Assignment: RFID Card Top-Up System

Instructor: Gabriel Baziramwabo

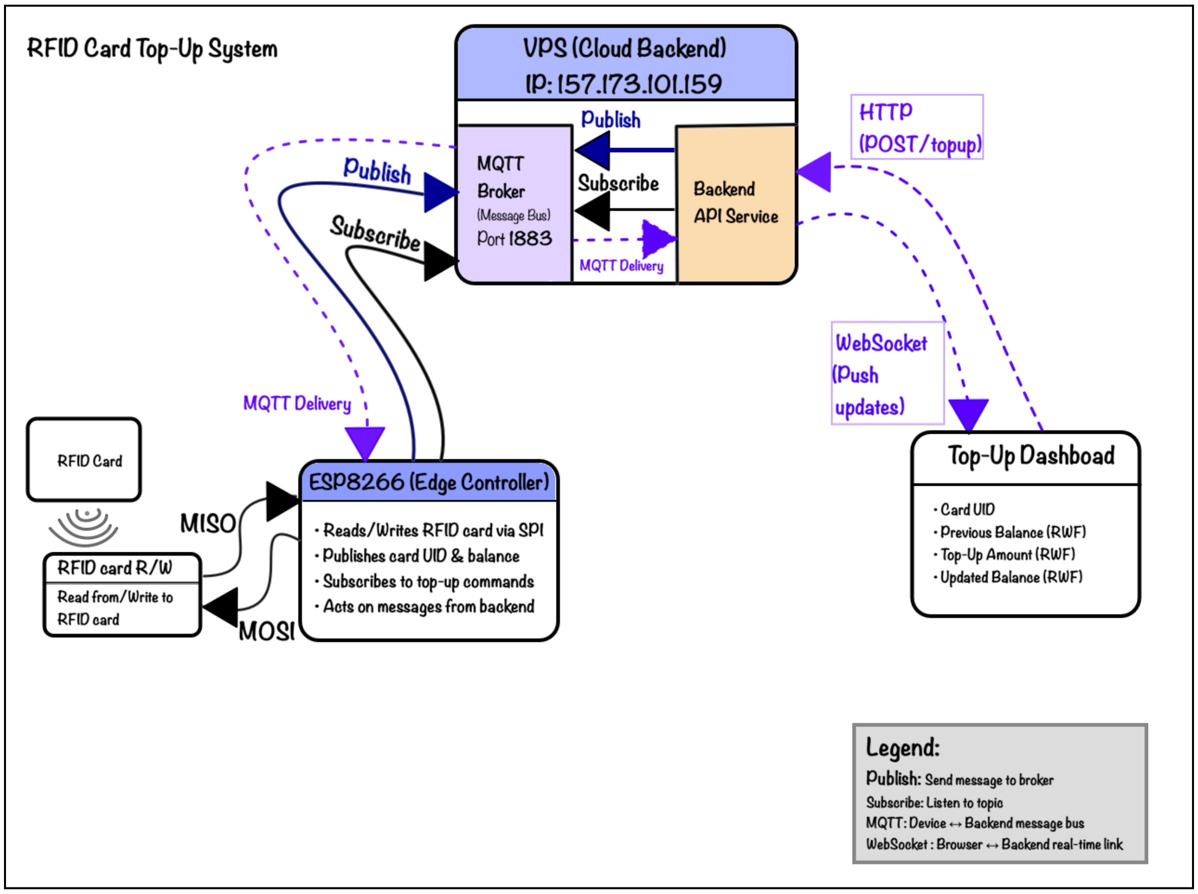
Keywords: *RFID Systems, IoT Architecture, MQTT Messaging, Publish–Subscribe Model, Edge Controllers, Cloud-Based Backend Services, Real-Time Web Communication, Embedded Networking*

1. Objective

Design and implement a complete RFID card top-up system using an ESP8266, a cloud backend (VPS), and a web dashboard.

The system must allow multiple teams to operate simultaneously on the same MQTT broker without interfering with one another.

2. System Architecture


RFID Card Top-Up System
VPS (Cloud Backend)
IP: 157.173.101.159
Publish
MQTT Broker (Message Bus) Port 1883
Subscribe
Backend API Service
MQTT Delivery
HTTP (POST/topup)
Publish
Subscribe
MQTT Delivery
WebSocket (Push updates)
RFID Card
MISO
MOSI
RFID card R/W
Read from/Write to RFID card
ESP8266 (Edge Controller)
· Reads/Writes RFID card via SPI
· Publishes card UID & balance
· Subscribes to top-up commands
· Acts on messages from backend
Top-Up Dashboad
· Card UID
· Previous Balance (RWF)
· Top-Up Amount (RWF)
· Updated Balance (RWF)
Legend:
Publish: Send message to broker
Subscribe: Listen to topic
MQTT: Device ↔ Backend message bus
WebSocket : Browser ↔ Backend real-time link

RFID Card Top-Up System Architecture

Your implementation must follow this architecture strictly:

- **ESP8266 (Edge Controller)**
  - Reads and writes RFID card data
  - Publishes card UID and balance via MQTT
  - Subscribes to top-up commands via MQTT
  - *Must NOT use HTTP or WebSocket*
- **Backend API Service (VPS)**
  - Receives user commands via HTTP
  - Communicates with ESP8266 via MQTT
  - Pushes real-time updates to browsers via WebSocket
- **Web Dashboard (Browser)**
  - Sends top-up requests via HTTP
  - Receives balance updates via WebSocket
  - *Must NOT communicate directly with MQTT*

3. Critical Rule: MQTT Topic Isolation

All teams share the same MQTT broker.

To avoid conflicts, each team must use a unique topic namespace.

Team Identifier: Each team must choose a unique team_id (example: team01, alpha, y2b_grp3).

Base Topic: rfid/<team_id>/

3.1. Required MQTT Topics

3.1.1. Card status (ESP8266 → Broker)

rfid/<team_id>/card/status

Payload example:

```
{
  "uid": "A1B2C3D4",
  "balance": 3000
}
```

3.1.2. Top-up command (Backend → ESP8266)

rfid/<team_id>/card/topup

Payload example:

```
{
  "uid": "A1B2C3D4",
  "amount": 500
}
```

3.1.3. Updated balance (ESP8266 → Broker)

rfid/<team_id>/card/balance

Payload example:

```
{
  "uid": "A1B2C3D4",
  "new_balance": 3500
}
```

3.2. Forbidden Practices

Using generic topics such as:

rfid/card
rfid/topup
balance

Subscribing to wildcard topics such as:

rfid/#

Publishing or subscribing to another team's namespace

*::**Any team interfering with others will lose marks.***

4.HTTP & WebSocket Requirements

- The backend must expose at least:

POST /topup

- The dashboard must:
    - Send top-up commands via HTTP
    - Receive real-time updates via WebSocket
    - ***Polling is not allowed***

5. Submission Requirements

- Each team member must individually push the project files to their own GitHub repository
- The GitHub repository link must be submitted individually via a Google Form shared later.
- The live dashboard URL (hosted on the VPS or any other hosting platform) must be submitted via the same Google Form
- A quiz link will also be included in the same form for assessment

⚠️ **Repositories must be:**

- Public
- Well structured
- Clearly documented *(README.md is mandatory)*

6. Evaluation Focus

- Correct and appropriate use of HTTP, WebSocket, and MQTT
- Proper MQTT topic isolation on the shared broker
- Complete end-to-end system functionality
- System stability in a shared-broker environment
- Code clarity, structure, and repository organization

📌 **Note:** The **Arduino IDE** may be used; however, **MicroPython is strongly encouraged**.

Golden Rule

> ***Devices speak MQTT.***
> ***Users speak HTTP and WebSocket.***
> ***The backend translates between them.***

Final note

> ***This assignment is designed to test real-world IoT system design, not just code execution. Architectural violations will be penalized even if the system appears to work.***