Internet Of Things (IOT) BTDSE613N

Supervised by

Dr. Shyam Gehlot

Associate Professor, CSE Dept. SVIIT

SVVV, Indore

COURSE OBJECTIVES

The student will have ability to:

1. Impart necessary and practical knowledge of components of Internet of Things.
2. Develop skills required to build real-life IoT based projects. .
3. To provide the information for hardware utilization methodology.
4. To impart knowledge of IOT System and inter-process communication.

COURSE OUTCOMES

Upon completion of the subject, Students will be able:

1. Understand internet of Things and its hardware and software components
2. Interface I/O devices, sensors & communication modules
3. Remotely monitor data and control devices
4. Develop real life IoT based projects.

Syllabus

Unit-III IoT Application Development

Solution framework for IoT applications- Implementation of Device integration

Data acquisition and integration.

INTRODUCTION

- **IoT Application Development** refers to the process of designing, developing, and deploying applications that connect **physical devices, sensors, and software systems through the Internet** to collect, process, and analyze data.
- IoT applications are used in many domains such as:
- Smart homes
- Healthcare monitoring
- Industrial automation
- Agriculture
- Smart cities
- Transportation
- The development of an IoT application requires integration of **hardware devices, communication networks, cloud platforms, and user applications**.

Solution Framework for IoT Applications

- A **solution framework** provides a structured architecture for developing IoT applications. It defines how different components such as devices, networks, cloud services, and applications interact with each other.
- **Main Layers of IoT Solution Framework**

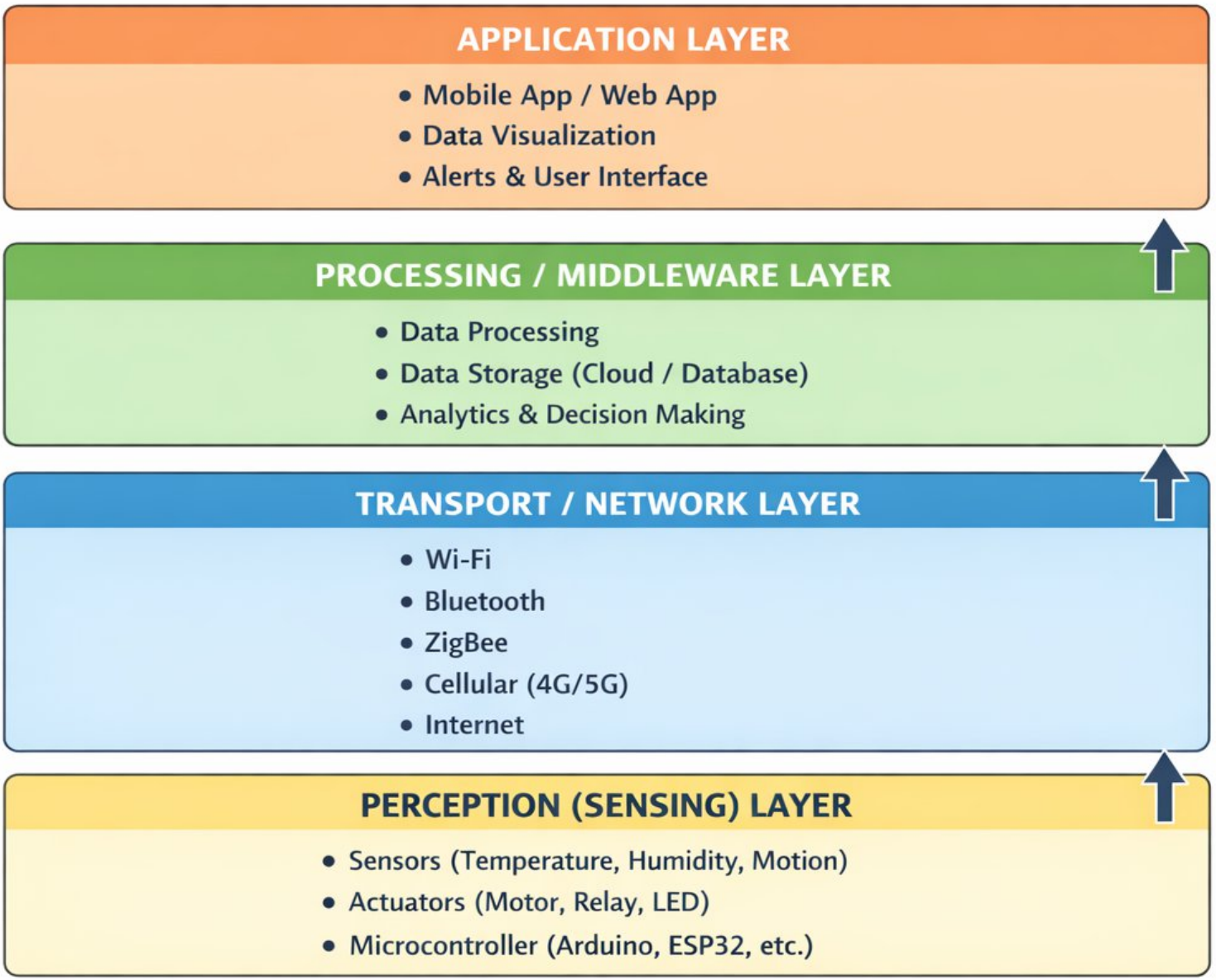
Device Layer

Communication Layer

Cloud / Data Processing Layer

Application Layer

IOT SYSTEM (LAYER ARCHITECTURE)



IoT architecture is a layered structure that divides an IoT system into perception, network, processing, and application layers, enabling efficient data sensing, communication, analysis, and user interaction.

1. Perception Layer (Sensing Layer)

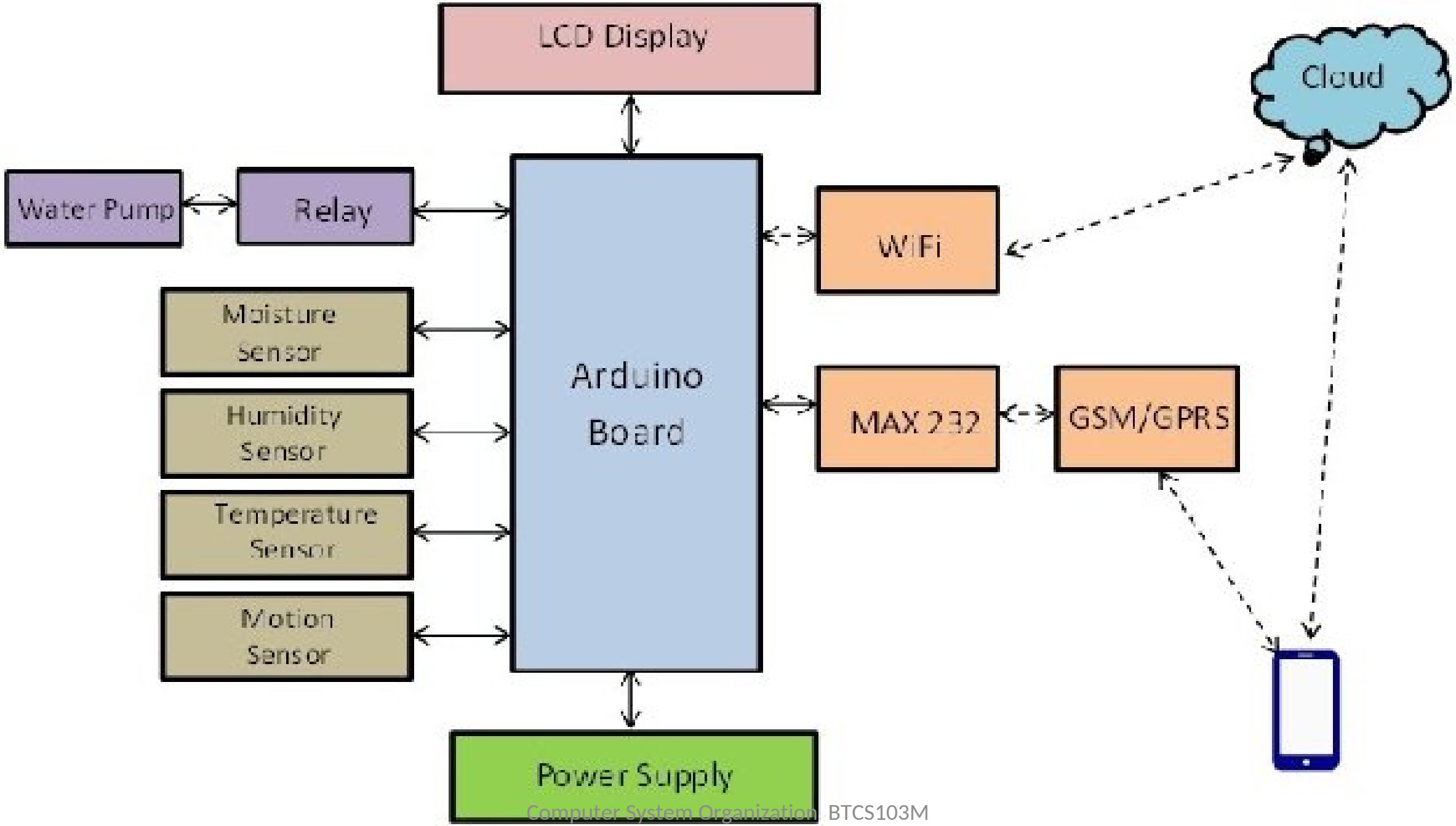
Its is lowest or bottom layer of IOT system .It is also know as physical or sensing layer.

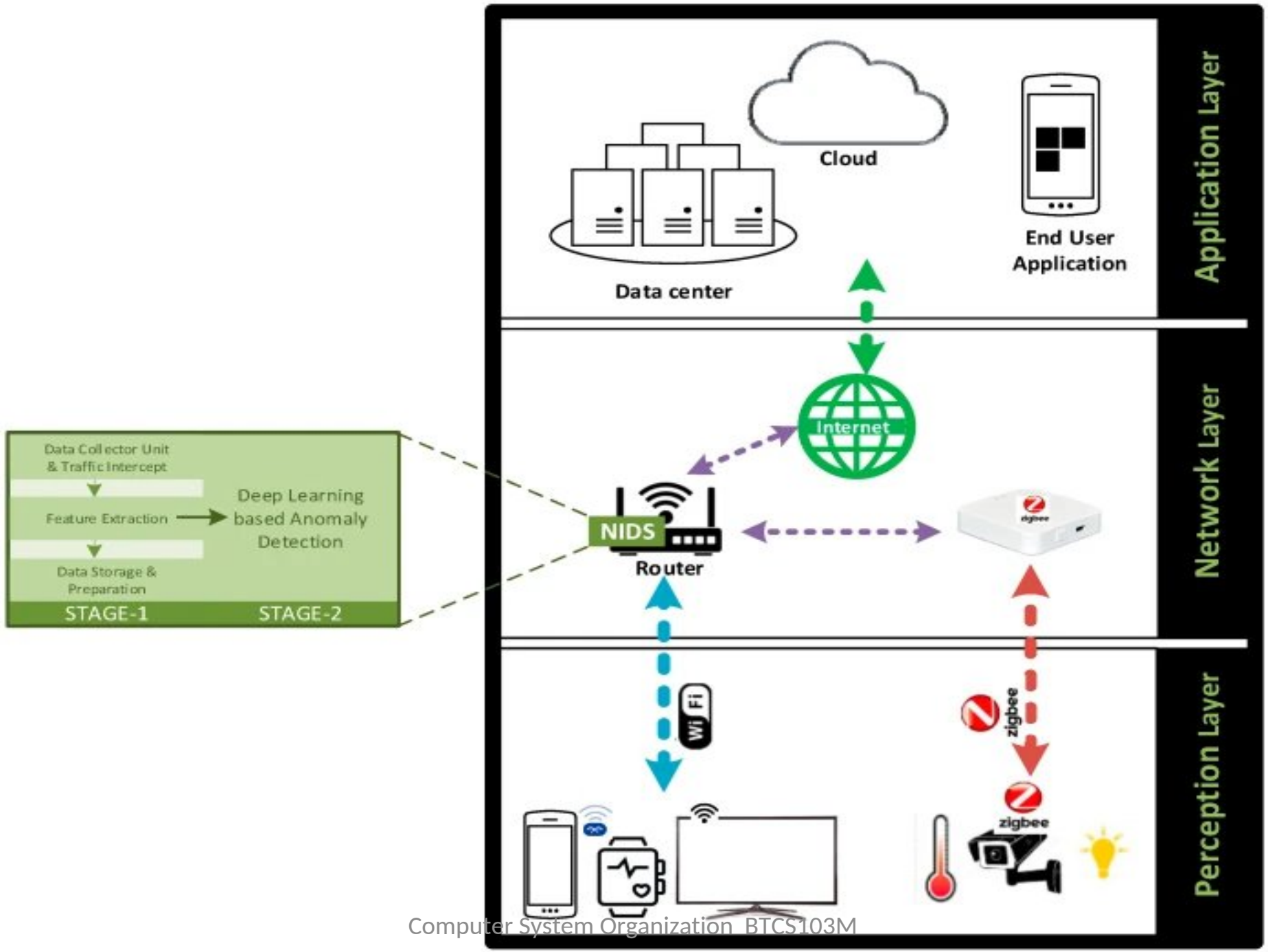
Function : 1) Collects data from the physical environment 2) Converts physical parameters into digital data 3) Performs actuation.

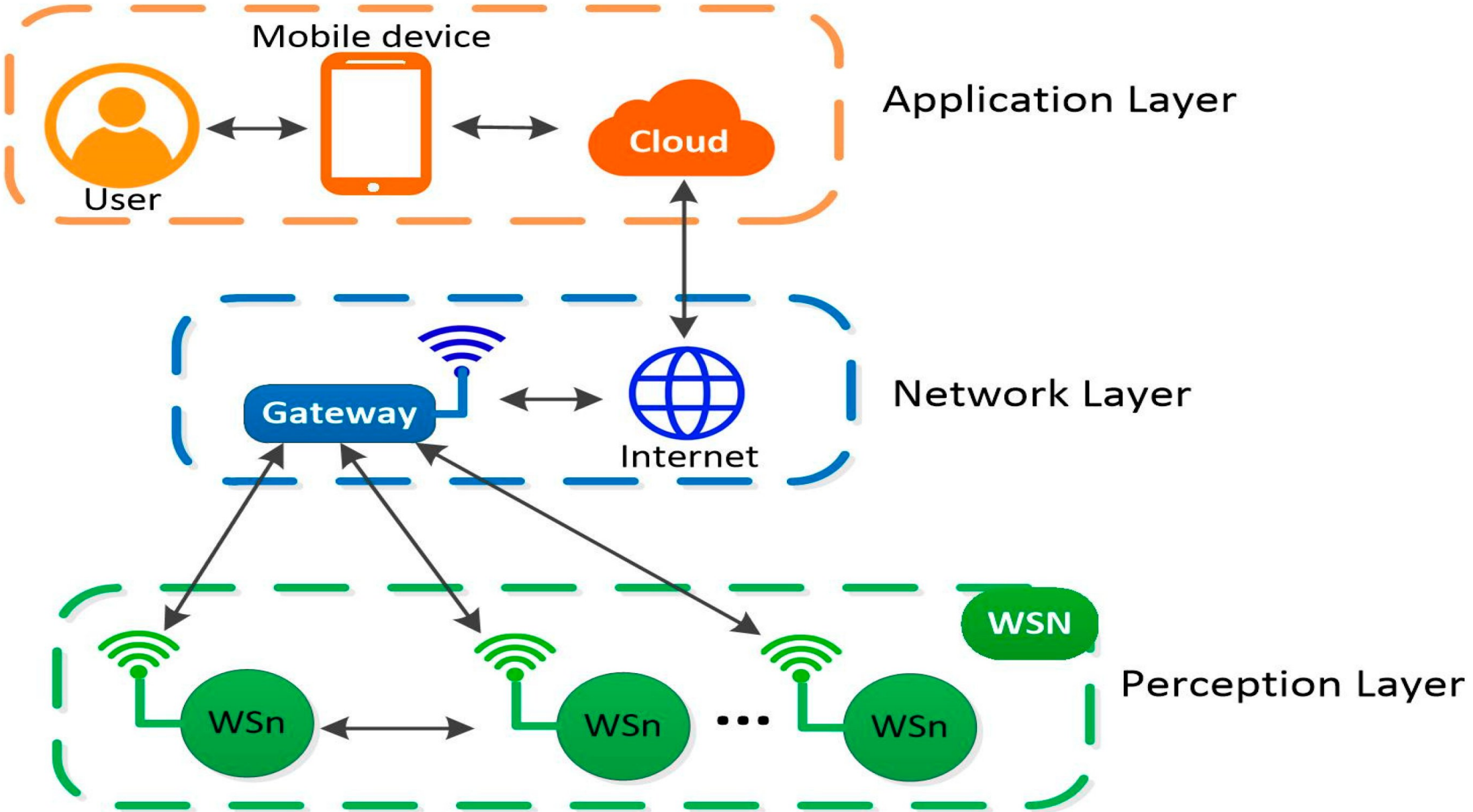
Devices Sensors: Temperature sensor (DHT11), Humidity sensor, Motion sensor (PIR), Gas sensor **Actuators:** Relay, Motor, LED **, Controller:** Arduino, ESP32.

Protocols / Interfaces GPIO 2) I2C 3) SPI 4) ADC

Example: A temperature sensor measures room temperature and sends the value to Arduino.


LCD Display
Cloud
Water Pump
Relay
WiFi
Moisture Sensor
Arduino Board
Humidity Sensor
MAX 232
GSM/GPRS
Temperature Sensor
Motion Sensor
Power Supply


Application Layer
Network Layer
Perception Layer
Cloud
Data center
End User Application
Internet
NIDS
Router
zigbee
zigbee
zigbee
Wi Fi
Data Collector Unit & Traffic Intercept
Feature Extraction
Data Storage & Preparation
Deep Learning based Anomaly Detection
STAGE-1
STAGE-2


Mobile device
User
Cloud
Application Layer
Gateway
Internet
Network Layer
WSN
WSn
WSn
WSn
Perception Layer

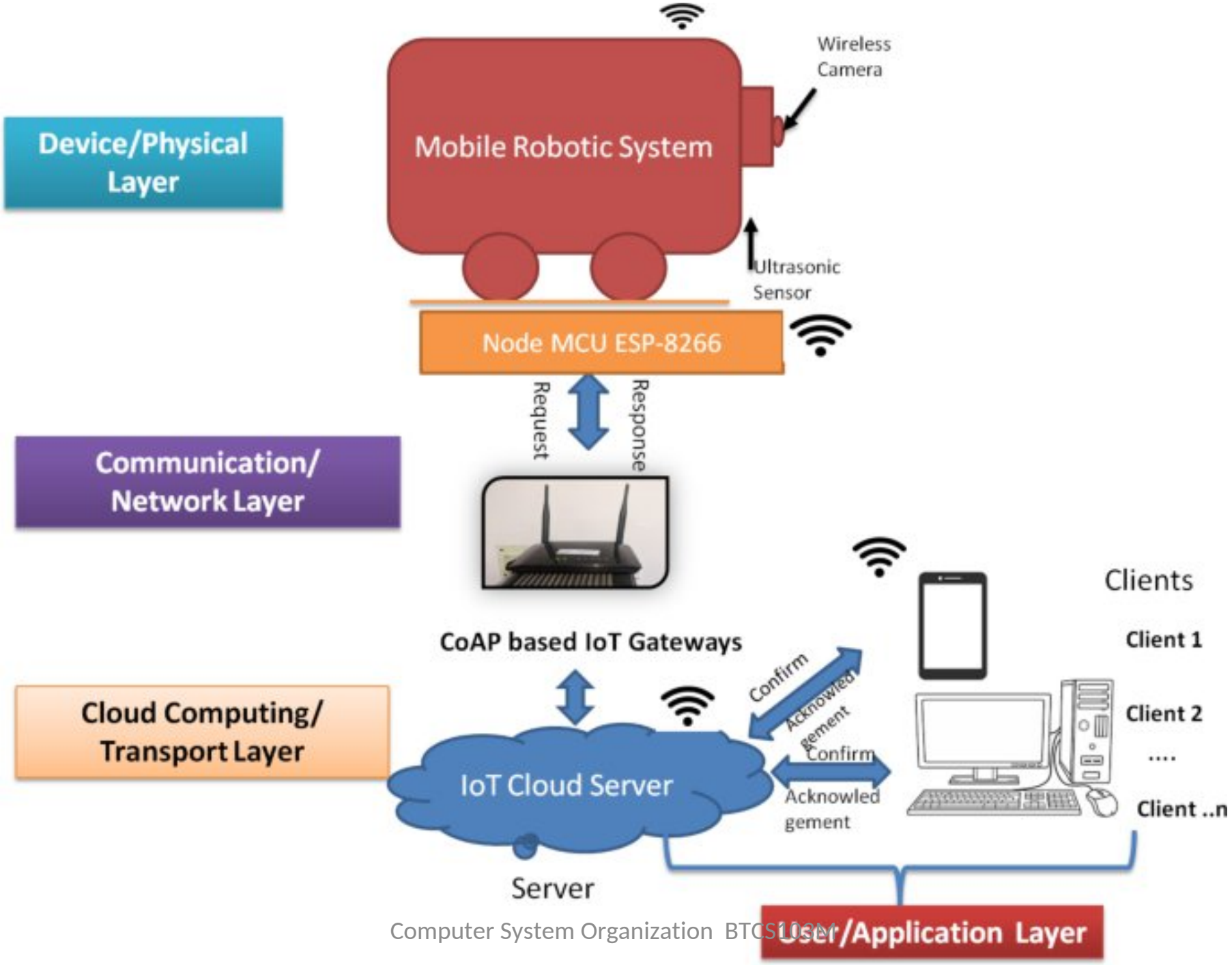
1. Network Layer / (Transport Layer)

This layer moves the data from the Perception layer to the processing system. It's the bridge between the edge and the cloud.

Function : 1) Transfers sensor data to cloud/server 2) Enables communication between devices and processing layer **Devices** Wi-Fi module (ESP8266 / ESP32) 2) Bluetooth module 3) GSM module 4) LoRa module

Protocols / Interfaces Wi-Fi , Bluetooth ,ZigBee ,LoRa WAN, TCP/IP

Example:ESP32 sends temperature data to cloud using Wi-Fi


Device/Physical Layer
Mobile Robotic System
Wireless Camera
Ultrasonic Sensor
Node MCU ESP-8266
Communication/ Network Layer
Request
Response
CoAP based IoT Gateways
Cloud Computing/ Transport Layer
IoT Cloud Server
Confirm
Acknowledgement
Confirm
Acknowledgement
Server
Clients
Client 1
Client 2
....
Client ..n
User/Application Layer

3. Processing / Middleware Layer

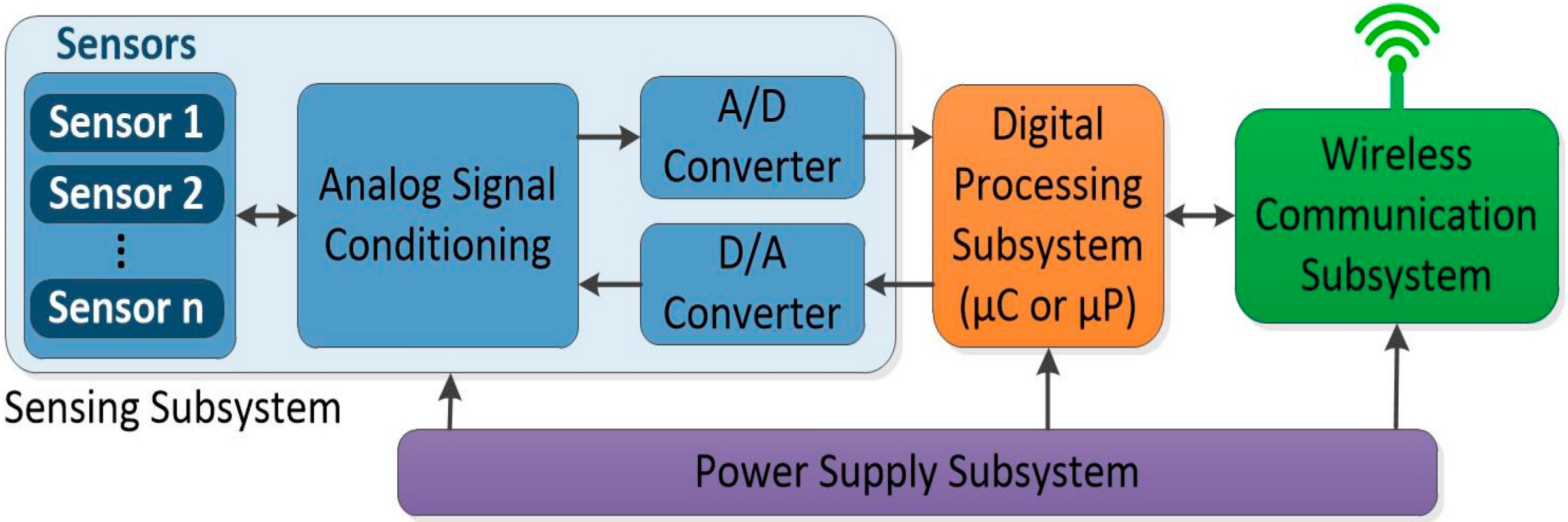
Think of this as the brain's sorting office. Before data reaches the user, it needs to be stored, filtered, and analyzed.

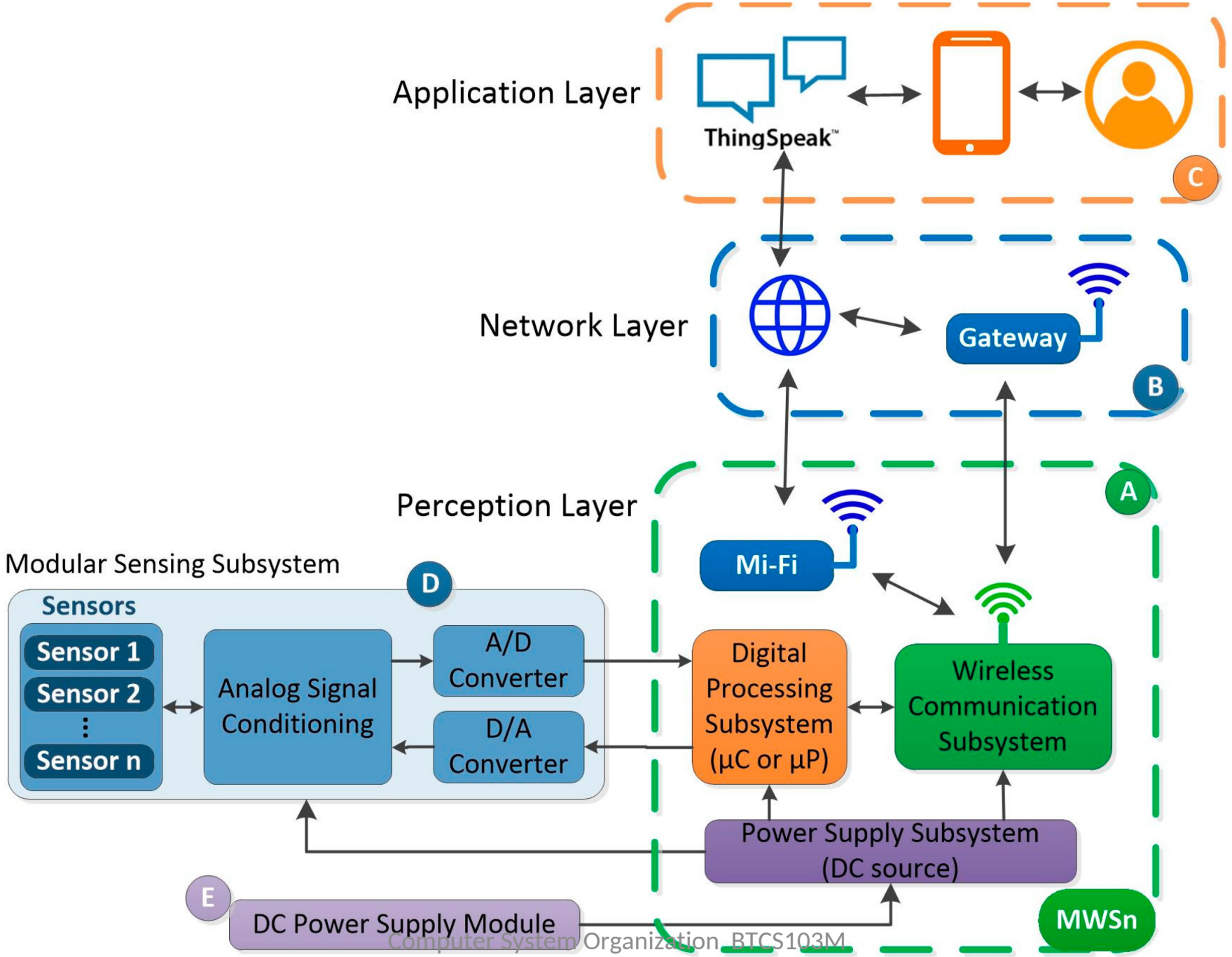
Function : 1) Stores and processes IoT data 2) Performs data analysis and decision making 3) Manages devices and security

Devices / Platforms: 1) Cloud servers 2) Databases 3) AWS IoT, Azure IoT, Google Cloud IoT

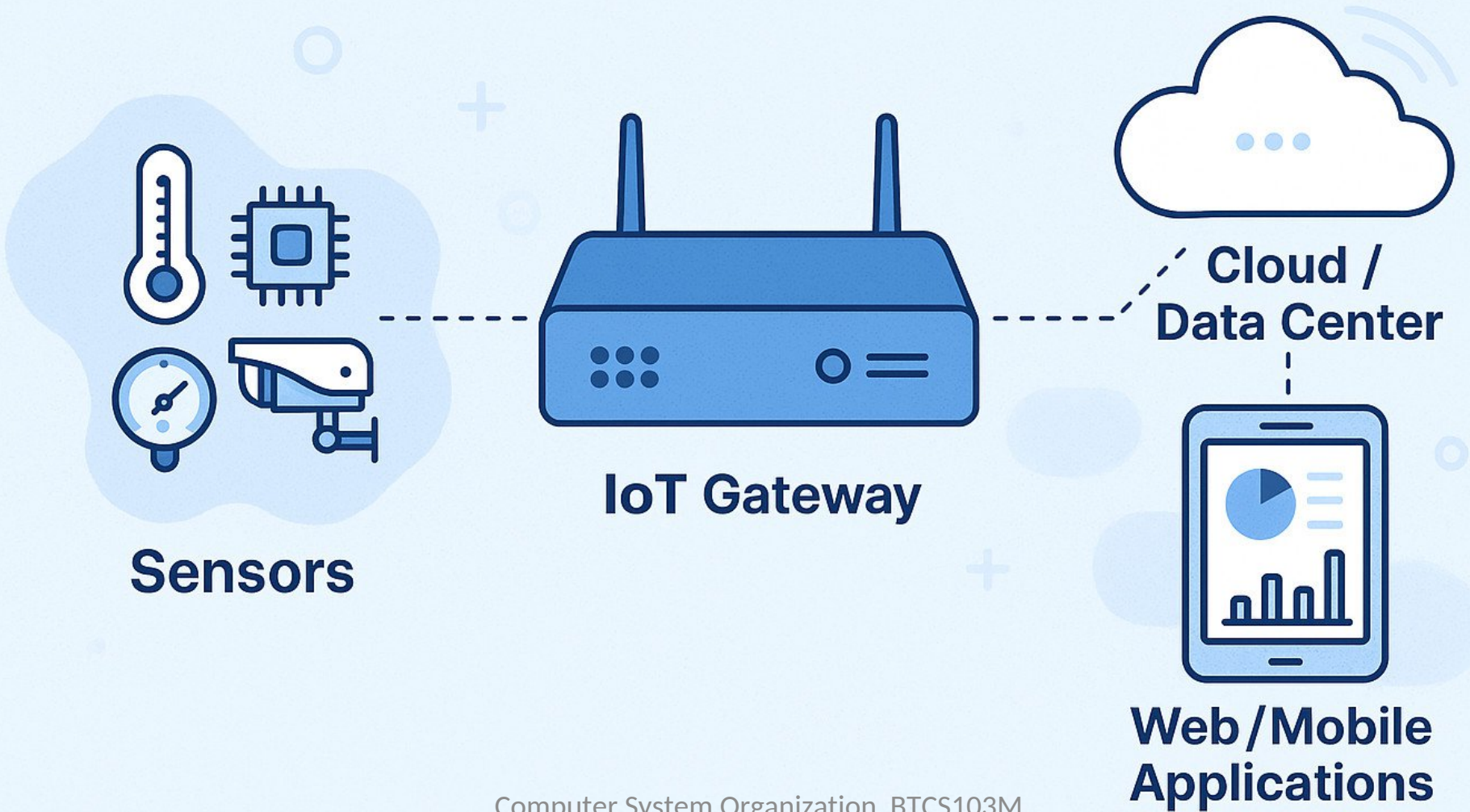
Protocols / Interfaces: MQT ,THTTP / HTTPS, CoAP

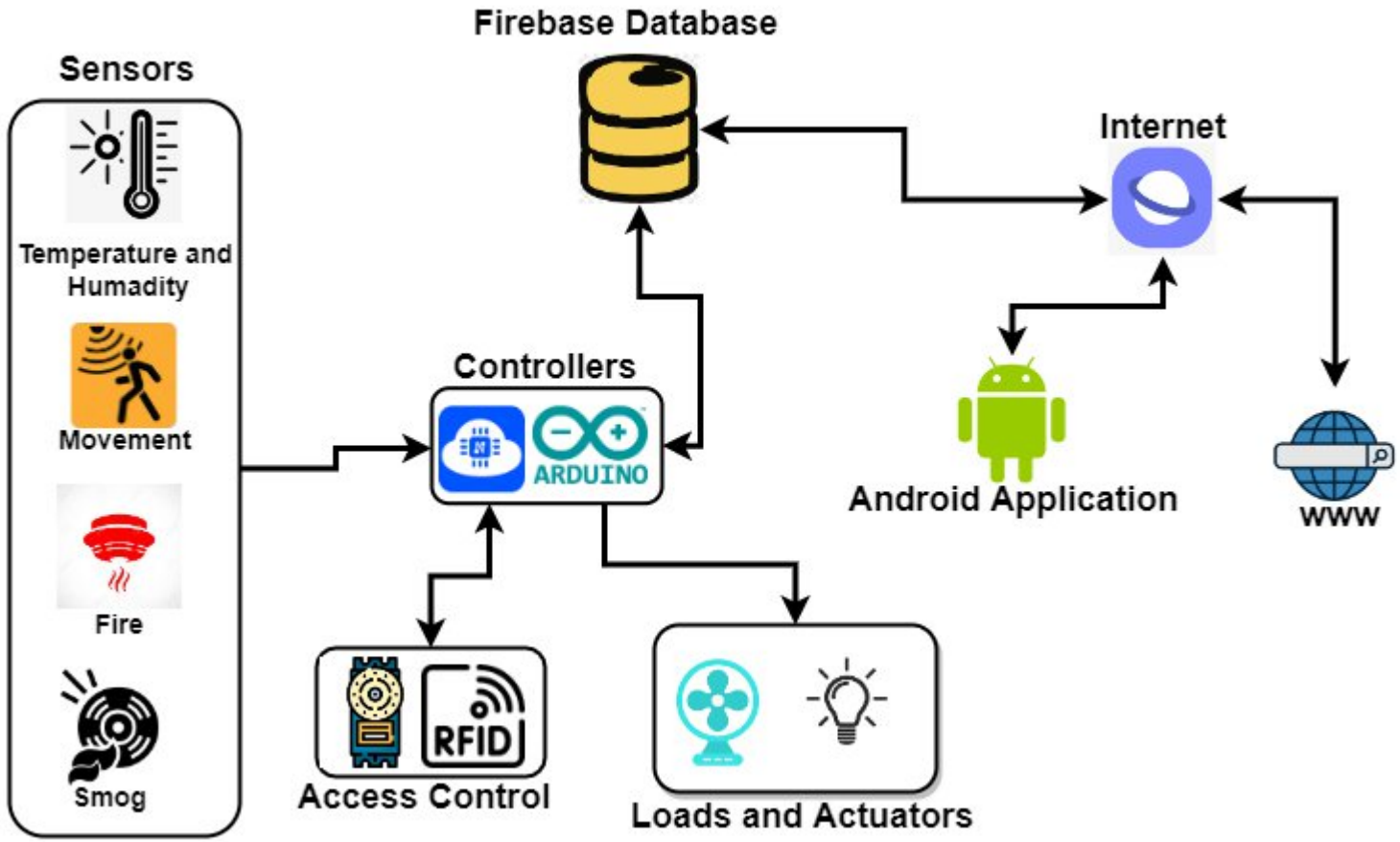
Example: Cloud checks temperature data and decides whether to turn ON a fan.


Sensors
Sensor 1
Sensor 2
Sensor n
Analog Signal Conditioning
A/D Converter
D/A Converter
Digital Processing Subsystem (µC or µP)
Wireless Communication Subsystem
Sensing Subsystem
Power Supply Subsystem


Application Layer
ThingSpeak™
C
Network Layer
Gateway
B
Perception Layer
A
Mi-Fi
Modular Sensing Subsystem
D
Sensors
Sensor 1
Sensor 2
Sensor n
Analog Signal Conditioning
A/D Converter
D/A Converter
Digital Processing Subsystem (µC or µP)
Wireless Communication Subsystem
Power Supply Subsystem (DC source)
E
DC Power Supply Module
MWSn

What is an IoT Gateway?




Firebase Database
Sensors
Temperature and Humadity
Movement
Fire
Smog
Internet
Controllers
ARDUINO
Android Application
www
RFID
Access Control
Loads and Actuators

4. Application Layer

It is topmost layer of IOT architecture. It act as interface. This is what the user interacts with it . This is the part users see and interact with. It shows useful data and allows users to control devices.

Function : 1) Provides user interface 2) Displays data and reports 3) Allows remote monitoring and control

Devices / Platforms: 1) Smartphone 2) Laptop / PC Tablet

Protocols / Interfaces: THTTP / HTTPS

Example: Mobile app displays live temperature and allows user to control devices.

2.1 Device Layer (Perception Layer)

This layer contains **physical IoT devices** that interact with the environment.

Components

- Sensors
- Actuators
- Embedded systems
- Microcontrollers

Examples

- Temperature sensor
- Soil moisture sensor
- Motion sensor
- Arduino / Raspberry Pi

Functions

- Collect real-time data from environment
- Send data to the processing system
- Control physical devices

Example:

A **temperature sensor** collects temperature data and sends it to the controller.

2.2 Communication Layer (Network Layer)

This layer transfers data between **devices and cloud platforms**.

Communication Technologies

- Wi-Fi
- Bluetooth
- ZigBee
- LoRaWAN
- Cellular networks (4G/5G)

Protocols Used

- HTTP
- MQTT
- CoAP
- TCP/IP

Functions

- Data transmission
- Device connectivity
- Secure communication

Example:

A **NodeMCU module sends sensor data to cloud through Wi-Fi.**

2.3 Cloud / Data Processing Layer

The cloud layer stores, processes, and analyzes IoT data.

Functions

- Data storage
- Data processing
- Analytics
- Device management

Common IoT Cloud Platforms

- AWS IoT
- Microsoft Azure IoT
- Google Cloud IoT

Example

Sensor data is uploaded to **ThingSpeak cloud**, where graphs and analytics are generated.

2.4 Application Layer

This layer provides **user interfaces and services.**

Components

- Mobile applications
- Web dashboards
- Data visualization tools

Functions

- Display sensor data
- Send control commands to devices
- Monitor system performance

Example:

A farmer can monitor **soil moisture using a ↓ artphone app** and control irrigation remotely.

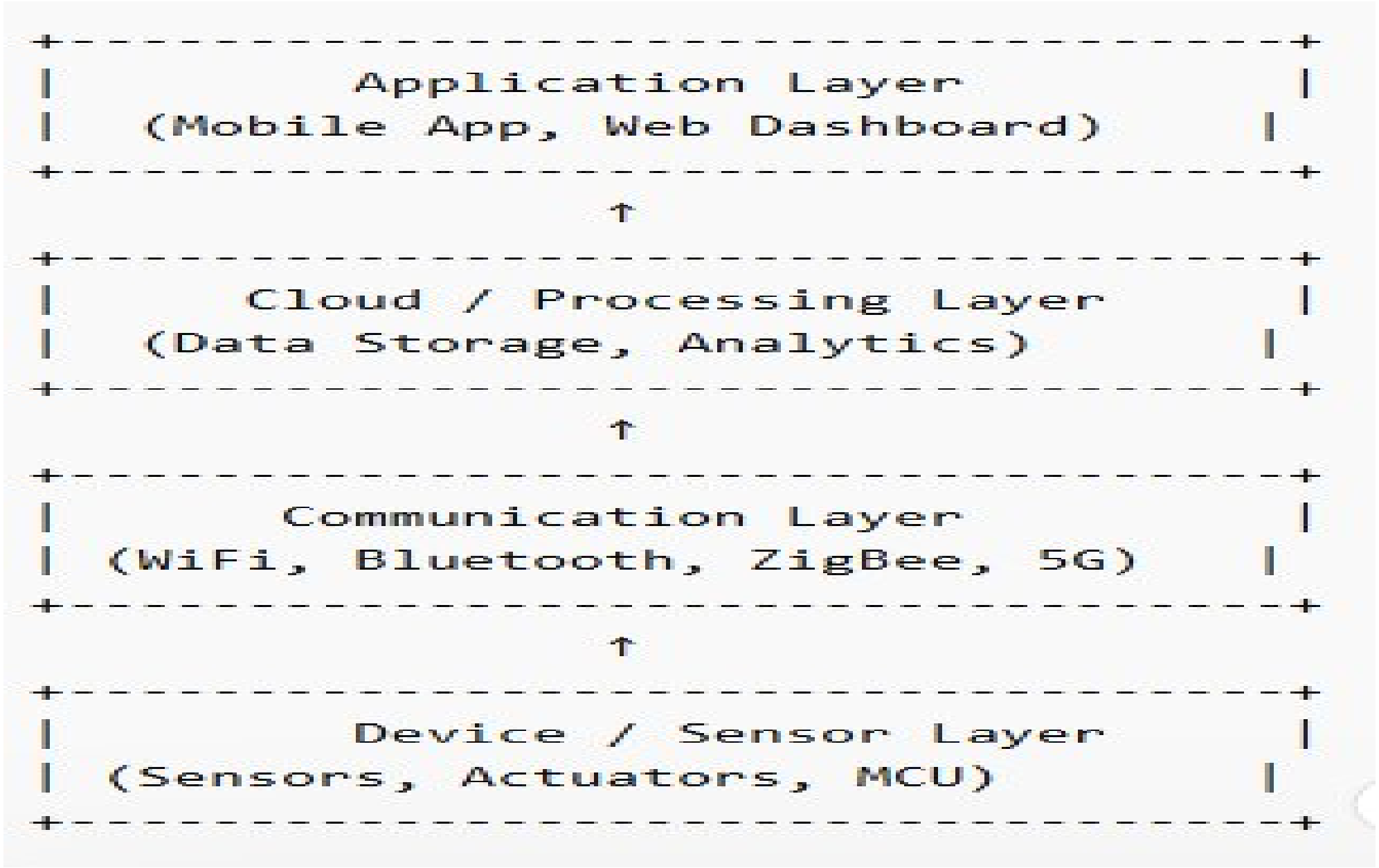
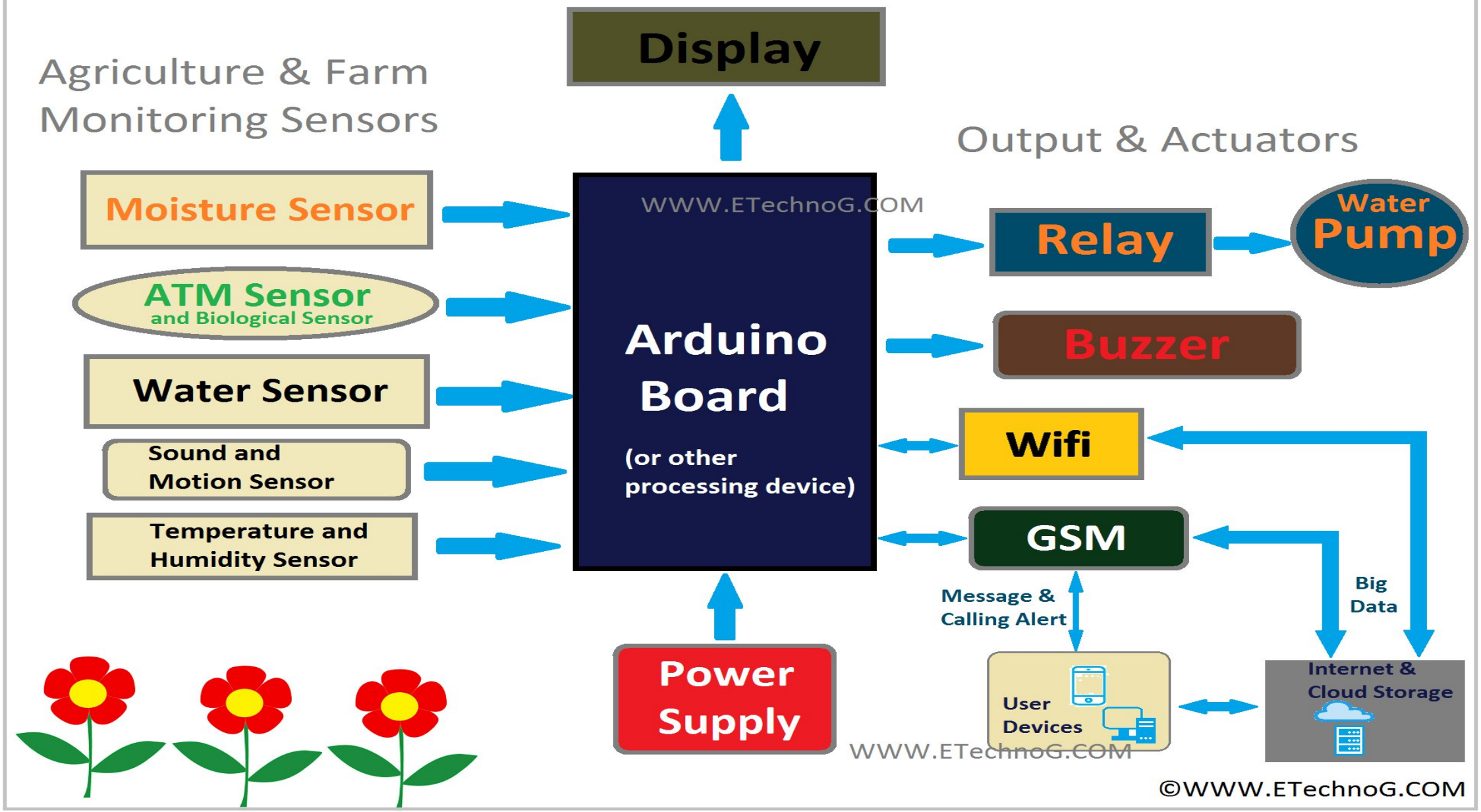

Application Layer
(Mobile App, Web Dashboard)
Cloud / Processing Layer
(Data Storage, Analytics)
Communication Layer
(WiFi, Bluetooth, ZigBee, 5G)
Device / Sensor Layer
(Sensors, Actuators, MCU)

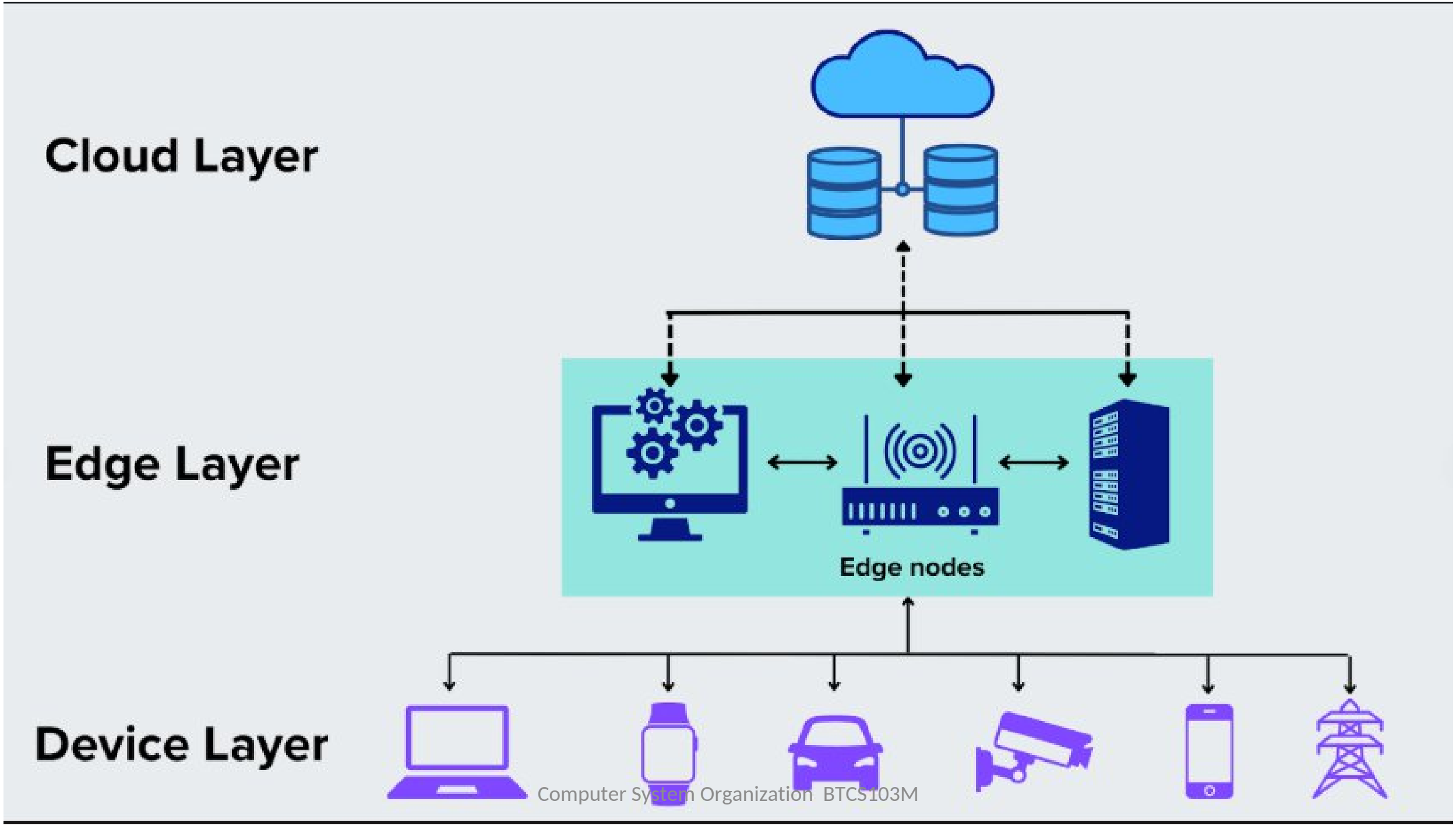
Figure. 01 IoT Based Smart Agriculture Block Diagram

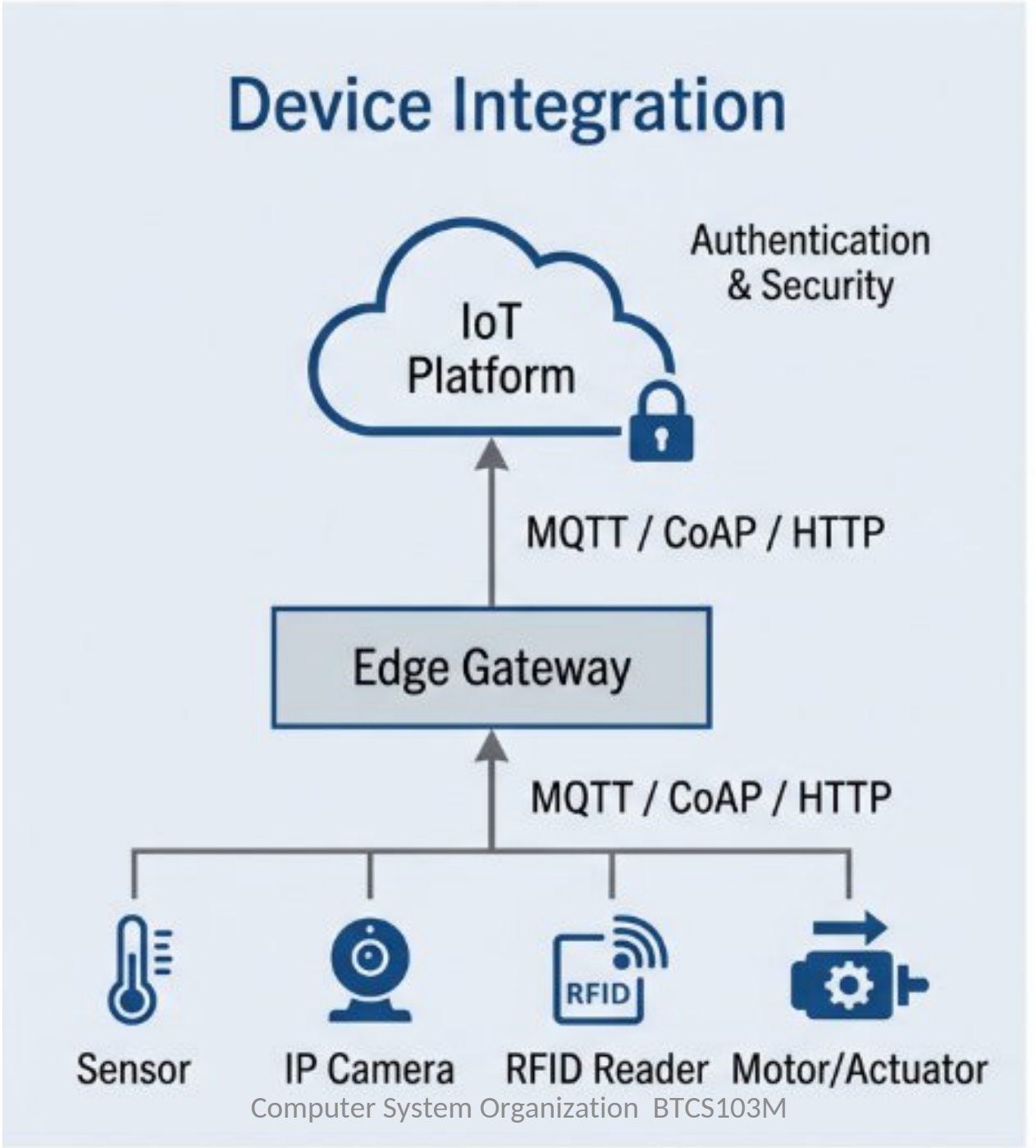


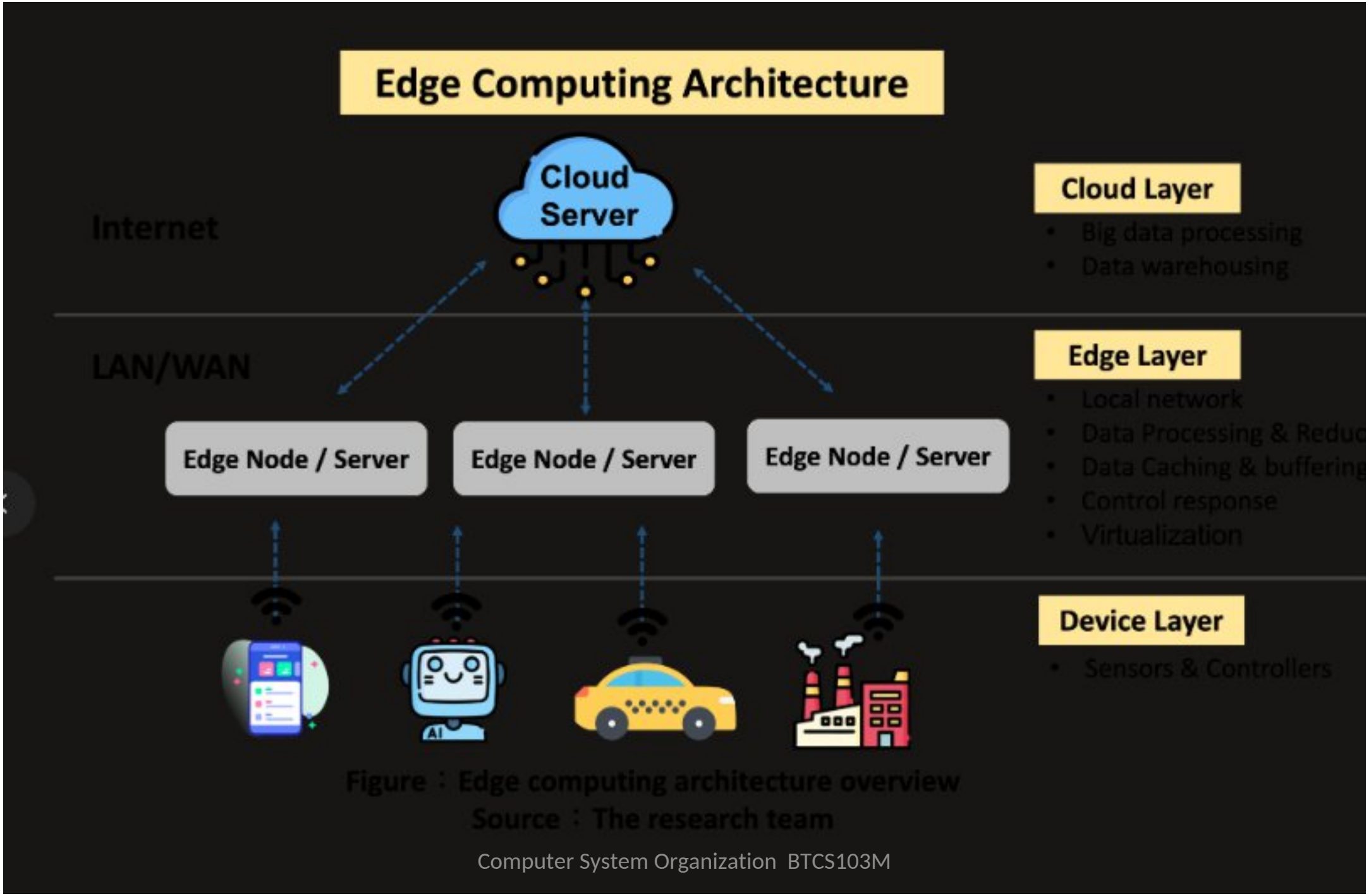
Device Integration

Device integration refers to the process of connecting diverse hardware devices—like sensors, cameras, and actuators—to a central system, enabling them to communicate, share data, and operate seamlessly. In IoT (Internet of Things) setups, this creates smart ecosystems, such as a factory monitoring temperatures or a smart home controlling lights.

A typical IoT device integration flow: devices connect to an Edge Gateway, which processes and forwards data via protocols like MQTT/CoAP/HTTP to an IoT Platform for authentication, security, and analysis. Arrows show the data flow upward, with secure protocols ensuring reliable transmission.


Cloud Layer
Edge Layer
Edge nodes
Device Layer


Device Integration
Authentication
& Security
IoT
Platform
MQTT / CoAP / HTTP
Edge Gateway
MQTT / CoAP / HTTP
RFID
Sensor
IP Camera
RFID Reader
Motor/Actuator



Figure：Edge computing architecture overview
Source：The research team

1. Devices (Sensors, IP Camera, RFID Reader, Motor/Actuator)

- These are the "endpoints" that collect data or perform actions. They generate raw inputs (e.g., temperature readings) or respond to commands (e.g., turning a motor).
- **How it works**: Devices use built-in interfaces like GPIO pins, Wi-Fi, Bluetooth, or wired connections to send/receive data. They often run lightweight firmware to package data into simple formats (e.g., JSON packets).

- **Example**:

Device	Function	Example in Action
Sensor (temp icon)	Measures environmental data	A thermostat in a warehouse sends $25^{\circ}C$every 10 seconds via Wi-Fi.
IP Camera	Captures video/images	Streams live feed from a security camera, detecting motion and alerting via HTTP.
RFID Reader	Scans tags for identification	At a factory gate, it reads employee badges to log entry times.
Motor/Actuator	Executes physical actions	A valve actuator opens based on a command to release water if temperature exceeds $30^{\circ}C$.

2. Edge Gateway

- This acts as a local "traffic cop," aggregating data from multiple devices, doing preliminary processing (filtering, compression), and routing it to the cloud. It reduces bandwidth by handling edge computing.
- **How it works**: Receives data via protocols like Modbus or Zigbee from devices, applies rules (e.g., "only send alerts if temp > 30°C"), then translates to cloud-friendly protocols (MQTT/CoAP/HTTP). It also buffers data during outages.

2. Edge Gateway

- **Example**: In a smart farm (matching the diagram), the gateway pulls temp from the sensor, video from the IP camera, and RFID scans. It fuses data—if a worker (RFID) is near overheating equipment (sensor), it commands the motor to cool it locally, then sends a summary packet via MQTT: {"device": "sensor", "temp": 32, "alert": true}.

3. Protocols (MQTT/CoAP/HTTP)

These are the "languages" for data transport—lightweight and secure for IoT's constrained networks.

How it works:

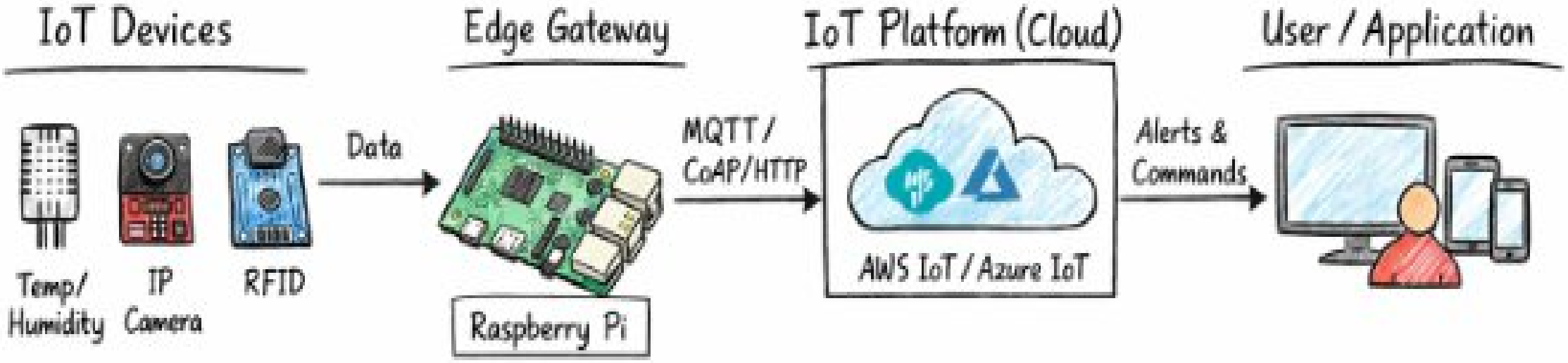
Protocol	Key Features	Use Case from Diagram
MQTT	Publish/subscribe, low-bandwidth	Edge Gateway publishes sensor data; platform subscribes for real-time updates.
CoAP	UDP-based, for tiny devices	IP Camera sends lightweight images over unreliable networks.
HTTP	Web-standard, request/response	RFID logs sent as REST API calls for easy integration.

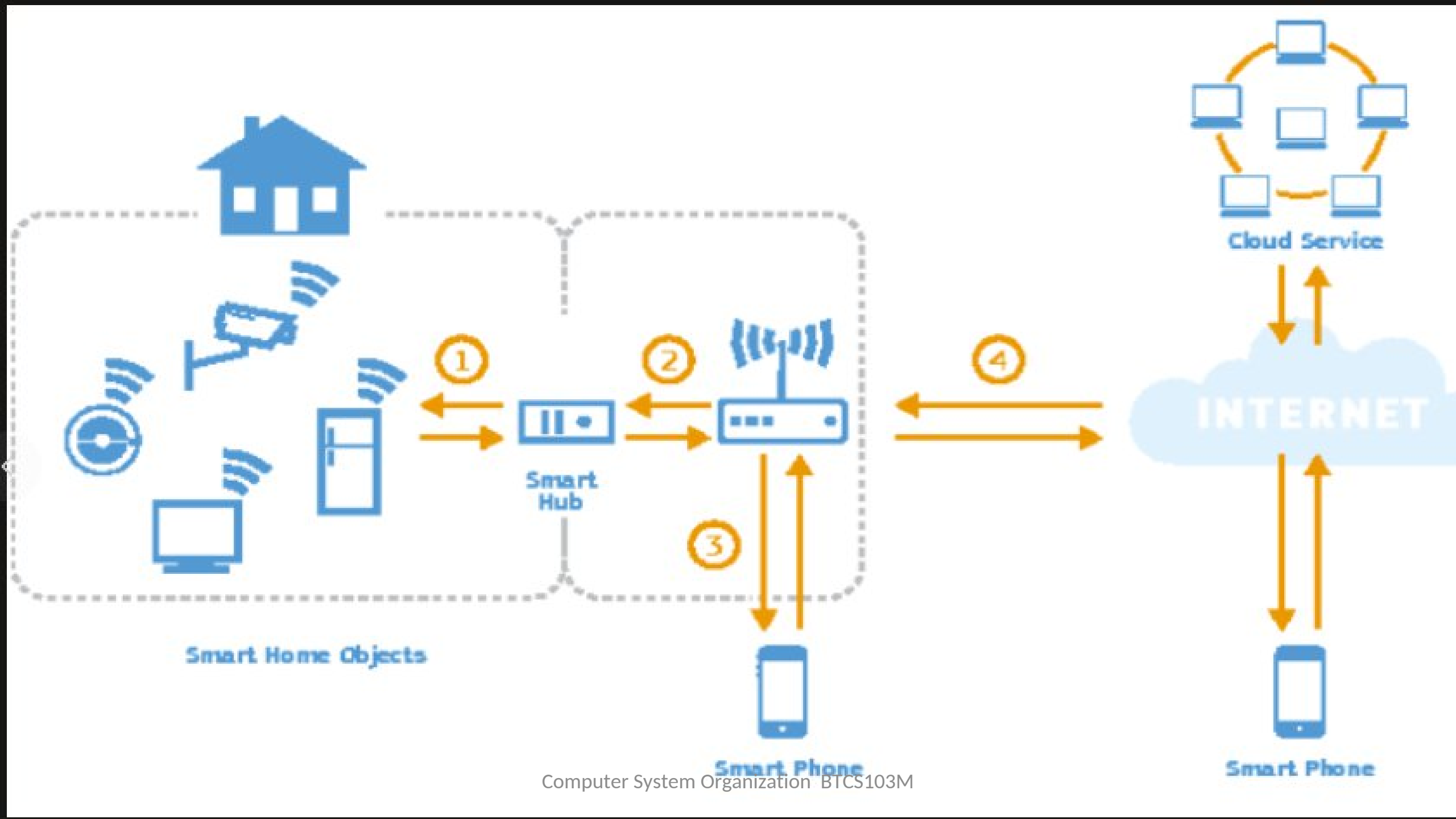
4. IoT Platform (with Authentication & Security)

The cloud brain: stores data, runs analytics, dashboards, and apps. The lock icon highlights security layers.

How it works: Ingests data via APIs, authenticates devices (e.g., API keys, TLS certificates), encrypts traffic, and applies rules (e.g., role-based access). It processes for insights, like ML predictions.

Example: Platforms like AWS IoT or Azure IoT Hub receive the farm's data. Authentication verifies the Edge Gateway's certificate. Security blocks unauthorized access—if the RFID detects an intruder, it triggers camera footage analysis and locks the motor. Dashboard shows: "Temp alert: Actuator activated at 7:56 AM."


IoT Devices
Edge Gateway
IoT Platform (Cloud)
User / Application
Temp/ Humidity
IP Camera
RFID
Data
Raspberry Pi
MQTT / CoAP/HTTP
AWS IoT / Azure IoT
Alerts & Commands


Cloud Service
1
2
4
Smart Hub
INTERNET
3
Smart Home Objects
Smart Phone
Smart Phone

Implementation of Device Integration in IoT (Step-by-Step)

Step 1: Plan and Gather Components

Identify requirements: List devices, data needs (e.g., temp readings), and goals (e.g., auto-cooling at 32°C).

Select hardware: Devices per diagram, Raspberry Pi 4 as Edge Gateway, Ethernet/Wi-Fi modules.

Choose software: Mosquito for MQTT broker, Node-RED for gateway logic, AWS IoT/Azure IoT Hub as platform.

Example: For smart farm—DHT22 sensor, ESP32 camera module, RFID-RC522, servo motor.

Step 1: Planning and Component Selection

Identify Requirements

- What data is needed? (Temperature, humidity, etc.)
- What action is required? (e.g., Motor ON at 32°C)

Select Hardware

- Sensor: DHT22
- Camera: ESP32 Camera
- RFID: RC522
- Actuator: Motor/Relay
- Gateway: Raspberry Pi

👉 **Example:**

Smart Farm → If temperature > 32°C → Turn ON cooling system

Select Software

- MQTT Broker: Mosquitto
- Flow Tool: Node-RED
- Cloud Platform: AWS IoT

Step 2: Set Up Individual Devices

Program each device to send/receive data locally.

Sensor: Connect to microcontroller (e.g., Arduino/ESP32), code to read data every 10s: temp = dht.readTemperature();.

IP Camera: Configure RTSP stream, enable HTTP endpoint for snapshots.

RFID Reader: Wire to pins, script scans tags: if (tag == "workerID") { sendStatus(); }.

Motor/Actuator: Use relay/GPIO: digitalWrite(MOTOR_PIN, HIGH); on command.

Test: Use Serial Monitor; ensure Wi-Fi/Bluetooth links to gateway IP.

Step 2: Setup Individual Devices

Each device is programmed separately.

Sensor (Temperature)

```cpp
temp = dht.readTemperature();
```

RFID Reader

```cpp
if(tag == "workerID") {
   sendStatus();
}
```

Motor / Actuator

```cpp
digitalWrite(MOTOR_PIN, HIGH);
```

IP Camera

- Configure live streaming (RTSP)
- Enable HTTP snapshot

Testing

- Use Serial Monitor
- Check Wi-Fi connection

Step 3: Configure Edge Gateway

Install OS (Raspberry Pi OS) on gateway hardware.

- **Install brokers:** sudo apt install mosquitto eclipse-mosquitto.
- **Add Node-RED:** bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered).
- **Connect devices:** Map GPIO ports or Wi-Fi (e.g., sensor to port 1883 MQTT).
- **Filter logic:** In Node-RED, "if temp > 30°C, publish to /alerts topic; else ignore."

Example: Gateway subscribes to device topics, aggregates: {"temp":32, "rfid":"ID123", "action":"cool"}.

Step 3: Configure Edge Gateway

Gateway acts as **brain (local processing unit)**.

✔ Install OS

- Raspberry Pi OS

✔ Install MQTT Broker

```bash
sudo apt install mosquitto
```

✔ Install Node-RED

```bash
bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers
```

✔ Data Processing Logic

Example in Node-RED:

- If temp > 30°C → Send alert
- Else ignore

👉 Example Data:

```
{"temp":32, "rfid":"ID123", "action":"cool"}
```

Step 4: Implement Protocols

Enable secure communication between gateway and cloud.

MQTT: Configure broker with TLS: Edit /etc/mosquitto/mosquitto.conf for certs, publish/subscribe topics.

HTTP: Node-RED HTTP nodes for REST: POST to platform API.

Test locally: mosquitto_pub -t /farm/temp -m "32°C".

Step 4: Implement Communication Protocols

Protocols are used for communication.

Protocol	Use
MQTT	Lightweight, most common
CoAP	Low-power devices
HTTP	Web APIs

MQTT Example

Bash

```
mosquitto_pub -t /sensors/temp -m 32
```

CoAP Example

Bash

```
coap-client -m post coap://edge:5683/temp 32
```

HTTP Example

Bash

```
curl -X POST http://platform/api/data '{"temp":32}'
```

Step 5: Integrate with IoT Platform

Create account on platform (e.g., AWS IoT Core).

- Register devices: Generate certificates, policies for auth (matches diagram's lock icon).
- Create rules: "If /alerts temp>30, trigger actuator via shadow."
- Set up dashboard: Visualize data (e.g., temp graphs), send commands back.
- Security: Enforce mutual TLS, API keys; encrypt payloads.

Step 5: Integration with IoT Platform (Cloud)

Steps

- Create account (AWS IoT / Azure IoT)
- Register devices
- Generate certificates (security)

Rules Example

- If temperature > 30°C → Turn ON motor

Dashboard

- Display graphs (temperature, RFID logs)
- Send control commands

Security

- Use TLS encryption
- Use API keys

Step 6: Test End-to-End Flow

Simulate: Sensor sends 32°C → Gateway filters/publishes MQTT → Platform analyzes → Command returns to motor (turns on).

Monitor: Use Wireshark for packets, platform logs for delays.

Scale: Add more devices; handle failures (e.g., gateway buffers offline data).

Step 6: End-to-End Testing

Working Flow

```
Sensor → Gateway → Cloud → Command → Motor
```

Example:

- Sensor sends 32°C
- Gateway processes data
- Cloud sends command
- Motor turns ON

Monitoring Tools

- Wireshark (network packets)
- Cloud logs

Step 7: Deploy and Maintain

- Go live: Use Docker for portability, cloud auto-scaling.
- Monitor: Set alerts for downtime; update firmware.
- Optimize: Compress video from IP camera, batch RFID logs.

Common Pitfalls	Fixes
Connection drops	Add retries, MQTT QoS=2
Security gaps	Always use TLS 1.3
High latency	Edge-process 80% of data

Step 7: Deployment and Maintenance

Deployment

- Use Docker for scalability
- Enable cloud auto-scaling

Maintenance

- Monitor system performance
- Update firmware regularly

Optimization

- Compress camera data
- Batch RFID records

1. The Device Layer (The "Senses")

These are the individual gadgets scattered around your house.

- **Examples:** Smart light bulbs, motion sensors, door locks, thermostats, and security cameras.
- **Function:** They collect raw data. For instance, a motion sensor detects a person entering the living room. It doesn't "know" what to do; it just reports the event to the Hub via a low-power protocol like **Zigbee** or **Matter**.

2. The Edge Layer (The "Local Brain")

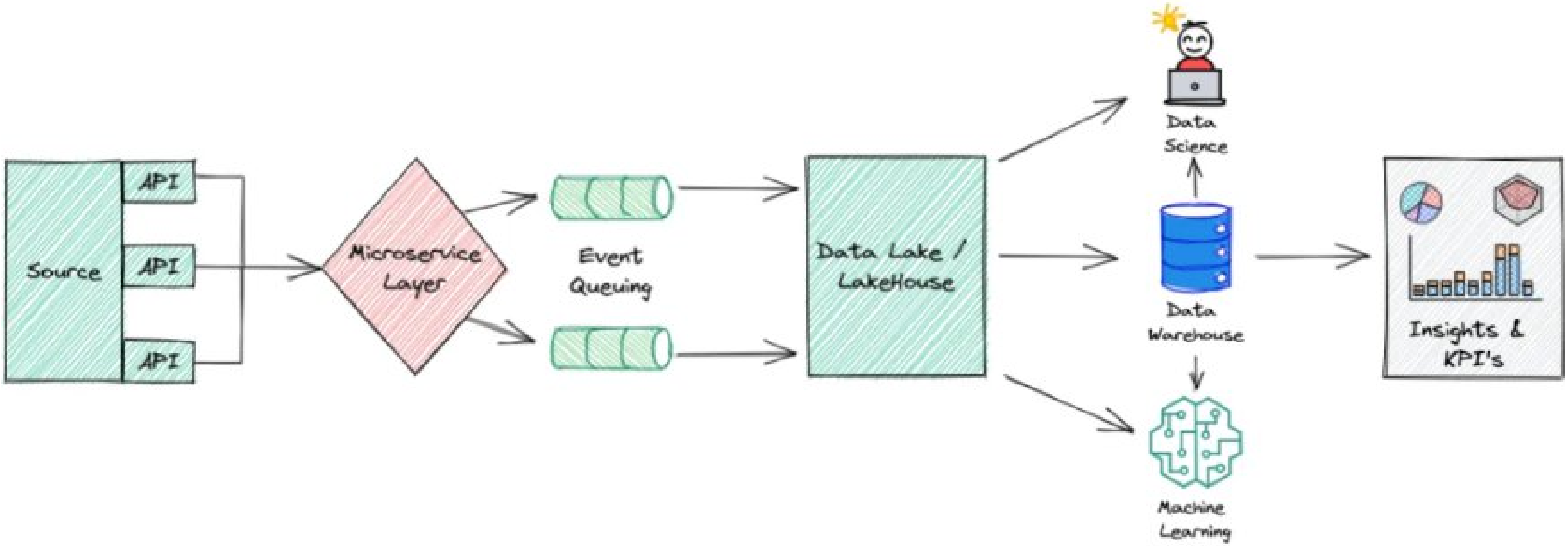
This is your **Smart Home Hub** located inside your house.

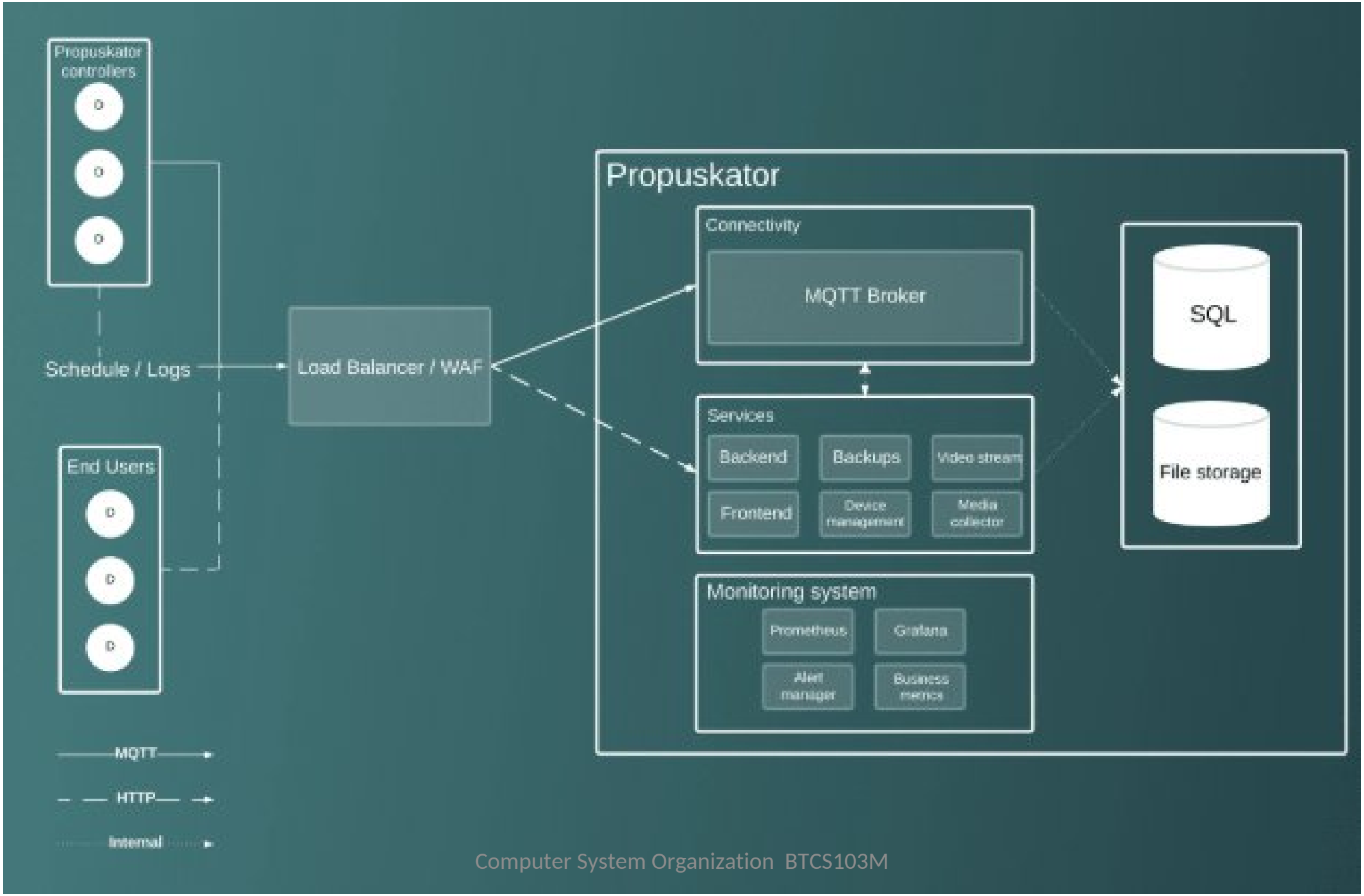
- **Local Control Response:** When the motion sensor triggers, the Hub immediately sends a command to turn on the lights. Because this happens at the **Edge**, there is almost zero delay (latency).
- **Privacy & Data Caching:** Instead of sending a constant video stream from your doorbell to the internet, the Hub can analyze the video locally to see if it's a "Person" or just a "Cat." It only alerts you (via the cloud) when necessary.
- **Offline Reliability:** If your ISP has an outage and you lose internet, your smart switches and automated schedules still work because the "Edge Node" is local.

3. The Cloud Layer (The "Remote Manager")

This is the manufacturer's server (e.g., AWS, Google Cloud, or Apple iCloud).

- **Remote Access:** When you are at work and check your phone to see if the front door is locked, your phone talks to the **Cloud**, which then talks down to your **Edge Hub**.
- **Big Data & Learning:** The cloud analyzes your energy usage over a month to suggest a more efficient heating schedule.
- **Updates:** The cloud pushes firmware updates to the Hub to ensure your home stays secure against new digital threats.


Source
API
API
API
Microservice Layer
Event Queuing
Data Lake / LakeHouse
Data Science
Data Warehouse
Machine Learning
Insights & KPI's


Propuskator controllers
Schedule / Logs
End Users
Load Balancer / WAF
Propuskator
Connectivity
MQTT Broker
Services
Backend
Backups
Video stream
Frontend
Device management
Media collector
Monitoring system
Prometheus
Grafana
Alert manager
Business metrics
SQL
File storage
MQTT
HTTP
Internal

Data Acquisition (DAQ) in IoT

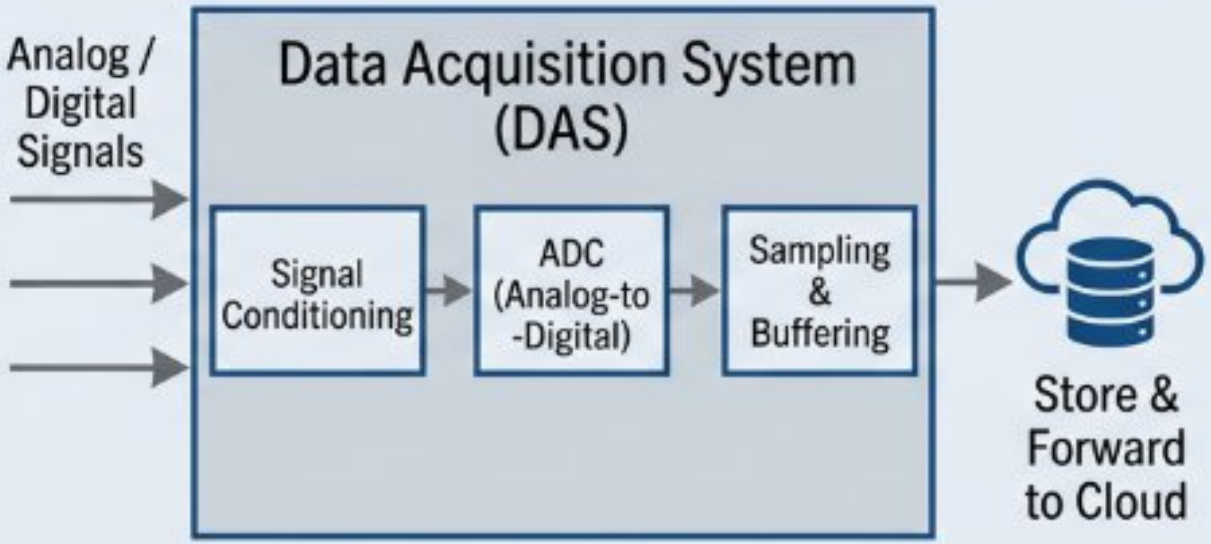
Data Acquisition (DAQ) is the process of sampling signals that measure real-world physical conditions—like temperature, pressure, or vibration—and converting them into digital numeric values for processing, storage, or analysis in IoT systems.

DAQ Components and Workflow

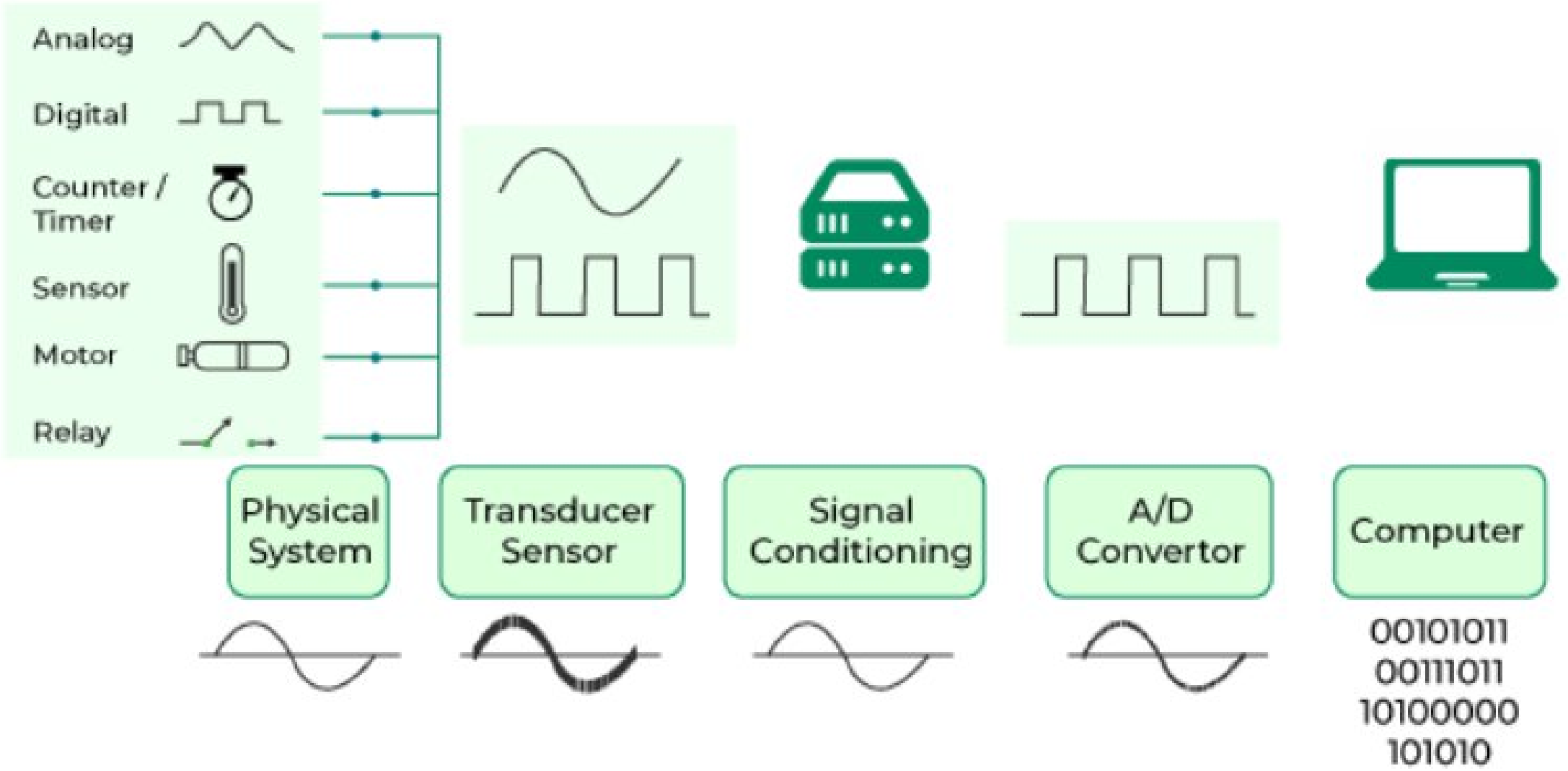
DAQ systems follow a structured flow: sensors capture analog signals, which undergo conditioning, multiplexing, analog-to-digital conversion, and digital processing before transmission to IoT gateways or cloud platforms.

Data Acquisition

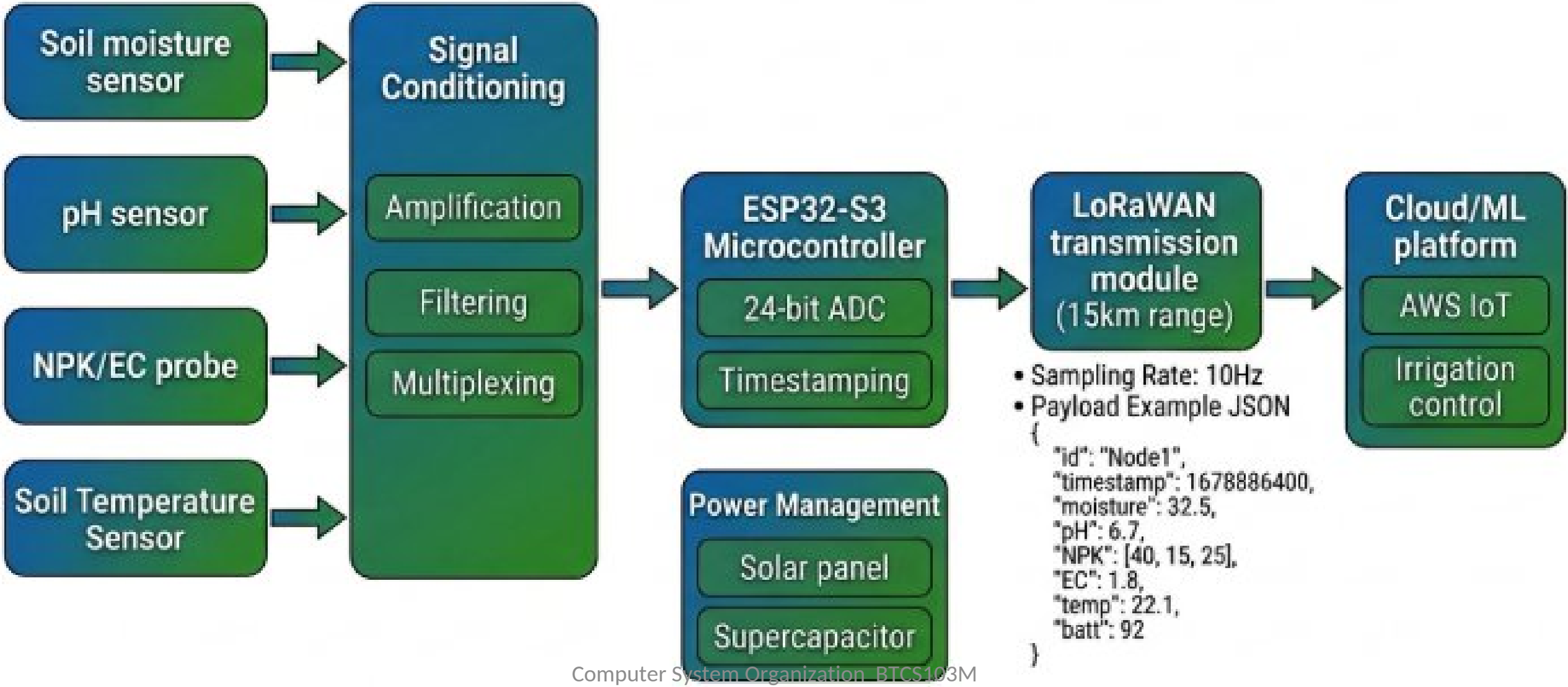
Data Acquisition Steps:
Sensing → Conditioning → ADC → Transmission



Data Acquisition System Component



IoT Precision Agriculture Soil Monitoring System - Technical Block Diagram



Data Acquisition (DAQ) in IoT

Sensors/Transducers: Convert physical parameters (e.g., DHT22 for humidity) into electrical signals (voltage/current).

Signal Conditioning: Amplifies weak signals, filters noise (low-pass filters), and provides isolation to prevent ground loops.

Multiplexer: Routes multiple sensor channels to a single ADC for efficient sampling

Analog-to-Digital Converter (ADC): Digitizes signals (e.g., 16-bit SAR ADC at 100 kS/s).

Digital Processing: Microcontroller (e.g., ESP32) handles buffering, timestamping, and protocol encapsulation (MQTT).

Data Transmission: Wireless interfaces (Wi-Fi, LoRa) send to IoT platforms..

Data Acquisition (DAQ) in IoT

Timestamping in a microcontroller involves attaching precise time markers to events, data samples, or interrupts using hardware timers or real-time clocks (RTC), essential for IoT data acquisition sequencing and synchronization.

Key Parameters:

Sampling Rate: Frequency of measurements (e.g., 1 kHz for vibration); must exceed 2x signal bandwidth (Nyquist theorem).

Resolution: Bits per sample (e.g., 12-bit = 4096 levels); higher values detect finer changes.

IoT Precision Agriculture Soil Monitoring (Advanced DAQ Example)

Precision agriculture uses IoT DAQ to monitor soil moisture, pH, NPK nutrients, and temperature in real-time across large farms, enabling automated irrigation and fertilizer application for 25-30% yield improvement.

Flow Explanation:

- Sensors capture analog signals (capacitive moisture 0-3V, pH 0-14→0-5V).
- Signal Conditioning amplifies (gain=100), filters noise (50Hz rejection), multiplexes 4 channels.
- ESP32-S3 performs 24-bit ADC conversion at 10 Hz, timestamps with RTC+micros(), runs edge ML for irrigation decisions.
- LoRaWAN transmits 15km to gateway (vs WiFi's 100m limitation).
- Cloud fuses farm-wide data, controls valves via 4-20mA actuators

2. Data Acquisition (DAQ)

Data Acquisition is the method of sampling signals that measure real-world physical conditions and converting them into digital numeric values.

- **Components of a DAQ System:**
  - **Sensors/Transducers:** Convert physical parameters (temp, pressure) into electrical signals.
  - **Signal Conditioning:** Raw signals are often noisy or weak. This step involves amplification, filtering, and isolation.
  - **Analog-to-Digital Converter (ADC):** Converts the conditioned analog signal into a digital format.

What is Data Integration in IoT?

Definition: Data Integration in IoT means collecting data from different devices/sources and combining it into a unified system for processing and decision-making. Data Integration in IoT refers to the process of collecting data from multiple IoT devices and combining it into a unified system for processing, analysis, and decision-making. In simple words: "Different sensors → one platform → meaningful information".

What is Data Integration in IoT?

Why Data Integration is Important?

- Devices are heterogeneous (different formats & protocols)
- Data comes from multiple sources
- Need centralized analysis
- Helps in: Real-time monitoring
- Automation Smart decision-making

IoT Data Integration Framework (Step-by-Step)

Step 1: Data Collection Sensors collect raw data such as temperature, humidity, motion, etc.

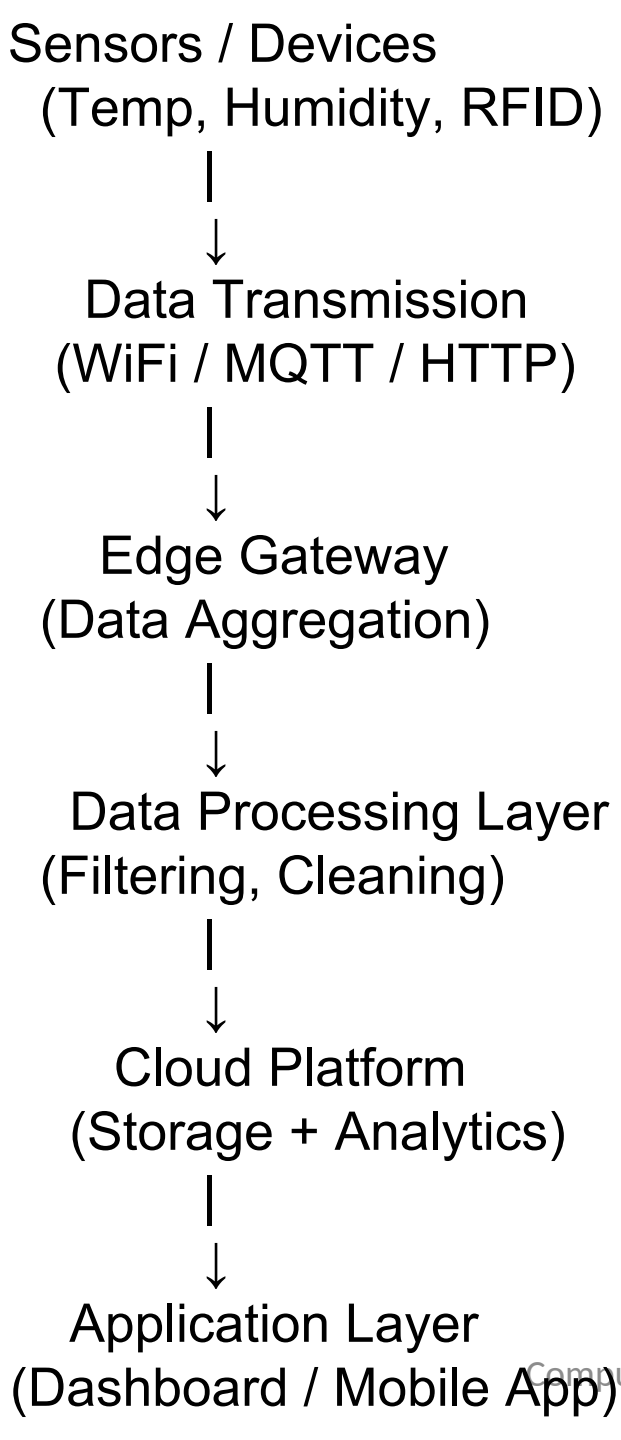
Step 2: Data Transmission Data is transmitted using communication protocols like: MQTT, HTTP , CoAP

Step 3: Data Aggregation (Gateway)An edge device (e.g., Raspberry Pi) gathers and combines data from multiple sensors.

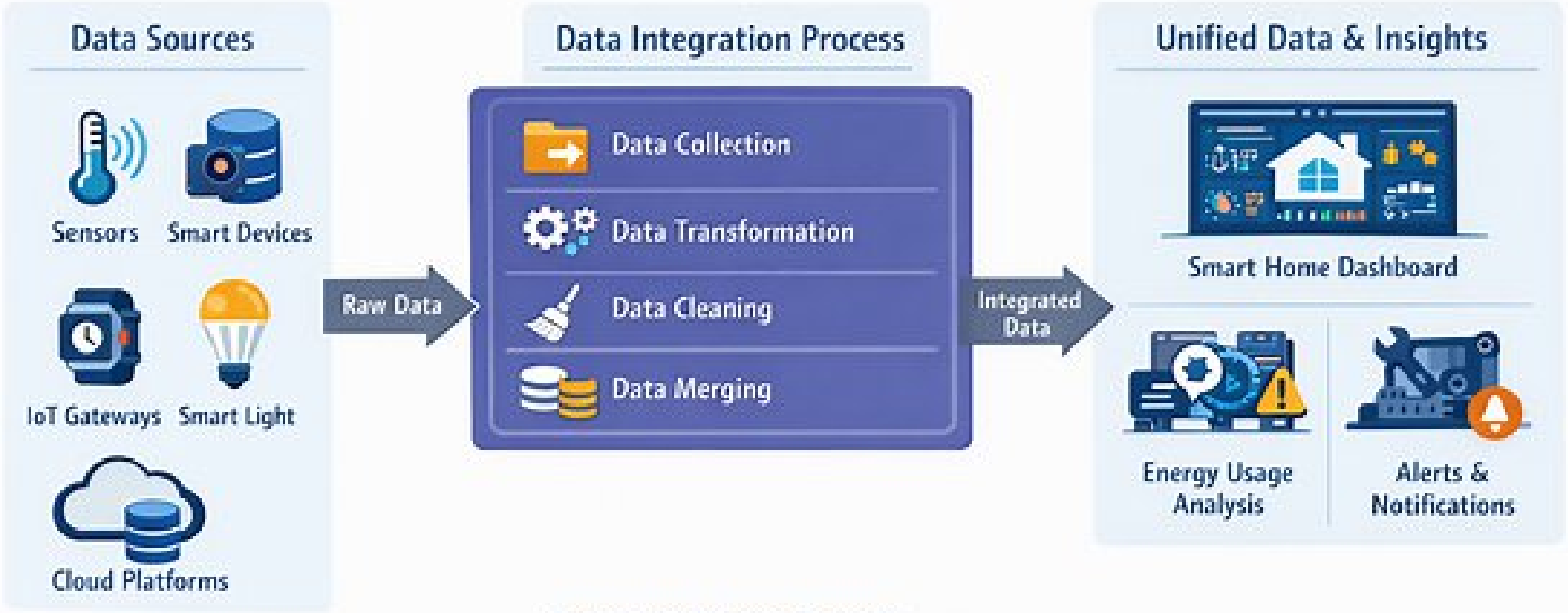
Step 4: Data Processing Data is processed through: Filtering ,Cleaning , Formatting

Step 5: Data Storage Processed data is stored in: Cloud platforms Databases

Step 6: Data Analytics & Visualization Data is analyzed Results are shown via dashboards, alerts, and reports

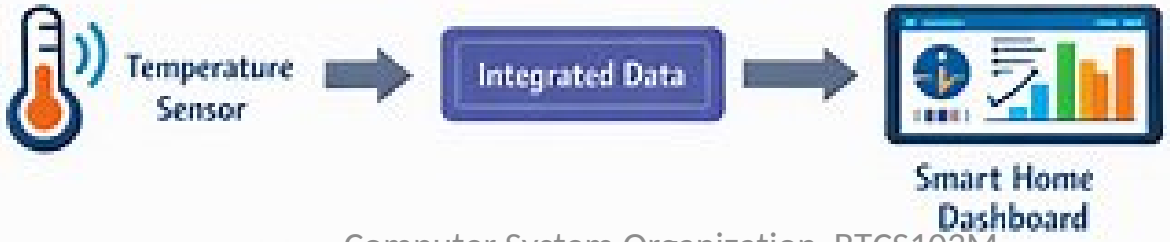

Sensors / Devices
(Temp, Humidity, RFID)
Data Transmission
(WiFi / MQTT / HTTP)
Edge Gateway
(Data Aggregation)
Data Processing Layer
(Filtering, Cleaning)
Cloud Platform
(Storage + Analytics)
Application Layer
(Dashboard / Mobile App)

Data Integration in IoT Systems



Example: IoT System Scenario

Monitoring a Smart Home



1. Data Sources (Left Side of Diagram)

This section represents **where data is generated**.

Components:

- **Sensors** → Temperature, humidity, etc.
- **Smart Devices** → Smart appliances
- **IoT Gateways** → Intermediate devices
- **Cloud Platforms** → Storage systems

These devices generate **raw data** continuously.

Data Integration Process (Middle Section)

This is the **core part** of the system where raw data is converted into meaningful information.

Steps:

1. Data Collection

•Data is gathered from all sensors and devices

2. Data Transformation

•Data is converted into a **standard format**

•Example: Celsius → Fahrenheit, analog → digital

3. Data Cleaning

•Removes:

- Errors
- Duplicate values
- Missing data

4. Data Merging

•Data from different sources is **combined into one dataset**

Output: **Integrated Data**

3. Unified Data & Insights (Right Side)

After integration, data is used for **analysis and decision-making**.

Applications:

- **Smart Home Dashboard** → Displays real-time data
- **Energy Usage Analysis** → Tracks power consumption
- **Alerts & Notifications** → Sends warnings (e.g., high temperature)

4. Flow Explanation

Raw Data → Processing → Integrated Data → Insights

- Raw data comes from sensors
- It is processed in the integration layer
- Final output is **useful information for users**

Bottom Example (From Diagram)

Temperature Sensor → Integrated Data → Dashboard

- Sensor collects temperature
- Data is processed and integrated
- Displayed as graphs/charts on dashboard

The diagram explains that **IoT data integration converts raw device data into meaningful insights**, enabling **automation, monitoring, and intelligent decision-making**.

Data integration combines **multi-source IoT data**

Includes:

- Collection Transformation Cleaning Merging

Produces **useful insights**

Used in:

- Smart homes Smart cities Healthcare Industry

Example: Smart Agriculture System

Scenario: A smart farming system uses IoT sensors to monitor field conditions and automate irrigation.

Devices Used: Temperature Sensor Soil Moisture Sensor Humidity Sensor Water Pump (Actuator)

Working:

- Sensors collect environmental data
- Data is transmitted via Wi-Fi / MQTT , Gateway
- integrates all sensor data
- Cloud platform stores and analyzes data
- Application displays data on dashboard
- If soil moisture is low → Water pump is automatically turned ON

5. Example: Smart Home System

Scenario:

A smart home uses IoT devices to monitor and control the environment.

Devices:

•Temperature sensor

•Smart light

•Energy meter

•Motion sensor

Working (Based on Diagram):

1.Data Collection

1. Temperature sensor sends temperature data
2. Smart devices send usage data

2. Data Transformation

1. Convert all data into a common format

1. Data Cleaning

1. Remove incorrect readings

2. Data Merging

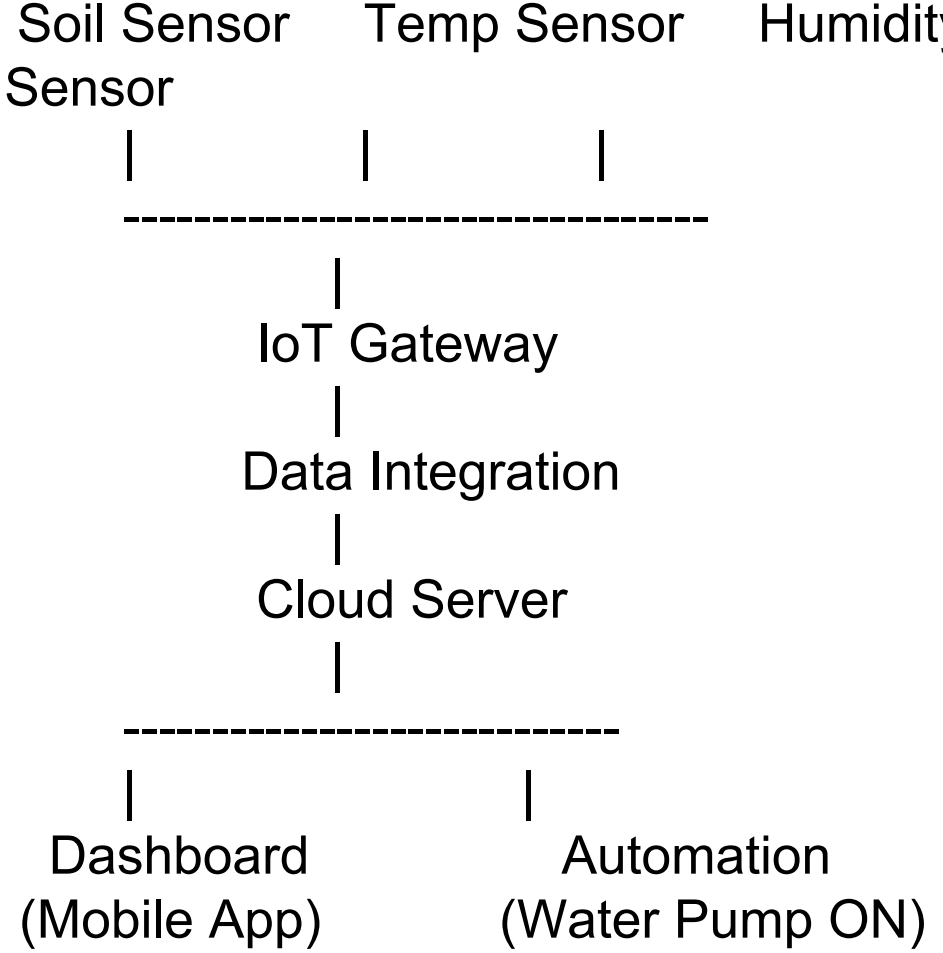
1. Combine temperature + energy + motion data

3.Integrated Data Output

1. Sent to cloud/dashboard

Result:

- User sees data on **Smart Home Dashboard**
- System performs actions:
  - Turn ON AC if temperature is high
  - Send alert if unusual activity detected


Soil Sensor
Temp Sensor
Humidity Sensor
IoT Gateway
Data Integration
Cloud Server
Dashboard (Mobile App)
Automation (Water Pump ON)