UNIVERSIDADE FEDERAL DO PAMPA

GERSON EVANDRO DE OLIVEIRA SENA

MEDIDOR DE CONSUMO DE ENERGIA ELÉTRICA COM ACESSO LOCAL E REMOTO USANDO PLATAFORMA ESP8266

Alegrete
2018

GERSON EVANDRO DE OLIVEIRA SENA

MEDIDOR DE CONSUMO DE ENERGIA ELÉTRICA COM ACESSO LOCAL E REMOTO USANDO PLATAFORMA ESP8266

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Elétrica da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia Elétrica.

Orientador: Jumar Luís Russi

Alegrete
2018

Ficha catalográfica elaborada automaticamente com os dados fornecidos
pelo(a) autor(a) através do Módulo de Biblioteca do
Sistema GURI (Gestão Unificada de Recursos Institucionais) .

S474m Sena, Gerson Evandro de Oliveira
MEDIDOR DE CONSUMO DE ENERGIA ELÉTRICA COM ACESSO LOCAL E REMOTO USANDO PLATAFORMA ESP8266 / Gerson Evandro de Oliveira Sena.
113 p.

Trabalho de Conclusão de Curso(Graduação)-- Universidade Federal do Pampa, ENGENHARIA ELÉTRICA, 2018.
"Orientação: Jumar Luis Russi".

1. Medidor de energia. 2. ESP8266. 3. Datalogger. 4. Protocolo MQTT. I. Título.

GERSON EVANDRO DE OLIVEIRA SENA

MEDIDOR DE CONSUMO DE ENERGIA ELÉTRICA COM ACESSO LOCAL E REMOTO USANDO PLATAFORMA ESP8266

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Elétrica da Universidade Federal do Pampa, como requisito parcial para obtenção do título de Bacharel em Engenharia Elétrica.

Área de Concentração: Eletrônica de Potência

Trabalho de Conclusão de Curso defendido e aprovado em: 26 de junho de 2018.

Banca examinadora:

Prof. Dr. Jumar Luís Russi
Orientador

Prof. Me. Lucas Compassi Severo
UNIPAMPA

Prof. Dr. Alessandro Botti Benevides
UNIPAMPA

A minha Suelen e a nossa pequena Líria, por darem propósito a esta conquista. E aos meus pais, Vera e Ernande, pela minha existência e por, de alguma forma, eu ter tomado as decisões que tomei até aqui.

AGRADECIMENTO

Ao Prof. Dr. Jumar Luís Russi, pela paciência e apoio na orientação deste trabalho e em outros momentos.

Aos professores que souberam ser condescendentes, justos e profissionais enquanto ministravam aulas e avaliações em toda minha trajetória acadêmica. Em especial aqueles que demonstravam prazer ao ensinar, que contagiaram com seu conhecimento e sua boa vontade para com seus alunos.

A todos os colegas de curso e percurso que ajudaram a entender conceitos, resolver problemas, entregar trabalhos no prazo, concluir projetos, que animaram, encorajaram, motivaram a persistir, foram amigos, ou seja, que estenderam a mão e me ajudaram a cumprir mais essa missão. E cujos nomes não caberiam numa única página.

“A grandeza não é onde permanecemos, mas em qual direção estamos nos movendo. Devemos navegar algumas vezes com o vento e outras vezes contra ele, mas devemos navegar, e não ficar à deriva, e nem ancorados”.

Oliver Wendell Holmes Sr.

RESUMO

O consumo de energia elétrica e o controle deste consumo é um tema muito importante no contexto atual brasileiro, já que os valores cobrados e taxados sobre esta energia são alguns dos mais altos a nível global, onde um terço do consumo consta de consumidores residenciais. O cenário se torna ainda mais crítico ao se notar a ausência e o difícil (ou inexistente) acesso a mecanismos de auditagem e controle do consumo pelo cliente, o que colabora negativamente para criação de uma cultura de economia e bom uso de energia elétrica. Como possibilidade de resposta a estas questões, este trabalho busca propor a arquitetura de um protótipo de medidor de energia elétrica. A iniciativa visa pesquisar, avaliar e propor um sistema de baixo custo, acessível e viável dos pontos de vista econômicos e práticos para clientes residenciais. Com isso espera-se motivar a pesquisa e desenvolvimento de medidores que não demandem complexidade nem custo acima da realidade de consumo de clientes residenciais. O sistema a ser apresentado consta de um medidor de tensão e corrente elétricas senoidais (corrente alternada), cujo núcleo é um módulo de desenvolvimento NodeMCU Lolin (equipado com um microcontrolador ESP8266-12E), e aproveitando os recursos nativos à plataforma Arduino. As funcionalidades do protótipo são as medições já citadas, o cálculo da potência ativa em tempo real, o armazenamento do consumo de energia, o registro de todos estes dados em uma memória temporal recuperável (datalogger) e também transmissão destes dados com um protocolo Message Queue Telemetry Transport (MQTT) para ser recuperado por aplicativos WEB ou de telefones celulares, com conexão via Internet.

Palavras-Chave: Medidor de Energia, ESP8266, Datalogger, protocolo MQTT.

ABSTRACT

The consumption of electric power and the control of this consumption is a very important issue in the current Brazilian context, since the values charged and taxed on this energy are some of the highest in the global level, where a third of the consumption consists of residential consumers. The scenario becomes even more critical when one notices the absence and the difficult (or nonexistent) access to mechanisms of auditing and control of consumption by the customer, which contributes negatively to the creation of a culture of economy and good use of electric energy. As a possible answer to these questions, this work intends to propose the architecture of an electric meter prototype. The initiative aims to research, evaluate and propose a low cost and affordable system from the economic and practical points of view for residential customers. This is expected to motivate the research and development of meters that do not demand complexity or cost above the consumption reality of residential customers. The system to be presented consists of a sine-wave (alternating current) voltage and current meter, whose core is a NodeMCU Lolin development module (equipped with an ESP8266-12E microcontroller), taking advantage of native resources to the Arduino platform. The features of the prototype are the measurements already mentioned, real-time active power calculation, energy consumption storage, recording of all this data in a datalogger and also transmission of this data with a Message Queue protocol Telemetry Transport (MQTT) to be retrieved by WEB applications or cell phones with Internet connection.

Keywords: Power Meter, ESP8266, Datalogger, MQTT protocol

LISTA DE FIGURAS

Figura 1 – Resumo da metodologia aplicada neste trabalho. 20
Figura 2 – Diagrama em blocos das partes componentes para o sistema proposto. 21
Figura 3 – Sensor de corrente elétrica invasivo ACS712 de efeito hall (a) e sensor de corrente não invasivo SCT013 (b). 25
Figura 4 – Encapsulamento do Circuito Integrado SD3004 (a), vista superior do módulo medidor de energia PZEM-004T (b) e versões dos TCs disponíveis para o módulo (c). .. 27
Figura 5 – ESP-01 (a), ESP-07 (b), ESP-12E (c), NodeMCU Lolin (d), Wemos D1 (*shield* similar Arduino) (e), Wemos D1 Mini (f), módulo desenvolvimento ESP-12F (g), módulo desenvolvimento ESP-201 (h). 31
Figura 6 – Diagrama de conexões em um broker MQTT, e sua estrutura: *publisher*, *subscriber* e *topics*. 36
Figura 7 – Arquitetura por trás do Blynk. 37
Figura 8 – Aparência exemplo de interface configurada com *widgets* do Blynk. 38
Figura 9 – Diagrama esquemático de exemplo de aplicação do RTC DS3231. 39
Figura 10 – Diagrama de pinagem de um SD Card padrão. 41
Figura 11 – Diagrama de conexões para medição do PZEM004T e também pinos da porta TTL. 45
Figura 12 – Código exemplo de teste do PZEM004T, com a biblioteca, variáveis globais e configurações de inicialização. 47
Figura 13 – Segunda parte do código agora mostrando o *loop* de execução do programa. 48
Figura 14 – Diagrama de elétrico de conexão da porta TTL do PZEM ao ESP. 50
Figura 15 –Detalhamento da inserção do resistor de 1k no PZEM (a) e Diagrama elétrico da alteração (b). 51
Figura 16 – Módulo RTC DS3231. 53
Figura 17 – Diagrama elétrico do módulo RTC DS3231 ao ESP. 54
Figura 18 – Roteador da Multilaser, N150. 55
Figura 19 – Módulo PCF8574T. 55
Figura 20 – Diagrama esquemático do módulo PCF8574T. 56
Figura 21 – Display LCD MGS 1602B, vistas frontal (a) e traseira (b). 57
Figura 22 – Módulo fonte de alimentação para Protoboard com reguladores de tensão de 5,0Vcc e 3,3Vcc da YwRobot. 58
Figura 23 – Diagrama elétrico do módulo YwRobot. 58
Figura 24 – Diagrama de fiação do protótipo finalizado. 60
Figura 25 – Interface do projeto configurada no Blynk rodando no smartphone. Em (a) primeiras configurações de testes, em (b) a aparência atual. 62
Figura 26 – Curvas comparativas de tensão: extraídas do analisador da Fluke (a) em comparação valores plotados com base nos dados salvos no cartão SD (do protótipo) (b). 63
Figura 27 – Curva de tensão gerada com dados Fluke 43B. 69
Figura 28 – Curva de tensão gerada com dados protótipo PZEM/ NodeMCU/ Blynk. 69
Figura 29 – Curva de corrente elétrica gerada com dados Fluke 43B. 70
Figura 30 – Curva de corrente elétrica gerada com dados protótipo PZEM/ NodeMCU/ Blynk. 70
Figura 31 – Tela final de logging gerada pelo analisador Fluke 43B. 71

LISTA DE TABELAS

Tabela 1: Comparação entre Tecnologias de Comunicação Comuns em IoT. 23
Tabela 2: Características elétricas e físicas do TC tipo anel sólido do PZEM-004T. 28
Tabela 3: Características elétricas de aplicação do PZEM-004T. 29
Tabela 4: Códigos do Protocolo de comunicação e suas funções no PZEM-004T. 30
Tabela 5: Comparativo de Consumo de Alguns Dispositivos Portáteis. 32
Tabela 6: Comparativo de Consumo de Alguns Dispositivos Portáteis. 34
Tabela 7: Características Principais do Circuito Integrado RTC DS3231. 40
Tabela 8: Valores Investidos no Protótipo. 61
Tabela 9: Descrição Característica do Ensaio Utilizando o Protótipo e o Analisador Fluke. 67

LISTA DE ABREVIATURAS E SIGLAS

A_{AC}	Ampéres em Corrente Alternada
ABESCO	Associação Brasileira das Empresas de Serviços de Conservação de Energia
A_{CC}	Ampéres em Corrente Contínua
ANEEL	Agencia Nacional de Energia Elétrica
API	*Application Programming Interface* (Ambiente de Programação de Aplicativos)
DHCP	*Dynamic Host Configuration Protocol* (Protocolo de Configuração Dinâmica de Endereços de Rede)
DNS	*Domain Name System* (Sistemas de Nomes e Domínios)
EEPROM	*Electrically-Erasable Programmable Read-Only Memory* (Memória de Somente Leitura Programável, Apagável Eletricamente)
EPE	Empresa de Pesquisa Energética
I2C	*Inter-Integrated Circuit* (Circuito Inter-Integrado)
IDE	*Integrated Development Environment* (Ambiente Integrado de Desenvolvimento)
IEEE	*Institute of Electrical and Electronic Engineers* (Instituto de Engenheiros Elétricos e Eletrônicos)
IoT	*Internet of Things* (Internet das Coisas)
IP	*Internet Protocol* (Protocolo de Internet)
IR	*Infrared* (Infravermelho).
LCD	*Liquid Crystal Display* (Display de Cristal Liquido)
LED	*Light Emitting Diode* (Diodo Emissor de Luz)
MISO	*Master Input Slave Output* (Saída Escrava Entrada Mestre)
MME	Ministério de Minas e Energia
MOSI	*Master Output Slave Input* (Saída Mestre Entrada Escrava)
MQTT	*Message Queue Telemetry Transport* (Transporte de Telemetria por Fila de Mensagens)
NTP	*Network Time Protocol* (Protocolo de Tempo em Rede)
RTC	*Real Time Clock* (Relógio de Tempo Real)
SCL	*Serial Clock* (Relógio Serial)

SD	*Secure Digital* (Digital Seguro)
SDA	*Serial Data* (Dados Seriais)
SoC	*System-on-Chip* (Sistema completo em um único circuito integrado)
SPI	*Serial Peripheral Interface* (Interface de Comunicação Serial)
SS/ CS	*Slave Select/ Chip Select* (Pino de Seleção de Chip)
TCP	*Transmission Control Protocol* (Protocolo de Controle de Transmissão)
UART	*Universal Asynchronous Receiver-Transmitter* (Transmissão e Recepção Assíncrona Universal)
V_{AC}	Tensão (volts) em Corrente Alternada
V_{CC}	Tensão (volts) em Corrente Contínua

SUMÁRIO

AGRADECIMENTO 6

RESUMO 8

ABSTRACT 9

SUMÁRIO 14

1. INTRODUÇÃO 14

1.1. JUSTIFICATIVA 16

1.2. MOTIVAÇÃO 17

1.3. OBJETIVO GERAL 18

1.3.1. Objetivos específicos: 18

1.4. ORGANIZAÇÃO DO TRABALHO 18

1.4.1. Revisão Bibliográfica. 18

1.4.2. Desenvolvimento do Protótipo 19

1.4.3. Resultados 19

1.4.4. Conclusões 19

1.4.5. Elementos Pós-Textuais. 19

2. REVISÃO BIBLIOGRÁFICA 20

2.1. METODOLOGIA E DETALHES DO PROJETO 20

2.2. Internet das Coisas, Código Aberto e Hardware Aberto 21

2.3. Código e Hardware Aberto 22

2.4. Resumo dos Protocolos e Padrões Atualmente Comuns para IoT 22

2.5. Medição de Sinais Elétricos e o Módulo de Medição PZEM004T 23

2.5.1. Normas Técnicas para Especificação 25

2.5.2. Módulo PZEM-004T da Peacefair 26

2.6. Placa de Desenvolvimento ESP8266 NodeMCU Lolin e Compatibilidade com a Plataforma Arduino 31

2.6.1. NodeMCU Lolin versus Arduino UNO R3 32

2.6.2. A Questão do Grau de Disponibilidade do Serviço 33

2.7. Computação em Nuvem e o Protocolo MQTT 34

2.7.1. Protocolo MQTT 35

2.7.2. Gerenciador MQTT Blynk 37

2.8. Datalogger 38

2.8.1. Relógio de Tempo Real 39

2.8.2. Data e Hora com Protocolo NTP ... 40
2.8.3. Time Epoch na Temporização (Data e Hora) ... 40
2.8.4. Cartão de Memória (Armazenamento) ... 41
2.8.5. Memória Flash do ESP8266 (Armazenamento) ... 41
2.8.6. Registrar Eventos em Nuvem ... 42

2.9. Outros Módulos Eletrônicos Utilizados ... 42
2.9.1. Visualização de Dados com Interface Digital ... 42
2.9.2. Fonte de Alimentação ... 43

3. DESENVOLVIMENTO DO PROTÓTIPO ... 44

3.1. Escolha pelo Módulo de Medição de Energia PZEM004T ... 44
3.1.1. Diagrama de Ligações à Rede Elétrica ... 44

3.2. Preparando a IDE Arduino para Trabalhar com ESP. ... 45

3.3. Motivação pela Mudança de Plataforma: NodeMCU (ESP) ao invés de Arduino (UNO R3). ... 46
3.3.1. Biblioteca e Comentários do Código Base do PZEM Rodando sobre o ESP ... 47
3.3.2. Sobre o que Esperar nas Leituras do PZEM: ... 49
3.3.3. Conexões Físicas do PZEM com o ESP ... 49
3.3.4. Tratamento de Leituras Erráticas ... 51
3.3.5. Usando módulo RTC. ... 53
3.3.6. Conexão à Internet via Wi-Fi. ... 54
3.3.7. Display LCD Usando Módulo Serial PCF8475A ... 55
3.3.8. Alimentação dos Módulos ... 57

3.4. Configuração do Servidor MQTT no NodeMCU Lolin ... 59
3.4.1. Usando o Blynk. ... 59

3.5. Diagrama de Fiação Completo ... 59

4. RESULTADOS ... 61

4.1. Investimentos Dedicados ao Projeto. ... 61

4.2. Captura de Tela dos Testes de Medição Usando Aplicativo Blynk ... 62

4.3. Teste Inicial de Precisão das Leituras do PZEM004T com Analisador de Qualidade de Energia 43B Fluke. ... 63

4.4. Problemas de Comunicação na Integração dos Módulos. ... 64

4.5. Ensaios em Ambiente Real com o Protótipo *versus* Fluke 43B. ... 66
4.5.1. Tratando o Registro dos Eventos em Ambos Sistemas com Planilhas. ... 67
4.5.2. Análise Final dos Resultados. ... 68

4.6. Recomendações Futuras. ... 71

5. CONCLUSÕES ... 73

REFERÊNCIAS ... 76

ANEXOS **79**

1. INTRODUÇÃO

É perceptível que a energia elétrica faz parte do nosso cotidiano. Neste trabalho, a perspectiva do consumidor é abordada.

Diversas atividades, das mais simples às mais sofisticadas, demandam consumo de energia elétrica, que por sua vez demandam uma crescente necessidade de produção. A demanda crescente é uma realidade, inclusive, entre pequenos consumidores.

Dados oficiais da Agência Nacional de Energia Elétrica (ANEEL) e do Ministério de Minas e Energia (MME) indicam que o consumo nacional residencial de energia elétrica já era próximo a 30% do total gerado (ANEEL, 2002), e mesmo tendo havido uma queda nesse percentual em meados de 2015 o consumo por esse grupo consumidor era próximo a 22% (MME, 2015). Dados da Empresa de Pesquisa Energética (EPE) também mostram um crescimento médio próximo a 2,5% a.a. (EPE, 2017) mesmo somando-se vários períodos de decrescimento e perdas econômicas.

Entretanto, a solução para o crescimento do consumo não reside apenas no crescimento da geração, mesmo este sendo necessário e estratégico. Segundo a Associação Brasileira das Empresas de Serviços de Conservação de Energia (Abesco) cerca de R$ 12,6 bilhões é o saldo negativo do desperdício de energia elétrica no Brasil, sendo que a principal fatia, em torno de R$ 5,51 bilhões, é do tipo de consumidor residencial (ABESCO, 2015; CUNHA, 2015). São estimadas perdas por desperdício de “460 mil GWh em quatro anos, suficientes para suprir a demanda do país em um ano” (GUADANIN, 2015).

É então perceptível que ações junto aos consumidores, especialmente aos pequenos, poderão vir a ter contribuição significativa, tanto na economia como na expansão dos serviços de fornecimento de energia elétrica. Também algumas hipóteses podem ser levantadas de que pode haver carência de ações políticas, mudanças de hábitos de consumo e de instrumentação e tecnologias acessíveis que permitiriam auxiliar nessa redução do desperdício. As primeiras hipóteses não são tratadas aqui.

O novo padrão tarifário residencial (Tarifa Branca) disponível desde janeiro de 2018 (ANEEL, 2017) agrega ainda mais importância de monitoramento de consumo por parte do cliente.

Focando na questão de instrumentação e tecnologias uma opção seria de medidores inteligentes. Ao pesquisar-se sobre o assunto nota-se que já é uma pauta e que houve algumas iniciativas, inclusive por parte das distribuidoras e concessionárias de energia elétrica, mas pelo que se nota em alguns artigos (HAYASHI, 2018; NUWER, 2015) ao que parece o foco

não é pela transparência dos dados em tempo real nas residências, mas para controle de tarifação dos usuários. Há instrumentos no mercado especialmente focados nas opções por Tarifa Branca e Geração Distribuída, mas quase nenhum produto acessível ao consumidor que lhe permita apenas monitorar seu consumo, sendo o caso mais próximo uma iniciativa AES Eletropaulo e WEG (WEG, 2014).

A questão da instrumentação também encontra um solo fértil de pesquisa e desenvolvimento em medidores do tipo bidirecionais, necessários em projetos de cogeração e geração distribuída, especialmente os mais próximos da realidade de clientes residenciais, como a geração fotovoltaica.

Agregando a este contexto, há ainda o surgimento crescente na última década de novas ferramentas e recursos voltados ao desenvolvimento de baixo custo para sistemas embarcados, agregar 'inteligência' e comunicação para aparelhos e "coisas" do cotidiano das pessoas, justamente denominado Internet das Coisas (ou mais popularmente *IoT*, do inglês *Internet of Things*). Este é justamente um termo cunhado que vem agregando uma quantidade enorme de equipamentos à internet, não mais se limitando aos computadores e mais recentemente aos telefones celulares.

Hoje é possível encontrar no mercado — ainda mais vastamente no cenário internacional — uma gama enorme de módulos (microcomputadores, microcontroladores, kits de desenvolvimento, etc.) com maior poder de processamento que muitos computadores antigos. O custo é de poucas dezenas ou unidades de dólar e o foco voltado para facilidade de integração e aprendizado, também fez com que a popularização de tais recursos tivesse não apenas grande aceitação como também maior democratização deste tipo de conhecimento.

Este trabalho procura, portanto, propor um medidor de consumo de energia elétrica de uso não industrial, aproveitando os recursos de desenvolvimento citados, dentro da filosofia Código Aberto[1]. Que permita ao usuário ter livre acesso às informações sobre seu consumo de energia elétrica, podendo inclusive ser alterado para agregar mais funções.

Protótipos dentro deste tema proposto já são encontrados na internet bem como em pesquisas acadêmicas. O que se pretende mostrar é a aquisição de habilidades e

---

[1] O "**Código aberto**, ou *Open Source* em inglês, é um modelo de desenvolvimento que promove um licenciamento livre para o design ou esquematização de um produto, e a redistribuição universal desse design ou esquema, dando a possibilidade para que qualquer um consulte, examine ou modifique o produto". Fonte: Wikipédia (2013), < https://pt.wikipedia.org/wiki/C%C3%B3digo_aberto>.

conhecimentos para a criação de um dispositivo nos moldes: acessibilidade, confiabilidade, baixo custo, transparência, modularidade.

Assim, as ideias de aplicação recaíram sobre módulos de desenvolvimento e circuitos pré-acabados utilizando microcontroladores em módulos de baixo custo. Inicialmente um projeto funcional foi concebido em plataforma Arduino (UNO R3), mas logo foi modificado e expandido para plataforma baseada em módulos ESP8266 (neste caso um NodeMCU v3, Lolin), que é o núcleo por trás de todo o projeto. O NodeMCU agrega a compatibilidade com o que já foi desenvolvido para Arduino e ainda a conexão Wi-Fi integrada ao sistema.

Para leitura dos sinais de tensão e corrente elétricas foram buscadas soluções em circuitos integrados dedicados, recaindo a tarefa sobre um módulo genérico da fabricante Peacefair, o modelo PZEM-004T. Cujo núcleo é do tipo 'sistema-em-um-chip' (*SoC – System-on-a-Chip*) SD3004 exclusivo, produzido pela SDICMicro (ambas empresas chinesas).

Com relação à comunicação, neste trabalho, optou-se por um protocolo 'máquina-à-máquina' do tipo Transporte por Telemetria em Filas de Mensagens (*MQTT – Message Queue Telemetry Transport*). Com base nesse protocolo há vários servidores livres (ou pagos), baseados em computação em "nuvem", de relativa facilidade de configuração, alguns disponibilizando aplicativos para dispositivos móveis (smartphones, tablets, etc.), permitindo também o aproveitamento de bibliotecas para uso em sistemas embarcados (como ESP8266, Arduino, RaspBerry Pi, etc.).

Em tempos de cogeração e geração distribuída, este é um trabalho que poderia facilmente servir de inspiração para projetos que envolvam medição bidirecional de energia elétrica. E sem prejuízo também poderia ser usado para estudos de consultoria para o novo modelo residencial de faturamento denominado de Tarifa Branca.

1.1. JUSTIFICATIVA

O consumidor tem (ou deveria ter) o direito de saber exatamente o que acontece em termos de consumo de energia em sua residência (ou aparelhos). Além do viés ético e econômico da transparência, há questões como de identificar anormalidades nas instalações elétricas (fugas, perdas, faltas), especialmente quando o usuário não se encontra no local (em horário de trabalho, férias, etc.).

A transparência do consumo se torna ainda mais relevante dada as novas formas de tarifação que vem sendo aplicadas ou estudadas para os clientes residenciais (como as bandeiras tarifárias: tarifa branca), demandas pré-contratadas (como ocorre com a telefonia

celular), sistemas de medição bidirecionais (clientes aplicando geração própria: fotovoltaicas e eólicas).

Ainda, os poucos medidores de consumo existentes no mercado nacional (em grande maioria importados) não agregam funções de ação remota (sejam de simples leitura ou mesmo de ações de telecomando) para o usuário.

Havia o desejo de aprendizado então (somado à oportunidade e acessibilidade), de desenvolver um sistema de medição nos moldes Código Aberto.

1.2. MOTIVAÇÃO

A área de embarcados e Internet das Coisas (*IoT – Internet of Things*) são movimentos que tomam cada vez mais força, popularidade e aporte de recursos e investimentos. Questões com relação às fontes de energia, automação de processos, solução de problemas em termos sociais, de economia ou ambientais, também são temas sempre em voga.

A ideia de criar algo que tivesse apelo e aplicação pragmática em algum desses quesitos (ou similares), que também fosse instigante já é motivadora por si só, seja nos esboços de projetos e até mesmo de ideias de negócios ou produtos. Uma dessas ideias era a necessidade de medição de consumo de energia elétrica de modo transparente e acessível, um produto que também poderia agregar alguma automação ou ser a porta de entrada para outras interconexões com aplicações diversas (em ambiente doméstico, por exemplo).

A filosofia de Código Aberto permite abertura para ensaios iniciais com fins didáticos e comerciais. Iniciar por plataformas prontas (como Arduino, Raspberry Pi, ESP8266, Teensy, etc.) resolve diversos problemas que acometem iniciantes (ou mesmo profissionais experientes), especialmente se fatores como tempo e recursos a serem investidos não são abundantes. É de se concordar que como produto acabado estas não parecem ter tanto propósito econômico muito maior que o didático.

Por trás de um módulo de desenvolvimento existe alto grau de complexidade e aprendizado (explícitos ou implícitos). Seu manuseio promove aprendizado, que incentiva e fortalece o desejo de se aprofundar mais nos assuntos. A linguagem de máquina, a arquitetura do hardware, o tratamento de sinais e os algoritmos são alguns dos pontos que vão tomando corpo mais complexo até um conhecimento mais sólido e pleno.

Este projeto, com todas as suas partes, detalhes e complexidades, não teria sido possível no curto espaço de tempo e com os poucos recursos disponíveis sem o acesso ao

Open Source. Desenvolver cada um dos módulos e soluções exigiria não apenas equipes multidisciplinares, como muito mais recursos e tempo disponíveis.

O que será percebido é que com poucos recursos se teve acesso a conhecimentos sobre tendências comerciais e tecnológicas, hardware, protocolos, circuitos dedicados, técnicas de comunicação, processamento de sinais, além de permitir criar uma relação com a comunidade colaborativa (fóruns, sites, repositórios) comum a este tipo de projeto.

1.3. OBJETIVO GERAL

Apresentar uma solução para leituras das grandezas básicas de sinais e de consumo de energia elétrica de um equipamento cujos valores estarão disponíveis ao usuário final por pelo menos três meios de acesso:

- Via internet (*on-line*): via dispositivo móvel ou navegador;
- Arquivo (*off-line*): gravação de arquivo de *logging* (datalogger), seja em alguma plataforma em nuvem ou mídia de estado sólido;
- Medição *in loco*: display digital.

1.3.1. Objetivos específicos:

- Adquirir, armazenar e permitir a visualização dos valores de tensão, corrente, potência ativa instantânea e consumo de energia elétrica;
- Permitir fácil conexão ao padrão de medição residencial (ou outra carga), critério de '*plug-and-play*';
- Ter um aplicativo com interface intuitiva em plataforma Android ou IOS, critério de multiplataforma;
- Possuir precisão equiparada a instrumentos comerciais, critério de acuricidade;
- Possuir back-up de dados e de alimentação dos módulos, critério de confiabilidade;
- Deve ainda permitir a inserção ou modificação do sistema sem custos significativos, agregando assim novas funções;
- Deve propiciar se possível, na sua construção, ideias e conhecimentos iniciais para desenvolvimento de um produto ou serviço comercial, em hardware mais robusto, simples e viável comercialmente.

1.4. ORGANIZAÇÃO DO TRABALHO

Após essa breve introdução de defesa do tema e título o restante deste trabalho aqui apresentado constará das seguintes partes:

1.4.1. Revisão Bibliográfica.

Esta seção inclui grande parte do trabalho, podendo ser observados:

Metodologia: que descreve como foi pensado o projeto e os passos seguidos até sua conclusão. Detalha o que foi realizado, na sequência em que ocorreu.

Conceitos, Hardware e Código:

- Trata cada questão a ser solucionada e da resposta dada;
- Faz análise de alguns itens usados;
- Delineia os produtos, partes e peças escolhidos, discorrendo sobre o que foi aprendido sobre;
- Em alguns pontos sinaliza outras possibilidades, quando pertinente.

1.4.2. Desenvolvimento do Protótipo

Nesta seção, a metodologia e aplicação dos conhecimentos gerados na seção de revisão bibliográfica são apresentadas ao leitor.

1.4.3. Resultados

Nesta seção procura-se mostrar os pontos onde se obteve sucesso na proposta e execução do protótipo. É inclusive onde os dados obtidos através do protótipo são comparados com valores lidos por outros instrumentos comerciais.

1.4.4. Conclusões

Última seção textual do trabalho trata de modo sucinto os fatos e considerações finais na conclusão do projeto. Cita ainda melhorias e possíveis caminhos que podem ser tomados para trabalhos futuros.

1.4.5. Elementos Pós-Textuais.

Contém material suplementar e corroborativo do trabalho:

- Referências Bibliográficas citadas;
- Anexos com esquemas, tutoriais, códigos e outros materiais necessários para maior entendimento, mas que não caberiam à fluidez do trabalho escrito.

2. REVISÃO BIBLIOGRÁFICA

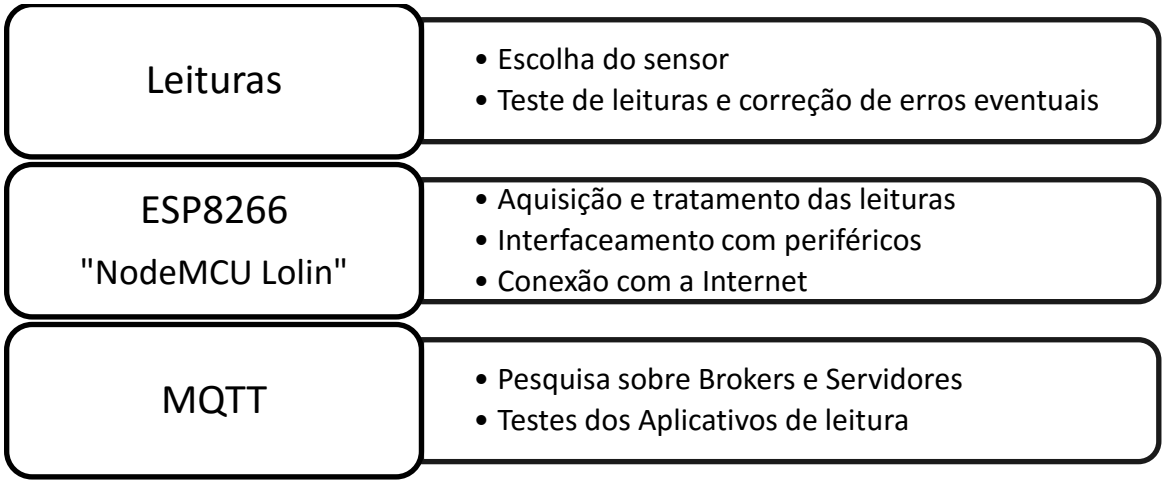
Nesta seção constam apenas os principais conhecimentos teóricos observados no trabalho (por questões de espaço e fluidez textual), referenciados quando necessário. Serão também apresentadas as informações técnicas que forem pertinentes. Algumas seções mais tutoriais ou pormenorizadas são apresentadas na seção de anexos.

2.1. METODOLOGIA E DETALHES DO PROJETO

O que se buscou neste trabalho foi conseguir desenvolvê-lo conforme os objetivos inicialmente propostos, com ênfase em modularidade e códigos abertos. Problemas e soluções propostas são citados ao longo desta seção.

O diagrama de blocos da Figura 1 ilustra as frentes de trabalho.

Figura 1 – Resumo da metodologia aplicada neste trabalho.

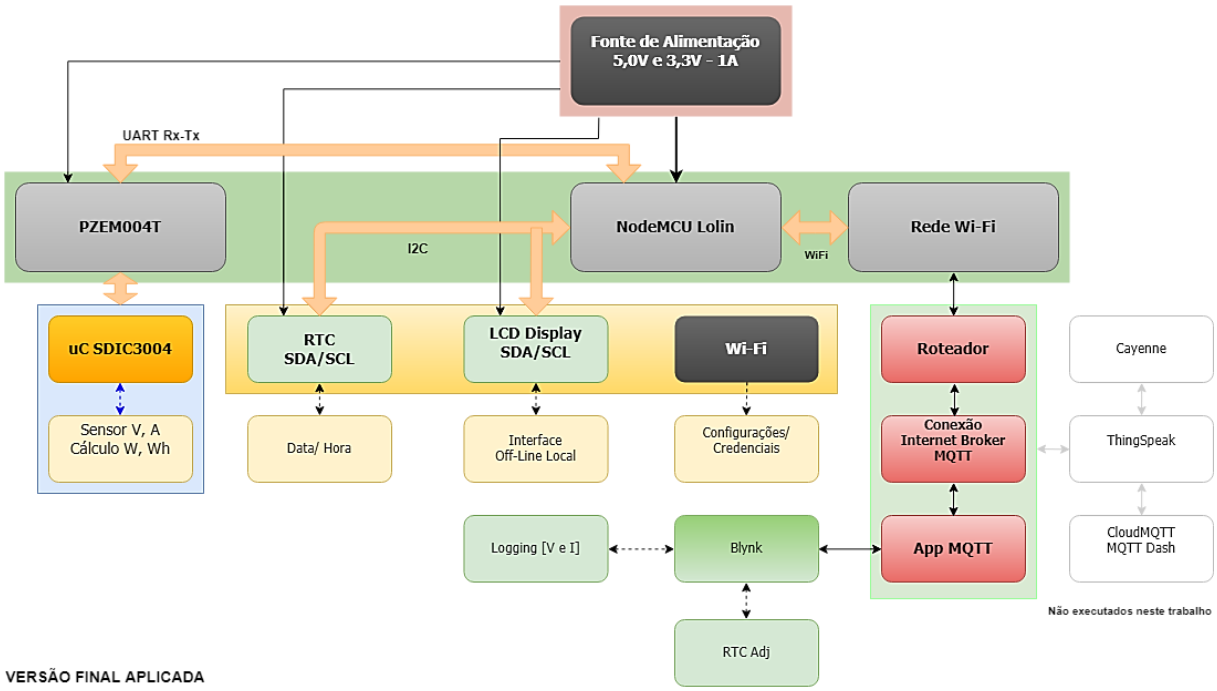


Fonte: Autor.

O esboço da metodologia permitiu organizar cronogramas de pesquisa por ordem de prioridades.

Já o diagrama de blocos de hardware da Figura 2 orienta e organiza o trabalho e dá a devida atenção a cada solução que seria necessária:

Figura 2 – Diagrama em blocos das partes componentes para o sistema proposto.



Fonte: Autor.

Este diagrama foi utilizado para determinar em que etapas ou módulos trabalham prioritariamente, resolvendo os problemas de cada parte assim que fossem surgindo. Os blocos em fundo verde foram resolvidos primeiramente, para garantir que se algo mais não tivesse resultados positivos, ainda assim haveria um projeto mínimo viável.

Os blocos na extrema direita da Figura 2 (em branco e cinza) foram apenas analisados neste trabalho, mas não houve tempo hábil de aplicação final.

2.2. Internet das Coisas, Código Aberto e Hardware Aberto

Mostrou-se um trabalho bem difícil definir em termos simples, concisos e abrangentes o que seria Internet das Coisas (referida a partir desse ponto do trabalho apenas como *IoT*, da siga em inglês). O termo mais simples seria o de conectar coisas (e não pessoas) à Internet, embora como lembrado por (JAVED, 2017, p. 14) seja mais do que isso.

Uma pesquisa sobre o tema, livros, artigos e outros formatos de publicações sobre o tema faz perceber que embora a prática de dispositivos conectados em redes de comunicação não seja uma coisa nova, a *IoT* parece ser algo desta última década. Relatório da (CISCO, 2011, pp. 2-3) trata como algo recente, "uma evolução da internet". De fato, a maioria das publicações são posteriores a 2010[2]. Segundo o mesmo artigo é estimado que a projeção

[2] Pesquisa realizada pela quantidade de buscas pelo termo em inglês de *IoT* no *Google Analytics*, por ano, desde 2000.

global de crescimento da *IoT* seja de 50 bilhões de dispositivos conectados até 2020. O que faz pensar que as projeções de faturamento e soluções para o seu desenvolvimento não serão pequenas.

MANCINI (2017) ainda nos lembra de um leque de novas possibilidades de aplicações como, por exemplo, as cidades inteligentes (*smart cities*); a saúde (*smart healthcare*); as casas inteligentes (*smart home*) e desafios que virão (regulamentações, segurança, padronizações). Que demandarão grande número de oportunidades para pesquisas, projetos e negócios.

2.3. Código e Hardware Aberto

O Código Aberto trata-se de uma definição que abrangia programas de computador cujos códigos poderiam ser acessados e modificados livremente sem a necessidade de pagar taxas aos desenvolvedores e nem haveria problemas com quebras de patentes ou plágio (contanto que os devidos créditos fossem dados). E em geral se popularizou pelos softwares livres, que muitas vezes são confundidos apenas por 'grátis', que são na verdade coisas distintas (ALECRIN, 2013).

O movimento se expandiu algum tempo depois para o conceito de Open Hardware[3].

2.4. Resumo dos Protocolos e Padrões Atualmente Comuns para IoT

Como é possível perceber, a *IoT* prescinde grandemente de comunicação (entre módulos, servidores, dispositivos, etc.). Ainda que haja desafios e se prospecte ainda novas tecnologias como resposta ou recursos a serem melhorados, a *IoT* faz ainda grande uso dos sistemas de comunicação populares ou de sistemas embarcados já existentes, listados na Tabela 1.

Um destaque especial para tecnologias recentes é inclusive o surgimento do Protocolo LoRaWAN[4]. No ambiente mais doméstico continuam a prevalecer as populares

---

[3] O conceito de Open Hardware é muito parecido com o conceito de software livre, popularizado pelo sistema operacional Linux, cujo código fonte é aberto, ou seja, você pode baixar o código, fazer as suas próprias modificações e criar um Linux customizado. Com o **Open Hardware** acontece praticamente a mesma coisa. São circuitos eletrônicos ou hardware de computador que podem ser copiados livremente, [todo projeto estará disponível]. Geralmente o desenvolvedor não cobra nenhuma taxa de licença, mas em alguns casos é exigido que o nome dele seja incluído nos créditos do projeto final. Também podendo exigir que qualquer projeto baseado em seu trabalho seja distribuído no esquema de Open Hardware. (THOMSEN, 2014).

[4] Mais algumas informações no artigo: Conheça a tecnologia LoRa® e o protocolo LoRaWAN™, do Eng. Vidal Pereira da Silva Junior, publicado no site Embarcados em 2016: <https://www.embarcados.com.br/conheca-tecnologia-lora-e-o-protocolo-lorawan/>.

redes Ethernet e Wi-Fi, com também grande participação de dispositivos 3G/4G graças ao advento dos *smartphones*.

Tabela 1: Comparação entre Tecnologias de Comunicação Comuns em IoT.

Protocolo	Alcance (m)	Frequência	Taxa (bps)	IPv6	Topologia
Ethernet	100/ 2000 m	N/A	10 Gbps	Sim	Variada
Wi-Fi	50 m	2,4/ 5 GHz	1300 Mbps	Sim	Estrela
Bluetooth BLE	80 m	2,4 GHz	1 Mbps	Sim	Estrela/ Mesh
ZigBee	100 m	915MHz/ 2,4 GHz	250 kbps	Sim	Estrela/ Mesh
3G/ 4G	35/ 200 km	1900/ 2100/ 2500 MHz	1/ 10 Mbps	Sim	Estrela
SigFox	10/ 50 km	868/ 902 MHz	10 – 1000 bps	-	-
LoRaWAN	2/ 5 km	Sub-GHz	0,3 – 50 kbps	Sim	Estrela

Fonte: Internet das Coisas: da Teoria à Prática (SANTOS, et al).

Há ainda uma gama de definições sobre redes e comunicações que precisam ser conhecidas para se trabalhar satisfatoriamente neste tipo de projeto. Algumas delas podem ser listadas aqui, mas seu detalhamento foge do escopo deste trabalho:

- Arquitetura Cliente-Servidor;
- TCP/IP, e suas camadas de aplicação, transporte, rede;
- DHCP e DNS, além do TFTP;
- Redes, Topologias e Infraestrutura;
- Protocolos pensados ou repensados para IoT: ZigBee, 6LoWPAN, WirelessHart, ISA100.11a, LoRaWAN, etc.;
- Outras tecnologias, como *Bluetooth* e IR;
- Além de linguagens de programação específicas para navegadores de internet ou aplicativos de celulares (Java, PHP, HTML, MySQL, Python, etc.) e desenvolvimento dos próprios *firmwares* dos sistemas embarcados (C++, Java, Lua, Assembly, etc.).

Ou seja, dependendo da aplicação que se pretende desenvolver, conhecimentos deste tipo precisarão ser levados em consideração.

2.5. Medição de Sinais Elétricos e o Módulo de Medição PZEM004T

A medição de um sinal passa pelo conhecimento de modelos que o descrevem. O conhecimento destes sinais e as características ideais dos sensores a serem empregados é crucial neste tipo de projeto, já que o sensor pode ser considerado o coração de todo o sistema de medição a ser projetado.

O módulo PZEM004T, com um μC SD3004 em seu núcleo, fará essas medições.

Os sinais elétricos principais deste trabalho são tensão e corrente elétricas, e sua forma de onda é aproximadamente senoidal (corrente alternada). As magnitudes serão da ordem de 260 *volts*, 100 *amperes* e 60 *hertz* (tensão, corrente e frequência, respectivamente).

O produto da tensão e corrente, medido diretamente, é a potência elétrica instantânea, como em (1). Esta é, portanto, a capacidade de trabalho instantâneo.

$$P = \frac{dW}{dt} \tag{1}$$

Onde: P é potência em *watts*, sendo $W = V.I$, V tensão em *volts* e I a corrente em *amperes* que por tratarem-se de medições discretas e já estarem convertidas para valores eficazes (o μC SD3004 já faz a conversão para valores *rms*), pode ser resumida a (2):

$$P = V.I \tag{2}$$

A energia consumida (e tarifada) é definida como sendo o trabalho pelo tempo (3). Logo a variável tempo também precisa ser considerada.

$$w_h = \int_{t_0}^{t} p\, dt \tag{3}$$

Onde: w_h é o trabalho realizado no período, p a potência instantânea infinitesimal e t_0 e t são o tempo inicial e final, respectivamente.

Em (3) é indicado que o somatório das potências consumidas (entregues) em um intervalo de tempo $t\ a\ t_0$ é a energia total do processo. Que também por tratar-se de um acumulador discreto (dentro do μC SD3004), resulta na simplificação:

$$Wh = P.t \tag{4}$$

Onde: *Wh* é a energia utilizada em *watt-hora*, P a potência instantânea e *t* é a unidade de tempo total do período.

Lembrando que o watt-hora é a quantidade de energia utilizada para alimentar uma carga com potência de um *watt* pelo período de uma hora e a tarifação do consumo de energia elétrica de clientes residenciais corresponde ao somatório horário desse consumo. Em nível de clientes residenciais o faturamento incide sobre a potência ativa instantânea consumida durante um determinado período de tempo, ou seja, a cobrança na classe Residencial B1 é sobre um valor médio quadrático (ou *rms*, do inglês *root mean square*).

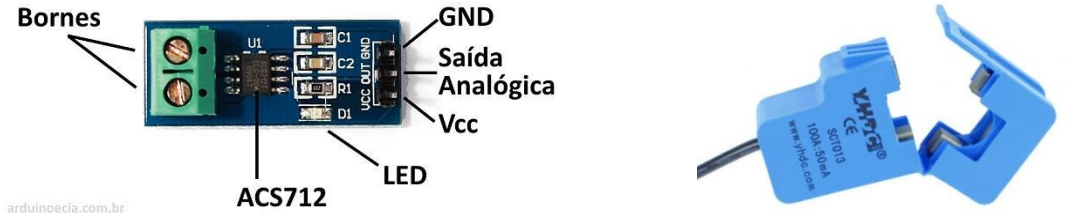
Com essas considerações iniciais, é preciso determinar os tipos e características de sensores (ou transdutores) necessários. O sensor basicamente reduz a um valor seguro ao mesmo tempo confiável o valor de uma grandeza, já o transdutor, além disso, converte o sinal lido para outro tipo de sinal (geralmente elétrico) em níveis que permitam a leitura.

Dentre as características a serem analisadas nos sensores, destacam-se[5]:

- *Sensibilidade*: termo que indica qual a menor variação de intensidade da grandeza medida, ou seja, o quão detalhado ou pequeno é o valor que pode ser lido.
- *Faixa Nominal* (ou *Range*): é basicamente a escala de menor a maior valor que o sensor ou instrumento é capaz de detectar.
- *Precisão* (ou *acuracidade*): diferente da sensibilidade é, em termos de percentagem, o quanto do valor lido reflete a realidade do valor da grandeza.
- *Linearidade*: dentro da faixa nominal, de que valores a que valores a resposta ao que é lido é sempre igual (margem de erro igual).
- *Histerese*: diz respeito ao modo de resposta do sensor tanto na variação positiva (subida) como negativa (descida) do valor da grandeza do sinal.
- *Tempo de Resposta*: basicamente é qual o limite de menor tempo para o qual o sensor consegue fazer a leitura (o sensor não conseguirá ler valores num intervalo de tempo menor).

Em termos de aquisição de sinais de corrente elétrica os sensores ainda podem ser classificados, entre tantas outras coisas, entre invasivos e não invasivos (quando não precisam alterar ou interferir na estrutura do sistema). Dois exemplos podem ser vistos na Figura 3:

Figura 3 – Sensor de corrente elétrica invasivo ACS712 de efeito hall (a) e sensor de corrente não invasivo SCT013 (b).



(a) (b)

Fonte: www.arduinopedia.com.br

Como se percebe, o sensor não invasivo tem a vantagem de não precisar interromper o fio para medir a corrente elétrica que circula por ele, além de propiciar maior segurança no manuseio e no projeto do hardware. Alguns desses sensores podem possuir circuitos internos que precisam ser levados em consideração no momento do projeto.

2.5.1. Normas Técnicas para Especificação

É importante salientar que este projeto tem fins didáticos. Nenhuma homologação ou certificado de calibração será requerido do sistema aqui apresentado.

Ainda assim, para aplicações comerciais (e seguras) uma série de normas (validadas por ensaios de homologação e calibração) devem ser aplicadas.

---

[5] Os termos podem diferir sutilmente em algumas literaturas técnicas.

Destaca-se neste trabalho a NBR 14519 (ABNT, 2000) que trata das especificações elétricas e mecânicas de medidores de consumo de energia elétrica mono e polifásicos de estado sólido (ou seja, medidores digitais).

Alguns dos dados interessantes dessa norma dizem respeito à presença de uma interface ótica serial assíncrona de transmissão de dados de dez bits (1 *start bit*, 8 *bits* de dados, 1 *stop bit*), numa taxa a partir de 9600 bauds, ou então de uma interface TTL, também serial assíncrona, com caracteres de dez bits reunidos em blocos de oito caracteres (80 bits), transmitidos a partir de 110 bauds. Nesta comunicação o intervalo de transmissão de blocos consecutivos é dito como de um segundo.

A norma trata dos diversos ensaios e dos valores que devem ser obtidos nestes.

Outra norma, a NBR 14522 (ABNT, 2000), "Intercâmbio de informações para sistemas de medição de energia elétrica – Padronização", trata dos protocolos que definem "o padrão de intercâmbio de informações no sistema de medição de energia elétrica, de forma a se alcançar a compatibilidade entre os sistemas e equipamentos de medição de energia elétrica de diferentes procedências".

Em outras palavras, tanto o hardware como o software desenvolvido precisam ser enviados para testes "às cegas"[6] e comparados com instrumentação homologada, necessitando ser aprovados em cada bateria de testes.

Este é, portanto, um resumo do que seria necessário se esse trabalho viesse a desenvolver um medidor de energia homologado e certificado (tanto para análise quanto para tarifação).

2.5.2. Módulo PZEM-004T da Peacefair

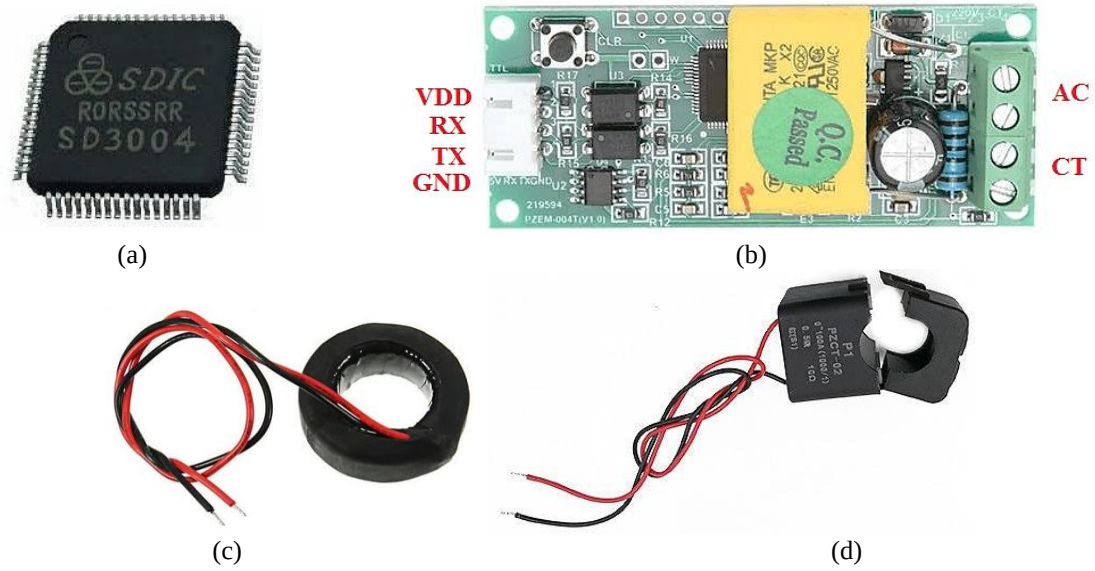
O módulo PZEM-004T é uma das versões de vários medidores de grandezas elétricas desenvolvido e distribuído pela Peacefair (China). O núcleo do módulo é um SoC SD3004, fabricado pela SDICMICRO[7]. Mas curiosamente, dentro do módulo da Peacefair, muitas das funções do circuito integrado (CI) estão desabilitadas (SDIC, 2012). Entre elas, não permitir e nem fornecer os códigos de acesso (talvez o firmware gravado nele não tenha essa função) ao valor do Fator de Potência (FP), do alarme de limite de leitura e aos sinais que habilitam usar um mostrador com display de LED de 7 segmentos (disponíveis em outras versões do produto).

---

[6] O autor não está identificado durante os testes, para garantia de imparcialidade e lisura.
[7] http://www.sdicmicro.com/index.html

Na Figura 4 consta a aparência do módulo e também do microcontrolador.

Figura 4 – Encapsulamento do Circuito Integrado SD3004 (a), vista superior do módulo medidor de energia PZEM-004T (b) e versões dos TCs disponíveis para o módulo (c).



(a) (b)

(c) (d)

Fonte: (Google).

Observa-se que ao adquirir o módulo é preciso ter atenção ao tipo de TC (ou CT) que acompanha, se será do tipo anel (Figura 4(c)) sólido ou alicate (Figura 4(d)). O módulo não funciona sem esse TC.

Recomenda-se adquirir o módulo com a versão de TC tipo alicate, pois facilita a passagem do fio pelo seu interior, a menos que o módulo seja instalado dentro de um circuito fixo que não necessite mover o cabo sob medição após instalado.

O TC que acompanha o módulo é calibrado para este, sendo que a tentativa de usar outros TCs causou leituras erradas. A Tabela 2 inclui alguns detalhes do TC que acompanhou o módulo (Figura 4(c)), usado neste trabalho.

Tabela 2: Características elétricas e físicas do TC tipo anel sólido do PZEM-004T.

Característica	Valores	Unidades
Impedância Paralela (Lp)	127,9	H
Impedância Série (Ls)	114,9	H
Fator D	0,31	-
Fator Q	2,9	-
Resistência Paralela	145,7	kΩ
Resistência Série	15,2	kΩ
Diâmetro Interno	22	mm
Diâmetro externo	32	mm
Largura	20	mm

Fonte: Peacefair

Para as medições foi utilizado um Medidor LCR de Precisão (Agilent/ Keysight, modelo E4980A) com os parâmetros de teste: frequência de 60 Hz e tensão 1,0 V, se notando uma variação de até 10% para algumas leituras. As dimensões físicas são aproximadas, e consideram a capa termo retrátil aplicada no acabamento do TC. Tanto a bitola do fio empregado, o número de espiras, quando as características físicas e eletromagnéticas do núcleo são desconhecidas.

Detalhes sobre o SD3004 encontrados constam no seu *datasheet*, no Anexo I.

O módulo de medição da Peacefair proporciona fácil conexão e também isolamento da rede elétrica sob medição (pelo uso de opto-acopladores em sua saída).

A Tabela 3 lista as características elétricas deste módulo.

Tabela 3: Características elétricas de aplicação do PZEM-004T.

Grandeza	Range do PZEM	Sensibilidade (Dígitos Visíveis)
Tensão	80,0 – 260,0 V	± 0,1 V
Corrente	0,00 – 99,99 A	± 0,01A
Potência	0 – 22 kW	± 1 W (de 0 a 9999W), ou ± 10 W (de 10000 a 22000W)[8]
Energia	0 – 9999 kWh	± 1 Wh
Frequência	50/ 60 Hz	Não Informada
Precisão AD	μC SD3004	16 bits

Fonte: Peacefair

Algumas das funcionalidades do módulo são:

- Medição das grandezas elétricas de tensão e corrente com saída RMS.
- Cálculo interno de potência ativa e registrador (com memória) de energia consumida.
- O circuito do módulo é alimentado com nível de 5V, comum à maioria dos microcontroladores.
- Função de exposição de tensão, corrente, potência ativa, energia consumida que podem ser lidas por comunicação serial TTL (UART e I2C interfaces), podendo se comunicar com diversos dispositivos. Com um barramento de 16 bits de dados.
- Um acumulador (memória) que armazena o valor de energia elétrica consumida desde a primeira conexão à carga, não perdendo este valor mesmo após perder-se a alimentação do módulo. Pode ser zerado com uma ação específica num botão de *clear* quando o módulo está alimentado.
- O μC SD3004 faz a conversão dos sinais lidos com 16 bits, e posteriormente os transmite pela UART em 4 pacotes (um para cada valor V, A, W, Wh), alocando em posições LSB e MSB (a UART recebe pacotes de 8 bits).
- O módulo PZEM possui uma memória EEPROM serial (two-wire) da Atmel (AT24C02N), responsável por guardar os últimos valores lidos, especialmente de energia (consumo) mesmo sem alimentação.

A comunicação com o módulo se dá através de comandos AT, comuns a roteadores de internet e outros dispositivos similares, mas esses comandos não foram pesquisados e nem analisados neste trabalho. Os protocolos de comunicação disponíveis para com o módulo estão listados na Tabela 4. Estes são os formatos de comunicação.

[8] Isso para medidores com displays 7 segmentos do produto original, mas na aquisição dos valores via serial não se sabe ainda a range máxima dos valores que serão obtidos.

Tabela 4: Códigos do Protocolo de comunicação e suas funções no PZEM-004T.

Função	Cabeçalho (Head)	Dados Enviados/ Recebidos	Soma (sum)
Tensão	B0 A0	**C0 A8 01 01 00** (envio da requisição do valor de leitura da tensão) 00 E6 02 00 00 (exemplo, responde que o valor da tensão é 230,2V)	1A 88
Corrente	B1 A1	**C0 A8 01 01 00** (requisição leitura corrente) 00 11 20 00 00 (exemplo, responde 17,32A)	1B D2
Potência Ativa	B2 A2	**C0 A8 01 01 00** (requisição potência ativa) 08 98 00 00 00 (exemplo, 2200W)	1C 42
Energia Consumida	B3 A3	**C0 A8 01 01 00** (requisição energia consumida) 01 86 9f 00 00 (exemplo, 99999Wh)	1D C9
Comunicação	B4 A4	**C0 A8 01 01 00** (corresponde ao endereço 192.168.1.1) [9] 00 00 00 00 00 (responde que a conexão foi estabelecida)	1E A4
Alarme[10]	B5 A5	C0 A8 01 01 14 (requisita um valor para definir o alarme de energia) 00 00 00 00 00 (responde que o limite foi definido)	33 A5

Fonte: Peacefair

Cada par alfanumérico é um valor em hexadecimal que corresponde, posição a posição, a um valor em dígitos decimais ou ao somatório destes. Os valores das grandezas lidas são concatenados posição a posição ou somados, conforme a requisição.

Os cabeçalhos identificam o tipo de função e a soma confere se os dados estão íntegros. Note que cada cabeçalho de requisição é sucedido pelo respectivo endereço "ip" configurado "C0 A8 01 01 00".

O endereço de IP e outros comandos possuem regra diferenciada, sendo representados no formato "255,255,255,255".

Atentar também para que "." são considerados "," e "," consideradas "." nos códigos e bibliotecas. Apenas as saídas (LCD, etc., devem ser ajustados para o formato brasileiro de representação).

Exemplos do que foi dito acima podem ser conferidos no Anexo G, e alguns detalhes da folha de dados (interpretados e não citados neste trabalho) no Anexo H.

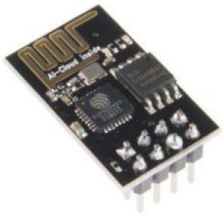
[9] Este endereço inicializa a transmissão de dados entre o módulo PZEM004T e o hardware que estiver requisitando seus dados. Precisa ser enviado a cada vez que o módulo perder alimentação.
[10] Função ainda não testada.

2.6. Placa de Desenvolvimento ESP8266 NodeMCU Lolin e Compatibilidade com a Plataforma Arduino

O microcontrolador ESP8266 é fabricado pela Espressif e agrega boa capacidade de memória de armazenamento global e conexão sem fio (WiFi, padrão IEEE 802.11) nativa, sendo também compatível com a maioria dos módulos e códigos da plataforma Arduino.

Trata-se de uma família de microcontroladores ESP8266 que se tornou tendência entre desenvolvedores e entusiastas do "Faça-Você-Mesmo"[11]. Os módulos mais conhecidos e algumas placas derivadas, estão ilustrados na Figura 5 a seguir:

Figura 5 – ESP-01 (a), ESP-07 (b), ESP-12E (c), NodeMCU Lolin (d), Wemos D1 (*shield* similar Arduino) (e), Wemos D1 Mini (f), módulo desenvolvimento ESP-12F (g), módulo desenvolvimento ESP-201 (h).



(a)

(b)

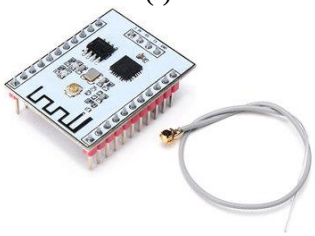
(c)

(d)

(e)

(f)

(g)



(h)

Fonte: Google.

Cada uma dessas versões libera um número definido de funções e pinos para o usuário. Por ser um microcontrolador de montagem em superfície, uma placa precisa ser

[11] Do inglês "*Do It Yourself"*, popular e comumente abreviado por DIY.

adquirida ou desenvolvida para soldar algumas dessas versões (permitindo acessar seus pinos de modo mais prático, como numa placa matriz de contatos, por exemplo).

Além disso, como todo microcontrolador, ele precisa de um circuito de gravação do firmware, seja adquirido ou montado para ele. Assim, a aquisição de módulos de desenvolvimento são uma vantagem, por propiciarem tanto o circuito de gravação quanto acesso aos pinos do ESP.

2.6.1. NodeMCU Lolin versus Arduino UNO R3

Ao se escolher um microcontrolador para um projeto é preciso saber de antemão algumas informações, entre elas o que se pretende desenvolver e se ele possui todos os recursos necessários.

A Tabela 5 lista as principais características do NodeMCU Lolin e ainda faz um comparativo destas com o Arduino UNO R3.

Tabela 5: Comparativo de Consumo de Alguns Dispositivos Portáteis.

Característica	Módulo	
	NodeMCU Lolin	**UNO R3**
Nível de Tensão	3,3 VCC	5,0 VCC
Entradas e Saídas Digitais (E/S)	17[12]	14
Entradas e Saídas Analógicas	1 (*in*) de 10 bits e 4 (*out*)	6, de 10 bits
Memória Flash	1 – 4 MB	32 kB
Memória RAM	20 kB	2 kB
Memória EEPROM	-	1 kB
Clock Principal	80 – 160 MHz	16 MHz
Alimentação do Módulo	5 -	7 – 12 Vcc
Corrente máxima E/S	12 mA	40 mA[13]
CPU	Tensilica l106 32 bits	AVR 8 bits
Wi-Fi nativo	Sim	Não
Linguagem Programação	LUA[14]	Processing[15]
Barramentos	UART0, UART1,SPI (master/ slave), HSPI (slave), I2C, I2S, IR.	UART0, UART1, SPI, I2C.

Fonte: *Datasheets* das fabricantes Espressif e Atmel.

[12] Nesta versão apenas 9 GPIOs (D0 a D8), além do pino A0, estão livres, as demais já são ocupadas ou dedicadas a outras funções.
[13] O máximo suportado somando o consumo de todas as portas é 200 mA.
[14] Referencial completo sobre a linguagem LUA: https://www.lua.org/portugues.html
[15] Um pouco sobre a linguagem *Processing* aqui: http://joaofaraco.com.br/arte_design/introducao-ao-processing/

Destaca-se nessa tabela a questão da linguagem de programação, que na verdade são como camadas de interpretação e não a forma nativa de acesso aos recursos do microcontrolador. Felizmente, ambos os módulos respondem à Interface de Desenvolvimento Integrado (ou do inglês *IDE - Integrated Development Environment*) muito popular aos usuários e iniciantes do Arduino.

Era intenção também fazer outros comparativos com outras placas com as funções Wi-Fi nativas ou adaptadas (como algumas versões de placas Arduino, Intel, etc.), mas não se encontrou muitas informações (condensadas e referenciadas) sobre elas que permitissem comparativos úteis a este projeto.

Usuários de placas Arduino também podem estranhar o modo de usar as portas e pinos de um ESP. A exemplo, o NodeMCU Lolin possui 30 pinos, entre alimentação e outras funções auxiliares. Mas é preciso notar que nem todos os pinos ou GPIOs são funcionais a todas as aplicações. Apenas 9 deles podem ser considerados pinos E/S clássicos, mas ainda assim alguns possuem limitações de uso por também serem dedicados a outras funções. É preciso ter atenção a isso tanto ao se escolher a plataforma quando à criação do código.

Como será observado durante o desenvolvimento do projeto, a conversão do projeto inicial de um datalogger mudando logo após para um sistema que também transmitisse dados de leituras para a internet foi determinante para a escolha do ESP8266 em detrimento a uma placa Arduino (como o UNO R3 já citado ou mesmo outra versão). Algumas das razões para isso foram:

- Uma alternativa Arduino com Wi-Fi se mostrou investimento maior que um ESP;
- O investimento em ESP se mostrou mais compensador que as demais placas Wi-Fi atualmente disponíveis (comparativos em dólar);
- O NodeMCU Lolin versão 3 foi preterido (entre os ESPs) por ser uma das poucas placas de desenvolvimento usando ESP que possui *firmware* atualizado e um maior número de GPIOs acessíveis.

2.6.2. A Questão do Grau de Disponibilidade do Serviço

Uma das questões envolvendo projetos embarcados ou de aplicação IoT é a questão da autonomia da alimentação, pois os módulos precisam de energia para funcionarem. Sistemas de fornecimento ininterrupto de energia (ou do inglês *UPS - Uninterruptible Power Supply*) são empregados para garantir o grau de disponibilidade do serviço.

Felizmente os módulos são muito econômicos, mas ainda assim possuem um consumo que limitará seu tempo de uso, especialmente ao agregar outros módulos ou durante a comunicação de dados.

Para um comparativo, levando em consideração apenas o módulo ESP envolvido no projeto, a Tabela 6 faz um comparativo do consumo de vários dispositivos conhecidos normalmente empregados em *IoT*.

Tabela 6: Comparativo de Consumo de Alguns Dispositivos Portáteis.

Módulo	Consumo por Modo Operação (mW)			
	Sleep	**Sem comunicação**	**Recebendo**	**Transmitindo**
Telos Motes	0,015	3	41	41 (0 dBm)
BLE nRF51822	0,0018	7	39	31,4 (0 dBm)
ESP8266	0,033	49,5	165 184,8	561 (17 dBm) 462 (15 dBm) 396 (13 dBm)
Arduino Nano	0,115	75	-	-
Raspberry Pi	-	1150 1500 1650	-	-
Smartphone	150	500	750	1000
Notebook	500	14000	14000	14000

Fonte: Internet das Coisas com ESP8266, Arduino e Raspberry Pi, p. 58.

Observa-se que mesmo comparado com uma das menores versões do Arduino, o ESP ainda leva vantagem com baixo consumo. Novamente, não foram encontradas referências sobre consumo de energia de outros módulos voltados para IoT (conexão Wi-Fi nativa), apenas das capacidades de corrente de portas que são muito parecidas entre os módulos pesquisados.

Algumas alternativas para autonomia de operação dos módulos é o uso de baterias recarregáveis e energia fotovoltaica.

2.7. Computação em Nuvem e o Protocolo MQTT

Segundo (OLIVEIRA, 2017, p. 75), computação em nuvem[16] é essencial aos projetos *IoT* por garantir confiabilidade a baixo custo, permitindo dispositivos em ambientes hostis, sem grande confiabilidade, desde que as informações sejam periodicamente armazenadas em nuvem. Assim, dispensando servidores e usufruindo de canais de comunicação de baixa

[16] A computação em nuvem (ou do inglês *Cloud Cumputing*) "é o fornecimento de serviços de computação – servidores, armazenamento, bancos de dados, rede, software, análise e muito mais – pela Internet ("a nuvem"). Fonte: https://azure.microsoft.com/pt-br/overview/what-is-cloud-computing/

confiabilidade. É algo que na prática seria como terceirizar servidores, como ocorreu desde os primórdios da Internet com a *web* e o *e-mail* (Google, Microsoft, etc.).

No cenário *IoT* ter dispositivos conectados à nuvem produz maior flexibilidade e escalabilidade. Além de disponibilizar um ambiente distribuído que pode ser programado em diversas linguagens e hospedar-se em uma infinidade de servidores. O protocolo já citado LoRaWAN, por exemplo, é um desses esforços de mover as coisas para uma rede própria e não congestionar as redes já existentes.

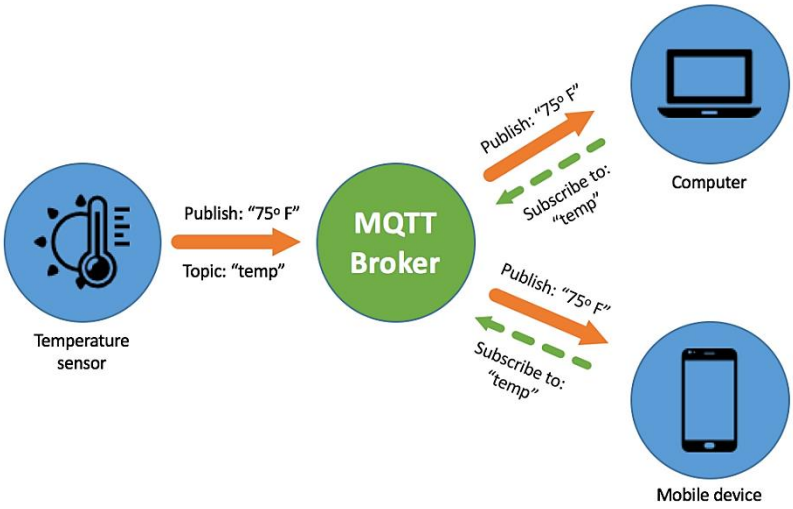
Neste trabalho não serão abordados conceitos de Web Services, Protocolos de Comunicação e Bancos de Dados, entre tantos outros assuntos relacionados ao tema, por não terem sido aplicados ao projeto. Mas conhecê-los é importante ao profissional que deseje se dedicar ao assunto.

2.7.1. Protocolo MQTT

De modo sucinto, é um protocolo "máquina a máquina" (M2M). Conforme explicado por (JAVED, 2017, p. 58), ele segue o modelo de "publicar-assinar" (*publish-subscribe*, no termo em inglês), fazendo uso de um "servidor" (conhecido como *broker*) para que um publicador (*publisher)* envie dados e um assinante (*subscriber*) os receba. Ambos não são conhecidos um do outro, eles apenas conectam-se ao *broker* (comunicação assíncrona). O servidor então avisa a todos os assinantes sobre a publicação de novos dados relevantes, num conceito denominado de "tópicos", que é semelhante a um *feed* de notícias.

Publicadores e assinantes podem ser, mas não se limitar, a sensores, máquinas, aplicativos móveis. A Figura 6 diagrama a simplicidade do processo:

Figura 6 – Diagrama de conexões em um broker MQTT, e sua estrutura: *publisher*, *subscriber* e *topics*.



Fonte: Leor Brenman, 2018[17].

Um mesmo dispositivo pode ser tanto publicador como assinante ao mesmo tempo. Um exemplo seria um aplicativo de celular que recebe informações e com base nelas o usuário pode agir enviando algum comando pelo mesmo aplicativo.

Em geral, servidores MQTT[18] se utilizam da Internet para conexão. E como ocorre com páginas web, existem diversos servidores livres ou pagos, sendo que pode-se usufruir de recursos extras como DNS, banco de dados, web, entre outros, de forma exclusiva, isolada e com camadas extras de segurança apenas nos serviços pagos.

Existe uma gama enorme de serviços de registro de informações IoT disponíveis (livres ou pagos) inclusive alguns dedicados ao ambiente acadêmico. Diversos destes se aproveitam do ambiente Linux. Alguns servidores possuem interfaces *web* configuráveis ou não. Alguns possuem também aplicativos móveis próprios, outros apenas aplicativos, e outros ainda podem ser usados com aplicativos de terceiros. As opções de recursos são variadas, mas neste trabalho não se teve intenção de verificar a grande maioria deles e seus respectivos servidores.

Como exemplo de servidores pagos tem-se: Microsoft Azure, Google Cloud Plataform, IBM Watson IoT Plataform, Amazon Web Services IoT. Do lado dos servidores livres o mais recorrente foi o Mosquitto[19], da Eclipse Foundation, muito citado em projetos colaborativos e com vasta documentação. Ainda assim, todos estes servidores requerem

---

[17] Artigo API Builder and MQTT for IoT: https://www.appcelerator.com/blog/2018/03/api-builder-and-mqtt-for-iot-part-1/
[18] Mais sobre IoT, MQTT e aplicativos: http://www.dobitaobyte.com.br/blynk-iot-e-mqtt/
[19] Mosquitto: https://mosquitto.org/

conhecimentos mais abrangentes e relativamente avançados, não cabendo aqui sua abordagem.

Felizmente, para facilitar o acesso e uso, algumas Interfaces de Programação de Aplicativos[20] (do inglês *API – Application Programming Interface*) amigáveis foram desenvolvidas por outras iniciativas para os servidores MQTT.

Além disso, plataformas como Arduino e ESP, por exemplo, possuem bibliotecas desenvolvidas e distribuídas especialmente para configurar e trabalhar com estas APIs.

2.7.2. Gerenciador MQTT Blynk

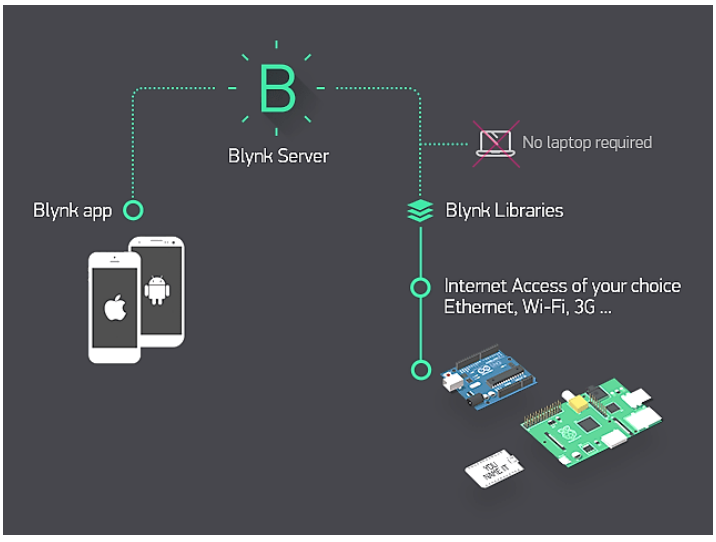
Segundo o site do desenvolvedor[21], o Blynk é uma plataforma de aplicativos que rodam em sistemas iOS® e Android® para uso com Arduino, Raspberry Pi e outros.

O dispositivo precisa apenas estar conectado à Internet - seja por Arduino, ESP8266, ou Raspberry Pi, com Wi-Fi ou Ethernet - que o Blynk poderá então ser usado para um projeto IoT. As conexões também podem ser realizadas também com: Bluetooth e BLE, USB (Serial) e GSM.

O aplicativo consiste de um painel digital onde é possível criar uma interface gráfica para o projeto arrastando ou soltando *widgets*[22]. Cada *widget* custa um valor diferente de 'créditos' chamados 'energias'. Mais créditos podem ser comprados, conforme a necessidade.

Na Figura 7 consta o resumo da arquitetura do Blynk.

Figura 7 – Arquitetura por trás do Blynk.



[20] API: https://canaltech.com.br/software/o-que-e-api/
[21] Blynk: https://www.blynk.cc/
[22] O termo *widget* refere-se a um componente que pode ser utilizado em computadores, celulares, *tablets* e outros aparelhos para simplificar o acesso a outro programa ou sistema. Eles geralmente contêm janelas, botões, ícones, menus, barras de rolagem e outras funcionalidades. Fonte: https://canaltech.com.br/produtos/O-que-e-widget/

Fonte: Blynk.

Pontos positivos:

- Relativamente de simples configuração, não se limitando a alguma placa ou módulo[23] específico.
- Ao trocar um *widget* por outro, os valores de 'energia' são estornados.

Limitações:

- Possui apenas 2000 créditos de energias.
- Não possui uma interface web para análise de dados ou envio de comandos.

A Figura 8 mostra exemplo de interface para celular:

Figura 8 – Aparência exemplo de interface configurada com *widgets* do Blynk.

Fonte: Blynk.

Segundo documentação[24] da desenvolvedora os serviços conectados Blynk podem enviar centenas de requisições por minuto (dependendo de como o código está escrito), o que pode forçar o servidor do aplicativo a desconectar o serviço (ao menos temporariamente).

2.8. Datalogger

É um aparelho, geralmente computadorizado, que tem por objetivo coletar dados, registrando essas informações para serem transmitidas para outros dispositivos, seja armazenando para download em um computador ou transmitindo a algum servidor remoto.

[23] O termo usado foi *shield* (escudo), muito comum entre usuários de Arduino e similares.
[24] Ver *Flood Error*: http://docs.blynk.cc/#troubleshooting-delay

São usados em locais onde a conexão direta a um computador se torna difícil ou onerosa ao projeto ou serviço.

O datalogger tem como principal característica coletar dados horários, em tempo real, registrando o momento em que foram lidos. Estes dados são armazenados em sua memória para posterior leitura ou transmissão, mas nada impede que paralelamente estes dados também sejam transmitidos em tempo real para outro local remoto.

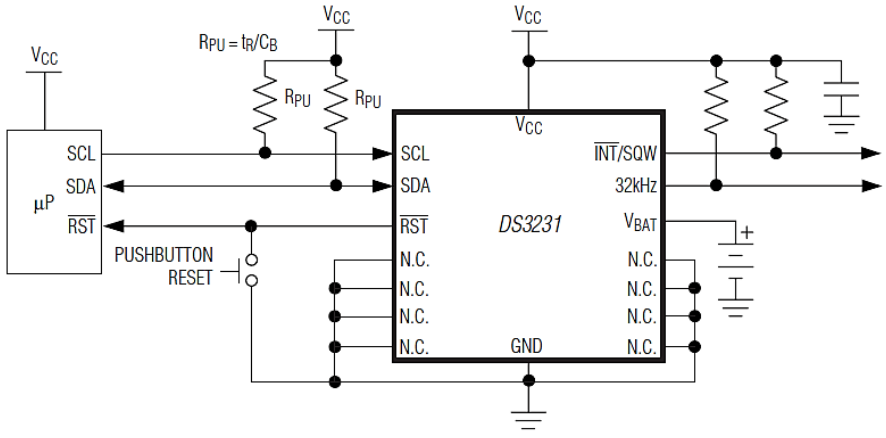
2.8.1. Relógio de Tempo Real

O dispositivo que registra o tempo de eventos é essencial num projeto de datalogger, por não fazer sentido ter os dados lidos, mas não se saber quando ocorreram.

Um relógio de tempo real (do inglês *RTC – Real Time Clock*) consta basicamente de um sistema que permite não apenas ajustar uma referência de tempo como também, a partir deste ajuste, manter este ajuste sincronizado com os demais sistemas — mantendo datas e horários, por exemplo, corretamente. A versão deste trabalho precisava ser digital, possuindo características de manter os valores ajustados independente da alimentação principal dos módulos ou da presença de dados a serem medidos.

O módulo RTC escolhido foi um baseado no CI DS3231, cujas informações básicas podem ser vistas na Figura 9:

Figura 9 – Diagrama esquemático de exemplo de aplicação do RTC DS3231.



Fonte: Datasheet DS3231.

E na Tabela 7 constam mais algumas informações sobre este módulo:

Tabela 7: Características Principais do Circuito Integrado RTC DS3231.

Característica	Valores
Tensão de Operação	3,3 Vcc (2,3 a 5,5 V)
Temperatura de Operação	0º a 70ºC
Tensão da Bateria	3,3 Vcc (2,3 a 5,5 V)
Corrente de Consumo	200 μA (110 a 650 μA)
Sensor de Temperatura do CI	± 3ºC de precisão

Fonte: Datasheet DS3231.

2.8.2. Data e Hora com Protocolo NTP

O NTP (do inglês, *Network Time Protocol*) é um protocolo utilizado para a sincronização dos relógios dos computadores. Os computadores possuem um relógio físico (RTC) interno, mas por algum motivo a data e hora do relógio podem ser perdidas ou corrompidas (perda da bateria, por exemplo). Assim, ao conectar-se num servidor (com data e hora atualizados), seja em rede local ou internet, o sistema terá condições de acertar seu próprio relógio.

Existem diversos servidores de NTP online, entre eles o ntp.br.

O NTP sempre trabalha na escala UTC e não é afetado pela mudança, nem interfere nos ajustes, do fuso horário ou horário de verão. O tratamento do horário local e horário de verão são funções dos Sistemas Operacionais (Windows, Linux, BSDs, Mac OS, etc). Se o Sistema Operacional estiver atualizado e corretamente configurado, a mudança acontecerá corretamente e de forma automática.

2.8.3. Time Epoch na Temporização (Data e Hora)

Notou-se importante citar que tanto ao usar dispositivos locais como remotos para registro de tempo, a maioria destes relógios faz uso de um formato de registro de tempo para facilitar a programação e manipulação por sistemas microcontrolados ou informáticos.

Tanto as bibliotecas para utilização de módulos RTC quanto os recursos de NTP fazem uso desse tipo de formato, que consiste, em geral, de contar a quantidade de milissegundos transcorridos desde uma data específica.

O time epoch[25] (também chamado UNIX timestamp) é um desses recursos, cujos milissegundos começaram a ser contados desde a data 1 de janeiro de 1970. Para converter

[25] Conversor e informações sobre Time Epoch: https://www.epochconverter.com/

data e hora em um padrão inteligível humano há ferramentas web, referências de códigos e também equações para planilhas de cálculo comuns.

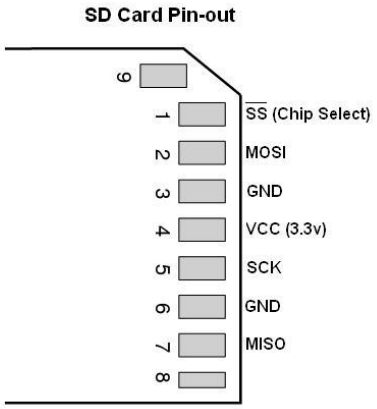
2.8.4. Cartão de Memória (Armazenamento)

Para armazenar os valores ou dados dos eventos em análise. Consiste em geral de um dispositivo não volátil (ou instável), uma mídia física que permita sua leitura posterior. Tal solução poderia ser tanto analógica (impressa em papel, por exemplo) como digital (num cartão de memória).

SD[26], miniSD, microSD, xD, Memory Stick e MMC são apenas alguns dos tipos de cartão de memória disponíveis no mercado. Embora executem bem o mesmo propósito, seja salvar músicas, fotos, vídeos, etc., a confusão sobre os diversos tipos é comum. Para este trabalho não é essencial discorrer sobre os diversos tipos.

O diagrama de pinagem de um cartão de memória SD pode ser observado na Figura 10:

Figura 10 – Diagrama de pinagem de um SD Card padrão.



Fonte: Google.

2.8.5. Memória Flash do ESP8266 (Armazenamento)

Alternativa para registro de eventos (não volátil) em memória é um recurso nativo do ESP8266 (e outros módulos), que usa o mesmo barramento e protocolo dos cartões de memória (SPI). Trata-se de um periférico de interface serial com um sistema de arquivos flash, conhecido no inglês por *SPI Flash File System* (abreviado por SPIFFS).

É uma alternativa atrativa, visto que alguns módulos ESP possuem até 4MB de memória flash (e os códigos de programas ocupam pouco espaço, em geral). A desvantagem é

[26] De *Secure Digital*, estando disponíveis em pelo menos três tamanhos dimensionais distintos.

a vida útil dessa memória, pois cada setor (ou ponto) dela aceita apenas algo em torno de dez mil gravações.

2.8.6. Registrar Eventos em Nuvem

Uma rápida pesquisa na internet indicou diversas opções de plataformas para receber, armazenar, analisar e apresentar as medições de temperatura, umidade, pressão, localização ou qualquer outra grandeza de interesse, com acesso online.

Fica claro que acesso via Internet a dados de medições físicas é especialmente benéfico nas áreas industrial, logística, saúde, predial, energia, saneamento e agronegócio.

Existem diversas plataformas pagas, inclusive que oferecem seus próprios equipamentos de medição. Uma plataforma deste tipo em geral é segura, escalável e possui um ambiente de desenvolvimento rápido de aplicações, que mesmo pessoas sem experiência em programação conseguem administrar.

As aplicações em nuvem ainda podem ser completamente personalizáveis, sendo possível criar diversos *dashboards* (páginas) com *widgets* (componentes gráficos) para exibição dos dados, configurar alarmes e eventos para as regras de negócio, enviar notificações via e-mail e configurar scripts para transformação ou programação lógica sobre os dados.

No caso deste trabalho, foi verificado que a plataforma Blynk (e os demais gerenciadores) possuem serviços similares que coletam e armazenam *logging* de eventos, cada um com suas especificidades. A escolha por este ou aquele serviço em nuvem, MQTT ou não, vai depender de análise das necessidades da aplicação.

2.9. Outros Módulos Eletrônicos Utilizados

Alguns módulos eletrônicos extras foram escolhidos para integrar este trabalho e permitir que os objetivos propostos fossem atingidos. Detalhamento sobre eles será dado no capítulo apropriado, pois em geral constam de peças não incomuns no meio eletrônico.

2.9.1. Visualização de Dados com Interface Digital

Neste trabalho foi inserido um visor de cristal líquido (do inglês *LCD – Liquid Cristal Display*) cuja única finalidade é conferir os valores de leitura e status do sistema sem precisar a recorrer a uma conexão com Internet (que pode estar indisponível). Este recurso também proporciona um rápido debug enquanto o projeto está sendo montado e testado.

2.9.2. Fonte de Alimentação

O protótipo faz medições de sinais de grandezas elétricas, mas para isso ser possível precisa ser alimentado com a energia necessária para o seu funcionamento.

A vantagem do baixo consumo de energia se perde um pouco com o acréscimo dos módulos de display LCD e do Leitor/Gravador de cartões de memória, que são os itens de maior consumo. O tráfego contínuo de sinais Wi-Fi em curtos espaços de tempo também contribui para menor autonomia.

A melhor solução para um caso como este é utilizar baterias recarregáveis em paralelo com uma fonte principal de energia (como a própria sendo medida), atuando em sua falta. Mas para os propósitos deste projeto, apenas a própria rede elétrica disponível será utilizada.

Os níveis corretos de tensão e corrente de cada módulo precisam ser supridos, pois há necessidade tanto de 5,0 como de 3,3 volts, o que requer reguladores de tensão. A tensão da rede local também é elevada (220 volts) e ao mesmo tempo do tipo alternada, precisando ser convertida.

A confiabilidade da alimentação dos módulos é crucial ao bom funcionamento e segurança do projeto.

3. DESENVOLVIMENTO DO PROTÓTIPO

Este capítulo aborda todo desenvolvimento prático e resultados deste trabalho. Procura dar detalhamento do processo, problemas encontrados e soluções aplicadas.

Alguns módulos de desenvolvimento (ESP8266-12F, ESP-201) foram utilizados durante testes de código e compatibilidade, mas para fins de clareza, todos os resultados citados aqui estão relacionados ao NodeMCU Lolin. Portanto, passam a ser referidos apenas como ESP no restante deste trabalho. Também será apenas utilizado o termo PZEM ao se referir ao módulo que faz as leituras de tensão e corrente e μC para microcontroladores.

3.1. Escolha pelo Módulo de Medição de Energia PZEM004T

Esta etapa do projeto é fundamental, obviamente, pois o que se precisa é justamente medir algo — o que fazer depois é consequência do sucesso nessa empreitada.

As primeiras tentativas foram de usar sensores simples e fazer todo tratamento do sinal via hardware e software. Por questões de limite de tempo e diversos contratempos tomou-se conhecimento de módulos prontos que resolviam problemas de precisão e confiabilidade da leitura das grandezas necessárias a este trabalho.

O ideal para um projeto de sensores modulares seria desenvolver um com base em sensores analógicos (sensores, transdutores, circuitos eletrônicos) ou digitais (com a aquisição de um SoC) [27]. O que se percebeu foi que isso, por si só, demandaria um trabalho único de pesquisa e desenvolvimento, detalhando demais o que seria necessário para um *datalogger*.

Por este e outros motivos, saber dos módulos da Peacefair durante a pesquisa propiciou motivação suficiente para testar como solução ao projeto. Mesmo não podendo ser considerado um sistema de Código Aberto, seu custo de aquisição e possíveis facilidades foram um incentivo ao seu uso.

Deste ponto em diante, ao se referir ao módulo utilizado se usará apenas a referência a PZEM, módulo de medição ou módulo (subentendido).

3.1.1. Diagrama de Ligações à Rede Elétrica

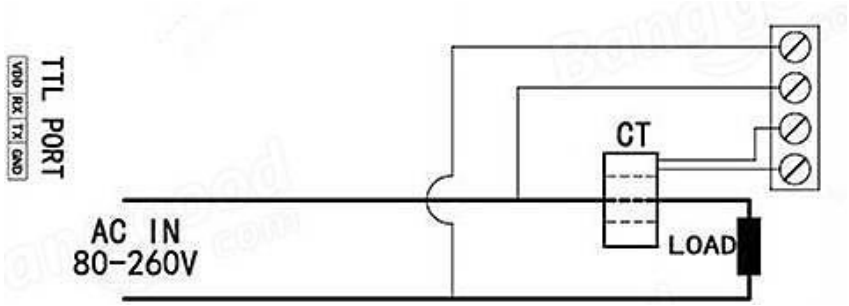
Devido ao fato de o PZEM não ser de código aberto, a documentação e detalhamento que poderiam ser úteis para solução de diversos problemas ou até de novas bibliotecas fica

[27] Como este oferecido pela ST Semiconductors: http://www.st.com/en/data-converters/stpm01.html#samplebuy-scroll ou este outro pela Cirrus Logic: https://www.cirrus.com/products/cs5463/.

comprometida. Ainda que um sistema funcional possa ser montado, algumas limitações ou problemas podem ser de difícil solução.

O modelo de módulo adquirido não possuía uma caixa e nem os displays LED 7 segmentos, consistindo apenas do módulo com seu TC. A conexão do módulo é detalhada na Figura 11:

Figura 11 – Diagrama de conexões para medição do PZEM004T e também pinos da porta TTL.



Fonte: Peacefair.

O sensor de corrente trata-se de um transformador de corrente (indicado por CT). Os pinos de conexão de dados e alimentação do módulo situam-se em lados opostos da placa de circuitos, com isolação por opto-acopladores. As conexões são do tipo pinos header ou conectores KRE, com marcações claras através da serigrafia da placa.

É de vital importância que tanto o CT quando os fios fase e neutro não sejam conectados aos terminais KRE na sequência errada. Portanto, deve se seguir exatamente a ordem mostrada. Considere a vista da Figura 14 como igual à da vista superior da Figura 4(b).

Também o manuseio do módulo deve ser realizado com cuidado para evitar curtos-circuitos e choque elétrico, pois todos os pontos de solda estão disponíveis ao contato. Recomenda-se usar uma superfície em madeira ou de vidro se o PZEM for testado fora de qualquer caixa, devendo a bancada estar limpa de resíduos ou outros objetos.

3.2. Preparando a IDE Arduino para Trabalhar com ESP.

Já foi citado que as placas ESP podem ser gravadas via a IDE da plataforma Arduino. Para que isso seja possível, é necessário instalar algumas bibliotecas e fontes externas. O modo de fazer isso está descrito sucintamente no Anexo F.

A numeração que aparece impressa no corpo do NodeMCU Lolin não tem qualquer relação com o número dos terminais do ESP8266 que são declarados dentro do código escrito na IDE. A pinagem equivalente do NodeMCU pode ser conferida no ANEXO D. Onde, por exemplo, “0” (D0) gravado na placa deve ser declarado no código como “16” (GPIO16) e não “0”.

3.3. Motivação pela Mudança de Plataforma: NodeMCU (ESP) ao invés de Arduino (UNO R3).

Se a correta leitura dos sinais é a primeira etapa a ser resolvida, esta segunda etapa é a que determina o microcontrolador (controle e comando) do sistema. Assim é necessário determinar qual melhor relação custo *versus* benefício.

Primando pela agilidade na execução do protótipo, a opção foi por plataformas que precisavam atender aos seguintes critérios:

- Que a preocupação principal não fosse a polarização correta do uC (que demanda tempo e incorre em diversos erros de prototipagem);
- A linguagem de programação fosse de simples aprendizado;
- Que houvesse fácil acesso a algum suporte profissional em caso de dúvidas;
- O custo do protótipo não fosse superior ao do "equipamento" a ser medido;
- Fácil disponibilidade de peças e módulos compatíveis, com bibliotecas e exemplos acessíveis.

Além disso, o projeto não se destina à produção em larga escala, então a questão de custo poderia ser sacrificada um pouco sem grandes prejuízos de aprendizado e propósito.

No começo do trabalho a plataforma escolhida foi o Arduino, com a sua placa UNO R3. O projeto do *datalogger* foi concluído (e apresentou-se funcional) naquele momento, mas existiu a demanda por uma interface independente via Internet. A fim de incorporar para leitura em tempo real, uma nova abordagem se fez necessária. Eis o cenário:

- Qual tipo ou ferramenta de interface em tempo real poderia ser usada?
- O que atenderia melhor a um protótipo sem o foco em um aparelho que pudesse ficar no padrão de medição de energia, ou então que fosse portátil?
- Que modificação permitiria ainda para que o projeto não precisasse de grandes intervenções de infraestrutura no local de seu uso?
- Como aproveitar todo o trabalho de pesquisa e desenvolvimento já realizado?
- O Arduino utilizado já estava com boa parte de sua capacidade de armazenamento utilizada, o que poderia gerar incompatibilidades ao agregar novas funções (sem um enxugamento de código).
- Os prazos de execução seriam viáveis?

A resposta a estas questões se apresentou em módulos ESP8266 e em sua apregoada capacidade de conexão com a Internet, aliadas à sua capacidade de memória e baixo custo.

A questão que agora precisava ser respondida é se os mesmos módulos e bibliotecas disponíveis no Arduino poderiam ser aproveitados no ESP. É o que será abordado a seguir.

3.3.1. Biblioteca e Comentários do Código Base do PZEM Rodando sobre o ESP

O código deste projeto em sua íntegra pode ser lido no Anexo C deste trabalho, e não será transcrito no corpo principal deste. O código também poderá ser baixado no repositório do GITHub[28] (onde dúvidas poderão ser respondidas e contribuições serão bem-vindas).

Ainda sobre o código postado, ele é funcional para este trabalho, mas com certeza pode ser adaptado, melhorado ou mudado. Inclusive para que certas funcionalidades sejam melhoradas, como não travar quando algum dos diversos blocos não inicializar.

Como percebido, para coletar os dados lidos pelo PZEM é necessário que se envie e se leia linhas de código de seu μC, o que pode ser bem trabalhoso quando se deseja um desenvolvimento de um sistema com certa rapidez. Além do mais, já foi citado que não há documentação clara nem sobre o módulo PZEM e nem sobe o μC SD3004. Felizmente, para uso na plataforma Arduino, foi criada uma biblioteca que permite acesso e leitura do PZEM (SOKOLOV, 2018), cujos comandos principais serão abordados neste momento.

As Figuras 12 e 13 apresentam trechos de código para teste de funcionamento do PZEM na IDE Arduino. As linhas serão comentadas a seguir:

Figura 12 – Código exemplo de teste do PZEM004T, com a biblioteca, variáveis globais e configurações de inicialização.

```
 1 #include <SoftwareSerial.h>   // IDE inferior a 1.6.6
 2 #include <PZEM004T.h>
 3
 4 PZEM004T pzem(10,11);         // IOs do RX, TX
 5 IPAddress ip(192,168,1,1);
 6
 7 Void setup() {
 8 Serial.begin(9600);
 9 Pzem.setAddress(ip);
10 }
```

Fonte: SOKOLOV (código).

A biblioteca <SoftwareSerial.h> é necessária apenas para versões inferiores a 1.6.6 da IDE. A inclusão da biblioteca <PZEM004T.h> permite manipular os dados do PZEM com linhas de código mais simples e inteligíveis. Esta biblioteca não se mostrou completamente compatível com o ESP, e ao contrário do Arduino, poucas foram as portas em que foi possível fazer leituras. Isso é perfeitamente compreensível, pois se tratam de arquiteturas de hardware diferentes.

O comando “IPAddress ip(192,168,1,1)” não tem relação com conexões de internet neste caso, mas é o mesmo usado em bibliotecas de conexão, justamente por que comando de

[28] GITHub: https://github.com/gersonsena/NodeMCULolin_PZEM004t_Meter

inicialização do PZEM tem este formato. Este é um valor que não deve ser alterado, nem as vírgulas trocadas por pontos. Declarar com o nome “ip” facilita a escrita repetitiva do código, mas deve ser mudado caso outras funções de configuração IP sejam testadas (o que não foi realizado neste trabalho).

O comando “PZEM004T pzem(10, 11)” indica em quais pinos do ESP os dados vão ser enviados e recebidos (RX, TX). Neste projeto precisaram ser mudados para as GPIO’s 00 e 02 (D3 e D4 respectivamente), pois outros dois pares de pinos possíveis foram ocupados para outros propósitos.

A linha cinco (Figura 12) foi modificada com a inclusão de um laço *while* (ver o código no Anexo C) para forçar a conexão do módulo tanto no Arduino como no PZEM. A necessidade desta modificação é de que quando a conexão não era estabelecida na primeira tentativa se faziam necessários diversos *resets* físicos no ESP (ou Arduino) até que esta fosse concretizada. Assim, o código força a conexão até sua concretização (sem a conexão o PZEM não retorna qualquer informação de erro, apenas não transfere os dados internos). O laço foi inserido no *setup* por ser necessária apenas uma requisição com sucesso enquanto o PZEM permanecesse alimentado[29].

Não é redundante frisar também que o módulo PZEM apenas consegue se conectar (inicializar comunicação) se houver sinal de tensão em sua entrada (pinos AC da Figura 4(b)). Após realizada a conexão via UART (ESP ou Arduino), a tensão AC pode cair abaixo de 80 Vac (falta) e ainda assim o módulo continuará a comunicação sem necessidade de *resetar* (ESP ou Arduino).

O *baudrate* usado foi o de 115200, diferente dos 9600 mostrados. Mas qualquer uma dessas taxas poderia ser usada.

Figura 13 – Segunda parte do código agora mostrando o *loop* de execução do programa.

```
12 Void loop();{
13 float v = pzem.voltage(ip;
14 if (v < 0.0) v = 0.0;
15 Serial.print(v); Serial.print("V; ");
16 float i = pzem.current(ip);
17 if (i >= 0.0) {Serial.print(i); Serial.print("A; ");}
18        float p = pzem.power(ip);
19        if (p >= 0.0) {Serial.print(p); Serial.print("W; ");}
20               float e = pzem.energy(ip);
21               if (e >= 0.0) {Serial.print(e); Serial.print("Wh; ");}
22 Serial.println();
23 Delay(1000);
24 }
```

Fonte: SOKOLOV (código).

[29] Refere-se a alimentar a placa do PZEM com os 5Vcc que ele precisa para executar suas funções, e não deve ser confundida com a tensão a ser medida da carga em teste.

As variáveis foram declaradas como tipo *float*, mas apenas as de tensão (v) e corrente (i) apresentam casas decimais.

Os comentários sobre as diversas outras bibliotecas utilizadas bem como seus exemplos não cabem neste trabalho, por justamente serem de ampla divulgação em repositórios e sites, portanto de fácil acesso e consulta, incluindo as páginas das próprias plataformas aqui citadas.

3.3.2. Sobre o que Esperar nas Leituras do PZEM:

Após estabelecida a comunicação, se o sinal de tensão cair a menos de 80 Vac, e o PZEM continuar alimentado (5,0 Vcc), os valores das quatro variáveis assumem todos um valor pré-determinado, que ao ser convertido em decimal resultaria "-1". Este indica que não há tensão da rede elétrica na entrada de AC.

A variável de energia consumida (e) não é zerada mesmo que falte energia e a alimentação do PZEM seja interrompida (5,0 Vcc), mas ainda assim o valor retornado, neste caso, ainda é "-1". Para que este valor armazenado na EEPROM do PZEM torne a zero (0 kWh) é preciso, enquanto o PZEM estiver alimentado, segurar o botão de *clear* (clr) da placa por cinco (5) segundos ou pouco mais, então soltar e tornar a pressionar rapidamente. Repetir até funcionar.

Um intervalo pode ser necessário após completar cada ciclo de leitura, para dar condições do ESP (ou Arduino) ler os dados corretamente.

Incompatibilidades da biblioteca <PZEM004T.h> com a arquitetura do ESP provocaram diversos erros de interpretação dos bits lidos. Os valores estavam nos pacotes, mas por algum motivo o ESP não os conseguia converter corretamente, retornando "-1" mesmo quando se sabia que o valor havia sido corretamente enviado pelo módulo PZEM. Esse foi um problema que precisou ser contornado.

3.3.3. Conexões Físicas do PZEM com o ESP

Como já se deve ter percebido, o módulo PZEM é alimentado com 5,0 Vcc, tendo uma comunicação nível TTL. Já o ESP possui entradas e saída em nível 3,3 Vcc (CMOS).

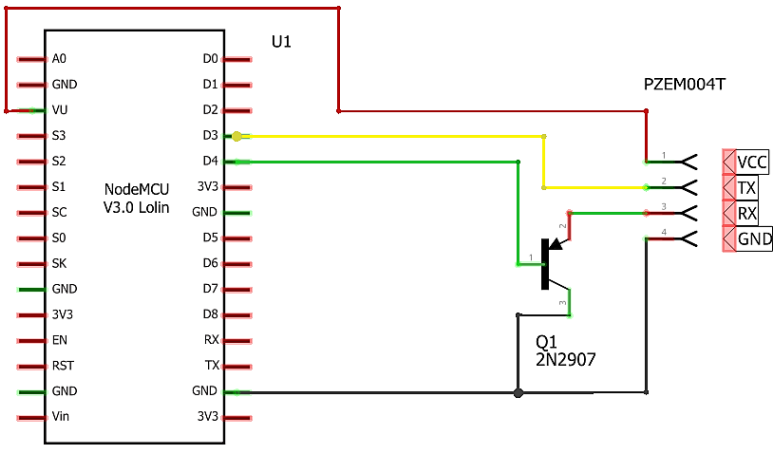
Também apenas 3 pares de GPIOs do ESP se mostraram funcionais em estabelecer a comunicação com o PZEM (sendo um deles os respectivos Rx-Tx do ESP, também aqui usados para debug via Monitor Serial da IDE).

Assim, ao invés de uma ligação direta, foi necessário readequar o sinal para que a comunicação entre PZEM e ESP fosse possível. Haveria alguns modos alternativos de

equalizar estes sinais, mas o que foi utilizado consistiu de fazer a alimentação do PZEM (5,0V) em separado e readequação apenas dos sinais de comunicação (Rx, Tx).

A Figura 14 mostra o esquema elétrico de fiação:

Figura 14 – Diagrama de elétrico de conexão da porta TTL do PZEM ao ESP.



Fonte: Autor, editado no Fritzing.

Foi realizada uma alteração física na saída da conexão da porta TTL do PZEM.

Um dos opto-acopladores (são dois PC817) possui uma ligação com os pinos 1 e 2 do PZEM através de um divisor de tensão com dois resistores (R1 e R2) de 1 kΩ em série (extraindo apenas 2,5 Vcc neste ponto). Com a inserção de um resistor (R3) de 1 kΩ de 0,125 W em paralelo com o resistor SMD (R1) naquele ponto (Figura 15(a)) obteve-se um novo divisor de tensão (5):

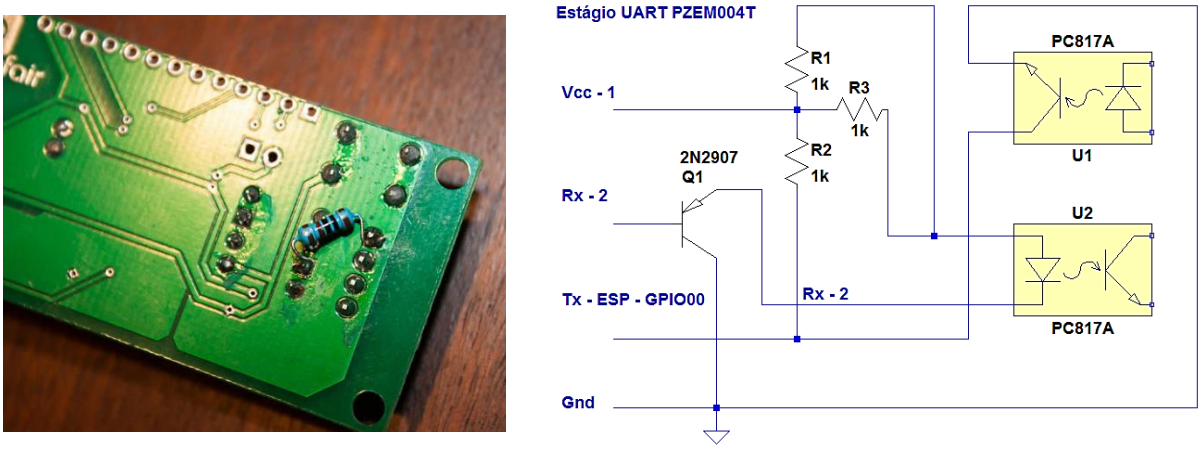
$$\frac{0{,}5k}{1{,}5k}.5V_{cc} = 1{,}67V_{cc} \qquad (5)$$

Observe que na configuração original neste ponto (anodo, pino 1 do opto) haveria 2,5 Vcc, ou seja, a diferença entre níveis (Hi-Lo, TTL) seria de 2,5 Vcc (quando Lo) e 7,5 Vcc (quando Hi). Assim, a mudança propiciou 1,67 Vcc (quando Lo) e 5,0 Vcc (quando Hi). Com isso foi possível garantir que o opto respondesse satisfatoriamente ao sinal 3,3V do Tx do ESP (pino D3, GPIO00) ao Rx do PZEM (pino 2).

Note ainda que, ao invés de fazer a ligação direta da porta D4 (GPIO02) do ESP com o pino Rx (pino 2) do PZEM esta foi realizada através de um transistor PNP (Q1). O transistor Q1 na porta D4 permite garantir que esta porta esteja em nível baixo ao reiniciar o ESP, para não causar erros de comunicação: o transistor evita que o pino esteja em nível alto. Pois se isto ocorrer o ESP não inicializa em modo gravação (modo *firmware*, ou seja, não compila o *sketch*), tendo que desligar e desconectar o módulo PZEM para conseguir fazer a gravação.

As alterações podem ser visualizadas na Figura 15:

Figura 15 –Detalhamento da inserção do resistor de 1k no PZEM (a) e Diagrama elétrico da alteração (b).



(a) (b)

Fonte: https://mysku.ru/blog/china-stores/43331.html e Autor.

Note que apenas R3 e Q1 não faziam parte do módulo PZEM original. Para usar com plataforma Arduino essa modificação precisa ser removida. As entradas de U1 e U2 são aquelas indicadas pelos pinos ligados aos diodos internos.

Após estas modificações as portas GPIOs D1, D2, D3, D4 do ESP tiveram resultados positivos em reconhecer e ler os dados do PZEM.

Atente que usar os pinos nativos Rx-Tx do ESP inviabilizam o *debug* via Monitor Serial da IDE, mas este nem é o maior problema, pois se o módulo PZEM estiver conectado a esses pinos a IDE simplesmente não consegue fazer a gravação do código (conflito de UART entre gravação e leitura).

Outra opção de adequação de sinais seria a aplicação de módulos conversores de níveis lógicos (*level shifters*), não disponíveis nem testados durante este trabalho. A construção destes conversores de níveis (idealmente os bi-direcionais) utiliza transistores FET do tipo NMOS, que além da indisponibilidade não se teria como confirmar que resolveriam esse caso de comunicação.

Simples divisores de tensão também não surtiram efeito positivo.

O diagrama de fiação usado desta etapa também pode ser visto no Anexo A.

3.3.4. Tratamento de Leituras Erráticas

Como já citado, toda a biblioteca que propicia a conexão dos hardwares dos módulos foi desenvolvida com base na arquitetura Arduino para o PZEM. Para readequar qualquer problema que pudesse vir a existir, a alteração ou criação de uma nova biblioteca seria o ideal, mas isto foge desta proposta. Ainda assim, para conexões com ESP várias publicações web não citaram encontrar problemas em seus projetos.

O que ocorreu neste trabalho é que em ciclos aleatórios, após leituras de alguns pacotes, valores estranhos e errados eram percebidos.

Como já explicado, quando um valor de sinal está ausente (ou é lido incorretamente) o PZEM envia como resposta o valor “-1”. Isso ocorre sempre que não houver sinal de tensão na entrada ou o PZEM não foi inicializado. Mas mesmo na presença de sinais em alguns momentos esse valor era devolvido no lugar do valor real de uma ou outra variável (V, A, W, Wh).

O normal era que todas as quatro variáveis lidas apresentassem como valor “-1”, quando o PZEM não estava com sinal de tensão (AC), o que não ocorria, indicando provável corrupção dos pacotes lidos. Ou seja, o PZEM estava enviando os valores, mas ao não receber o bit de paridade correto como resposta do ESP, aquele enviava a variável em questão com valor “-1” para a variável com problemas de interpretação. Observe maiores detalhes dessa operação no Anexo G.

Depois de tentativas de alguns tratamentos via código como média móvel ou laços condicionais de valores, o que resolveu o problema foi um laço comparativo, pegando apenas o valor maior de cada amostra. Considerando:

- Para o propósito deste trabalho, não era fundamental que cada leitura fosse aproveitada. Bastavam apenas alguns valores amostrados em intervalos de alguns segundos (o que é uma eternidade para microcontroladores), logo algumas leituras poderiam ser desconsideradas.
- Levando em conta também que numa ocorrência de falta de frações de segundo não eram cruciais, estabeleceu-se que seria considerada uma falta apenas se três sequências de leituras indicassem isso (e não apenas uma).

Assim, a cada três ciclos de extração dos dados do PZEM, estas seriam comparadas uma a uma para mostrar apenas o maior valor entre elas.

Após testes com resultados positivos, a função precisou ser aplicada a cada variável enviada do PZEM ao ESP, pois o problema não era nos dados do PZEM, mas sim no modo como o ESP conseguia ler e interpretar estes.

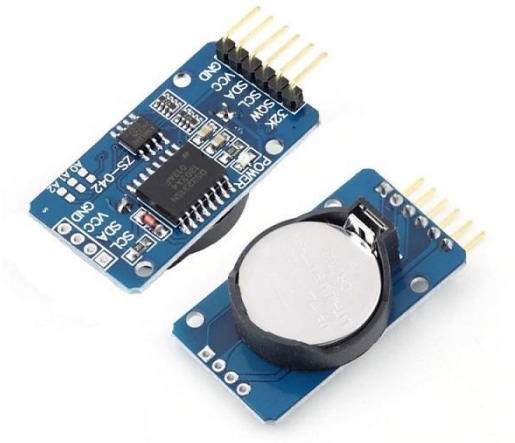
Além disso, o evento de falta de tensão (sinal AC da entrada do módulo menor que 80Vac) precisaria ser mostrado apenas quando as três leituras seguidas confirmassem esse evento.

O problema dessa abordagem foi que o número de amostras por minuto reduziu-se a algo em torno de sete a oito segundos (tempo gasto para rodar todas as rotinas e apresentar os valores ou mensagens). Esse é um problema não solucionado nesta versão.

3.3.5. Usando módulo RTC.

O módulo de RTC utilizado neste trabalho é mostrado na Figura 16:

Figura 16 – Módulo RTC DS3231.



Fonte: Google.

O módulo DS3231 usa uma bateria CR2032 de 3,6 V (que pode ser recarregável) e é alimentado em 3,3 Vcc, mas suporta bem até 5,5 Vcc.

Não existe muita documentação sobre o DS3231 citando suas características, como:

- Definição de alarmes e medição da temperatura.
- O módulo mostrado disponibiliza ainda uma segunda interface I2C para ligar um dispositivo em cascata, como um LCD, por exemplo.
- Também possui uma memória EEPROM (AT24C32) de 32K, útil para armazenar a configuração do relógio e como *datalogger***.** O endereço da memória pode ser configurado através dos jumpers A0, A1 e A2.
- O pino SQW é útil para o controle de alarmes por interrupções.
- O pino 32k raramente é utilizado, apenas se quiser obter os pulsos do oscilador de cristal (para escovadores de bits).

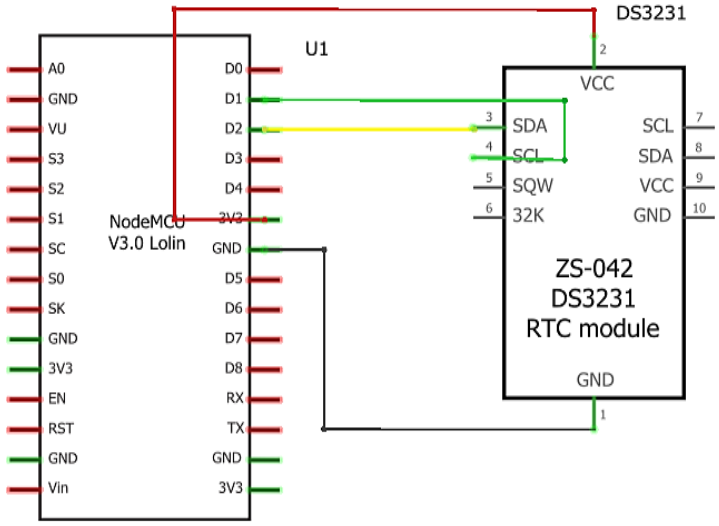
Mas essas funcionalidades tampouco foram exploradas aqui.

O módulo tem comunicação através de um barramento I2C. Por isso as bibliotecas utilizadas foram a <Wire.h> e <RTCLib.h>, a primeira para comunicação I2C e a segunda para leitura e gravação das informações essenciais do RTC. Caso seja necessário fazer uso de outras funções do módulo, vão ser necessárias, provavelmente, outras bibliotecas.

No final, este módulo acabou por não ser necessário ao sistemas, mas ainda assim foi mantido para mostrar que o barramento I2C estava operando bem com mais de um módulo conectado a ele ao mesmo tempo.

O diagrama de ligações ao ESP é mostrado na Figura 17:

Figura 17 – Diagrama elétrico do módulo RTC DS3231 ao ESP.



Fonte: Autor, editado no Fritzing.

Note que os dados lidos e escritos correspondem às ligações das GPIOs 05 e 04 (respectivamente D1 e D2).

O diagrama de fiação do RTC pode ser analisado no Anexo A.

3.3.6. Conexão à Internet via Wi-Fi.

Para usar os recursos de um servidor em nuvem é necessário acesso à Internet. A fim de facilitar as configurações durante toda a fase de testes foi criada uma rede doméstica com roteador Wi-Fi. Isso permitiu melhor dinâmica durante os diversos deslocamentos com o protótipo. A opção pelos testes em ambiente doméstico é explicada a seguir.

Por conta das políticas de segurança e arquitetura das redes de internet no ambiente institucional acadêmico, um roteador não poderia ser anexado à rede, o que exigiria do projeto conexão direta do ESP com a rede institucional, via MAC ID. Mas mesmo com a liberação de acesso (via MAC ID) das placas ESP testadas as configurações da rede EduRoam[30] (ou mesmo à rede principal com os modos de segurança WPA2 Enterprise) exigia configurações de IP dinâmicos e credenciais de acesso.

Os certificados de autenticação, além de consumir muita memória, são de difícil execução nessa linha ESP8266 da Espressif. Apesar de ser sinalizado que essa é uma das melhorias nos novos modelos ESP32 (mais caros, mais recentes, com menos documentação até esta edição). Alguns códigos e bibliotecas para esse tipo de conexão estão disponíveis apenas em placas Arduino (houve incompatibilidade com ESP).

[30] Um pouco sobre redes EduRoam: https://www.eduroam.pt/pt/sobre/descricao

Até a presente publicação deste trabalho o problema da rede institucional não foi resolvido. A solução para esta parte foi usar o sistema em ambiente doméstico, onde as configurações de redes não eram tão exigentes.

O roteador utilizado foi um Multilaser M150, cuja única configuração necessária foi estabelecer o padrão de IPs, nome e senha da rede sem fio e monitorar as placas conectadas. Na Figura 18 consta o modelo usado.

Figura 18 – Roteador da Multilaser, N150.

Fonte: Google.

O roteador personalizado deu agilidade ao processo de testes de conectividade e tráfego de dados.

3.3.7. Display LCD Usando Módulo Serial PCF8475A

Dada a pouca disponibilidade de GPIOs livres no ESP a alternativa é a possibilidade de usar barramentos seriais. No caso do display LCD isto então é necessário, dado o número de saídas que seriam necessárias. Assim, se fez uso também do barramento I2C para o LCD.

Uma das vantagens das filosofias de hardware aberto é a oferta de módulos com soluções prontas. Este é o caso do módulo PCF8574T mostrado na Figura 19, faz exatamente a conexão dos diversos pinos do LCD display apenas aos dois fios do barramento I2C:

Figura 19 – Módulo PCF8574T.

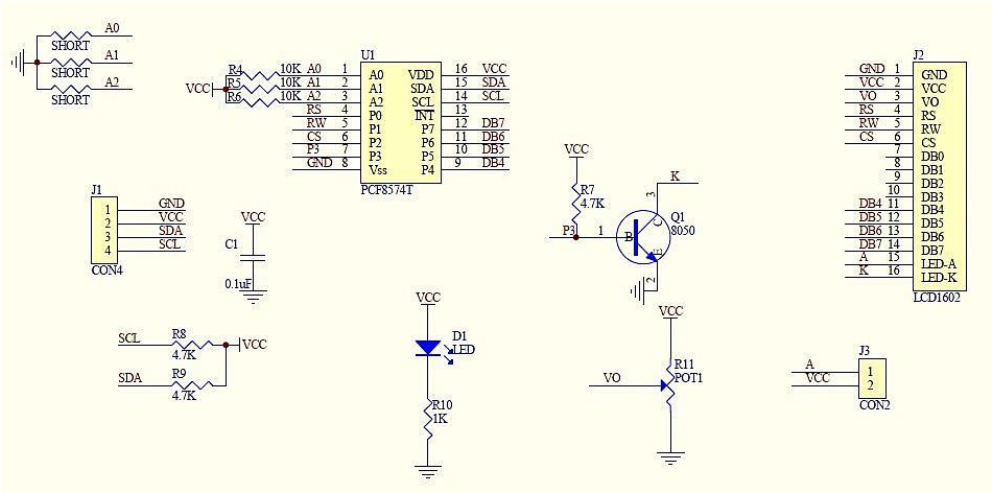
Fonte: Google.

O *jumper* “LED” permite a opção de controlar o *backlight* do LCD via código ou via hardware. Os pontos “A0, A1, A2” permitem alterar o endereço de comunicação do módulo. São possíveis mais de um módulo aplicado no mesmo barramento simplesmente alterando a conexão de algum desses pontos.

Os pinos “SDA" e “SCL” são ligados nas mesmas portas GPIO do módulo RTC já citado. Para conferir possíveis conflitos de endereços, um código[31] de teste foi rodado permitindo identificar os endereços atribuídos a cada módulo conectado no ESP.

O diagrama esquemático pode ser visualizado na Figura 20:

Figura 20 – Diagrama esquemático do módulo PCF8574T.



Fonte: sunrom.com.

Observe que na prática apenas 4 pinos de dados do display são usados na comunicação com o módulo.

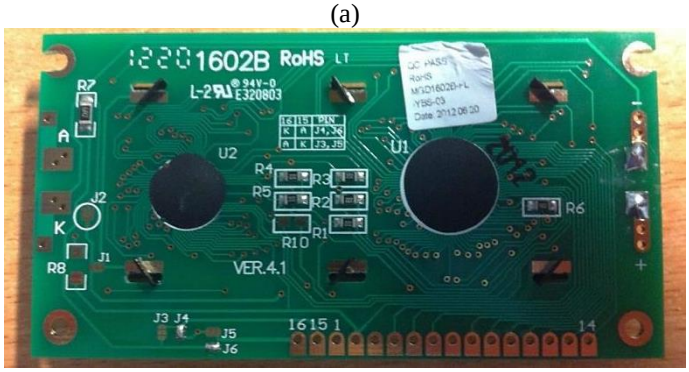
Uma alternativa ao módulo PCF8574T, usando apenas o circuito integrado, é mostrada no Anexo A (diagrama esquemático para o MDS1602 display).

O módulo display LCD utilizado foi o MDS 1602, mostrado na Figura 21:

[31] Código e Informações em: https://playground.arduino.cc/Main/I2cScanner.

Figura 21 – Display LCD MGS 1602B, vistas frontal (a) e traseira (b).

(a)



(b)

Fonte: Eletrodex e Laboratório de Garagem.

Este módulo também possui 16 pinos, sendo 8 pinos para comunicação de dados e os demais para controle e alimentação do módulo. É preciso atentar que os pinos 16 e 15 deste módulo ficam ao lado do pino 1 e não na sequência que seria esperada.

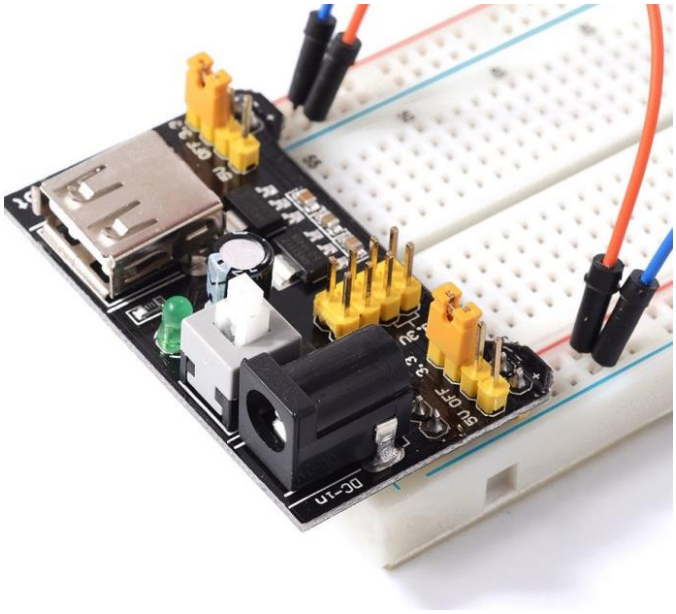
Para o uso deste display juntamente com o módulo PCF8574T foi utilizada a biblioteca <LiquidCrystal_PCF8574.h> no ESP. Outras bibliotecas não foram testadas no ESP até este momento.

3.3.8. Alimentação dos Módulos

Por se tratar de um sistema que deverá operar autonomamente essa é uma questão crucial que precisa ser pensada e projetada em conjunto.

Foi escolhida uma fonte de alimentação com capacidade de 1,0 A, que converte 220 V_{AC} para 15 Vcc (que é mais que suficiente para o projeto). Essa fonte alimenta o módulo da YwRobot Corporation (chinesa) com reguladores de tensão tanto para 5,0 Vcc quanto para 3,3 Vcc através de um plugue P4. Mais detalhes na Figura 22:

Figura 22 – Módulo fonte de alimentação para Protoboard com reguladores de tensão de 5,0Vcc e 3,3Vcc da YwRobot.

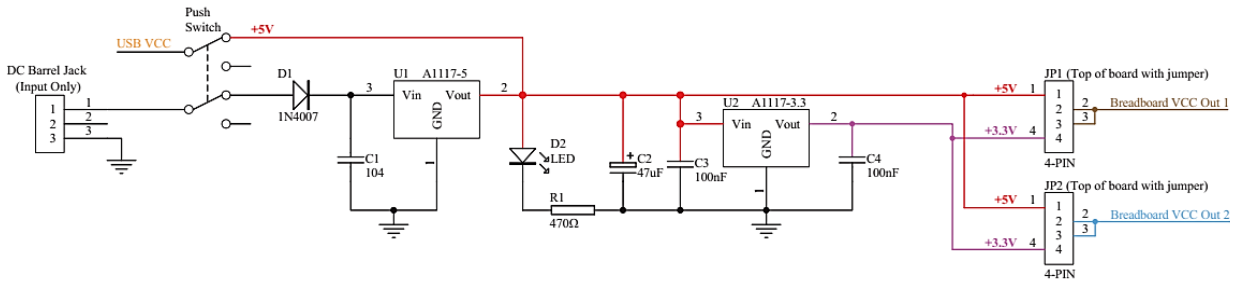


Fonte: YwRobot.

No módulo também há uma porta USB tipo A, um LED indicador de funcionamento, uma chave ON/OFF (também seleciona se a alimentação vem pela USB ou pelo Plugue P4), jumpers que selecionam se o barramento da *protoboard* é de 5,0 Vcc ou 3,3 Vcc. Há ainda saídas de 8 pinos para conexão sobre o módulo.

A Figura 23 mostra o diagrama elétrico (a versão completa está no ANEXO B):

Figura 23 – Diagrama elétrico do módulo YwRobot.



Fonte: Addicore.

Conforme descrito nos datasheets dos reguladores, essa fonte tem corrente nominal de 1,0 A, mas pode suportar picos de até 1,2 A por alguns segundos.

Note ainda que, embora os diagramas de fiação mostrados no Anexo A e nas figuras anteriores mostrem a alimentação dos diversos módulos diretamente no ESP, isto aconteceu apenas durante os testes isolados módulo a módulo. Quando os diversos blocos foram interligados a alimentação passou a ser exclusivamente desta fonte, inclusive o ESP (após desconexão da porta USB), via pinos Vin e Gnd (com 5,0 Vcc).

3.4. Configuração do Servidor MQTT no NodeMCU Lolin

A etapa final do projeto é fazer com que todos os dados contidos no ESP sejam enviados a um broker MQTT. Como já citado, havia muitas opções para isto. Neste trabalho testamos apenas o servidor Blynk, cujas bibliotecas e principais detalhes serão tratados em sequência.

Cabe lembrar que há diversos códigos de configuração e mapeamento de redes disponíveis para o ESP e o Arduino. Códigos para identificar Endereços, MAC, redes disponíveis, etc., que estão disponíveis em fóruns online. Alguns deles são os WiFiScan e o comando *macAdress* da biblioteca <ESP8266WiFi.h>. Algumas dessas informações foram úteis para testes diversos.

3.4.1. Usando o Blynk.

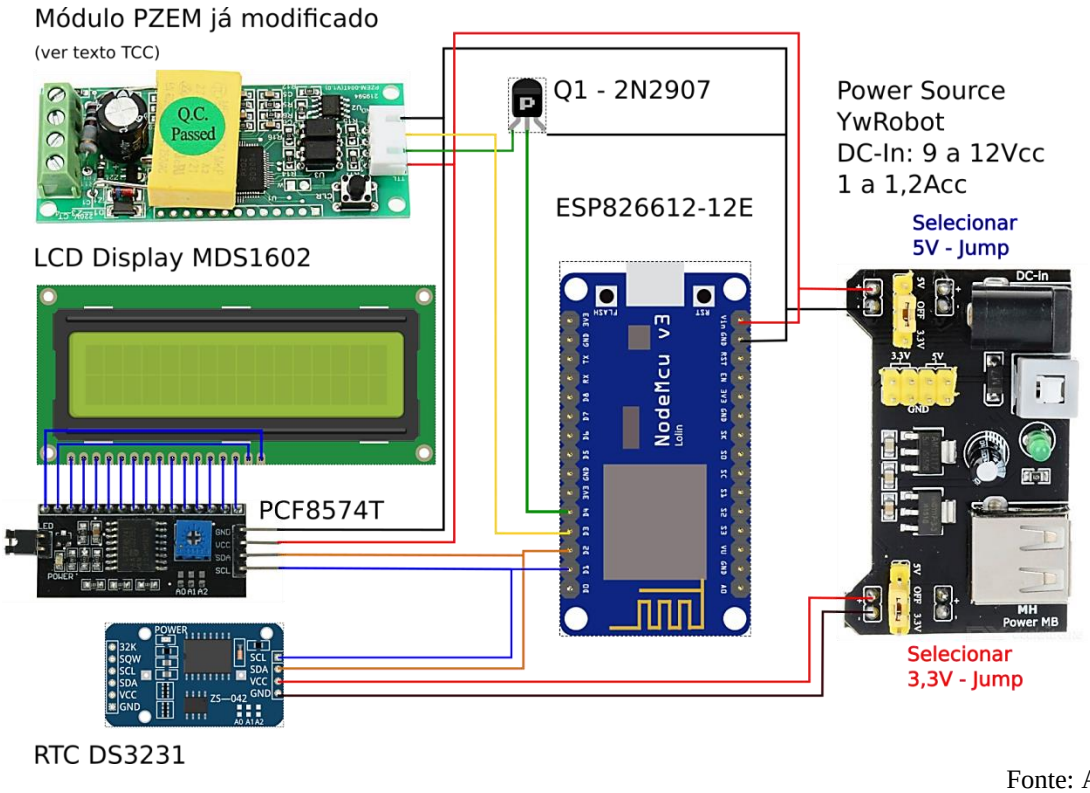
O teste do aplicativo Blynk consistiu no registro de usuário e download*:* (1) dos arquivos de configuração (um pacote compactado de diversos arquivos) para IDE Arduino e (2) do aplicativo para o *smartphone* (Android). Logo a seguir, foi realizada a configuração do aplicativo e teste do código gravado no ESP (ver Anexo E).

Uma das muitas facilidades do Blynk é possuir já bibliotecas diversas para arquiteturas distintas (ESP, Arduino, etc.). Uma dessas bibliotecas é a do serviço de NTP (já citado). Assim, se poderia utilizar para corrigir ou atualizar o relógio do RTC quando fosse necessário.

3.5. Diagrama de Fiação Completo

A fim de facilitar a reprodução do projeto, o diagrama de conexões físicas (diagrama de fiação) definitivo é apresentado na Figura 24 a seguir:

Figura 24 – Diagrama de fiação do protótipo finalizado.



Fonte: Autor.

Atente para que o módulo RTC deve ser alimentado pela linha de 3,3Vcc (separadamente da linha de 5,0Vcc) e que ao menos um dos GND do ESP seja comum a todos os módulos (tanto ao usar a alimentação via USB quanto pela fonte externa). O módulo regulador de tensão YwRobot possui GND comum a todos os pinos (tanto no barramento 5,0Vcc quando no de 3,3Vcc).

Outra observação é no uso de outro modelo de display LCD, pois o MDS tem os pinos 15 e 16 antes dos pinos 1 em diante. Já para outros modelos isso pode diferir. Em todo caso, é preciso consultar o datasheet de todos os módulos para eventuais mudanças.

4. RESULTADOS

O resultado final do trabalho tem por base os registros visuais e análise dos valores reportados, bem como do comportamento do protótipo durante os períodos de ensaios. Assim será possível abstrair questões como funcionalidade, coerência de leituras, viabilidade do projeto, possíveis aplicações e melhorias, e se os objetivos foram satisfeitos.

4.1. Investimentos Dedicados ao Projeto.

A Tabela 8 lista os valores que foram destinados para execução do protótipo.

Tabela 8: Valores Investidos no Protótipo.

Módulo ou Componente (Local Compra)	Valor (R$)
PZEM004T (China)	24,00
NodeMCU Lolin (Mercado Livre)	29,90
Módulo Alimentação Protoboard (China)	12,10
Módulo RTC (China)	3,90
Módulo SD Card (China)	1,80
Display LCD MGS1602 (China)	4,90
Módulo PCF8574T (Mercado Livre)	5,40
Roteador Multilaser N150 (Brasil)	60,00
Fonte Alimentação 15Vcc/ 1A	18,90
TOTAL INVESTIDO	**160,90**

Fonte: Autor.

Considerando outros custos, pode se aproximar que o investimento necessário foi algo em torno de R$ 200,00 (duzentos reais). Em comparação, os medidores da Peacefair chineses não baixam de R$ 40,00 (módulos completos, com caixinha de displays de 7 segmentos). E medidores similares, para barramento DIN, tão pouco baixam de R$ 80,00 (lojas chinesas).

A guisa de comparação, alguns medidores homologados pelo Inmetro vendidos em lojas brasileiras:

- Landis Gyr Plus: E23A, R$ 170,00;
- Lansen: Lumen 2MC, R$ 158,00.

Nenhum deles com qualquer outra função além de registrar o consumo.

Embora não se tenha estimado com precisão um produto comercial poderia sair ainda por menor valor de venda. Alguns circuitos integrados (SoC) dedicados às medições de energia elétrica nos mesmos níveis deste trabalho tem custos a partir de R$ 3,00:

- Texas: μC MSP430AFE, R$ 10,00;

- Analog Device: ADE9153B, R$ 4,00;
- ST: STPM32, R$ 3,00;
- SDICMicro: SD3004, R$ 10,00;
- Cirrus Logic: CS5463, R$ 5,50.

Note que há uma significativa diferença de valores em comparação ao módulo PZEM. Assim, estima-se que outros módulos e recursos poderão também ter reduções significativas ao se optar por um sistema próprio em detrimento de módulos prontos.

Levando em consideração que o roteador não faz parte do projeto, além das outras considerações já apresentadas, e que nos valores estão incluídos frete e lucro dos vendedores, estima-se que é possível ter um protótipo com todas as funções aqui incluídas em torno de R$ 100,00. Estimativa esta que não será comprovada aqui.

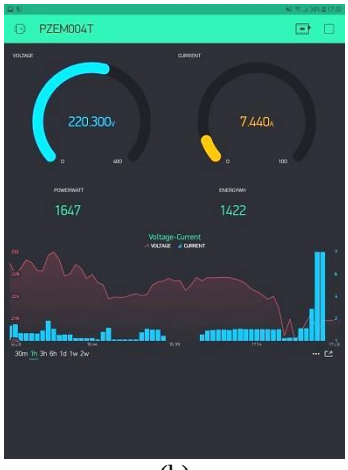
A título de informação, para aquisição de um módulo de desenvolvimento para medidor comercial da Cirrus Logic, nesta data, o valor fica em torno de R$ 1100,00. Módulos desse tipo permitem não só acesso a uma plataforma de desenvolvimento como de testes e ensaios nos SoC desenvolvidos pelas empresas.

4.2. Captura de Tela dos Testes de Medição Usando Aplicativo Blynk

A sequência de capturas de tela da Figura 25 mostra a interface criada no Blynk para o projeto:

Figura 25 – Interface do projeto configurada no Blynk rodando no smartphone. Em (a) primeiras configurações de testes, em (b) a aparência atual.

(a)



(b)

Fonte: Autor.

Em (a) o exemplo de como a tela ficou inicialmente configurada durante os testes de comunicação com os pinos virtuais (V0, V1, V2 e V3)[32]. Em (b) como fica durante as leituras em tempo real quando foi testado em conjunto com o analisador de qualidade de energia (Fluke). Os valores de tensão foram configurados para serem mostrados em linhas cheias em vermelho, já os valores de corrente em barras azuis. Para melhor visualização o widget não mostra os valores proporcionais uns aos outros.

Cada *widget* corresponde à leitura de um pino ou variáveis (configurados no código do ESP).

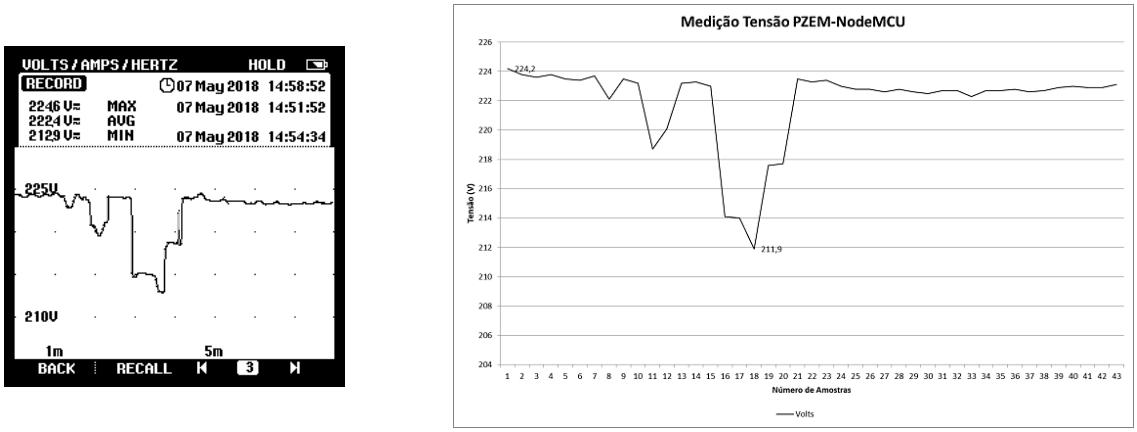
4.3. Teste Inicial de Precisão das Leituras do PZEM004T com Analisador de Qualidade de Energia 43B Fluke.

É importante determinar o quão precisos são os valores de tensão e corrente retornados pelo PZEM, especialmente em um ambiente residencial real na presença de cargas não lineares.

Para uma análise primária foi utilizado o Analisador de Qualidade de Energia 43B da fabricante Fluke para comparar os valores do PZEM (armazenados via código no SD Card) com os lidos por um instrumento calibrado e certificado.

Uma janela de amostras de oito minutos aproximadamente foi coletada usando o analisador e o protótipo simultaneamente (embora não sincronizados). As curvas obtidas podem ser vistas na Figura 26:

Figura 26 – Curvas comparativas de tensão: captura de tela do analisador da Fluke (a) em comparação valores plotados com base nos dados salvos no cartão SD (do protótipo) (b).



(a) (b)

Fonte: Autor.

[32] Lembrando que os pinos virtuais são uma característica do Blynk, cuja função é agir como variáveis para armazenar os dados (valores) a serem transmitidos. Ver *Virtual Pins*: http://docs.blynk.cc/#blynk-main-operations-virtual-pins

O analisador foi configurado para:

- Medindo: Volt/Amperes/Hertz;
- Probe 1: Test Leads;
- Probe 2: 10mV/A.

Os relógios estavam configurados para data e hora locais.

A taxa de amostragem é diferente entre o analisador e o protótipo, por conta dos diversos valores excluídos no tratamento das leituras e do tempo necessário para rodar as rotinas do código. O SD Card recebe uma *string* completa de dados a cada sete (7) a dez (10) segundos, o que faz com que nem a quantidade de pontos e nem a coincidência destes com o gráfico gerado no Fluke correspondam.

O ambiente de controle foi uma residência (Fluke e protótipo ligados à entrada do padrão de medição) onde em tempo real diversas cargas (aparelhos) estavam sendo monitorados dentro da janela de tempo de oito minutos.

Os valores máximos e mínimos de tensão registrados:

- Máximos: 224,6 V_{AC} (Fluke) e 224,2 V_{AC} (protótipo).
- Mínimos: 212,9 V_{AC} (Fluke) e 211,9 V_{AC} (protótipo).

O que mostra que os valores devolvidos pelo protótipo indicam confiabilidade na aplicação em uma carga real.

No entanto esta é apenas uma comparação qualitativa e simples, e outra análise foi realizada.

Cabe informar que foi neste estágio de desenvolvimento que se percebeu que os valores de data e hora estavam sendo gravados incorretamente no cartão SD.

4.4. Problemas de Comunicação na Integração dos Módulos.

Quando este trabalho já estava avançando para sua conclusão descobriu-se que algum tipo de conflito impedia o correto funcionamento do barramento I2C (que não possui relação alguma com SPI) quando o cartão SD e sua biblioteca estavam em uso. Não se sabe até aqui o real motivo disto deste erro.

Basicamente, o ESP não conseguia ler ou escrever nem data e hora (no RTC) e nem as mensagens (no LCD). Após compilar o código, e enquanto conectado ao sistema (computador com IDE que fez a compilação) debug até apresentava data e horas corretamente, mas ao resetar ou reiniciar o sistema fora dessa conexão as datas gravadas no SD card simplesmente tornavam-se as datas gravadas previamente (default) do RTC (1/1/1970 ou algo parecido, conforme o modelo de RTC usado).

Diante desse problema haviam duas alternativas, no contexto que segue:

1. Usar o LCD display junto ao RTC físico, com recurso de logging de valores através do widget “SuperCart” (Blynk), ou;
2. Usar o SD Card perdendo a opção de ter o LCD e o RTC (apenas atualização via NTP, Blynk).

A primeira opção permite ainda a atualização do relógio RTC (via NTP, provido por recursos do próprio Blynk), embora o logging de dados possa ter um decréscimo em qualidade (menos pontos coletados) do que o visto com uso de SD Card. Essa opção se dá inserindo o widget SuperChart do Blynk, e configurando-o para mostrar graficamente as variáveis das quais se deseja o logging de eventos.

A segunda opção mantém a sequência de gravação dos dados no SD Card, mas também deixa o usuário sem informações visuais e é ainda muito mais dependente de uma conexão (presente ou estável) com a internet.

Em certo momento do projeto foi notado que após desconectar da porta USB o ESP perdia comunicação com a memória EEPROM do RTC, não mais mostrando a data e hora gravadas inicialmente. Várias análises foram realizadas, desde conferir ligações, níveis dos sinais, alimentação dos módulos, bibliotecas diferentes, condição da bateria, sem sucesso.

Nesse estágio partiu-se para a opção de ao menos usar um relógio NTP (Network Time Protocol), o que foi desenvolvido.

Logo a seguir, ao usar o LCD display, o mesmo problema ocorria, ainda que o RTC não estivesse presente. Após inicializar, o ESP perdia a comunicação com o módulo LCD (que também faria uso da comunicação I2C). O código de scanner de endereços foi novamente rodado antes disso.

Notou-se, então, que o problema não estava no barramento I2C (RTC e LCD), mas sim no SPI. De alguma forma, a biblioteca SPI disponível para Arduino não conciliava (ao que parece) múltiplos barramentos ao mesmo tempo.

É provável que os modos de comunicação dos barramentos do ESP não tenham a mesma estrutura de controle de um Arduino, o que poderia causar esse problema (*pipeline*[33], ou algo que precisará ser descoberto em outra oportunidade). Outra suspeita recai que de alguma maneira, a biblioteca do PZEM seja também causadora desse problema.

---

[33] “A **segmentação de instruções** (em inglês, ***pipeline***) é uma técnica *hardware* que permite que a CPU realize a busca de uma ou mais instruções além da próxima a ser executada. Estas instruções são colocadas em uma fila de memória dentro do processador (CPU) onde aguardam o momento de serem executadas: assim que uma instrução termina o primeiro estágio e parte para o segundo, a próxima instrução já ocupa o primeiro estágio.” Fonte: Wikipedia, https://pt.wikipedia.org/wiki/Pipeline_(hardware)

Ao contrário dos módulos embarcados com Arduino, não se encontrou informações, até o momento, de um modo de ativar e encerrar essas comunicações dentro do hardware do ESP (permitindo multitarefa e testando possíveis motivos de conflitos).

Assim, para a conclusão deste trabalho optou-se pela primeira opção.

Também o uso do recurso de memória SPIFFS do ESP não pareceu uma boa opção, visto que por se tratar de gravações a uma frequência relativamente alta (entre 6 e 10 gravações por minuto) iria reduzir em muito a vida útil dos setores (e provavelmente da própria) memória flash.

Há de se considerar que muitos dados fossem gravados em modo volátil, por modo matricial ou similar, e depois de um tempo transferidos para a memória flash. Ainda assim a abordagem não pareceu muito promissora, sendo que o ideal seria investir numa memória EEPROM externa (algo que não será abordado neste momento, conforme discutido a seguir).

O Blynk tem num dos widgets a função logging de eventos. Isso foi testado e é funcional igualmente; e apesar de inicialmente não se saber nada sobre sua acuricidade ainda assim pareceu servir aos propósitos básicos deste trabalho.

Funciona da seguinte forma: com o aplicativo do Blynk em modo "*on*", com ou sem conexão ao ESP, ao clicar no menu "..." do widget "*SuperChart*", surgirá a opção "*Export to CSV*" ou "*Erase Data*". Basta clicar na primeira opção e um logging dos eventos sendo medidos até aquele momento será transmitido ao e-mail cadastrado. Mais detalhes sobre os resultados desta abordagem serão dados nas próximas seções.

4.5. Ensaios em Ambiente Real com o Protótipo *versus* Fluke 43B.

Para este ensaio foi utilizado a mesma carga anterior (uma residência unifamiliar). Os instrumentos foram então ligados ao padrão de medição do domicílio permitindo analisar todo o histórico de consumo durante quatro horas ininterruptas, estando os usuários em suas tarefas normais. Tomou-se o cuidado de também utilizar instrumentação homologada (em paralelo) na coleta de dados de tensão e corrente elétrica.

O resumo do ensaio (qualitativo e quantitativo) será analisado a seguir, a começar pela Tabela 9, que descreve as características da medição:

Tabela 9: Descrição Característica do Ensaio Utilizando o Protótipo e o Analisador Fluke.

Descrição	Fluke 43B		Protótipo ESP	
	Tensão	Corrente	Tensão	Corrente
Média das Leituras	224,39 V	3,74 A	224,07 V	3,89 A
Desvio Padrão das Leituras	6,04	5,32	5,54	3,33
Valor Mínimo Lido	208,30 V	0,60 A	207,75 V	0,62 A
Valor Máximo Lido	233,50 V	25,97 A	233,22 V	25,31 A
Pontos da Amostra	240		356	
Pontos Usados para Plotar Gráficos	240		255	
Data e Hora do Início da Leitura	13/06/2018 – 16h03min		13/06/2018 – 16h03min	
Horário do Encerramento do Ensaio	20h18min		21h59min	
Período para Cada Tomada de Média (aproximado)	64s		60s	

Fonte: Autor.

Note que independentemente do modo como os dados foram tratados internamente, tanto no analisador ou no protótipo ou pelo servidor de logging do Blynk, percebeu-se que os períodos de registro de leituras são levemente diferentes entre o Blynk e o Fluke. Essa constatação, por si só, impediu um comparativo mais preciso sobre a qualidade do erro presente nas médias.

A falta de sincronia e os períodos distintos não permitem que os mesmos dados de entrada de um período coincidam entre si nos respectivos registros (Blynk e Fluke). A falta de um valor relevante, seja no início ou fim do período compromete toda a média reportada.

Ainda assim, dos valores entre máximos e mínimos da Tabela 9, se pode extrair os seguintes erros de amplitude de medição (logging) entre o Fluke e o Blynk:

- Tensão mínima: 0,55 V_{AC};
- Tensão máxima: 0,28V_{AC};
- Corrente mínima: 0,02 A_{AC};
- Corrente máxima: 0,66 $A_{AC.}$

Outro ponto negativo para a validação é que, o analisador Fluke leva considerável vantagem ao não depender de perdas de dados de pacotes ou conexão com a internet. Algo a que o sistema utilizando servidor em nuvem pode estar exposto.

4.5.1. Tratando o Registro dos Eventos em Ambos Sistemas com Planilhas.

Percebeu-se, também, que o arquivo .csv gerado pelo Fluke não marca data e hora de cada registro, mas sim o tempo, em segundos. Isso foi percebido apenas depois de usar o instrumento e, portanto, o *start* de ambos os sistemas teve uma diferença de alguns segundos (estima-se 30 segundos), diminuindo ainda mais a sincronia.

O analisador foi programado para parar os registros após quatro horas (240 pontos de registro), já o protótipo permaneceu ligado um pouco mais. Aliado ao que já foi dito, este é o motivo de ao invés de usar apenas 240 pontos de logging (de cada sistema) foram utilizados 15 pontos a mais do sistema compreendido pelo protótipo. Senão, uma parte que era visível ao final (nos gráficos gerados pelos dados do analisador) não seria visível no respectivo gráfico do protótipo.

Tanto o arquivo .csv gerado pelo analisador quanto o gerado pelo servidor do Blynk possuem layout e modos diferentes de apresentação dos valores. Ambos arquivos abriram facilmente numa planilha eletrônica[34] de dados, mas tiveram, cada, suas peculiaridades a serem ajustadas para se conseguir extrair as informações.

O arquivo gerado pelo analisador é mais amigável, mas apenas contém a hora de *start* de gravação, e ainda sem os segundos, de modo que para obter a hora final precisaram-se converter os segundos finais da leitura para um formato de hora (via equações de planilha).

Já o arquivo enviado pelo Blynk (ao e-mail cadastrado) contava o tempo no formato *epoch time*[35] (*Unix time*[36]) e o valor da coluna estava em milissegundos (que precisou ser convertida em segundos para aplicar a conversão). O Blynk gera um arquivo para cada variável mostrada no *widget* "*SuperChart*", sendo assim, dois arquivos foram enviados ao e-mail e foram ambos copiados para uma planilha única.

4.5.2. Análise Final dos Resultados.

Após o ajuste dos dados e análises destes, os arquivos foram unidos em uma única planilha e os gráficos comparados. Além dos dados já observados na Tabela 9, a partir de agora serão mostrados também os gráficos de tensão e corrente elétricas coletados por ambos os sistemas (o analisador e o protótipo).

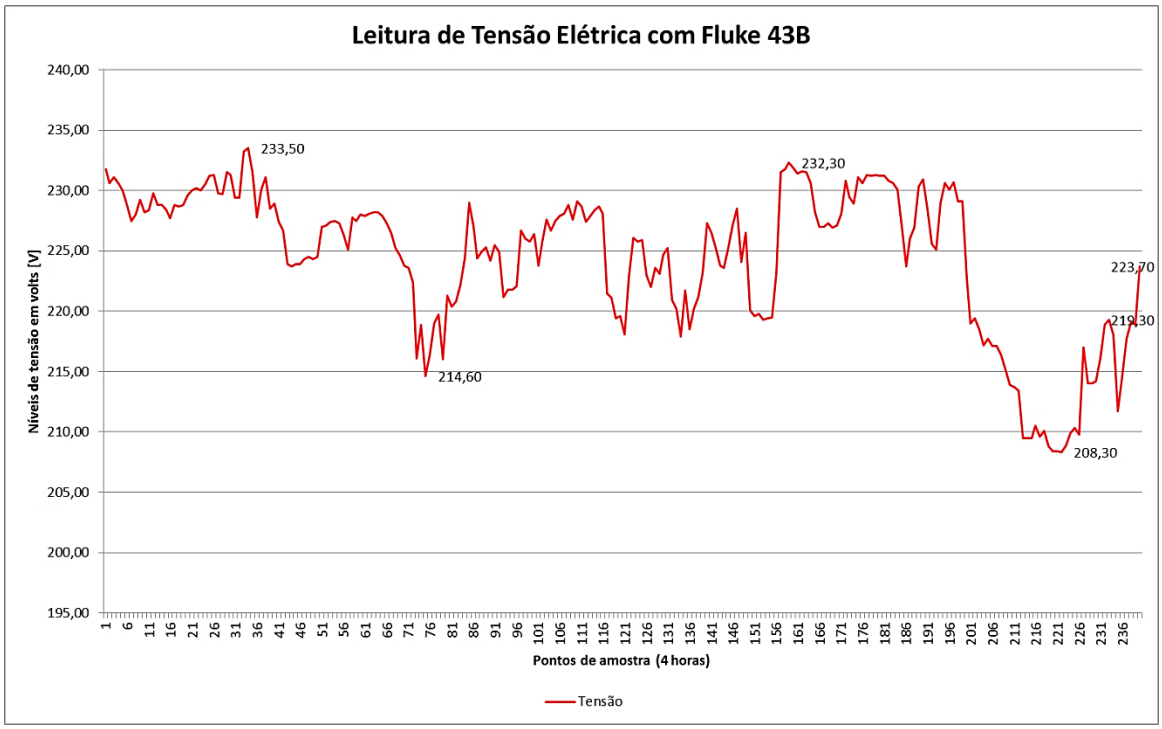
Mesmo que a precisão de validação desejada não tenha sido obtida neste momento, ainda que não desqualificasse o projeto como um todo, nem seus resultados, uma tentativa de análise de erros entre os valores registrados foi organizada. Um resumo dessa parte pode ser visto no Anexo K, deste trabalho. Alguns complicadores nesta etapa também são citados naquela seção.

Nas Figuras 27 e 28 os valores de logging de tensão gerados:

---

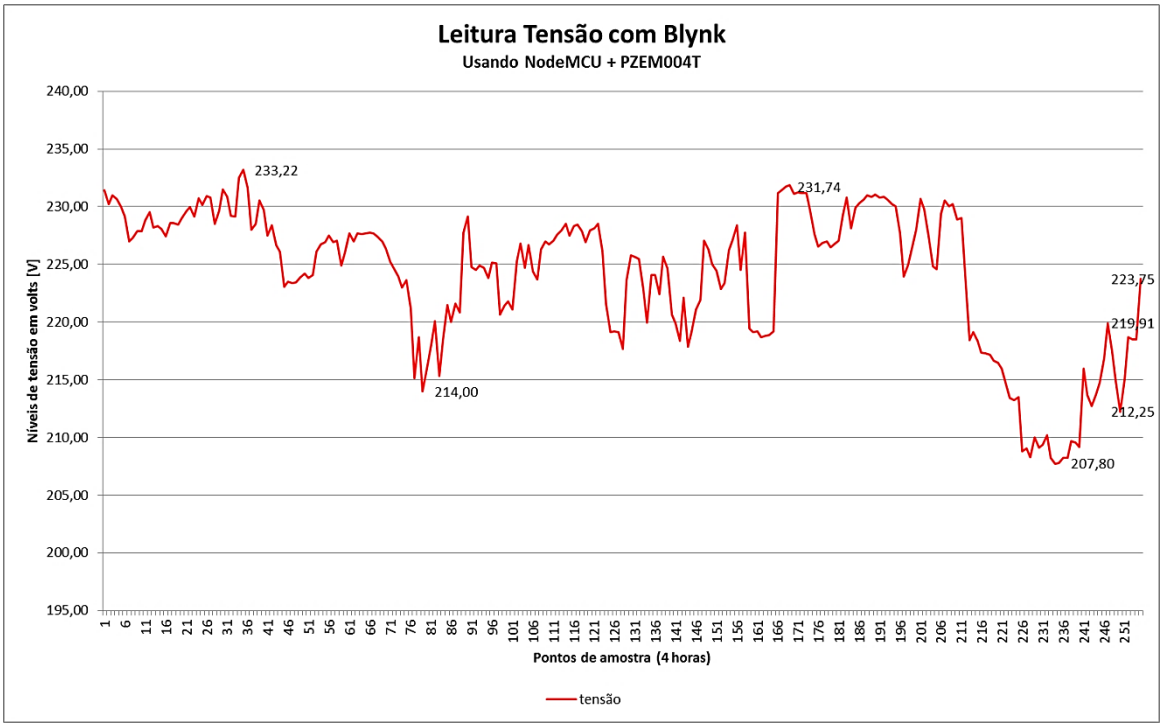
[34] A planilha completa, com os dados já extraídos e harmonizados também está disponível no repositório GitHub, junto ao restante do material deste trabalho (*ver* nota de rodapé 33). Informações sobre conversões de data e hora foram estudadas em tutoriais disponíveis na internet.
[35] *Cf.* Epoch Time: https://www.epochconverter.com/ .
[36] *Cf.* Unix Timestamp: https://www.unixtimestamp.com/ .

Figura 27 – Curva de tensão gerada com dados Fluke 43B.



Fonte: Autor.

Figura 28 – Curva de tensão gerada com dados protótipo PZEM/ NodeMCU/ Blynk.

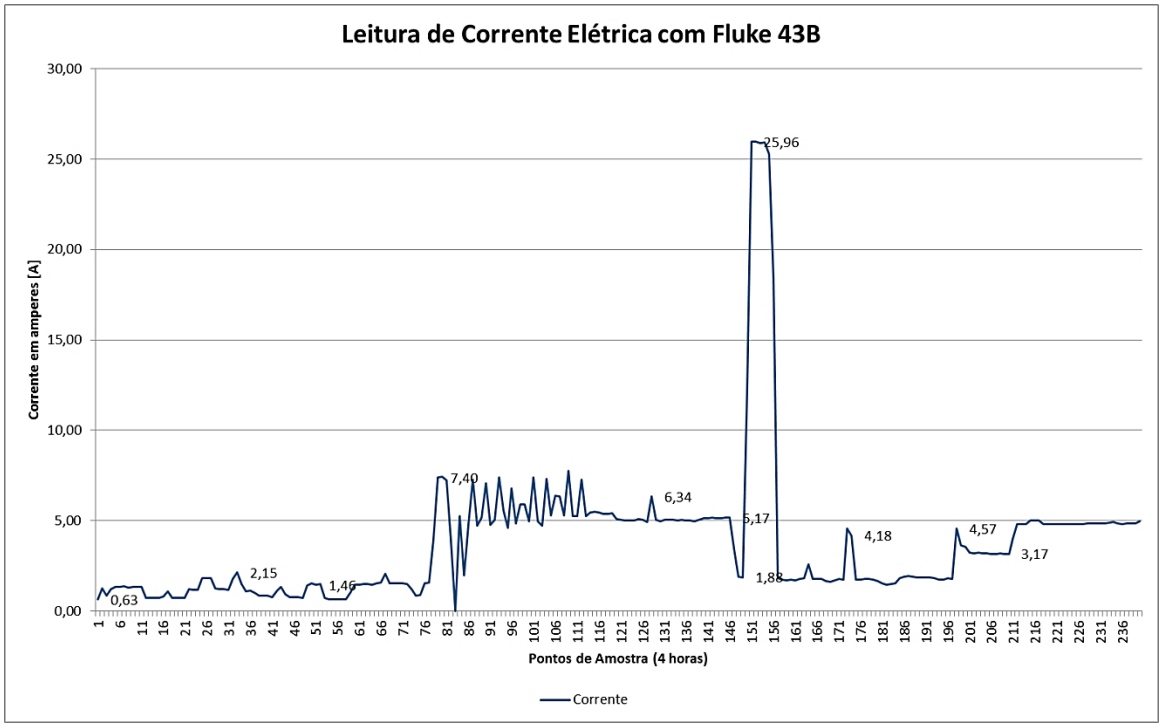


Fonte: Autor.

Conforme já percebido as curvas de tensão obtidas pelo protótipo mostraram-se muito eficientes para aplicação, ainda mais levando-se em conta a disparidade do potencial esperado entre ambos os sistemas de medição utilizados.

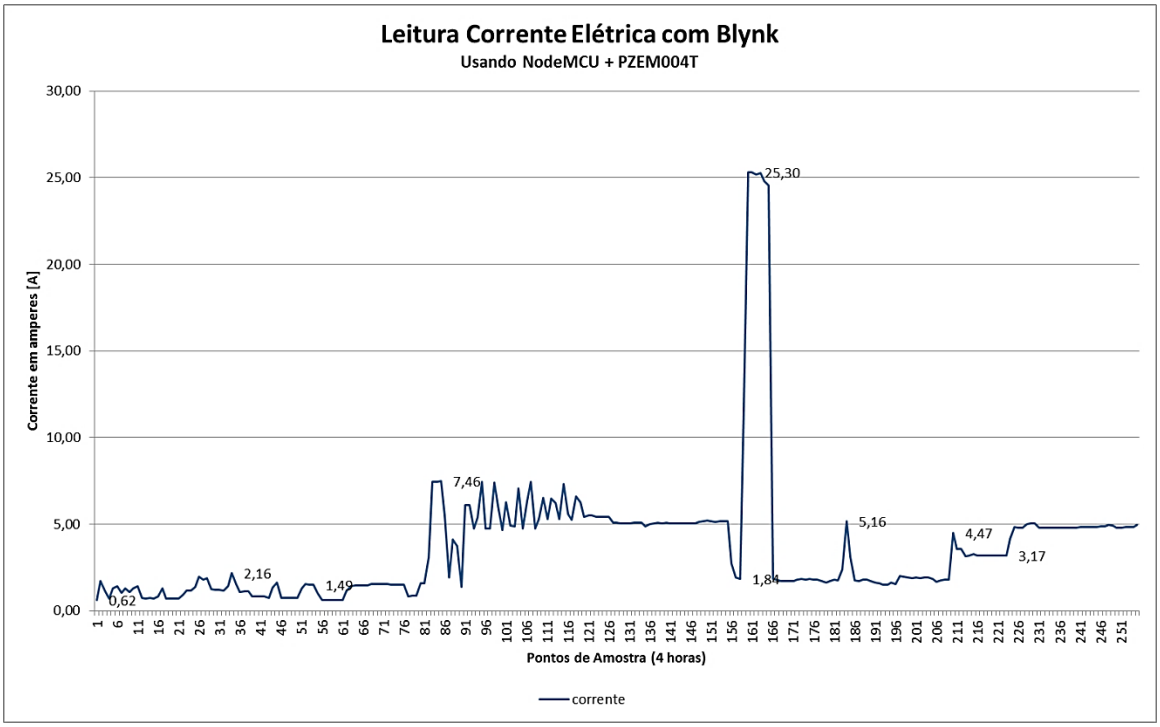
Nas Figuras 29 e 30 serão mostrados os resultados de logging de corrente elétrica, mostrados nos gráficos gerados:

Figura 29 – Curva de corrente elétrica gerada com dados Fluke 43B.



Fonte: Autor.

Figura 30 – Curva de corrente elétrica gerada com dados protótipo PZEM/ NodeMCU/ Blynk.



Fonte: Autor.

Observe que alguns detalhes de valores de corrente não foram adquiridos pelo sistema composto pelo protótipo.

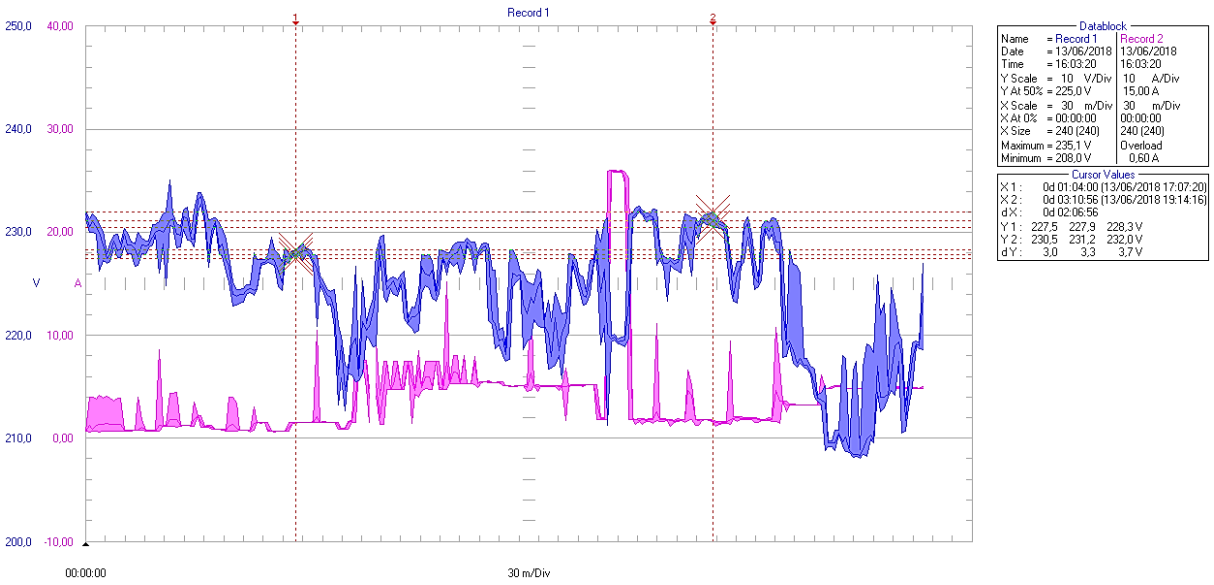
Para medição de corrente o Fluke utiliza um TC tipo alicate com range de 0,5A a 40,0A (com saída de 10mV/A) ou 5,0A a 400,0A (com saída de 1mV/A). A acuricidade indicada pelo fabricante é de 0,015A para um range até 40,0A ou 0,04A para um range de 400,0A.

Levando-se em consideração que um analisador de qualidade de energia tem aplicação não só superior, mas também diferente da proposta desse trabalho, e que ainda assim os valores registrados pelo protótipo são coerentes e aceitáveis diante de um instrumento calibrado e certificado, entende-se que o sistema concebido é satisfatório.

Em tempo, o arquivo .csv do analisador fornece três colunas de valores de medição de cada ponteira (tensão e corrente), constando de valor mínimo, médio e máximo ocorridos no intervalo de cada logging. Para plotar os gráficos, apenas os valores médios foram utilizados e analisados.

Um resumo do que foi citado acima pode ser visualizado na Figura 31 a seguir:

Figura 31 – Tela final de logging gerada pelo analisador Fluke 43B.



Fonte: Autor.

Note que as áreas sombreadas em torno da linha de média consistem justamente dos valores citados como mínimos e máximos para cada registro.

4.6. Recomendações Futuras.

Este projeto pode ser facilmente estendido para:

- Desenvolver o próprio servidor (broker) e API (web ou aplicativo);
- Adicionar funcionalidades ou modificar recursos, testando outras aplicações;
- Estudar e aplicar a funcionalidade de estimar valores de faturamento para informação ao usuário, inclusive podendo automatizar o processo sincronizando com a data de leitura pela concessionária de energia elétrica;

- Desenvolver o próprio circuito de medição, com base em algum SoC de baixo custo;
- Aperfeiçoar a transmissão de dados por meio de uma rede de longo alcance (como LoRa);
- Testar conceitos de aplicação de medições bidirecionais (necessárias em cogeração, por exemplo) e/ ou tarifa branca, inclusive verificando se o sistema poderia auxiliar na criação de curvas de carga;
- Fazer um estudo sobre as exigências técnicas de homologação e certificação de medidores de energia elétrica no Brasil;
- Fazer um estudo sobre criação ou modificação de bibliotecas (nativas para AVR) para serem usadas no ESP;
- Modificar a plataforma existente para uso com outros tipos de sensores, como uma estação meteorológica, por exemplo;
- Estabelecer uma comunicação Internet via 3G/4G, ou algo que inclua GSM.

5. CONCLUSÕES

Este é um trabalho que envolve um grande conjunto de partes interligadas, não havendo muito aprofundamento em cada bloco. É tipo de abordagem que exige, para sua conclusão satisfatória, disciplina, organização, foco e boa dose de pesquisa até obter-se alguma certeza de entendimento ou rumo correto. Especificamente foi também necessário conhecimento de relações interpessoais em fóruns e comunidades e operar tradutores de texto.

Os sistemas embarcados e internet das coisas é um tema da atualidade e ignorar suas aplicações é no mínimo imprudente para aqueles envolvidos direta ou indiretamente (setores administrativos, técnicos, de gestão, usuários, etc.), como exemplificado neste trabalho: a medição do consumo de energia.

A análise tanto preliminar como final dos resultados durante os testes do protótipo (em comparativo com instrumentação homologada) reportou valores com precisão e taxa de aquisição satisfatórias, mostrando-se o protótipo funcional e aplicável. Tais resultados não são nada desprezíveis levando-se em conta a diferença de investimento (um Fluke 43B foi encontrado por algo em torno de R$ 12000,00, no exterior). Para aplicações em campo, que não exijam resultados homologados, o protótipo cumpre bem sua função de dar transparência ao consumo dos usuários.

É também necessário dizer que frente a um produto comercial o protótipo cumpre não apenas com fins didáticos e de aprendizado, mas pode ter aplicação prática satisfatória de baixo custo e acessível. Mesmo assim, sua produção precisaria levar em conta refazer ou substituir diversos módulos por seus equivalentes CIs em versões mais baratas montados com placas específicas. Há visível sobra de recursos de hardware no sistema apresentado que poderiam ser simplificados para a aplicação ou mesmo ampliados para outras funções.

É preciso destacar que a aplicação dos módulos que compõem este trabalho não precisa limitar-se à aplicação mostrada. A plataforma com conexão em nuvem está pronta para embarcar diversas outras funcionalidades e monitorar outras grandezas. Pode ainda ser aplicada para fins de amostragens de qualidade de energia ou padrões de consumo compondo uma rede uma rede de dados geograficamente instalada.

Algumas modificações de código podem permitir que o sistema não seja tão dependente de suas funções, para que ao não inicializar uma delas isso não seja prejuízo às demais (como nos modos multitarefa).

Mas como já dito, este é um projeto didático, para ser ponto de partida para novos projetos ou melhorias, portanto o código cumpriu seu papel assim como está. Ainda assim, outras considerações se fazem necessárias.

O módulo da Peacefair não é o tipo de placa classificado como “hardware aberto”, carecendo inclusive de disponibilidade de documentação, mas não se pode negar que é uma alternativa viável de baixo custo e que também inspira a criação de módulos compactos similares. O mesmo ocorre com as APIs mostradas, são livres, mas não abertas.

Percebeu-se ainda que é algo no mínimo interessante desenvolver uma placa de código aberto que pudesse cumprir a mesma função de medição de grandezas elétricas, mas um pouco mais completo que o utilizado aqui.

Ao se desenvolver sistemas como o proposto neste trabalho, que envolvam o uso de plataformas e dispositivos de origens diversas é preciso considerar a ocorrência de alguma incompatibilidade, que foi exatamente o que aconteceu. Talvez um estudo futuro mais aprofundado descubra os motivos de não se conseguir usar o barramento SPI junto ao módulo PZEM sem comprometer o barramento I2C.

Neste cenário, é visível a necessidade de aprendizado em diversas áreas para que quem desejar se aventurar nesse mercado tenha condições de atender e resolver diversas demandas, onde a experiência e conhecimento agilizam a obtenção de respostas.

Neste estudo também se teve contato com diversas plataformas e tipo de serviços baseados em nuvem e internet das coisas, grande maioria são citados. Algo que mostra que, especialmente nas áreas de tecnologia, é um conteúdo que deve estar presente. Na verdade, o profissional consiste daquele que se mantém sempre atualizado em seu ramo, naquilo que lhe é não só contemporâneo como também tendência ou futuro, para que assim consiga disputar melhores espaços e oportunidades.

Uma dessas linhas de estudos envolvem questões de segurança (invasões de sistemas por terceiros) que sempre precisa ser considerado. Embora neste protótipo não foi incluída nenhuma funcionalidade de telecomando, ainda assim, o acesso às portas de comunicação (e por consequência acesso à rede e outros dispositivos) é sempre algo a se analisar.

Ao final, exceto pelo módulo PZEM (que necessitou as modificações citadas) todos os demais módulos funcionaram perfeitamente e sem erros quando testados isoladamente no ESP. Este é um cuidado que se precisa ter ao se integrar diversos módulos funcionais em Arduino dentro de uma arquitetura ESP, eles podem funcionar isoladamente, mas dependendo de alguns fatores não funcionar juntos (no ESP). Este trabalho, por exemplo, precisou receber algumas modificações por uma situação deste tipo.

Ao encerrar, considera-se que o trabalho atingiu seus objetivos, e os resultados foram proveitosos e coincidentes com o que era esperado, podendo ainda ser replicado, aperfeiçoado ou ampliado.

REFERÊNCIAS

ABESCO. **Desperdício de energia gera perdas de R$ 12,6 bilhões**. Associação Brasileira das Empresas de Serviços de Conservação de Energia, 2015. Disponível em: <http://www.abesco.com.br/pt/novidade/desperdicio-de-energia-gera-perdas-de-r-126-bilhoes>. Acesso em: 22 abr. 2018.

ALECRIN, Emerson. **Software livre, código aberto e software gratuito: as diferenças**. InfoWester, 2013. Disponível em: <https://www.infowester.com/freexopen.php>. Acesso em 23 abr. 2018.

ANEEL. **Energia no Brasil e no Mundo**. Agência Nacional de Energia Elétrica, Atlas de Energia Elétrica do Brasil, Box 2, Parte I], 2002. Disponível em: <http://www2.aneel.gov.br/arquivos/pdf/atlas_par1_cap2.pdf>. Acesso em 22 abr. 2018.

_____. **Tarifa branca é nova opção para os consumidores a partir de 2018.** Acessoria de imprensa, 2017. Disponível em: <http://www.aneel.gov.br/sala-de-imprensa-exibicao/-/asset_publisher/XGPXSqdMFHrE/content/tarifa-branca-e-nova-opcao-para-os-consumidores-a-partir-de-2018/656877?inheritRedirect=false>. Acesso em 30 mai. 2018.

ASSOCIAÇÃO BRASILEIRA DE NORMAR TÉCNICAS. NBR 14519: **Medidores eletrônicos de energia elétrica (estáticos) – Especificação.** Rio de Janeiro, 2000.

ASSOCIAÇÃO BRASILEIRA DE NORMAR TÉCNICAS. NBR 14522: **Intercâmbio de informações para sistemas de medição de energia elétrica - Padronização**
Rio de Janeiro, 2000.

CUNHA, Joana. "**Desperdício consome 10% da energia elétrica no país, diz associação"**. Jornal Folha Mercado, 2015. Disponível em: <http://www1.folha.uol.com.br/mercado/2015/02/1586778-desperdicio-consome-10-da-energia-eletrica-no-pais-diz-associacao.shtml>. Acesso em 22 abr. 2018.

EPE. **Consumo anual de energia elétrica por classe (nacional)** - 1995-2017. Empresa de Pesquisa Energética, 2017. Disponível em: <http://www.epe.gov.br/pt/publicacoes-dados-abertos/publicacoes/Consumo-Anual-de-Energia-Eletrica-por-classe-nacional>. Acesso em 22 abr. 2018.

ESPRESSIF. **ESP8266EX Datasheet**. Version 5.8, 2018 [English]. Disponível em: <https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf>. Acesso em 24 abr. 2018.

EVANS, Dave. **A Internet das Coisas: Como a próxima evolução da Internet está mudando tudo**. White Papper da Cisco, 2011. Disponível em: <https://www.cisco.com/c/dam/global/pt_br/assets/executives/pdf/internet_of_things_iot_ibsg_0411final.pdf>. Acesso em 23 abr. 2018.

GUADANIN, Cláudia. **De 2011 a 2015, Brasil desperdiçou energia suficiente para um ano de consumo**. Jornal Gazeta do Povo, 2016. Disponível em: <http://www.gazetadopovo.com.br/economia/energia-e-sustentabilidade/de-2011-a-2015-brasil-desperdicou-energia-suficiente-para-um-ano-de-consumo-8bnk42j8ibd25of8e9yiwb5h1>. Acesso em: 22 abr. 2018.

HAYASHI, Ricardo. **Automação por meio de medidores inteligentes de energia elétrica permite adequação das distribuidoras à Tarifa Branca**. Editora Brasil Energia [online], 2018. Disponível em: <https://brasilenergia.editorabrasilenergia.com.br/artigo-automacao-por-meio-de-medidores-inteligentes-de-energia-eletrica-permite-adequacao-das-distribuidoras-a-tarifa-branca>. Acesso em 22 abr. 2018.

JAVED, Adeel. **Criando Projetos com Arduino para a Internet das Coisas**. São Paulo: Novatec, 2017.

MANCINI, Mônica. **Internet das Coisas: História, Conceitos, Aplicações e Desafios.** USP, 2017. Disponível em: <https://pmisp.org.br/documents/acervo-arquivos/241-internet-das-coisas-historia-conceitos-aplicacoes-e-desafios/file>. Acesso em 24 abr. 2018.

MME. **Resenha Energética Brasileira**. Ministério de Minas e Energia, 2015. Disponível em: <http://www.mme.gov.br/documents/1138787/1732840/Resenha+Energ%C3%A9tica+-+Brasil+2015.pdf/4e6b9a34-6b2e-48fa-9ef8-dc7008470bf2>. Acesso em 22 abr. 2018

NUWER, Rachel. "**Por que empresas de energia adoram medidores inteligentes?"**. Revista Exame [online], 2015. Disponível em: <https://exame.abril.com.br/tecnologia/por-que-empresas-de-energia-adoram-medidores-inteligentes>. Acesso em 22 abr. 2018.

OLIVEIRA, Sérgio de. **Internet das Coisas com ESP8266, Arduino e Raspbery Pi**. São Paulo: Novatec, 2017.

SDIC. **Energy Measurement SOC SD3004**. SDICMICRO, 2012. Disponível em: <http://www.sdicmicro.com/DataSheet/SD3004%20datasheet%20v0.2c.pdf>. Acesso em 22 abr. 2018.

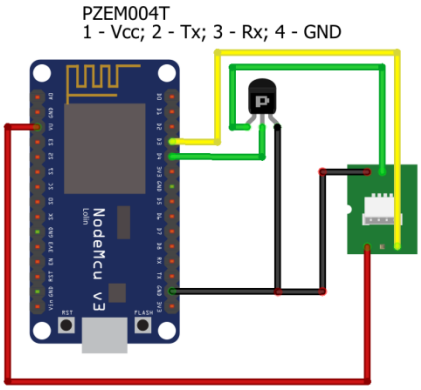
SOKOLOV, Oleg. **Arduino communication library for Peacefair PZEM-004T Energy Monitor**. GitHub, 2018. Disponível em: <https://github.com/olehs/PZEM004T>. Acesso em 27 abr. 2018.

THOMSEN, Adilson. “**Vamos falar de Open Hardware?**”. FilipiFlop, 2014. Disponível em: <https://www.filipeflop.com/blog/open-hardware-livre/>. Acesso em 23 abr. 2018.

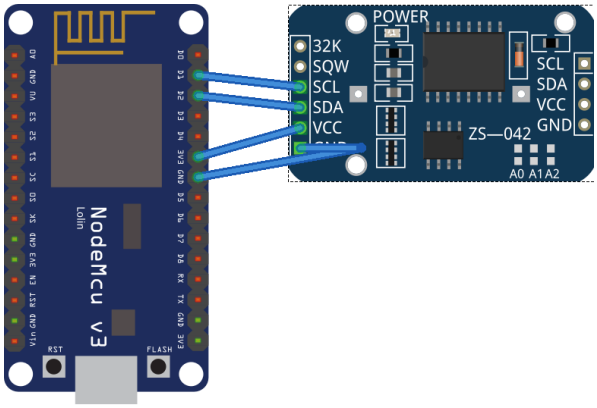
WEG. **Medidores inteligentes de energia WEG no maior projeto de Smart Grid do Brasil**. WEG Energia, 2016. Disponível em: <https://www.weg.net/institutional/BR/pt/news/produtos-e-solucoes/medidores-inteligentes-de-energia-weg-no-maior-projeto-de-smart-grid-do-brasil>. Acesso em 22 abr. 2018.

ANEXOS

A. — Diagramas de Fiação por Blocos



Fiação PZEM004T



Fiação RTC DS3231

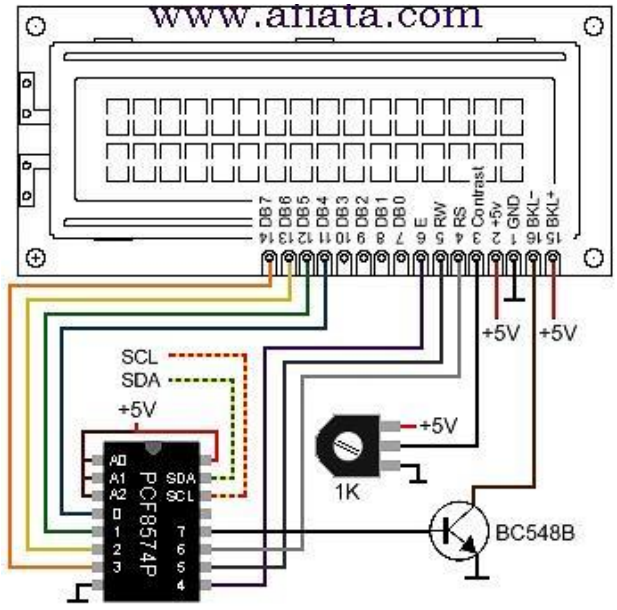
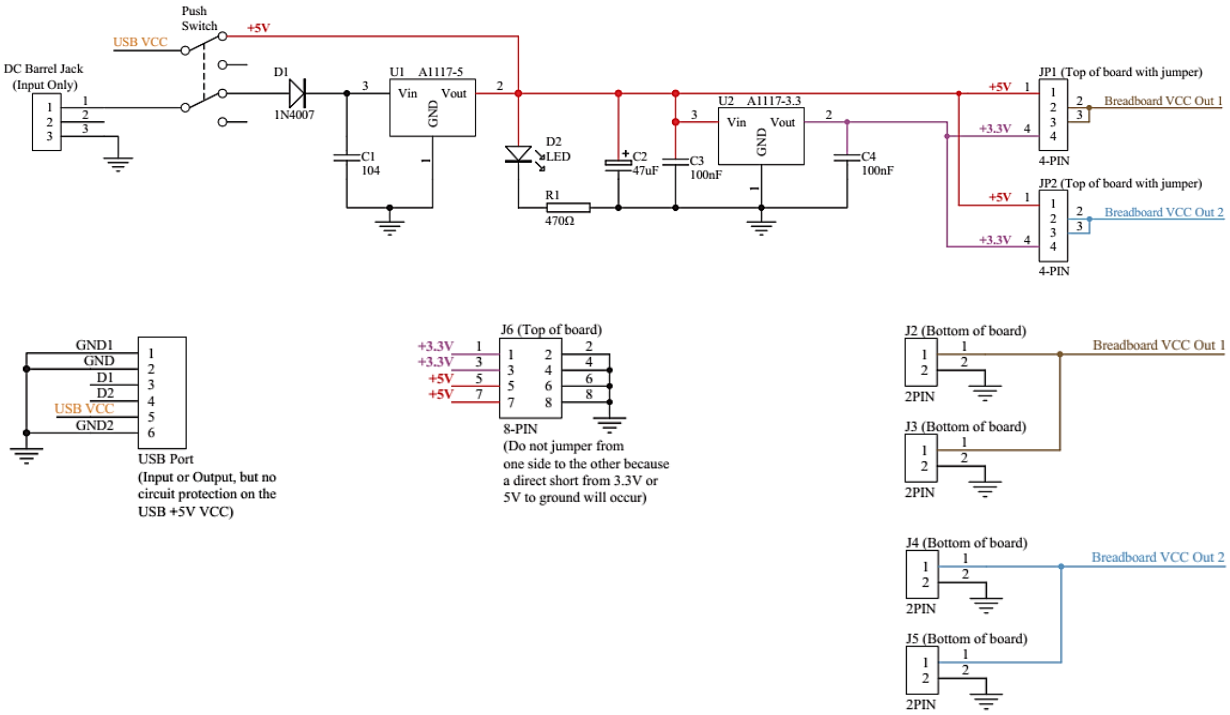


Diagrama Esquemático para o CI PCF8574 com o LCD display MDS1602

B. — Diagrama Esquemático do Módulo de Alimentação YWROBOT



C. — Código Fonte Completo Comentado

```
/*CABEÇALHO DE IDENTIFICAÇÃO AUTORAL
 * Autor: Gerson E O Sena
 * Projeto: Medidor Datalogger de Energia Elétrica OpenSource com NodeMCU Lolin
 * Data: 2018
 * Local: Unipampa, Alegrete, RS, Brasil
 * Contribuições: Comunidade OpenSource, desenvolvedores, autores de artigos WEB.
 * REFERÊNCIAS: Comentadas ao final do Código
 * ------------------------------------------------------------------------------------------
-------------------------
 * IMPORTANTE: Esta versão apresentou problemas com o barramento SPI (SD Card),
 * inviabilizando a gravações in locco de arquivos de logging.
 * Não se sabe se são problemas de conflitos de hardware ou bliblioteca (nativa Arduino),
 * sendo que outras questões foram testadas diversas vezes, sem sucesso. Desconfiava-se
 * da tensão de alimentação, inclusive, mas como o módulo tinha a opção de 3,3V e
 * mesmo assim o erro persistiu, este bloco (SD Card) foi descontinuado.
 * Com isso o relógio RTC voltou a funcionar assim como foi possível adicionar o LCD,
 * ambos não funcionais na presenta do módulo SD Card.
 *
 * A solução de relógio NTP do próprio Blynk foi testada enquanto isso (julgava-se ser
problema
 * no RTC) e mostrou-se funcional. Nesta versão de código ela será usada para ajustar o
 * RTC quando este perder a memória por algum motivo (carga da bateria e estando
desligado).
 * Mas tanto o RTC quando o NTP são blocos que podem ser removidos sem prejuízo algum das
 * funções pensadas para este sistema
 */

/*=========================================================================================
==========================
 * INCLUSÃO DAS BIBLIOTECAS NECESSÁRIAS
 *
 */
#include <PZEM004T.h>               //Biblioteca exclusiva do PZEM004T escrita para
Arduino
//#include <SD.h>                     //Biblioteca de comunicação com o cartão SD nativa
#include <Wire.h>                   //Biblioteca de comunicação I2C nativa
#include "RTClib.h"                 //Biblioteca de comunidação RTC que funcionou com
Arduino/ ESP
#include <ESP8266WiFi.h>            //Biblioteca nativa do ESP para conexões WiFi
#include <BlynkSimpleEsp8266.h>     //Biblioteca de configuração do MQTT Blynk sobre
ESP8266
#include <LiquidCrystal_PCF8574.h>  //Bilblioteca usada no módulo PCF8574T

#include <TimeLib.h>
#include <WidgetRTC.h>

/* Comentar esta linha para desabilitar debug SerialMonitor e economia de espaço memória
*/
#define BLYNK_PRINT Serial

/*Comunicação I2C: RTC, PCF8574.
* Ver http://playground.arduino.cc/Main/I2cScanner
* Deve-se usar o código acima para identificar os endereço I2C
 * D1 (05) = SCL
 * D2 (04) = SDA
 * Endereços scaneados:
 * 0x3F = LCD Display
 * 0x57 = EEPROM RTC DS3231
 * 0x68 = PCF8574

* Comunicação SPI: SD Card
 * D8 (15) = CS
 * D7 (13) = MOSI
 * D6 (12) = SCK
 * D5 (14) = MISO
 */


/*=========================================================================================
==========================
 * DEFINIÇÃO DAS VARIÁVEIS GLOBAIS
 */
//RTC
//RTC_Millis rtc;               // Cria o objeto rtc - lib testada e funcional, precisa
```

```
mudar código (ver exemplos)
RTC_DS3231 rtc;               // Cria o objeto rtc associado ao chip DS3231
String loggEvent, date, hr = "";


//LCD
LiquidCrystal_PCF8574 lcd(0x3F); //Seta o endereço do LCD para 0x3F
int error = 0;


//SD


//PZEM
PZEM004T pzem(0, 2);          // D3 (GPIO00) = RX (recebe TX do PZEM), D4 (GPIO02) = TX
(recebe RX do Pzem pelo Q 2N2907)
IPAddress ip(192,168,1,1);    // IP de comunicação do PZEM. Inicializa a comunicação com
o módulo
bool pzemrdy = false;         // Variável tipo bolean para identificar se a comunicação
com o ESP teve sucesso
float v, i, p, e, maior, menor, x, y, z = 0; // Variáveis para armazenar e tratar
leituras do PZEM


//BLYNK
/* Inserir o "Auth Token" obtido no Blynk App. Vá até "Project Settings" (ícone
parafuso),
 *  entre "" a seguir.
 *  É enviado ao e-mail cadastrado ou pode ser copiado para área transferência
SmartPhone.
 *  WiFi: Fazer o NodeMCU Lolin conectado na Rede,
 *  SSID e Senha devem ser escritos entre respectivas "" a seguir.
*/
char auth[] = "e05a2d366f984181a1ad6306d59a90d1";

char ssid[] = "REDETESTE";
char pass[] = "SENHARED";

float voltage_blynk = 0;
float current_blynk = 0;
float power_blynk = 0;
float energy_blynk = 0;

unsigned long lastMillis = 0; // Armazenar a base de tempo para cada transmissão o
CloudServer Blynk

//NTP
BlynkTimer timer;
WidgetRTC rtc2;

//OTHERS
unsigned long timerun, prun = 0; // Usado para contar o tempo completo de cada ciclo de
loop

/*=============================================================================================
============================
 * SETUP, INICIALIZAÇÃO DAS VARIÁVEIS, FUNÇÕES E BIBLIOTECAS
 */

void setup() {
timerun = millis();

Serial.begin(115200);

//LCD
Serial.println("LCD...");

  while (!Serial);
    Serial.println("Checking LCD...");
    Wire.begin(4,5);
    Wire.beginTransmission(0x3F);   //Testa o endereço de comunicação
    error = Wire.endTransmission(); //Armazena o retorno do teste
      Serial.print("LCD Status (erro): ");      //Imprime o
      Serial.print(error);                      //erro retornado

      if (error == 0) {                 //Se o erro = 0, ok
```

```
151         lcd.setBacklight(255);                  //Liga backlight do LCD máximo brilho
152         Serial.println(": LCD found.");
153           lcd.clear();
154           lcd.print("LCD found.");
155         delay(1000);
156         }
157         else {                          //Se o erro = (1 a 4), problema de comunicação
158           Serial.println(": LCD not found.");
159           }
160
161 lcd.begin(16, 2);     //Inicializa o LCD 16x2
162
163 //RTC
164 if (!rtc.begin()) {
165     Serial.println("RTC NÃO encontrado!");
166     lcd.clear();
167         lcd.print("Not find RTC");
168     while (1);
169   }
170   else {
171     Serial.println("RTC encontrado!");
172     lcd.clear();
173         lcd.print("RTC found!");
174         delay(1000);
175   }
176
177   if (rtc.lostPower()) {
178     Serial.println("RTC perdeu alimentação, vamos ajustar data-hora!");
179     lcd.clear();
180     lcd.setCursor(0,0);
181         lcd.print("RTC pwr LOST");
182         delay(1000);
183     // following line sets the RTC to the date & time this sketch was compiled
184     rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
185     lcd.setCursor(0,1);
186         lcd.print("System Adj...");
187         delay(1000);
188     // This line sets the RTC with an explicit date & time, for example to set
189     // January 21, 2014 at 3am you would call:
190     // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
191     // lcd.setCursor(0,1);
192     //    lcd.print("Manual Adj...");
193     // This line sets the RTC with an NTP date & time:
194     // rtc.adjust(DateTime(year(), month(), (day()), (hour()), minute(), second()));
195     // lcd.setCursor(0,1);
196     //    lcd.print("NTP Time Adj...");
197   }
198
199 //SD
200
201
202 //PZEM
203   while (!pzemrdy) {                            // Testa e força a comunicação com o PZEM
204       Serial.println("Conectando ao PZEM...");
205       pzemrdy = pzem.setAddress(ip);
206         lcd.setBacklight(255);                  //Liga backlight do LCD máximo brilho
207         lcd.setCursor(0,0);                     //Indica posição escrever status
208         lcd.print("Connecting PZEM");           //Imprime status LCD
209         delay(900);                             //Aguarda
210         lcd.clear();                            //Limpa tela LCD
211         lcd.setBacklight(0);                    //Desliga backlight LCD
212       delay(100);
213    }
214
215
216 //BLYNK
217   Serial.println("Iniciando Blink...");
218   lcd.setBacklight(255);                        //Liga backlight do LCD máximo brilho
219     lcd.setCursor(0,0);                           //Indica posição escrever status
220         lcd.print("Blynk begin...");           //Imprime Blynk tentando conexão...
221   Blynk.begin(auth, ssid, pass);                  //Faz a conexão com o Blynk
222   Serial.println("Blink Inicializado: OK");
223     lcd.setCursor(0,1);
224         lcd.print("Blynk OK!");                 //Imprime Sucesso na conexão...
225   delay(500);
226   lcd.setBacklight(0);                          //Liga backlight do LCD máximo brilho
```

```
227
228 //NTP
229 timer.setInterval(5000L, clockDisplay);
230
231 }
232
233
234 /*================================================================================
235 ==========================
236  * LAÇO DE EXECUÇÃO DAS ROTINAS
237  */
238 void loop() {
239 //LCD
240
241
242 //RTC
243 DateTime now = rtc.now();
244
245     loggEvent = String(now.year(), DEC) + "/" + String(now.month(), DEC) + "/" +
246 String(now.day(), DEC) + ";"
247     + String(now.hour(), DEC) + ":" + String(now.minute(), DEC) + ":" +
248 String(now.second(), DEC);
249
250 //SD
251
252
253 //PZEM
254
255
256 //BLYNK
257   Blynk.run();
258   Serial.println("Blink run...");
259   //NTP
260   timer.run();
261   Serial.println("timer run...");
262
263
264 //PZEM
265 /* Tratamento de valores espúrios (a lib do PZEM foi escrita para arquitetura AVR, não
266 ESP
267  *  Armazena 3 valores em sequência e prepara para comparação/ limpeza de valores
268 espúrios
269  */
270 //Leitura Tensão para Testar FALTA
271 x = pzem.voltage(ip); y = pzem.voltage(ip); z = pzem.voltage(ip);
272 maior = x; menor = x;
273   /*
274    * LAÇO CONDICIONAL PRINCIPAL
275    */
276   if (x < 0.0 && y < 0.0 && z < 0.0) {      // Laço que identifica e confirma falta de
277 energia x = y = z = -1 do PZEM
278     v = 0.0; i = 0.0; p = 0.0; e = 0.0;     // Zera todos os valores para não 'printar'
279 código '-1'
280     Serial.print(v); Serial.println("V FALTA");
281
282     //RTC
283     date = String(now.year(), DEC) + "/" + String(now.month(), DEC) + "/" +
284 String(now.day(), DEC);
285     hr = String(now.hour(), DEC) + ":" + String(now.minute(), DEC) + ":" +
286 String(now.second(), DEC);
287
288     //LCD
289     lcd.setBacklight(255);
290     lcd.clear();
291       lcd.setCursor(0,0); lcd.print("FALTA_V");
292       lcd.setCursor(8,0); lcd.print(hr);
293       lcd.setCursor(0,1); lcd.print(date);
294
295     //SD
296
297
298     //BLYNK
299       voltage_blynk = 1.00;                   // TENDO DIFICULDADES PARA MOSTRAR VALOR
300 0.0 NO BLYNK...
301       current_blynk = 1.00;
302       power_blynk = 1.00;
```

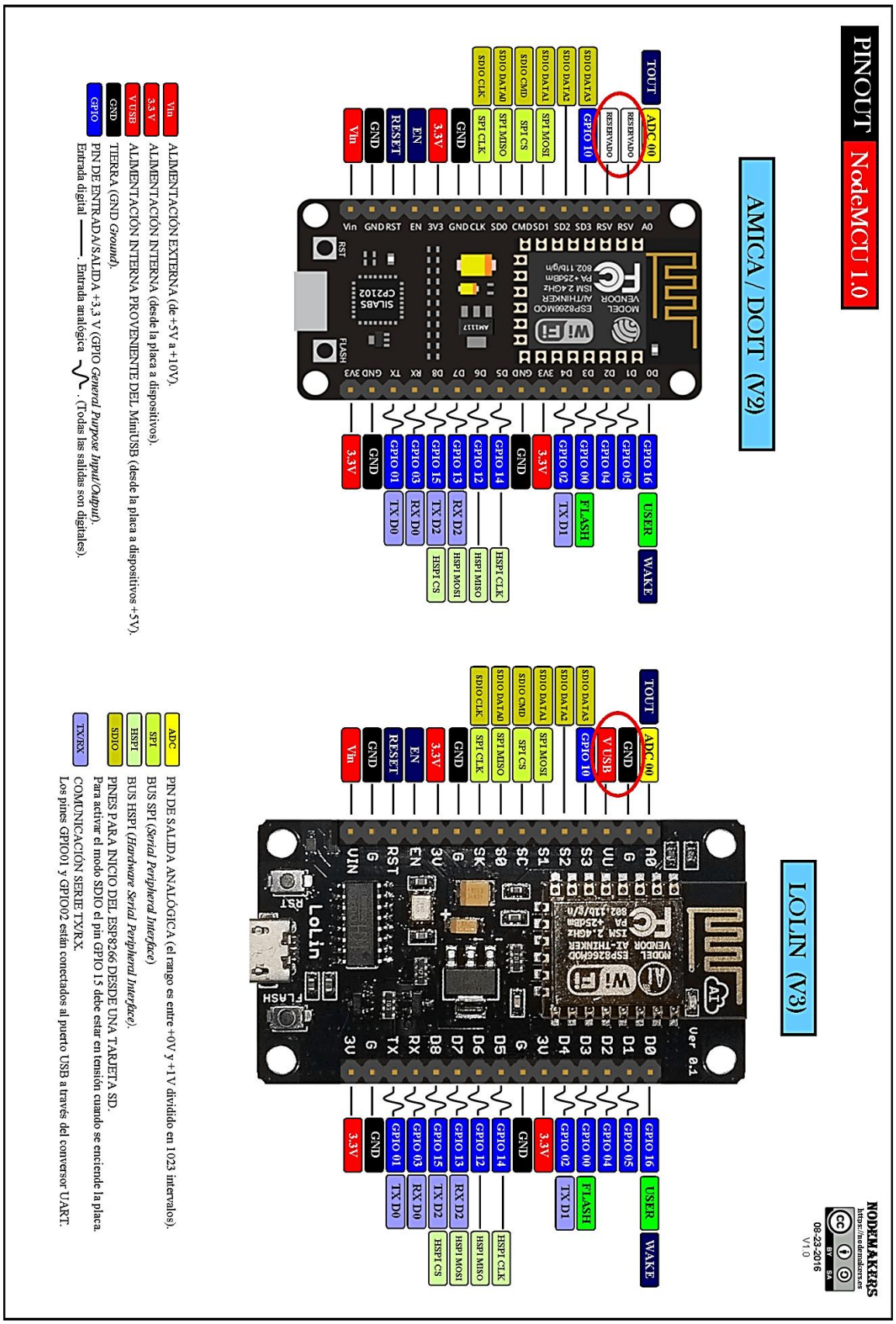
```
303
304     if (millis() - lastMillis > 5000) {       // 06/05/18... testar isso aqui...
305        lastMillis = millis();
306        Blynk.virtualWrite(V0, voltage_blynk); // Envia o valor de tensão para o
307 VirtualPin V0
308        Blynk.virtualWrite(V1, current_blynk); // ... VirtualPin V1
309        Blynk.virtualWrite(V2, power_blynk);   // ... VirtualPin V2
310        Blynk.virtualWrite(V3, energy_blynk);  // ... VirtualPin V3
311     }
312   }
313     else {                                    // Leitura de valores PZEM em Regime (sem
314 FALTA)
315     espurgov(pzem.voltage(ip), pzem.voltage(ip), pzem.voltage(ip));
316     espurgoi(pzem.current(ip), pzem.current(ip), pzem.current(ip));
317     espurgop(pzem.power(ip), pzem.power(ip), pzem.power(ip));
318     espurgoe(pzem.energy(ip), pzem.energy(ip), pzem.energy(ip));
319
320       //RTC Serial Monitor
321       Serial.print(now.year(), DEC); Serial.print('/'); Serial.print(now.month(), DEC);
322 Serial.print('/'); Serial.print(now.day(), DEC);
323       Serial.print(" - ");
323       Serial.print(now.hour(), DEC); Serial.print(':'); Serial.print(now.minute(), DEC);
324 Serial.print(':'); Serial.print(now.second(), DEC);
325       Serial.print("; ");
326
327       //PZEM Serial Monitor
328       Serial.print(v);Serial.print("V; ");
329       Serial.print(i);Serial.print("A; ");
330       Serial.print(p);Serial.print("W; ");
331       Serial.print(e);Serial.println("Wh; ");
332
333       //LCD
334       lcd.setBacklight(255);
335       lcd.clear();
336       lcd.setCursor(0,0);
337       lcd.print(v); lcd.print("V");
338       lcd.setCursor(8,0);
339       lcd.print(i); lcd.print("A");
340       lcd.setCursor(0,1);
341       lcd.print(p); lcd.print("W");
342       lcd.setCursor(7,1);
343       lcd.print(e); lcd.print("Wh");
344
345       //SD
346
347       //BLYNK
348       //Publica (publisher) no servidor a cada 10s (10000 milliseconds). Mudar o valor
349 altera o intervalo.
350       if (millis() - lastMillis > 5000) {
351         lastMillis = millis();
352           Blynk.virtualWrite(V0, voltage_blynk); // Envia o valor de tensão para o
353 VirtualPin V0
354           Blynk.virtualWrite(V1, current_blynk); // ... VirtualPin V1
355           Blynk.virtualWrite(V2, power_blynk);   // ... VirtualPin V2
356           Blynk.virtualWrite(V3, energy_blynk);  // ... VirtualPin V3
357       }
358     }//else
359
360 prun = (millis() - timerun)/1000;
361 Serial.print(prun); Serial.println("s ");
362
363 }//loop
364 /*=============================================================================================
365 ============================
366  * BLOCO DE FUNÇÕES UTILIZADAS
367  * Funções de espurdo dados quebrados
368  */
369 // Leitura e Tratamento dos valores de Tensão
370 void *espurgov(float x, float y, float z){
371   float maior = x; float menor = x;
372       if (y > menor){
373         menor = y;
374         }
375         else {
376           menor = 0.0;
377         }
```

```
        if (y > maior && y >= 0.0){
          maior = y;
          }
          if (z > menor){
            menor = z;
            }
          else {
            menor = 0.0;
            }
            if (z > maior && z >= 0.0){
              maior = z;
              }
      v = maior;
      voltage_blynk = v;
}

// Leitura e Tratamento dos valores de Corrente
void *espurgoi(float x, float y, float z){
  float maior = x; float menor = x;
      if (y > menor){
        menor = y;
        }
        else {
          menor = 0.0;
        }
        if (y > maior && y >= 0.0){
          maior = y;
          }
          if (z > menor){
            menor = z;
            }
          else {
            menor = 0.0;
            }
            if (z > maior && z >= 0.0){
              maior = z;
              }
      i = maior;
      current_blynk = i;
}

// Leitura e Tratamento dos valores de Potência Instantânea
void *espurgop(float x, float y, float z){
  float maior = x; float menor = x;
      if (y > menor){
        menor = y;
        }
        else {
          menor = 0.0;
        }
        if (y > maior && y >= 0.0){
          maior = y;
          }
          if (z > menor){
            menor = z;
            }
          else {
            menor = 0.0;
            }
            if (z > maior && z >= 0.0){
              maior = z;
              }
      p = maior;
      power_blynk = p;
}

// Leitura e Tratamento dos valores de Energia Consumida
void *espurgoe(float x, float y, float z){
  float maior = x; float menor = x;
      if (y > menor){
        menor = y;
        }
        else {
          menor = 0.0;
        }
        if (y > maior && y >= 0.0){
```

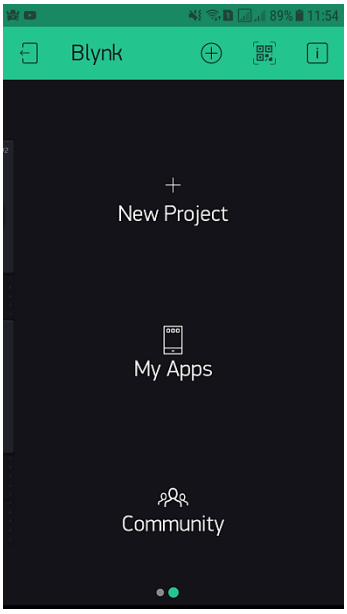
```
          maior = y;
          }
          if (z > menor){
            menor = z;
            }
          else {
            menor = 0.0;
            }
            if (z > maior && z >= 0.0){
              maior = z;
              }
      e = maior;
      energy_blynk = e;
}

void clockDisplay(){
  // You can call hour(), minute(), ... at any time
  // Please see Time library examples for details

  String currentTime = String(hour()) + ":" + minute() + ":" + second();
  String currentDate = String(day()) + " " + month() + " " + year();
  Serial.print("Current time: ");
  Serial.print(currentTime);
  Serial.print(" ");
  Serial.print(currentDate);
  Serial.println();

  // Send time to the App
  Blynk.virtualWrite(V5, currentTime);
  // Send date to the App
  Blynk.virtualWrite(V6, currentDate);
}

/*REFERÊNCIAS
 * Todas as referências foram pontos de partida do projeto. Várias delas
 * não funcionaram a contento precisando consultar diversas outras fontes
 * a fim de verificar os erros de configuração, pinagens, integração de
 * módulos e bibliotecas, conflitos diversos e funcionamento do próprio
 * hardware.
 ************************************************************

 * Blynk: versão 2.20.2, Server: Blynk Cloud
  Download latest Blynk library here:
    https://github.com/blynkkk/blynk-library/releases/latest

  Blynk is a platform with iOS and Android apps to control
  Arduino, Raspberry Pi and the likes over the Internet.
  You can easily build graphic interfaces for all your
  projects by simply dragging and dropping widgets.

    Downloads, docs, tutorials: http://www.blynk.cc
    Sketch generator:           http://examples.blynk.cc
    Blynk community:            http://community.blynk.cc
    Follow us:                  http://www.fb.com/blynkapp
                                http://twitter.com/blynk_app

  Blynk library is licensed under MIT license
  This example code is in public domain.

  Examples blynk virtual data :

https://examples.blynk.cc/?board=ESP8266&shield=ESP8266%20WiFi&example=GettingStarted%2FVirtualPinWrite

          Blynk Github:
          https://github.com/blynkkk/blynk-library/blob/master/examples/Boards_WiFi/ESP8266_Standalone/ESP8266_Standalone.ino

  FULL material for this job (portuguese):
  GITHub: https://github.com/gersonsena/NodeMCULolin_PZEM004t_Meter

 ************************************************************
  * ESP8266: NodeMCU Lolin (v3)
  This example runs directly on ESP8266 chip.

  Note: This requires ESP8266 support package:
```

```
530     https://github.com/esp8266/Arduino
531
532   Please be sure to select the right ESP8266 module
533   in the Tools -> Board menu!
534
535   Change WiFi ssid, pass, and Blynk auth token to run :)
536   Feel free to apply it to any other example. It's simple!
537
538   SoftwareSerial for esp8266:
539   https://github.com/plerup/espsoftwareserial
540
541  ************************************************************
542   * PZEM-004T
543   Complete Tutorial English:
544   http://pdacontrolen.com/meter-pzem-004-esp8266-platform-iot-blynk-app/
545
546   PZEM004T library for olehs:
547   https://github.com/olehs/PZEM004T
548
549  */
```

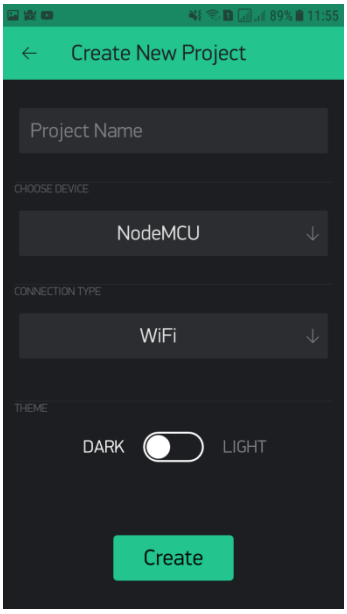
D. — Pinout do NodeMCU Lolin



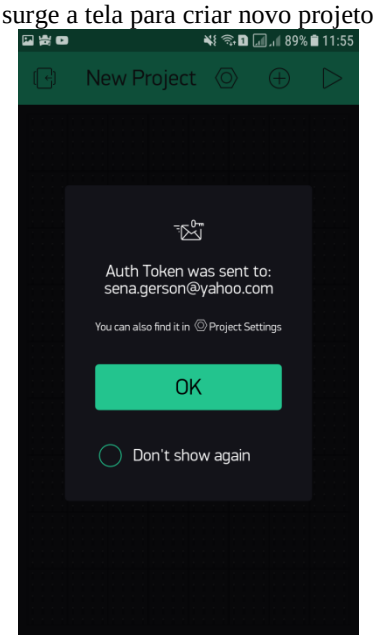
E. — Configuração Inicial Básica do Aplicativo Blynk no Smartphone.



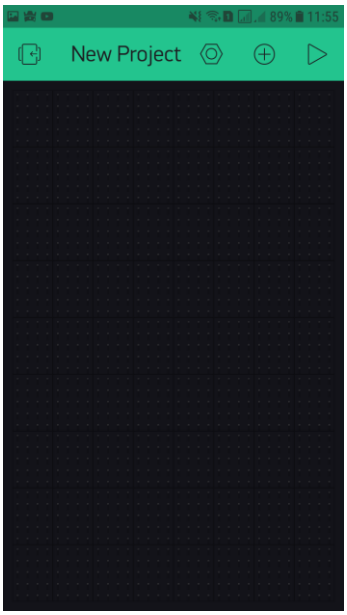
Após fazer acessar o app com usuário e senha criados, surge a tela para criar novo projeto



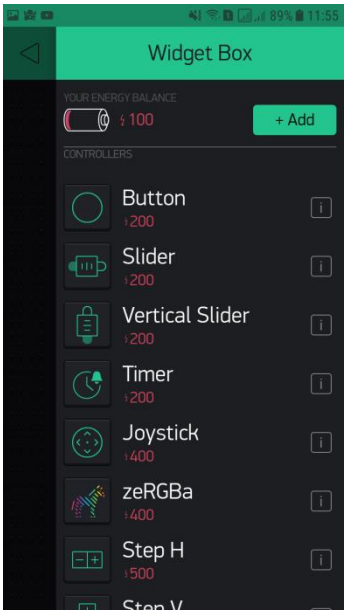
Ao clicar em New Project, a tela que surge é esta.



Um código “token” será gerado e enviado ao e-mail cadastrado. Clique em “OK”. Abra seu e-mail (pode ter ido parar em ‘spam’) e copie o a chave token (vai usar no código inserido no ESP)



Uma área de trabalho em branco será exibida, e no botão + widgets serão mostrados. Após adicionar algum widget este pode ser arrastado pela tela.



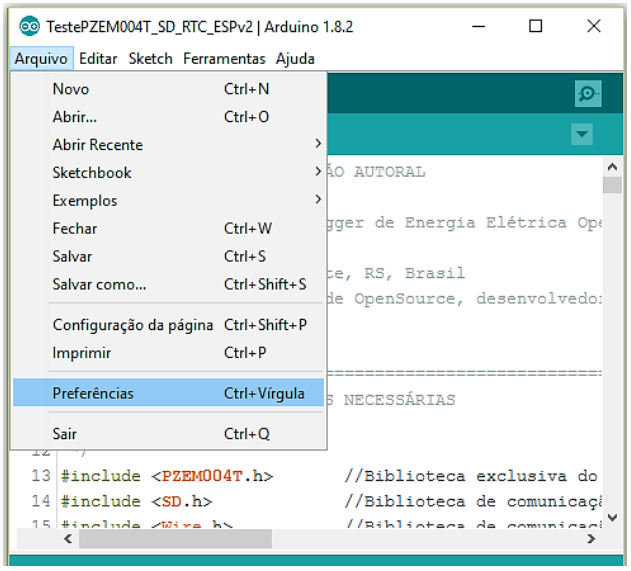
Alguns exemplos de widgets disponíveis. Observe que por já ter criado um projeto, o ícone da bateria logo acima não consta 2000, mas sim apenas 100, pois já usei 1900. Se precisar de mais basta clicar em +Add e comprar.

Aqui um exemplo de como a tela ficou inicialmente configurada durante os testes de comunicação com os pinos virtuais. Cada widget corresponde a leitura de um desses pinos (setados no código do ESP).

F. — Usando o ESP8266 com a IDE Arduino

Nesta seção será mostrado o que é necessário para possibilitar isso, configurando a IDE. A primeira tarefa é abrir a IDE. Isto feito o primeiro passo é o da Figura F1:

Figura F1 – Passo 1: Configuração IDE Arduino.

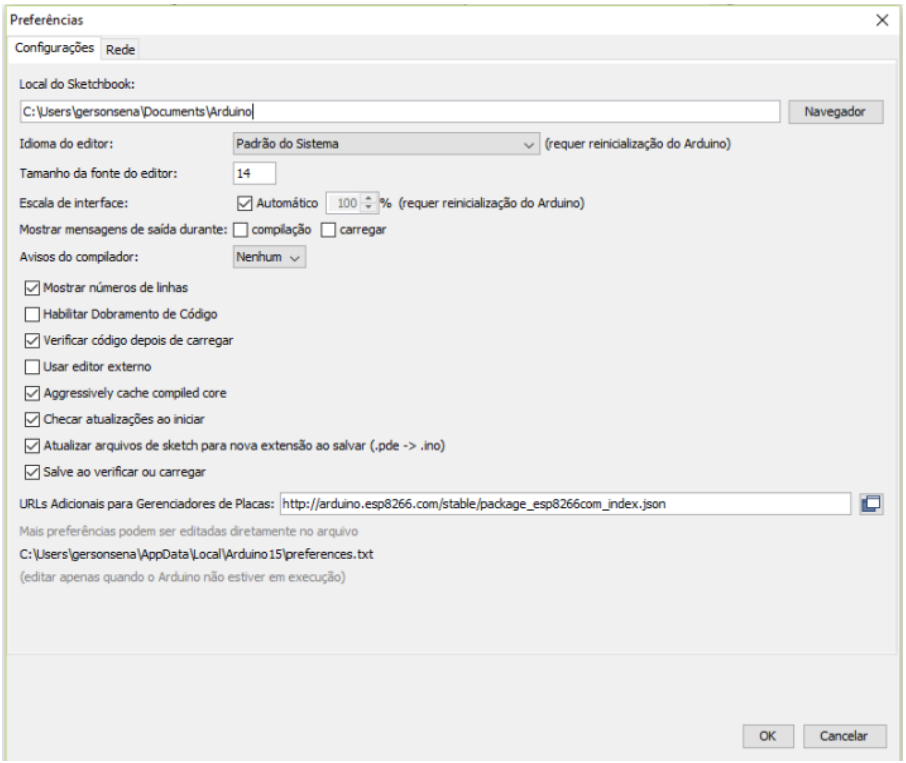


Fonte: Autor.

Na janela da Figura F2 insere-se o endereço[37] do repositório a ser buscado para adição das bibliotecas dos módulos que permitirão à IDE reconhecer e gravar nos modelos de ESP:

[37] Link do repositório: http://arduino.esp8266.com/stable/package_esp8266com_index.json

Figura F2 – Passo 2: Configuração IDE Arduino, repositório.

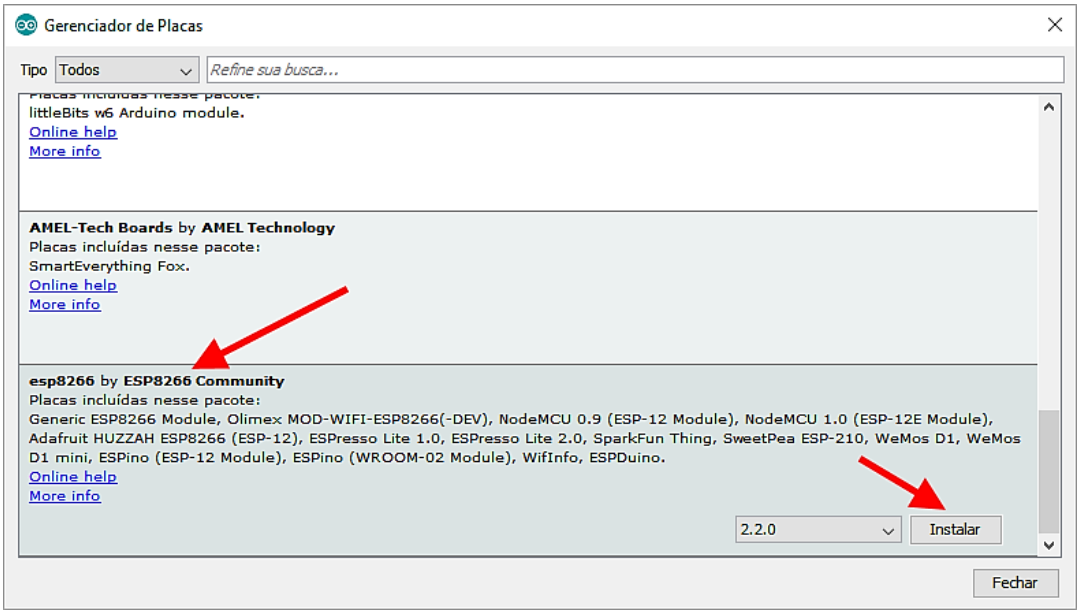


Fonte: Autor.

Basta clicar em “OK”.

É preciso então fazer a instalação da biblioteca que agora surgirá no gerenciador de bibliotecas, como mostrado na Figura F3:

Figura F3 – Passo 3: Configuração IDE Arduino, instalando a biblioteca ESP.



Fonte: Autor.

Agora, basta conectar a placa do ESP numa das portas USB do computador e verificar se ela foi reconhecida procurando no Menu: "*Ferramentas > Placa: ...*", e depois em: "*Ferramentas > Porta: ...*".

Selecionadas tanto a placa como a porta USB corretas, basta agora carregar o sketch desejado na IDE e fazer a compilação para o ESP. Lembrando que todas as bibliotecas listadas pelo código precisam ter sido instaladas previamente.

Não há espaço neste trabalho para ensinar todos os detalhes dessas configurações, mas a Internet tem uma infinidade de publicações ensinando a fazer e resolver essas etapas.

G. — Exemplo do protocolo de comunicação dos comandos AT usados no PZEM004T e Erros de paridade:

Definindo o endereço de comunicação: 192.168.1.1
Enviar comando: B4 C0 A8 01 01 00 1E.
Dados de resposta: A4 00 00 00 00 00 A4.

O exemplo acima ilustra que definir o endereço de comunicação como 192.168.1.1 (o usuário pode definir seu próprio endereço com base em suas preferências e necessidades), enviando comandos e respondendo os dados automaticamente, como mostrado acima. Sendo:

- $B4_{HEX} = 180_{DEC}$;
- $C0_{HEX} = 192_{DEC}$;
- $A8_{HEX} = 168_{DEC}$;
- $01_{HEX} = 001_{DEC}$;
- $01_{HEX} = 001_{DEC}$;
- $00_{HEX} = 000_{DEC}$;
- $21E_{HEX} = 542_{DEC}$;

"Note que a resposta $1E_{HEX} = 030_{DEC}$ ao invés de 21E se deve à capacidade de armazenar apenas dois dígitos por valor (sendo que 21E equivale a soma dos seis pares de dígitos anteriores a ele). Assim o sistema não faz a conversão para conferir o valor, apenas compara se os últimos dois dígitos são os mesmos. Observe também que os dígitos (Dx) são agrupados como D0 D1 D2 D3 D4 D5 = D6" — (*contribuição do autor*).

Os dados são expressos em hexadecimal; o último byte dos dados de envio e de resposta são 1E e A4, pertencendo à soma cumulativa.

Ao enviar comandos: B4 + C0 + A8 + 01 + 01 + 00 = 21E (use o acréscimo hexadecimal), os dados de soma cumulativa são 21E, mas o valor enviado leva apenas os últimos dois bytes "1E" para serem usados os dados de soma cumulativa nos comandos de envio (outros casos em que resulta mais de dois algarismos o processo é o mesmo, apenas os dois últimos);

Dados em resposta: A4 + 00 + 00 + 00 + 00 + 00 = A4 (use a adição hexadecimal), os dados de soma cumulativa são A4, que são os dados de soma cumulativa em resposta. Sendo:

- $A4_{HEX} = 164_{DEC}$;
- $00_{HEX} = 000_{DEC}$;
- $00_{HEX} = 000_{DEC}$;
- $00_{HEX} = 000_{DEC}$;
- $00_{HEX} = 000_{DEC}$;
- $00_{HEX} = 000_{DEC}$;
- $A4_{HEX} = 164_{DEC}$;

A explicação da soma cumulativa está agora concluída, os exemplos de parâmetros a seguir são os mesmos, não há mais nenhuma explicação.

Leitura da tensão em tempo real:

Enviar comando: B0 C0 A8 01 01 00 1A
Dados da resposta: A0 00 E6 02 00 00 88

Os dados de tensão (resposta) são os dígitos "D1D2D3 = 00 E6 02, onde os bits "00 E6" representam o bit inteiro da tensão, e o bit "02" representa o decimal da tensão (a casa decimal é de apenas um dígito). Quando convertidos "00 E6" em decimal o resultado é "230"; e ao converter "02" para decimal o resultado é "2", então o valor atual da tensão é "230.2 V" (o ponto representa a vírgula, neste caso, no sistema brasileiro).

Leitura da corrente em tempo real:

Enviar comando: B1 C0 A8 01 01 00 1B.

Dados de resposta: A1 00 11 20 00 00 D2.

Os dados lidos são os dígitos “D2 D3 = 11 20”, onde o bit “11” representa inteiro da corrente e “20” representam o decimal da corrente (o decimal é dois dígitos). Quando convertido em decimal o 11_{HEX} resulta em 17_{DEC}, e convertendo hexadecimal “20” em decimal resulta em “32”. Então o valor lido da corrente é de “17.32 A”.

Leitura da potência ativa em tempo real:

Enviar comando: B2 C0 A8 01 01 00 1C.

Dados da resposta: A2 08 98 00 00 00 42.

Os dados de potência de resposta são “D1D2 = 0898”, que convertidos “898” em decimal é resulta em “2200”, ou seja, o valor de potência ativa lido é 2200 W.

Leitura do consumo de energia acumulado:

Enviar comando: B3 C0 A8 01 01 00 1D.

Dados da resposta: A3 01 86 9F 00 00 C9.

Os dados de energia de resposta são “D1D2D3 = 01 86 9F”, os valores convertidos de “1869F” para decimal resulta em “99999”, portanto, a energia acumulada é de 99999 Wh (ou 99,999 kWh).

Definindo o alarme para o limitede energia: 20 KW

Enviar comando: B5 C0 A8 01 01 14 33

Dados de resposta: A5 00 00 00 00 00 A5

O dado 14 no comando de envio é o valor do alarme (14 é uma representação de dados hexadecimais, que convertida em decimal é 20). O que você deve observar é que o valor do alarme de energia deste módulo é baseado em unidades KW, o que significa que o valor mínimo do alarme é 1KW, o valor máximo é 22KW.

Set the power alarm threshold:20 KW

Send command: B5 C0 A8 01 01 14 33

Reply data: A5 00 00 00 00 00 A5

Note: 14 in the sending command is the alarm value (14 is a hexadecimal data representation, which converted to decimal is 20). What you should note is the power alarm value of this module is based on KW units, which means the minimum alarm value is 1KW, the maximum value is 22KW.

H. — Extrato Folha de Dados do PZEM004 (v.3): AC digital display Multifunction Meter

Funções:

Medição de parâmetros elétricos (tensão, corrente, potência ativa, energia).

Alarme de sobrecarga (acionaria o modo flash do LED indicador de status – power on – e um *buzzer*; ambos disponíveis no módulo com caixa plástica).

Predefinição de limite para o alarme (pode definir o limite em que o alarme é ativado).

Reset, para zerar o registro de consumo de energia.

Armazenamento de dados na perda de alimentação (registra os dados de consumo de energia antes de desligar).

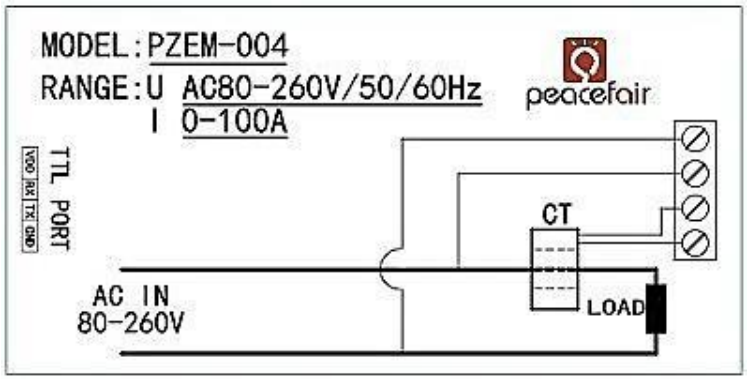
Display digital LED 7 segmentos (tensão, corrente, potência ativa, energia), disponível apenas na versão com caixa plástica.

Comunicação serial (com interface serial TTL embarcada, permitindo comunicação com uma variedade sistemas, permitindo ler e definir parâmetros).

Diagrama de Fiação:

Na Figura H1 a seguir consta o que vem gravado na tampa da caixa plástica do módulo

Figura H1 – Curva de tensão gerada com dados Fluke 43B.



Fonte: Peacefair.

A fiação deste módulo é dividida em duas partes: a fiação do terminal de entrada de teste (tensão e corrente) e a fiação de comunicação serial TTL.

Os valores de tensão e de corrente medidos são da ordem de 1,0 V e 10,0 mA (mínimo que o instrumento consegue capturar). Para valores de potência ativa e energia as leituras mínimas possíveis são de 1,0 W e 1,0 Wh.

Precauções:

Este módulo é adequado para uso interno, por favor, não use ao ar livre.

A carga aplicada não deve exceder a potência nominal.

A ordem de fiação não pode estar errada.

Parâmetros de especificação:

Tensão de funcionamento: 80 ~ 260VAC

Tensão de teste: 80 ~ 260VAC

Potência nominal: 100A / 22000W

Frequência de operação: 45-65Hz

I. — Datasheet Resumido do µC SD3004

SD3004

Energy Measurement SOC

Features

- High precision energy measurement
- Provide RMS voltage and RMS current
- Calculates active power and power factor
- Calculates AC frequency
- High frequency CF pulse for calibration
- Calculates total energy usage over time
- 24 seg × 4 com LCD drivers, can be switched to become I/O ports
- Supports LED driving
- Real time clock, can output second signal
- UART and I^2C interfaces
- 2K*16 bits OTP program memory, support online programming, 128 bytes data memory
- Operating voltage: energy measurement circuit 4.75 – 5.25V, rest of the IC 2.4 – 5.25V

General Description

The SD3004 is an electric energy measurement SOC with built in MCU, energy/voltage/current measurement circuit, LCD/LED display drivers, and UART communication interface. It greatly simplifies circuit designs and reduces production costs for energy meter, metering socket, and similar products.

Ordering Information

LQFP64-10×10-0.5 package

Pin Diagram and Descriptions

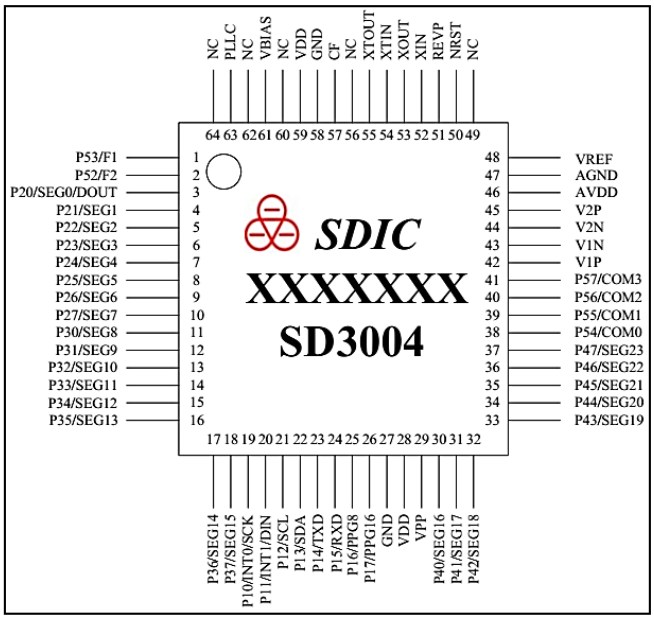


Figure 1. Pin out diagram

Table 1. Pin Descriptions

Pin No.	Pin Name	Pin Attribute	Pin Description
1	P53/F1	I/O	Port P53 or step motor drive F1
2	P52/F2	I/O	Port P52 or step motor drive F2
3-18	P20/SEG0/DOUT--P37/SEG15	LCD driver, I/O	Port P20-P27, P30-P37, or LCD SEG0-SEG15 Pin 3 is data output DOUT during OTP programming
19	P10/INT0/SCK	I/O	Port P10 or interrupt INT0, interrupt edge selectable Clock input SCK during OTP programming
20	P11/INT1/DIN	I/O	Port P11 or interrupt INT1, interrupt edge selectable Data input DIN during OTP programming
21	P12/SCL	I/O	Port P12 or I^2C clock SCL
22	P13/SDA	I/O	Port P13 or I^2C data SDA
23	P14/TXD	I/O	Port P14 or UART data transmit TXD
24	P15/RXD	I/O	Port P15 or UART data receive RXD
25	P16/PPG8	I/O	Port P16 or 8 bits PPG output PPG8
26	P17/PPG16	I/O	Port P17 or 16 bits PPG output PPG16
27	GND	Ground	Digital ground
28	VDD	Power	Digital supply voltage
29	VPP	HV power	High voltage power for OTP programming
30-37	P40/SEG16--P47/SEG23	LCD driver, I/O	Port P40-P47 or LCD SEG16-SEG23
38-41	P54/COM0--P57/COM3	LCD driver, I/O	Port P54-P57 or LCD common COM0-COM3
42	V1P	Analog	Channel 1 (Current) positive input
43	V1N	Analog	Channel 1 (Current) negative input
44	V2N	Analog	Channel 2 (Voltage) negative input
45	V2P	Analog	Channel 2 (Voltage) positive input
46	AVDD	Power	Analog supply voltage
47	AGND	Ground	Analog ground
48	VREF	Analog	2.5V reference output
49	NC	-	No connect, can connect to supply or ground
50	NRST	I	Reset pin, active low
51	REVP	O	Goes high when phase difference between voltage and current is greater than 90 degrees. REVP updates its logic state when a CF pulse is issued.
52	XIN	Analog	3.58MHz crystal oscillator input
53	XOUT	Analog	3.58MHz crystal oscillator output
54	XTIN	Analog	32.768kHz crystal oscillator input
55	XTOUT	Analog	32.768kHz crystal oscillator output
56	NC	-	No connect, can connect to supply or ground
57	CF	O	Calibration frequency output
58	GND	Ground	Digital ground

59	VDD	Power	Digital supply voltage
60	NC	-	No connect, can connect to supply or ground
61	VBIAS	Analog	LCD bias voltage, adjustable through external resistor
62	NC	-	No connect, can connect to supply or ground
63	PLLC	Analog	External capacitor for PLL
64	NC	-	No connect, can connect to supply or ground

Typical Application

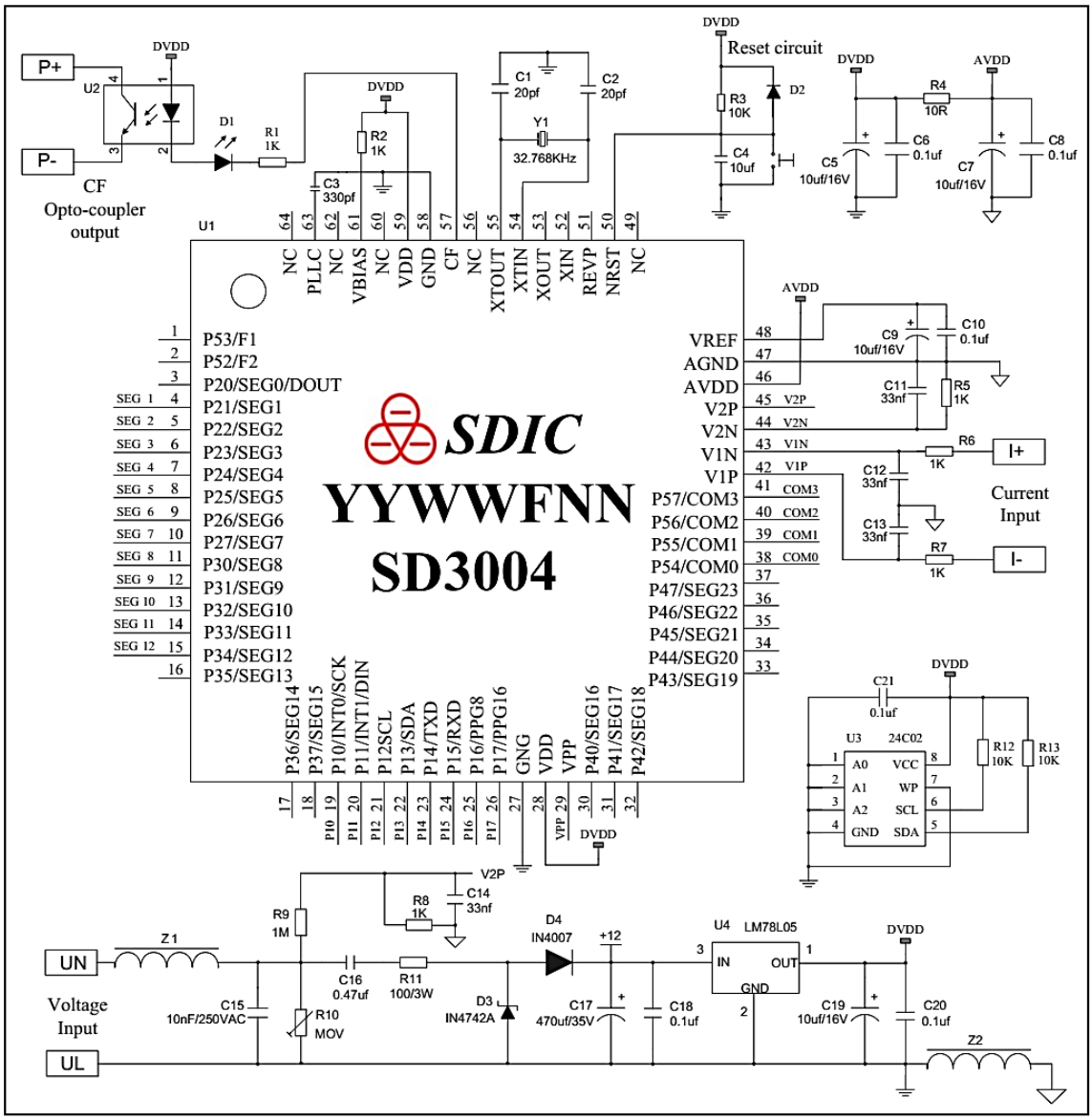


Figure 2. Typical application diagram

Electrical Specifications

Table 2. Absolute Maximum Ratings

Symbol	Parameter	Minimum	Maximum	Unit
T_A	Operating temperature	-40	+85	℃
T_S	Storage temperature	-55	+150	℃
V_{DD}	Supply voltage	-0.2	+7.0	V
Vpp	Programming voltage	-0.2	+13	V
V_{IN}, V_{OUT}	Digital input/output voltage	-0.2	V_{DD}+0.3	V
T_L	Reflow temperature profile		Per IPC/JEDECJ-STD-020C	℃

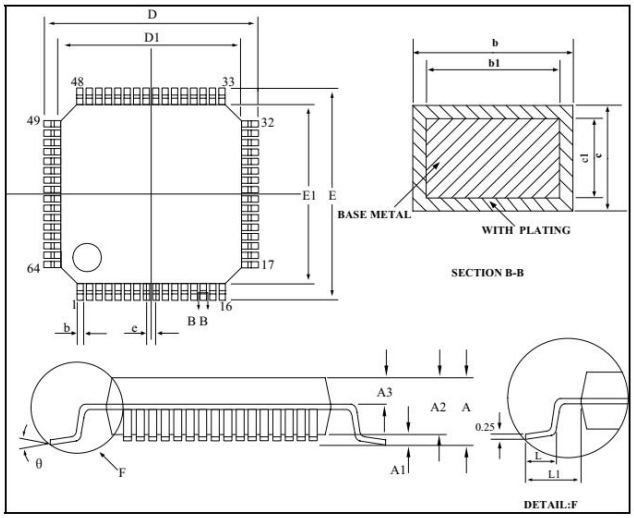
Remarks:
1. CMOS device can easily be damaged by electrostatics. It must be stored in conductive foam, and careful not to exceed the operating voltage range.
2. Turn off power before insert or remove the device.

Table 3. Electrical Specifications（VDD=5V，AVDD=5V）

Symbol	Parameter	Minimum	Typical	Maximum	Unit	Conditions/Remarks
PLLOSC	Operating frequency 1	--	3.604	--	MHz	PLL clock
OSC32K	Operating frequency 2	--	32.768	--	kHz	External crystal oscillator
HOSC	Operating frequency 3	--	3.58	--	MHz	External high frequency crystal oscillator
RC32K	Operating frequency 4	16	--	--	kHz	Internal RC oscillator
FOSC	Operating frequency	--	3.58	--	MHz	Operating frequency 1-4 or a selectable frequency derived from them
VDD	Digital power supply	2.4		5.25	V	
AVDD	Analog power supply	4.75	5	5.25	V	
IDD1	Operating current 1	--	5	--	mA	3.58MHz clock, MCU active, energy measuring, LCD displaying
IDD2	Operating current 2	--	15	30	uA	32.768KHz clock, MCU sleep, energy measuring stops, LCD displaying
IDD3	Operating current 3	--	--	1	uA	All oscillators stop, MCU stops
VIL	Digital input low voltage	--	--	0.3VDD		PORT2/PORT3/PORT4/PORT5
		--	--	0.2VDD		PORT1
		--	--	0.2VDD		NRST
VIH	Digital input high voltage	0.7VDD	--	--		PORT2/PORT3/PORT4/PORT5
		0.8VDD	--	--		PORT1
		0.8VDD	--	--		NRST
Rpu	Pull up resistance	50K	--	100K	Ω	PORT1/NRST
VOL	Digital output low voltage	--	--	0.3VDD	V	
VOH	Digital output high voltage	0.7VDD	--	--	V	
VPP	Programming voltage	11.75	12	12.25	V	

VREF	Reference value	2.3	2.5	2.7	V	
T_VREF	Reference TC		30	60	ppm/℃	－40℃～85℃
I_ACCU	Measurement accuracy	--	0.5	1	%FSR	Voltage/current channels
RANGE1	Channel 1 input signal range	--	--	450	mV	Current input, 50/60 Hz
RANGE2	Channel 2 input signal range	--	--	650	mV	Voltage input, 50/60 Hz
CMR	Common mode range	0		2.7	V	Channel 1 and channel 2

Packaging Information



Dimensions: mm

Symbol	Min.	Nom.	Max.
A	——	——	1.60
A1	0.05	——	0.20
A2	1.35	1.40	1.45
A3	0.59	0.64	0.69
b	0.19	——	0.27
b1	0.18	0.20	0.23
c	0.13	——	0.18
c1	0.12	0.13	0.14
D	11.80	12.00	12.20
D1	9.90	10.00	10.10
E	11.80	12.00	12.20
E1	9.90	10.00	10.10
e	0.50BSC		
L	0.45	——	0.75
L1	1.00BSC		
θ	0	——	7

Figure 3. Mechanical specification

J. — Servidores e Módulo Estudados, mas não utilizados neste trabalho.

a. Gerenciador CloudMQTT

O Blynk é funcional, mas não possui uma interface web para visualização das mensagens trocadas entre publicadores e assinantes, o que limita seu uso a celulares que permitam usar esse aplicativo. Também não é totalmente livre, pois além de não oferecer acesso aos seus servidores tampouco é gratuito após ultrapassar um limite de funções ou recursos. A criação de um link de compartilhamento de interface também gera um 'custo de energia', mas ao contrário dos *widgets* este recurso do link não é reembolsável depois de usado.

O CloudMQTT[38] é gerenciado por servidores Mosquitto em nuvem. O Mosquitto implementa o protocolo MQTT. O CloudMQTT permite que você se concentre no aplicativo, em vez de gastar tempo no escalonamento do broker ou na atualização da plataforma. É um serviço que pode ser assinado, mas pode ser usado gratuitamente.

Limitações:

- Não possuir uma interface gráfica web (do tipo *widgets*), respondendo apenas com textos, mas já é um bom começo para conferir tráfego e comunicação.
- Na versão gratuita o número de conexões é limitado a cinco usuários, assim como a largura de banda de 10 kbit/s.

Pontos positivos:

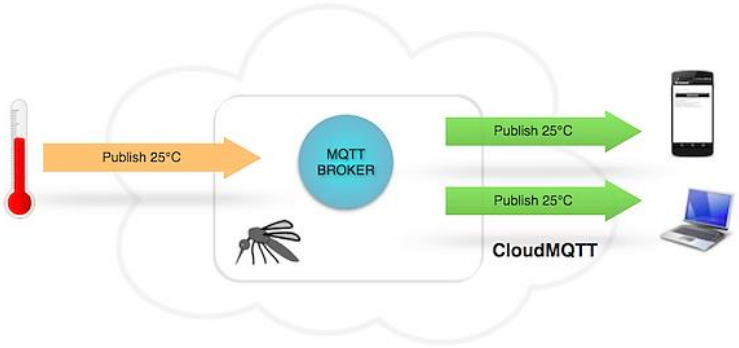
- Possui uma conexão *WebSoquet*[39] que é usada, como citado, para comunicação via textos na página do servidor.
- Possui aplicativos móveis (como o *MQTTDash* e *MQTT Dashboard*, ambos testados durante este trabalho e semelhantes à interface Blynk) que são compatíveis.
- Compatível com plataformas Arduino e ESP, entre outras.

Exemplos de conexão e interface web do CloudMQTT, visto nas Figuras J1 e J2:

[38] CloudMQTT: https://www.cloudmqtt.com/

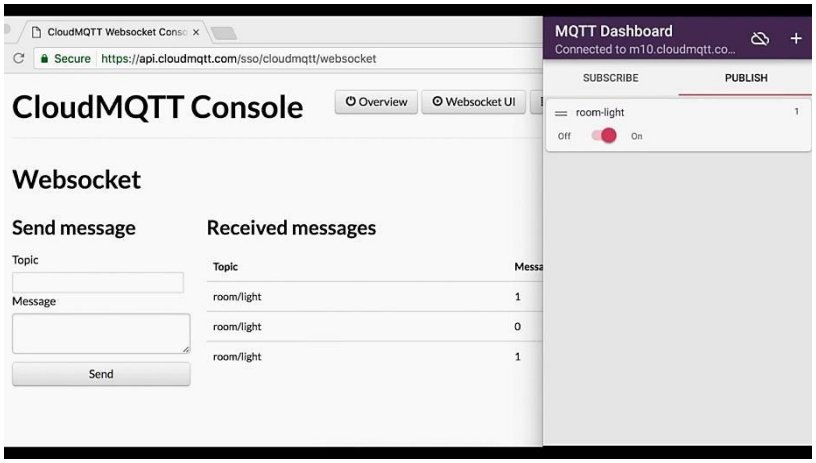
[39] A especificação *WebSocket* define uma API que estabelece conexões de "soquete" entre um navegador da web e um servidor. Em outras palavras, há uma conexão persistente entre o cliente e o servidor e ambas as partes podem começar a enviar dados a qualquer momento. Fonte: https://www.html5rocks.com/pt/tutorials/websockets/basics/

Figura J1 – Exemplo de conexões do CloudMQTT.



Fonte: CloudMQTT.

Figura J2 – Exemplo de aparência das Interfaces web e aplicativo móvel do CloudMQTT.



Fonte: CloudMQTT.

Por questões de prazos esta API não foi utilizada no desenvolvimento do protótipo, ficando aqui apenas como referência ao leitor

b. Gerenciador MQTT Cayenne

O Cayenne[40], da empresa myDevices[41], é outra API encontrada e avaliada para este trabalho. É um meio termo entre Blynk e CloudMQTT, associando o melhor de ambos. Eis alguns pontos positivos:

- Possui interface web gráfica, configurável conforme o projeto.
- Possuem um aplicativo configurável próprios, de fácil configuração.
- Compatível com plataformas Arduino, ESP, Raspbery Pi e LoRa.

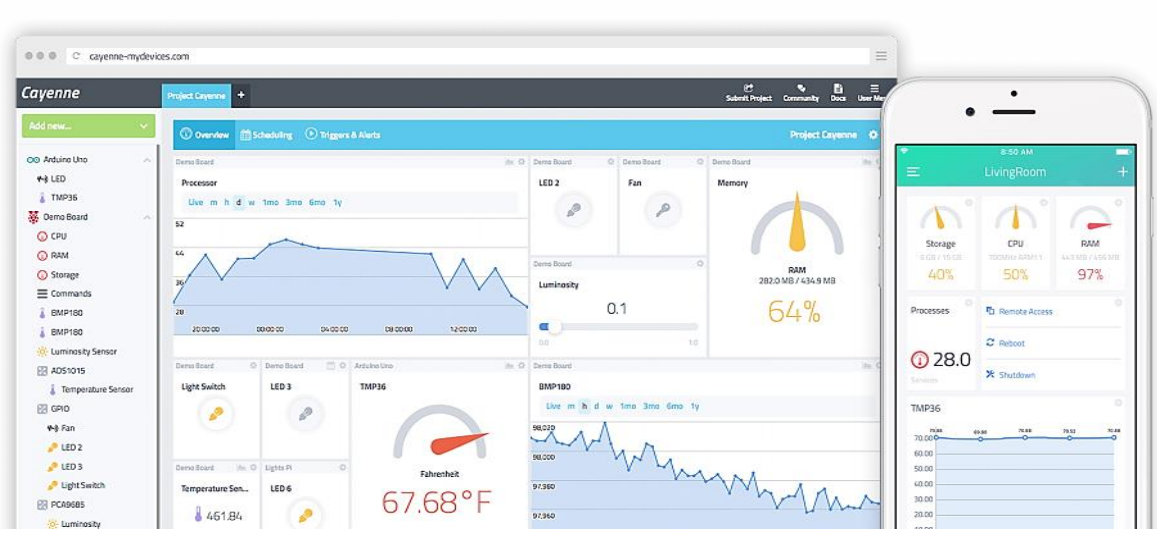
Limitações:

- Também é um serviço pago, mas aparentementea empresa oferece soluções personalizadas para clientes dos mais variados segmentos (e cobra por isso).

[40] Cayenne: https://mydevices.com/cayenne/features/api/
[41] myDevices: https://mydevices.com/

Um exemplo de interface com Cayenne por ser visualizado na Figura J3:

Figura J3 – Interface web e aplicativo móvel customizados do Cayenne, da myDevices.



Fonte: myDevices.

Por questões de prazos esta API não foi utilizada no desenvolvimento do protótipo, ficando aqui apenas como referência ao leitor.

c. Gerenciador ThingSpeak

O ThingSpeak[42] é um dos mais populares serviços de IoT em nuvem, frequentemente citado em bibliografia e páginas tutoriais. Inclui análise de dados por softwares matemáticos, visualizações de gráficos e plug-ins para gerar arquivos HTML, JavaScript ou CSS. Algumas ações incluem envio de *tweets*, que podem ser usados para comandar outras plataformas. O site do ThingSpeak atua como intermediário remoto das ações programadas.

Alguns pontos positivos:

- Possui interface web intuitiva e com diversos recursos, como análise de dados (integração Matlab).
- Disponibiliza um App Android, o Thingview, que permite acessar controles e gráficos pré-programados.

Limitações:

- Não possui interface widgets programáveis ou configuráveis no App, sendo apenas um canal de visualização do que foi configurado no site.

Assim como nos demais casos, uma biblioteca precisa ser baixada e incluída no código (ESP ou outro) para que a comunicação se estabeleça. No caso do aplicativo, soube-se

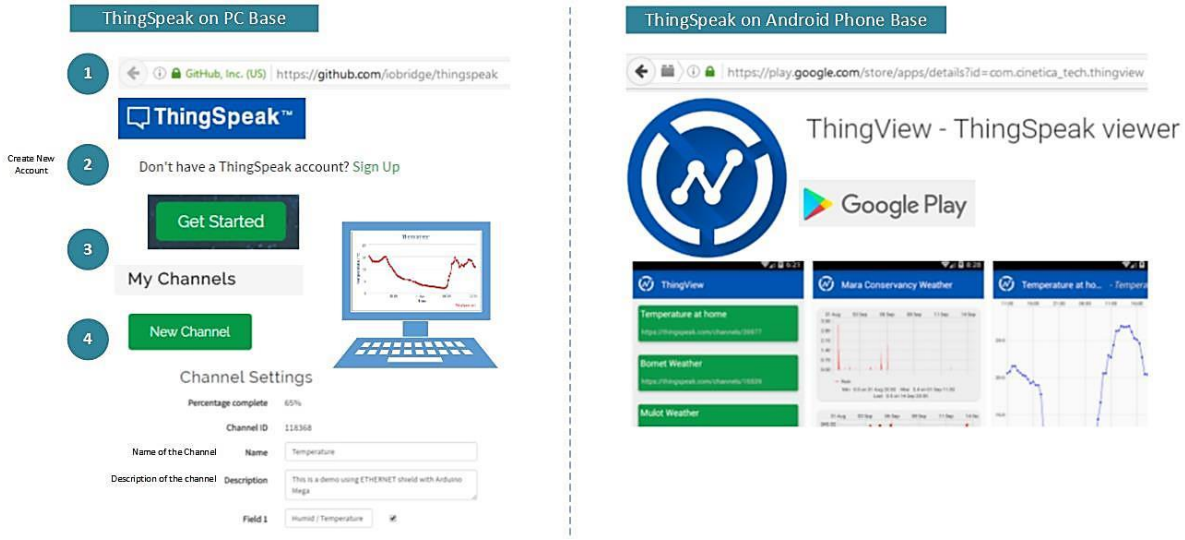
[42] ThingSpeak: https://thingspeak.com/

de outro App, chamado Virtuino, que além de ter conexão com o ThingSpeak facilitada também possui uma API configurável.

Ao fazer cadastro no site do ThingSpeak e também criar um canal neste, uma ID e uma chave de acesso serão disponibilizadas para o projeto. Cada canal tem espaço para até oito sensores ou linhas de comunicação.

Exemplos do que esperar das APIs do ThingSpeak podem ser observadas na Figura J4:

Figura J4 – Tela exemplo da plataforma ThingSpeak.



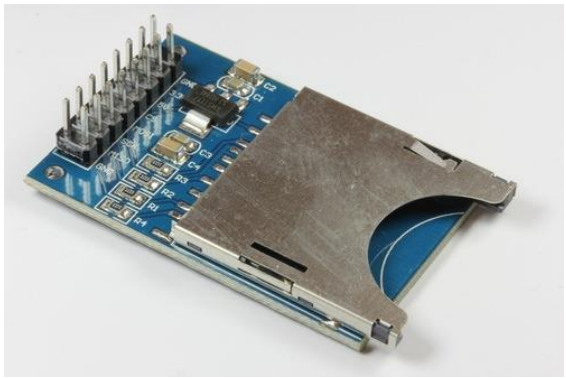
Fonte: 14Core.com.

Por questões de prazos esta API não foi utilizada no desenvolvimento do protótipo, ficando aqui apenas como referência ao leitor.

d. Usando módulo SD Card

O módulo do cartão SD utilizado neste trabalho é mostrado na Figura J5:

Figura J5 – Módulo SD Card.



Fonte: Google.

Consta de uma simples placa de circuito impresso com soquete para cartões SD grandes ou usando adaptadores, um regulador de tensão e componentes discretos de

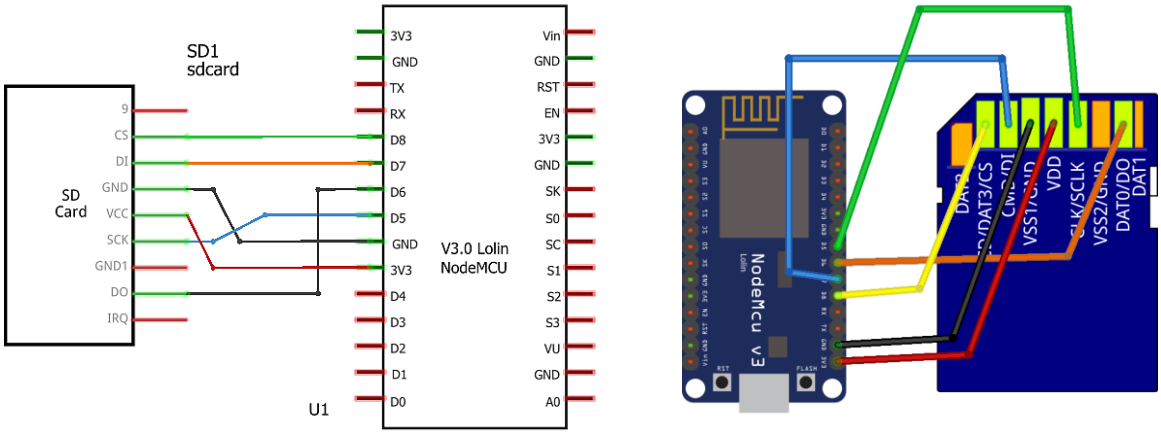
polarização. A alimentação desta placa pode ser tanto em 5,0 Vcc quanto em 3,3 Vcc. Por razão do consumo de energia considerável (perto de 1,0 A) durante a gravação do cartão, optou-se por deixar no barramento de 5,0 Vcc. Ainda assim testes foram realizados com alimentação em 3,3 Vcc.

Mas há outros modelos de leitoras/ gravadoras de cartão, inclusive algumas placas já com RTC acoplado.

O cartão usado durante o trabalho foi uma mídia de 2GB de capacidade formatada no padrão FAT32.

O diagrama de fiação da ligação ao NodeMCU é o da Figura J6, que segue:

Figura J6 – Diagrama elétrico do módulo SD Card ao ESP (a) e diagrama de fiação (b).



(a) (b)

Fonte: Autor, editado no Fritzing.

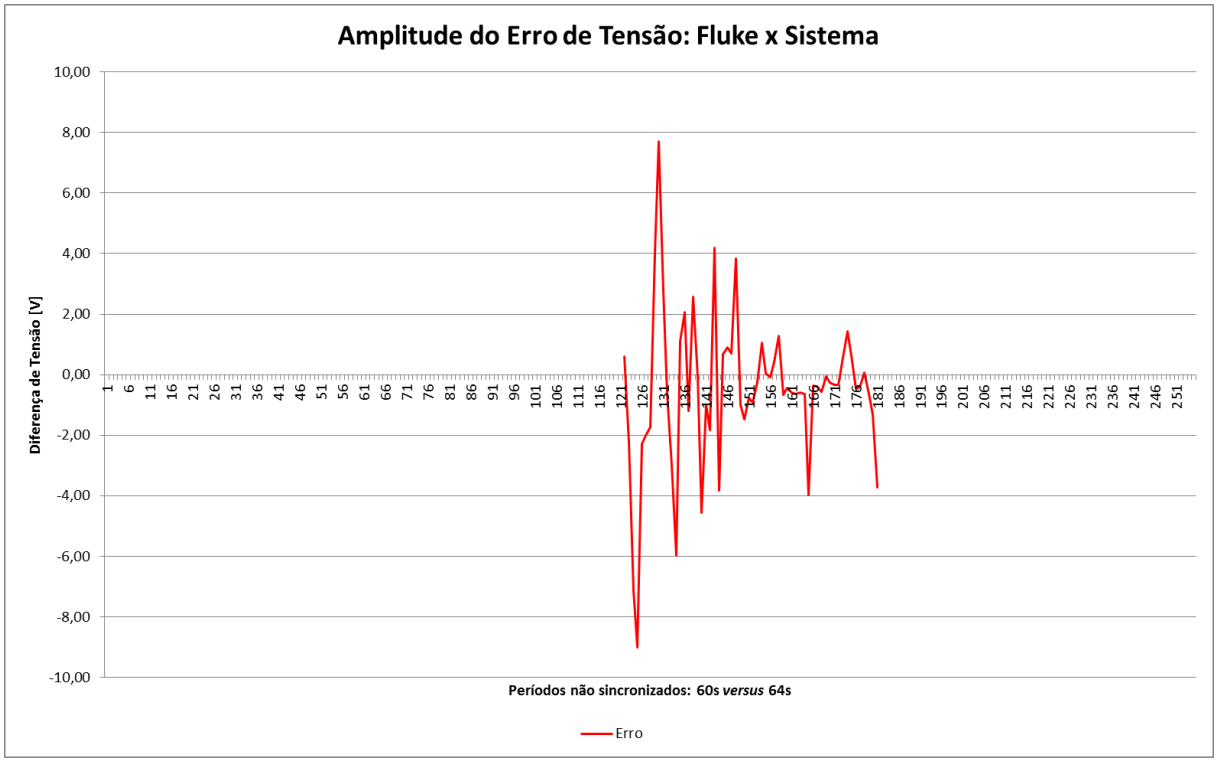
Este tipo de módulo em geral faz uso do barramento SPI. A biblioteca utilizada com o SD Card no ESP foi a <SD.h> que é referenciada à biblioteca <SPI.h>, mas esta última não precisou ser declarada, embora estivesse instalada na IDE.

Mais adiante, na seção de resultados, será visto que o uso do cartão SD não foi possível, descrevendo com mais detalhes.

K. — Tentativa de Verificação de Erros de Leituras: Fluke *versus* Protótipo.

As Figuras K1 e K2 representam o resumo das tentativas inconclusivas de se obter valores de erros entre as leituras do analisador de qualidade de energia e o protótipo apresentado neste trabalho.

Figura K1 – Curvas de Erros de Amplitude de Tensão: Fluke 43B versus Protótipo.

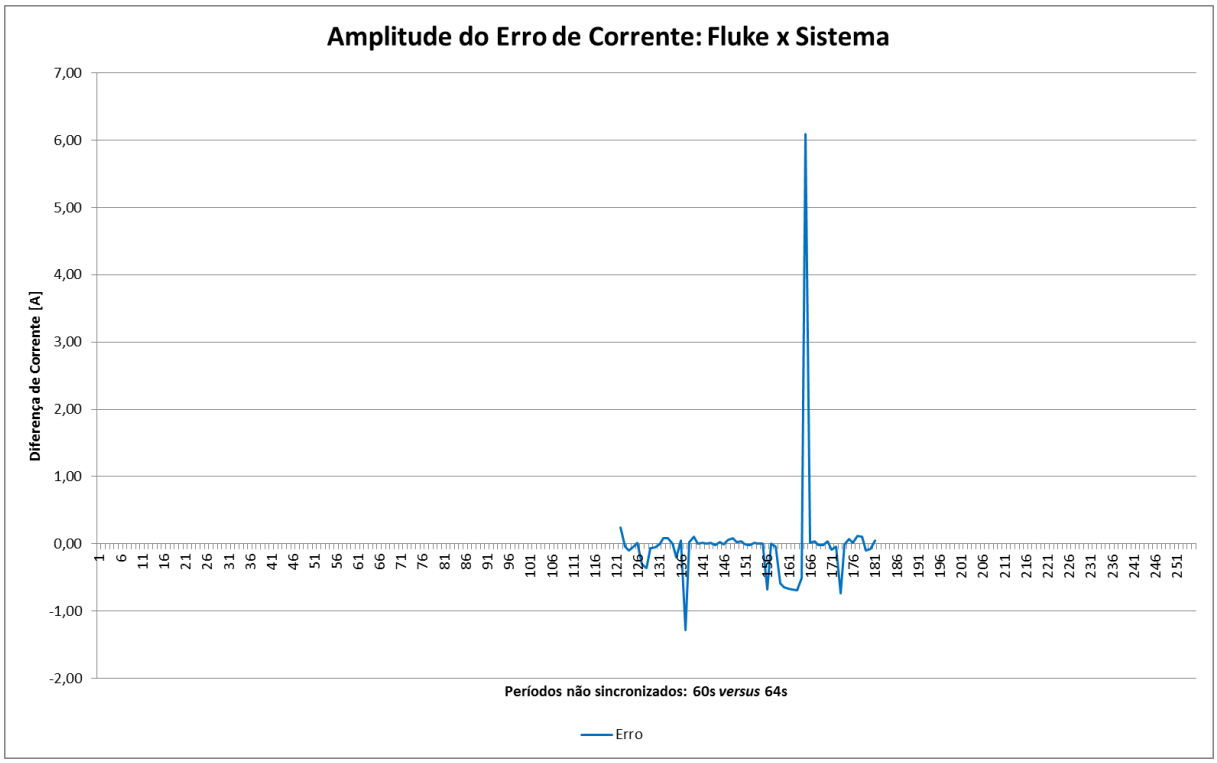


Fonte: Autor.

Note que a dificuldade de sincronia entre o protótipo e o analisador Fluke, e prováveis perdas de pacotes ou dados na transmissão do servidor impossibilitam que se tire alguma definição dos erros. O que se sabe é que em tempo real nenhum dos sistemas deixava a desejar em termos de valores que mostravam em tela, com margens muito menores do que a mostrada na Figura K1 e K2, por exemplo.

Como as variações dos sinais eram praticamente instantâneas, qualquer valor alto capturado em períodos diferentes já iria promover valores de erro significativos, que perfeitamente poderiam não condizer com a realidade.

Figura K2 – Curvas de Erros de Amplitude de Corrente: Fluke 43B versus Protótipo.



Fonte: Autor.

a. Complicadores para uma Análise mais Precisa:

- O operador do sistema precisou leva-lo para casa, pois os problemas de conexão com a internet no campus não permitiram que um ambiente de testes fosse criado para tal, o que resulta em outros problemas;
- A questão de segurança, sendo que tanto o instrumento (de alto valor) como o a possível insegurança (curto-circuito, etc.) em relação ao protótipo demandavam atenção constante do operador, não podendo ser abandonados em teste;
- A indisponibilidade do operador ou outra pessoa capacitatada de fazer os testes fora dos períodos de fins de semanas, aliado a intempéries comuns na estação (mai-jun);
- O curto prazo para obter algum resultado e outras demandas do projeto, como ajuste textual e outras pesquisas.

Estes são os fatores que atrapalharam e não permitiram um ensaio subsequente, maior do que 4h, como desejado.

Sugestões de aplicação ou possíveis melhorias:

Alterar o projeto (e talvez o servidor MQTT, se for o caso) para permitir o envio de dados de uma única vez em períodos sincronizados, fazendo as médias dentro do próprio ESP8266.

Usar algum tipo de módulo ou circuito que permitisse substituir em hardware a falta do SD Card, se o problema que atrapalhava seu uso (falta do relógio e não poder usar o display LCD) não conseguisse ser resolvido.

b. Registro de Transientes:

O analisador de qualidade de energia Fluke acusou uma sobrecarga (overload) no sinal de corrente, que foi registrada, tanto no arquivo .csv (valor tendendo ao infinito) quando no display LCD do instrumento (a plotagem do gráfico ignorou esse valor automaticamente).

Como a ponteira de corrente tem capacidade para até 400 A, logo deve ter ocorrido um surto de curtíssima duração muito mais elevado. Como foi um evento único e outros testes não foram realizados, entende-se que foi um evento real, mas também não se descarta falha do analisador.

Deve-se levar em consideração que se (ou para que) o sensor PZEM conseguisse captar algo similar provavelmente tanto o μC quando o código de extração de seus dados precisaria conseguir abstrair uma ocorrência desse tipo.