Proposta Detalhada – Sistema de Monitoramento de Energia Residencial com Raspberry Pi Pico W

1) Resumo Executivo

Desenvolver um sistema IoT para medição e análise do consumo elétrico residencial, com coleta local via Raspberry Pi Pico W, cálculo de métricas elétricas (Vrms, Irms, potência ativa/reativa/aparente, fator de potência), publicação segura via MQTT/TLS para nuvem (HiveMQ ou AWS IoT) e visualização em dashboard (Node-RED + InfluxDB + Grafana ou Home Assistant). O projeto inclui detecção de eventos de cargas (liga/desliga) e alertas de consumo anômalo.

---

2) Objetivos

- **Mensurar** tensão, corrente e potência por circuito/ramal ou por tomada.
- **Calcular** métricas: Vrms, Irms, P (ativa), Q (reativa), S (aparente), PF (fator de potência), energia (Wh/kWh).
- **Publicar** dados com segurança (MQTT/TLS) e **armazenar** histórico.
- **Detectar eventos** de mudança de carga (on/off) e disparar **alertas**.
- **Exibir** dados em tempo real e históricos em dashboard responsivo.
- **Fornecer** base para expansão (múltiplos canais, OTA, integração Home Assistant).

---

3) Arquitetura de Alto Nível

- **Sensoriamento (AC 127/220 V, 50/60 Hz)**:
    - **Tensão:** transformador de acoplamento (ex.: ZMPT101B) para isolamento + rede RC de anti-aliasing.
    - **Corrente:** transformador de corrente não invasivo **SCT-013-000** (100 A:50 mA) + resistor de carga (burden) + condicionamento.
    - **Alternativa integrada:** CI de medição (HLW8012, ATM90E26/ATM90E32) para tomada inteligente ou quadro elétrico.
- **Aquisição:** ADC (preferência por **ADS1115** 16-bit via I2C). Alternativa: ADC interno do RP2040 (12-bit) com DMA.
- **Processamento:** Raspberry Pi Pico W (RP2040) calcula métricas e energia acumulada.
- **Conectividade:** Wi-Fi 2.4 GHz + **MQTT sobre TLS** (porta 8883) → **HiveMQ** ou **AWS IoT**.

- **Armazenamento local:** Flash (LittleFS) para buffers/estatísticas e retentiva (ex.: energia acumulada e calibração).
- **Visualização:** Node-RED (flows) → InfluxDB (time-series) → Grafana (dashboards) **ou** Home Assistant (via MQTT Discovery).

---

4) Hardware – Opções e Dimensionamento

4.1 Opção A – Não Invasiva (Quadro Elétrico)

Uso indicado: Medir circuito geral ou ramais (chuveiro, ar-condicionado, tomadas). - **Tensão (ZMPT101B):** - Alimentação: 5 V/3.3 V conforme módulo. - Saída: sinal AC reduzido e isolado → offset para 1.65 V (meia escala do ADC 3.3 V). - Filtro: RC (ex.: R=1 kΩ, C=100 nF) como anti-aliasing. - **Corrente (SCT-013-000):** - Burden típico: 62 Ω–100 Ω (ajustar para amplitude segura ao ADC). - Retificação NÃO desejada (usar sinal AC puro). - Offset DC (1.65 V) + anti-aliasing. - **ADC: ADS1115** (±4.096 V FS, 16-bit, 860 SPS máx. por canal) – precisão superior. Para amostragem mais alta, considerar **ADS1015** (12-bit, 3.3 kSPS) ou ADC interno com DMA.

4.2 Opção B – Tomada Inteligente (Módulo Integrado)

Uso indicado: Medição por tomada específica/eletrodoméstico. - **Medição:** CI **HLW8012** (usado em smart plugs) ou **ATM90E26/32** – entrega P, Vrms, Irms, PF via SPI/serial. - **Simplicidade:** Menos processamento no Pico W; foco em integração IoT e recursos de software.

4.3 Considerações de Segurança

- **Isolamento galvânico obrigatório** entre rede e microcontrolador.
- Em **nenhum** momento expor partes vivas; usar **caixa plástica** com prensa-cabos e trilhas de escoamento.
- **Fusíveis** e **PTC** conforme corrente do circuito monitorado.
- Aterramento, **proteção contra surtos (MOV/TVS)** se necessário.
- Trabalhar **sempre com rede desenergizada**; validação por profissional habilitado.

4.4 Lista de Materiais (BOM) – Opção A (exemplo 1 canal)

- 1× Raspberry Pi Pico W
- 1× Módulo **ADS1115** (I2C)
- 1× **SCT-013-000**
- 1× **ZMPT101B**
- Resistores p/ burden e divisores (precisão 1% ou melhor)
- Capacitores p/ filtros RC
- 1× **OLED SSD1306** I2C 0.96” (opcional)

- 1× Botão + buzzer (opcional)
- 1× Fonte 5 V isolada (ex.: 5 V/1 A)
- PCB/perfboard, conectores, caixa ABS, prensa-cabos, fusível

---

5) Firmware – Arquitetura

5.1 Pilha de Software

- **SDK:** pico-sdk (C/C++)
- **RTOS (opcional):** FreeRTOS para tarefas determinísticas.
- **Drivers:** I2C (ADS1115, SSD1306), Wi-Fi, MQTT (paho.mqtt.embedded-C), TLS (mbedTLS), GPIO, Timer/DMA (se ADC interno), LittleFS.

5.2 Módulos

- **sensor_voltage.c / sensor_current.c** – leitura bruta + calibração + offset.
- **power_calc.c** – RMS, potência instantânea, janelas, energia (Wh).
- **events.c** – detecção de passos (on/off), histerese, classificação simples.
- **net_mqtt.c** – conexão segura, reconexão exponencial, QoS, LWT.
- **storage.c** – calibração e energia acumulada persistente.
- **ui_oled.c** – status, leituras, RSSI, IP.
- **main.c** – orquestração (estado: BOOT→NET→MEASURE→PUBLISH→IDLE).

5.3 Taxas de Amostragem e Processamento

- **60 Hz (Brasil 60 Hz):** alvo ≥ 2 kHz por canal (≥ 33 amostras/ciclo).
- Janela de **1 s** para RMS/energia por segundo; agregações em 10 s e 60 s para publicar.
- Filtro digital: **HPF** para remover DC de offset (IIR 1ª ordem) e **LPF** leve p/ ruído.

5.4 Cálculos (janelamento N amostras)

- Offset removido: $x\_ac[n] = x[n] - mean(x)$.
- **Vrms** = sqrt((1/N)·Σ v_ac[n]^2), **Irms** = sqrt((1/N)·Σ i_ac[n]^2).
- **Potência instantânea**: p[n] = v_ac[n] · i_ac[n].
- **Potência ativa**: P = (1/N)·Σ p[n].
- **Potência aparente**: S = Vrms·Irms.
- **Fator de potência**: PF = P/S (limitar ±1.0).
- **Energia**: E_Wh += P(W)·Δt(h). Para kWh dividir por 1000.
- **Atenção à defasagem**: ajustar via calibração ou atraso fracionário (FIR) p/ alinhar V e I.

5.5 Detecção de Eventos de Carga (NILM básico)

- **On/Off por degrau**: detectar |ΔP| > limiar (ex.: 30–80 W) com histerese e janela curta (200–500 ms).
- **Clusters de assinaturas**: manter tabela simples (ΔP médio) p/ identificar cargas comuns (geladeira, micro-ondas, lâmpada LED).
- **Alertas**: consumo anômalo quando P média sobe > X% versus baseline do período.

5.6 Robustez

- **Watchdog**, reconexão MQTT exponencial, **TLS** com verificação de certificado, **NTP** para timestamp.
- **Buffer offline**: armazenar até N publicações; reenvio quando on-line.

---

6) Protocolo e Payload MQTT

- **Broker:** HiveMQ (teste) ou **AWS IoT** (produção), porta 8883 com TLS.
- **Tópicos (exemplo):**
  - `casa/energia/medidas/{canal}` – publicações periódicas
  - `casa/energia/eventos` – on/off e alertas
  - `casa/energia/cmd` – comandos (ex.: alterar intervalo, reset energia)
  - `casa/energia/status` – LWT/online
- **QoS:** 1 para medidas, 0 para status.
- **Payload JSON (medidas):**

```
{
  "ts": 1734638400,
  "canal": 1,
  "vrms": 126.8,
  "irms": 3.42,
  "p": 387.5,
  "q": 120.3,
  "s": 433.0,
  "pf": 0.90,
  "freq": 60.02,
  "kwh": 12.457,
  "rssi": -62
}
```

- **Eventos:**

```
{ "ts": 1734638420, "tipo": "on", "delta_w": 85.0 }
```

- **Home Assistant (opcional):** suporte a MQTT Discovery com entidades `sensor` (P, kWh) e `binary_sensor` (evento de carga).

---

7) Dashboard e Backend

7.1 Stack 1 – Node-RED + InfluxDB + Grafana

- **Node-RED**: recebe MQTT, normaliza JSON e grava em **InfluxDB**.
- **Grafana**: painéis com gráficos de P, kWh diário/mensal, PF, ranking de cargas por evento.
- **Alertas**: Node-RED → Telegram/Email em thresholds.

7.2 Stack 2 – Home Assistant

- Descoberta via MQTT; cartões de energia; automações (ex.: notificar quando PF < 0.7 por 5 min).

---

8) Calibração e Testes

8.1 Calibração

- **Ganho de tensão**: comparar Vrms medido com multímetro true-RMS → ajustar constante.
- **Ganho de corrente**: usar carga resistiva conhecida (ex.: 1000 W) → ajustar Irms.
- **Fase V↔I**: medir com carga indutiva (motor/ventilador) → minimizar erro de P ajustando atraso.

8.2 Testes

- **Linearidade** (100–2000 W), **ruído** (sem carga), **temperatura** (0–50 °C), **queda de Wi-Fi** (reconexão), **persistência** (reinício mantém kWh), **estresse** (publicações por 72 h).

---

9) Plano de Implementação (6 a 8 semanas)

1. **Semana 1** – Especificação final, compra de componentes, desenho de esquemas, riscos.
2. **Semana 2** – Montagem protótipo (bancada), drivers I2C (ADS1115/SSD1306), leitura bruta.
3. **Semana 3** – Cálculo de RMS e potência, calibração inicial, OLED.
4. **Semana 4** – MQTT/TLS, reconexão, buffer offline, timestamps.
5. **Semana 5** – Dashboard (Node-RED/Influx/Grafana) **ou** Home Assistant, painéis.
6. **Semana 6** – Detecção de eventos, alertas, tunning de thresholds.
7. **Semana 7** – Caixa/enclosure, segurança, testes prolongados (72 h).
8. **Semana 8** – Documentação final, vídeo demo, relatório técnico e poster.

10) Entregáveis

- **Hardware** montado e enclausurado com conectores seguros.
- **Código-fonte** (Git) com README, instruções de build e calibração.
- **Dashboards** prontos (JSON de export do Grafana/flows do Node-RED ou YAML do Home Assistant).
- **Relatório técnico** (metodologia, esquemas, testes, resultados e custos).
- **Vídeo demo** (5–8 min).

---

11) Critérios de Sucesso

- Erro de **Vrms/Irms ≤ 2–3%** após calibração.
- Erro de **potência ativa ≤ 5%** em cargas resistivas.
- **Uptime ≥ 99%** em teste de 72 h.
- Reconexão automática Wi-Fi/MQTT em < 10 s.
- Dashboard com histórico de **≥ 7 dias** e export de dados.

---

12) Riscos e Mitigações

- **Risco:** ruído/aliasing → **Mitigação:** filtros RC e digital, amostragem ≥ 2 kHz.
- **Risco:** erro por defasagem V/I → **Mitigação:** calibração de fase, FIR fracionário.
- **Risco:** Wi-Fi instável → **Mitigação:** backoff exponencial, buffer offline, watchdog.
- **Risco:** segurança elétrica → **Mitigação:** isolamento, fusíveis, caixa fechada, testes com técnico.

---

13) Extensões Futuras

- **Multicanais** (3–6 ramos) com multiplexação e timestamps sincronizados (NTP/PTP).
- **OTA** (firmware update) via HTTP/MQTT.
- **Classificação de cargas avançada** (NILM com features de harmônicos/FFT).
- **Integração AWS IoT** (Rules → Timestream → QuickSight) e IAM fine-grained.
- **Medição trifásica** (3× ZMPT + 3× CT) com cálculo de desequilíbrio.

---

14) Custos Estimados (ex. 1 canal)

- Pico W: R$ 60–90

- ADS1115: R$ 30–45
- SCT-013-000: R$ 60–80
- ZMPT101B: R$ 20–30
- OLED: R$ 25–40
- Caixa, conectores, PCB, fonte, passivos: R$ 60–120 **Total aproximado:** R$ 255–405 (dependendo do fornecedor)

---

15) Estrutura do Repositório (sugestão)

```
energy-monitor-pico-w/
├─ firmware/
│  ├─ CMakeLists.txt
│  ├─ src/
│  │  ├─ main.c
│  │  ├─ power_calc.c
│  │  ├─ sensor_voltage.c
│  │  ├─ sensor_current.c
│  │  ├─ net_mqtt.c
│  │  ├─ storage.c
│  │  ├─ ui_oled.c
│  │  └─ events.c
│  └─ include/
│     ├─ power_calc.h
│     ├─ sensors.h
│     ├─ net_mqtt.h
│     ├─ storage.h
│     ├─ ui_oled.h
│     └─ events.h
├─ hardware/
│  ├─ schematics.pdf
│  ├─ pcb/
│  └─ enclosure/
├─ dashboards/
│  ├─ node-red-flows.json
│  ├─ grafana-dashboard.json
│  └─ home-assistant/
├─ docs/
│  ├─ README.md
│  ├─ CALIBRATION.md
│  ├─ SAFETY.md
│  └─ TESTS.md
└─ media/
   ├─ photos/
   └─ demo.mp4
```

---

16) Pseudocódigo Essencial (Aquisição→Cálculo→Publicação)

```
init_hw(); // I2C, Wi-Fi, MQTT TLS, ADC/ADS1115, OLED, timers
load_calibration();
start_sampling(); // DMA/loop periódico em 2 kHz

for (;;) {
  if (window_ready()) {
    remove_dc_offset(v_buf, i_buf);
    compute_rms(&Vrms, &Irms);
    align_phase_if_needed(v_buf, i_buf);
    P = mean(mul(v_buf, i_buf));
    S = Vrms * Irms; PF = clamp(P / S, -1, 1);
    E_Wh += P * window_seconds / 3600.0;

    if (detect_event(P, &evt)) publish_event(evt);

    if (time_to_publish()) {
      mqtt_publish_json(Vrms, Irms, P, S, PF, E_Wh, rssi, ts);
    }
  }
  mqtt_yield();
  feed_watchdog();
}
```

---

17) Conclusão

Este projeto equilibra **profundidade técnica** (DSP simples, protocolos seguros, integração IoT) com **aplicação prática** (monitorar e reduzir consumo). A modularidade permite desde um protótipo de bancada até uma solução escalável para múltiplos circuitos e integração com plataformas modernas (Home Assistant/AWS IoT).