

UNIT 3

IoT Platform Design Methodology

(6 Hours) — Internet of Things (CACS460)

⚠ Please Note

These notes were prepared quickly and **may contain errors or omissions**. Please cross-check important definitions, formulas, code and diagrams with your textbook, class notes and the official TU syllabus before relying on them in exams.

3.1 Why a Design Methodology?

Designing an IoT system involves hardware, networking, cloud and applications. A **design methodology** gives a step-by-step, technology-independent process so that requirements are captured first and technology choices come later. The syllabus uses the **10-step IoT design methodology** (Bahga & Madiseti) illustrated through a **weather monitoring** case study.

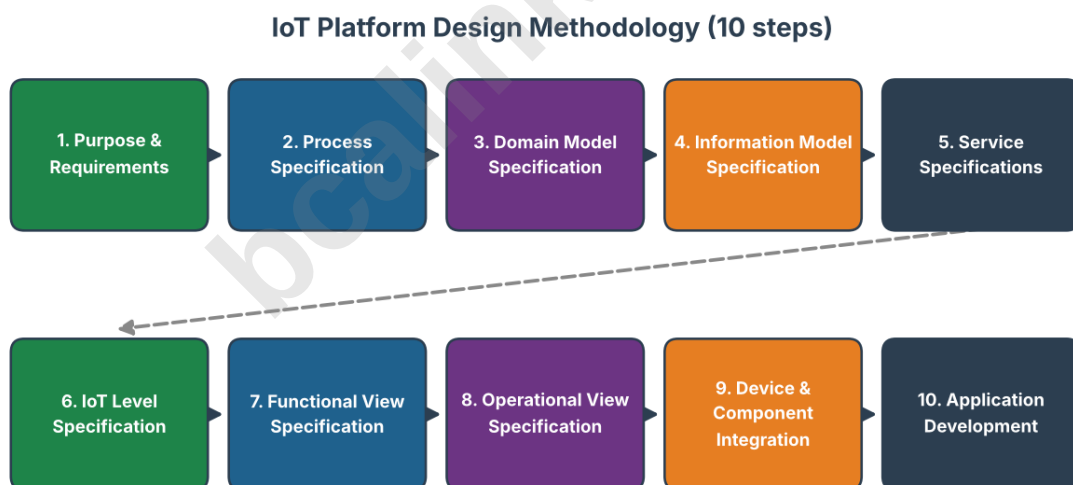


Figure 3.1 — The 10-step IoT platform design methodology

3.2 IoT Levels and Deployment Templates

Before step 6 we need to know the six **IoT levels**, which describe how devices, local processing, cloud and applications are arranged.

Level	Description	Example
Level 1	Single node; sensing, local storage and local analysis; app may be cloud-hosted. For low data volume and simple analysis.	Home automation with one controller node
Level 2	Single node; analysis done locally but data stored in the cloud ; for large data with light analysis.	Smart irrigation: local decision, cloud logging
Level 3	Single node; both storage and analysis in the cloud ; for big data and computationally heavy analysis.	Tracking a package with cloud-side analysis
Level 4	Multiple nodes; local analysis in nodes, storage in cloud; observer nodes can process the data.	Noise monitoring across a city with local analysis
Level 5	Multiple end nodes plus one coordinator node ; coordinator sends data to the cloud for analysis.	Forest-fire detection with WSN
Level 6	Multiple independent end nodes send data to cloud; centralised controller; cloud analysis and app control.	Weather monitoring system

Note: Different books describe the levels slightly differently, so check the description given in your class notes. The essential idea is: more nodes and heavier analysis → move processing and storage to the cloud.

3.3 The Ten Steps

Step 1 — Purpose and Requirements Specification

State what the system is for and gather requirements: **purpose & behaviour, system management, data analysis, application deployment, security**.

Step 2 — Process Specification

Define the use cases as a **process model** (flow chart or state diagram): e.g., “every N seconds, read sensors; if reading received → publish; if value out of range → alert”.

Step 3 — Domain Model Specification

Describe the main **concepts, entities and objects** and their relationships: *Physical Entity* (the thing/environment being monitored), *Device* (hardware carrying sensors), *Resource* (software on the device or network), *Service* (provides access to resources, e.g., a web service).

Step 4 — Information Model Specification

Defines the structure of all information in the system — attributes, relationships, data types — but not how it is stored. For weather monitoring: *Virtual Entity* = weather station; attributes = temperature, humidity, pressure, location, timestamp.

Step 5 — Service Specifications

Define services that operate on the information (e.g., “Controller service”, “Publish-data service”, “Analysis service”), their types, inputs/outputs, endpoints, schedules and pre/post-conditions.

Step 6 — IoT Level Specification

Choose one of the six IoT levels from the requirements. The textbook’s weather monitoring system uses **Level 6**: multiple independent stations send data to the cloud, where it is stored and analysed.

Step 7 — Functional View Specification

Maps the system to functional groups: **Device, Communication, Services, Management, Security, Application** (the same functional blocks studied in Unit 1).

Step 8 — Operational View Specification

Decides the real deployment choices: device options (Arduino/Pi), network (WiFi), protocols (MQTT/HTTP), cloud service, storage, application hosting.

Step 9 — Device and Component Integration

Integrate sensors, board, network and cloud into a working prototype and connect them physically and logically.

Step 10 — Application Development

Build the user-facing application: web dashboard/mobile app with charts, alerts and controls; test and deploy.

3.4 Case Study — IoT Weather Monitoring System

Goal: collect temperature, humidity, pressure, rainfall and wind speed from a station, store the readings in the cloud, display them on a dashboard and raise alerts (e.g., heavy rain). This is highly relevant for hilly and flood-prone regions.

Weather Monitoring System — Block Diagram

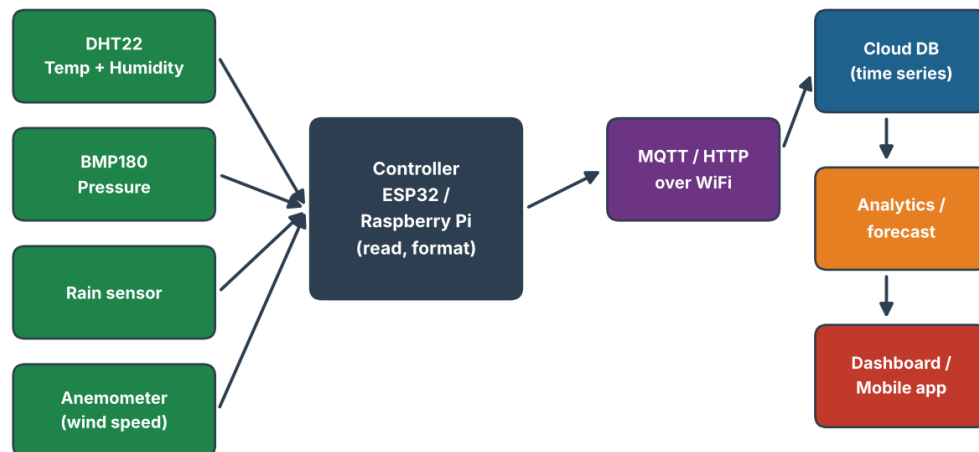


Figure 3.2 — Weather monitoring system block diagram

Step	Weather monitoring decision
1. Purpose & requirements	Monitor temperature, humidity, pressure, rain and wind; data every minute; alerts; web dashboard; secure access.
2. Process specification	Loop: read sensors → format → publish to cloud → if rainfall > threshold, send alert.
3. Domain model	Physical entity: the environment/weather; Device: ESP32 with sensors; Resource: sensor drivers, MQTT client; Service: weather service.
4. Information model	Virtual entity “Weather Station” with attributes temperature, humidity, pressure, rain, wind speed, time.
5. Service specification	Controller service (sampling), publish service (MQTT/REST), analysis service (alerts/forecast).
6. IoT level	Level 6: many independent stations, cloud storage/analysis, centralised controller/app.
7. Functional view	Device = sensors; Communication = WiFi/MQTT; Services = publish/query; Management = remote config; Security = TLS and auth; Application = dashboard.
8. Operational view	ESP32/Raspberry Pi, WiFi, MQTT broker, cloud DB, web hosting.
9. Integration	Wire sensors via I ² C/I-Wire, flash firmware, test connectivity, calibrate sensors.
10. Application	Web/mobile app showing live values and charts; push notifications.

Sample Code — Publishing Weather Data (Raspberry Pi, Python)

```
import time, json, board, adafruit_dht
import paho.mqtt.client as mqtt

dht = adafruit_dht.DHT22(board.D4)
client = mqtt.Client()
client.connect("broker.hivemq.com", 1883)      # public test broker (not secure)
client.loop_start()

while True:
    try:
        data = {"temp": dht.temperature, "hum": dht.humidity, "ts": time.time()}
        client.publish("college/weather/station1", json.dumps(data))
        print("sent", data)
    except RuntimeError:
        pass
    time.sleep(60)
```

Note: For real projects use a private broker with username/password and TLS instead of a public test broker.

Example 3.1 — Choosing an IoT level

A farm has 20 soil-moisture nodes spread over a large area. A coordinator collects readings and sends them to a cloud server that decides when to irrigate. Which IoT level?

Solution

Multiple end nodes + one coordinator forwarding to the cloud for analysis → **IoT Level 5**.

Key Takeaways

- ✓ The methodology has 10 steps, moving from requirements to application development.
- ✓ Steps 2–5 (process, domain, information, service) are technology-independent models; steps 6–8 select level and views.
- ✓ IoT levels 1–6 differ in number of nodes and where analysis/storage happen.
- ✓ Weather monitoring case: sensors → controller → MQTT/WiFi → cloud DB → analytics → dashboard.
- ✓ Always secure the design (authentication, TLS) and plan management (remote configuration).

— End of Unit 3 —