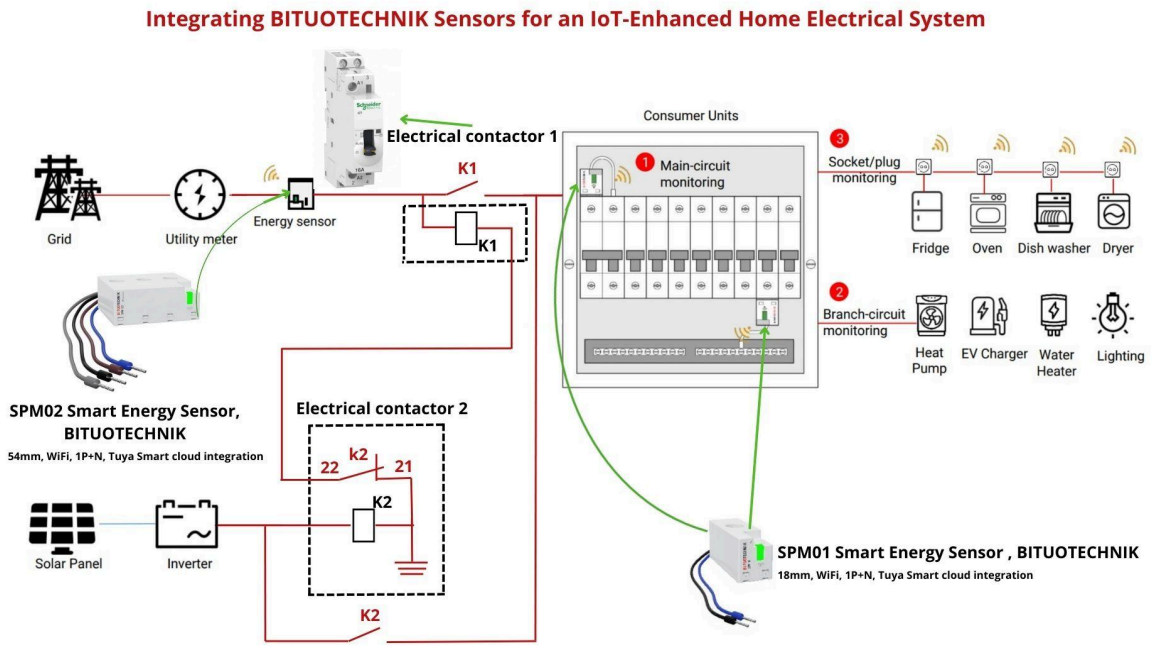
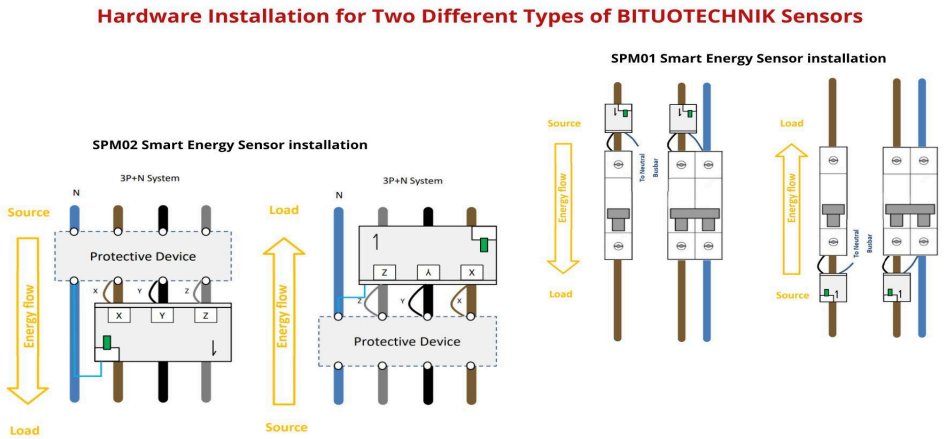
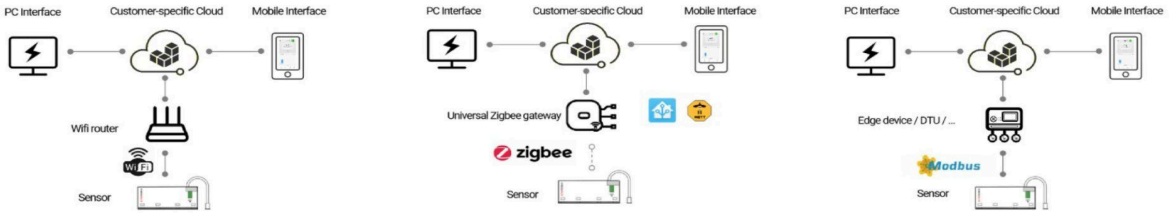
After conducting extensive research on the meters currently available in the market, we've identified several limitations. Many smart meters require complex installations that necessitate technical intervention, thereby increasing costs and limiting their market appeal. Additionally, these meters often face issues such as configuration difficulties and the risk of data congestion due to limited range and reliance on Wi-Fi network coverage, which can also lead to potential network congestion.

Here’s a detailed explanation of one of the most famous meters in the Chinese market. After trying to develop the Cabling process:





Various Wireless Communication Methods with BITUOTECHNIK Sensors



Wifi + Ble (With ESP-32 SoC) | Zigbee (With EFR32MG21 SoC) | Modbus RTU (with isolated RS485 IC)

→ **In our study was based on using sensors that work with Wi-Fi as the method of wireless communication.**

In light of these challenges, we've decided to use LoRaWAN technology instead of Wi-Fi. Here's an explanation of our smart meter:

Components Needed for the Smart IoT Meter

1. **Raspberry Pi(microprocessor) or Arduino (Microcontroller):**
   - **Role:** Acts as the main processing unit that collects data from the sensors, processes it, and sends it to the server via LoRaWAN.
2. **Current Sensor (e.g., ACS712):**
   - **Role:** Measures the current flowing through the electrical circuit.
3. **Voltage Sensor (e.g., ZMPT101B):**
   - **Role:** Measures the voltage in the electrical circuit.
4. **LoRaWAN Module (e.g., RAK811, Dragino LoRa/GPS HAT):**
   - **Role:** Provides LoRaWAN communication capabilities to the Raspberry Pi, allowing it to send data over long distances with low power consumption.
5. **Smart Meter Enclosure:**
   - **Role:** Protects the components and allows the meter to be easily attached to the electrical grid.
6. **Power Supply:**
   - **Role:** Powers the Raspberry Pi and other components.
7. **MicroSD Card:**
   - **Role:** Stores the Raspberry Pi's operating system and application software.
8. **Light dependent resistor**
   - **Role:** Is put to the microcontroller pins D1 and D2

9. **IoT Platform (e.g., The Things Network, AWS IoT Core):**
    - **Role:** Receives data from the LoRaWAN network and makes it accessible for further processing and storage.
10. **Web3 Application (Smart Contract on Ethereum):**
    - **Role:** Manages user rewards and interactions with the blockchain.

How the Smart IoT Meter Works in the Whole Process

Data Collection and Transmission

1. **Installation:**
    - The smart meter should be attached to the electricity meter in the house (using a double sided tape provided).It should be aligned with the light of the electricity meter in order to detect the status change.
    - The smart meter is connected to a web IDE, where the pins d1 is set to ground and d2 is set to pull-up resistor which will counter the changing resistance of the light sensor
2. **Data Acquisition:**
    - The current sensor measures the current flowing through the circuit, while the voltage sensor measures the voltage. These sensors are connected to the Raspberry Pi.
3. **Data Processing:**
    - The Raspberry Pi collects the raw data from the sensors. It processes this data to calculate the power consumption (Power = Voltage x Current).
4. **LoRaWAN Communication:**
    - The processed data is then sent to the LoRaWAN module, which transmits the data to a LoRaWAN gateway. LoRaWAN is chosen for its long-range and low-power communication capabilities, making it ideal for IoT applications.
5. **Data Reception:**
    - The LoRaWAN gateway forwards the data to an IoT platform (like The Things Network or AWS IoT Core), which acts as an intermediary between the IoT devices and the backend server.

Backend Processing

6. **Data Ingestion:**
    - The IoT platform receives the data and makes it accessible via APIs. The backend server fetches this data for further processing.
7. **AI Processing:**
    - An AI model running on the backend server processes the data to disaggregate the energy consumption per appliance. This model uses patterns in the data to identify which appliance is consuming how much energy.
    - **AI-based Disaggregation:**
        i. An AI model (such as a NILM algorithm) processes the stored data to identify and separate the power consumption of individual appliances.
        ii. The model uses patterns in the power usage data to recognize the unique signatures of different appliances. For example:

1. A refrigerator has a distinct on-off cycling pattern.
2. A washing machine has varying power levels corresponding to different wash cycles.

- **Algorithm Implementation:**
  i. Common NILM algorithms include:
    1. **Edge Detection:** Identifies changes in power consumption to detect when appliances turn on or off.
    2. **Pattern Recognition:** Uses machine learning to match patterns in the power consumption data to known appliance signatures.
    3. **Deep Learning Models:** More advanced models that can learn complex patterns and improve accuracy over time.

Step 4: Data Visualization and User Interaction

1. **Web3 Application:**
   - Users interact with a web application where they can view detailed reports on their energy consumption.
   - The application fetches the disaggregated data from the backend and displays it in an easy-to-understand format, showing the consumption of each appliance.
2. **Incentives and Rewards:**
   - Users receive tokens as rewards for reducing their energy consumption. The smart contract on the blockchain manages these rewards.
8. **Blockchain Interaction:**
   - The processed data, now containing detailed information about each appliance's energy consumption, is stored on a blockchain. A smart contract on Ethereum handles the storage and manages user interactions, such as issuing tokens for energy savings.
   - **Web Application:** Users interact with a web application where they can view detailed reports on their energy consumption. The app fetches data from the blockchain to provide real-time updates.
   - **Incentives and Rewards:** Users receive tokens as rewards for reducing their energy consumption. The smart contract on Ethereum manages these rewards, issuing tokens based on predefined criteria.
   - **Trading Tokens:** Users can buy, sell, or trade these tokens within the application. This incentivizes users to save energy and engage more with the platform.

Implementation Outline

Step 1: Setting Up Raspberry Pi

1. **Install Raspbian OS:**

- Download and install Raspbian on the MicroSD card.

Install Required Libraries:

```
sudo apt update
sudo apt install python3-pip
pip3 install numpy pandas scikit-learn
sudo apt install wiringpi
```

2.

Step 2: Connecting Sensors

1. **Current Sensor (ACS712):**
   - Connect the sensor to the Raspberry Pi's GPIO pins.
2. **Voltage Sensor (ZMPT101B):**
   - Connect the sensor to the Raspberry Pi's GPIO pins.

Step 3: LoRaWAN Setup

1. **Connect the LoRaWAN Module:**
   - Attach the LoRaWAN module to the Raspberry Pi.

Install LoRaWAN Libraries:

```
sudo apt install git
git clone https://github.com/dragino/rpi-lora-tranceiver
cd rpi-lora-tranceiver
sudo ./install.sh
```

2.

Step 4: Writing the Firmware

1. **Data Collection Script:**
   - Write a Python script to collect data from the sensors and send it via LoRaWAN.

```
import wiringpi as wp
import time
from lora import LoRa

lora = LoRa()

def read_current_sensor():
    # Code to read current sensor
    return current_value
```

```
def read_voltage_sensor():
    # Code to read voltage sensor
    return voltage_value

while True:
    current = read_current_sensor()
    voltage = read_voltage_sensor()
    power = current * voltage

    data = f"{current},{voltage},{power}"
    lora.send(data)

    time.sleep(60)
```

2. **Setup Example with Puck.Js opensource:**

```
var counter = 0;

// Update BLE advertising
function update() {
  var a = new ArrayBuffer(4);
  var d = new DataView(a);
  d.setUint32(0, counter, false/*big endian*/);
  NRF.setAdvertising({},{
    name: "Puck.js \xE2\x9A\xA1",
    manufacturer:0x0590,
    manufacturerData: a,
    interval: 600 // default is 375 - save a bit of power
  });
}

// Set up pin modes
D1.write(0);
pinMode(D2,"input_pullup");

// Watch pin changes
setWatch(function(e) {
  counter++;
  update();
  digitalPulse(LED,1,50);
}, D2, { repeat:true, edge:"falling" });
```

Step 5: Backend and Blockchain Integration

1. **Develop Backend API:**
   - Create an Express.js server to handle incoming data from the IoT platform.
2. **Smart Contract Deployment:**
   - Deploy the smart contract to the Ethereum network.
3. **Web Application Development:**
   - Develop the frontend using React to display energy consumption data and manage token rewards.

Conclusion

By following these steps, we can create a smart IoT meter that uses LoRaWAN for communication, collects detailed energy consumption data, and integrates with a blockchain-based reward system. This solution is designed to be easy to install and user-friendly, providing users with actionable insights into their energy usage and incentivizing them to save energy.