move/control a mechanism or system.
    - **Energy Sources:** Electrical, hydraulic, pneumatic, thermal, magnetic, or mechanical energy.
    - **Types of Motion:** Linear (straight-line), Rotary (rotational).
    - **Control Input:** Operate based on digital or analog control signals.
    - **Applications:** Robotics, industrial machinery, automotive systems, household devices.
- **Features of Actuators (from Page 29, Doc 7):**

- Assists in managing the environment based on sensor readings.
- Converts electrical signals into mechanical movement.
- Requires an additional power source to function.
- Receives an electrical signal as input.
- Connected to a system's output.
- Produces mechanical work.

- **Types of Actuators (from Pages 30-35, Doc 7 and Page 21, Doc 1):**



  - **1. Manual Actuator:** Manually operated via gears, levers, wheels. Powered by human action.
  - **2. Spring Actuator:** Loaded spring triggered/released to generate mechanical work.
  - **3. Hydraulic Actuators:** Use hydraulic power (compressed fluid in a cylinder) to perform mechanical operation (rotary, linear, oscillatory). Generate large force.
    - *Advantages:* Large force, high speed; used in welding, clamping, vehicle lifts.
    - *Disadvantages:* Fluid leaks (efficiency loss, cleaning); expensive; needs noise reduction, heat exchangers, high maintenance.
  - **4. Pneumatic Actuators:** Use energy from vacuum or compressed air at high pressure for linear/rotary motion.
    - *Example:* Robotics (sensors like human fingers using compressed air).
    - *Advantages:* Low-cost; used at extreme temps (air safer than chemicals); low maintenance, durable, long life; quick start/stop.
    - *Disadvantages:* Pressure loss (less efficient); air compressor runs continuously; air can be polluted, needs maintenance.
  - **5. Electric Actuators (Electrical):** Uses electrical energy, usually actuated by a motor converting electrical energy into mechanical torque.
    - *Example:* Solenoid based electric bell.
    - *Advantages:* Automates industrial valves; less noise, safe (no fluid leaks); re-programmable, highest control precision.
    - *Disadvantages:* Expensive; depends on environmental conditions.
  - **6. Thermal/Magnetic Actuators:** Actuated by thermal or mechanical energy. Shape Memory Alloys (SMAs) or Magnetic Shape-Memory Alloys (MSMAs) used.

        - *Example:* Piezo motor using SMA.
    - **7. Mechanical Actuators:** Executes movement by converting rotary to linear motion. Involves pulleys, chains, gears, rails. *Example:* Crankshaft.
- **Different Classes of Actuators (from Pages 36-38, Doc 7):**
    - **1. Electric Actuators:**
        - *DC Motors:* Continuous rotation, precise speed control.
        - *AC Motors:* Industrial apps for high power/efficiency.
        - *Stepper Motors:* Precise control over movement/positioning.
        - *Solenoids:* Create linear motion, often in locking mechanisms.
    - **2. Hydraulic Actuators:**
        - Use pressurized hydraulic fluid for motion.
        - High force, common in heavy machinery.
        - *Examples:* Hydraulic cylinders, hydraulic motors.
    - **3. Pneumatic Actuators:**
        - Use compressed air for motion.
        - Widely used in automation/industrial for speed/reliability.
        - *Examples:* Pneumatic cylinders, pneumatic motors.
    - **4. Thermal and Magnetic Actuators:**
        - *Thermal Actuators:* Rely on temperature changes to induce motion (thermostats).
        - *Magnetic Actuators:* Utilize magnetic fields for motion (magnetic levitation).
    - **5. Mechanical Actuators:**
        - Convert manual input into mechanical movement.
        - *Examples:* Gears, pulleys, levers, cams.
    - **6. Piezoelectric Actuators:**
        - Use piezoelectric materials that change shape with applied electric field.
        - Ideal for precision applications (micro-positioning).
- **Challenges of Employing Actuators in IoT (from Page 39, Doc 7):**
    - **Compatibility:** May not be interoperable with all IoT devices/systems.
    - **Precision:** May not always provide precise control for certain applications.
    - **Power Consumption:** Can consume substantial power (problem for battery-reliant IoT).
    - **Maintenance:** Require routine maintenance (time-consuming, costly).
    - **Cost:** Can be expensive.
- **Few Examples of Actuators in IoT (from Pages 40-41, Doc 7):**

- **Smart Home Systems:** Control lights, heating, security. Smart thermostat uses temp sensors -> actuator for heating/cooling.
- **Industrial Automation:** Control machines. Hydraulic for robotic arms, electric for conveyor belts.
- **Agriculture:** Automate irrigation (control water flow), harvesting (robotic arm position).
- **Healthcare:** Prostheses, medical equipment (regulate prosthetic limb movement, surgical tool location).

- **Key Difference between Sensors & Actuators (from Pages 42-45, Doc 7):** [PYQ Q1c (June 2024)]

Feature	Sensors	Actuators
Definition	Detects changes/events in environment, transmits data to electronic devices.	Machine component that moves and controls mechanisms.
Basic Function	Converts physical properties of environment into electrical signals.	Converts system's electrical signals into physical characteristics for environments.
Type of Output	Electrical signals are generated.	Generates energy in the form of heat or motion.
Source of Input	Receives input from the environment.	Receives input from the system's output conditioning unit.
Placement	Placed at a system's input port.	Placed at a system's output port.
Output Generation	Produces output for the input conditioning unit of a system.	Produces output for their environment.
Examples	Magnetometer, cameras, microphones, biosensors, image sensors, chemical.	LEDs, loudspeakers, motor controllers, lasers, electric motors, comb drives, hydraulic.

  - Sensors measure physical quantities; Actuators measure discrete/continuous process parameters (this point seems a bit off, actuators *act* based on parameters). Actuators *control* parameters.

6. Basis of Hardware Design Needed to Build Useful Circuits Using Basic Sensors and Actuators (from Pages 46-51, Doc 7)

- **Fundamental Principles:** Clear understanding of components, interactions, signal processing, and control.
- **1. Understanding Sensors and Actuators:** (Types already covered)
- **2. Signal Conditioning:**
  - *Amplification:* Op-amps for low-level sensor signals.
  - *Filtering:* Low-pass, high-pass, band-pass filters to remove noise.
  - *Analog-to-Digital Conversion (ADC):* For analog sensor outputs to digital for MCUs.

- *Level Shifting:* Match voltage levels between circuit elements (esp. MCU interfacing).

- **3. Power Management:**
  - *Power Supply Design:* Stable and suitable power for sensors/actuators.
  - *Voltage Regulation:* Maintain constant voltage for sensitive sensors.
  - *Battery Management:* Low-power design techniques (sleep modes, reduced duty cycles).
- **4. Interfacing Sensors:**
  - *Direct Connection:* Simple sensors (switches, LDRs) with pull-up/down resistors to MCU.
  - *Wheatstone Bridge:* Precision measurement for sensors like strain gauges.
  - *Communication Protocols:* I2C, SPI, UART for digital sensors, requiring proper interfacing.
- **5. Interfacing Actuators:**
  - *Transistor/MOSFET Switches:* Drive actuators needing more current than MCU can provide.
  - *H-Bridge Circuits:* Control DC motor direction.
  - *Relays:* Low-power circuit controls higher power circuit (motors, lights).
  - *PWM (Pulse Width Modulation):* Control motor speed or LED brightness by varying duty cycle.
- **(Microcontroller Integration - Implied, but essential for processing sensor data and controlling actuators based on logic.)**
- **7. Safety Considerations:** (Item 6 not in slides, directly jumps to 7)
  - *Overcurrent Protection:* Fuses, resettable fuses, current limiters.
  - *Isolation:* Optocouplers, transformers for high voltage/current circuits.
  - *Debouncing:* Hardware (capacitors) or software for mechanical switches to prevent false triggering.
- **8. Prototyping and Testing:**
  - *Breadboarding:* Quick prototyping and testing.
  - *Simulation Tools:* Proteus, TinkerCAD, LTSpice to simulate circuit behavior.
  - *Testing Tools:* Oscilloscopes, multimeters, logic analyzers for debugging/verifying.
- **9. Environmental Considerations:**
  - *Temperature Tolerance:* Ensure components are rated for environmental conditions.
  - *Protection Against Humidity and Dust:* Enclosures, conformal coatings.
  - *Electromagnetic Interference (EMI):* Design for minimal interference (grounding, shielding).
- **10. Documentation and Planning:**
  - *Circuit Schematics:* Clear documentation of design, connections, components.
  - *Bill of Materials (BOM):* List all components, specifications, suppliers.
  - *Testing Procedures:* Define clear procedures to verify circuit works as intended.

7. NFC and RFID Protocols for Device Communication in IoT [PYQ Q4a (June 2024)]

- *(This content is drawn from Unit 1 notes, Section 23, as it's now a Unit 2 PYQ).*
- **NFC (Near Field Communication):**
  - **Principle:** Short-range (typically 0-5 cm, up to 20 cm) wireless communication technology based on RFID principles, operating at 13.56 MHz. Enables two-way communication.
  - **Modes:** Card emulation (device acts like a smart card), reader/writer (device reads/writes to NFC tags), peer-to-peer (two NFC devices exchange data).
  - **Data Rate:** Up to 424 kbit/s.
  - **Power:** Passive tags are powered by the reader's field; active devices have their own power.
  - **Setup:** Very fast connection setup (tap-to-connect).
  - **Security:** Short range provides inherent physical security. Supports encryption.
  - **Common IoT Uses:** Mobile payments, access control (e-keys), smart posters/tags for information retrieval, device pairing (e.g., Bluetooth), inventory tracking (item-level), authentication.
- **RFID (Radio Frequency Identification System):**
  - **Principle:** Uses electromagnetic fields to automatically identify and track tags attached to objects. Tags contain electronically stored information.
  - **Components:**
    - *Tag (Transponder):* Microchip with an antenna. Can be passive (powered by reader's signal, shorter range, cheaper) or active (battery-powered, longer range, more expensive, can include sensors).
    - *Reader (Interrogator):* Transmits radio waves to activate tags and read/write data.
    - *Antenna:* Emits radio signals to activate the tag and read/write data.
  - **Frequency Bands:** Low Frequency (LF: ~125-134 kHz), High Frequency (HF: 13.56 MHz - same as NFC), Ultra-High Frequency (UHF: ~860-960 MHz), Microwave (~2.45 GHz, ~5.8 GHz). Range and characteristics vary significantly with frequency.
  - **Data Rate:** Varies widely depending on frequency and tag type.
  - **Security:** Varies; some tags have minimal security, others support encryption and authentication. Susceptible to cloning or unauthorized reading if not secured.
  - **Common IoT Uses:** Supply chain management, inventory tracking (pallet/case level), asset management, livestock tracking, toll collection, library systems, access control, healthcare (patient/equipment tracking).
- **Comparison: NFC vs. RFID [PYQ Q4a (June 2024)]**

Feature	NFC (Near Field Communication)	RFID (Radio Frequency Identification)
Communication Range	Very short (typically < 10 cm, max ~20 cm)	Variable: Short (cm) to Long (meters), depending on frequency & tag type (passive/active)

Feature	NFC (Near Field Communication)	RFID (Radio Frequency Identification)
Communication Type	Two-way (peer-to-peer possible)	Primarily one-way (reader to tag, tag response), though some systems allow writing to tags
Frequency	13.56 MHz (standardized)	Multiple bands (LF, HF, UHF, Microwave)
Setup	Simple, intuitive "tap" or "touch"	Requires reader to be within range of tag; may need line-of-sight for some UHF
Power Source (Tag)	Passive tags powered by reader; active devices have own power.	Passive (powered by reader), Active (battery-powered)
Data Rate	Up to 424 kbit/s	Varies widely (few kbit/s to Mbit/s for active tags)
Cost	Readers/chips can be slightly more complex/costly than simple RFID. Tags are generally low cost.	Passive tags very cheap; Active tags more expensive. Readers vary.
Primary Use Cases	Secure short-range transactions, smart posters, device pairing, fine-grained interaction.	Asset tracking, inventory management, logistics, identification over varied distances.
Security	Inherent physical security due to short range; supports encryption.	Varies; can be vulnerable if not implemented securely. Active tags can have better security.
Standardization	Well-standardized globally.	Multiple standards exist for different frequencies and applications.
Interaction	Often user-initiated (e.g., tapping a phone).	Can be automated (e.g., passing through a reader portal).

    - **Key Distinction:** NFC is essentially a specialized subset of HF RFID, offering more complex interactions and peer-to-peer capabilities over a very short range. RFID is a broader term covering various frequencies and primarily focused on identification and tracking.

8. Overview of IoT Hardware Platforms (Arduino, Raspberry Pi, etc.) [PYQ Q4b (June 2024), Q2a (Arduino sections from Dec 2024)]

- **Introduction:**
    - Selection of a hardware platform is crucial for IoT project development, depending on project complexity, processing needs, connectivity requirements, and ease of use.
    - Common platforms include Arduino, Raspberry Pi, Netduino, BeagleBone, and Intel Galileo, each with distinct features and target applications.
- **A. Arduino [PYQ Q8 (June 2024), Q2a (Dec 2024)]**

- **Definition (from Page 10, Doc: Unit 2):** An open-source electronics platform that is easy to use and is designed for beginners. It is based on a microcontroller and provides an environment for building and programming projects in a simple and intuitive way.

- **Usage in IoT:** The Arduino platform is widely used for building IoT projects because it is low-cost and has a vast community of developers.
- **Programming:** Arduino boards can be programmed using the Arduino IDE (Integrated Development Environment) which is available for Windows, Mac, and Linux.
- **Basic Features (from Page 10, Doc: Unit 2):**
  - Many different types of Arduino boards available, all share basic features.
  - Have digital and analog inputs and outputs, allowing interaction with a wide range of sensors and actuators.
  - Have built-in communication protocols such as USB, UART, SPI, and I2C, allowing them to communicate with other devices.
- **Arduino Platform for IoT (Details from Que 2.17, Page 12, Doc: Unit 2):**
  - Arduino boards, modules, and shields are popular AVR MCU-based products.
  - Each board has clear markings on the connection pins, sockets, and in-circuit connections.
  - Arduino boards are easy to program. For example, the Arduino Uno board is robust and widely used for getting started with electronics and coding.
  - The analog input pins and PWM (Pulse Width Modulation) pins on the board can connect sensors, actuators, and analog circuits.
  - The digital I/O pins can connect to On-Off states, handle digital input from sensors, and provide digital outputs to actuators and other digital circuits.

- **Architecture of Arduino Board (Uno Example from Fig. 2.17.1, Page 12, Doc: Unit 2):**

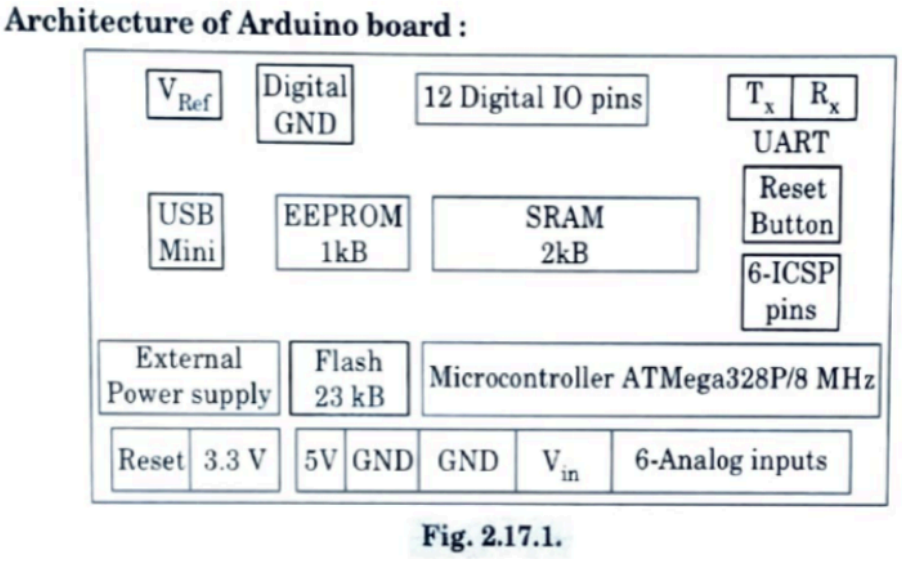


Fig. 2.17.1.

  - **Key Components Illustrated:**
    - **Microcontroller:** ATMega328P (typically running at 8 MHz or 16 MHz).
    - **Digital I/O Pins:** 12 Digital IO pins (some with PWM capability).
    - **Analog Input Pins:** 6 Analog inputs.
    - **Memory:**
      - EEPROM: 1kB (for persistent storage).
      - SRAM: 2kB (for variable storage during runtime).
      - Flash Memory: 23kB (typically 32kB for ATmega328P, part used by bootloader) for program code.
    - **Communication:**
      - USB Mini (for programming and serial communication).
      - Tx/Rx Pins (for UART serial communication).
      - ICSP Pins (In-Circuit Serial Programming).
    - **Power:**
      - VRef (Analog Reference Voltage).
      - Digital GND.
      - External Power supply jack.
      - Reset pin/button.
      - 3.3V and 5V power outputs.
      - GND pins.
      - Vin (Input Voltage).
    - **Other:** Reset Button.

- **B. Raspberry Pi [PYQ Q4b (June 2024)]**

- **Definition (from Page 10, Doc: Unit 2):** A low-cost, credit-card-sized computer that was designed for education and experimentation. It is based on an ARM processor and is capable of running a full-fledged operating system.

- **Power & Suitability:** Raspberry Pi boards are more powerful than Arduino boards and have more RAM and processing power. This makes them suitable for more complex IoT projects.
- **Community & OS & Programming (from Page 11, Doc: Unit 2):**
  - Large community of developers and enthusiasts.
  - Can run a variety of operating systems including Raspbian (now Raspberry Pi OS), Ubuntu, and Windows 10 IoT Core.
  - Can be programmed using a variety of programming languages including Python, JavaScript, and C/C++.
- **Connectivity & Graphics (from Page 11, Doc: Unit 2):**
  - More built-in connectivity options such as Ethernet, Wi-Fi, and Bluetooth.
  - Has a built-in HDMI output, making it easy to connect to a monitor or TV.
  - These features make Raspberry Pi a popular choice for building IoT projects that require more connectivity options and graphics capabilities.
- **Basic Setup Requirements (from Que 2.20, Page 13, Doc: Unit 2):**
  - HDMI cable, monitor, keyboard, mouse, 5-volt power adapter, LAN cable, 2 GB micro SD card (minimum).
- **Architecture of Raspberry Pi Board (from Fig. 2.20.1, Page 14, Doc: Unit 2):**

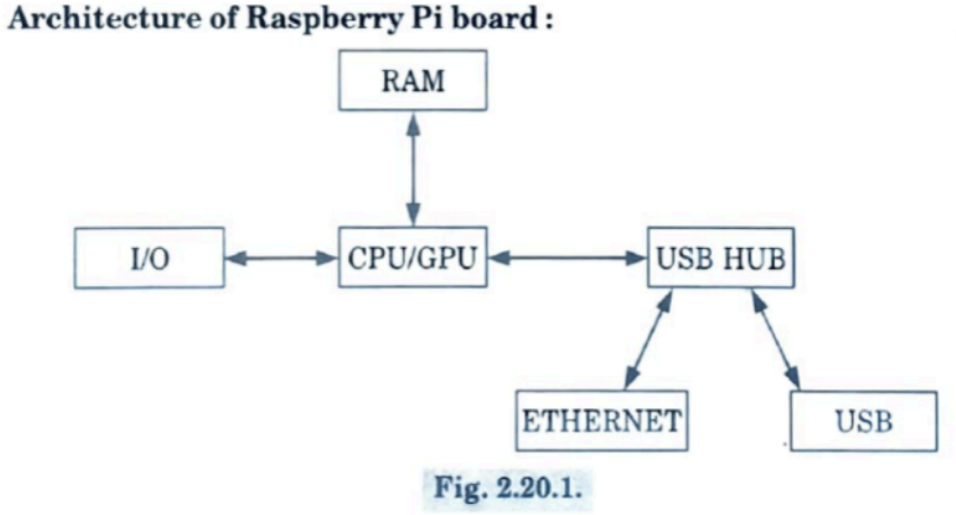
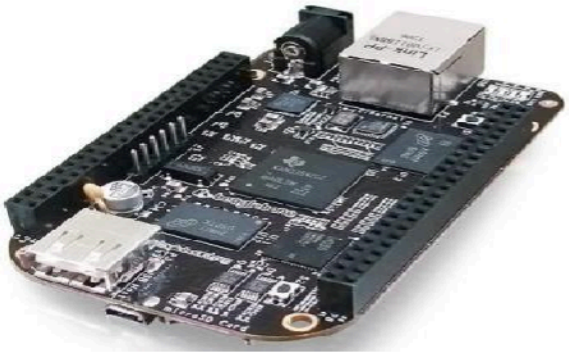


Fig. 2.20.1.

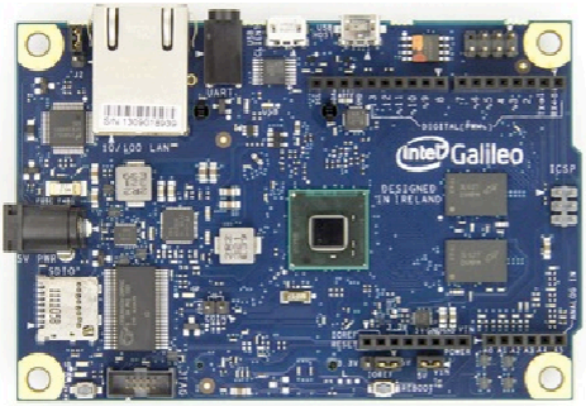
        - **Key Components Illustrated:**
            - **RAM**
            - **CPU/GPU** (Central Processing Unit / Graphics Processing Unit)
            - **I/O** (General Purpose Input/Output pins)
            - **USB HUB** (leading to multiple USB ports)
            - **ETHERNET** port
            - **USB** ports (as peripherals)
    - **Features making Raspberry Pi widely used (from Que 2.21, Page 14, Doc: Unit 2):**
        - Computer-like prototyping, easy for developing media servers.
        - Coding in Python, C++ and access to extensive libraries.
        - Software runs on multiple environments/OSes (Python, Scratch, Squeek, IDLE, C, Linux, BSD OSes, Windows 10).
        - Flexibility and ease of connecting hardware to external systems; connectivity via two USB host hubs and Ethernet connector.
        - Connectivity to extended memory through a micro SD slot.
- **C. Netduino [PYQ Q4b (June 2024)]**
    - **Definition (from Que 2.19, Page 13, Doc: Unit 2):** An open-source electronics prototyping platform based on the .NET micro-framework.
    - **Processor:** Uses the ARM Cortex-M 32-bit RISC ARM processor core as a 32-bit ARM-microcontroller.
    - **Compatibility:** The Netduino boards are designed to be pin-compatible with most Arduino shields.
    - **Development Environment:** Applications can be built on Windows (with Visual Studio), or on Mac OS (with Xamarin Studio).
    - **Power:** It is more powerful than the Arduino platform.
    - **Programming Language:** Programming used to embed Netduino is written in C# (C sharp), which makes it more powerful and allows use of high-level language constructs.
    - **Board Features:** Consists of 22 General Purpose Input/Output (GPIO) ports with 6 Pulse Width Modulation (PWM) hardware outputs, 4 UARTs (serial communication), I2C, and SPI (Serial Peripheral Interface Bus), and 6 ADC channels.
- **D. BeagleBone [PYQ Q4b (June 2024)]**
    - **Definition (from Que 2.22, Page 15, Doc: Unit 2):** BeagleBone X15 (BB-X15) is a single-board computer for computing and communication. (Note: BeagleBone Black is another popular

variant).



- **Operating Systems:** BB runs on OS Linux, RSIX OS, FreeBSD, OpenBSD, and additional distributions of Linux such as Ubuntu boards.
- **SoC (System on Chip):** Uses processor (ARM Cortex A15 Core for BB-X15) and DSP processor (TMS320C64x plus multimedia 4 GB eMMC) for graphic and video.
- **Power & Memory:** Power required is 2W. Memory on-board is 2GB (for BB-X15), plus support for microSD cards.
- **Applications:** BB applications are for media, 2D and 3D graphics, and video servers.
- **Features of BeagleBone (from Que 2.23, Page 15, Doc: Unit 2):**
  - Single-board computer and communication board.
  - Prototyping ease for media, graphics, and video servers needing IoT applications.
  - IDEs for BB include environments for code development in Python, Scratch, Squeek, Cloud9/Linux, C.
  - Programmability for number of times downloading of codes takes place through USB port during edit-test-debug cycles of development.
  - Flexibility and ease of connecting the extended memory and hardware connectivity board to external sockets for the 2-channel PCI peripheral connect interconnect express (PCIe) slot (which also functions and WiFi adapted), stereo audio, Ethernet x2, and micro SD slot.
  - Extended interfacing capabilities using 157 GPIO pins.

- **E. Intel Galileo [PYQ Q4b (June 2024)]**
  - **Definition (from Que 2.24, Page 16, Doc: Unit 2):** Intel Galileo boards are Arduino certified boards for development and prototyping.



- **Architecture:** A Galileo is based on the Intel Pentium architecture (Intel Quark SoC X1000) which includes features of a single-threaded single core and 400 MHz constant speed processor.
- **Graphics:** No separate graphic and video processors support is included in the board.
- **Compatibility:** The Galileo is hardware and software pin-to-pin compatible with shields designed for Arduino Uno R3 and Arduino IDEs.
- **Memory & OS Features:** Galileo additionally provides large 8 MB SPI flash to store firmware (Bootloader) and enables users to incorporate Linux firmware calls in Arduino sketch programming (can run a full Linux OS).
- **Sensor Support:** Galileo supports a set of 30 sensors and accessories for Arduino.
- **Features making Intel Galileo boards widely used (from Que 2.25, Page 17, Doc: Unit 2):**
  - It has single board computations and networking support.
  - IDE latest version and appropriate OS from open source.
  - IDE and software runs on multiple environments: Linux, Windows, and Mac OS X.
  - Programmability for number of times on downloading of codes through USB port, which enables multiple download occurrences during edit-test and debug cycles.
  - Flexibility and ease of connecting the extended memory and hardware connectivity board to an external full-sized mini-PCI Peripheral Connect Interconnect Express (PCIe) slot, Ethernet port socket, Micro-SD slot.
  - Extended interfacing capabilities using SPI.

- **F. ARM Cortex (Processor Architecture Context) [PYQ usage in boards]**
  - *(Referenced in Netduino section, but good to have a general note as many IoT MCUs are ARM-based)*
  - **Definition (from Que 2.26, Page 18, Doc: Unit 2):** The ARM Cortex-M is a group of 32-bit RISC (Reduced Instruction Set Computer) ARM processor cores.
  - **Intended Use:** They are intended for microcontroller use and have been shipped in tens of billions of devices.
  - **Core Family Examples:** The cores consist of the Cortex-M0, Cortex-M0+, Cortex-M1, Cortex-M3, Cortex-M4, Cortex-M7, Cortex-M23, Cortex-M33, Cortex-M35P.
  - **FPU Option:** The Cortex-M4/M7/M33/M35P cores have an FPU (Floating Point Unit) silicon option; when included, these cores are known as "Cortex-Mx with FPU" or "Cortex-MxF", where 'x' is the core number.
  - **Target Devices:** The ARM Cortex-M family is ARM microprocessor cores which are designed for use in microcontrollers, ASICs (Application-Specific Integrated Circuits), ASSPs (Application-Specific Standard Products), FPGAs (Field-Programmable Gate Arrays), and SoCs (Systems on Chip).
  - **Common Usage:** Cortex-M cores are commonly used as dedicated microcontroller chips but are also "hidden" inside SoC chips as power management controllers, I/O controllers, system

controllers, touch screen controllers, smart battery controllers, and sensors controllers.

- **Which one to choose? (Arduino vs. Raspberry Pi - from Page 11, Doc: Unit 2):**
  - The choice depends on the specific requirements of your IoT project.
  - **Arduino:** If building a simple project that only requires basic sensing and actuation, Arduino is a good choice. It is easy to use and designed for beginners.
  - **Raspberry Pi:** If building a more complex project that requires more processing power and connectivity options (like running a web server, complex analytics, or needing full OS capabilities), then Raspberry Pi is a better choice.