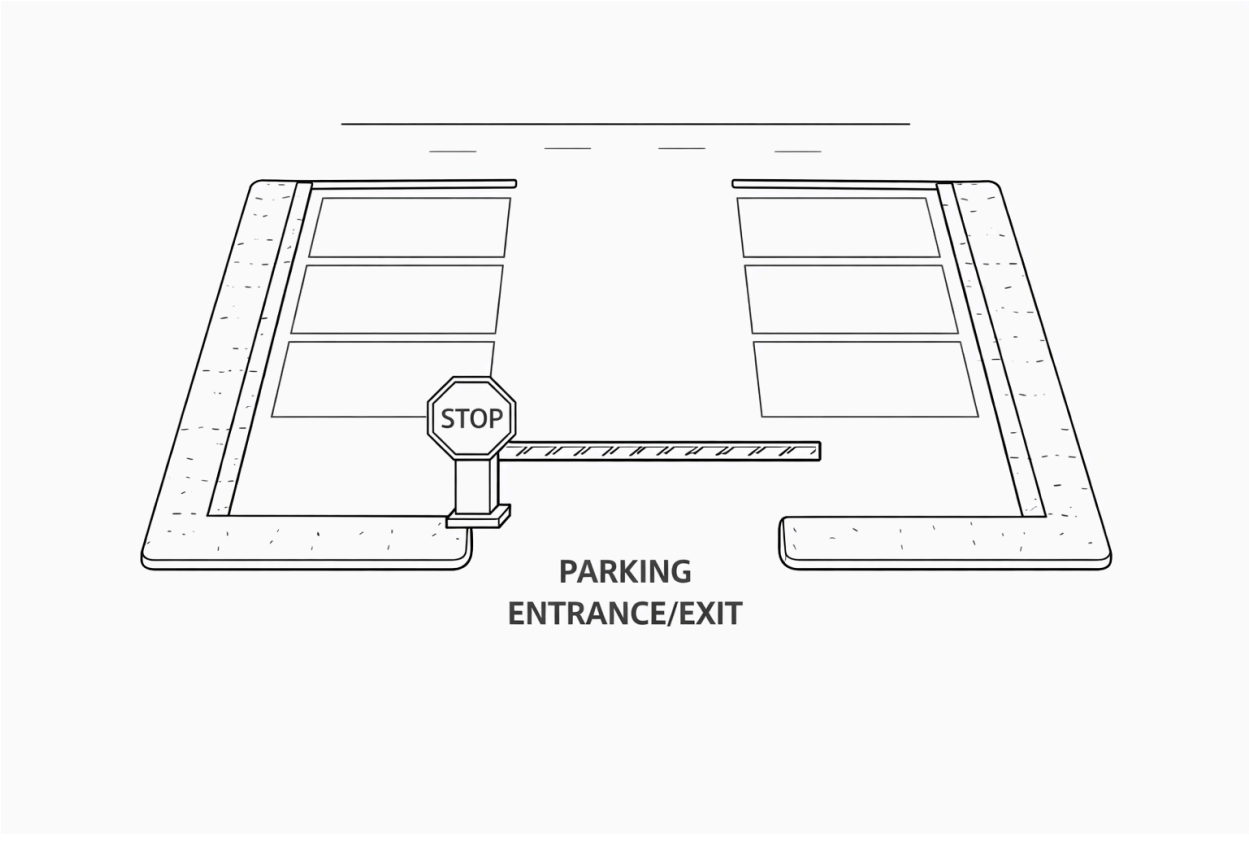
University Echahid Hamma Lakhdar El Oued
Faculty of Exact Sciences - Department of Computer Science
Level: Master 2 (2025-2026)
Module: Developing IoT application - typical solution-

Exercise: 20 pts
Smart Parking:

Drivers enter a small parking area with six places. Each place can be empty or occupied. Cars enter through a barrier gate. The gate must not open if the parking is full. It should open only when at least one place is empty.

When a car wants to exit, the gate must open to allow it out. The whole system must be monitored and controlled from a pc.

1- Add the list of things, microcontrollers, and any device with the name in the following drawing (**2 pts**).



List of Things, sensors, and actuators:

Things (IoT objects):

- Smart Parking Controller (main IoT node)
- Monitoring PC (Dashboard / Control)

Sensors:

- 6× presence sensors (one per spot): IR sensor / ultrasonic / magnetic sensor (detect occupied/empty)
- Entrance car detector (at gate): IR/ultrasonic (detect car waiting to enter)
- Exit car detector (at gate): IR/ultrasonic (detect car leaving)

Actuators / Outputs:

- Barrier gate motor/servo (open/close)
- LED indicators (FULL / AVAILABLE or per-spot LEDs)
- Buzzer (warning when full)

Microcontroller + communication:

- ESP32 / ESP8266 (NodeMCU) (Wi-Fi) or Arduino + Wi-Fi module.

2- General behavior (**2 pts**)**:**
The system always counts available spots (0 to 6).
If parking is full (0 free) → do not open the gate for entry.
If at least 1 spot is free and a car is detected at entrance → open gate, then close after the car passes.
For exit: when a car is detected at exit → open gate to allow leaving, then close.
The PC displays: each spot status + free count + gate state, and can send commands (open/close/manual mode).
3- Specific Behavior of Each Thing (**4 pts**):

Write about the behaviour, the microcontroller, sensors, actuators, monitoring PC.

A) Each Parking Spot (Spot1…Spot6)

- Sensor reads: **occupied = 1**, **empty = 0**
- Sends status to controller
- (Optional) Spot LED: green empty / red occupied

B) Smart Parking Controller (ESP32/NodeMCU)

- Reads 6 spot sensors + entrance + exit sensors
- Computes: `free = 6 - occupied_count`

- Logic:
  - **Entry request** (entrance sensor triggered):
    - if `free > 0` → open gate
    - else → keep gate closed + show "FULL"
  - **Exit request** (exit sensor triggered):
    - open gate (exit always allowed)
- Publishes data to PC (spot states, free count, gate state)
- Receives commands from PC (manual open/close, reset, mode auto/manual)

C) Gate System (Actuator)

- Actuator: **servo motor / DC motor + driver**
- Actions:
  - `OPEN` for a few seconds (or until sensor shows car passed)
  - then `CLOSE`

D) Monitoring PC

- Dashboard shows:
  - Spot1..Spot6: empty/occupied
  - free places number
  - gate open/closed
- Sends control commands:
  - open/close gate
  - switch **Auto / Manual**
  - emergency stop

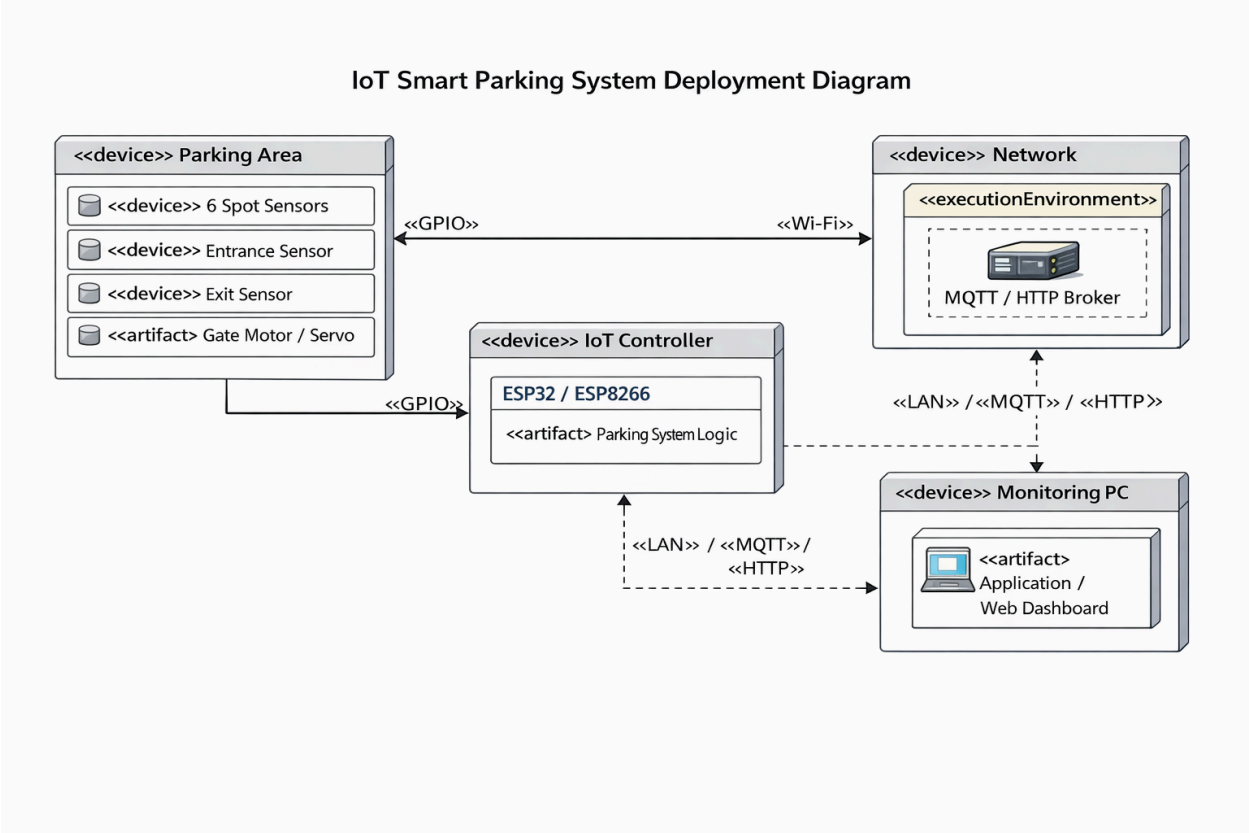
4- Connectivity and Protocols (2 pts):

Wi-Fi from ESP32/ESP8266 to router
Protocol: **MQTT** (publish/subscribe)
Topics example:

- `parking/spots`
- `parking/free`
- `parking/gate/state`
- `parking/gate/cmd`

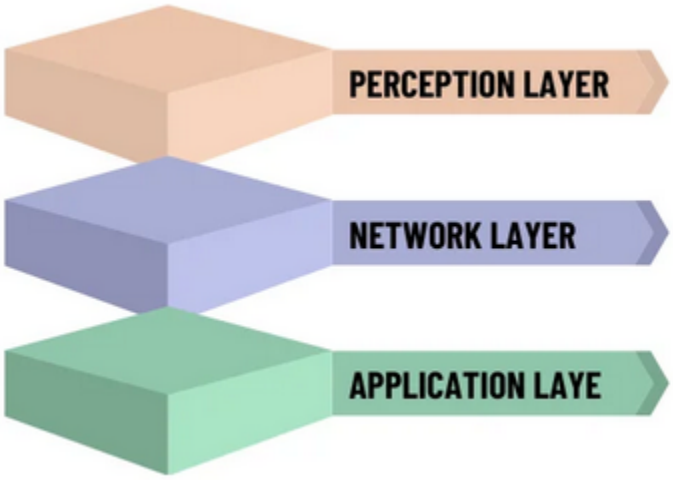
5- Deployment diagram (5 pts):

Model the system behavior using the deployment diagram.



5- Draw the IoT Architecture **(2pts)**.

Describe how an IoT architecture can be applied to this parking system. Explain how information is collected, transmitted, and processed to control the gate, and identify the role of each IoT layer. **(3pts)**



4 layers is also accepted

1. **Perception Layer (Sensing/Actuation)**

- Spot sensors (6), entrance/exit sensors
- Gate motor/servo, LEDs

2. **Network / Transport Layer**

- Wi-Fi connection
- MQTT or HTTP messages between controller and PC/broker

3. **Application Layer**

- Controller logic (counting free spots, decisions)
- MQTT broker + data formatting (JSON), storage
- PC dashboard (monitoring, control, alerts “FULL”)
- Policies/statistics: peak hours, reports, maintenance logs