Solar Monitoring System with ESP32

1. System Overview

A low-cost ESP32-based solution to monitor solar production, grid power, and home consumption, with data visualized in a Flutter app.

Key Components

- **ESP32**: Data collection and wireless transmission.
- **PZEM-004T**: Measures voltage/current from grid or loads.
- **CT Clamps (SCT-013)**: Track consumption.
- **Solar Inverter (Growatt/Solis)**: Provides PV production data via Modbus.
- **Flutter App**: Real-time dashboards for iOS/Android/Web.

---

2. System Flowchart

(Data Flow & Interactions)

```
Solar Panels → Inverter → ESP32 (Modbus) → MQTT Broker → Flutter App
                      ↑
Grid/Consumption → PZEM-004T/CT Clamps
```

Steps Explained:

1. **Inverter (Modbus RTU)**: ESP32 queries solar production data.
2. **PZEM-004T**: Measures grid voltage/current via UART.
3. **CT Clamps**: Analog readings for load monitoring.
4. **ESP32**: Combines data → Publishes to MQTT (e.g., `solar/data`).
5. **Flutter App**: Subscribes to MQTT → Updates UI in real-time.

---

3. Wiring Diagram

(ESP32 Connections)

```
ESP32 Pinout:
- GPIO16 (RX) → MAX485 (A) → Inverter RS485-A
- GPIO17 (TX) → MAX485 (B) → Inverter RS485-B
- GPIO2 (RX2) → PZEM-004T TX
- GPIO4 (TX2) → PZEM-004T RX
- GPIO34 (ADC1) → CT Clamp Output
- 3.3V/GND → Power sensors
```

Note: Use a `MAX485` module for RS485 communication with the inverter.

---

4. ESP32 Firmware Logic

(Simplified Pseudocode)

```
void setup() {
  Initialize Serial (UART for PZEM);
  Initialize Modbus (RS485 for Inverter);
  Connect WiFi/MQTT;
}

void loop() {
  solar_power = read_inverter(register=32080);  // Growatt example
  grid_power = pzem.readPower();
  consumption = ct_clamp.readAnalog();
  mqtt.publish("solar/data", format(solar_power, grid_power, consumption));
  deep_sleep(10_seconds);
}
```

---

5. Flutter App Architecture

(Data Visualization)

```
1. MQTT Subscription:
   - Topic: `solar/data` (Payload: "1200, -300, 900" = PV, Grid, Load).
2. State Management:
   - Use `Provider` or `Riverpod` to update UI.
3. Charts:
   - `charts_flutter` for time-series graphs.
4. Alerts:
   - Notify if PV production drops suddenly.
```

---

6. Error Handling Flow

```
ESP32 Boot → Connect WiFi → Read Sensors → Publish MQTT
    ↑________Retry (x3) ← Fail ←_________|
        ↑
    Reboot ESP32
```

---

7. Bill of Materials (BOM)

Part	Quantity	Unit Price	Total
ESP32 Dev Board	1	$8	$8
PZEM-004T	1	$15	$15
CT Clamp (SCT-013)	2	$12	$24
MAX485 Module	1	$2	$2
Total			**$49**

---

8. Next Steps

1. **Hardware Prototyping**:
   - Wire ESP32 to PZEM/CT clamps.
   - Test Modbus registers with your inverter.
2. **Software Testing**:

    - Verify MQTT data flow using `MQTT Explorer`.
3. **App Development**:
    - Start with a basic Flutter subscription UI.

Attachments

1. ESP32 Pinout Reference
2. Modbus Register Map for Growatt