Smart Water Meter with Customer Management System

Water is one of the most critical resources on our planet, and efficient management of its consumption is essential to address growing demands and minimize waste. A **Smart Water Meter** with integrated **customer management** is an innovative solution that ensures accurate tracking and billing of water usage for multiple users sharing the same tap. This essay details the features, functionality, components, and benefits of the project.

Project Overview

The **Smart Water Meter** aims to measure water usage in real-time, detect leaks, provide alerts, and ensure equitable billing among customers sharing a communal tap. With the integration of **RFID technology**, each customer is assigned a unique RFID card, allowing the system to identify individual users and attribute their water consumption accordingly. This system not only promotes accountability and transparency but also makes water management more efficient, particularly in rural or communal setups.

Features and Workflow

1. **RFID-Based Customer Management**:
    - Each customer is issued an RFID card containing a unique ID.
    - Before accessing the shared tap, the customer taps their RFID card on the RFID reader. The system logs their ID and starts monitoring water usage specific to their account.
2. **Water Flow Measurement**:
    - A flow sensor captures the rate of water usage in real-time.
    - The microcontroller processes this data and links it to the respective RFID ID.
3. **Data Storage and Billing**:
    - Usage data is stored locally or on a cloud server for real-time monitoring and long-term tracking.
    - Individual bills are generated based on the logged consumption.
4. **Alerts and Notifications**:
    - A GSM module sends SMS notifications to customers for unusual usage patterns, such as leaks or exceeded limits.
    - Alerts are also displayed locally on an LCD for immediate feedback.
5. **Automated Water Flow Control**:
    - A solenoid valve controls water flow, ensuring access only for authorized customers based on RFID validation.

Key Components

1. **Hardware**:
    - **RFID Module**: An RFID reader (e.g., MFRC522) reads customer IDs from their cards.
    - **Flow Sensor**: Measures water usage with precision.
    - **Microcontroller**: Arduino, ESP32, or Raspberry Pi serves as the processing unit for sensor data and RFID validation.
    - **GSM Module**: Enables communication via SMS for usage alerts and customer notifications.
    - **Solenoid Valve**: Automatically regulates water flow based on RFID access.
    - **Display Unit**: LCD or OLED displays the current customer ID and real-time usage data.
    - **Power Supply**: Battery or mains power with backup capabilities for uninterrupted functionality.
2. **Software**:
    - RFID authentication logic.
    - Code for flow sensor data processing and real-time tracking.
    - Communication protocols for GSM and cloud integration.
    - A user interface for data visualization and billing.

Benefits of the System

1. **Accountability**:
    - Each customer's usage is tracked individually, preventing misuse and promoting fairness in billing.
2. **Efficiency**:
    - Real-time monitoring and automated control ensure optimal use of water resources.
3. **Cost Savings**:
    - Leak detection and usage alerts prevent unnecessary wastage, reducing water bills.
4. **Convenience**:
    - Customers receive detailed usage data via SMS and can access billing information remotely.
5. **Scalability**:
    - The system can be expanded to include more customers or integrate advanced features like prepaid systems or IoT connectivity.

Challenges and Solutions

1. **Shared Tap Usage**:
    - Challenge: Ensuring fair tracking in a shared setup.
    - Solution: RFID-based user identification and logging of individual consumption.

2. **Sensor Accuracy**:
    - Challenge: Ensuring precise measurements of water flow.
    - Solution: Regular calibration and use of high-quality sensors.
3. **Power Management**:
    - Challenge: Maintaining operation during power outages.
    - Solution: Integration of battery backups and low-power modes.
4. **System Security**:
    - Challenge: Preventing unauthorized access or tampering.
    - Solution: Secure RFID authentication and encrypted data transmission.

Future Enhancements

This system has tremendous potential for scalability and improvement. Future iterations could include:

- **IoT Integration**: Enable remote monitoring and control via mobile apps or web dashboards.
- **Prepaid Systems**: Allow customers to pay in advance based on desired usage limits.
- **Advanced Leak Detection**: Incorporate machine learning algorithms for anomaly detection.
- **Renewable Power Source**: Use solar panels for sustainable energy.

System Design for LCD-Based Interface & GSM Queries

1. Displaying User Info on LCD

Since the **LCD is the primary interface**, it should:

- Show the **current customer ID** (when RFID is scanned).
- Display **water consumption in real-time** (e.g., liters used).
- Indicate **available balance** (for prepaid usage).
- Provide **warnings** (e.g., "Low Balance," "Leak Detected").

Implementation

- Use a **16x2 or 20x4 LCD** (I2C interface recommended for easy wiring).
- Format **real-time updates** to refresh every few seconds.
- Integrate **buttons** (if needed) to navigate between display modes.

---

2. GSM-Based Balance Queries & Payment Updates

The GSM module will:

- **Allow users to check balance** by sending an SMS command.
- **Update balance** when payment is confirmed.
- **Send alerts** (low balance, leaks, excessive usage).

Implementation

- Use a **SIM800L GSM Module** connected to ESP32 via UART.
- Set up **commands** like:
  - **User sends:** `"BAL#1234"` (1234 = RFID ID)
  - **System replies:** `"User 1234, Balance: 10L"`
  - **Admin sends:** `"PAY#1234#50L"` (updates user balance)
  - **System confirms:** `"User 1234, 50L added"`
- **Automatic alerts**:
  - `"Low balance - Recharge required"`
  - `"Leak detected - Check system"`

---

3. RFID-Based Water Control

- When a user scans their **RFID card**, ESP32 verifies balance.

- If **valid**, solenoid valve opens, and water consumption starts logging.
- If **low balance**, LCD & GSM notify the user.

---

4. Data Storage

Since you're working on **ESP32 storage formats**, structured **JSON/CSV** logs will help store:

- **User ID**
- **Total liters used**
- **Remaining balance**
- **Last update timestamp**

HARDWARE LAYOUT

The structure is ensuring **CH341 programmer connections** don't interfere with normal operation for both **ESP32-WROOM-32** and **STM32G030F6P6**. I'll also include a **flashing guide** for smooth programming.

Updated Hardware Layout with CH341 Programmer Connections

This setup ensures each component remains functional without interfering during programming.

1. RFID-RC522 (SPI) → ESP32-WROOM-32 / STM32G030F6P6

RFID-RC522	ESP32-WROOM-32	STM32G030F6P6
SDA (SS)	GPIO 5	PA4
SCK	GPIO 18	PA5
MOSI	GPIO 23	PA7
MISO	GPIO 19	PA6
IRQ	(Not used)	(Not used)
GND	GND	GND
RST	GPIO 0	PA3
3.3V	3.3V	3.3V

🔻 Interference Consideration:

- **ESP32:** Ensure **SPI pins are free** when flashing.
- **STM32:** SPI pins **don't interfere with programming**, as SWD/UART are used.

---

2. Flow Sensor (SEN-HZ06D) → ESP32-WROOM-32 / STM32G030F6P6

Flow Sensor	ESP32-WROOM-32	STM32G030F6P6
VCC	5V	5V

Flow Sensor	ESP32-WROOM-32	STM32G030F6P6
GND	GND	GND
Signal	GPIO 14	PA0 (ADC)

🔻 Interference Consideration:

- **ESP32:** GPIO 14 is free during programming.
- **STM32:** PA0 is ADC input, independent of programming interface.

---

3. GSM Module (SIM900A) → ESP32-WROOM-32 / STM32G030F6P6

SIM900A	ESP32-WROOM-32	STM32G030F6P6
VBAT	4.2V	4.2V
GND	GND	GND
TXD	GPIO 16 (RX)	PA10 (USART RX)
RXD	GPIO 17 (TX)	PA9 (USART TX)
PWRKEY	GPIO 4	PA1

🔻 Interference Consideration:

- **ESP32:** GPIO 16 and 17 are used for GSM but must be **free during programming** (UART0 conflict).
- **STM32:** PA9 and PA10 are used for GSM but **don't interfere with SWD programming**.

---

4. Solenoid Valve (PWM Control) → ESP32-WROOM-32 / STM32G030F6P6

Solenoid Valve	ESP32-WROOM-32	STM32G030F6P6
VCC	3.3V or 4.2V	3.3V or 4.2V
GND	GND	GND

Solenoid Valve	ESP32-WROOM-32	STM32G030F6P6
Control	GPIO 27 (PWM)	PA8 (PWM)

🔻 Interference Consideration:

- **ESP32:** GPIO 27 is free during flashing.
- **STM32:** PA8 is PWM and doesn't interfere with SWD.

---

5. TFT LCD (ST7789, SPI) → ESP32-WROOM-32 / STM32G030F6P6

TFT LCD	ESP32-WROOM-32	STM32G030F6P6
GND	GND	GND
VCC	3.3V	3.3V
SCK	GPIO 18	PA5
SDA (MOSI)	GPIO 23	PA7
RES (Reset)	GPIO 4	PA3
DC (Data/Command)	GPIO 2	PA2
BLK (Backlight)	GPIO 15	PA1

🔻 Interference Consideration:

- **ESP32:** SPI pins must be **free when flashing** (disconnect display temporarily).
- **STM32:** SPI doesn't interfere with **SWD programming**.

---

6. LED Indicators

LED Type	ESP32-WROOM-32 GPIO	STM32G030F6P6 GPIO	Resistor
Power Indicator (Blue/White)	GPIO 13	PA9	220Ω – 470Ω
RFID Accepted (Green)	GPIO 12	PA8	220Ω – 470Ω

LED Type	ESP32-WROOM-32 GPIO	STM32G030F6P6 GPIO	Resistor
RFID Rejected (Red)	GPIO 14	PA7	220Ω – 470Ω

🔻 Interference Consideration:

- **ESP32:** LED pins remain usable without conflicts.
- **STM32:** PA7-PA9 don't interfere with flashing.

---

7. CH341 Programmer Connections (Avoiding Interference)

CH341 Programmer → ESP32-WROOM-32 (UART Boot Mode)

CH341 Pin	ESP32-WROOM-32 Pin	Notes
TXD (CH341)	GPIO3 (RX)	UART0 RX (for programming)
RXD (CH341)	GPIO1 (TX)	UART0 TX (for programming)
GND (CH341)	GND	Common ground
5V (CH341)	5V	Power if needed
DTR (CH341)	EN (ESP32 Reset)	Auto-reset feature
RTS (CH341)	IO0 (Boot Mode)	Flashing mode activation

🔻 Interference Prevention

- **Disconnect GSM module (GPIO16 & 17) during programming**, since it uses UART.
- **Ensure SPI devices (RFID, TFT) are disconnected during flashing**, to free GPIOs.

CH341 Programmer → STM32G030F6P6 (SWD Mode)

CH341 Pin	STM32G030F6P6 Pin	Notes
TXD (CH341)	PA9 (USART RX)	UART TX (for flashing)
RXD (CH341)	PA10 (USART TX)	UART RX (for flashing)

CH341 Pin	STM32G030F6P6 Pin	Notes
GND (CH341)	GND	Common ground
3.3V (CH341)	3.3V	Use if needed
SWCLK (Debugger)	PA14 (SWCLK)	SWD clock
SWDIO (Debugger)	PA13 (SWDIO)	SWD data
BOOT0	GND / 3.3V	Boot selection (GND = normal, 3.3V = bootloader mode)

🔻 Interference Prevention

- **Keep PA9 & PA10 free for flashing. If GSM is wired, disconnect it during programming.**
- **SWD pins (PA13, PA14) are dedicated, no interference with other functions.**

Flashing Guide for CH341 Programmer

1️⃣ Prepare ESP32 or STM32 Board

- For ESP32 → Hold **IO0 (Boot Mode)** LOW before flashing.
- For STM32 → Pull **BOOT0 to 3.3V** to enter bootloader mode.

2️⃣ Connect CH341 Programmer Properly

- Double-check UART or SWD connections as per tables above.
- Ensure **GSM, RFID, and TFT modules are disconnected** during flashing.

3️⃣ Use Flashing Software

- **ESP32:** Use **ESPTool or Arduino IDE** → Select proper COM port.
- **STM32:** Use **STM32CubeProgrammer or Flash Loader** → Select UART/SWD mode.

4️⃣ Flash the Firmware

- Upload code via CH341.
- When done, **reset the board** and return boot pins to normal mode.

This should keep everything **programming-friendly** while avoiding conflicts!