**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**

* + **RFID Module**: RFID-RC522
  + **Flow Sensor**: SEN-HZ06D.
  + **Microcontroller**: wroom32 cause its module got burned on the power control chip today so I guess the wroom32 on it might still be alive if not I intend on using STM32G030F6P6 or CH32V003J4M6
  + **GSM Module**: sim900A module
  + **Solenoid Valve**: a 3v-4.2v 3pin valve 350mA rated
  + **Display Unit**: a TFT 240x240 lcd 1.3’ black with st7789 controller
  + **Power Supply**: two 3.7v lion series connected and two highly precise buck boost converter one for 3.3v supply and another for 5.0v supply

This is a **complex but well-structured setup**, integrating **RFID, flow sensors, GSM, solenoid valves, and a TFT display** with **ESP32-WROOM-32 or STM32/CH32V**. Let's break down the **pinout and connections** for each component.

### ****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 needed) | (Not needed) |
| **GND** | GND | GND |
| **RST** | GPIO 0 | PA3 |
| **3.3V** | 3.3V | 3.3V |

🔹 **SPI Communication**: Both ESP32 and STM32 support **SPI**, so the wiring remains similar.

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

| **Flow Sensor** | **ESP32-WROOM-32** | **STM32G030F6P6** |
| --- | --- | --- |
| **VCC** | 5V | 5V |
| **GND** | GND | GND |
| **Signal** | GPIO 14 | PA0 (ADC) |

🔹 **Pulse Output**: The sensor outputs **pulses** based on water flow, which can be read using **interrupts or ADC**.

### ****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 |

🔹 **UART Communication**: SIM900A communicates via **UART**, so ensure **baud rate settings** match.

### ****4. Solenoid Valve (3V-4.2V, 350mA) → ESP32-WROOM-32 / STM32G030F6P6****

| **Solenoid Valve** | **ESP32-WROOM-32** | **STM32G030F6P6** |
| --- | --- | --- |
| **VCC** | 3.3V or 4.2V | 3.3V or 4.2V |
| **GND** | GND | GND |
| **Control** | GPIO 27 (PWM) | PA8 (PWM) |

🔹 **PWM Control**: Use **MOSFET or relay** to switch the solenoid valve safely.

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

| **TFT LCD (ST7789)** | **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 |

🔹 **SPI Display**: Ensure **proper initialization** in code.

### ****6. Power Supply Setup****

* **Two 3.7V Li-ion batteries in series** → **7.4V total**
* **Buck-Boost Converter**:
  + **3.3V output** → ESP32, STM32, RFID, TFT LCD
  + **5V output** → Flow sensor, GSM module

🔹 **Voltage Regulation**: Ensure **stable power** with capacitors (10µF + 100nF).

### ****7. Wiring for LED Indicators****

| **LED Indicator** | **ESP32-WROOM-32 GPIO** | **STM32G030F6P6 GPIO** | **Resistor** |
| --- | --- | --- | --- |
| **Power Indicator (Blue/White)** | GPIO 13 | PA9 | 220Ω – 470Ω |
| **RFID Accepted (Green)** | GPIO 12 | PA8 | 220Ω – 470Ω |
| **RFID Rejected (Red)** | GPIO 14 | PA7 | 220Ω – 470Ω |

### ****General Wiring Notes****

* **Resistor placement**: Connected **in series** between the **GPIO pin and the LED anode** (+).
* **LED Orientation**: Anode (+) to GPIO via resistor, Cathode (-) to **GND**.
* If brightness needs adjustment, tweak the resistor value between **220Ω** and **470Ω**.
* For **STM32**, configure GPIO pins in **Push-Pull output mode**.

**FIRMWARE DEVELOPMENT**