1. Signalverarbeitung über GPIO15

Ziel:
Erfassung von Signalflanken (steigend/fallend) und Berechnung der Zeitdifferenz zwischen fallenden Flanken zur Bestimmung des momentanen Verbrauchs.

Interrupt-Routine: `Signal_INT()`

- Wird bei jeder Signaländerung (steigende/fallende Flanke) ausgelöst:
  - **Steigende Flanke:**
    - Speichert den Zeitpunkt (`tck1`).
    - Aktiviert die onboard-LED.
  - **Fallende Flanke:**
    - Speichert den Zeitpunkt (`tck2`).
    - Berechnet die Periode:

      ```
      periode = tck2 - tck2_v
      ```

    - Aktualisiert den Zähler `icounter`.

Flankendetektion:

- **GPIO15** als Eingang mit Pull-Down konfiguriert:

```
Signal = machine.Pin(15, machine.Pin.IN, machine.Pin.PULL_DOWN)
Signal.irq(trigger=machine.Pin.IRQ_RISING | machine.Pin.IRQ_FALLING,
handler=Signal_INT)
```

---

2. Leistungsberechnung

Formel:

Die Leistung wird aus der Dauer der Periode berechnet:

Leistung (W) = 3600000 / periode

Anwendung:

- Berechnete Werte werden auf dem LCD angezeigt:

```
disp_lcd(str(counter), str(mverbrauch))
```

- Anschließend wird der Wert an einen MQTT-Broker gesendet.

---

3. WLAN-Verbindung

Die Funktion **`wlanConnect()`** stellt eine Verbindung zu einem WLAN her, indem SSID und Passwort aus einer privaten Datei geladen werden.

Details:

- **Verbindungsaufbau:**

  ```
  wlan.connect(wlanSSID, wlanPW)
  ```

  - LEDs (Gelb/Onboard) zeigen den Verbindungsstatus an.
  - Bei Erfolg: Anzeige der IPv4-Adresse.
  - Bei Misserfolg: Neustart des Prozesses.

---

4. MQTT-Integration

Verbindung herstellen: `mqttConnect()`

Stellt eine Verbindung zu einem MQTT-Broker mit Authentifizierung her:

```
client = MQTTClient(mqttClient, mqttBroker, user=mqttUser, password=mqttPW,
keepalive=60)
client.connect()
```

Daten senden:
Die berechnete Leistung wird an das Topic `mpower` gesendet:

```
client.publish(mqttTopic0, myValue0)
```

- Nach dem Senden wird die Verbindung getrennt.
- Bei Fehlern wird der Mikrocontroller zurückgesetzt.

---

5. LEDs und LCD-Anzeige

LEDs:

- **Rot/Gelb:** Blinken zur Anzeige der Hauptaktivität.
- **Grün:** Zeigt erfolgreiche Verbrauchsberechnungen.

LCD:

- Ausgabe der Berechnungen auf einem 16x2-LCD über I2C:

```
disp_lcd(z1, z2)
```

---

6. Hauptschleife

- **Funktionen:**
  - Berechnung der Leistung, Anzeige auf dem LCD und Übertragung via MQTT.
  - Blinken der LEDs während des Wartens.

- **Neustart:** Nach 10.000 Interrupts oder bei Fehlern wird der Mikrocontroller zurückgesetzt.

Zusammenfassung

Der Code bietet:

1. **Signalverarbeitung**: Zeitmessung und Verbrauchsberechnung basierend auf GPIO-Signalen.
2. **Netzwerkfähigkeit**: Verbindung zu einem WLAN und einem MQTT-Broker.
3. **IoT-Funktionalität**: Übertragung der Verbrauchsdaten an einen entfernten Server.

Ideal für IoT-Projekte wie **Energieüberwachung** oder **Verbrauchsmessung**.