Ahsanullah University of Science and Technology

Department of Computer Science and Engineering

CSE-4264 Internet of Things Lab

RFID-Enabled Smart Water Dispensing System with Card Control and Live Billing Usage Tracking
Group : 04

Submitted by

1. Sadman Salman Saad 20210104114
2. Abdullah Al Maruf 20210104116
3. Sharun Tawsif 20210104124
4. Nafisa Tabassum Shouty 20210104127

Submitted To:

Mr. Mustofa Ahmed & Ms.Nusrat Jahan

Submission Date : 23/07/2025

Contents

1 Progress Summary

In this phase, we focused on laying the foundation for the core functionality of our RFID-enabled smart water dispenser system. Specifically, we implemented a multiple tap simulation using two RDM6300 RFID readers connected to an ESP32 microcontroller.This allows multiple users to access individual taps using their own unique RFID cards.

2 IoT Workflow for RFID-Based Dual-Line Water Dispensing System

The IoT workflow describes the sequence of steps involved in collecting, processing, transmitting, and acting on data from the RFID-controlled water dispensing system. The following components and steps illustrate the complete operation of the system:

2.1 System Overview

The system utilizes two RFID readers (RDM6300), an ESP32 microcontroller, two water pumps, and supporting components to control water dispensing based on user authentication via RFID cards. Billing and data logging are managed through integration with Google Sheets. Status indicators are provided via LEDs and a buzzer.

2.2 IoT Workflow Steps

2.2.1 Device Enrollment & Authentication

The system uses two RDM6300 RFID readers connected to an ESP32 microcontroller. Each reader is assigned to a specific water line (left or right). When a user holds a 125kHz RFID card near a reader, the ESP32 detects the card ID and identifies which line was triggered. This authentication step ensures that only authorized cards can access the water dispensing system.

2.2.2 Data Acquisition (Sensing/Actuation)

Once a card is scanned, the RDM6300 module transfers the card's unique ID to the ESP32. Simultaneously, the ESP32 begins a dispensing session by initiating a timer and enabling the associated water pump. The duration of water flow is tracked in real time based on how long the user holds the RFID card over the sensor.

2.2.3 Data Communication

The ESP32, connected via Wi-Fi, sends real-time data to a Google Sheets document using HTTP requests or a Web App Script endpoint. Transmitted data includes RFID card ID, which line (left or right) was used, start and end time of dispensing, and calculated water usage duration. This communication enables live remote monitoring and logging.

2.2.4 Edge Processing

Edge processing is performed directly on the ESP32 microcontroller. It parses the card ID, validates its authenticity, decides which water line to activate, and controls pump operation via TIP122 transistors. The ESP32 also measures how long the pump remains active, which is later used to compute billing information. This local processing reduces latency and avoids constant cloud dependence.

2.2.5 Cloud Integration

After edge processing, the ESP32 uploads all collected data to the cloud—in this case, Google Sheets—using the Google Apps Script Web API. The integration allows for centralized billing, analytics, and monitoring of system usage from any device with internet access. It also makes long-term data archival possible.

2.2.6 Data Storage

Google Sheets serves as a lightweight cloud database. Each dispensing session is recorded in a structured format with fields such as timestamp, RFID card ID, line number, water usage time, and billing amount. This setup ensures historical record-keeping and supports future data analysis.

2.2.7 Data Visualization

Logged data can be visualized through built-in features in Google Sheets like charts or pivot tables. On the hardware side, LEDs provide immediate visual feedback: a green LED lights up during water flow, indicating an active and valid session, while a red LED indicates denial of access or error. This dual-layer visualization helps both remote and on-site monitoring.

2.2.8 Decision Making / Action / Response

Based on card validation and line detection, the ESP32 activates the water pump connected to the selected line. Water flows only through that specific path. The action is initiated as long as the RFID card remains present, and once removed, the system stops the pump automatically. This ensures efficient and user-controlled operation.

2.2.9 Notification and Alert System

The system provides real-time feedback using both LEDs and a buzzer. When the system allows water flow, the green LED is turned on. If an invalid card is detected or a fault occurs (e.g., pump failure or unauthorized access), a red LED is illuminated and the buzzer is activated to alert the user immediately.

2.2.10 Security and Privacy Considerations

Each RFID card has a unique identifier, ensuring that access is personalized and traceable. Only registered card IDs are allowed to activate the system. No personal data is stored in the ESP32, and the cloud system maintains isolation of user sessions, ensuring data privacy and security against unauthorized access.

IoT Step	Description
1. Device Enrollment & Authentication	RFID card scanning from 2 readers via ESP32
2. Data Acquisition (Sensing)	RDM6300 reads RFID card, ESP32 fetches the card ID
3. Data Communication	ESP32 sends data (Card ID, Water Use) to Google Sheets via Wi-Fi
4. Edge Processing (ESP32)	Card parsing and validation from each reader
5. Cloud Integration	Google Sheets stores data for billing/logging
6. Data Storage	Google Sheets acts as a real-time database
7. Visualization	Use Google Sheets and LED
8. Action/Response	Water tap will activate based on card presence
9. Notification System	LED + Buzzer indicate status (green = water flow, red = no flow)
10. Security/Privacy	RFID tags are unique, so user data is isolated

Table 1: IoT Workflow Steps

2.3 Flowchart of IoT Workflow

```
START
  ↓
[User taps RFID card on reader 1 or 2]
  ↓
[ESP32 receives card ID from RDM6300]
  ↓
[ESP32 validates card ID]
 > If valid:
      ↓
   [Activate corresponding water pump]
      ↓
   [Green LED ON, water flows]
      ↓
   [Track time water is flowing]
      ↓
   [Send card ID + duration to Google Sheets]
      ↓
   [Turn off pump when card removed]
      ↓
   [Log complete → billing entry saved]
      ↓
      END

 > If invalid:
       ↓
   [Red LED + Buzzer ON]
       ↓
   [No pump activation]-> END
```

3 Building Blocks of IoT Architecture

3.1 Perception (Sensing) Layer

This layer is the foundation where your system interacts with the physical world, detecting user presence and controlling water flow. Components Used:

- **RDM6300 RFID Readers (x2):** Detect RFID cards for tap access.
- **ESP32 Development Board:** Controls all hardware components.
- **Water Pumps (x2):** Dispense water upon successful authentication.
- **Tip122 & Resistors (2x 220Ω):** Act as switch drivers for the pumps.
- **MP1584 Buck Converters (x2):** Convert 7.4V battery to 5V (ESP32) and 3.3V (pump).
- **Buzzer & LEDs:** Provide audio-visual feedback based on system state (green = success, red = error).

These devices connect the physical world (RFID scan, water movement) with the digital system logic.

Functionality:

- Card detection triggers a corresponding pump if card-to-tap match is valid.
- Buzzer and red LED warn users of mismatched access attempts.

3.2 Network Layer

This layer handles communication between devices and the cloud. **Technologies Used:** Wi-Fi (ESP32 built-in): Transmits billing and usage data to the internet.

Functionality:

ESP32 sends real-time usage data to Google Sheets.

3.3 Data Processing Layer

This layer processes input, performs calculations, and stores data.

Processing Steps:

- Card UID is matched locally using **ESP32**.
- If matched, corresponding pump is activated, and usage is tracked.
- Billing is calculated based on time or volume (customizable).
- Processed data (card ID, line used, water used, cost) is pushed to **Google Sheets** for logging.

Tools:

- **Local (Edge) Processing:** Done on **ESP32**.
- **Cloud Logging: Google Sheets API**.

3.4 Application Layer:

This layer is where users interact with the system or view results.

Interfaces:

- **LED Indicators:**
  - Green LED: Water is flowing.
  - Red LED: Access denied or idle state.
- **Buzzer:** Alerts on unauthorized attempts.
- **Google Sheets Dashboard (cloud-based):** Shows usage history and bills.

3.5 Security Management Layer:

This layer ensures system security and long-term stability. **Implemented:**

- Offline Protection: System logic ensures pump only activates when valid card is present.
- Basic Access Control is embedded in ESP32 logic.

Table 2: System Layers and Components/Functions

Layer	System
Perception Layer	RFID readers, pumps, buzzer, LEDs, ESP32, MP1584
Network Layer	Wi-Fi (ESP32 to Google Sheets), jumper wiring
Data Processing Layer	Logic inside ESP32: RFID read, pump control, usage tracking, LED control
Application Layer	Google Sheets receives usage + billing info based on water dispensing
Security Layer	user access control

4 Steps to Implement IoT Use Cases

4.1 Hardware Setup:

- Selected **ESP32 Development Board** as the microcontroller.
- Connected **2x RDM6300 RFID readers** via UART1 and UART2 for dual tap authentication.
- Connected **2x water pumps** using **TIP122 transistors** and **220Ω resistors** for controlled activation.
- Used **MP1584 buck converters**:
  - One to convert **7.4V battery to 5V** for ESP32.
  - Another to provide **3.3V to pumps**.
- Added **buzzer and 2 LEDs** for feedback (green = water flowing, red = access denied).
- Mounted components on **veroboard** and **PVC white board** for physical setup.
- Ensured proper wiring, power supply from **2x 3.7V batteries** in a **2S battery case**.

4.2 Sensor Calibration and Testing:

- Verified RFID readers individually by monitoring UID output via serial monitor.
- Tested card detection with two known UIDs assigned to separate taps.
- Calibrated voltage output from MP1584 converters using multimeter.
- Checked transistor switching with multimeter and dummy loads.
- Validated pump operation and LED/buzzer triggers for correct and incorrect access cases.

4.3 Software Development:

- Developed embedded C/C++ code in Arduino IDE:
  - Used **HardwareSerial** to manage 2 RFID readers.
  - Logic written to match each card UID to a specific water line.
  - Real-time billing calculated based on activation duration.
  - Data formatted and transmitted to **Google Sheets**.
- Error handling includes buzzer alert and red LED for mismatch.

4.4 App and User Interface Development

- Physical interface implemented using:
  - Green LED: Indicates authorized access and water flow.
  - Red LED + Buzzer: Indicates invalid access attempt.
- Google Sheets acts as a basic cloud dashboard for viewing usage logs and billing data.

4.5 Network Integration

- Configured Wi-Fi on ESP32 to enable cloud data logging.
- Used HTTPS POST requests to push data to Google Sheets through Apps Script Webhook.
- Tested network reliability and auto-reconnect logic on ESP32.
- Validated secure data transmission over Wi-Fi.

4.6 System Integration

- Combined hardware and software to form a complete dispensing system.
- Integration tests confirmed:
  - Each RFID card controls only one specific water line.
  - Mismatched usage triggers buzzer and denies access.
  - Billing and logging operate concurrently with water flow.
- End-to-end test: card scan → water dispense → usage logged to cloud.

4.7 Feedback Mechanisms

- **Visual Feedback:**
  - Green LED = water flow allowed.
  - Red LED = invalid access or no activity.
- **Audible Feedback:**
  - Buzzer sounds if wrong card is placed on wrong tap.
- **Cloud Feedback:**
  - Billing and usage details logged and viewable via Google Sheets.

4.8 Data Logging and Trend Analysis

- Every transaction (card UID, tap ID, usage duration, bill) is logged to **Google Sheets**.
- Each entry includes:
  - Timestamp
  - Card ID
  - Tap number
  - Bill amount

4.9 Deployment and Testing

- Deployed system on lab desk.
- Conducted real-world tests:
  - Verified correct access control and water dispensing.
  - Stress-tested by frequent card swaps and access errors.
- Monitored Google Sheets log for accuracy and consistency.

4.10 Documentation and Maintenance

- All circuit diagrams, pin mappings, and UID assignments are documented.
- Code is well-commented and version-controlled.

5 Gantt Chart

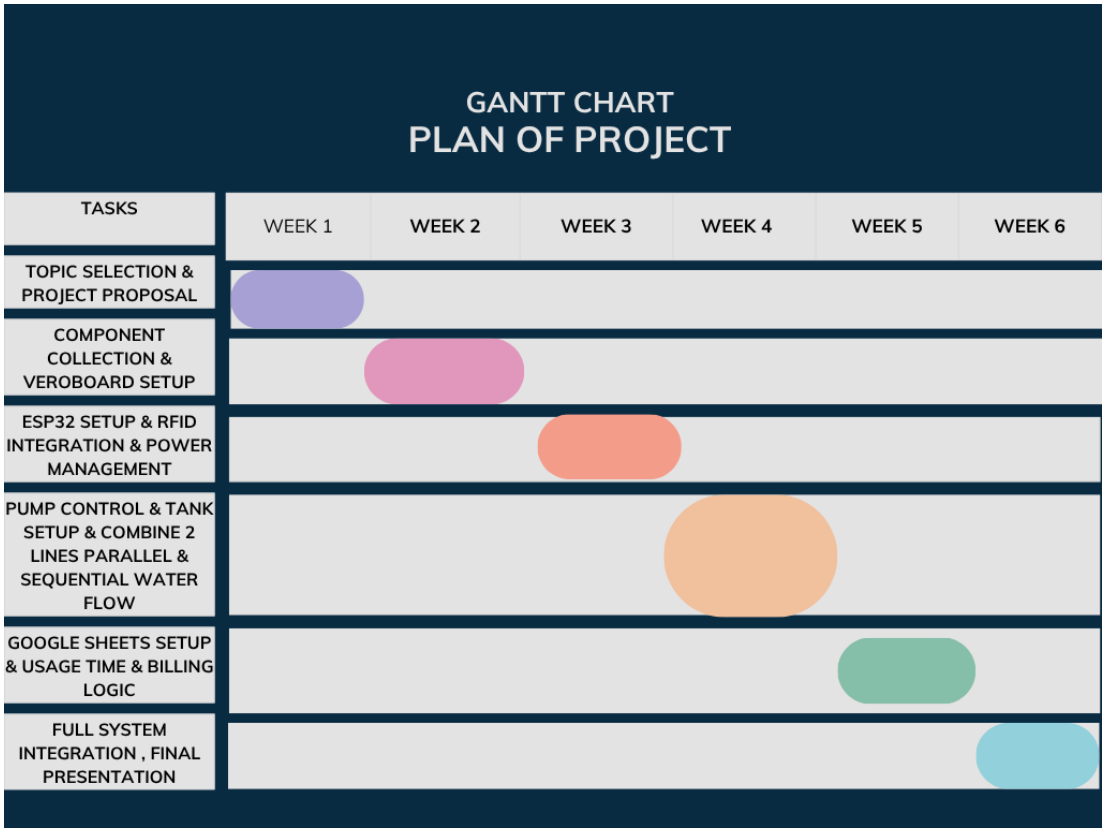

Figure 1: Gantt Chart

6 Workload Distribution Among Team Members

6.1 Individual Contribution using Table

Week	**Sadman Salman Saad (114)**	**Abdullah Al Maruf (116)**	**Sharun Tawsif (124)**	**Nafisa Tabassum (127)**
1. Topic Selection & Project Proposal	Literature Review, Initial Idea Draft	Architecture Sketch, Feature Planning	Documentation, Problem Definition	Use Case Scenario, Contribution Planning
2. Component Collection & Veroboard Setup	Purchase Components, Battery Holder	Schematic Design, Female Header Mounting	MP1584 Soldering, Jumper Wiring	Veroboard Layout, RFID Reader Pins Setup
3. ESP32 Setup, RFID & Power Management	ESP32 Firmware Burn, Serial Test	RFID Reader Communication (RDM6300)	Voltage Regulator Setup (MP1584)	Power Routing, Breadboard-to-Veroboard Shift
4. Pump Control, Tank Setup, Dual Line Flow	TIP122 Transistor Control Coding	Water Pump Activation, Line Switching Logic	Parallel vs Sequential Testing, Flow Stability	Tank Mounting, Pipe Layout and Water Flow Test
5. Google Sheets, Usage Time & Billing Logic	Apps Script/HTTP API to Google Sheets	Data Fields: Card ID, Time, Line Usage	Billing Algorithm & Timer-based Costing	Live Monitoring Dashboard, Billing Format
6. Full System Integration & Final Presentation	RFID + Pump + Billing End-to-End Demo	Final Hardware Testing and Debugging	Poster and Presentation Slide Design	Oral Presentation, Report Compilation

Table 3: Workload Distribution Among Team Members

6.2 Pie Chart Distribution

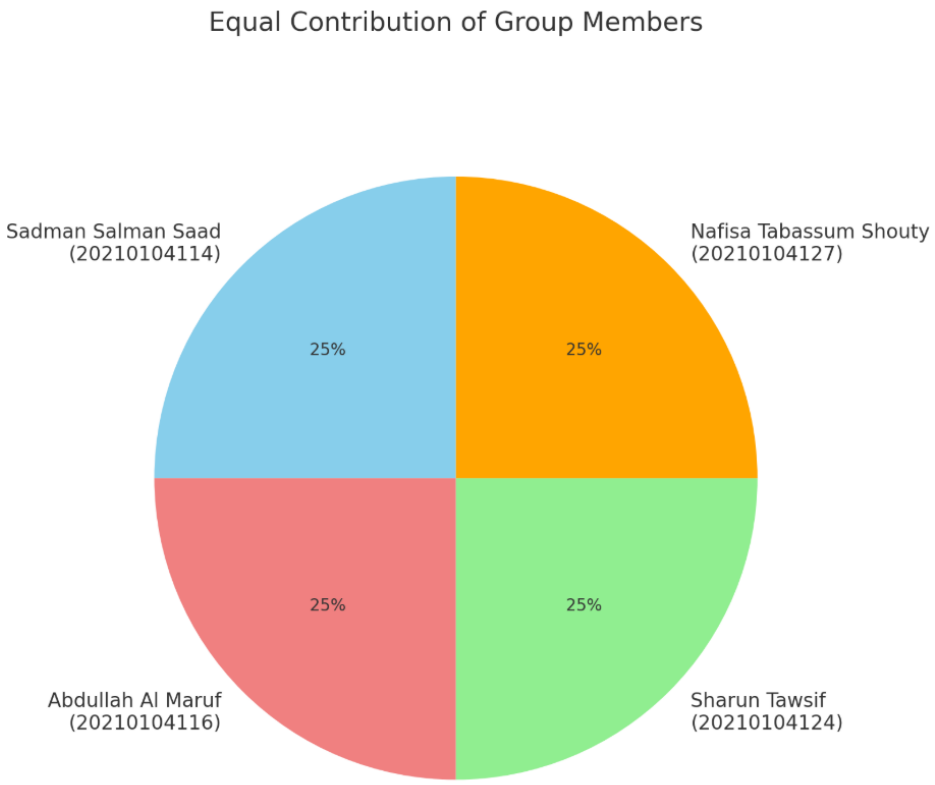

Figure 2: Equal Contribution Pie Chart Among Team Members