UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

WYARA MARIA CARLOS SOUZA PONTES

IMPLANTAÇÃO DE ESTAÇÃO SOLARIMÉTRICA E DE SISTEMA SUPERVISÓRIO COM SCADABR E PLATAFORMA IoT EM USINA FOTOVOLTAICA NA UNILAB-CE

FORTALEZA

2021

WYARA MARIA CARLOS SOUZA PONTES

IMPLANTAÇÃO DE ESTAÇÃO SOLARIMÉTRICA E DE SISTEMA SUPERVISÓRIO COM SCADABR E PLATAFORMA IoT EM USINA FOTOVOLTAICA NA UNILAB-CE

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Engenharia Elétrica.

Orientadora: Prof. Ph.D. Ruth Pastôra Saraiva Leão

Coorientadora: Prof. Dra. Lígia Maria Carvalho Sousa Cordeiro

FORTALEZA

2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

P859i Pontes, Wyara Maria Carlos Souza.
IMPLANTAÇÃO DE ESTAÇÃO SOLARIMÉTRICA E DE SISTEMA SUPERVISÓRIO COM SCADABR E PLATAFORMA IoT EM USINA FOTOVOLTAICA NA UNILAB-CE / Wyara Maria Carlos Souza Pontes. – 2021.
150 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Fortaleza, 2021.
Orientação: Profa. Ph.D. Ruth Pastôra Saraiva Leão.
Coorientação: Profa. Dra. Lígia Maria Carvalho Sousa Cordeiro.

1. Aquisição de dados. 2. Estação solarimétrica. 3. Minigeração solar fotovoltaica . 4. ScadaBR. 5. Supervisão. I. Título.

CDD 621.3

WYARA MARIA CARLOS SOUZA PONTES

IMPLANTAÇÃO DE ESTAÇÃO SOLARIMÉTRICA E DE SISTEMA SUPERVISÓRIO COM SCADABR E PLATAFORMA IoT EM USINA FOTOVOLTAICA NA UNILAB-CE

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Engenharia Elétrica.

Aprovada em ___/___/___.

BANCA EXAMINADORA

_____

Prof. Ph.D. Ruth Pastôra Saraiva Leão (Orientadora)
Universidade Federal do Ceará (UFC)

_____

Prof. Dra. Lígia Maria Carvalho Sousa Cordeiro (Coorientadora)
Universidade da Integração Internacional da Lusofonia Afro-Brasileira (UNILAB)

_____

Prof. Dr. Hermínio Miguel de Oliveira Filho
Universidade da Integração Internacional da Lusofonia Afro-Brasileira (UNILAB)

_____

Prof. Dr. Raimundo Furtado Sampaio
Universidade Federal do Ceará (UFC)

A Deus.
A minha mãe, Vera, e meu esposo,
Werllen.

AGRADECIMENTOS

Agradeço primeiramente a Deus que por sua infinita graça me permitiu alcançar mais esta realização.

Agradeço à minha mãe, Vera, meu maior exemplo de força e perseverança. Agradeço ao meu esposo Werllen, por sempre estar ao meu lado e me incentivar a lutar pelos meus sonhos.

À prof. PhD Ruth, pela paciência, disponibilidade e incentivos durante a orientação e pelos direcionamentos que contribuíram valiosamente com este trabalho.

À prof. Dra. Lígia pela orientação, paciência e sensibilidade durante a orientação desde a graduação e enquanto coordenadora e coorientadora durante o mestrado e por ter me dado a oportunidade de ministrar a primeira aula para alunos de graduação.

Agradeço ao prof. Dr. Hermínio, que como coordenador do projeto, como membro da banca examinadora realizou apontamentos que contribuíram para o desenvolvimento e melhoria deste trabalho.

Ao prof. Dr. Raimundo pela disponibilidade em participar da banca e pelas sugestões que tanto contribuíram para a melhoria deste trabalho.

Agradeço ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), ao Programa de Eficiência Energética (PEE) e ao Programa de Pesquisa e Desenvolvimento (P&D) da Agência Nacional de Energia Elétrica (ANEEL) e à Enel Distribuição Ceará por todo o suporte oferecido para a execução desta pesquisa.

À Universidade da Integração Internacional da Lusofonia Afro-Brasileira (UNILAB), especialmente ao IEDS, pela disponibilização dos laboratórios e a DTI pelo suporte disponibilizado.

Aos colegas Caíke, Kevin, Cleison, Jairo, Gilmar, Carla, Elefson, Jeferson, Caio e Marcelo pelo apoio durante a jornada do mestrado.

Aos meus amigos Evilásio, Izabela e Lívia que sempre me incentivaram e me apoiaram.

Ao meu chefe, Jonas, pela compreensão e apoio durante a finalização deste trabalho.

Ao Roni (suporte técnico da Fimer (ABB)) e Saulo (suporte técnico Sigma Sensors) pela disposição em ajudar e disponibilizar informações relevantes para o desenvolvimento do trabalho.

RESUMO

Em uma mudança de paradigma no segmento de geração de energia elétrica, instituições privadas, de serviços públicos e clientes estão instalando unidades de geração distribuída para produzir eletricidade mais perto de seu uso final. No Brasil, o crescimento da geração distribuída a partir de recursos renováveis, especialmente a solar fotovoltaica, tem crescido desde a implementação de políticas de medição *net-metering* em 2012. O desempenho dos sistemas fotovoltaicos é influenciado principalmente pelos índices de irradiação incidente e pela temperatura do ar que, por sua vez, influencia a temperatura do módulo fotovoltaico. Estações solarimétricas e sistemas de supervisão e monitoramento são de fundamental importância para aquisição e análise de dados que auxiliam na tomada de decisão e no gerenciamento das plantas de geração. Este trabalho tem como objetivo apresentar a implantação de uma estação solarimétrica comercial e de sistemas supervisórios baseados em SCADABR e plataforma de monitoramento IoT da estação solarimétrica e da usina de minigeração solar fotovoltaica de 254,2 kWp, instalada no campus das Auroras da Universidade da Integração Internacional da Lusofonia Afro-Brasileira – UNILAB, no Ceará. O sistema supervisório ScadaBR é capaz de monitorar remotamente a estação solarimétrica e a planta de geração. Para acesso remoto por dispositivo móvel, foi desenvolvido um protótipo de sistema de aquisição de dados e de comunicação de baixo custo baseado em plataforma eletrônica IoT para a estação solarimétrica bem como usina fotovoltaica. Além do desenvolvimento de um sistema de monitoramento via website, usando aplicativo para dispositivos móveis e através de SCADA, o presente trabalho também apresenta análise de desempenho do protótipo de aquisição de dados, tendo como referência a estação solarimétrica. A supervisão da estação solarimétrica e da planta de geração permitem o monitoramento contínuo e em tempo real dos equipamentos para verificação do desempenho.

Palavras-chave: Aquisição de dados, Estação solarimétrica, Minigeração solar fotovoltaica, ScadaBR, Supervisão, Plataforma IoT.

ABSTRACT

In a paradigm shift in the electric power generation segment, private institutions, utilities and customers are installing distributed generation units to produce electricity closer to their end use. In Brazil, the growth of distributed generation from renewable resources, especially solar photovoltaic, has grown since the implementation of net-metering measurement policies in 2012. The performance of photovoltaic systems is mainly influenced by the incident radiation levels and the temperature air, which in turn influences the temperature of the photovoltaic module. Solarimetric stations and supervision and monitoring systems are of fundamental importance for the acquisition and analysis of data that assist in decision making and in the management of generation plants. This work presents the implementation of a commercial solarimetric station and supervisory systems based on SCADABR and through the IoT monitoring platform of the solarimetric station and the 254.2 kWp photovoltaic solar mini-generation plant, installed on the Auroras campus of the Universidade da International Integration of Afro-Brazilian Lusophony - UNILAB, in Ceará. The ScadaBR supervisory system is capable of remotely monitoring the solarimetric station and the generation plant. For remote access by mobile device, a prototype of a low-cost data acquisition and communication system based on an electronic IoT platform for the solarimetric station was developed. In addition to the development of a monitoring system via the website, via an application for mobile devices and through Scada, the present work also presents performance analysis of the data acquisition prototype, using the solarimetric station as a reference. The supervision of the solarimetric station and the generation plant allowed continuous and real-time monitoring of the equipment to verify performance.

Keywords: Data Acquisition, Solarimetric Station, Photovoltaic solar mini generation, ScadaBR, Supervision, IoT platform.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ANEEL	Agência Nacional de Energia Elétrica
FF	Fator de forma
HMI	Human-Machine Interface (Interface Homem-Máquina)
IDE	*Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)
INPE	Instituto Nacional de Pesquisas Espaciais
IPI	Imposto sobre Produto Industrializado
MME	Ministério de Minas e Energia
MPPT	*Maximum Power Point Tracking* (Rastreamento do ponto de máxima potência)
PEE	Programa de Eficiência Energética
RTU	*Remote Terminal Unit* (Unidade Terminal Remota)
SCADA	Sistema de Supervisão e Aquisição de Dados
SGBD	Sistema de Gerenciamento de Banco de dados
STC	Condição Padrão de Teste
UCP	Unidade de condicionamento de Potência

SUMÁRIO

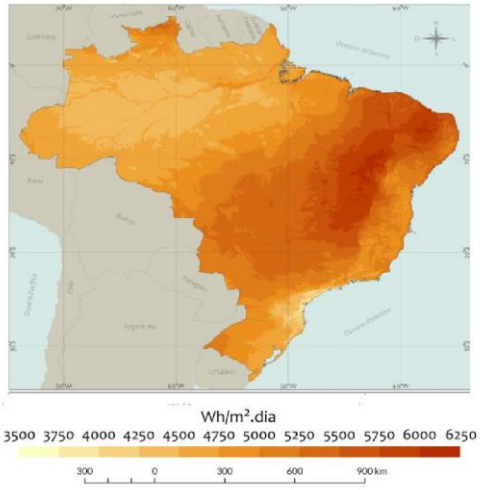
1 INTRODUÇÃO

A preocupação com a sustentabilidade ambiental e segurança energética, e a procura pela diversificação da matriz elétrica nos últimos anos, tem impulsionado a geração de energia elétrica através de fontes renováveis como a solar fotovoltaica (FV) e a eólica (NASCIMENTO, 2017).

As hidrelétricas possuem participação significativa na matriz elétrica brasileira, proporcionando ao sistema elétrico do país aspectos únicos quanto à baixa emissão de gases de efeito estufa. Entretanto, a geração por fonte hidráulica, assim como outras fontes renováveis, é suscetível às características climáticas, de forma que o recurso hídrico armazenado nos reservatórios pode atingir níveis críticos durante períodos de estiagem, oferecendo riscos à segurança energética. Associado a esses fatos, no Brasil predomina a geração hidrelétrica de grande porte, centralizada, e, portanto, distante do usuário final, com desdobramentos como necessidade de investimento em redes de transmissão e, consequentemente, perdas no transporte de energia dos centros de geração aos centros consumidores (TOLMASQUIM, 2016).

O Brasil, no entanto, possui um considerável potencial de energia solar em toda sua extensão territorial, como mostrado na Figura 1.

Figura 1- Média anual do total diário de irradiação no plano inclinado da latitude no Brasil.



Fonte: Adaptado de INPE (2017).

Desse modo, o uso do recurso solar para geração de energia elétrica representa uma importante alternativa no contexto brasileiro para complementar a geração por fontes já

consolidadas como as hidrelétricas, uma vez que nos períodos de seca, quando há diminuição da disponibilidade de recursos hídricos, os níveis de irradiação solar são mais elevados, favorecendo a produção por fonte solar (INPE, 2017).

A Agência Nacional de Energia Elétrica (ANEEL) através da Resolução 687/2015 define as condições gerais para a conexão de planta de microgeração, com potência instalada menor ou igual a 75 kW, e minigeração, com potência instalada entre 75 kW e 5 MW, à rede elétrica no Brasil. A resolução citada estabelece o Sistema de Compensação de Energia que viabiliza a compensação de excedente de energia ativa injetada na rede elétrica por unidade consumidora com micro e minigeração distribuída. A compensação consiste na obtenção de créditos mediante injeção de energia na rede elétrica, que é posteriormente usado, em prazo de até 60 meses, para abater o consumo de energia elétrica da mesma unidade consumidora ou de outra unidade consumidora de mesma titularidade (ANEEL, 2015). Esse sistema é também conhecido pelo termo em inglês *net metering*, e tem sido responsável pelo grande desenvolvimento da energia solar fotovoltaica no Brasil.

Além disso, outros incentivos foram criados para impulsionar a geração distribuída. A Lei 8.922/20 garante a isenção do Imposto sobre Circulação de Mercadorias e Serviços (ICMS) sobre a energia gerada a partir de micro e minigeração de energia solar FV e injetada na rede elétrica quando a produção de energia excede o consumo (ABSOLAR, 2020).

Dentre outros incentivos criados no âmbito nacional estão a redução de IPI (Imposto sobre Produto Industrializado) e II (Imposto de Importação) incidentes na importação de equipamentos e componentes e dos insumos empregados na produção de módulos fotovoltaicos e redução de ICMS sobre as operações envolvendo alguns equipamentos utilizados para a geração de energia elétrica solar e eólica (ABSOLAR, 2019).

Os incentivos, aliados ao potencial brasileiro, resultaram no alcance da marca de mais de 4,8 GW de potência instalada em geração distribuída derivada dos sistemas fotovoltaicos instalados e com projeções bastante promissoras (ANEEL, 2020).

Em relação a outras fontes renováveis, como a energia eólica, a conversão solar fotovoltaica apresenta muitas vantagens, pois é silenciosa, sem partes móveis; de baixo impacto visual por ser estática e posicionada no plano horizontal; de baixo impacto ambiental por não emitir gases de efeito estufa em sua operação, como também não requer grandes manutenções, nem grande infraestrutura de construção civil (ABSOLAR, 2020). Entretanto, existem aspectos que ainda inibem alguns consumidores a aderirem ao uso de sistemas fotovoltaicos, como:

investimento inicial elevado e o fato de que as células fotovoltaicas ainda possuem uma eficiência relativamente baixa.

O potencial de geração das células fotovoltaicas depende principalmente da irradiância solar incidente. A temperatura das células, que por sua vez é uma função do microclima local, também influencia significativamente na eficiência desses dispositivos. O local de instalação tem um microclima associado, portanto, a influência da climatologia local na eficiência da conversão solar fotovoltaica (FV) deve ser considerada (FLUKE ACADEMY, 2020).

Os processos térmicos que relacionam um painel fotovoltaico ao ambiente são modulados por quatro variáveis ambientais principais: insolação, temperatura do ar, velocidade do vento e umidade relativa do ar (ADEH et al., 2019). Outros aspectos que interferem na produção das células fotovoltaicas é o sombreamento de nuvens, árvores e prédios, sujidade como excrementos de aves, poeira e particulados e acúmulo de água provenientes de chuva (ALVES, 2017).

Uma estação solarimétrica é um dos instrumentos de maior relevância em usinas de geração fotovoltaica. Os dados da estação podem ser usados para acompanhar e avaliar a operação da usina, auxiliando na tomada de decisão para aumentar a produção da planta.

No Brasil, o uso de estações equipadas com pelo menos dois piranômetros e sensores de umidade relativa, temperatura e velocidade do vento é obrigatório para projetos de usinas solares de grande porte que se destinam à venda de energia em leilões de energia (EPE, 2016).

Sistemas de monitoramento em usinas FV possibilitam detectar defeitos antes da ocorrência de falhas e anormalidades de forma eficaz e permite o acompanhamento de grandezas que são monitoradas em intervalos de tempo relativamente pequenos. Logo, um sistema de monitoramento na usina FV é muito importante para análise de dados de geração, solução de problemas e na tomada de decisão (TINA, 2016).

Desta forma, o monitoramento dos dados da estação solarimétrica e da geração de energia da planta FV possibilita o acompanhamento de grandezas que influenciam no desempenho da planta, a identificação de fatores indesejados na geração e viabiliza a concepção de modelos de análise de eficiência e qualidade da energia gerada.

Na literatura são encontradas alternativas de aquisição e monitoramento baseados em sistemas SCADA (*Supervisory Control and Data Acquisition*). O ScadaBR é uma das alternativas encontradas e possuem a vantagem de apresentar compatibilidade com a maioria dos protocolos de comunicação dos registradores de dados (RIBEIRO, 2017; GUIMARÃES, 2014).

Também são encontrados sistemas de baixo custo baseados em plataformas de prototipagem eletrônica como Arduino, Raspberry Pi e as placas ESP. Entretanto, a maioria dos sistemas desenvolvidos é instalada em sistemas FV de pequeno porte (microgeração), como nos trabalhos desenvolvidos por Anand et al. (2016), Zubair e Ahmed (2015) e Martins (2019), que são baseados na aquisição direta a partir de sensores de baixo custo menos robustos.

1.1 Objetivos

O objetivo geral desta dissertação é apresentar o projeto, especificação e implantação de uma estação solarimétrica comercial e de sistemas supervisórios capazes de coletar, registrar e visualizar informações sobre variáveis meteorológicas e da usina de minigeração solar fotovoltaica do Campus Auroras da Unilab.

Os objetivos específicos são:

- Especificar e implantar uma estação solarimétrica comercial para a usina solar FV do Campus das Auroras na UNILAB-CE;
- Desenvolver e integrar um sistema supervisório remoto fixo usando o ScadaBR, um software livre, gratuito e de código-fonte aberto, para a estação solarimétrica e a usina FV no Campus das Auroras na UNILAB-CE;
- Projetar, desenvolver e integrar uma plataforma de análise IoT baseada na plataforma ThingSpeak, que oferece conectividade Wi-Fi à Internet e permite acesso remoto a partir de dispositivos móveis, para a estação solarimétrica e a usina solar FV no Campus das Auroras na UNILAB-CE;
- Proporcionar o monitoramento da estação solarimétrica e da usina solar FV do Campus das Auroras na UNILAB-CE.

A usina de minigeração FV, instalada no campus Auroras da UNILAB-CE, faz parte do Projeto Estratégico de Pesquisa e Desenvolvimento (P&D) e do Projeto de Eficiência Energética e Minigeração de Energia Fotovoltaica da UNILAB em parceria com a Enel Distribuição Ceará e Agência Nacional de Energia Elétrica - ANEEL.

1.2 Estrutura do Trabalho

O presente trabalho foi estruturado e dividido em seis capítulos apresentados a seguir.

Capítulo 1 contextualiza a geração distribuída por fonte solar fotovoltaica no Brasil e a importância das estações solarimétricas e sistemas supervisório para acompanhamento do desempenho das plantas de geração. Objetivos e estrutura do trabalho também compõem esse capítulo.

Capítulo 2 traz uma Revisão Bibliográfica que aborda os principais aspectos da energia solar fotovoltaica e o estado da arte em tecnologias fotovoltaicas. Apresenta também os principais parâmetros e índices técnicos de um sistema de geração fotovoltaica, influências de fatores externos no desempenho e principais configurações de conexão com a rede elétrica. O capítulo finaliza apresentando os principais sistemas fotovoltaicos instalados em universidades públicas federais brasileiras, incluindo a usina de minigeração solar fotovoltaica do Campus Auroras na UNILAB-CE.

Capítulo 3 aborda os principais aspectos e características de sistemas de aquisição de dados aplicados a sistemas fotovoltaicos. São apresentados os principais *hardwares e softwares* encontrados na literatura, bem como os principais protocolos de comunicação Modbus e seus parâmetros.

Capítulo 4 descreve a metodologia utilizada no desenvolvimento do sistema de aquisição de dados e monitoramento da estação solarimétrica e da usina do Campus das Auroras-UNILAB-CE.

Os resultados e as conclusões prévias obtidas no desenvolvimento do trabalho são apresentados no Capítulo 5. As conclusões finais obtidas foram apresentadas no capítulo 6 juntamente com as perspectivas de desenvolvimentos futuros e aprimoramentos para o presente trabalho.

2. PANORAMA SOBRE SISTEMAS DE CONVERSÃO SOLAR FOTOVOLTAICA

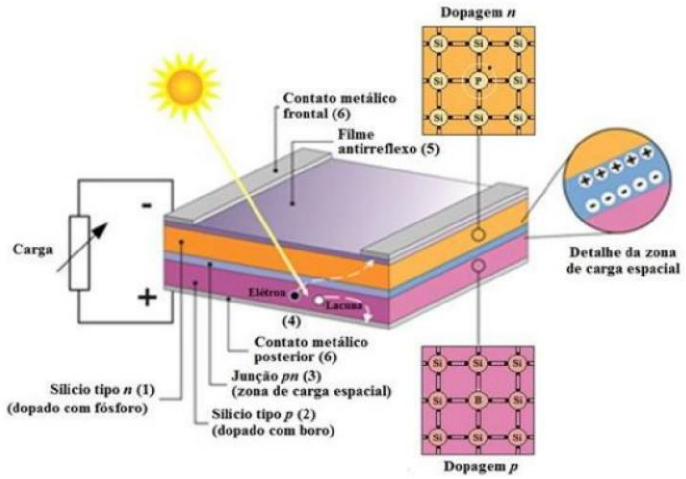
A seguir são apresentados os principais conceitos que envolvem a geração de energia através de conversão solar fotovoltaica.

2.1 Células e Módulos Fotovoltaicos

O princípio de funcionamento das células fotovoltaicas é baseado no efeito fotovoltaico, o qual é responsável pela conversão direta de energia solar em energia elétrica. O efeito fotovoltaico ocorre devido à excitação dos elétrons de materiais semicondutores quando estes absorvem fótons [1]da luz solar, resultando em uma diferença de potencial na estrutura desses materiais (BESSO, 2017; CRESESB, 2008).

A composição das células fotovoltaicas, conforme a Figura 2, consiste em materiais semicondutores, principalmente silício e germânio, que são materiais cuja banda de valência é separada por um vão de banda (*band gap*) da banda de condução. A banda de valência é a última banda onde os elétrons estão conectados ao átomo e o termo '*band gap*' refere-se à diferença de energia entre o topo da banda de valência e a parte inferior da banda de condução. Os elétrons são capazes de pular de uma banda para outra (MANUAL DE ENGENHARIA PARA SISTEMAS FOTOVOLTAICOS, 2014).

Figura 2 - Efeito fotovoltaico.



Fonte: CEPEL-CRESESB, 2014.

[1] Fóton é uma partícula subatômica de luz ou outra radiação eletromagnética, que é usada como uma unidade de energia eletromagnética.

Os semicondutores que compõem as células FVs são dopados de tal forma que um material passe a ter elétrons em excesso em suas ligações (material tipo N), enquanto o outro passa a apresentar um acúmulo de lacunas (material tipo P). Na junção dos materiais tipo N e P forma-se uma região, chamada zona de depleção, na qual se forma um campo elétrico inibindo a difusão de cargas entre os dois materiais (CEPEL-CRESESB, 2014).

Ao expor a junção P-N aos raios solares, os fótons absorvidos pela mesma e que tenham energia superior à do *bandgap*, provocam o deslocamento de elétrons da banda de valência para a banda de condução, formando pares de elétrons-lacunas. Se os fótons portadores são gerados na região de depleção, eles são então separados pelo campo elétrico resultante, que provoca a aceleração dos elétrons e lacunas em direção oposta. As lacunas são aceleradas para o lado P e os elétrons para o lado N, resultando na formação de uma corrente através da junção (MENDONÇA, 2017).

Entre as tecnologias de células fotovoltaicas disponíveis no mercado, destacam-se as de silício policristalino e de silício mono cristalino, classificadas como células fotovoltaicas de primeira geração. As células de silício policristalino são construídas a partir de um processo de fundição do silício em estado bruto e depois resfriado, ocorrendo a formação de inúmeros cristais que serrados em blocos, dão origem às células.

As células de silício mono cristalino, por sua vez, são fabricadas a partir de um único cristal com orientação definida e imerso em silício fundido, produzindo lingotes de vários metros de comprimento. Devido à distinção de pureza do silício na estrutura, as células de silício mono cristalinas possuem uma eficiência na faixa de 13 a 18%, valor 2% maior que nas células policristalinas, porém requer um processo de fabricação mais rigoroso resultando em aumento do custo dessa tecnologia.

As células fotovoltaicas de segunda geração, conhecidas como tecnologia de filme fino são menos sensíveis a temperaturas elevadas, aos efeitos do sombreamento na eficiência de conversão e, portanto, possuem desempenho superior às células de primeira geração quando expostas à radiação difusa (ORTEGA, 2013).

A célula de Silício Amorfo (a-Si) é um exemplo das tecnologias de filme fino. Sua eficiência está entre 6 e 9%, valor que sofre redução com o tempo devido à degradação causada pela luz, limitando a vida útil do painel em cerca 10 anos. Uma vantagem dessa tecnologia está na maior possibilidade de aplicação devido a sua estrutura flexível e por apresentar menor custo em seu processo de fabricação (NAKANO, 2017; NOVAK, 2016; PEROZA, 2015).

A principal distinção entre as diferentes tecnologias baseadas no uso de silício está relacionada principalmente às características de rendimento e custo, conforme descrito na Tabela 1.

Tabela 1-Eficiência e custo de diferentes tecnologias de células fotovoltaicas.

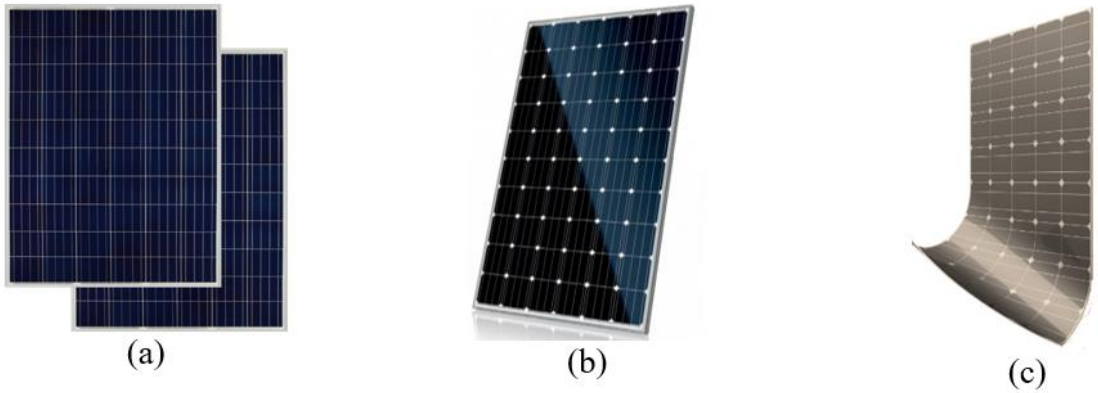
Tipo de Célula	Eficiência (%)			Custo ($/Wp)
	Teórico	Laboratório	Comercial	
Silício de cristal simples	30,0	26,7	12 a 14	1 a 2
Silício policristalino	25,0	22,3	11 a 13	0,6 a 1,2
Silício amorfo	17,0	4 a 7	3 a 5	-

Fonte: adaptado de ALBUQUERQUE (2017) e OLIVEIRA (2019).

Outra tecnologia de filme fino são as células de Telureto de Cádmio (TeCd), que possuem eficiência em torno de 9%. Há também as células de Disseleneto de Cobre e Índio (CIS), porém o uso do Índio em sua composição torna alto o custo dessa tecnologia (ORTEGA, 2013; VALADARES, 2019).

As células fotovoltaicas podem ser interligadas para obter valores de tensão desejados na saída do sistema FV, formando estruturas maiores chamadas de módulos fotovoltaicos. As Figuras 3 ilustra módulos FVs com as tecnologias de silício policristalino, mono cristalino e filme fino.

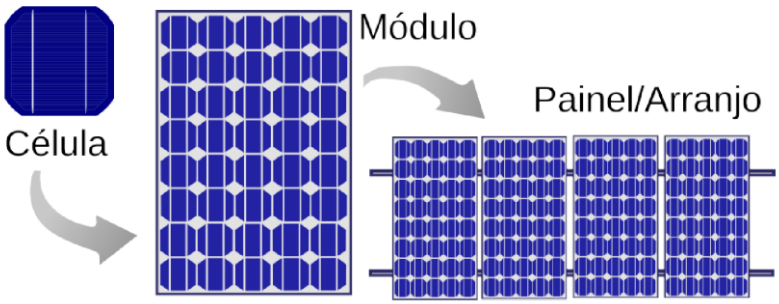
Figura 3 - Módulos fotovoltaicos (a) Si policristalino; (b) Si mono cristalino; (c) filme fino.



Fonte: http://blog.minhacasasolar.com.br/tipos-de-paineis-solares/ (2018), https://www.eet.eng.br/evolucao-da-celula-fotovoltaica/ (2019)

Os módulos, por sua vez, podem ser interligados para construir arranjos (painéis) fotovoltaicos, como representado na Figura 4.

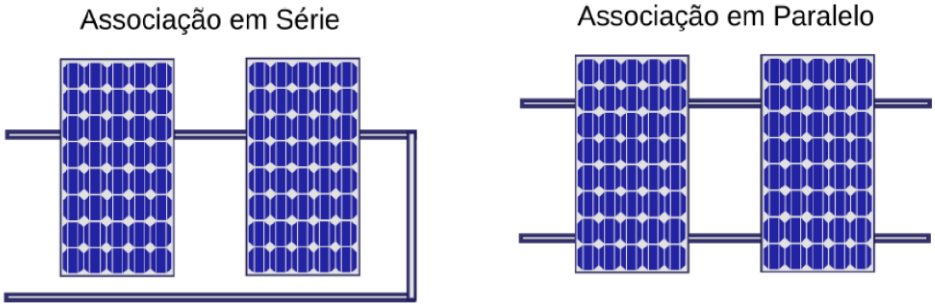
Figura 4- Estrutura de módulos e painéis fotovoltaicos.



Fonte: ALBUQUERQUE (2017)

De forma a obter a tensão e corrente necessária, os módulos podem ser conectados em série, quando se requer valores de tensão mais elevados ou paralelo, para uma corrente mais elevada, como representado na Figura 5. Os módulos podem ainda combinar os dois tipos de conexão (VALENTE, 2011).

Figura 5- Associação em série e em paralelo de módulos fotovoltaicos.



Fonte: ALBUQUERQUE, 2017.

2.2 Parâmetros para Avaliação de Desempenho de Sistemas Fotovoltaicos

Normalmente, o módulo fotovoltaico é caracterizado pela potência de pico (W_p). Entretanto, alguns cuidados devem ser tomados com relação à aplicação e condições ambientais do local de instalação do módulo, uma vez que a potência de pico é determinada sob condições-padrão de ensaio (STC-*Standard Test Conditions*), as quais considera a temperatura da célula de 25ºC e irradiância solar de 1000 W/m² (SOUZA et Al, 2016; ALBUQUERQUE, 2017).

A potência máxima (P_{max}) é o parâmetro determinante na escolha do módulo, porém existem outros aspectos importantes que caracterizam estes dispositivos. A corrente de curto-

circuito (I_{sc}) é a corrente máxima que um dispositivo fotovoltaico pode fornecer sob determinadas condições de temperatura e radiação quando a tensão é nula. A tensão de circuito aberto (V_{oc}) é a máxima tensão que o dispositivo fotovoltaico pode entregar sob determinadas condições de temperatura e radiação, sendo a corrente nula. Em ambos os pontos de operação, a potência é nula.

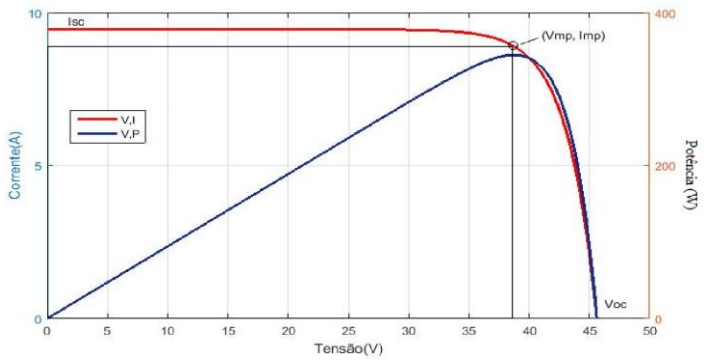
Outro parâmetro relevante é o Fator de Forma (FF), que consiste na razão entre a máxima potência do módulo (P_{max}) e o produto da tensão de circuito aberto pela corrente de curto circuito. O fator de forma depende da tecnologia usada. Para as células de Silício, FF encontra-se entre 80,9% e 82,8% (CEPEL-CRESESB, 2014).

A eficiência (η) dos módulos indica quão efetivo é o processo de conversão fotovoltaica e é dada pela relação entre a potência (W_p) produzida pelo módulo nas condições STC e potência da energia solar incidente.

Embora relevantes, as informações de I_{sc} e V_{oc} pouco têm a contribuir sobre o real comportamento do módulo, principalmente sobre sua potência real. Para melhor representar o comportamento do módulo, é traçada uma curva I-V que fornece um ensaio mais completo para determinar suas características. Para a obtenção da curva, o módulo é submetido às condições padrão de ensaio e registrados pares ordenados de tensão e corrente que são utilizados para a obtenção da curva característica I-V do módulo, como mostrada na Figura 6.

A Figura 6 também mostra outra representação que fornece informações sobre o módulo é a curva P-V de potência em função da tensão, onde se identifica o ponto máximo de potência. O ponto de máxima potência corresponde a um ponto da curva I-V, onde estão os valores de tensão de máxima potência (V_{mp}) e corrente de máxima potência (I_{mp}), respectivamente (BESSO, 2017; CEPEL-CRESESB, 2014; PEROZA, 2015).

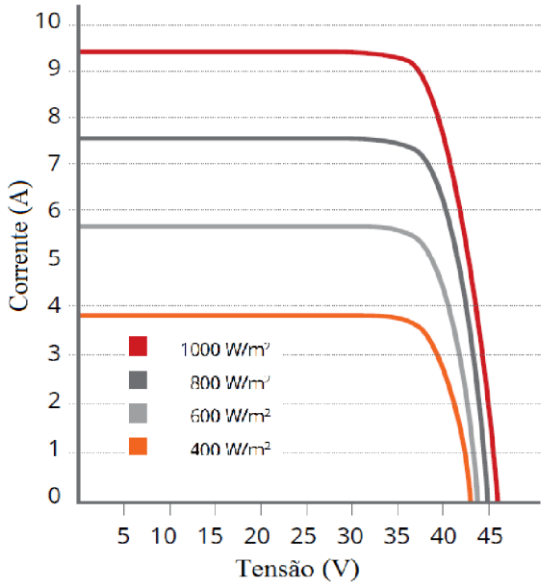
Figura 6 - Curvas I-V e P-V de módulo fotovoltaico.



Fonte: CEPEL-CRESESB, 2014.

O desempenho dos módulos fotovoltaicos é afetado por fatores ambientais como a irradiância solar incidente e a temperatura do ar. Em relação à irradiância solar, a corrente foto gerada é proporcional à radiação absorvida pelas células. A corrente de curto-circuito cresce linearmente em função da irradiância, e quanto à tensão, esta sofre variação de forma logarítmica em relação à variação da irradiância. A Figura 7 mostra as curvas I-V geradas para um módulo fotovoltaico de 72 células de silício cristalino, variando-se a irradiância e mantendo a temperatura de célula constante em 25°C (BESSO, 2017; LOPES, 2013).

Figura 7-Efeito de variação da irradiância solar.



Fonte: Datasheet módulo CS6U Canadian Solar.

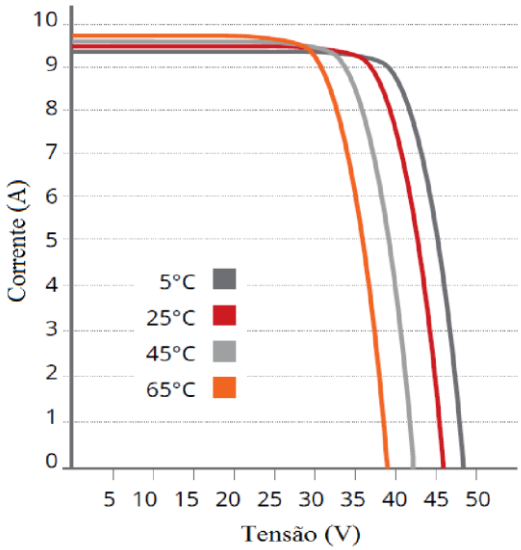
As características elétricas dos módulos também são influenciadas pela temperatura do ar, uma vez que a temperatura do módulo é dada em função desta. A Figura 8 mostra as curvas I-V geradas para um módulo de 72 células de silício cristalino, mantendo-se a irradiância constante a 1000W/m² e variando a temperatura (PEROZA, 2015; BESSO, 2017; LIMA, 2014).

A elevação da temperatura do módulo resulta em uma significativa queda na tensão gerada. Já a corrente apresenta um pequeno aumento, porém não supre a perda causada pela redução da tensão. Desta forma, o aumento da temperatura provoca redução da potência máxima fornecida pelo sistema.

É relevante avaliar os parâmetros que interferem na temperatura dos painéis fotovoltaicos. A velocidade do vento afeta diretamente no resfriamento destes dispositivos, sendo, portanto, um fator a avaliar uma vez que seu impacto geralmente é negligenciado nos

cálculos e análise encontradas na literatura. Além disso, a abordagem padrão mais comumente utilizada para estimar/prever a temperatura de operação das células não inclui o efeito de resfriamento causados pelos ventos, sendo limitada a medidas da temperatura do ar ambiente e da irradiação solar sobre o plano do módulo fotovoltaico (REGES, 2017).

Figura 8- Efeito de variação da temperatura.



Fonte: Datasheet módulo CS6U da Canadian Solar.

2.3 Sistemas Fotovoltaicos conectados à rede elétrica (*On-grid)* e isolados *(Off-grid)*

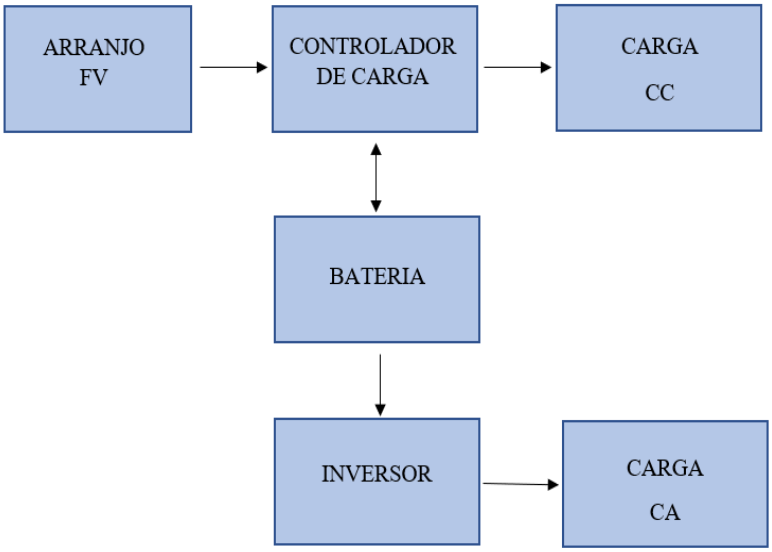
Existem tipicamente três configurações para os sistemas fotovoltaicos: sistemas isolados ou desconectados da rede (*off-grid*); sistemas conectados à rede, ou sistemas *on-grid*; sistemas híbridos que possuem características dos sistemas *on-grid* e *off-grid*. A seguir, são descritas as principais características de cada configuração.

2.3.1 *Sistemas isolados (off-grid)*

Sistemas *off-grid* ou autônomos são sistemas desconectados de uma rede hospedeira, que utilizam diversos equipamentos para entrega de energia elétrica em cc ou ca como ilustrado na Figura 9. Em sistemas *off-grid* normalmente a energia gerada é armazenada em um banco de baterias para uso posterior, nos horários em que não há sol. Os principais componentes de um sistema *off-grid* são os painéis fotovoltaicos, controladores de carga, baterias e inversores. Sua aplicação é mais indicada para abastecer a demanda de energia elétrica em

locais remotos. Além disso, o alto custo do banco de baterias e controladores de carga em sistemas *off-grid* colaboram para a limitação de seu uso (OLIVEIRA, 2019).

Figura 9 - Diagrama do sistema fotovoltaico *off-grid.*



Fonte: Adaptado de ASOWATA et al. (2014)

Os componentes dos sistemas *off-grid* são:

- Arranjo FV: conjunto de módulos fotovoltaicos onde a energia elétrica é gerada;
- Baterias: armazenam energia possibilitando que a energia seja utilizada em momentos em que não há produção de energia por parte do arranjo FV;
- Controlador de carga: garante o correto armazenamento de energia nas baterias, evitando sobrecargas e descargas profundas e com isso aumentando a vida útil das baterias. O controlador de carga inclui um conversor para rastreamento da máxima potência e para alimentação de cargas cc;
- Inversores: convertem a corrente contínua (cc) para a corrente alternada (ca), viabilizando que a energia seja utilizada nos equipamentos elétricos comumente utilizados na rede elétrica.

Desta forma, a topologia do sistema *off-grid* possibilita que este alimente tanto cargas cc quanto cargas ca.

2.3.2 Sistema conectados à rede elétrica (on-grid)

Os sistemas *on-grid* ou *grid-tie*, são sistemas conectados à rede elétrica. Em sistemas *on-grid,* quando há excedente de produção de energia, o excesso é enviado para a rede. Desta forma, a rede funciona como armazenamento de energia quando a geração for maior que o consumo. A Figura 10 ilustra um sistema FV *on-grid*.

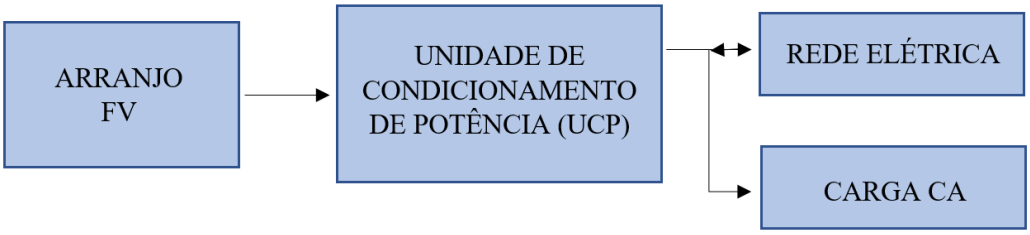
O usuário paga para a concessionária de energia quando consome mais energia do que gera ou recebe créditos caso a produção seja maior que o consumo. Os sistemas *on-grid* podem ser classificados em geração centralizada e geração distribuída, esta última dividindo-se quanto à potência em: microgeração e minigeração (DUAIK, 2019). A geração distribuída se diz ser microgeração quando a potência instalada for de até 75 kW, ou minigeração quando a potência instalada do sistema estiver entre 75 kW e 5 MW (ANEEL, 2015).

Figura 10 - Sistema de geração fotovoltaica *on-grid.*

Fonte: DUAIK, 2019.

De forma simplificada, o sistema *on-grid* pode ser representado conforme o esquema mostrado na Figura 11.

Figura 11- Esquema sistema *on-grid.*



Fonte: Adaptado de Macêdo, 2006.

Em sistemas *on-grid* é utilizado um inversor $cc - ca$, também denominado Unidade de Condicionamento de Potência (UCP) cuja função é transformar a corrente contínua gerada em corrente alternada, viabilizando a injeção da energia gerada na rede elétrica pela operação adequada do sistema com a rede elétrica (sincronismo).

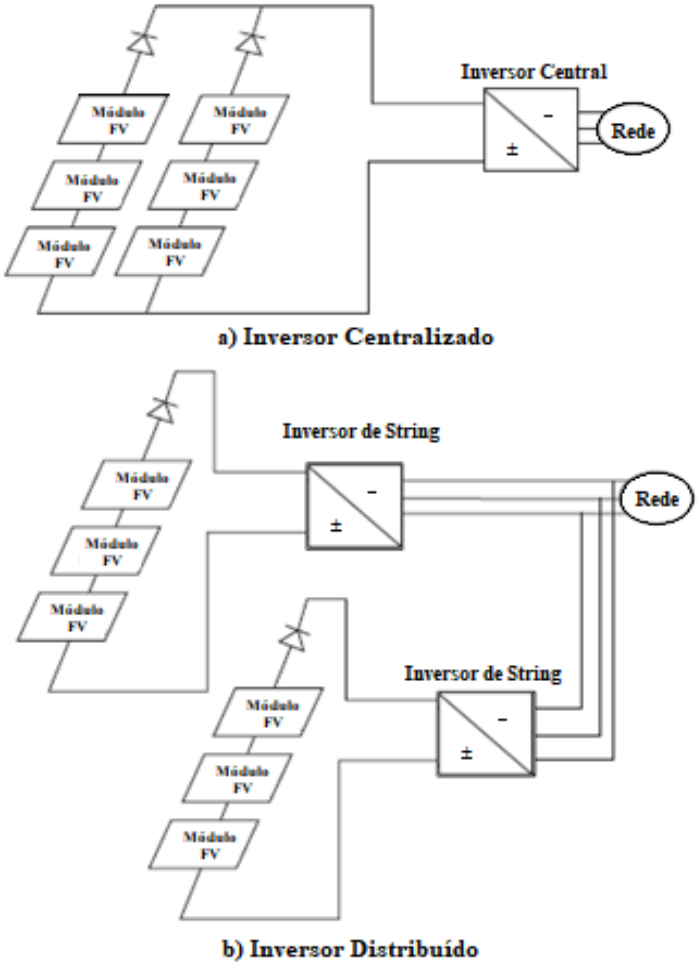
Basicamente, os componentes da UCP têm a função de:

- realizar seguimento do ponto de máxima potência (MPPT), de forma a otimizar a produção de energia;
- realizar a modulação por largura de pulso (PWM) para conversão de corrente CC em CA;
- interface com a rede e monitoramento.

A topologia do sistema em relação ao layout do inversor pode ser com inversor centralizado ou com inversores distribuídos; neste último caso, tem-se um inversor para cada agrupamento de módulos, denominado de *string* ou fileira, como mostra a Figura 12. O layout com inversor centralizado, mostrado na Figura 12 a), oferece um controle de operação centralizado, custos reduzidos e maior simplicidade, porém é menos tolerante à falha. O layout com inversor distribuído, mostrado na Figura 12 b), proporciona fácil substituição e menor impacto na geração ao ocorrer uma falha em uma *string*. Entretanto, por necessitar de mais inversores de menor potência, os custos por kW_p instalado são maiores (KRÖGER-VODDE et al, 2010; ARAÚJO et al, 2016).

No presente trabalho, o sistema fotovoltaico monitorado pelo sistema desenvolvido, trata-se de um sistema *on-grid* com módulos fotovoltaicos a Si-policristalino e inversores distribuídos, cujas demais características são descritas nas seções a seguir.

Figura 12 - Layout de sistema FV com inversor centralizado (a) e com inversores distribuídos (b).



Fonte: Adaptado de CHAAR, 2015.

No presente trabalho, o sistema fotovoltaico monitorado pelo sistema desenvolvido, trata-se de um sistema *on-grid* com módulos fotovoltaicos a Si-policristalino e inversores distribuídos, cujas demais características são descritas nas seções a seguir.

2.4 Sistemas Fotovoltaicos Instalados em Campus Universitários

Nos campi das universidades, os gastos significativos com consumo de energia elétrica têm impulsionado a busca por alternativas para diversificação do suprimento visando a redução de custos. Entre as soluções disponíveis, a geração solar fotovoltaica pode representar uma alternativa viável. Além de proporcionar redução da dependência da distribuidora de energia,

redução dos custos com energia e disseminação do uso de geração de energia elétrica através de fonte renovável, um sistema fotovoltaico nos *campi* universitários também representa um recurso relevante para o desenvolvimento de pesquisa relacionada à eficiência energética, qualidade da energia elétrica, microrredes, entre outras áreas relacionadas à geração de energia elétrica.

No âmbito das universidades públicas, a instalação de sistemas de geração fotovoltaica pode partir de diversas iniciativas, com investimento por agências de fomento à pesquisa ou órgãos públicos, por recurso próprio ou em parceria com empresas do ramo energético.

No Brasil, várias universidades públicas federais contam com geração distribuída solar fotovoltaica. A Universidade Federal do Espírito Santo (UFES) possui uma das maiores usinas de geração fotovoltaica dentre as instituições públicas federais, com 5.441 kW_p de capacidade instalada, com mais de 17.000 painéis fotovoltaicos (CANAL SOLAR, 2020).

A Universidade Federal da Grande Dourados (UFGD), no estado do Mato Grosso, possui uma usina de minigeração com capacidade instalada de 1.125 kW_p. Os recursos para aquisição do sistema foram provenientes do Ministério da Educação (PORTAL SOLAR, 2019).

Como forma de incentivar o uso eficiente da energia em todos os setores da economia, a ANEEL lança mão de editais de chamada pública para selecionar Projetos Prioritários que demonstrem a importância e a viabilidade econômica de melhoria da eficiência energética de equipamentos, processos e usos finais de energia. Estes projetos têm a finalidade de testar, incentivar ou definir ações de destaque como política pública para incrementar a eficiência energética no país (ANEEL 2016).

Na Universidade Federal do Paraná (UFPR), foi inaugurada em 2020 a maior usina de geração fotovoltaica instalada em prédio público. A usina possui 1.166 kW_p de capacidade instalada e geração anual de 1.300 MWh com estimativa de economia anual de cerca de R$ 1,5 milhão. O investimento para implantação da usina partiu da Companhia Paranaense de Energia (Copel) através do Programa de Eficiência Energética, uma iniciativa da ANEEL que tem como objetivo promover projeto de eficiência energética (UFPR, 2020).

A Universidade Federal do Ceará (UFC) também possui um complexo de usinas de geração fotovoltaica localizado no Campus do Benfica. O sistema é composto por 1700 módulos fotovoltaicos com capacidade de produção de energia de 970 MWh/ano. O Grupo de Redes Elétricas Inteligentes (GREI), no Departamento de Engenharia Elétrica da UFC, projetou, desenvolveu e instalou uma microrrede formada por três sistemas solar fotovoltaicos

monofásicos de 2 kW_p cada. O sistema foi financiado pelo CNPq em projeto de pesquisa (UFC, 2020).

O Instituto Federal do Ceará (IFCE) possui instalada três plantas de energia solar fotovoltaicas no campus de Maracanaú, que juntas possuem potência instalada de 8,2 kW_p.

A Universidade da Integração Internacional da Lusofonia Afro-Brasileira (UNILAB), no Ceará, em 2016, através de Chamada Pública da ANEEL em parceria com a empresa Enel Distribuição do Ceará, teve aprovado o Projeto Prioritário de Eficiência Energética e Minigeração de Energia Fotovoltaica da UNILAB para implantação de usina de minigeração solar fotovoltaica com potência instalada de 254,2 kW_p. A usina é descrita em maiores detalhes na próxima subseção.

2.5 Usina de Minigeração do Campus das Auroras – UNILAB – CE

O Projeto Prioritário de Eficiência Energética na UNILAB-CE possui duas vertentes: o *retrofit* de iluminação do Campus das Auroras e a implantação de uma usina de minigeração Fotovoltaica, ambos proporcionando uma economia anual de 640 MWh/ano.

A usina de minigeração fotovoltaica é composta de 762 módulos de silício policristalino de 330 W_p cada e 10 painéis de 275 W_p (Árvore Solar) da fabricante Canadian Solar, totalizando uma potência instalada de 254,2 kWp. A Figura 13 mostra uma parte da usina de minigeração. Os dados dos módulos FV da usina estão descritos na Tabela 2.

Figura 13– Usina de Minigeração do Campus das Auroras- UNILAB.

Fonte: Arquivo próprio.

Tabela 2-Características STC e NMOT do módulo CS6U-Canadian Solar. Fonte: Datasheet do fabricante.

Parâmetro	Variável	Grandeza STC	Grandeza NMOT
Potência máxima	$P_{mp}\ (W)$	330	243
Tensão em MP	$V_{mp}\ (V)$	37,20	34,20
Corrente em MP	$I_{mp}\ (A)$	8.88	7,10
Tensão de circuito aberto	$V_{oc}\ (V)$	45,60	42,50
Corrente de curto-circuito	$I_{sc}\ (A)$	9,45	7,63
Coeficiente de temperatura de I_{sc}	$k_I\ \left(\frac{A}{°C}\right)$	0,05	0,05
Coeficiente de temperatura de V_{oc}	$k_V\ \left(\frac{V}{°C}\right)$	-0,31	-0,31

Fonte: Datasheet do CS6U-Canadian Solar

O projeto da usina consiste em seis arranjos de 127 painéis cada. Cada arranjo possui uma potência instalada de 41,28 kW_p. A cada arranjo da usina de minigeração está conectado um inversor solar da fabricante ABB, modelo PRO-33.0-TL (Figura 14). A energia gerada é injetada diretamente na rede e os créditos contabilizados são descontados na fatura de energia do Campus das Auroras. O sistema é *on-grid*, logo não dispõe de baterias para armazenamento de energia.

Figura 14– Inversores ABB da usina de minigeração.

Fonte: Arquivo próprio.

O sistema de aquisição de dados e supervisão em estação fixa e móvel da planta solar FV e da estação solarimétrica da usina de minigeração da UNILAB-CE é objeto de desenvolvimento e implementação deste trabalho de dissertação de mestrado.

2.6 Considerações Finais sobre o Capítulo

Neste capítulo foram apresentados mais detalhadamente os principais fatores climáticos e suas influências no desempenho de sistemas FV, bem como os principais parâmetros desses sistemas.

Desta forma, percebe-se a importância do monitoramento das variáveis climáticas e dos parâmetros de geração dos sistemas FV, uma vez que o acompanhamento destas grandezas permite verificar como desempenho da planta é influenciado por elas. Além disso, possibilita a identificação de fatores indesejados na geração e a intervenção de forma mais ágil. O monitoramento consiste em um recurso imprescindível para a concepção de modelos de análise de eficiência e qualidade da energia gerada.

No próximo capítulo são mostrados os principais aspectos dos sistemas de aquisição e monitoramento de dados, bem como as principais soluções encontradas na literatura.

3. SISTEMA DE AQUISIÇÃO, SUPERVISÃO E COMUNICAÇÃO DE DADOS

Um sistema de aquisição e monitoramento de dados é composto por vários estágios, como:

- Aquisição de dados;
- Processamento dos dados;
- Comunicação entre a plataforma de processamento e o computador;
- Armazenamento de dados;
- Supervisão e Gerenciamento dos dados.

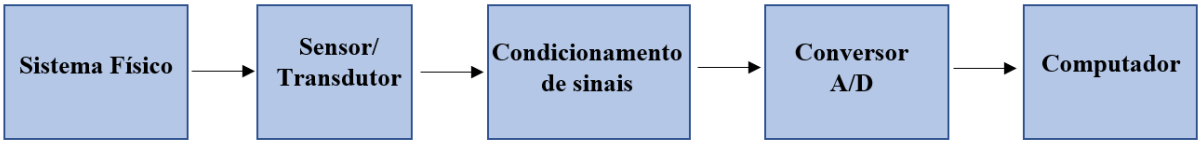
As principais técnicas utilizadas na aquisição, supervisão e comunicação de dados de sistemas fotovoltaicos encontradas na literatura são abordadas neste capítulo. O uso de plantas de conversão FV tem crescido na matriz elétrica brasileira e com isso se faz necessária a aplicação de sistemas de supervisão para acompanhar o desempenho técnico e econômico da operação desses sistemas. Ao longo do capítulo são abordados sistemas de aquisição de dados e supervisão de desenvolvimentos próprios e sistemas encontrados no mercado, podendo ser proprietário, bem como de código aberto.

Em sendo o capítulo voltado à descrição de sistemas supervisórios e seus componentes, vale salientar a diferença entre os termos ‘monitoramento’ e ‘supervisão’ que muitas vezes são usados como sinônimos. Entretanto, o ato de supervisionar implica em maior interação do que monitorar. A supervisão implica na responsabilidade de informar e orientar, enquanto o monitoramento está associado a observar sem instruir. Neste trabalho, o termo ‘supervisão’ será adotado uma vez que os sistemas desenvolvidos e implantados permitem a ação de monitorar, alarmar e controlar.

3.1 Sistemas de Aquisição e Monitoramento de Dados

O sistema de aquisição e monitoramento consiste em um conjunto de componentes interligados que executam uma atividade de medição de uma determinada grandeza ou de diversas grandezas e conversão dos dados medidos em um formato que possa ser visualizado e manipulado em um sistema computacional (REGES, 2017). A Figura 15 mostra um esquema resumido de um sistema de processamento de dados.

Figura 15- Esquema geral de um sistema de processamento de dados.



Fonte: Medium (2020)

Os sistemas de aquisição e monitoramento de dados podem permitir apenas a coleta dos dados no local de medição ou podem estar conectados a um sistema de transmissão, que envia, de forma automática e periódica, os dados coletados para um computador remoto.

De forma geral o processo de aquisição de dados pode ser dividido nas seguintes etapas (DUPONT, 2017):

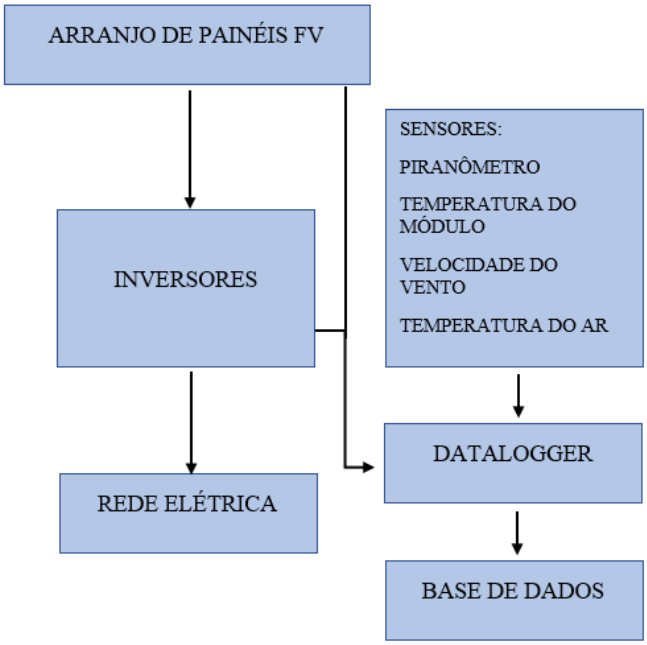
- Aquisição de dados através dos sensores analógicos e/ou digitais;
- Condicionamento dos sinais recebidos pelos sensores, de forma a garantir o nível adequado de tensão na entrada de um conversor analógico-digital (A/D) do microcontrolador. Como os sensores são projetados de forma a consumir a menor quantidade de energia possível, é necessário o uso de amplificadores de forma a amplificar o sinal e minimizar ruídos. Devido a limitação do número de portas de um PC, esta etapa geralmente é realizada pelo próprio sistema de aquisição de dados;
- Transmissão dos dados para um servidor de banco de dados e interface capaz de gerar gráficos e manipulá-los de acordo com as necessidades do usuário. O *software* de gerenciamento faz uso dos dados coletados transformando em informação de interesse do usuário.

O uso de sistemas de aquisição e monitoramento em usinas FV permite a verificação dos parâmetros em intervalos de tempo relativamente pequenos, possibilitando identificar anormalidades de forma eficaz e agilizar intervenções. Logo, é relevante a presença de um sistema de monitoramento em uma usina FV para análise de dados ambientais e de geração, como subsídio à tomada de decisão (GUSA, 2018). A Figura 16 mostra um esquema de um sistema de aquisição de dados no contexto de plantas FV.

Existem alguns modelos de controladores de carga que apresentam sistemas de aquisição e armazenamento de dados integrados, facilitando o monitoramento das instalações. Os inversores decentralizados, além de sua função principal que é a de converter a energia CC

em CA, também monitoram parâmetros elétricos do arranjo fotovoltaico, como tensão e corrente de saída, bem como os parâmetros elétricos do próprio inversor, fornecendo a potência da planta, a curva de geração, redução de CO2 e o histórico desses dados (VICTO, 2019). Caso estes dispositivos não possuam essa funcionalidade, equipamentos de medição e registro de dados podem ser instalados em diversos pontos do sistema fotovoltaico. Para isso, são usados transdutores de tensão e corrente e *data loggers* para registro dos dados. Os dados monitorados na maior parte das aplicações são tensão, corrente, potência ativa e energia ativa, tanto no lado cc quanto no lado ca (PINHO, 2014).

Figura 16- Esquema de sistema de aquisição de dados aplicado à sistema de energia solar fotovoltaica.



Fonte: Adaptado de THEE, 2014.

A escolha do tipo de dispositivo de monitoramento depende da topologia da planta FV. Se o sistema FV possui grandes dimensões, a utilização de redes de comunicação sem fio para realizar a transmissão dos dados de geração e parâmetros técnicos é extremamente importante, uma vez que sistemas FV de grande porte podem ser instalados em locais remotos. Já os sistemas de monitoramento de instalações FV de pequeno porte não necessitam do uso redes sem fio, podendo utilizar apenas de um *data logger* para registro dos dados e coleta local (REGES, 2017).

Desta forma, devido ao crescente uso da energia solar para geração de energia elétrica e a necessidade de se conhecer as variáveis que interferem no rendimento dos painéis fotovoltaicos, o desenvolvimento de sistemas de aquisição de dados de baixo custo para plantas de pequeno porte para obter dados ambientais e de performance dos equipamentos tem sido alvo de estudos no âmbito acadêmico.

A seguir são apresentadas algumas das principais plataformas para aquisição de dados cujo objetivo é formar uma base de conhecimento para orientar no desenvolvimento de uma plataforma para a usina FV UNILAB-CE.

3.1.1 *Plataformas de desenvolvimento e processamento de dados de baixo custo*

Tanto os dados de geração quanto os dados meteorológicos (radiação, temperatura e velocidade do vento) são obtidos por meio de sensores/transdutores. A aquisição e armazenamento dos dados das leituras destes sensores podem ser feitas por um sistema de coleta de dados comercial, o que em alguns casos pode ser inviável devido ao custo relativamente alto.

Na literatura, podem ser encontrados sistemas de aquisição de dados de baixo custo baseado em plataformas de prototipagem eletrônica bastante difundidas como o Arduino, o Raspberry Pi ou placas da Família ESP (ESP8266 e ESP32). Entre as principais aplicações em que estas plataformas podem ser utilizadas estão aquelas voltadas para controle, IoT (*Internet of Things*) e sistemas de monitoramento.

Raspberry Pi é um minicomputador de baixo custo capaz de executar um sistema operacional do tipo Linux, desenvolvido primariamente para o ensino de computação e utilizado em diversas aplicações de internet das coisas. O Raspberry Pi apresenta um conjunto de pinos de E/S (GPIO – *General Purpose Input Output*), onde é possível a conexão de sensores e atuadores (CARVALHO, 2019).

No contexto dos sistemas de aquisição de dados, o Raspberry Pi é encontrado na literatura em várias áreas de aplicação. Nos trabalhos de Souza (2019) e Oliveira (2018), o Raspberry Pi foi utilizado em protótipos de aquisição de dados ambientais utilizando sensores de baixo custo. No trabalho de Wofford (2019), um Raspberry Pi foi programado para operar em conjunto com um *data logger* CR1000 da fabricante Campbell Scientific para coletar, organizar e disponibilizar os dados ao público.

O Arduino, mostrado na Figura 17, é uma placa de microcontrolador desenvolvido como um recurso auxiliar para estudantes sendo depois desenvolvido comercialmente. As placas de Arduino estão disponíveis em várias versões como Uno, Nano e Mega e dispõem de tecnologia de baixo custo, que pode ser utilizada em aplicações baseadas em microcontrolador (NOVAK, 2016).

O Arduino Mega é a placa microcontrolada de mais alta performance dentre as placas da família Arduino. O Arduino Mega possui 54 pinos de I/O digitais, 16 entradas analógicas e 4 portas de comunicação serial. A alimentação da placa Arduino Mega pode ser feita tanto pela USB, como por uma alimentação externa. A linguagem de programação utilizada no Arduino é a linguagem C ou C++. O [2]IDE ou ambiente integrado de desenvolvimento do Arduino é disponibilizado gratuitamente no site do fabricante.

O sistema de aquisição de dados utilizados em sistemas fotovoltaicos de pequeno porte, como em Peroza (2015), Albuquerque (2017), Valente (2011) e Souza (2016), foram desenvolvidos na plataforma Arduino.

Figura 17- Arduino Mega 2560.

Fonte: Ortega, 2013.

Dependendo da aplicação, podem ser utilizados os [3]*shields* que conferem à placa original outras funcionalidades ampliando sua aplicabilidade. Entre os *shields* mais comuns estão Ethernet, *bluetooth*, motor etc.

[2] IDE – *Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software.
[3] *Shield* – placas de circuito que podem ser conectadas ao Arduino e expandindo suas funcionalidades.

Os projetos desenvolvidos por Gusa et al. (2018), Parthasarathy e Anandkumar (2016) e Fanourakis e MacCarthy (2017) também tratam de um sistema de aquisição de dados baseado na plataforma de prototipagem de baixo custo Arduino, que faz a aquisição de dados ambientais e de geração como tensão, corrente, temperatura da bateria e estado de carga da bateria de sistemas FV de pequeno porte.

Em Moura et al. (2018) se utiliza o conceito de IoT no desenvolvimento de um sistema que envia os dados de tensão gerada pelo módulo FV, a temperatura ambiente e a incidência de luz para um banco de dados na nuvem. O sistema de aquisição desenvolvido por Bogan (2017) também se baseia no conceito de IoT utilizando um website hospedado na nuvem como recurso de exibição dos dados. Ambos os sistemas foram desenvolvidos na plataforma livre e de baixo custo Arduino modelo Mega 2560.

Os trabalhos desenvolvidos por Anand et al. (2016) e Zubair e Ahmed (2015) destacam-se por apresentarem uma abordagem mais completa dos parâmetros necessários. Em Anand et al. (2016) foi desenvolvido um sistema de aquisição de dados com Arduino, utilizando sensores com precisão suficientes para obter curvas I-V e P-V com aplicação apenas para um único painel FV. Já o sistema de Zubair e Ahmed (2015) mede os parâmetros de tensão, corrente, umidade, potência, temperatura e intensidade de luz para uma matriz de painel solar.

Outra classe de microcontroladores são os da família ESP que oferece módulos de baixo custo e versáteis para várias soluções principalmente no contexto de aplicações IoT (*Internet of Things*). Entre as placas da família ESP destacam-se as séries ESP32 e ESP8266.

A placa de desenvolvimento NodeMcu, mostrada na Figura 18, é um *firmware* de código aberto, interativo, programável, de baixo custo, simples, inteligente e habilitado para Wi-Fi. A placa de desenvolvimento usa o microcontrolador ESP8266-12E, com a vantagem de poder ser programado diretamente no IDE do Arduino e é compatível com vários *shields* e sensores utilizados no Arduino, possibilitando utilizá-lo em uma diversidade de aplicações.

Figura 18- Placa NodeMcu ESP8266.

Fonte: Robocore (2017).

Em Ricardo (2018), Melo (2018) e Cunha e Batista (2018), o NodeMcu ESP8266 é utilizado em sistemas de monitoramento no contexto de IoT.

Diante do exposto, nota-se que no contexto de sistemas de aquisição de dados de sistemas fotovoltaicos baseado nas plataformas supracitadas, a grande maioria trata-se de aplicações para sistemas FV de pequeno porte (microgeração).

Para sistemas fotovoltaicos de maior porte, a partir da minigeração, não foram encontrados sistemas de aquisição desenvolvidos e de fato implantados. Nesse contexto, destaca-se o trabalho de Corrêa (2019), no qual um *software* supervisório baseado em SCADA, foi desenvolvido para monitorar uma usina solar fotovoltaica de potência de 400 kW. O trabalho de Etamaly et al (2020) projeta uma arquitetura de rede de comunicação para o monitoramento remoto de usinas fotovoltaicas de grande porte baseada na norma IEC 61850 utilizando o simulador OPNET.

A placa de desenvolvimento NodeMcu ESP8266 e o microcontrolador Arduino Mega 2560 foram utilizados neste trabalho para desenvolvimento de sistema de aquisição, supervisão e comunicação móvel da usina de minigeração FV do campus das Auroras.

3.2 Desenvolvimento de *Softwares* de Monitoramento

As variáveis ambientais podem ser monitoradas em tempo real por sistemas que disponibilizam na rede as suas mudanças (*online*) ou apenas registram em seus bancos de dados pessoais (*offline*). O sistema escolhido depende diretamente da aplicação e do investimento necessário para sua utilização (REGES, 2017).

A visualização das medições, bem como a interpretação dos dados é indispensável ao monitoramento FV. O monitoramento *online* das variáveis de um sistema FV é uma boa alternativa para unidades descentralizadas que necessitam da aquisição de dados para verificação do desempenho do sistema, pois além do acesso remoto por parte do usuário permitem uma visualização mais atrativa dos dados. Desta forma, os sistemas que funcionam *online* dependem diretamente de aplicações web, módulos Wi-Fi e servidores (REGES, 2017).

O sistema desenvolvido por Halmeman (2014) também foi desenvolvido em *hardware* livre (Arduino) onde os dados eram transmitidos para uma central de gerenciamento via rede sem fio em protocolo ZigBee. Uma página na *web* foi desenvolvida para monitorar o

funcionamento da instalação FV onde também é possível realizar o *download* dos dados em formato CSV. A eficiência na transmissão e armazenamento dos dados foi de 96%.

O trabalho de Zago (2018) consiste em um sistema de baixo custo para monitoramento da geração de energia solar envolvendo o conceito de IoT. O *software* desenvolvido recebe os dados da rede de sensores sem fio, armazena e envia os dados para a nuvem, utilizando o serviço Ubidots. O Ubidots possibilita ainda o acesso aos dados de geração por meio de um aplicativo de Smartphone.

Em Syafii et al. (2017), uma página na internet para monitoramento FV foi construída utilizando servidor de programação PHP. Os dados gravados dos sensores são enviados para a central de gerenciamento por meio do sistema de rede sem fio ZigBee podendo ser acessados remotamente pelo usuário.

O sistema desenvolvido por Shariff et al. (2015) também consiste em um site projetado usando o *software* Adobe Dreamweaver em linguagem HTML. O site permite aos usuários observar os dados em tempo real dos parâmetros monitorados, bem como a potência e a energia geradas no sistema.

Já Ruschel (2015) realizou modificações no *software* FVCONECT, que simula o funcionamento típico de um sistema em determinado local baseando-se nos dados climáticos médios da região, para criar o SPV- SuPerVisor que através de entradas das condições meteorológicas instantâneas fornecidas, calcula os dados das condições de operação esperadas, como tensão, corrente e potência.

A aplicação de *software* previamente elaborados, como fizeram Ribeiro (2017) e Kandimalla e Kishore (2017), é uma boa alternativa para utilização no monitoramento de parâmetros, pois possuem sistemas padronizados de comunicação que facilitam sua aquisição. Além disso, por ser *open source*, seu uso acaba reduzindo os custos totais de um sistema, possibilitando ainda que as informações recebidas dos módulos de aquisição de dados possam ser exibidas em páginas na internet, viabilizando o gerenciamento e análise dos dados.

Em Ribeiro (2017) foi desenvolvido um sistema supervisório com o *software* livre ScadaBR para monitorar os dados de tensão, corrente e potência de dois conjuntos de placas FV, uma de silício policristalino e outra de silício mono cristalino. Foi criado um banco de dados que recebe os dados enviados pelo Arduino e uma interface gráfica para visualização e supervisão dos dados facilitando a interação com o usuário. Desta forma, o usuário pode ter acesso a gráficos relacionando as potências dos dois conjuntos de placas em função do tempo,

gráficos históricos dos dados de geração individual, histórico de dados e relatórios (RIBEIRO, 2017).

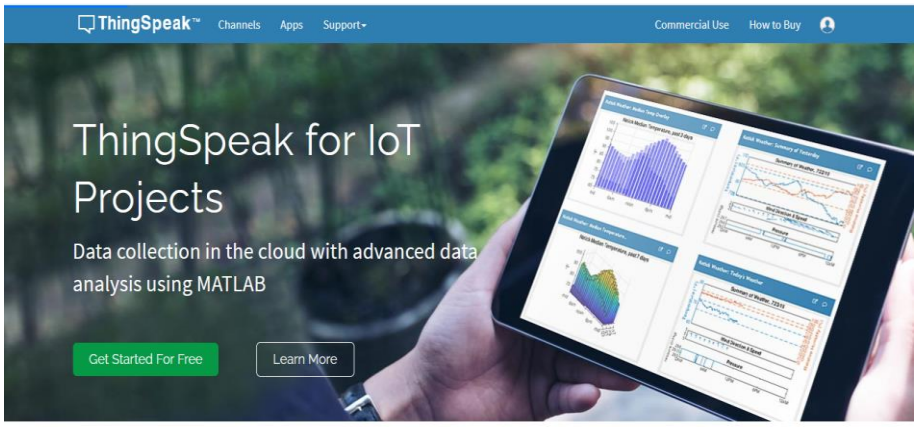
Casagrande e Severo (2018) e Barros Filho et al. (2019) também desenvolveram um *software* para monitoramento utilizando o ScadaBR de uma estação de coleta de dados ambientais desenvolvida a partir da plataforma Arduino.

Em Kandimalla e Kishore (2017), o autor utilizou a plataforma gratuita ThingSpeak para desenvolver um sistema de monitoramento para uma planta FV baseado em Arduino. O ThingSpeak consiste em uma plataforma de prestação de diversos serviços direcionados para a construção de aplicações de Internet das coisas (IoT). Ele permite a coleta de dados em tempo real, visualizando os dados recolhidos na forma de gráficos, tabelas e histogramas. Desta forma, os parâmetros de operação dos painéis solares são monitorados possibilitando a melhora dos aspectos de eficiência da planta FV.

No trabalho de Martins (2019), o Arduino é utilizado em conjunto com um módulo ESP 01 8266 é utilizado na aquisição de dados de um gerador FV de 20W e utiliza o ThingSpeak para realizar o monitoramento dos parâmetros do sistema.

O ThingSpeak® é uma plataforma analítica *open source* que fornece vários serviços voltados para aplicações de internet das coisas (IoT), que permite agregar, visualizar e analisar fluxos de dados, em forma de gráficos, salvos na nuvem em tempo real. Com a possibilidade de utilizar diversos recursos do *software* MATLAB® no ThingSpeak®, é possível realizar análise e processamento online dos dados conforme eles chegam, sem a necessidade da compra de uma licença. O Thingspeak permite realizar registros de informações fornecidas por dispositivos conectados à internet (THINGSPEAK HOME PAGE, 2020).

Figura 19- Layout da página inicial do ThingSpeak®.



Fonte: https://thingspeak.com/.

Os dados enviados para o ThingSpeak são armazenados em canais. A versão gratuita da plataforma permite criar até 4 canais com 8 campos para armazenar dados que pode ser utilizado para armazenar os dados de um sensor ou um dispositivo incorporado. Também possui 3 campos de localização que armazenam informações de latitude, longitude e a elevação e são muito úteis para rastrear um dispositivo em movimento (REIS, 2019).

3.3 Sistema Supervisório SCADA

Sistema Supervisório de Controle e Aquisição de Dados (*Supervisory Control and Data Aquisition* - SCADA) é definido como uma arquitetura de sistema de controle de processo que usa computadores, comunicação de dados em rede e interfaces gráficas homem-máquina (IHMs) para permitir um gerenciamento e controle supervisório de alto nível do processo. O sistema SCADA compreende elementos de *software* e *hardware*, que permite controlar processos local e remotamente, monitorar e coletar dados do processo em tempo real, interagir diretamente com sensores, válvulas, bombas, motores etc. por meio de *software* IHM, registrar eventos etc (MULLER, 2017; BONIFÁCIO, 2018).

Um sistema SCADA possui basicamente três funções (MULLER, 2017):

- Função de supervisão: realiza o monitoramento das variáveis do processo apresentando na forma de gráficos de tendência de variáveis analógicas e digitais, relatórios etc.
- Função de operação: abrange a ação direta sobre os atuadores permitindo enviar comandos como ligar/desligar equipamentos e sequência de equipamentos, mudança de modo de operação de equipamentos etc.
- Função de controle: Alguns sistemas possuem opções específicas para atuação automática sobre o sistema em determinadas situações pré-programadas de acordo com a necessidade e possibilidade de ter esse tipo de automatismo sobre o processo supervisionado (VIANNA, 2008).

Os sistemas SCADA são amplamente utilizados na indústria para controle supervisório e aquisição de dados de processos industriais na área petrolífera, química, alimentícia, metalúrgica e de geração de energia elétrica. Esses sistemas tornam disponíveis ao operador ou usuário as informações do processo e devem ser capazes de realizar ações e tomar decisões sobre o processo (BONIFÁCIO, 2018).

Os seguintes componentes estão geralmente presentes no sistema SCADA (BASU, SWAPAN, 2015):

- Dispositivos de interface de campo UTR (Unidade Terminal Remota): UTRs conectadas a sensores, transmissores e atuadores, convertem os sinais eletrônicos dos sensores para a linguagem de dados digital, utilizada para transmitir através do canal de comunicação para o sistema de supervisão. As UTRs também fazem a interface para converter os dados do sistema de supervisão em sinais eletrônicos necessários para os atuadores;
- Estação mestre ou servidor: computador ou rede de servidores utilizado para fornecer a interface (IHM) do operador para o SCADA, adquirir e processar dados e enviar comandos de supervisão;
- Interface Homem-máquina: se refere a interface de trabalho do operador e é bastante importante para o SCADA. A interface inclui *software* necessário para fornecer tendências, diagnósticos, apresentação de alarmes, acesso a banco de dados do sistema, programação de manutenção entre outros;
- Componente de *software*: o *software* da interface deve ser bem definido e projetado para que o SCADA tenha um bom desempenho. Vários componentes de *software* incluem: programa de automação, *software* de servidor, ferramenta IHM e de comunicação.

3.3.1 Softwares SCADA

Um *software* SCADA é um pacote de *software* constituído de um ambiente de desenvolvimento e um programa de execução. O ambiente de desenvolvimento inclui utilidades relacionadas com a criação e edição de janelas de aplicativos diversos e suas características (textos, desenhos, cores, propriedades de objetos, programas etc.). Já o programa de execução ou *runtime* permite executar a aplicação criada com o programa de desenvolvimento (para o usuário final é entregue como produto o *runtime* e a aplicação). Esse pacote também inclui os controladores ou *drivers* que permitem a comunicação do *software* SCADA com os dispositivos de controle da planta e com a rede de gestão da empresa (PENIN, 2007).

O *software* SCADA identifica os *tags* que são variáveis numéricas ou alfanuméricas envolvidas na aplicação, podendo executar funções computacionais (operações matemáticas, lógicas, com vetores ou *strings* etc.) ou representar pontos de entrada/saída de dados do processo que está sendo controlado. Nesse caso, correspondem às variáveis do processo real

(temperatura, nível, vazão etc.), se comportando como a ligação entre o controlador e o sistema. É com base nos valores das *tags* que os dados coletados são apresentados ao usuário (SILVA e SALVADOR, 2005).

Há no mercado uma série de *softwares* para supervisão e aquisição de dados, alguns deles são (MORAES, 2016):

- InduSoftWeb Studio da InduSoft;
- Mango Automation;
- Elipse SCADA;
- ScadaBR.

3.3.1.1 Indusoft Web Studio

O InduSoft Web Studio é um *software* de desenvolvimento que permite construir aplicações completas SCADA ou IHM para a indústria de Automação (SCHOLL, 2015; INDUSOFT, 2014).

Em Riel (2017), um supervisório foi desenvolvido com o Indusoft, utilizado para monitorar um conversor elétrico de baixo custo. Somessari (2019) também utilizou o Indusoft no desenvolvimento de um sistema automatizado para controle operacional de um acelerador industrial de elétrons.

3.3.1.2 Mango Automation

O *software* Mango Automation serviu como base para a criação do *software* ScadaBR. Entretanto, o Mango Automation engloba as funções presentes no ScadaBR e novas funcionalidades (ADAM, 2019).

O Mango apresenta diversos barramentos ou protocolos de comunicação, aplicação de sensores variados, construção de telas interativas, alertas via e-mail, utilização de *scripts* para algoritmos de controle e automação, alarmes etc. Além disso, suporta uma grande variedade de protocolos como Modbus, BACnet, OPC DA, Dallas 1-Wire, SNMP, SQL, HTTP, POP3, NMEA 0183, MBus, DNP3, OpenV, vmstat etc. (SCHOLL, 2015).

3.3.1.3 Elipse SCADA

O Elipse é um *software* robusto, sendo bastante utilizado em situações críticas como automação de usinas e subestações de energia elétrica, parques eólicos e gestão de linhas ferroviárias. A empresa fornece diversos produtos para atender todos os requisitos dos clientes,

começando pelo *software* ElipseE3 que é um sistema ideal para sistemas de missão crítica, fornecendo desde uma interface homem-máquina simples até complexos centros de operação.

No trabalho de Gercia e Muynarsk (2014) foi desenvolvido um sistema de controle e monitoramento de máquinas elétricas utilizando um microcontrolador Arduino e o Elipse SCADA.

Em Corrêa (2019) foi desenvolvido um *software* supervisório utilizando o ElipseE3, um dos produtos da Elipse, para realizar o monitoramento de uma usina solar fotovoltaica de potência de 400 kW via MQTT.

Ribeiro et al. (2020) implantou o ElipsePower em uma microrrede formada por três sistemas FV monofásicos de 2 kW cada, no Grupo de Redes Elétricas Inteligentes da UFC.

O software Elipse E3 é uma solução avançada de supervisão, permitindo visualizar e operar o sistema através de tablets, gerenciar alarmes da aplicação e smartphones e manipular grandes massas de dados.

O software Elipse apresenta vários recursos como módulos para gestão de alarmes e eventos, segurança e compactação na transmissão de dados, utilização de scripts e integração com bancos de dados comerciais como SQL Server e Oracle. Entretanto, a desvantagem do uso do Elipse, é a restrição imposta por ser um *software* pago, o que acaba limitando sua aplicação (BONIFÁCIO, 2018).

3.3.1.4 ScadaBR

O ScadaBR é um *software* supervisório de uso gratuito usado para desenvolvimento de aplicações de automação, controle e aquisição de dados. O ScadaBR pode ser executado em computadores com sistema operacional Windows e Linux a partir de um servidor de aplicações, sendo a escolha padrão o Apache Tomcat (Manual ScadaBR).

Por ser uma plataforma gratuita, possui a vantagem de suporte e documentação de fácil acesso. A interface principal do ScadaBR é bastante intuitiva tornando-o fácil de utilizar e pode ser iniciado nos principais navegadores disponíveis. O *software* oferece visualização das variáveis, gráficos, configuração dos protocolos, alarmes, construção de telas tipo IHM entre outras opções de configuração. Além disso, o *software* é compatível com grande parte dos protocolos de comunicação disponíveis como Modbus Serial, Modbus IP e outros (BONIFÁCIO, 2017).

Em CasaGrande (2018), o ScadaBR é utilizado para monitoramento de um sistema de aquisição de dados de variáveis climáticas. O sistema é composto por um sensor de radiação,

um anemômetro, um barômetro e um sensor de temperatura PT100 conectados a um circuito de condicionamento de sinal para adequar os sinais dos sensores aos níveis de leitura de um microcontrolador (Arduíno). Os dados são enviados via protocolo Modbus serial ao sistema supervisório desenvolvido no ScadaBR.

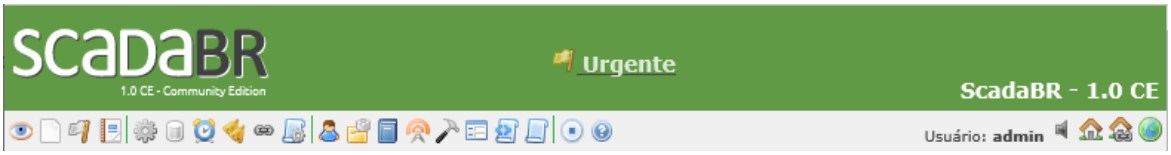
Em Ribeiro (2017), foi desenvolvido um programa supervisório no ScadaBR com o objetivo de monitorar o desempenho de duas plantas fotovoltaicas de 3kW baseadas em tecnologias distintas, com módulos de silício mono cristalino e policristalino. Feitosa e Mota (2013) também utilizaram o ScadaBR para monitorar os dados de corrente e tensão de um sistema FV residencial. Ambos utilizaram o Arduino para realizar a aquisição de dados.

Guimarães et al. (2014) desenvolveram um *software* de monitoramento da geração de um sistema fotovoltaico *off-grid* de utilizando o ScadaBR. O *hardware* foi desenvolvido na plataforma Arduino Mega de microcontrolador *Atmega*2560 com um circuito divisor de tensão e um CI LTSP 25-NP, onde através de um *trimpot* para regular a resolução dele obtém-se uma saída em tensão que representa a corrente circulando através dele.

O trabalho desenvolvido por CARDOSO (2016) trata de um sistema para interconexão de redes de sensores sem fio e internet utilizando sensores de baixo custo para monitoramento do nível de córregos e rios urbanos utilizando o SCADABR.

A Figura 20 mostra o cabeçalho da interface de usuário do ScadaBR. Os ícones mostram um resumo de sua funcionalidade.

Figura 20- Cabeçalho ScadaBR.



Fonte: ScadaBR.

Quanto ao armazenamento dos dados coletados, o ScadaBR suporta dois SGBD (Sistema de Gerenciamento de Banco de Dados), o MySQL e o Apache Derby. Este último é o banco utilizado na versão padrão do ScadaBR.

O ScadaBR suporta tipos de dados como: valores binários, que representam dois estados, p.ex. “ligado - desligado”; valores de estados múltiplos, que representa mais estados que o tipo binário como “ligado/desligado/desativado”; valores alfanuméricos, sequência de

caracteres; valores numéricos, representados como variável de ponto flutuante, por exemplo: temperatura e umidade; valores em imagens, com representações binárias de dados de imagens.

Um procedimento fundamental para a aplicação no ScadaBR é a configuração de *data sources* (fontes de dados), onde os dados são recebidos. Para configurar o *data source* é necessário que o ScadaBR suporte o protocolo de comunicação do dispositivo. Entre os protocolos permitidos estão Modbus Serial (RTU), Modbus TCP/IP além da possibilidade de consulta a banco de dados SQL.

Dentro dos *data sources* são criados e configurados os *data points* (pontos de dados). Os *data points* podem ser uma leitura de temperatura de um ambiente ou valores de controle que indicam o estado de ligado/desligado de um equipamento. Ao configurar o *data source* pode-se utilizar o recurso *point locator,* que permite encontrar os dados para um determinado ponto (SCADABR, 2017).

Após configurar *data sources* e *data points*, pode-se realizar o monitoramento do sistema de duas formas, através das *watch lists* que são listas dinâmicas de pontos e seus valores, conforme o período de atualização desejado, e os gráficos de informações históricas, dependendo da configuração do ponto (tipo de dado), ambos atualizados em tempo real. Além destes recursos pode-se criar representações gráficas para representar o sistema a ser monitorado, adicionando gráficos e valores dos data points sobre uma imagem de fundo (SCADABR).

3.4 Protocolo de Comunicação Modbus

O Modbus é um protocolo de comunicação de dados do tipo mestre/escravo desenvolvido pela Modicon Industrial Automation Systems, que atualmente faz parte da Schneider Eletric. Por se adequar facilmente a diversos meios físicos existentes e ser umas das soluções mais baratas em automação, sendo criado nos anos 70, o Modbus ainda é um dos protocolos mais utilizados em redes de comunicação em automação industrial, além de estar presente em aplicações como automação residencial e de navios, equipamento de laboratório, equipamentos fotovoltaicos etc. (CORREA, 2019).

O Modbus pode ser acessado através de padrões de meios físico como os conhecidos padrões RS-232, RS-485 e Ethernet TCP/IP. A comunicação é do tipo Mestre-Escravo, quando o dispositivo mestre envia uma solicitação para o escravo que responde.

O padrão RS-232 é usado em comunicação do tipo ponto a ponto, contemplando dois dispositivos na rede, um mestre e um escravo. A velocidade máxima do RS-232 está em torno de 115 Kbps e a distância máxima do cabeamento que conecta os dois equipamentos é de cerca de 30 m (BASU, SWAPAN, 2015).

O padrão TIA/EIA-485, popularmente conhecido como RS485, é comumente encontrado em equipamentos industriais por possuir características que permitem maiores possibilidades de aplicação que o RS-232. O cabeamento característico desse padrão é menos sensível a ruído e apresenta menor custo que o do padrão RS-232. O padrão RS-485 possui velocidade de comunicação de até 12 Mbps e o comprimento máximo dos dispositivos pode ser até 1200 m, permitindo até 32 dispositivos na rede.

As ligações em uma rede RS485 devem ser realizadas por um par trançado de condutores com seção mínima de 24AWG (0,2mm^2) e um condutor terra. Geralmente as conexões em uma rede RS485 são do tipo *half-duplex*, em que um único par de fios é utilizado na recepção e transmissão de dados que, portanto, não ocorrem simultaneamente. Já a conexão do tipo *full-duplex* permite que a transmissão e recepção de dados ocorra de forma simultânea, uma vez que possui um par trançado responsável pela transmissão e um para a recepção de dados. (NOVUS, 2020).

Uma reflexão em uma linha de transmissão é resultado de uma descontinuidade de impedância ao longo da linha que podem causar efeitos como o corrompimentos dos dados. Para minimizar estes reflexos nas extremidades do cabo RS485 é necessário colocar uma terminação de linha próximo a cada uma das extremidades do barramento. Desta forma, um resistor de terminação, geralmente de 120 ohm) deve ser conectado aos dois condutores da linha TX e RX, que no padrão RS-485 geralmente são nomeados como A e B, respectivamente. Entretanto, redes de curta distância (até 100m) e taxa de transmissão de até 19200 bps operam adequadamente sem a necessidade de resistores de terminação. Outra situação que pode provocar efeitos indesejados na comunicação é quando não há atividade de dados em um par RS485 as linhas não são acionadas e ficam vulneráveis a ruídos externos ou interferências. Para garantir que seu receptor permaneça em um estado constante, quando nenhum sinal de dados está presente, alguns dispositivos precisam polarizar a rede. Os dispositivos Modbus devem possuir especificação sobre a necessidade ou não de polarização de linha. (NOVUS, 2020; MAXIM INTEGRATED, 2001; JUNIOR 2018).

O padrão Ethernet possui taxa de transmissão de 100 Mbps ou até 10 Gpbs, com distância máxima na faixa de 100 a 200 metros. Esse padrão apresenta a vantagem de poder utilizar a comunicação *wireless* (CARVALHO, 2019; BASU, SWAPAN, 2015).

Vale ressaltar que quanto maior o comprimento da rede mais limitada fica a taxa de comunicação devido às interferências e resistência do cabeamento necessário.

3.4.1 Modbus TCP/IP

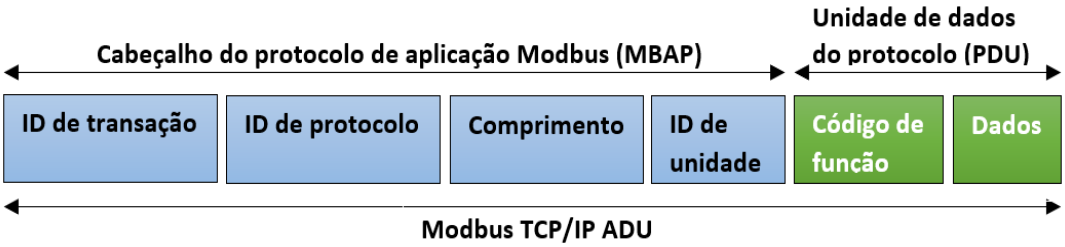
Dependendo do meio físico utilizado para a implementação do Modbus, existem procedimentos padrão que devem ser seguidos para que a comunicação entre os equipamentos ocorra corretamente.

O protocolo Modbus TCP/IP ou Modbus-TCP é o protocolo Modbus RTU com uma interface TCP (*Transmission Control Protocol*) que funciona em Ethernet. Devido à sua flexibilidade e ampla funcionalidade, esse modelo de referência também está em uma rede de controle. A função primária do TCP é assegurar que todos os pacotes de dados são recebidos corretamente, enquanto o IP (*Internet Protocol*) garante que as mensagens sejam direcionadas corretamente e encaminhadas. Logo, a combinação de TCP/IP trata-se apenas de um protocolo de transporte e não define o que o significado dos dados ou como os mesmos devem ser interpretados, esta é a função do protocolo de aplicação Modbus. Dessa forma, Modbus TCP/IP utiliza o padrão de rede TCP/IP e Ethernet (meio físico) para transmitir os dados da estrutura da mensagem Modbus, mostrada na Figura 21, entre dispositivos compatíveis (CALDIERE, 2016; VIEIRA, 2018).

O protocolo Ethernet é utilizado na interconexão de redes locais e é padronizado pela IEEE 802.3. O Ethernet é dividido em duas camadas (BASU, SWAPAN, 2015):

- Camada física: meio físico que realiza a comunicação entre os vários nós (computadores) da rede;
- Camada de enlace de dados: é subdividida em duas subcamadas, a subcamada MAC cujas funções são o encapsulamento de dados antes da transmissão, análise de pacote e detecção de erros após a transmissão e a subcamada LLC (*Logical Link Control*).

Figura 21- Estrutura da mensagem Modbus TCP/IP.



Fonte: Adaptado de Modbus (2019).

A arquitetura TCP/IP divide o processo de comunicação em quatro camadas básicas, que são:

- Camada de rede: consiste na camada física, ou seja, aquela composta pelo *hardware*. Entre as funções desta camada estão a verificação de erros de pacotes de dados recebidos, confirmação de dados e interface de rede com computador. Entre os protocolos deste nível estão o Ethernet e PPP (point-to-point);
- Camada de internet: é nessa camada que se posiciona o protocolo IP. Esta camada é responsável pela circulação dos pacotes de dados. O protocolo IP faz parte dos vários protocolos desta camada;
- Camada de transporte: nesse nível é efetivada a conectividade de rede. As funções desta camada são a verificação de erros, controle de fluxo e verificação de integridade do pacote. O protocolo TCP é um dos protocolos da camada de transporte;
- Camada de aplicação: nesse nível encontram-se aplicativos de rede e serviços que viabilizam interface para usuários. Alguns dos protocolos da camada de aplicação são o FTP e HTTP (BASU, SWAPAN, 2015).

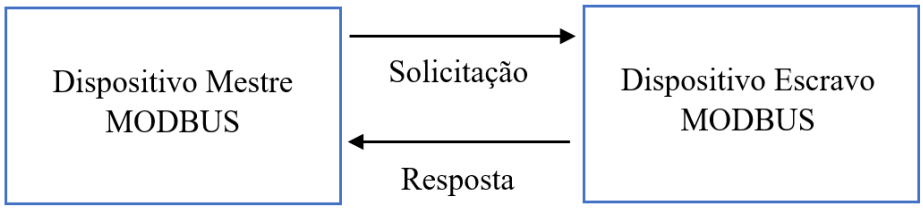
Em Corrêa (2019) os inversores de uma usina FV foram conectados via TCP/IP em um sistema supervisório desenvolvido em SCADA para monitoramento local.

3.4.2 Modbus RTU

O protocolo Modbus RTU é baseado na comunicação mestre-escravo, onde apenas o único dispositivo mestre pode inicializar a comunicação (*query*), enquanto os dispositivos escravos respondem enviando os dados solicitados pelo mestre ou realizando alguma ação

solicitada, como representado na Figura 22. Os equipamentos que possuem o padrão de meio físico RS-232 ou RS-485 geralmente podem utilizar o padrão de comunicação RTU.

Figura 22- Dinâmica de comunicação Modbus.

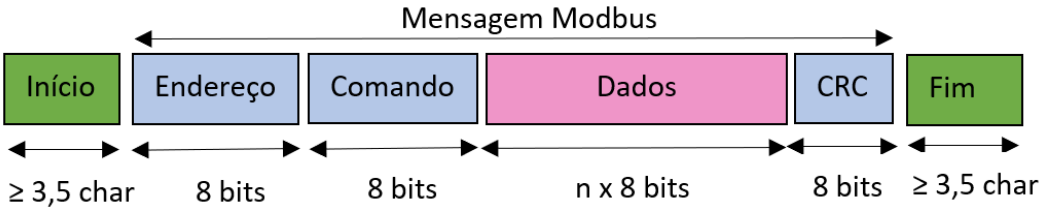


Fonte: Adaptado de BASU e SWAPAN, 2015.

O modo de transmissão define a codificação da informação transmitida. Quando os dispositivos comunicam em uma linha serial MODBUS usando o modo de transmissão RTU, cada byte em uma mensagem contém dois caracteres hexadecimais. O formato da mensagem enviada no protocolo MODBUS RTU, também chamado de frame, é composto por 8 bits de dados, 1 start bit, 1 bit de paridade e 1 stop bit, ou 2 stop bits quando não há bit de paridade, como mostrado na Figura 23.

O controle de erros no modo RTU é efetuado por CRC e em modo ASCII por LRC (COSTA, 2016).

Figura 23- *Framing* Modbus RTU.



Fonte: Adaptado de COSTA (2016).

A efetividade da comunicação Modbus RTU requer o conhecimento de algumas características do protocolo (ROCHA, 2015). As partes que compõem a mensagem Modbus são:

− Endereço Escravo: o escravo deve possuir um endereço entre 0 -247 (decimal). Desta forma, o mestre coloca neste campo da mensagem o endereço do escravo com o qual ele

deseja se comunicar. A resposta enviada pelo escravo contém seu próprio endereço para informar ao mestre qual escravo está respondendo.

- Código Função: o código de função indica um comando ao escravo, ou seja, que tipo de ação o mestre solicita que ele exerça. O Modbus apresenta vários tipos de funções para leitura e escrita de bits ou bytes.
- Dados: campo da mensagem onde o mestre informa parâmetros adicionais que são necessários de acordo com a função utilizada. Para o escravo, é o campo utilizado para responder com os dados solicitados pelo mestre.
- CRC (*Cyclical Redundancy Checking*): o campo de verificação de erro é o resultado de um cálculo de verificação de redundância chamado CRC16, que é realizado no conteúdo completo da mensagem.

A metodologia de transmissão entre mestre e escravo ocorre com o mestre enviando uma solicitação ao escravo contendo os parâmetros mostrados na mensagem representada na Figura 23. A solicitação contém o endereço do escravo para o qual a mensagem é destinada. O código de função indica ao escravo que ação deve ser executada, o campo de dados contém informações sobre a quantidade de registros que devem ser lidos e a verificação de erro serve para o escravo validar os dados recebidos e verificar se houve erro na transmissão (BASU e SWAPAN, 2015).

Os principais códigos de função Modbus e suas funcionalidades são apresentados na Tabela 3.

Tabela 3– Códigos de função do protocolo Modbus.

Código da função	Descrição
01	Leitura de status de saídas discretas.
02	Leitura de status de entradas discretas.
03	Leitura de bloco de registradores de *holding*.
04	Leitura de bloco de registradores de entrada.
05	Altera o estado de uma saída digital.
06	Define o valor de um registrador de *holding*.

Fonte: MODICON (1996).

A mensagem completa deve ser transmitida como um fluxo contínuo de caracteres. Quando um dispositivo transmite uma mensagem Modbus, ela contém um identificador para início e fim da mensagem. Para o Modbus RTU, o identificador entre mensagens é um intervalo entre mensagens de pelo menos 3,5 vezes o tempo de um caractere. Caso o tempo entre as mensagens seja menor que esse intervalo, significa que a mensagem está incompleta e deve ser descartada pelo receptor (MELO, 2019).

No contexto da aquisição de dados em usinas fotovoltaicas, como mostrado na subseção 3.1.1, a grande maioria de sistemas de aquisição aplicadas a sistemas fotovoltaicos são baseados na aquisição direta de tensão e corrente por meio do uso de sensores pouco robustos em sistemas de pequeno porte. A aplicação de redes de comunicação baseadas no protocolo Modbus é bastante escassa no contexto de sistemas de aquisição para plantas FV. Apenas o trabalho desenvolvido por Krishna (2019) se destaca no contexto do uso de comunicação Modbus. O sistema desenvolvido usa Raspberry Pi para receber os dados do inversor solar centralizado como meio de aquisição dos dados de geração através da interface RS485 do inversor de uma usina FV de 18 kW. Os dados são enviados para a plataforma IoT Node-RED de armazenamento em nuvem. Entretanto, para usinas com mais de um inversor (inversores decentralizados) e de maior potência, não foram encontrados trabalhos desenvolvidos.

3.5 Considerações Finais sobre o Capítulo

Diante da investigação realizada na literatura, nota-se uma lacuna no desenvolvimento de sistemas de monitoramento de baixo custo implementados em usinas fotovoltaicas de maior porte. Na grande maioria, o monitoramento é instalado de forma provisória em sistemas de microgeração ou em módulos em escala de laboratório cuja aquisição dos dados é realizada através de sensores de baixo custo, pouco robustos, que fornecem apenas alguns dos principais parâmetros de geração.

Tendo em vista essa escassez e considerando que os sistemas de coletas de dado e monitoramento comerciais possuem custo relativamente alto, o presente trabalho tem como objetivo o desenvolvimento de um sistema de aquisição de dados e monitoramento de uma estação solarimétrica e de usina de minigeração fotovoltaica, contemplando três segmentos de monitoramento pelo usuário:

- Um *website* para monitoramento e acesso à base de dados remotamente;

- Possibilidade de acesso através de *smartphone* com uma versão *mobile* do site;
- Supervisão através de sistema SCADA, representando um sistema mais robusto.

Os dados de geração da usina fotovoltaica foram requisitados diretamente dos inversores da usina. O ScadaBR foi SCADA escolhido para realizar o monitoramento dos dados da estação meteorológica e da usina FV, pois trata-se de um software livre e que possui suporte acessível na internet.

A plataforma de prototipagem Arduino e a placa ESP-8266 foram utilizadas para realizar a aquisição de dados da usina FV e da estação solarimétrica baseando-se no protocolo Modbus RTU com padrões RS485 e RS232, respectivamente, e realizar a transferência de dados para a plataforma *online*.

Para realizar o monitoramento remoto foi escolhida a plataforma IoT ThingSpeak por possuir recursos que são imprescindíveis para um monitoramento eficiente, como acesso a histórico de dados através de planilhas de extensão amplamente conhecida e através de gráficos que podem ser configurados de acordo com a necessidade do usuário. Além disso, o ThingSpeak possuir um app disponível, o ThingView, que se comunica diretamente com a plataforma.

O sistema de monitoramento integrado será instalado na usina de minigeração do Campus das Auroras da Universidade da Integração Internacional e da Lusofonia Afro-brasileira.

No capítulo a seguir, será apresentada a metodologia de desenvolvimento e os principais procedimentos adotados.

4. METODOLOGIA

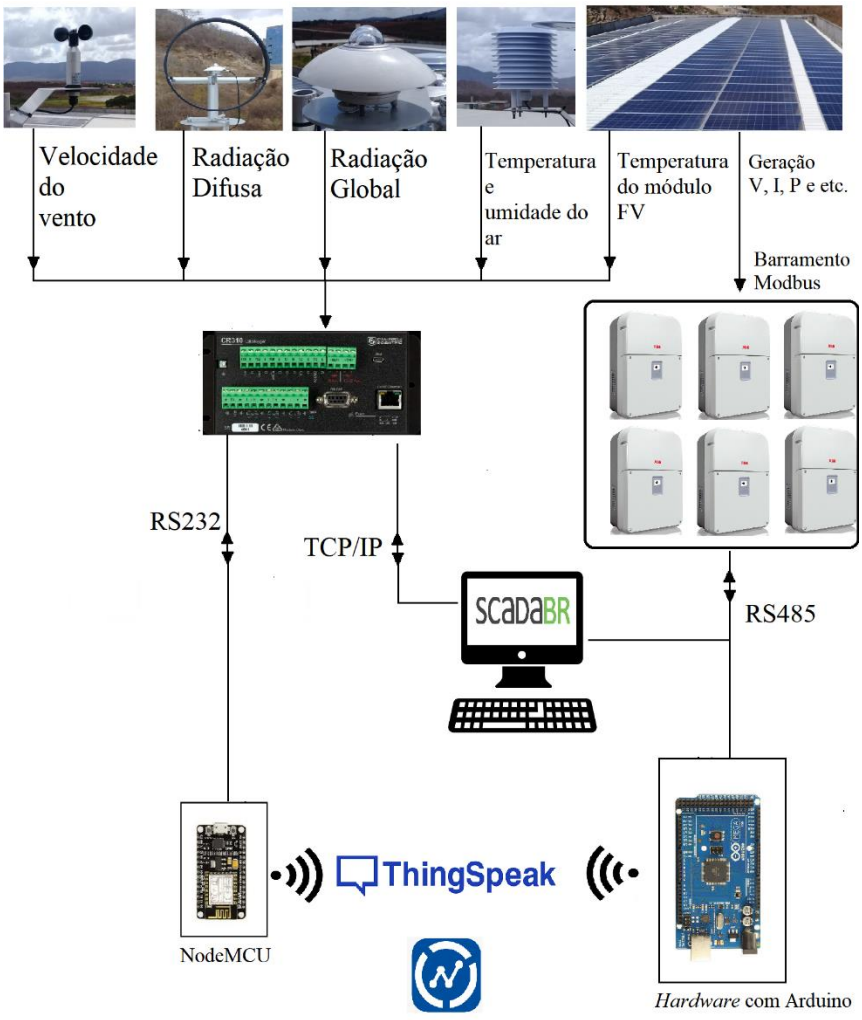
Neste capítulo é apresentado o esquema geral do sistema desenvolvido. Também é apresentada a estação solarimétrica instalada na usina fotovoltaica do Campus das Auroras na UNILAB-CE.

O objetivo do capítulo é apresentar a estação solarimétrica em estudo e a metodologia utilizada no desenvolvimento do sistema de supervisão baseado em ScadaBR e o sistema de aquisição e monitoramento baseado em plataforma IoT da estação solarimétrica e da usina fotovoltaica.

4.1 Esquema Geral do Sistema

A Figura 24 mostra o esquema do sistema desenvolvido com todos os componentes.

Figura 24– Esquema Geral do sistema desenvolvido.



Fonte: Autora.

O esquema da Figura 24 do sistema proposto mostra o datalogger, conectado aos diversos sensores que compõem a estação meteorológica, e se comunicando tanto com o ScadaBR, via ModBus IP, quanto com o NodeMCU Esp 8266, via protocolo Modbus RTU. O Esp8266 NodeMCU, se conecta a uma rede *wifi* local para enviar os dados da estação meteorológica ao ThingSpeak.

O esquema também mostra o barramento RS485, que conecta todos os inversores da usina, comunicando-se com o ScadaBR e com o Arduino, via protocolo Modbus RTU. O *hardware* baseado na plataforma Arduino Mega 2560 é responsável pelo envio de dados a plataforma ThingSpeak.

Nas seções a seguir estão apresentados de forma detalhada os componentes e dispositivos que fazem parte do sistema proposto, bem como a metodologia utilizada em seu desenvolvimento.

4.2 Estação Solarimétrica

Uma estação solarimétrica em usina fotovoltaica além de seu objetivo essencial, que é mensurar as principais grandezas meteorológicas para geração de energia fotovoltaica, tem papel relevante na formação de uma base de dados a serviço da pesquisa acadêmica, de órgãos civis e governamentais municipal e estadual. Principalmente, considerando-se que o alto custo dos equipamentos corroboram para a escassez da disponibilidade de dados meteorológicos.

A EPE (Empresa de Pesquisa Energética) especifica que para empreendimentos fotovoltaicos de grande porte que almejem concorrer em leilões de energia no Brasil, deve existir uma estação solarimétrica que possua pelo menos dois piranômetros de classificação *first class,* segundo a norma ISO 9060, e que mensure umidade relativa, temperatura e velocidade do vento (EPE, 2016).

Embora a usina fotovoltaica de Auroras, não esteja dentro do contexto citado acima, foi realizado um estudo para determinar quais sensores, segundo suas especificações, melhor atenderiam a usina. O estudo realizado serviu como base no processo de aquisição dos equipamentos.

A estação solarimétrica da usina FV, mostrada na Figura 25, fica localizada na laje do prédio principal do Campus Auroras da UNILAB e é composta pelos equipamentos a seguir:

- Dois piranômetros Classe A baseados em termopilha: um para mensurar radiação global e um para medir a radiação difusa em conjunto com um anel de sombreamento;
- Sensor de temperatura do ambiente e umidade relativa do ar;
- Sensor de temperatura do módulo FV;

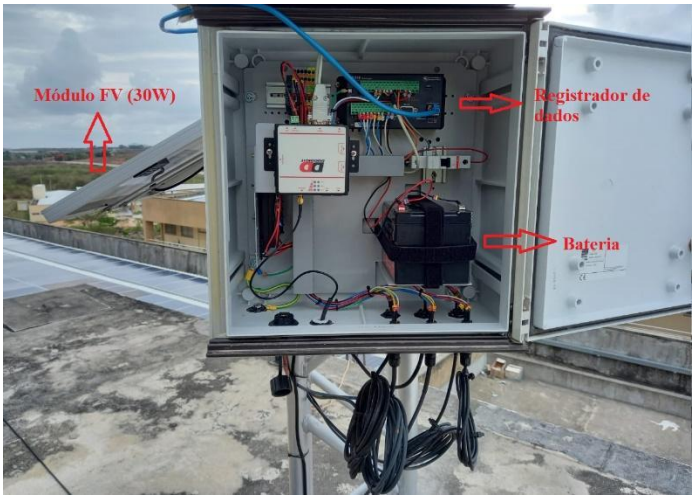
- Anemômetro: sensor que mede a velocidade do vento;
- Registrador de dados (*datalogger*);
- Sistema de geração e armazenamento de energia para alimentar a estação.

A estação é autônoma e possui um módulo fotovoltaico de 30 W que carrega uma bateria chumbo ácida regulada por válvula de 12V e 12 Ah que alimenta a estação, como mostrado na Figura 26.

Figura 25- Estação solarimétrica.

Fonte: Arquivo próprio.

Figura 26– Painel Elétrico e módulo FV da estação solarimétrica.



Fonte: Arquivo próprio.

O piranômetro SR20-D2, de fabricação Hukseflux, está classificado na categoria Classe A, a mais alta da norma ISO 9060: 2018. O piranômetro é do tipo digital, com tempo de resposta inferior a 15 s, e é projetado para a indústria de energia solar fotovoltaica para medir radiação solar recebida pela superfície de um plano, em W/m^2, e ângulo de visão é de 180º. O protocolo de comunicação do SR20-D2 é o padrão Modbus RTU RS485 sobre dois fios (*half duplex).* O SR20-D2 é capaz de medir radiação global, que é composta de radiação direta e difusa, bem como unicamente a radiação difusa, e para isso utiliza um anel de sombreamento como mostrado na Figura 27. Entretanto, o método do anel de sombreamento pode produzir subestimativas na medida da radiação difusa e, por isso, necessita de fatores de correção para compensar a irradiância difusa bloqueada pelo anel. Os fatores de correção não são fixos, podendo variar com o hemisfério, latitude e inclinação solar. O *datalogger* do sistema é programado pelo fabricante para calcular a irradiância difusa, que considera os fatores de correção. O registrador de dados também foi programado para estimar a radiação direta, que é obtida calculando-se a diferença entre a radiação global e a radiação difusa corrigida.

Figura 27- Piranômetro com anel de sombreamento.

Fonte: Arquivo próprio.

Para medir a temperatura e umidade relativa do ar é utilizado um sensor EE060 com um abrigo termométrico que mede uma faixa de temperatura de -40 a +60 ºC e umidade de 0% a 100%. Para medir a temperatura do módulo fotovoltaico, foi fixado um sensor PT100 Classe A com faixa de medição de -30 a +200 ºC. Para medir a velocidade do vento é utilizado um anemômetro do tipo caneca, mostrado na Figura 28. O anemômetro possui faixa de medição de 0,7 a 50 m/s e precisão menor que +/- 2%.

Além de obter as medições das variáveis supracitadas, é possível verificar o estado da bateria, temperatura interna do painel elétrico, e informações sobre angulação e elevação solar.

O registrador de dados da estação é o modelo CR310, mostrado na Figura 29, da fabricante Campbell Scientific. O CR310 possui 6 portas analógicas, 2 portas digitais I/O, uma porta USB micro, uma porta de comunicação RS-232 e uma porta Ethernet integrada. A comunicação é viabilizada pelos principais padrões como PakBus, Modbus, entre outros.

Figura 28- Anemômetro tipo caneca.

Fonte: Arquivo próprio.

Figura 29– *Data logger* CR310.

Fonte: Campbell Scientific.

O sistema de coleta de dados acompanha um controlador de carga para painel solar possibilitando o carregamento da bateria e a alimentação do *data logger*. O CR310 possui memória de 10 MB para armazenamento de dados.

O fabricante Campbell disponibiliza um *software* que pode ser utilizado para coletar dados *in loco*, através da porta USB micro ou RS-232, entretanto esse método de coleta pode

limitar bastante o acesso aos dados para monitoramento, uma vez que todo o sistema de coleta de dados está instalado na cobertura do último pavimento do prédio do Campus, em local de acesso remoto.

Desta forma, para ampliar as possibilidades de monitoramento e facilitar o acesso aos dados por parte do usuário, é apresentada a seguir a metodologia das soluções de monitoramento propostas pelo presente trabalho.

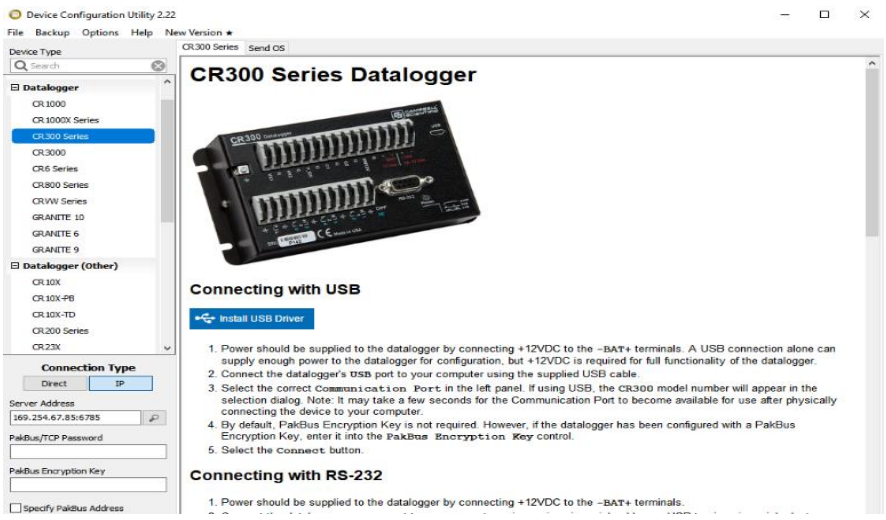
4.3 ScadaBR

4.3.1 Estação Solarimétrica

Para estabelecer a conexão de forma eficaz entre o CR310 e o PC no qual o ScadaBR está instalado, é necessário realizar alguns procedimentos que são descritos a seguir. Optou-se por integrar o *datalogger* ao PC por meio da porta Ethernet do CR310 para realizar a comunicação com o PC, pois a mesma viabiliza o acesso através do protocolo Modbus TCP/IP que permite um maior comprimento de cabeamento que o protocolo Modbus Serial e visto que a estação fica localizada em locar de difícil acesso, um cabeamento mais longo proporciona maior flexibilidade na comunicação.

Para acesso através da porta Ethernet do *datalogger*, foi realizada a configuração inicial para estabelecer a comunicação entre o *datalogger* e o PC com o *software Device Configuration Utility* disponibilizado no site da Campbell Scientific. A Figura 30 mostra o *layout* principal do *Device Configuration Utility*.

Figura 30– *Layout Device Configuration Utility.*

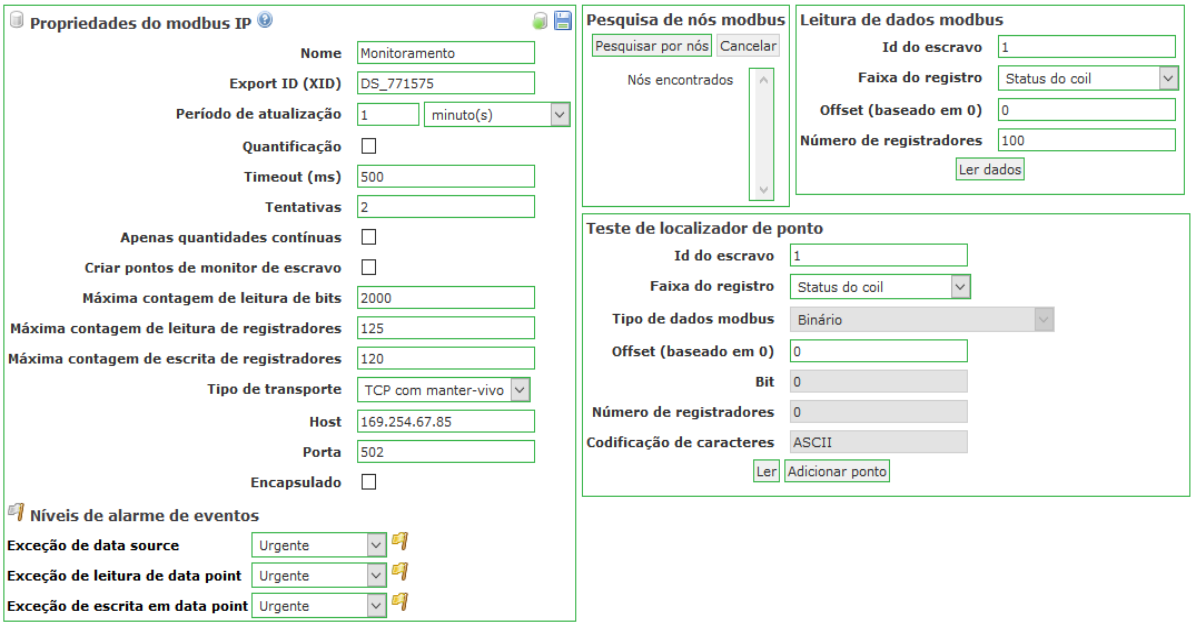


Fonte: Campbell Scientific

Após a configuração inicial e conexão *do datalogger* com um PC através de um cabo de rede com conectores RJ, foi criado um *Data Source* do tipo Modbus IP, como mostrado na Figura 31. Os campos *Host* e *Port* determinam como encontrar o equipamento Modbus na rede. No campo *Host* foi inserido o endereço de IP do CR310 e em *Port* o padrão 502.

Os campos "Timeout" e "Tentativas" determinam o comportamento do sistema quando uma solicitação de leitura falhar. O *timeout* define o tempo em milissegundos em que o *data source* aguarda uma resposta; caso não receba a resposta solicitará novamente quantas vezes estiver definido no campo "Tentativas".

Figura 31– Configuração de data source.



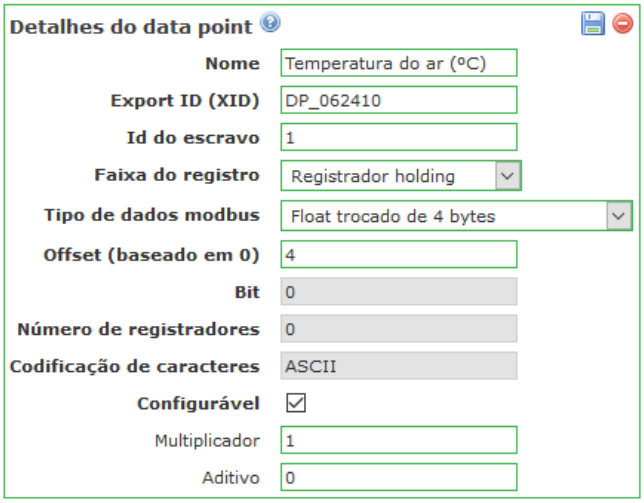
Fonte: Autora.

Os valores de cada variável coletada pelo CR310 podem ser acessados através dos endereços dos registradores e são do tipo *float* trocado de 4 bytes. Os registradores de *holding* apresentam o range hexadecimal 0x40000 a 0x4FFFF e como possuem 16 bits (2 bytes), cada valor (leitura) é dado por 2 registradores. Dessa forma, na configuração de cada *Data Point*, mostrada na Figura 32, o campo Faixa de registro foi configurado como Registrador de *holding* de Tipo de dado Modbus *float* de 4 bytes e no endereço inicial (*offset*) foram inseridos os endereços dos registradores que contêm os dados dos sensores.

Assim, por exemplo, para acessar os dados de temperatura interna do painel, o *offset* é 0, ou seja, o valor da temperatura é dado pelos registradores de endereço 0 e 1 que juntos

possuem 4 bytes. Para o próximo valor, estado da bateria, o *offset* é 2 e assim por diante. A Figura 33 mostra a lista de todos os *Data Points* criados.

Figura 32– Configuração de Data Points.



Fonte: Autora.

Figura 33 - Lista dos Data Points criados.

Data points

Nome	Tipo de dado	Status	Escravo	Faixa	Offset (baseado em 0)
Bateria (V)	Numérico		1	Registrador holding	2
Elevação Solar (°)	Numérico		1	Registrador holding	22
Radiação Difusa (W/m2)	Numérico		1	Registrador holding	14
Radiação Difusa Corrigida (W/m2)	Numérico		1	Registrador holding	16
Radiação Direta Calculada (W/m2)	Numérico		1	Registrador holding	18
Radiação Global (W/m2)	Numérico		1	Registrador holding	12
Temp interna do painel (ºC)	Numérico		1	Registrador holding	0
Temperatura do ar (ºC)	Numérico		1	Registrador holding	4
Temperatura do módulo (ºC)	Numérico		1	Registrador holding	8
Umidade do ar (%)	Numérico		1	Registrador holding	6
Velocidade do vento (m/s)	Numérico		1	Registrador holding	10
Ângulo solar Azimute (ºC)	Numérico		1	Registrador holding	20

Fonte: Autora.

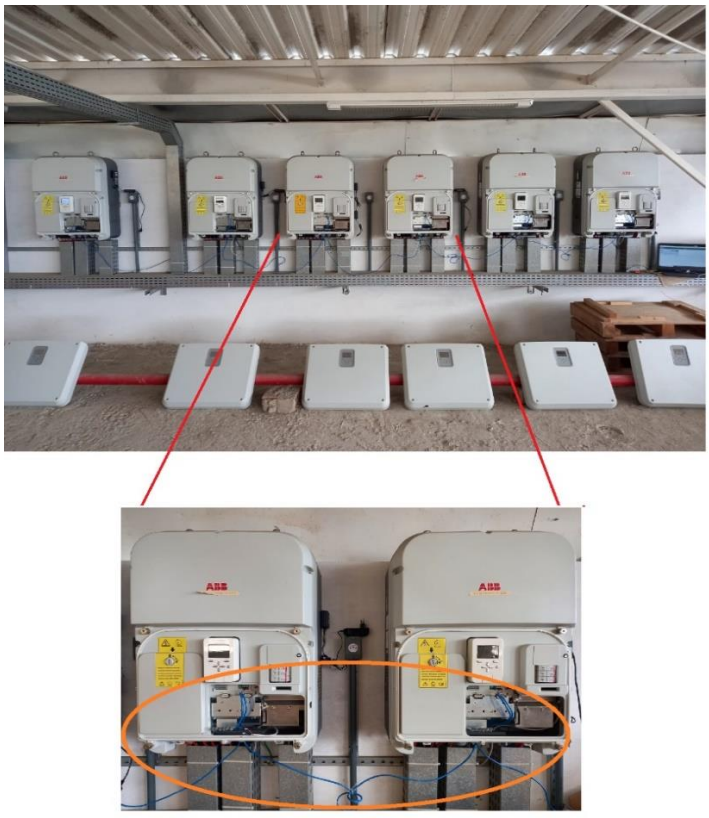
A interface gráfica criada no ScadaBR e a lista de observação com as variáveis aquisitadas é apresentada na próxima seção. As configurações de renderização de texto, unidade de medida, gráficos e históricos de dados podem ser acessados em detalhes do *data point* na *watch list*.

4.3.2 Geração fotovoltaica

A aquisição dos dados de geração foi realizada a através da comunicação entre o PC e os inversores ABB PRO-33.0 da usina. A comunicação com inversores da usina ocorre através do protocolo ModBus RTU com especificações de meio físico definidas pelo padrão RS485. O Modbus RTU é um protocolo de comunicação serial, deste modo foi criado um *data source* do tipo Modbus Serial no ScadaBR. A taxa de transferência de dados é de 19200 bps.

O manual do inversor fornece algumas orientações para conectar os inversores ao barramento RS-485. A conexão é feita a 3 fios (*Half-duplex*) onde as linhas A e B devem ser conectadas ao barramento, assim como o GND. Os inversores possuem no circuito de comunicação chaves que ativam resistores de terminação caso o comprimento da rede seja longo o suficiente ao ponto de provocar reflexões que corrompam os dados transmitidos. Como a rede desenvolvida possui comprimento inferior a 100 m, os resistores de terminação não foram ativados. Desta forma, os 6 inversores da usina foram conectados ao barramento RS-485, conforme mostra a Figura 34.

Figura 34– Barramento RS-485 com 6 inversores conectados.



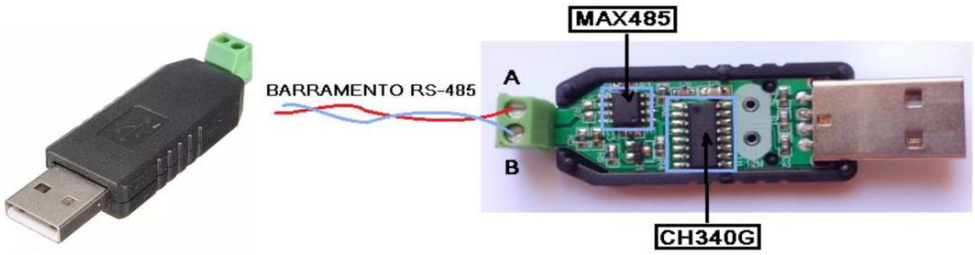
Fonte: Arquivo próprio.

Para viabilizar a conexão entre o barramento com os inversores e o computador, foi utilizado um conversor de interface serial USB para RS485, como o mostrado na Figura 35.

O conversor USB para RS-485 é composto pelos CIs MAX485 e CH340G, em que o CH340G é responsável pela conversão USB/UART que transcreve os dados vindos da porta USB e disponibiliza no barramento UART onde será acessado pelo MAX485 (GADELHA, 2019).

O conversor possui uma entrada RX e uma saída TX, responsáveis pela recepção e transmissão dos dados e que são conectados ao RX e TX da interface RS485 do inversor.

Figura 35- Conversor serial USB para RS-485.



Fonte: Gadelha (2019).

Após a conexão física, foi realizada a configuração no ScadaBR para receber os dados do inversor.

A taxa de transmissão de dados é de 19200 bps. O manual do inversor PRO-33 possui uma tabela na qual os parâmetros são organizados em grupos de parâmetros e identificados por um índice. Por exemplo, a potência (kW) pertence ao grupo de parâmetro 101 e possui índice 12. Os parâmetros são armazenados em registradores de *holding* cujo endereço é dado através da expressão a seguir:

$$E = 512(PG - 100) + 2(PI) - 1 \qquad (1)$$

em que E é o endereço inicial, PG é o grupo de parâmetro e PI é o índice do parâmetro.

As variáveis são do tipo inteiro trocado de 4 bytes, portanto devem ser lidos dois registradores seguidos com a ordem dos *bytes* invertida. Conforme orientação do manual, o valor obtido deve ser dividido por 100 para obter a leitura com duas casas decimais.

Os avisos ativos (*active warning*) são parâmetros que indicam, por meio de códigos, uma condição de anormalidade no sistema. Esses parâmetros ocupam um registrador e são do

tipo inteiro de 2 *bytes*.

A Tabela 4 mostra as variáveis e os respectivos endereços dos registradores de *holding* que foram aquisitadas pelo sistema.

Tabela 4- Registradores e dados.

Endereço	**VARIÁVEL**
513	Tensão CC (V)
515	Corrente de linha (A)
517	Frequência (Hz)
535	Potência (kW)
543	Fator de potência
571	Corrente CC (A)
2059	Aviso Ativo 1
42547	Energia Total (kWh)
46081	VAN (V)
46083	VBN (V)
46085	VCN (V)
46091	VAB (V)
46093	VBC (V)
46095	VCA (V)

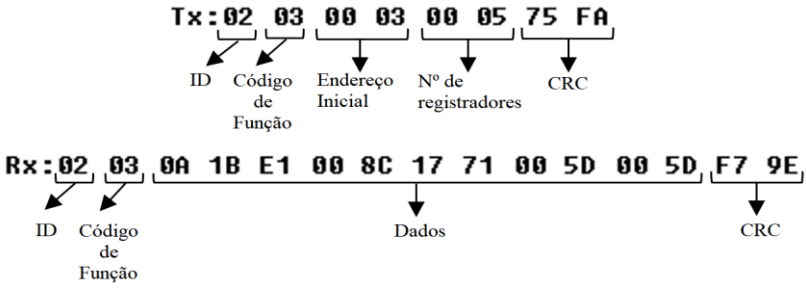
Fonte: Manual do inversor PRO-33.0-TL.

Antes de realizar a conexão entre o barramento RS485 e o ScadaBR foi realizado um breve teste de conexão com o PC através do *software* de simulação Modbus Poll. O *software* permite simular um mestre na comunicação Modbus RTU e visualizar o tráfego de comunicação possibilitando verificar se os dispositivos escravos (os inversores) estão respondendo corretamente com os dados requisitados. A Figura 36 mostra um exemplo do tráfego de dados mostrado no Modbus Poll entre o inversor 2 da usina e o PC. No exemplo é solicitada a leitura de 5 registradores de *holding* a partir do endereço inicial 3.

Além de conferir a efetiva comunicação através do Modbus Poll, os valores dos registros foram sendo conferidos durante os testes em campo através da unidade de monitoramento local

de cada inversor, no qual é possível verificar os valores em tempo real dos parâmetros tensão Vcc, potência, energia, fator de potência, frequência, dentre outros.

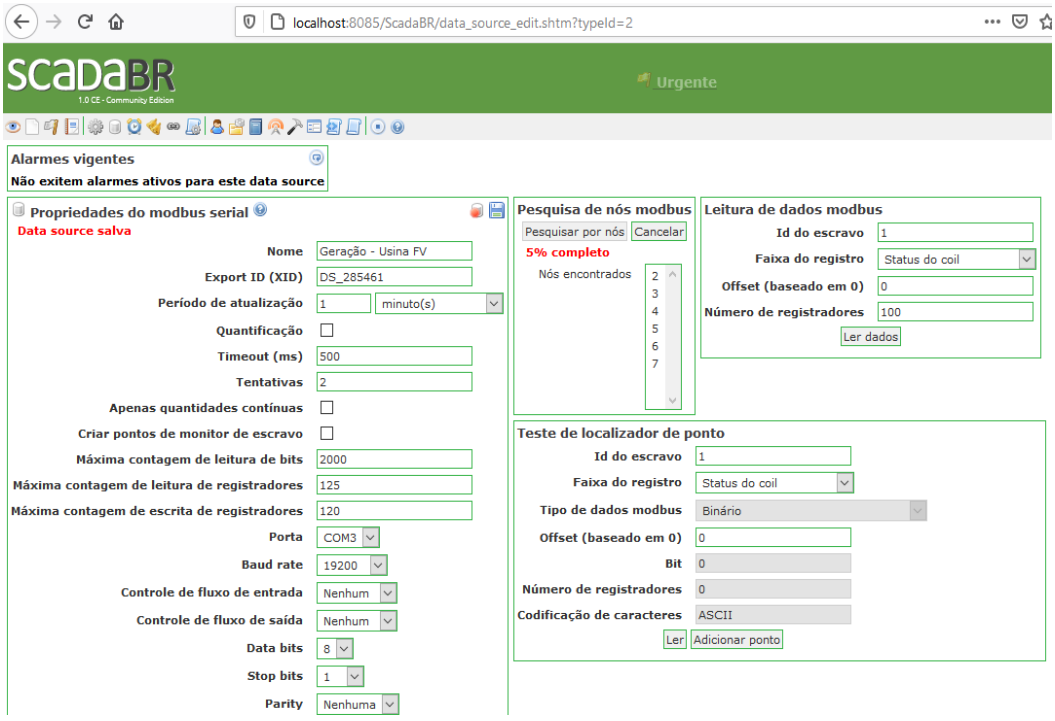
Figura 36– Exemplo do tráfego de comunicação com o inversor 2.



Fonte: Autora.

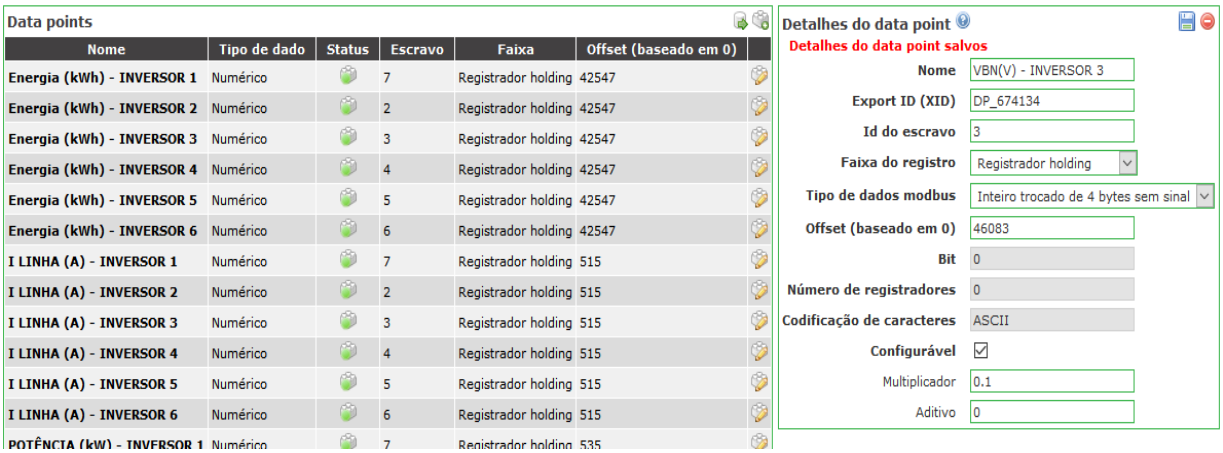
Foi criado o *data source* do tipo Modbus Serial, como mostrado na Figura 37, onde foram configurados os parâmetros de comunicação, conforme informado do manual do inversor. A Figura 38 mostra parte da lista dos *data points* criados para as variáveis do inversor. Ao todo foram criados 14 pontos de dados para cada um dos 6 inversores resultando, portanto, em um total de 84 pontos de dados.

Figura 37- Configuração do *data source* para monitoramento da geração.



Fonte: Autora.

Figura 38- Configuração dos pontos de dados dos inversores.



Data points

Nome	Tipo de dado	Status	Escravo	Faixa	Offset (baseado em 0)
Energia (kWh) - INVERSOR 1	Numérico		7	Registrador holding	42547
Energia (kWh) - INVERSOR 2	Numérico		2	Registrador holding	42547
Energia (kWh) - INVERSOR 3	Numérico		3	Registrador holding	42547
Energia (kWh) - INVERSOR 4	Numérico		4	Registrador holding	42547
Energia (kWh) - INVERSOR 5	Numérico		5	Registrador holding	42547
Energia (kWh) - INVERSOR 6	Numérico		6	Registrador holding	42547
I LINHA (A) - INVERSOR 1	Numérico		7	Registrador holding	515
I LINHA (A) - INVERSOR 2	Numérico		2	Registrador holding	515
I LINHA (A) - INVERSOR 3	Numérico		3	Registrador holding	515
I LINHA (A) - INVERSOR 4	Numérico		4	Registrador holding	515
I LINHA (A) - INVERSOR 5	Numérico		5	Registrador holding	515
I LINHA (A) - INVERSOR 6	Numérico		6	Registrador holding	515
POTÊNCIA (kW) - INVERSOR 1	Numérico		7	Registrador holding	535

Detalhes do data point

Detalhes do data point salvos

Nome	VBN(V) - INVERSOR 3
Export ID (XID)	DP_674134
Id do escravo	3
Faixa do registro	Registrador holding
Tipo de dados modbus	Inteiro trocado de 4 bytes sem sinal
Offset (baseado em 0)	46083
Bit	0
Número de registradores	0
Codificação de caracteres	ASCII
Configurável	☑
Multiplicador	0.1
Aditivo	0

Fonte: Autora.

4.4 Módulo de Comunicação

Além do sistema SCADA, foi desenvolvido um sistema de comunicação para fazer a aquisição dos dados do *datalogger* CR-310 e transferência desses dados via Wi-Fi para a plataforma IoT ThingSpeak. Os dados poderão ser acessados pelos usuários através de uma interface que também se adapta a dispositivos móveis.

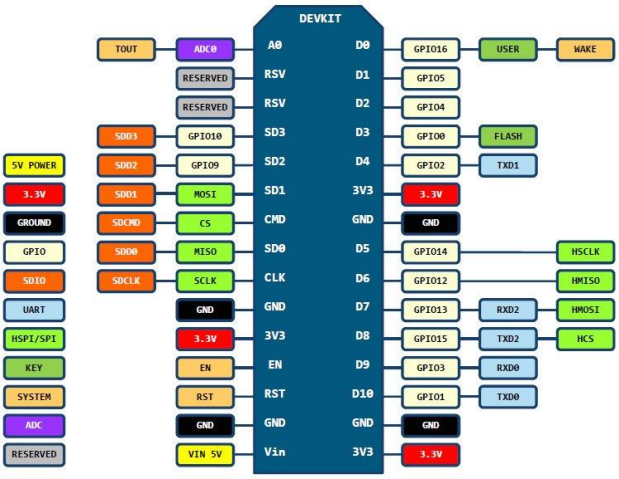
4.4.1 Estação meteorológica

Foi utilizada uma placa de desenvolvimento NodeMCU Esp8266, cujo esquema de pinos é mostrado na Figura 39. As principais especificações técnicas são (FILIPEFLOP, 2020):

- Wireless padrão 802.11 b/g/n;
- Antena embutida;
- Conector micro-USB;
- Modos de operação: STA/AP/STA+AP;
- Suporta 5 conexões TCP/IP;
- 11 portas GPIO;
- Tensão de operação: 4,5 ~ 9V;
- Taxa de transferência entre 110-460800 bps;
- Suporta Upgrade remoto de firmware;

- Conversor analógico digital (ADC).

Figura 39- Esquemático de pinos do NodeMcu ESP 8266.



Fonte: FilipeFlop (2016).

O *datalogger* CR310 permite a comunicação através do protocolo Modbus RTU por meio físico de padrão RS232. Portanto, é necessária a utilização de um conversor que viabilize a conexão entre NodeMCU Esp-8266 e o *datalogger*. Desta forma, foi utilizado um módulo MAX3232. A Figura 40 mostra o módulo MAX3232.

Figura 40 - Módulo Max3232.

Fonte: InforBatista.

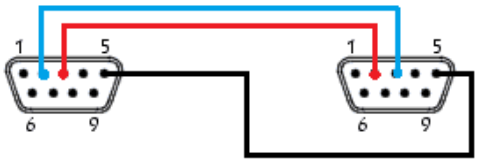
O módulo conversor utiliza um circuito integrado MAX3232, que converte um sinal serial padrão RS232 para um sinal de nível TTL sem a necessidade de instalação de *drivers*. Possui um conector serial DB9 e conector de 4 pinos para interface com microcontrolador: GND/RXD/TXD/VCC. Utiliza tensão de trabalho de 3V a 5,5V e taxa de transmissão máxima de 250 kbit/s.

Tanto o módulo MAX3232 utilizado como o *datalogger* em estudo possuem conectores

DB9 fêmea, portanto, foi necessária a confecção de um cabo DB9 macho/macho. Para isso, foram utilizados um cabo serial de 45 cm, dois conectores DB9 (macho) e dois kits de capa para proteção.

De posse dos materiais, foram soldados os pinos 5 (GND) de ambos os conectores, o pino 2 do conector 1 ao pino 3 do conector 2 e o pino 3 do conector 1 ao pino 2 do conector 2, referentes aos pinos RX e TX de transmissão de dados, como mostrado na Figura 41.

Figura 41- Conexão dos pinos do cabo DB9 macho/macho.

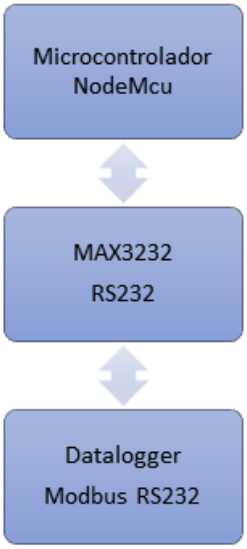


Fonte: Autora.

Com o cabo montado, foi realizada a montagem do *hardware* do sistema de comunicação, onde foram conectados os pinos TX e RX do módulo MAX3232 aos pinos TX e RX do NodeMcu. Os pinos GND e VCC do módulo conversor também foram conectados aos respectivos pinos GND e 3V do NodeMcu.

Após a conexão física entre o data logger e o NodeMcu através do cabo DB9 montado, foram realizados os testes de comunicação Modbus. O esquema do *hardware* é mostrado na Figura 42.

Figura 42- Esquema do *Hardware* do sistema de comunicação.



Fonte: Autora.

No protocolo Modbus, a comunicação é baseada na interação mestre-escravo. O dispositivo mestre, o NodeMcu, realiza solicitações de envio de dados por parte do dispositivo escravo, o *data logger* CR310.

O NodeMcu pode ser programado diretamente no IDE do Arduino, portanto, optou-se pelo uso dessa plataforma de desenvolvimento, fazendo apenas algumas alterações de configuração.

O *firmware* deve conter o ID ou endereço do dispositivo escravo conectado à rede, para o qual as solicitações de leitura dos valores dos registradores são enviadas. No caso, o ID do *data logger* é 1. A taxa de transmissão de dados ocorre a 9600 bps. A primeira parte do *firmware* desenvolvida foi baseada na requisição por parte do mestre. O NodeMcu solicita ao *data logger* o envio das leituras dos sensores. Estas, por sua vez, estão armazenadas em registradores de *holding* em endereços específicos. Os registradores de *holding* são registradores de 16 bits ou 2 bytes. Os dados dos sensores são do tipo *Float* trocado de 4 bytes (32 bits). Isso significa que cada variável ocupa 2 registradores de *holding*.

O código de função 03 é o código usado para ler o conteúdo de um ou vários registradores em um dispositivo escravo. Para realizar esta tarefa deve ser implementado no *firmware* do microcontrolador a função Modbus ReadHoldingRegisters através do comando **node.readHoldingRegisters** junto do qual deve ser especificado o endereço inicial e o número de registradores que se deseja acessar.

As informações relativas ao tipo de dado e endereços dos registradores através dos quais as leituras dos sensores podem ser obtidas, foram disponibilizadas pelo fabricante (SIGMA SENSORS, 2019).

Obrigatoriamente, os registradores são lidos em sequência a partir de um endereço inicial. Para ler vários registradores que não estejam em sequência, deve ser lido um registrador de cada vez (ROCHA, 2017).

Diferente do padrão para o protocolo Modbus comumente encontrado, os registros no *data logger* CR310 não possuem um deslocamento (*offset*) de 40000 ou 30000.

Assim como os *shields*, muitas bibliotecas feitas para o Arduino são compatíveis com as placas ESP. Desta forma, foi utilizada a biblioteca ModbusMaster232 que possui funções que facilitam o direcionamento do endereço de um sensor. A Tabela 5 mostra o endereço dos sensores.

Tabela 5- Registradores e dados.

REGISTRADOR	PARÂMETRO
0	Temperatura do painel (ºC)
2	Estado da bateria (V)
4	Temperatura do ar (ºC)
6	Umidade do ar (%)
8	Temperatura do módulo (ºC)
10	Velocidade do vento (m/s)
12	Radiação Global (W/m^2)
14	Radiação Difusa (W/m^2)
16	Radiação Difusa Corrigida (W/m^2)
18	Radiação Direta Calculada (W/m^2)
20	Ângulo Azimute (º)
22	Elevação Solar (º)

Fonte: SIGMA SENSORS, 2020.

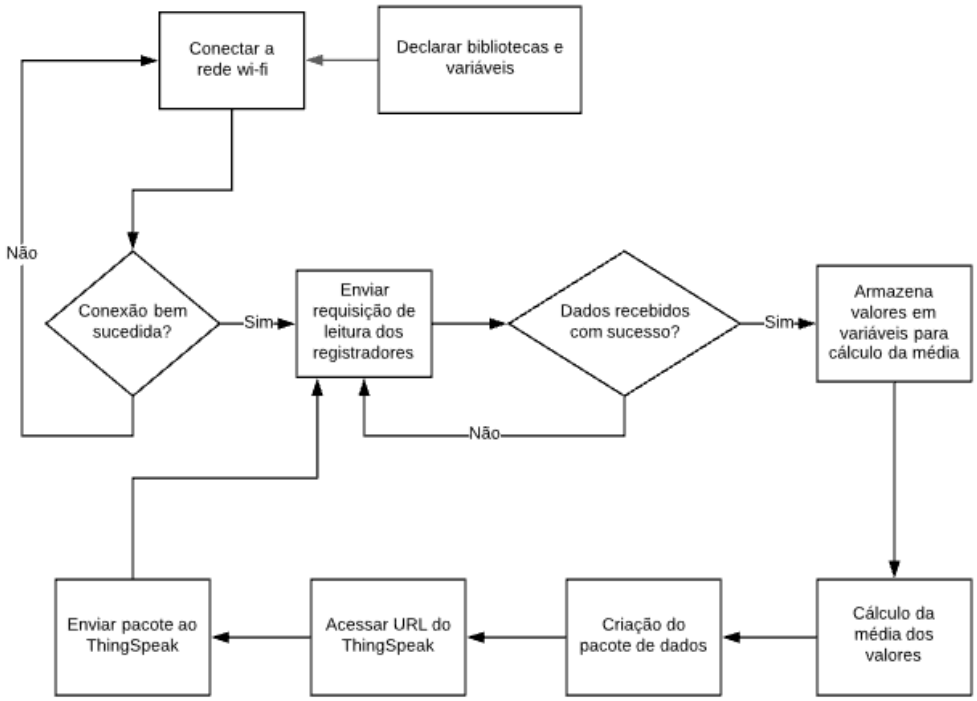
Os dados enviados chegam em formato *float* trocado de 4 bytes, portanto, foi necessário incluir no *firmware* uma etapa de processamento dos dados, invertendo a ordem de leitura dos *bytes*, de forma a tornar legível a informação obtida. Após a etapa de processamento ocorre a etapa de preparação do pacote de dados que é enviado ao ThingSpeak, especificando os *fields* das variáveis.

Posteriormente, é realizado o acesso a URL do ThingSpeak e se a coneocorrer com sucesso, o pacote de dados é então enviado à plataforma. Para estabelecer a comunicação do módulo Wi-Fi ESP 8266 com o ThingSpeak é necessário ter em mãos a chave API do canal configurado, definir o endereço do servidor e ter conexão com a internet.

O fluxograma mostrado na Figura 43 apresenta o algoritmo implementado no *firmware* desenvolvido.

O *firmware* desenvolvido está descrito no Anexo A. O código está com suas linhas de programação comentadas, para um melhor entendimento do funcionamento.

Figura 43- Fluxograma do algoritmo implementado no *firmware* do NodeMCU.



Fonte: Autora.

4.4.2 Geração fotovoltaica

Para realizar a aquisição de dados e o monitoramento das variáveis da planta de geração FV será realizado um procedimento similar ao que foi feito para a estação meteorológica. O inversor solar modelo PRO-33 TL da fabricante ABB possui um sistema de armazenamento interno dos parâmetros de geração, que podem ser acessados através da comunicação via protocolo Modbus RTU pela interface RS485 com conexão do tipo *half-duplex* (2 fios). Entretanto, para efetuar a conexão e comunicação entre o inversor e uma plataforma microcontrolada como Arduino e as placas ESP, é necessária a utilização de um conversor de sinal RS485 para sinal TTL.

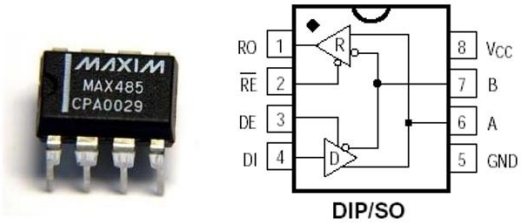
O circuito integrado MAX485, mostrado na Figura 44 a), realiza a conversão de sinais RS485 e RS422 em sinal de nível TTL. Desta forma, o MAX485 foi utilizado junto à plataforma microcontrolada Arduino para aquisição, processamento e envio dos parâmetros de geração da planta FV à plataforma IoT ThingSpeak.

O circuito do MAX485 possui 8 pinos, conforme mostrado a Figura 44 b). Os pinos RO e DI correspondem aos pinos de recepção e transmissão de dados, respectivamente, e foram conectados aos respectivos pinos RX e TX de uma das portas seriais do Arduino Mega utilizado.

Os pinos RE e DE são responsáveis por permitir o controle entre os modos de transmissão e recepção. Na conexão do tipo *half-duplex,* a transmissão e a recepção não ocorrem simultaneamente, logo, os pinos RE e DE do MAX485 foram conectados aos pinos 2 e 3 do Arduino para posterior comutação entre os modos de operação.

A alimentação do MAX485 deve ser em 5 V. Desta forma, os pinos VCC e o GND, foram conectados aos respectivos pinos 5 V e GND do Arduino. Os pinos A e B do MAX485 foram conectados aos pinos RS485-A e RS485-B da interface RS485 do inversor.

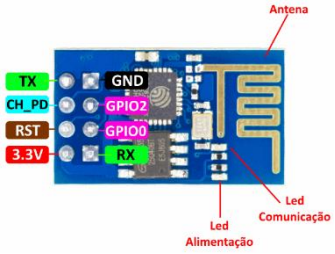
Figura 44– Circuito Integrado MAX485.



Fonte: DATASHEET MAX 485.

Para envio dos dados ao ThingSpeak, foi utilizado um dos módulos da família EPS8266, o modelo ESP8266-01, mostrado na Figura 45. O ESP 01 8266 possui 8 pinos: Vcc, GND, RX, TX, Chip Enable, Reset, GPIO-0 e GPIO-2 e tensão de alimentação de 3,3 V.

Figura 45– ESP 01 8266.



Fonte: Almeida (2016).

O *firmware* do módulo ESP8266-01 funciona nos modos AP (Access Point/Ponto de acesso) e STA (Station). O modo AP é o modo que será utilizado no presente trabalho, onde o ESP-01 trabalha em conjunto com um microcontrolador que é responsável por enviar os comandos AT para o ESP-01. No presente trabalho, o microcontrolador ATmega 2560 da placa Arduino Mega é capaz de realizar os comandos.

Para evitar a perda de dados durante os períodos de instabilidade na rede wi-fi na qual

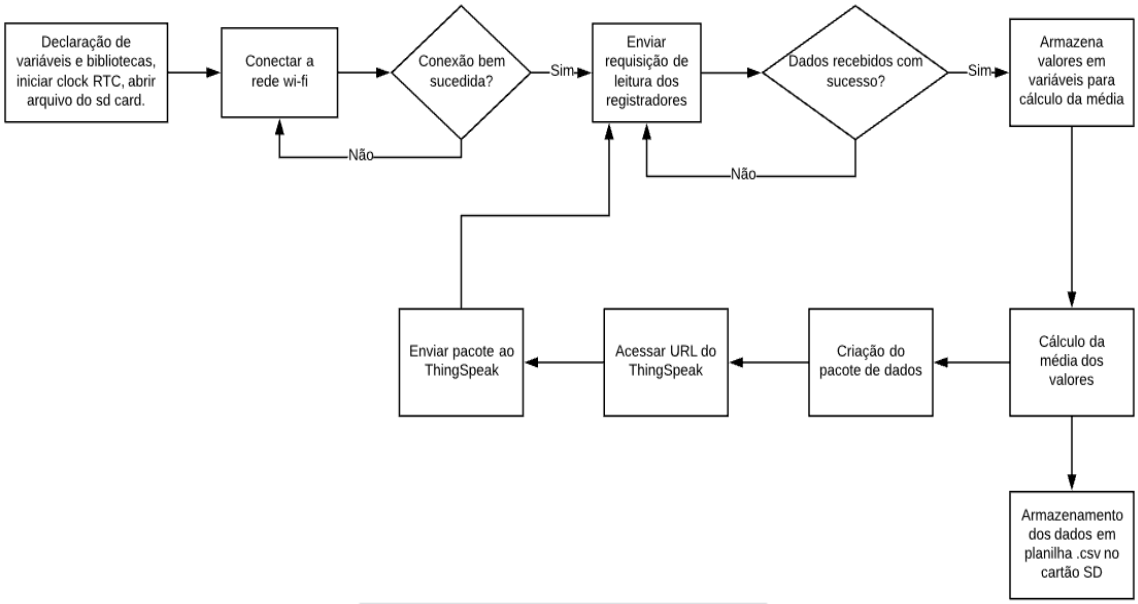
o ESP 01 8266 envia os dados para o ThingSpeak, foi inserido ao hardware do sistema o armazenamento em um cartão micro SD. O esquemático do *hardware* completo de aquisição e transferência dos dados de geração FV é mostrado no Apêndice B. Um esquema ilustrando a conexão do hardware com o barramento RS485 é mostrado na Figura 46.

Figura 46– Esquema do *hardware* de aquisição dos dados de geração dos inversores.

Fonte: Autora.

O algoritmo do *firmware* implementado no Arduino é mostrado na Figura 47. Foi utilizada a biblioteca ModbusMaster para facilitar o direcionamento das requisições.

Figura 47-*Firmware* de aquisição dos dados do Arduino Mega.



Fonte: Autora.

Como mencionado na Seção 4.3.2, a simulação com Modbus Poll mostrou que todos os inversores respondem adequadamente ao mestre. Desta forma, para verificar se o *firmware* implementado ao Arduino funciona adequadamente, foi realizado um teste de comunicação utilizando o *software* ModbusSlave. O *software* permite simular vários dispositivos escravos simultaneamente na comunicação Modbus RTU.

Desta forma, foram criados 6 escravos com os mesmos ID dos inversores. Os registradores foram preenchidos com valores aleatórios apenas para conferência. Desta forma, após a conexão, os valores foram sendo conferidos no próprio monitor serial no ambiente de desenvolvimento do arduino, constatando que o *firmware* executa as funções para as quais foi desenvolvido. A Figura 48 mostra um exemplo da requisição realizada pelo Arduino e a resposta do ModbusSlave simulando o inversor 3. Após esta constatação, o *hardware* foi conectado ao barramento RS485 com os inversores. Os valores obtidos pela leitura de registradores, foram sendo conferidos através do monitor serial, como mostra a Figura 49 e comparados com as informações da unidade de monitoramento local do próprio inversor.

Figura 48-Tráfego de comunicação no ModbusSlave.

Fonte: Autora.

Figura 49- Monitor serial do Arduino com dados do inversor 3.

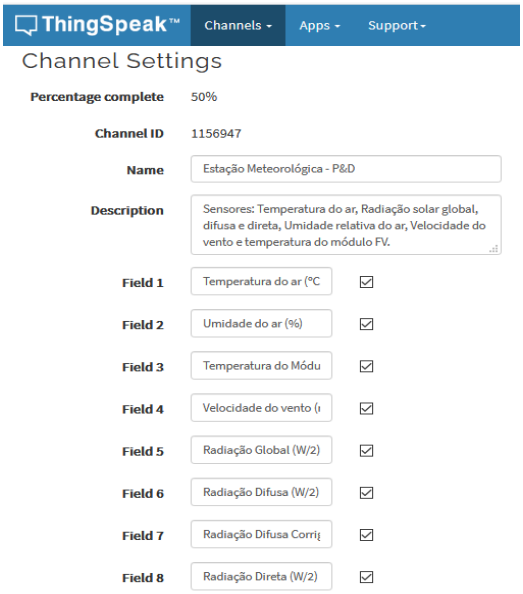
```
Frequência 3:  60.01
Potência 3:  17.10
fator de potencia 3:  1.00
Corrente de entrada 3:
24.40
Aviso 3:
0
VAN3:
224.0
VBN 3:
224.0
VCN 3:
227.0
VAB 3:
387.0
VBC 3:
393.0
VCA 3:
389.0
Ennergia 3:
146631.0
```

Fonte: Autora.

4.5 Configuração do Thingspeak e ThingView

Para utilizar o ThingSpeak é necessário se cadastrar no site. Para criar um canal deve-se clicar em “Channels” → “My Channels” → “New Channel” e configurar inserindo o nome do canal e o dos campos, como mostra a Figura 50. Na aba “API Keys” ficam disponíveis as chaves de escrita e leitura que são exclusivas de cada canal.

Figura 50 - Tela de configuração do canal no *ThingSpeak.*

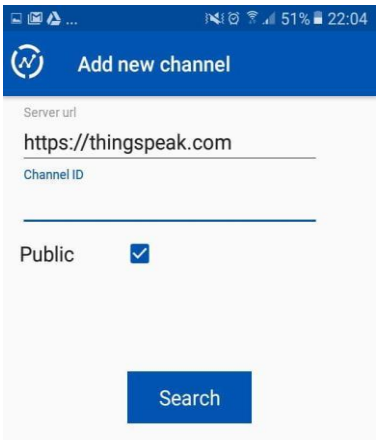


Fonte: Autora.

Os gráficos são criados automaticamente para cada campo configurado. No padrão da plataforma os gráficos são atualizados a cada 15 segundos, porém essa taxa de amostragem pode ser modificada para períodos maiores. Para mostrar os valores recebidos em tempo real foi adicionado um *display* para cada *field*. A versão gratuita da plataforma permite a criação de até 4 canais com 8 *fields* cada um. É possível determinar o período em pontos de dados ou dias entre outros recursos.

Para monitoramento remoto através de smartphone, foi utilizado o ThingView, um aplicativo para sistemas IOS e Android disponível gratuitamente que permite a visualização de todos os gráficos dinâmicos criados no ThingSpeak. Após a criação do canal no ThingSpeak, o ThingView também foi configurado para acessar o canal criado. A Figura 51 mostra a tela inicial de configuração do ThingView.

Figura 51 - Layout inicial do ThingView.



Fonte: ThingView.

4.6 Considerações Finais sobre o Capítulo

Durante o desenvolvimento do sistema Scada constatou-se que o ScadaBR é uma plataforma bastante intuitiva e de fácil compreensão. Entretanto, dependendo do equipamento, é necessário auxílio do suporte técnico do fabricante do equipamento ou sistema a ser monitorado, uma vez que algumas informações podem estar desatualizadas ou implícitas nos manuais.

A interface gráfica criada no ScadaBR e a lista de observação com as variáveis aquisitadas é apresentada na próxima seção. As configurações de renderização de texto, unidade de medida, gráficos e históricos de dados podem ser acessados em detalhes do *data point* na *watch list*.

Para aplicações como a apresentada no presente trabalho, a versão gratuita do ThingSpeak embora limitada, é suficiente. Já para aplicações que possuem mais sensores/variáveis a ser monitoradas é necessário a aquisição da licença. O ThingView representa uma boa ferramenta de fácil manuseio, sendo necessário apenas o conhecimento do ID e da chave API do canal configurado para acessá-lo pelo ThingView.

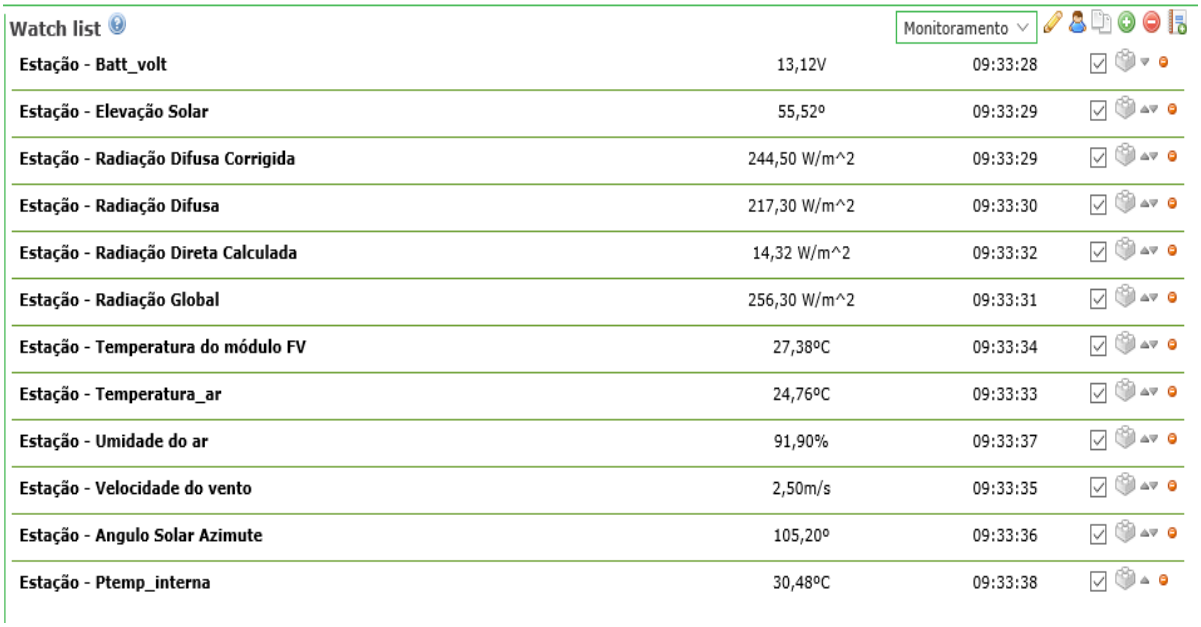
5. RESULTADOS DO SISTEMA SUPERVISÓRIO E DO SISTEMA DE AQUISIÇÃO PARA MONITORAMENTO DA ESTAÇÃO SOLARIMÉTRICA E USINA FOTOVOLTAICA -AURORAS

Nesta seção será apresentada a interface de monitoramento desenvolvida no ScadaBR e a aquisição de dados para monitoramento remoto da estação solarimétrica e da usina do Campus das Auroras.

5.1 Sistema supervisório da Estação Solarimétrica e da Usina FV- Unilab

Após a configuração dos pontos de dados (*data points*) é possível visualizar os dados na lista de observação (*watch list*), mostrada na Figura 52, conforme o tempo de atualização configurado e ver a atualização dos valores em tempo real. As configurações de renderização de texto, unidade de medida, gráficos e históricos de dados podem ser acessados em detalhes no *data point* na *watch list*.

Figura 52– Watch list no monitor principal do ScadaBR.

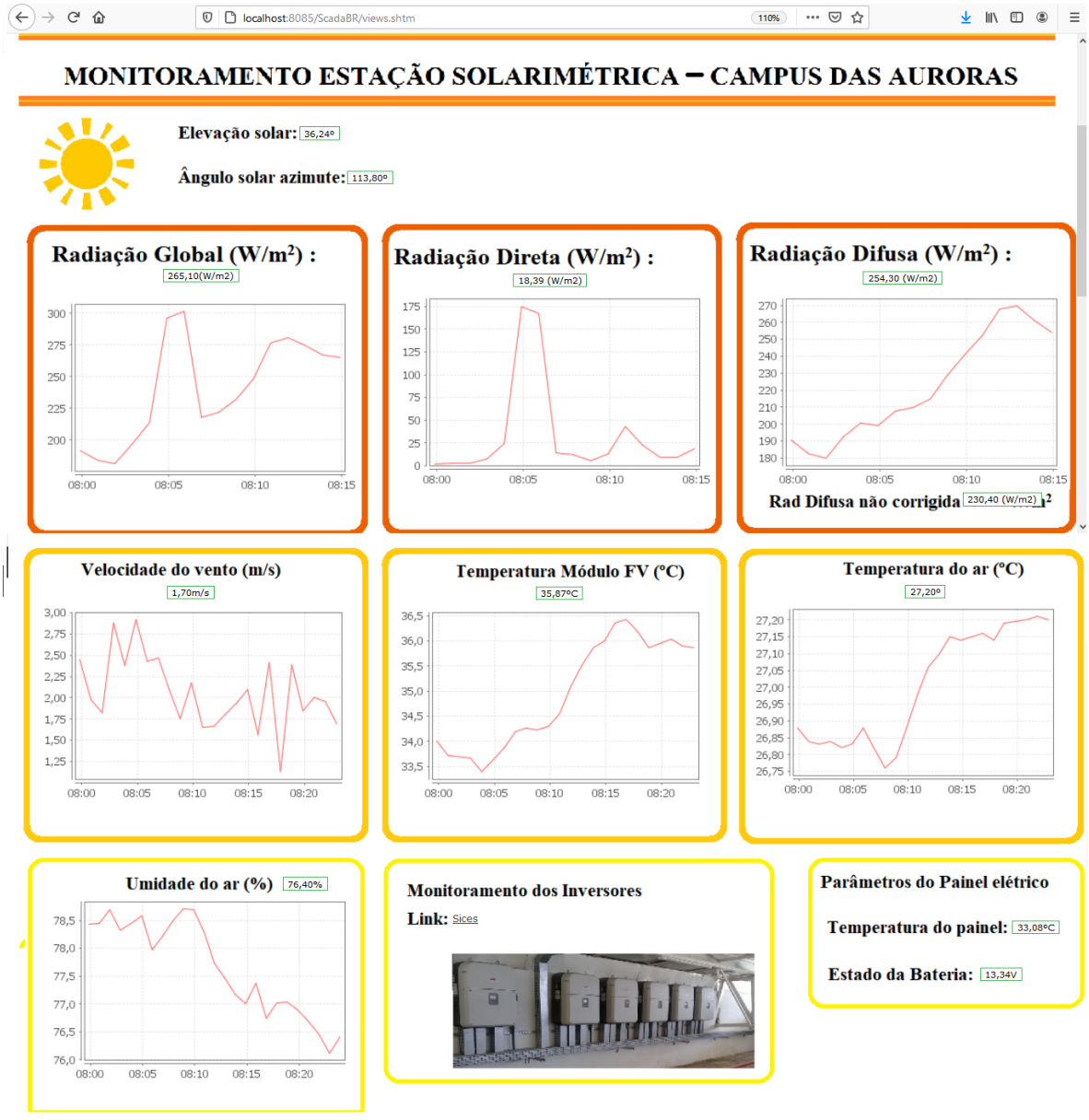


Fonte: Autora.

No ambiente de desenvolvimento de representação gráfica foi construída uma interface de acesso mais amigável ao usuário. Foram criados gráficos, que mostram em tempo real a evolução das variáveis durante o período configurado.

A Figura 53 mostra a interface desenvolvida para o monitoramento local da estação, com as variáveis meteorológicas de radiação global, radiação direta, radiação difusa, velocidade do vento, temperatura do módulo, temperatura do ambiente e umidade.

Figura 53- Layout de monitoramento da estação solarimétrica.



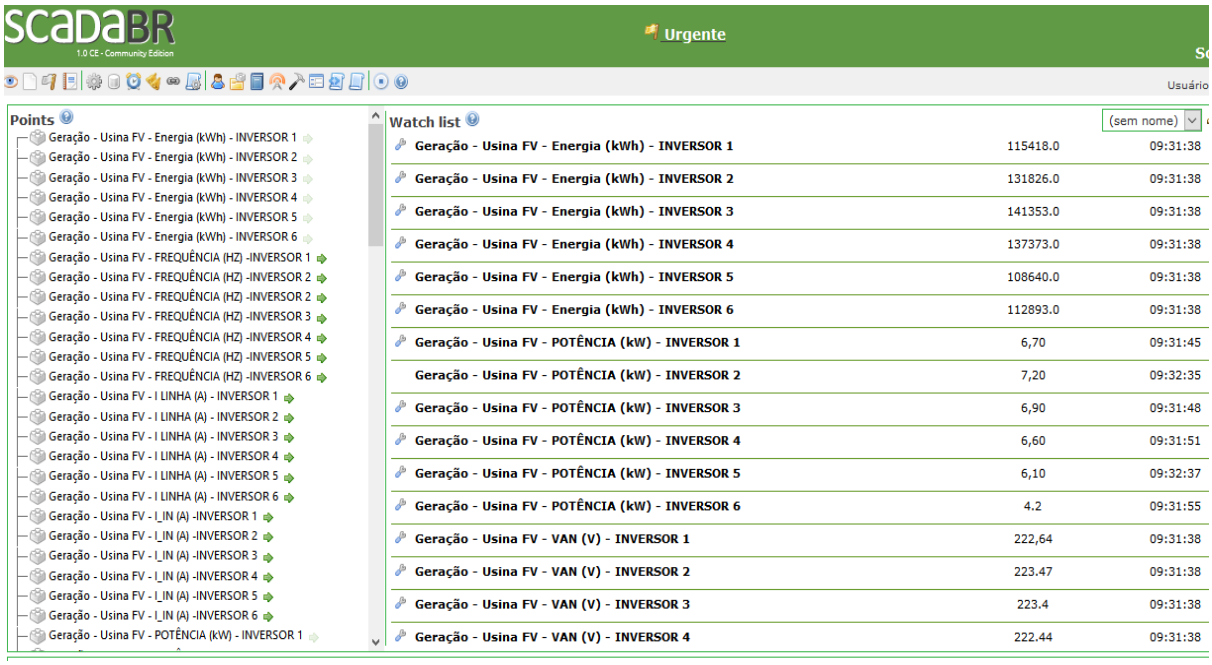
Fonte: Autora.

O intervalo de tempo dos gráficos pode ser determinado durante a configuração dos recursos desejados na representação gráfica. O dado sobre a Bateria mostra o nível de tensão nos terminais da bateria, tendo sido configurado para indicar alerta ao usuário caso o nível esteja

abaixo de 12V. O dado sobre “Radiação Difusa” também foi configurado para indicar alerta quando o valor da radiação medida estiver próximo da radiação global, indicando necessidade de ajuste do anel de sombreamento.

Para monitorar a geração de energia foram configurados pontos de dados para os principais parâmetros de geração. Além disso, foram criados pontos para os parâmetros de operação como os avisos ativos (*active warnings*), os quais indicam, através de códigos, quando uma condição anormal estiver ocorrendo no sistema, indicando ao usuário a necessidade de reparo. A Figura 54 mostra a lista de observação (*watch list*) para os dados e geração e outros parâmetros aquisitados do inversor em estudo, enquanto a Figura 55 mostra a interface desenvolvida para o usuário e que mostra os principais parâmetros de geração.

Figura 54- *Watch list* no monitor principal do ScadaBR (inversores).



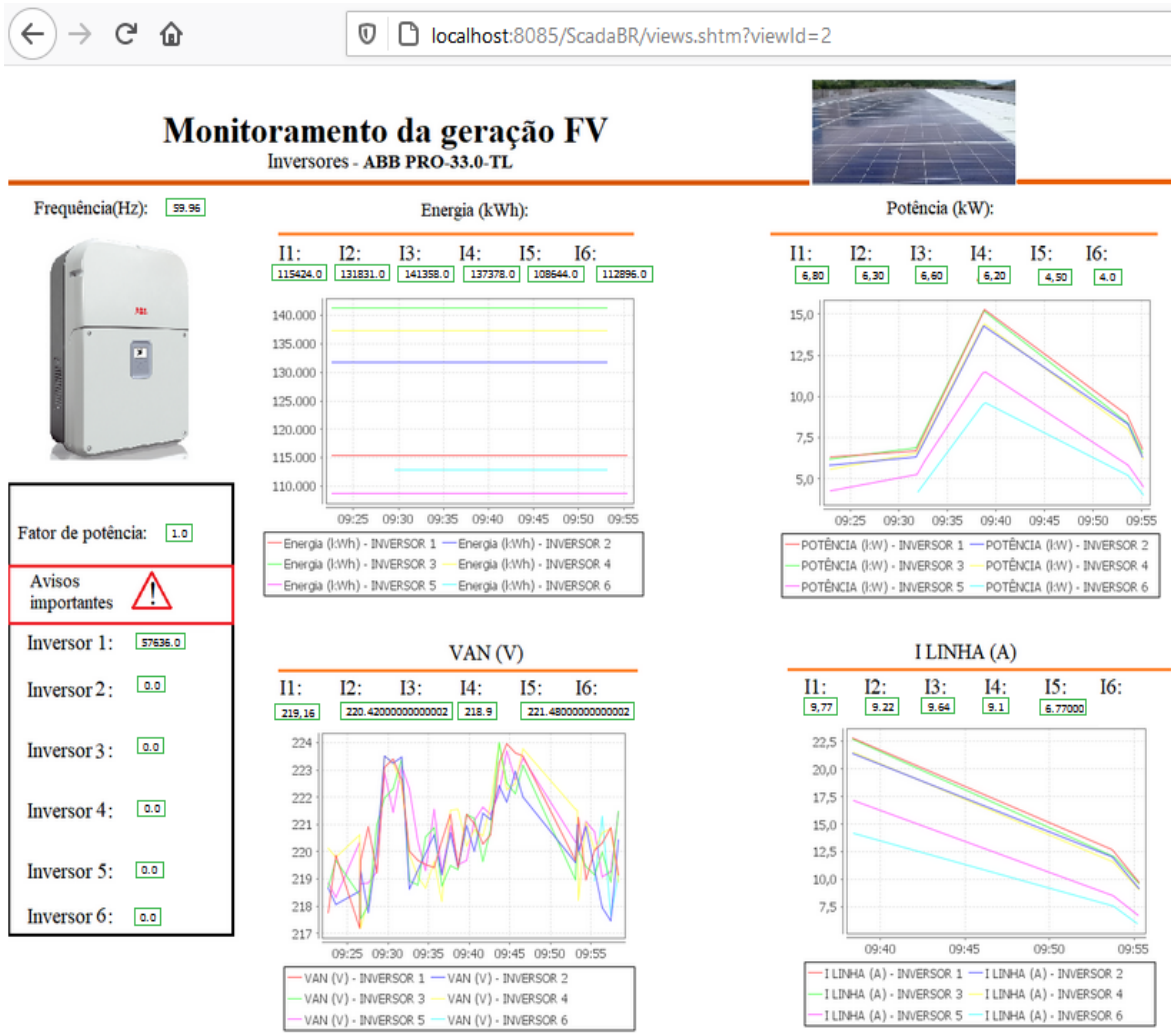
Fonte: Autora.

O sistema supervisório criado para o inversor em estudo possibilita a criação de uma rede Modbus abrangendo os 6 inversores da usina, uma vez que o padrão RS-485 permite a conexão de até 32 dispositivos escravos à rede de comunicação, diferente da comunicação RS-232 que permite a conexão de apenas um dispositivo.

Embora o tempo de conexão com o inversor tenha sido curto, durante esse período o sistema supervisório desenvolvido apresentou bom funcionamento não apresentando interrupções na comunicação.

O ScadaBR permite a visualização de uma pequena parte do histórico de dados, entretanto para acessar todas as informações de um determinado *data point* deve-se recorrer ao uso de ferramentas de gerenciamento de banco de dados Derby, o que pode limitar o acesso a informação, uma vez que é necessário que o usuário tenha certo domínio dessas ferramentas.

Figura 55- Interface de monitoramento de geração FV.



Fonte: Autora.

Para a implementação do sistema de forma permanente, são necessários procedimentos logísticos que viabilizem o monitoramento de forma contínua, uma vez que a estação fica localizada na cobertura do prédio principal do Campus das Auroras. Desta forma, para implementação do sistema é necessário cabo de rede de comprimento suficiente, *switch* e um computador disponível.

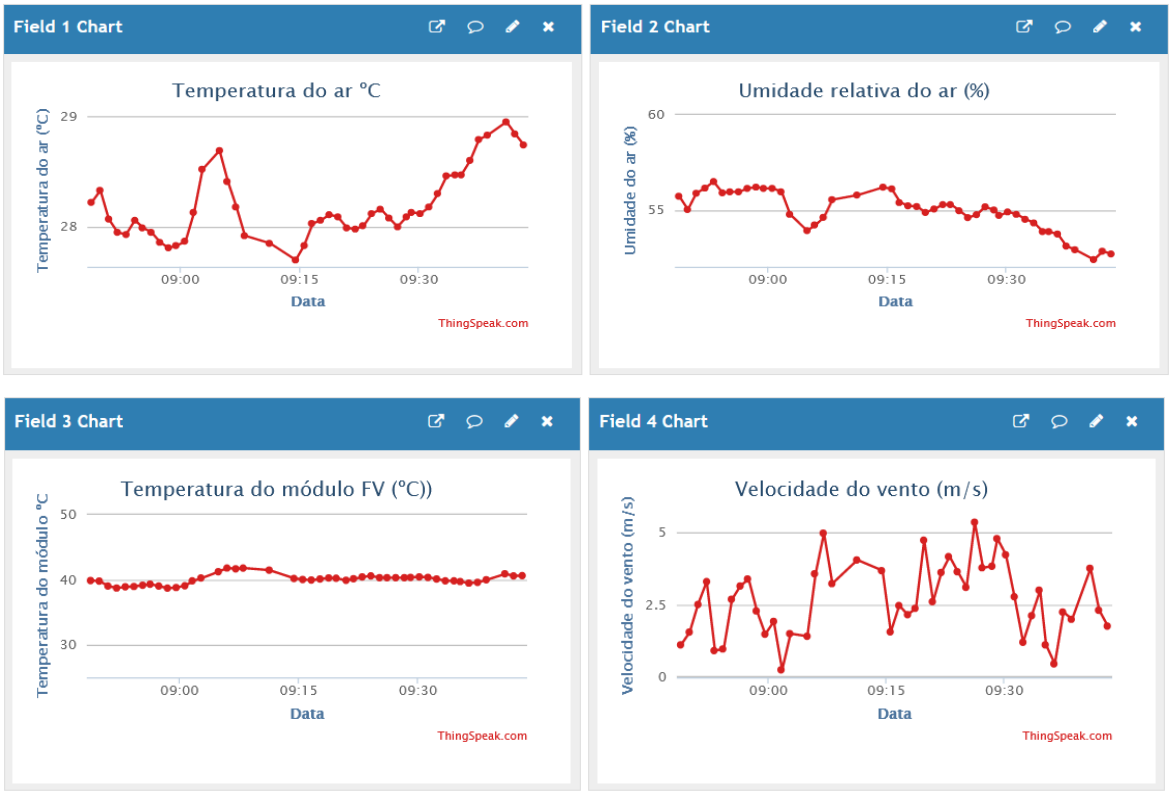
5.2 Sistema de Aquisição para Monitoramento Remoto da Estação Solarimétrica e da Usina FV-Unilab

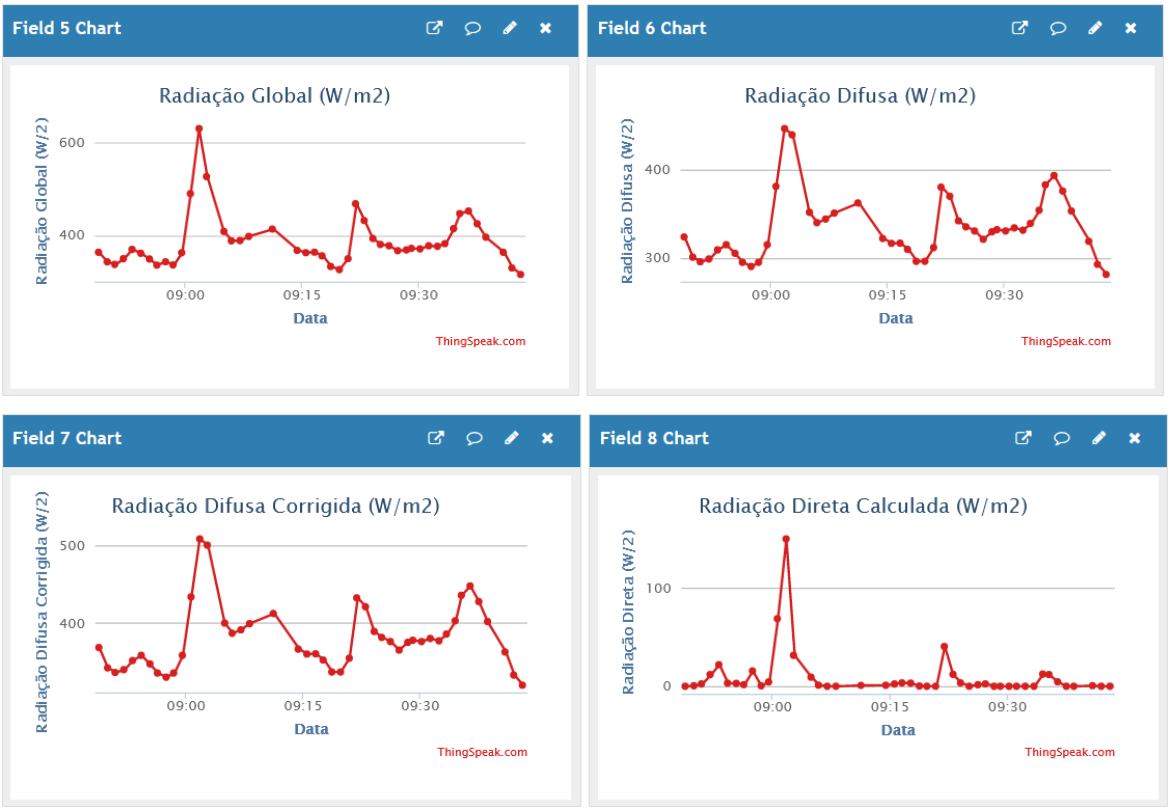
A comunicação entre o *datalogger* da estação meteorológica e o microcontrolador Wi-Fi NodeMCU ESP8266 e o envio dos dados ao ThingSpeak ocorreu eficientemente. A Figura 56 mostra os gráficos das grandezas no ThingSpeak. No dia em que ocorreram as medições mostradas nos gráficos o céu estava totalmente nublado.

O ThingSpeak mostrou-se como uma eficiente alternativa para o monitoramento das grandezas meteorológicas medidas pela estação, uma vez que os dados podem ser facilmente acessados através de login e senha, e visualizados e armazenados em planilhas *.csv* que é um formato largamente difundido no âmbito da análise de dados. Além disso, na plataforma há vários recursos de formatação de dados, taxa de amostragem e manipulação de gráficos.

A Figura 57 a) mostra a visualização no ThingView. O valor sobrescrito sobre o gráfico corresponde ao valor da última medição da grandeza. Ao clicar em um dos gráficos, obtêm-se a visualização da tela como mostrada na Figura 57 b). Nessa tela pode-se manipular o gráfico através do zoom de forma a visualizar apenas uma faixa de dados de interesse.

Figura 56- Gráficos no ThingSpeak.

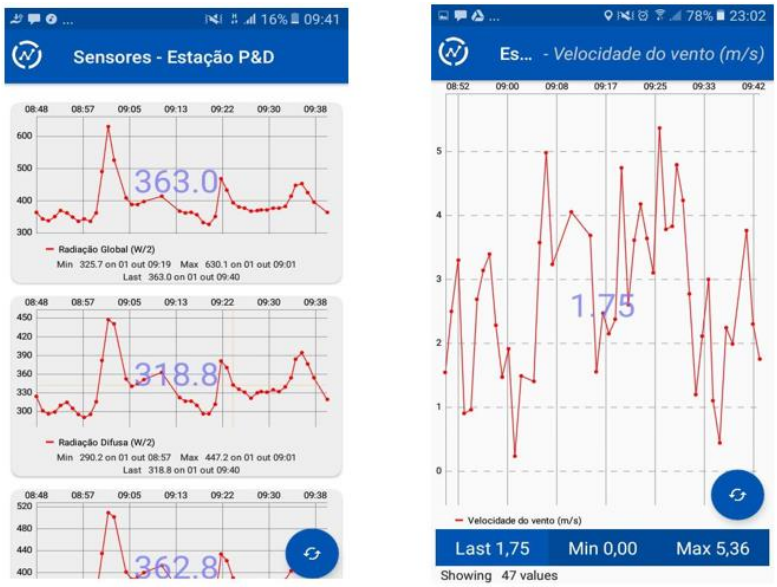




Fonte: Autora.

Como referência para os valores obtidos através da leitura de registradores, os valores enviados para o ThingSpeak foram sendo comparadas aos dos valores encontrados no armazenamento do datalogger.

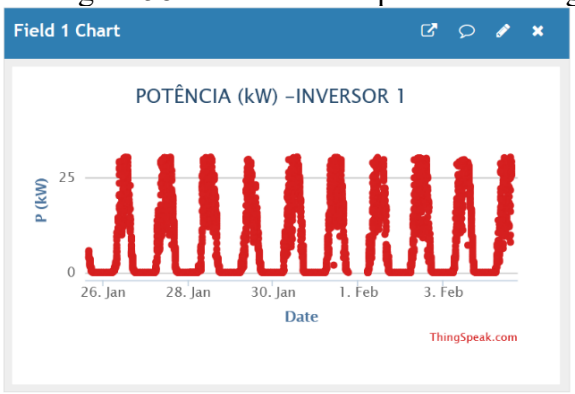
Figura 57- Visualização no *ThingView*.



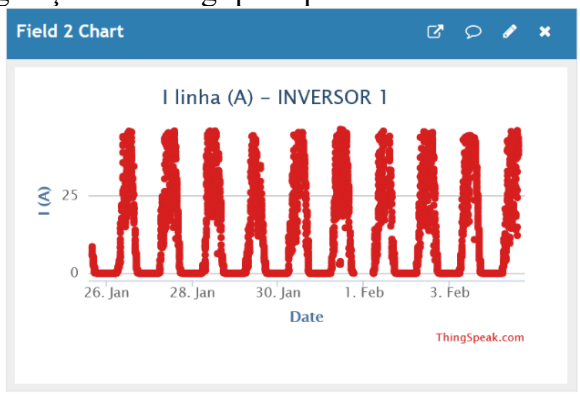
Fonte: Autora.

Alguns parâmetros dos inversores também foram enviados para a plataforma ThingSpeak, como mostram as Figuras 58 a 63. Os gráficos de potência e corrente demonstram o comportamento esperado para os sistemas fotovoltaicos. Durante o dia, a geração vai crescendo até o período em que há maior incidência de radiação sobre os paineis, e após esse pico, vai decrescendo até o dia posterior.

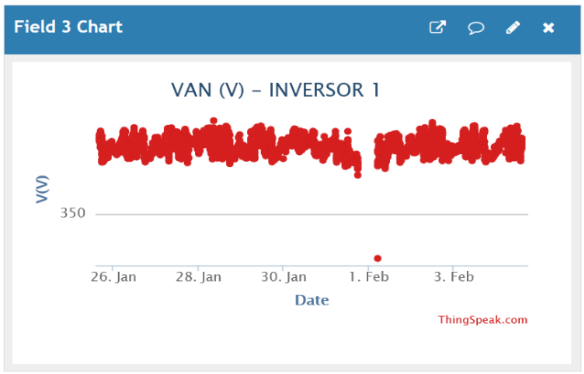
Figura 58 – Gráficos dos parâmetros de geração no ThingSpeak para o Inversor 1.



a) Gráfico de Potência – Inversor 1.

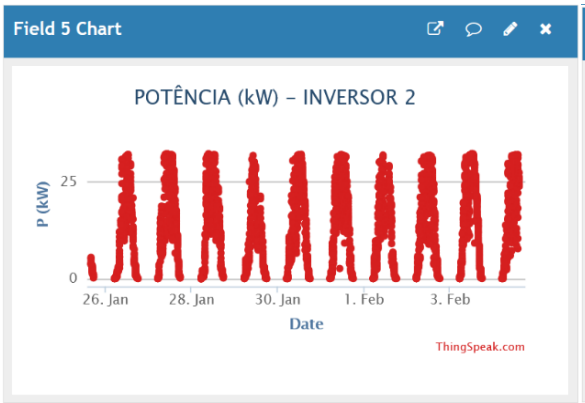


b) Gráfico da Corrente de linha – Inversor 1.

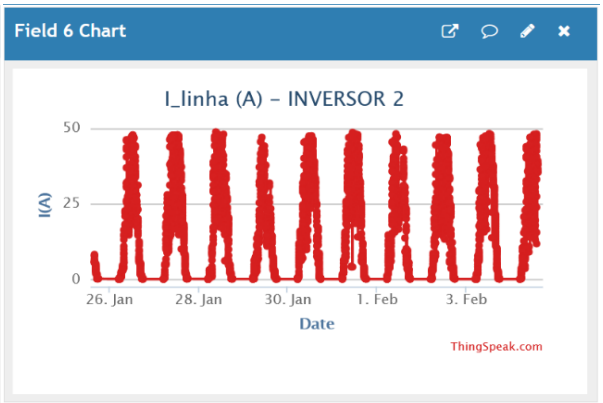


c) Gráfico da Tensão de fase A – Inversor 1.
Fonte: Autora.

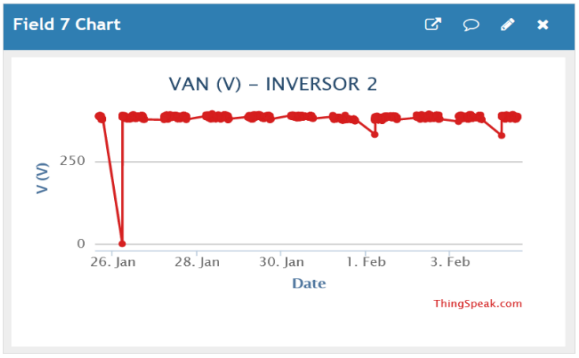
Figura 59 – Gráficos dos parâmetros de geração no ThingSpeak para o Inversor 2.



a) Gráfico de Potência – Inversor 2.

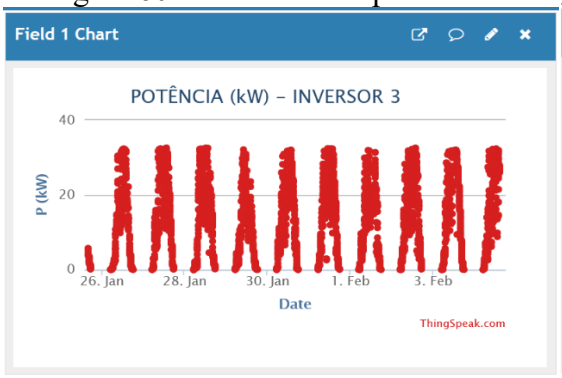


b) Gráfico da Corrente de linha – Inversor 2.

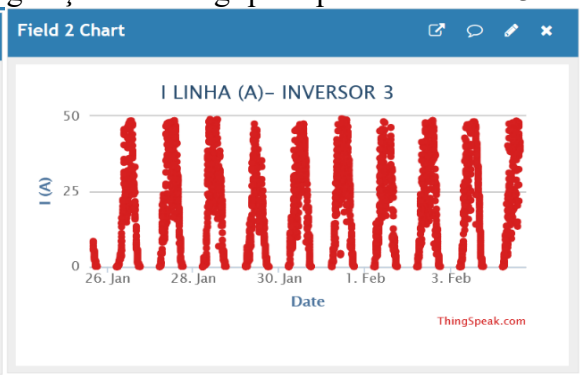


c) Gráfico da Tensão de fase A – Inversor 2.
Fonte: Autora.

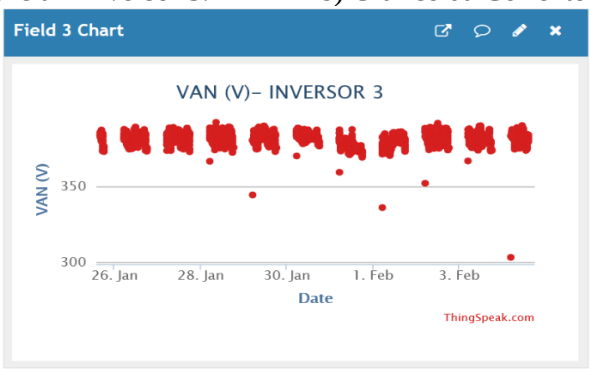
Figura 60 – Gráficos dos parâmetros de geração no ThingSpeak para o Inversor 3.



a) Gráfico de Potência – Inversor 3.

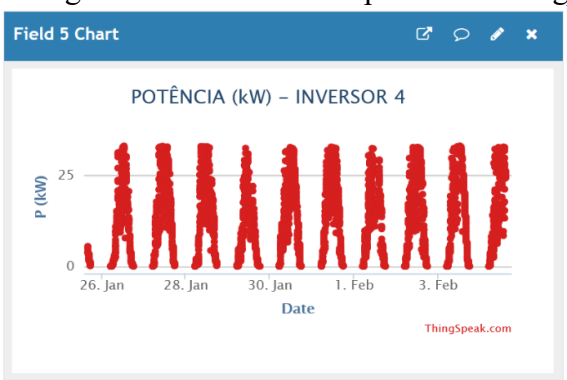


b) Gráfico da Corrente de linha – Inversor 3.

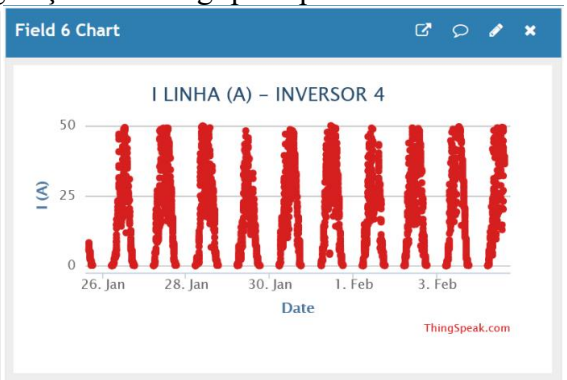


c) Gráfico da Tensão de fase A – Inversor 3.
Fonte: Autora.

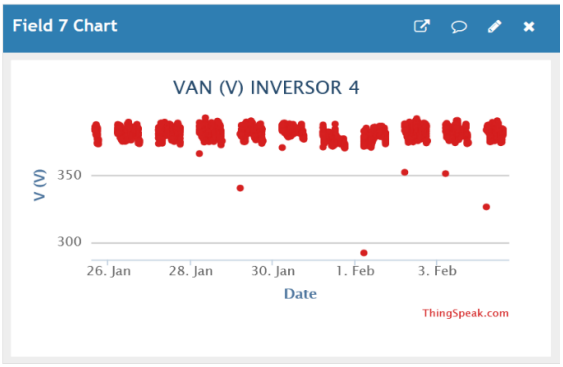
Figura 61 – Gráficos dos parâmetros de geração no ThingSpeak para o Inversor 4.



a) Gráfico de Potência – Inversor 4.

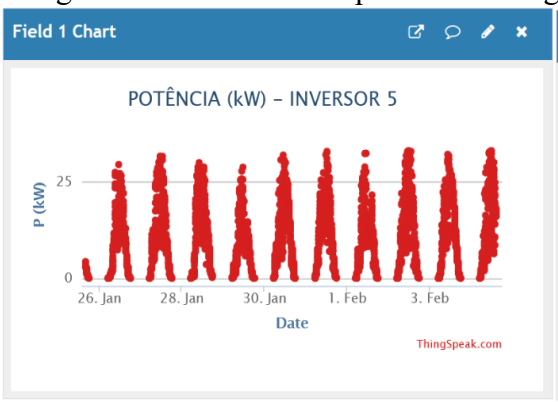


b) Gráfico da Corrente de linha – Inversor 4.

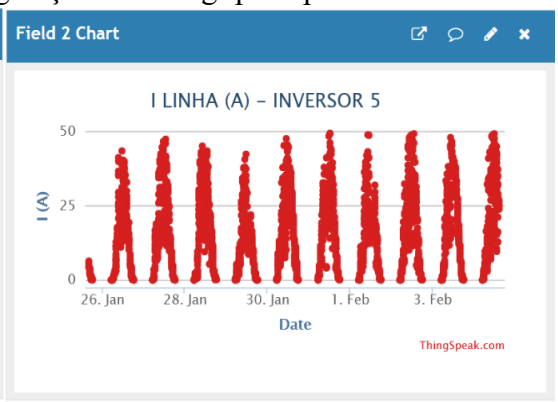


c) Gráfico da Tensão de fase A – Inversor 4.
Fonte: Autora.

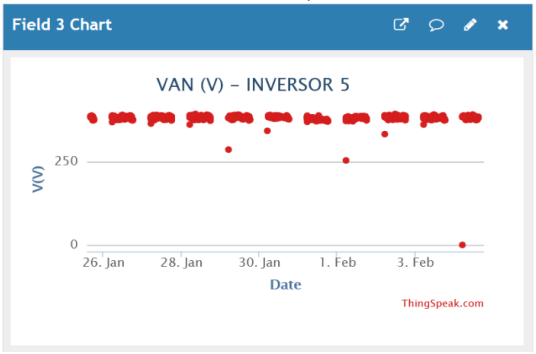
Figura 62 – Gráficos dos parâmetros de geração no ThingSpeak para o Inversor 5.



a) Gráfico de Potência – Inversor 5.

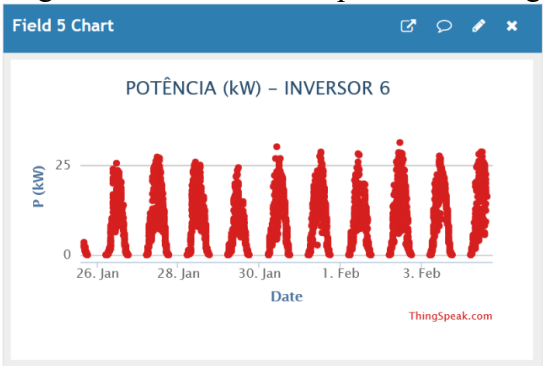


b) Gráfico da Corrente de linha – Inversor 5.

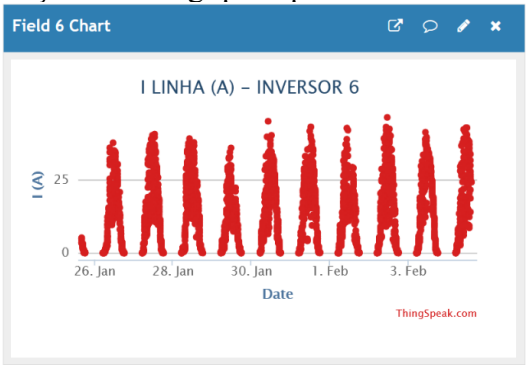


c) Gráfico da Tensão de fase A – Inversor 5.
Fonte: Autora.

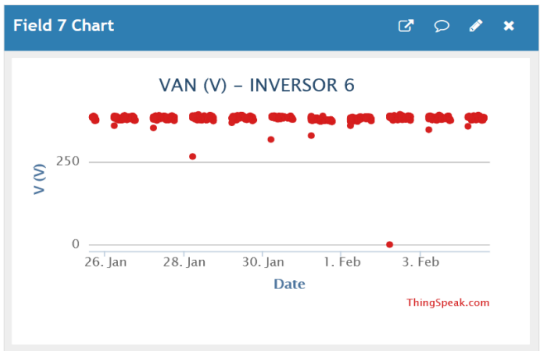
Figura 63 – Gráficos dos parâmetros de geração no ThingSpeak para o Inversor 6.



a) Gráfico de Potência – Inversor 6.



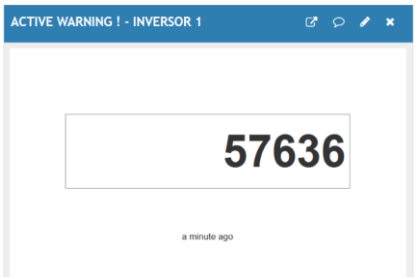
b) Gráfico da Corrente de linha – Inversor 6.



c) Gráfico da Tensão de fase A – Inversor 6.
Fonte: Autora.

Um outro parâmetro enviado ao ThingSpeak é o *active warning,* que indica quando há alguma anormalidade na operação. Na Figura 64 é possível ver que existe uma anormalidade na operação do inversor 1. No manual do PRO 33.0, o código se refere a um problema no cooler do inversor. Nesse caso, o sistema de operação do inversor é programado para limitar a potência de forma a evitar o superaquecimento dos componentes do equipamento. Este exemplo evidencia a importância do monitoramento remoto em sistemas fotovoltaicos para a operação e manutenção da planta.

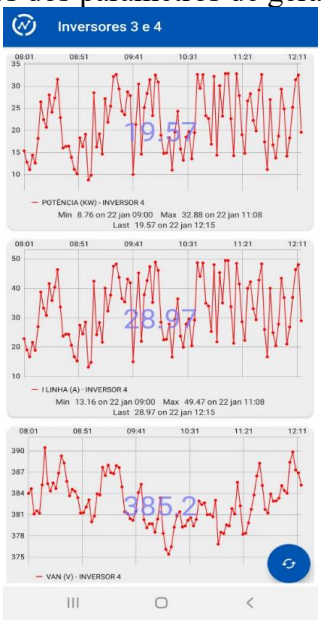
Figura 64 – *Active warning* inversor 1.



Fonte: Autora.

O ThingView também foi configurado para acessar os gráficos e valores dos parâmetros do inversor. A Figura 65 mostra no ThingView os gráficos de Potência, Tensão de linha e corrente de linha no inversor 4.

Figura 65 - Gráficos dos parâmetros de geração no ThingView.



Fonte: Autora.

Os parâmetros mostrados nos gráficos e os demais parâmetros aquisitados pelo sistema desenvolvido foram armazenados em um arquivo de planilha .csv em um cartão de memória. A Figura 66 mostra um exemplo dos parâmetros coletados para o inversor 1 salvos em planilha.

Vale ressaltar que para facilitar a compreensão do leitor e a título de organização os dados do arquivo foram apenas separados em mais colunas.

Figura 66 – Planilha com parâmetros de geração.

Data	Horario	VCC_I1(V)	Corrente_	F_I1 (Hz)	P_I1 (kW	FP_I1	Iin_I1 (A)	VAN_I1 (	VBN_I1 (\	VCN_I1 (\	VAB_I1 (\	VBC_I1(V	VCA_i1(V	E1(KWh)
19/02/2021	09:52:08	789.40	44.99	60.00	29.50	1.00	43.43	384.73	393.88	389.22	221.73	224.78	227.93	120568.68
19/02/2021	09:54:22	744.50	39.89	60.02	28.70	1.00	40.72	384.17	393.03	389.32	221.95	223.98	227.50	120569.78
19/02/2021	09:56:38	795.86	45.15	60.01	30.50	1.00	43.26	384.70	393.85	389.82	221.93	224.67	227.98	120570.82
19/02/2021	09:58:55	808.15	45.76	60.00	30.40	1.00	42.84	381.43	390.35	385.62	219.45	222.87	225.73	120571.97
19/02/2021	10:01:08	815.72	45.69	59.99	30.40	1.00	42.31	380.08	390.72	385.25	218.85	222.77	225.82	120573.24
19/02/2021	10:03:24	803.70	45.84	59.98	30.50	1.00	43.07	380.08	389.93	385.75	219.32	222.27	225.70	120574.34
19/02/2021	10:05:39	809.10	45.74	59.99	30.50	1.00	42.71	380.98	390.45	386.08	219.68	222.67	226.02	120575.46
19/02/2021	10:07:55	807.82	45.66	59.98	30.40	1.00	42.76	381.12	391.28	386.65	219.38	223.17	226.53	120576.57
19/02/2021	10:10:10	790.57	45.49	60.00	30.60	1.00	43.73	383.78	393.22	389.02	221.12	224.37	227.65	120577.68
19/02/2021	10:12:26	755.24	45.41	59.99	29.30	1.00	45.95	384.45	394.40	389.27	221.67	225.03	227.95	120578.87
19/02/2021	10:14:42	707.05	40.03	59.98	25.80	1.00	43.04	381.82	391.68	387.05	220.30	223.15	226.57	120580.00
19/02/2021	10:16:57	714.66	37.53	59.99	26.70	1.00	40.06	382.33	391.68	387.18	220.55	223.23	226.68	120580.96
19/02/2021	10:19:11	715.52	42.20	59.99	28.70	1.00	44.99	382.82	392.55	387.50	220.73	223.78	227.07	120581.92
19/02/2021	10:21:24	763.42	44.43	60.02	30.10	1.00	45.15	385.33	395.05	390.23	222.25	225.15	228.55	120582.99
19/02/2021	10:23:39	736.45	28.84	60.01	13.40	1.00	29.77	382.10	391.40	388.13	221.27	222.60	226.63	120584.00
19/02/2021	10:25:55	758.86	32.73	60.01	30.00	1.00	32.44	382.30	391.47	386.93	220.43	223.13	226.42	120584.64

Fonte: Autora.

Na Figura 64 é possível ver que os parâmetros de tensão CC (VCC), corrente CC, frequência (Hz), potência (kW), fator de potência, corrente de entrada (A), tensões de fase (V), tensões de linha (A) e energia total gerada (kWh).

5.3 Considerações finais sobre o capítulo

Embora a solução proposta para a aquisição de dados de geração a partir do armazenamento interno dos inversores seja relativamente simples, ela é eficaz e garante a confiabilidade dos dados obtidos.

O ThingSpeak se mostrou uma boa ferramenta. Entretanto, como mostrado na Figura 58, nem todos os parâmetros obtidos puderam ser enviados para a plataforma devido ao número limitado de pontos de dados que a versão gratuita possui. Além disso, a disponibilização do login e senha ao público pode resultar em problemas, uma vez que a configuração dos canais fica suscetível a alterações e isso pode causar problemas com o recebimento dos dados caso o usuário não tenha pleno conhecimento de como usar a plataforma.

O ThingView se mostrou uma alternativa versátil que possibilita monitoramento remoto através de *smartphone*. De forma geral, as duas plataformas atenderam bem ao propósito do presente trabalho.

6. CONCLUSÃO

Neste trabalho foram apresentadas as etapas de projeto, especificação e instalação de uma estação solarimétrica para a usina de geração solar fotovoltaica de 254,2 kWp situada no campus das Auroras na Universidade da Integração Internacional da Lusofonia Afro-Brasileira – UNILAB - CE.

Para supervisão dos dados da estação solarimétrica e dos dados de geração da usina foram usados dois diferentes sistemas de aquisição de dados e controle supervisório, um sistema SCADA de prateleira e um sistema desenvolvido em laboratório. No primeiro caso, fez-se uso do sistema ScadaBR, um *software* livre, gratuito e de código-fonte aberto. O *layout* desenvolvido possui recursos gráficos que facilitam a visualização por parte do usuário O ScadaBr permite a visualização em tempo real e disponibiliza parte do histórico com poucos pontos de dados. Entretanto, caso o usuário deseje acessar ao histórico completo de dados deve dispor de ferramenta de gestão de banco de dados e domínio dela.

No segundo caso, para a estação meteorológica, o sistema desenvolvido em plataforma de prototipagem IoT, que usa o *firmware* NodeMcu com processador ESP8266-12E, realizou satisfatoriamente a aquisição e transferência dos parâmetros meteorológicos para monitoramento através da interface IoT que permite visualização *mobile.* Frente aos sistemas comerciais de transferência de dados para estações meteorológicas, o sistema desenvolvido é vantajoso pois possui um custo significativamente inferior.

No sistema desenvolvido para o monitoramento da planta FV, a comunicação Modbus entre o Arduino Mega e os inversores ocorreu satisfatoriamente assim como a transferência de dados para a plataforma ThingSpeak, possibilitando o monitoramento remoto.

Considerando que geralmente o monitoramento de usinas utiliza um dispositivo de transferência de dados para cada inversor instalado, o sistema proposto no presente trabalho além de apresentar baixo custo frente as soluções comerciais, possui a vantagem de utilizar apenas um *hardware* para realizar a aquisição e transferência de dados de todos os inversores conectados ao barramento RS485, reduzindo ainda mais os custos com componentes eletrônicos.

A plataforma escolhida Thingspeak para armazenamento de dados atendeu às expectativas, registrando e disponibilizando os dados dos sensores tanto da estação quanto dos parâmetros de geração da usina solar FV. Entretanto, a principal limitação da versão gratuita é o número de pontos de dados disponíveis, impossibilitando que todos os parâmetros de geração

aquisitados possam ser monitorados remotamente. Além disso, não se recomenda que o acesso à plataforma seja disponibilizado ao público, uma vez que uma possível falta de domínio do usuário sobre a plataforma possa levá-lo a desconfigurar os canais, compromentendo o recebimento dos dados de forma correta.

O ThingView se mostrou como uma boa alternativa de monitoramento via *smartphone*, ampliando as formas de monitoramento.

De forma geral, o sistema desenvolvido atendeu ao propósito do respectivo tema do Projeto de Pesquisa e Desenvolvimento (P&D), permitindo o monitoramento dos parâmetros de forma remota e também através de um sistam SCADA. Os dados coletados já puderam ser utilizados em estudos de qualidade da energia em outros temas do projeto de P&D.

A principal contribuição dos sistemas propostos no presente trabalho é a possibilidade de realizar o monitoramento tanto de uma estação solarimétrica quanto de uma usina solar FV a um custo consideravelmente baixo em comparação as soluções comerciais. Vale salientar que, além do baixo custo, o sistema desenvolvido para a usina solar FV utiliza apenas um *hardware* para realizar a aquisição de dados de todos os inversores.

Embora o sistema desenvolvido tenha apresentado funcionamento satisfatório, reconhece-se que algumas melhorias podem ser realizadas futuramente com a finalidade de aprimorar o sistema. Desta forma, como sugestão de trabalhos futuro está:

- a utilização dos dados obtidos pelos sistemas propostos com o objetivo de desenvolver mecanismos que viabilizem o gerenciamento e operação eficiente da usina solar FV;
- o desenvolvimento de uma plataforma com *layout* em que todos os parâmetros aquisitados dos inversores possam ser visualizados pelo usuário;
- confecção de placas de circuito impresso de forma a conferir aos sistemas maior robustez e compacidade;
- o desenvolvimento de uma interface que acesse e disponibilize ao usuário as informações do banco de dados configurado no ScadaBR, viabilizando o acesso ao histórico de dados pelo usuário.

7. REFERÊNCIAS

ABSOLAR. Notícias Externas: **Especialistas defendem isonomia tributária para desenvolver setor de energia solar.** 2019. Disponível em: http://www.absolar.org.br/noticia/noticias-externas/especialistas-defendem-isonomia-tributaria-para-desenvolver-setor-de-energia-solar.html

ABSOLAR. Notícias Externas: **Incentivos fiscais para geradores de energia solar passam a ter validade até 2022**. 2020. Disponível em: http://www.absolar.org.br/noticia/noticias-externas/incentivos-fiscais-para-geradores-de-energia-solar-passam-a-ter-validade-ate-2022.html.

ADEH, E. H. GOOD, S. P. CALAF, M. HIGGINS, C. W. **Solar PV Power Potential is Greatest over croplands**. *Sci Rep* 9, 11442 (2019).

ALBUQUERQUE, L. D. **Sistema de conexão e supervisão de painéis solares em microgrid de corrente contínua.** Dissertação de mestrado. 90 f. Programa de Pós-Graduação em Energia Elétrica. Universidade Federal do Rio Grande do Norte. Natal, 2017. (1)

ALMEIDA, R. C. **Proposta de Sistema de Segurança Interativo Baseado em Conceitos de Internet of Things.** Trabalho de Conclusão de Curso. Universidade de São Carlos. 2016.

ALVES, R. H. F. **Estudo de Geração Fotovoltaica Distribuída: Análise Econômica e o Uso de Redes Neurais Artificiais.** Dissertação de Mestrado. Universidade Federal de Goiás. 2017.
ANAND, R., PACHAURI, R. K., GUPTA, A. e CHAUHAN, Y. K., "**Design and analysis of a low cost PV analyzer using Arduino UNO**". 2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES), pp. 1-4. Delhi, 2016.

ANEEL. **Resolução Normativa 687**. 2015. Disponível em: http://www2.aneel.gov.br/cedoc/ren2015687.pdf

ANEEL. **Chamada nº 001/2016 Projeto Prioritário de Eficiência Energética e estaratégico de P&D: "EFICIÊNCIA ENERGÉTICA E MINIGERAÇÃO E MINSTITUIÇÕES PÚBLICAS DE EDUCAÇÃO SUPERIOR".** 2016.

ANEEL. **GERACAO DISTRIBUIDA**: UNIDADES CONSUMIDORAS COM GERAÇÃO DISTRIBUÍDA. 2020. Disponível em: http://www2.aneel.gov.br/scg/gd/GD_Fonte.asp

ASOWATA, O. SWART, J. PIENAAR, C. **Evaluating the effect of orientation angles on the output power of a stationary photovoltaic panel**. Article. Department of Electronic Engineering, Vaal University of Technology. Journal of Renewable and Sustainable Energy. 2014.

BASU, S. DEBNATH, A. K. **Power plant instrumentation and control handbook: A guide to Thermal Power Plants.** 2ª Ed. Academic Press, 2015.

BESSO, R. **Sistema solar fotovoltaico conectado à rede – estudo de caso no Centro de Tecnologia da UFRJ.** Monografia. Universidade Federal do Rio de Janeiro. Rio de Janeiro, fevereiro, 2017.

BOGAN, H. **Design Of Zigbee Based Wireless Online Monitoring System For Photovoltaic Power Systems.** 2nd World Conference on Technology, Innovation and Entrepreneurship. Istanbul, Turkey, 2017.

BUENO, P. H. **Modelagem analítica e numérica semi-empírica de células fotovoltaicas.** Dissertação de mestrado. Programa de pós graduação em Engenharia Elétrica. Universidade Federal de Minas Gerais. Belo Horizonte. Agosto, 2016.

CANTOR, Guillermo Andrés Rodríguez. **Influência dos Fatores Climáticos no Desempenho de Módulos Fotovoltaicos em Regiões de Clima Tropical**. Dissertação de Mestrado. 177 f. Universidade Federal da Paraíba. João Pessoa, 2017.

CARVALHO, L. **Raspberry Pi: o que é, para que serve e como comprar.** Disponível em: https://olhardigital.com.br/noticia/raspberry-pi-o-que-e-para-que-serve-e-como-comprar/82921. 2019.

CHAAR, L. E. RASHID, M. H. **Alternative Energy in Power Electronics**: **Photovoltaic System Conversion.** Butterworth-Heinemann. 1ª Ed. 2015.

CRESESB**. Radiação Solar e Energia Solar Fotovoltaica**. 2008.

CUNHA, C. H. O. BATISTA, L. S. C. **Aplicação de Internet of Things no desenvolvimento de um sistema de monitoramento e gerenciamento de energia elétrica. Artigo Científico.** Revista de Informática Aplicada, Volume 14, Número 2, 2018.

DATASHEET MAX 485. Disponível em: https://ko.datasheetbank.com/datasheet-download/182056/1/Unspecified1/MAX485.

Eng&Tech. **EVOLUÇÃO DA CÉLULA FOTOVOLTAICA.** Disponível em: https://www.eet.eng.br/evolucao-da-celula-fotovoltaica/. 2019.

EPE. Empresa de Pesquisa Energética. **EXPANSÃO DA GERAÇÃO EMPREENDIMENTOS FOTOVOLTAICOS: Instruções para Solicitação de Cadastramento e Habilitação Técnica com vistas à participação nosLeilões de Energia Elétrica**. 2016. Disponível em: http://recursosolar.geodesign.com.br/PDF/EPE-DEE-RE-065_2013-r3_UFV.pdf

ETAMALY, A. M. AHMED, M. A. ALOTAIBI, M. A. ALOLAH, A. I. KIM, Y. **Performance of Communication Network for Monitoring Utility Scale Photovoltaic Power Plants**. Artigo Científico. Revista Energies. 2020.

FANOURAKIS, S. WANG, K. MacCARTHY, P. JIAO, L. **Low-Cost Data Acquisition Systems for Photovoltaic System Monitoring and Usage Statistics.** Journal of Energy and Power Engineering, páginas 719-728. 2017.

FEITOSA, K. A. MOTA, A. A**. Redes de Sensores Sem Fio para Monitoramento da Incidência Solar Distribuída em Edificações**. III Encontro de Iniciação em Desenvolvimento Tecnológico e Inovação da PUC-Campinas, 2013.

FilipeFlop. Tutorial: Como programar o módulo ESP8266 NodeMCU. 2016. Disponível em: https://www.filipeflop.com/produto/modulo-wifi-esp8266-nodemcu-esp-12/

FLUKE ACADEMY. Artigo do Blog: **A importância da temperatura para os sistemas fotovoltaicos.** 2020. Disponível em: https://www.flukeacademy.com.br/blog/post/52/a_import%C3%A2ncia_da_temperatura_para_os_sistemas_fotovoltaicos.

Folha de dados do módulo CS6U da Canadian Solar.

GADELHA, E. D. **Desenvolvimento De Um Protótipo Para Aquisição de Dados de Produção Industrial Utilizando Protocolo Modbus Rtu Em Uma Rede Rs-485 e Interface Gráfica Integrada Ao Gerenciador De Banco de Dados Mysql**. Trabalho de Conclusão de Curso. Universidade Federal do Ceará. 2019.

GUSA, R. F. SUNANDA, W. DINATA, I. HANDAYANI, T.P. **Monitoring System for Solar Panel Using Smartphone Based on Microcontroller.** Artigo Científico. 2nd International Conference on Green Energy and Applications. 2018.

HALMEMAN, R. J. **Desenvolvimento de um Sistema para Monitoramento Remoto em Centrais de Microgeração Fotovoltaica.** 202 f. Tese de doutorado. Universidade Estadual Paulista “Julio De Mesquita Filho”. Botucatu, 2014.

INPE, **Altas Brasileiro de Energia Solar.** 2ª Edição, São Paulo, 2017.

JUNIOR, A. C. L. **Transmissor de Temperatura Multicanal com Comunicação RS485 – MODBUS**. Dissertação de Mestrado. Universidade Federal do Triângulo Mineiro. 2018.

INFORBATISTA. MAX 3232 RS232 A TTL MÓDULO CONVERSOR DE PORTA SERIAL . Disponível em: http://shop.inforbatista.pt/pt/outros/4095-max3232-rs232-a-ttl-modulo-conversor-de-porta-serial.html

KANDIMALLA, J. KISHORE, D. R. **Web Based Monitoring of Solar Power Plant Using Open Source IOT Platform Thingspeak and Arduino.** International Journal for Modern Trends in Science and Technology Vol. 03, Issue nº 04, 2017, pp 16-21.

KRISHNA, M. S. R. DINESH, K. SHANBOG, N. S. **Low Cost Remote Monitoring of Solar Plant through RS485 Communication.** International Journal of Innovative Technology and Exploring Engineering (IJITEE). Volume 8, 2019.

KRÖGER-VODDE, A. ARMBRUSTER, A. HADEK, V. Distributed vs Central Inverters - A comparison of monitores PV Systems for the 25TH EU PVSEC / WCPEC-5 A.

MACÊDO, W. N. **Análise do fator de potência de dimensionamento do inversor aplicado a sistemas fotovoltaicos conectados à rede.** Tese de doutorado. Universidade de São Paulo. 2006.

MANUAL SCADABR. **Manual do Software.** Fundação Centros de Referência em Tecnologias Inovadoras. 2010.

MARTINS, J.M. **Sistema de monitoramento de dados provenientes da energia fotovoltaica através de uma plataforma de aquisição e controle.** Monografia. Universidade Federal Rural da Amazônia. 2019.

MATHWORKS. Disponível em https://www.mathworks.com/products/thingspeak.html. 2019.

MAXIM INTEGRATED. **Tutorials 763: RS-485 Cable Specification Guide.** 2001. Disponível em: https://www.maximintegrated.com/en/design/technical-documents/tutorials/7/763.html.

MELO, R. R. M. **Concepção de um sistema de propulsão elétrica para um trator de 9kW adequado para agricultura familiar**. Tese de Doutorado. Universidade Federal do Ceará. 2019.

MELO, R. C. S. M. **Sistema de monitoramento de consumo de água utilizando o protocolo de comunicação MQTT.** Monografia. Universidade Federal de Uberlândia. 2018.

MINHACASASOLAR. 6 tipos de painéis solares para você escolher o seu. Disponível em: http://blog.minhacasasolar.com.br/tipos-de-paineis-solares/. 2018.

MME. **Energia Solar no Brasil e Mundo** (Ano de referência – 2016). Departamento de informações e estudos energéticos. 2017.

MODICON. **Modicon Modbus Protocol Reference Guide**. 1996.

KHATRI, P. GUPTA, K. K. GUPTA, R. K. **Raspberry Pi-based smart sensing platform fordrinking-water quality monitoring system: a Pythonframework approach.** Drink. Scientific Article. Water Eng. Science, 12, 31–37, 2019.

MOURA, V. V. PEREIRA, R. I. JUCÁ, S. C. S. **IoT Embedded System for Data Acquisition using MQTT Protocol.** Universidade Federal do Ceará. International Journal of Computer Applications (0975 - 8887) Volume 182 - No.11. 2018.

NAKANO, A. **Simulação de desempenho energético de tecnologias fotovoltaicas em fachadas de edifício no município de São Paulo.** Dissertação de mestrado. Mestrado em Ciências. Universidade de São Paulo. São Paulo, 2017.

NASCIMENTO, R. L. **Energia Solar no Brasil: situação e perspectivas**. Estudo Técnico. Consultoria Legislativa. 2017.

NOVAK, M. V. **Análise de modelos matemáticos de temperatura de módulos fotovoltaicos e avaliação energética a partir de dados da casa solar eficiente.** Monografia. 55 f. curso de Engenharia de Energia. Universidade Federal De Santa Catarina. Araranguá, 2016.

NOVUS. **Conceitos Básicos de RS485 e RS422**. Disponível em: https://www.novus.com.br/downloads/Arquivos/conceitos%20b%C3%A1sicos%20de%20rs485%20e%20rs422.pdf

OLIVEIRA, G. F. S**istema de aquisição de dados agrometeorológicos utilizando estações de sensores sem fio.** Monografia. Passo Fundo. 2018.

ORTEGA, L. L. M. **Conversão Fotovoltaica: Comparação de modelos de desempenho.** Dissertação de Mestrado. Programa de pós-graduação em Metrologia. Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro, 2013.

PEROZA, J**. Caracterização elétrica de módulos fotovoltaicos de distintas tecnologias a partir de ensaios com simulador solar e iluminação natural.** Monografia. 97 f. Universidade Federal de Santa Catarina. Araranguá, 2015.

PINHO, J. T. GALDINO, M. A. **Manual de Engenharia para Sistemas Fotovoltaicos**. CEPEL – CRESESB. 2014.

SOUZA, N. T. **Estação meteorológica utilizando azure cloud e raspberry pi.** Monografia. Universidade Federal de Uberlândia. Patos de Minas, Minas Gerais. 2019.

REGES, J. P. **Desenvolvimento de um sistema de aquisição de dados para sistemas fotovoltaicos.** Dissertação de Mestrado. 119 f. Instituto Federal De Educação, Ciência e Tecnologia Do Ceará. Fortaleza, 2017.

REIS, A. A. **Aplicação de internet das coisas no monitoramento de corrente, tensão e temperatura em moto de indução trifásico**. Dissertação de mestrado. Universidade Federal de Uberlândia. 2019.

RIBEIRO, A. L. **Desenvolvimento de um Sistema de Aquisição de Dados e um Programa Supervisório para Monitoramento de Geração de Energia Fotovoltaica**. 68 f. Monografia. Centro Federal de Educação Tecnológica de Minas Gerais Unidade Araxá. Araxá, 2017.

RIBEIRO, J. C. C. Desenvolvimento de sistema de gerenciamento, supervisão, controle e aquisição de dados da Microrrede universitária do Campus do Pici-UFC. Dissertação de Mestrado. Universidade Federal do Ceará. 2020.

RICARDO JUNIOR, O. **Sistema de monitoramento residencial baseado em Internet das Coisas.** Monografia.Universidade Estadual de Londrina. 2017.

ROBOCORE. Tutorial: Como programar o NodeMCU com Arduino IDE. 2017.

ROCHA, L. P. **Sistema embarcado de comunicação em ambiente corporativo de fábrica.** Trabalho de conclusão de curso. Universidade de Caxias do Sul. 2017.

RUSCHEL, C. S. **Desenvolvimento de Software para Supervisão de Usinas Solares Fotovoltaicas.** 106 f. Dissertação de Mestrado. Universidade Federal do Rio Grande do Sul. Porto Alegre, 2015.

SHARIFF, F. RAHIM, N. A. PING, H. W. **Photovoltaic Remote Monitoring System Based on GSM.** IEEE Conference on Clean Energy and Technology (CEAT), 2013.

SYAFII. PUTRA, R. PURA, H. **Online Monitoring of Grid Connected Residential Photovoltaic System using Zigbee and Web Server.** Indonesian Journal of Electrical Engineering and Computer Science Vol. 7, No. 3, 2017, pp. 668- 675.

THEE, Q. Y. **Installation of a Three-Phase Grid-Tied Photovoltaic System in a Tropical Area and its Performance Ratio**. Annual International Conference on Sustainable Energy and Environmental Sciences. 2014.

TINA, G. M. COSENTINO, F. VENTURA, C. **Monitoring and diagnostics of photovoltaic power plants.** Artigo Científico. 2016. Disponível em: https://www.researchgate.net/publication/301269139_Monitoring_and_Diagnostics_of_Photovoltaic_Power_Plants?enrichId=rgreq-ac49b6857a9ec821f16e3ec86daff617-XXX&enrichSource=Y292ZXJQYWdlOzMwMTI2OTEzOTtBUzo1NzQzMTI3ODYwODM4NDBAMTUxMzkzODI0NjA4MA%3D%3D&el=1_x_2&_esc=publicationCoverPdf

TOLMASQUIM, M. T. **Energia Renovável. Hidrálica, Biomassa, Eólica, Solar, Oceânica.** Empresa de Pesquisa Energética (EPE). 2016.

VALENTE, M. A. S. **Caracterização Automática de um Painel Fotovoltaico.** Dissertação de mestrado. 82 f. Departamento de Engenharia Eletrotécnica. Universidade Nova de Lisboa. Lisboa, 2011.

VICTOR, J. F. L. JUCÁ, S. C. S. PEREIRA, R. I. S. CARVALHO, P. C. M. FERNÁNDEZ-RAMÍREZ, L. M. **IoT Monitoring systems applied to photovoltaic generation: The relevance for increasing decentralized plants.** Renewable Energy and Power Quality Journal. Vol. Nº 17. 2019.

ZAGO, R. M. **Sistema de baixo custo para monitoramento da geração de energia solar com conexão para Internet das Coisas**. Dissertação de Mestrado. 153 f. Universidade Estadual de Campinas. Campinas, 2018.

WOFFORD, Z. **Design of Remote Datalogger Connection and Live Data Tweeting System.** Monografia. University of Arkansas. 2019.

APÊNDICE A

```
/*Firmware de comunicação entre o NODEMCU e o datalogger CR300 da Campbell Scientifics
 * Protocolo: Modbus RTU via RS232
 * Baud rate: 9600 bps ///Slave (ID): 1///Offset: 0///
 * Formato dos dados: float de 4 bytes com ordem trocada dos bytes
*/
//Incluindo bibliotecas
#include <ModbusMaster232.h>
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ThingSpeak.h>
#include <SoftwareSerial.h>

//DADOS DA REDE WIFI E DO THIGNSPEAK
String AP = "REDE";       // NOME DA REDE WIFI
String PASS = "SENHA"; // SENHA DA REDE
String API1 = “CHAVEAPI";  // API KEY DO CANAL DOS SENSORES
String HOST = "api.thingspeak.com";
String PORT = "80";

ModbusMaster232 node(1);
WiFiClient client;

  //NOME DAS VARIÁVEIS
  float Tempainel;//ModbusVar(1)
  float Bateria; //ModbusVar(2)
  float tempar;//ModbusVar(3)
  float Umidade;//ModbusVar(4)
  float tempmod;//ModbusVar(5)
  float velvento;//ModbusVar (6)
  float radglob;//ModbusVar(7)
```

```
  float radifusa;//ModbusVar(8)
  float radifcor;//ModbusVar(9)
  float radireta;//ModbusVar(10)

void setup() {
   Serial.begin(9600);
   node.begin(9600);
   ThingSpeak.begin(client);
   WiFi.begin(AP, PASS);
//CONECTANDO A REDE WIFI
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  if (WiFi.status() == WL_CONNECTED) {
     Serial.print("Conectado a rede:");
     Serial.print(AP);
  }
}

void loop() {

  uint8_t j, result;
  uint16_t data[2];

//DECLARAÇÃO DE VARIÁVEIS AUXILIARES
 float Tempainel = 0;//ModbusVar(1)
  float Bateria = 0; //ModbusVar(2)
  float tempar = 0;//ModbusVar(3)
  float Umidade = 0;//ModbusVar(4)
  float tempmod = 0;//ModbusVar(5)
  float velvento = 0;//ModbusVar (6)
  float radglob = 0;//ModbusVar(7)
```

```
  float radifusa = 0;//ModbusVar(8)
  float radifcor = 0;//ModbusVar(9)
  float radireta = 0;//ModbusVar(10)

float v1 = 0;
float v2 = 0;
float v3 = 0;
float v4 = 0;
float v5 = 0;
float v6 = 0;
float v7 = 0;
float v8 = 0;
float v9 = 0;
float v10 = 0;
float v11 = 0;
float v12 = 0;
float v13 = 0;
float v14 = 0;
float v15 = 0;
int i = 0;
int k = 0;

for (k = 0; k <60; k++){
//TEMPERATURA DO AR
 result = node.readHoldingRegisters (4, 2);
    if (result == node.ku8MBSuccess)
    i = i  + 0;
    {      for(j = 0; j <2; j ++)
       {        data[j] = node.getResponseBuffer (j);
       }
 Serial.println(" Temperatura do ar (ºC) ");
tempar = *((float* ) data);
 v3 = v3 + tempar;
```

```
Serial.print(" ");
Serial.println(tempar);
}
node.clearResponseBuffer();
delay(50);

//UMIDADE
result = node.readHoldingRegisters (6, 2);
    if (result == node.ku8MBSuccess)
    {for(j = 0; j <2; j ++)
        {data[j] = node.getResponseBuffer (j);}
Serial.println("Umidade do ar(%)");
Umidade = *((float* ) data);
v4 = v4 + Umidade;
Serial.print(" ");
Serial.println(Umidade);
    }
node.clearResponseBuffer();
delay(50);

//TEMPERATURA DO MÓDULO FV
result = node.readHoldingRegisters (8, 2);
    if (result == node.ku8MBSuccess)
    {for(j = 0; j <2; j ++)
        {data[j] = node.getResponseBuffer (j);
        }
Serial.println("Temperatura do módulo (ºC) ");
 tempmod = *((float* ) data);
 v5 = v5 + tempmod;
Serial.print(" ");
Serial.println(tempmod);
}
node.clearResponseBuffer();
```

```
delay(50);

//VELOCIDADE DO VENTO
result = node.readHoldingRegisters (10, 2);
    if (result == node.ku8MBSuccess)
    {for(j = 0; j <2; j ++)
        {data[j] = node.getResponseBuffer (j);
        }
Serial.println("Velocidade do vento (m/s) ");
velvento = *((float* ) data);
v6 = v6 + velvento;
Serial.print(" ");
Serial.println(velvento);
}
node.clearResponseBuffer();
delay(50);

//RADIAÇÃO GLOBAL
result = node.readHoldingRegisters (12, 2);
    if (result  == node.ku8MBSuccess)
    {for(j = 0; j <2; j ++)
        {data[j] = node.getResponseBuffer (j);}
Serial.println("Radiação global (W/m2)");
radglob = *((float* ) data);
v7 = v7 + radglob;
Serial.print(" ");
Serial.println(radglob);
}
node.clearResponseBuffer();
delay(50);

//RADIAÇÃO DIFUSA
result = node.readHoldingRegisters (14, 2);
```

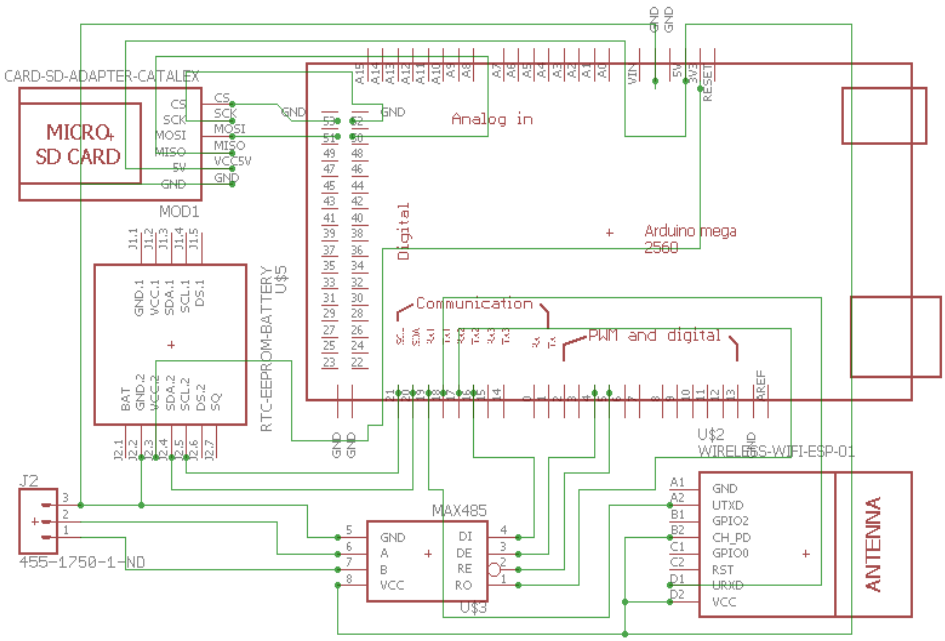
```
    if (result == node.ku8MBSuccess)
    {for(j = 0; j <2; j ++)
       {data[j] = node.getResponseBuffer (j);}
Serial.println("Radiação difusa (W/m2)");
radifusa = *((float* ) data);
v8 = v8 + radifusa;
Serial.print(" ");
Serial.println(radifusa);
}
node.clearResponseBuffer();
delay(50);

//RADIAÇÃO DIFUSA CORRIGIDA
result= node.readHoldingRegisters (16, 2);
    if (result == node.ku8MBSuccess)
    {for(j = 0; j <2; j ++)
       {data[j] = node.getResponseBuffer (j);
       }
Serial.println(" radiação difusa corrigida (W/m2) ");
 radifcor = *((float* ) data);
 v9 = v9 + radifcor;
Serial.print(" ");
Serial.println(radifcor);
}
node.clearResponseBuffer();
delay(50);

//RADIAÇÃO DIRETA
result = node.readHoldingRegisters (18, 2);
    if (result == node.ku8MBSuccess)
    {   for(j = 0; j <2; j ++)
       {  data[j] = node.getResponseBuffer (j);
       }
```

```
Serial.println("Radiação direta (W/m2)");
 radireta = *((float* ) data);
 v10 = v10 + radireta;
Serial.print(" ");
Serial.println(radireta);
    }
node.clearResponseBuffer();
delay (1000);
}

//CÁLCULO DE MÉDIAS
float Temp1 = v3/i;
float Umity = v4/i;
float Temp2 = v5/i;
float VVento = v6/i;
float RadGlobal = v7/i;
float RaDifusa = v8/i;
float RaDifusaCorrigida = v9/i;
float RaDireta = v10/i;

//ENVIO DE DADOS PARA O THINGSPEAK
WiFiClient client;
String datapck="field1="+String(Temp1) +"&field2="+String(Umity)
+"&field3="+String(Temp2) +"&field4="+String(VVento)
+"&field5="+String(RadGlobal)+"&field6="+String(RaDifusa)+"&field7="+String(RaDifusa
Corrigida)+"&field8="+String(RaDireta);

if (client.connect(HOST, 80)) {
client.println("POST /update HTTP/1.1");
client.println("Host: api.thingspeak.com");
client.println("Connection: close");
client.println("User-Agent: ESP32WiFi/1.1");
client.println("X-THINGSPEAKAPIKEY: "+ API1);
```

```
client.println("Content-Type: application/x-www-form-urlencoded");
client.print("Content-Length: ");
client.print(datapck.length());
client.print("\n\n");
client.print(datapck);
}
delay(500);
client.stop();
Serial.println("ThingSpeak");
Serial.println("Fim do loop");
}
```

APÊNDICE B

Esquemático do *hardware* de aquisição de dados dos parâmetros de geração fotovoltaica.



APÊNDICE C

Firmware implementado ao Arduino para leitura de registradores dos inversores da usina solar FV.

```
#include <ThingSpeak.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <WiFiServer.h>
#include <WiFiUdp.h>
#include <SD.h>
#include <SPI.h>
#include <SerialESP8266wifi.h>
#include <AltSoftSerial.h>
#include <ESP8266.h>
#include <SD.h>
#include <SPI.h>
#include "RTClib.h"
#include <Wire.h>


const int chipSelect = 53;
File usina;
RTC_DS1307 rtc;

String AP = "REDE";     // AP NAME Teste_DTI
String PASS = "xxxxx"; // AP PASSWORD  teste@2019@solar
String API1 = "API1";   // Write API KEY
String API2 = "API2";   // Write API KEY
String API3 = "API3";   // Write API KEY
String HOST = "api.thingspeak.com";
String PORT = "80";
int countTrueCommand;
```

```
int countTimeCommand;
boolean found = false;

#include <SoftwareSerial.h>
#include <ModbusMaster.h>

#define MAX485_DE      3
#define MAX485_RE_NEG  2

ModbusMaster node;

String c, texto;
String ID, REGISTER, VALUE;

void preTransmission()
{
  digitalWrite(MAX485_RE_NEG, 1);
  digitalWrite(MAX485_DE, 1);
}

void postTransmission()
{
  digitalWrite(MAX485_RE_NEG, 0);
  digitalWrite(MAX485_DE, 0);

}
void setup()
{
//Inciando portas para o max485
  Serial.begin(115200);
  Serial2.begin(19200); //max485_1

//ESP8266
```

```
  Serial1.begin(115200); //ESP8266
  sendCommand("AT",5,"OK");
  sendCommand("AT+CWMODE=1",5,"OK");
  sendCommand("AT+CWJAP=\""+ AP +"\",\""+ PASS +"\"",20,"OK");

//RTC
while(!Serial);
    if(! rtc.begin()) {
      Serial.println("Não foi possível encontrar o RTC");
      while (1);
    }
    else {
          rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
    }
    if(! rtc.isrunning()) {
      Serial.println("O RTC NÃO está sendo executado!");
    }

  //Cartão SD
  Serial.print("Inicializando o SD Card...");
  if(!SD.begin(chipSelect)) {
    Serial.println("falha na inicialização!");
    return;
  }
  Serial.println("inicialização feita.");

  //abrir arquivo
   usina =SD.open("DATA.csv", FILE_WRITE);
  // se o arquivo foi aberto, escreva para ele:
  if (usina) {
    Serial.println("Arquivo aberto ok");
    // imprima os títulos dos nossos dados
```

```
  usina.println("Data,Horario,VCC_INVERSOR1 (V), Corrente de linha_INVERSOR1 (A), Frequência_INVERSOR1 (Hz), Potência_INVERSOR1 (kW), I in_INVERSOR1 (A), VAN_INVERSOR1 (V), VBN_INVERSOR1 (V), VCN_INVERSOR1 (V), VAB_INVERSOR1 (V), VBC_INVERSOR1 (V), VCA (V)_INVERSOR1, VCC_INVERSOR2 (V), Corrente de linha_INVERSOR2 (A), Frequência_INVERSOR2 (Hz), Potência_INVERSOR2 (kW), I in_INVERSOR2 (A), VAN_INVERSOR2 (V), VBN_INVERSOR2 (V), VCN_INVERSOR2 (V), VAB_INVERSOR2 (V), VBC_INVERSOR2 (V), VCA (V)_INVERSOR2,VCC_INVERSOR3 (V), Corrente de linha_INVERSOR3 (A), Frequência_INVERSOR3 (Hz), Potência_INVERSOR3 (kW),I in_INVERSOR3 (A), VAN_INVERSOR3 (V), VBN_INVERSOR3 (V), VCN_INVERSOR3 (V), VAB_INVERSOR3 (V), VBC_INVERSOR3 (V), VCA (V)_INVERSOR3,VCC_INVERSOR4 (V), Corrente de linha_INVERSOR4 (A), Frequência_INVERSOR4 (Hz), Potência_INVERSOR4 (kW), I in_INVERSOR4 (A), VAN_INVERSOR4 (V), VBN_INVERSOR4 (V), VCN_INVERSOR4 (V), VAB_INVERSOR4 (V), VBC_INVERSOR4 (V), VCA (V)_INVERSOR4,VCC_INVERSOR5 (V), Corrente de linha_INVERSOR5 (A), Frequência_INVERSOR5 (Hz), Potência_INVERSOR5 (kW),I in_INVERSOR5 (A), VAN_INVERSOR5 (V), VBN_INVERSOR5 (V), VCN_INVERSOR1 (V), VAB_INVERSOR1 (V), VBC_INVERSOR1 (V), VCA (V)_INVERSOR1,VCC_INVERSOR1 (V), Corrente de linha_INVERSOR1 (A), Frequência_INVERSOR1 (Hz), Potência_INVERSOR1 (kW), Potência Aparente_INVERSOR1 (VA), Potência ReativaINVERSOR1 (VAr), I in_INVERSOR1 (A), VAN_INVERSOR1 (V), VBN_INVERSOR1 (V), VCN_INVERSOR1 (V), VAB_INVERSOR1 (V), VBC_INVERSOR1 (V), VCA (V)_INVERSOR1, ");
  }
  usina.close();

  pinMode(MAX485_RE_NEG, OUTPUT);
  pinMode(MAX485_DE, OUTPUT);

  digitalWrite(MAX485_RE_NEG, 0);
  digitalWrite(MAX485_DE, 0);
```

```
  node.preTransmission(preTransmission);
  node.postTransmission(postTransmission);

  delay(20);

}

bool state = true;

void loop()
{
uint8_t result;
 uint16_t data[6];

//variáveis inversor 1
float V1 = 0; float IL1 = 0; float F1 = 0; float P1 = 0;  float II1 = 0; int Aviso1 = 0; float VAN1 = 0; float VBN1 = 0;  float VCN1= 0;  float VAB1 = 0; float VBC1 = 0; float VCA1 = 0; float Energia1 =0; float fp1 = 0;
//variáveis inversor 2
float V2 = 0; float IL2 = 0; float F2 = 0;  float P2 = 0; float II2 = 0; int Aviso2 = 0; float VAN2 = 0; float VBN2 = 0; float VCN2= 0;  float VAB2 = 0; float VBC2 = 0; float VCA2 = 0; float Energia2 = 0; float fp2 = 0;
//variáveis inversor 3
float V3 = 0; float IL3 = 0; float F3 = 0;  float P3 = 0; float II3 = 0; int Aviso3 = 0; float VAN3 = 0; float VBN3 = 0;  float VCN3= 0;  float VAB3 = 0; float VBC3 = 0; float VCA3 = 0; float Energia3 = 0; float fp3 = 0;
//variáveis inversor 4
float V4 = 0; float IL4 = 0; float F4 = 0; float P4 = 0;  float II4 = 0; int Aviso4 = 0; float VAN4 = 0; float VBN4 = 0;  float VCN4= 0;  float VAB4 = 0; float VBC4 = 0; float VCA4 = 0; float Energia4 = 0; float fp4 = 0;
//variáveis inversor 5
```

```
float V5 = 0; float IL5 = 0; float F5 = 0;  float P5 = 0; float II5 = 0; int Aviso5 = 0; float VAN5 = 0; float VBN5 = 0;  float VCN5= 0;  float VAB5 = 0; float VBC5 = 0; float VCA5 = 0; float Energia5 =0; float fp5 = 0;
//variáveis inversor 6
float V6 = 0; float IL6 = 0; float F6 = 0; float P6 = 0; float II6 = 0; int Aviso6 = 0; float VAN6 = 0; float VBN6 = 0; float VCN6= 0;  float VAB6 = 0; float VBC6 = 0; float VCA6 = 0; float Energia6 = 0; float fp6 = 0;
//inversor 5
float v1 = 0;
float v2 = 0;
float v3 = 0;
float v4 = 0;
float v5 = 0;
float v6 = 0;
float v7 = 0;
float v8 = 0;
float v9 = 0;
float v10 = 0;
float v11 = 0;
float v12 = 0;
float v13 = 0;
float v14 = 0;
float v15 = 0;

//inversor 7
float b1 = 0;
float b2 = 0;
float b3 = 0;
float b4 = 0;
float b5 = 0;
float b6 = 0;
float b7 = 0;
float b8 = 0;
```

```
float b9 = 0;
float b10 = 0;
float b11 = 0;
float b12 = 0;
float b13 = 0;
float b14 = 0;
float b15 = 0;

//inversor 4
float c1 = 0;
float c2 = 0;
float c3 = 0;
float c4 = 0;
float c5 = 0;
float c6 = 0;
float c7 = 0;
float c8 = 0;
float c9 = 0;
float c10 = 0;
float c11 = 0;
float c12 = 0;
float c13 = 0;
float c14 = 0;
float c15 = 0;

//inversor 3
float d1 = 0;
float d2 = 0;
float d3 = 0;
float d4 = 0;
float d5 = 0;
float d6 = 0;
float d7 = 0;
```

```
float d8 = 0;
float d9 = 0;
float d10 = 0;
float d11 = 0;
float d12 = 0;
float d13 = 0;
float d14 = 0;
float d15 = 0;

//inversor 2
float f1 = 0;
float f2 = 0;
float f3 = 0;
float f4 = 0;
float f5 = 0;
float f6 = 0;
float f7 = 0;
float f8 = 0;
float f9 = 0;
float f10 = 0;
float f11 = 0;
float f12 = 0;
float f13 = 0;
float f14 = 0;
float f15 = 0;

//inversor 1
float x1 = 0;
float x2 = 0;
float x3 = 0;
float x4 = 0;
float x5 = 0;
float x6 = 0;
```

```
float x7 = 0;
float x8 = 0;
float x9 = 0;
float x10 = 0;
float x11 = 0;
float x12 = 0;
float x13 = 0;
float x14 = 0;
float x15 = 0;

//enviando requisição
byte j = 0;
int i = 0; int i2 = 0; int i3 = 0; int i4 = 0; int i5 = 0; int i6 = 0;

for (j = 0; j <60; j++){

//INICIANDO AQUISIÇÃO NO INVERSOR 1////////////////////////////////////////////////////////////////////
node.begin(5,Serial2);
delay(20);
result = node.readHoldingRegisters(0, 53);
Serial.println(result, HEX);
delay(100);
      if (result == node.ku8MBSuccess) {
      i = i+1;

      Serial.println("LEITURA Nº:");
      Serial.println(i);
        //tensão CC
      V1 = node.getResponseBuffer(3)/10.0; //reg 3
      Serial.println("VCC 1:  ");
      Serial.println(V1);
      x1 = x1 + V1;
      Serial.println("x1:");
```

```
Serial.println(x1);

//corrente de linha
IL1 = node.getResponseBuffer(4)/10.0; //reg 4
Serial.print("I de linha 1:  ");
Serial.println(IL1);
x2 = x2 + IL1;

//frequência
F1 = node.getResponseBuffer(5)/100.0; //reg 5
Serial.print("Frequência 1:  ");
Serial.println(F1);
x3 = x3 + F1;

//potência w
P1 = node.getResponseBuffer(7)/10.0; //reg 7
Serial.print("Potência 1:  ");
Serial.println(P1);
x4 = x4 + P1;

//FATOR DE POTENCIA  reg 9
fp1 = node.getResponseBuffer(9)/100.0; //REG 6
Serial.print("fator de potencia 1:  ");
Serial.println(fp1);//reg 7
x5 = x5 + fp1;

//corrente de entrada
II1 = node.getResponseBuffer(10)/10.0; //REG10
Serial.println("Corrente de entrada 7:  ");
Serial.println(II1);
x7 = x7 + II1;

//aviso
```

```
Aviso1 = node.getResponseBuffer(12);
Serial.println("Aviso 1:   ");
Serial.println(Aviso1);
x8 = x8 + Aviso1;

//VAN
VAN1 = node.getResponseBuffer(48);
Serial.println("VAN1:   ");
Serial.println(VAN1,1);
x9 = x9 + VAN1;

//VBN
VBN1 = node.getResponseBuffer(49);
Serial.println("VBN 1:   ");
Serial.println(VBN1,1);//reg 8
x10 = x10 + VBN1;

//VCN
VCN1 = node.getResponseBuffer(50); //REG50
Serial.println("VCN 1:   ");
Serial.println(VCN1,1);//reg 8
x11 = x11 + VCN1;

//VAB
VAB1 = node.getResponseBuffer(51); //REG51
Serial.println("VAB 1:   ");
Serial.println(VAB1,1);
x12 = x12 + VAB1;

//VBC
VBC1 = node.getResponseBuffer(52);
Serial.println("VBC 1:   ");
Serial.println(VBC1,1);
```

```
      x13 = x13 + VBC1;

      //VCA
      VCA1 = node.getResponseBuffer(53);
      Serial.println("VCA 1:   ");
      Serial.println(VCA1,1);
      x14 = x14 + VCA1;

      }//FIM IF

node.clearResponseBuffer();
delay(50);

//AQUISIÇÃO INVERSOR 2////////////////////////////////////////////////////////////////////////////
node.begin(2,Serial2);
delay(20);
result = node.readHoldingRegisters(0, 53);
Serial.println(result, HEX);
delay(100);
      if (result == node.ku8MBSuccess) {
      i2 = i2+1;
        Serial.println("LEITURA Nº:");
      Serial.println(i2);

       //tensão CC
      V2 = node.getResponseBuffer(3)/10.0; //reg 3
      Serial.println("VCC 2:  ");
      Serial.println(V2);
      f1 = f1 + V2;
      Serial.println("f1:");
      Serial.println(f1);

      //corrente de linha
```

```
IL2 = node.getResponseBuffer(4)/10.0; //reg 4
Serial.print("I de linha 2:  ");
Serial.println(IL2);
f2 = f2 + IL2;

//frequência
F2 = node.getResponseBuffer(5)/100.0; //reg 5
Serial.print("Frequência 2:  ");
Serial.println(F2);
f3 = f3 + F2;

//potência w
P2 = node.getResponseBuffer(7)/10.0; //reg 7
Serial.print("Potência 2:  ");
Serial.println(P2);
f4 = f4 + P2;

//FATOR DE POTENCIA  reg 9
fp2 = node.getResponseBuffer(9)/100.0; //REG 6
Serial.print("fator de potencia 2:  ");
Serial.println(fp2);//reg 7
f5 = f5 + fp2;

//corrente de entrada
II2 = node.getResponseBuffer(10)/10.0; //REG10
Serial.println("Corrente de entrada 2:  ");
Serial.println(II2);
f7 = f7 + II2;

//aviso
Aviso2 = node.getResponseBuffer(12);
Serial.println("Aviso 2:   ");
Serial.println(Aviso2);
```

```
f8 = f8 + Aviso2;

//VAN
VAN2 = node.getResponseBuffer(48);
Serial.println("VAN2:   ");
Serial.println(VAN2,1);
f9 = f9 + VAN2;

//VBN
VBN2 = node.getResponseBuffer(49);
Serial.println("VBN 2:   ");
Serial.println(VBN2,1);//reg 8
f10 = f10 + VBN2;

//VCN
VCN2 = node.getResponseBuffer(50); //REG50
Serial.println("VCN 2:   ");
Serial.println(VCN2,1);//reg 8
f11 = f11 + VCN2;

//VAB
VAB2 = node.getResponseBuffer(51); //REG51
Serial.println("VAB 2:   ");
Serial.println(VAB2,1);
f12 = f12 + VAB2;

//VBC
VBC2 = node.getResponseBuffer(52);
Serial.println("VBC 2:   ");
Serial.println(VBC2,1);
f13 = f13 + VBC2;

//VCA
```

```
      VCA2 = node.getResponseBuffer(53);
      Serial.println("VCA 2:   ");
      Serial.println(VCA2,1);
      f14 = f14 + VCA2;

      }//FIM IF

node.clearResponseBuffer();
delay(50);

//INICIANDO AQUISIÇÃO NO INVERSOR 3////////////////////////////////////////////////////////////////////
node.begin(3,Serial2);
delay(20);
result = node.readHoldingRegisters(0, 53);
Serial.println(result, HEX);
delay(100);
      if (result == node.ku8MBSuccess) {
      i3 = i3+1;

        Serial.println("LEITURA Nº:");
      Serial.println(i3);

       //tensão CC
      V3 = node.getResponseBuffer(3)/10.0; //reg 3
      Serial.println("VCC 3:  ");
      Serial.println(V3);
      d3 = d3 + V3;
      Serial.println("d1:");
      Serial.println(d1);

      //corrente de linha
      IL3 = node.getResponseBuffer(4)/10.0; //reg 4
      Serial.print("I de linha 3:  ");
```

```
Serial.println(IL3);
d2 = d2 + IL3;

//frequência
F3 = node.getResponseBuffer(5)/100.0; //reg 5
Serial.print("Frequência 3:  ");
Serial.println(F3);
d3 = d3 + F3;

//potência w
P3 = node.getResponseBuffer(7)/10.0; //reg 7
Serial.print("Potência 3:  ");
Serial.println(P3);
d4 = d4 + P3;

//FATOR DE POTENCIA  reg 9
fp3 = node.getResponseBuffer(9)/100.0; //REG 6
Serial.print("fator de potencia 3:  ");
Serial.println(fp3);//reg 7
d5 = d5 + fp3;

//corrente de entrada
II3 = node.getResponseBuffer(10)/10.0; //REG10
Serial.println("Corrente de entrada 3:  ");
Serial.println(II3);
d7 = d7 + II3;

//aviso
Aviso3 = node.getResponseBuffer(12);
Serial.println("Aviso 3:   ");
Serial.println(Aviso3);
d8 = d8 + Aviso3;
```

```
//VAN
VAN3 = node.getResponseBuffer(48);
Serial.println("VAN3:   ");
Serial.println(VAN3,1);
d9 = d9 + VAN3;

//VBN
VBN3 = node.getResponseBuffer(49);
Serial.println("VBN 3:   ");
Serial.println(VBN3,1);//reg 8
d10 = d10 + VBN3;

//VCN
VCN3 = node.getResponseBuffer(50); //REG50
Serial.println("VCN 3:   ");
Serial.println(VCN3,1);//reg 8
d11 = d11 + VCN3;

//VAB
VAB3 = node.getResponseBuffer(51); //REG51
Serial.println("VAB 3:   ");
Serial.println(VAB3,1);
d12 = d12 + VAB3;

//VBC
VBC3 = node.getResponseBuffer(52);
Serial.println("VBC 3:   ");
Serial.println(VBC3,1);
d13 = d13 + VBC3;

//VCA
VCA3 = node.getResponseBuffer(53);
Serial.println("VCA 3:   ");
```

```
        Serial.println(VCA3,1);
        d14 = d14 + VCA3;

        }//FIM IF

node.clearResponseBuffer();
delay(20);

//INVERSOR 4////////////////////////////////////////////////////////////////////////////////
node.begin(4,Serial2);
delay(20);
result = node.readHoldingRegisters(0, 53);
Serial.println(result, HEX);
delay(100);
        if (result == node.ku8MBSuccess) {
        i4 = i4+1;

   Serial.println("LEITURA Nº:");
        Serial.println(i4);

          //tensão CC
        V4 = node.getResponseBuffer(3)/10.0; //reg 3
        Serial.println("V4:  ");
        Serial.println(V4);
        c1 = c1 + V4;
        Serial.println("c1:");
        Serial.println(c1);

        //corrente de linha
        IL4 = node.getResponseBuffer(4)/10.0; //reg 4
        Serial.print("I de linha 4:  ");
        Serial.println(IL4);
        c2 = c2 + IL4;
```

```
//frequência
F4 = node.getResponseBuffer(5)/100.0; //reg 5
Serial.print("Frequência 4:  ");
Serial.println(F4);
c3 = c3 + F4;

//potência w
P4 = node.getResponseBuffer(7)/10.0; //reg 7
Serial.print("Potência 4:  ");
Serial.println(P4);
c4 = c4 + P4;

//FATOR DE POTENCIA  reg 9
fp4 = node.getResponseBuffer(9)/100.0; //REG 6
Serial.print("fator de potencia 4:  ");
Serial.println(fp4);//reg 7
c5 = c5 + fp4;

//corrente de entrada
float II4 = node.getResponseBuffer(10)/10.0; //REG10
Serial.println("Corrente de entrada 4:  ");
Serial.println(II4);
c7 = c7 + II4;

//aviso
Aviso4 = node.getResponseBuffer(12);
Serial.println("Aviso 4:   ");
Serial.println(Aviso4);
c8 = c8 + Aviso4;

//VAN
VAN4 = node.getResponseBuffer(48);
```

```
Serial.println("VAN4:   ");
Serial.println(VAN4,1);
c9 = c9 + VAN4;

//VBN
VBN4 = node.getResponseBuffer(49);
Serial.println("VBN 4:   ");
Serial.println(VBN4,1);//reg 8
c10 = c10 + VBN4;

//VCN
VCN4 = node.getResponseBuffer(50); //REG50
Serial.println("VCN 4:   ");
Serial.println(VCN4,1);//reg 8
c11 = c11 + VCN4;

//VAB
VAB4 = node.getResponseBuffer(51); //REG51
Serial.println("VAB 4:   ");
Serial.println(VAB4,1);
c12 = c12 + VAB4;

//VBC
VBC4 = node.getResponseBuffer(52);
Serial.println("VBC 4:   ");
Serial.println(VBC4,1);
c13 = c13 + VBC4;

//VCA
VCA4 = node.getResponseBuffer(53);
Serial.println("VCA 4:   ");
Serial.println(VCA4,1);
c14 = c14 + VCA4;
```

```
    }//FIM IF I4

node.clearResponseBuffer();
delay(20);

//////INVERSOR 5 ////////////////////////////////////////////////////////////////////////////////
node.begin(5,Serial2);
result = node.readHoldingRegisters(0, 53);
delay(100);
Serial.println(result, HEX);
    if (result == node.ku8MBSuccess) {
    i5 = i5+1;

      Serial.println("LEITURA Nº:");
    Serial.println(i5);
     //tensão CC
    V5 = node.getResponseBuffer(3)/10.0; //reg 3
    Serial.println("V5:  ");
    Serial.println(V5);
    v1 = v1 + V5;
    Serial.println("v1:");
    Serial.println(v1);

    //corrente de linha
    IL5 = node.getResponseBuffer(4)/10.0; //reg 4
    Serial.print("I de linha 5:  ");
    Serial.println(IL5);
    v2 = v2 + IL5;

    //frequência
    F5 = node.getResponseBuffer(5)/100.0; //reg 5
    Serial.print("Frequência 5:  ");
```

```
Serial.println(F5);
v3 = v3 + F5;

//potência w
P5 = node.getResponseBuffer(7)/10.0; //reg 7
Serial.print("Potência 5:  ");
Serial.println(P5);
v4 = v4 + P5;

//FATOR DE POTENCIA  reg 9
fp5 = node.getResponseBuffer(9)/100.0; //REG 6
Serial.print("fator de potencia 5 :  ");
Serial.println(fp5);//reg 7
v5 = v5 + fp5;

//corrente de entrada
float II5 = node.getResponseBuffer(10)/10.0; //REG10
Serial.println("Corrente de entrada 5:  ");
Serial.println(II5);
v7 = v7 + II5;

//aviso
Aviso5 = node.getResponseBuffer(12);
Serial.println("Aviso 5:   ");
Serial.println(Aviso5);
v8 = v8 + Aviso5;

//VAN
VAN5 = node.getResponseBuffer(48);
Serial.println("VAN 5:   ");
Serial.println(VAN5,1);
v9 = v9 + VAN5;
```

```
        //VBN
        VBN5 = node.getResponseBuffer(49);
        Serial.println("VBN 5:   ");
        Serial.println(VBN5,1);//reg 8
        v10 = v10 + VBN5;

        //VCN
        VCN5 = node.getResponseBuffer(50); //REG50
        Serial.println("VCN 5:   ");
        Serial.println(VCN5,1);//reg 8
        v11 = v11 + VCN5;

        //VAB
        VAB5 = node.getResponseBuffer(51); //REG51
        Serial.println("VAB5:   ");
        Serial.println(VAB5,1);
        v12 = v12 + VAB5;

        //VBC
        VBC5 = node.getResponseBuffer(52);
        Serial.println("VBC 5:   ");
        Serial.println(VBC5,1);
        v13 = v13 + VBC5;

        //VCA
        VCA5 = node.getResponseBuffer(53);
        Serial.println("VCA 5:   ");
        Serial.println(VCA5,1);
        v14 = v14 + VCA5;
        }//FIM IF

node.clearResponseBuffer();
delay(20);
```

```
////AQUISIÇÃO INVERSOR 6 (ADDRESS 2)////////////////////////////////////////////////////////////////
node.begin(6,Serial2);
//leitura dos registradores
result = node.readHoldingRegisters(0, 53);
delay(100);
  Serial.println(result, HEX);
      if (result == node.ku8MBSuccess) {
      i6 = i6+1;
        //tensão CC
      V6 = node.getResponseBuffer(3)/10.0; //reg 3
      Serial.println("V6:  ");
      Serial.println(V6);
      b1 = b1 + V6;
      Serial.println("b1:");
      Serial.println(b1);

      //corrente de linha
      IL6 = node.getResponseBuffer(4)/10.0; //reg 4
      Serial.print("I de linha 6:  ");
      Serial.println(IL6);
      b2 = b2 + IL6;

      //frequência
      F6 = node.getResponseBuffer(5)/100.0; //reg 5
      Serial.print("Frequência 6:  ");
      Serial.println(F6);
      b3 = b3 + F6;

      //potência w
      P6 = node.getResponseBuffer(7)/10.0; //reg 7
      Serial.print("Potência 6:  ");
      Serial.println(P6);
      b4 = b4 + P6;
```

```
//FATOR DE POTENCIA  reg 9
fp6 = node.getResponseBuffer(9)/100.0; //REG 6
Serial.print("fator de potencia 6:  ");
Serial.println(fp6);//reg 7
b5 = b5 + fp6;

//corrente de entrada
float II6 = node.getResponseBuffer(10)/10.0; //REG10
Serial.println("Corrente de entrada 6:  ");
Serial.println(II6);
b7 = b7 + II6;

//aviso
Aviso6 = node.getResponseBuffer(12);
Serial.println("Aviso 6:   ");
Serial.println(Aviso6);
b8 = b8 + Aviso6;

//VAN
VAN6 = node.getResponseBuffer(48);
Serial.println("VAN6:   ");
Serial.println(VAN6,1);
b9 = b9 + VAN6;

//VBN
VBN6 = node.getResponseBuffer(49);
Serial.println("VBN 6:   ");
Serial.println(VBN6,1);//reg 8
b10 = b10 + VBN6;

//VCN
VCN6 = node.getResponseBuffer(50); //REG50
```

```
        Serial.println("VCN 6:   ");
        Serial.println(VCN6,1);//reg 8
        b11 = b11 + VCN6;
        //VAB
        VAB6 = node.getResponseBuffer(51); //REG51
        Serial.println("VAB 6:   ");
        Serial.println(VAB6,1);
        b12 = b12 + VAB6;
        //VBC
        VBC6 = node.getResponseBuffer(52);
        Serial.println("VBC 6:   ");
        Serial.println(VBC6,1);
        b13 = b13 + VBC6;
        //VCA
        VCA6 = node.getResponseBuffer(53);
        Serial.println("VCA 6:   ");
        Serial.println(VCA6,1);
        b14 = b14 + VCA6;
        }//FIM IF
node.clearResponseBuffer();
delay(1000);
}
Serial.println("Num de leituras bem sucedidas: ");
Serial.println(i);

//CÁLCULO DAS MÉDIAS PARA OS DADOS DOS INVERSORES
//inversor 1

float VCC1 = x1/i;
float vab1 = x9/i;
float I1= x2/i;
float vbc1 = x10/i;
float vca1 = x11/i;
```

```
float van1 = x12/i;
float vbn1 = x13/i;
float vcn1 = x14/i;
float POT1 = x4/i;
float AVISO1 = x8/i;
float freq1 = x3/i;
float iin1 = x7/i;
float E1 = x15/i;
float FP1 = x5/i;

float VCC2 = f1/i2;
float vab2 = f9/i2;
float I2= f2/i2;
float vbc2 = f10/i2;
float vca2 = f11/i2;
float van2 = f12/i2;
float vbn2 = f13/i2;
float vcn2 = f14/i2;
float POT2 = f4/i2;
float AVISO2 = f8/i2;
float freq2 = f3/i2;
float iin2 = f7/i2;
float E2 = f15/i2;
float FP2 = f5/i;

float VCC3 = d1/i3;
float vab3 = d9/i3;
float I3 = d2/i3;
float vbc3 = d10/i3;
float vca3 = d11/i3;
float van3 = d12/i3;
float vbn3 = d13/i3;
float vcn3= d14/i3;
```

```
float POT3 = d4/i3;
float AVISO3 = d8/i3;
float freq3 = d3/i3;
float iin3 = d7/i3;
float E3 = d15/i3;
float FP3 = d5/i3;

float VCC4 = c1/i4;
float vab4 = c9/i4;
float I4 = c2/i4;
float vbc4 = c10/i4;
float vca4 = c11/i4;
float van4 = c12/i4;
float vbn4 = c13/i4;
float vcn4 = c14/i4;
float POT4 = c4/i4;
float AVISO4 = c8/i4;
float freq4 = c3/i4;
float iin4 = c7/i4;
float E4 = c15/i4;
float FP4 = c5/i4;

float VCC5 = v1/i5;
float vab5 = v9/i5;
float I5 = v2/i5;
float vbc5 = v10/i5;
float vca5 = v11/i5;
float van5 = v12/i5;
float vbn5 = v13/i5;
float vcn5 = v14/i5;
float POT5 = v4/i5;
float AVISO5= v8/i5;
float freq5 = v3/i5;
```

```
float iin5 = v7/i5;
float E5 = v15/i5;
float FP5 = v5/i5;

float VCC6 = b1/i6;
float vab6 = b9/i6;
float I6 = b2/i6;
float vbc6 = b10/i6;
float vca6 = b11/i6;
float van6 = b12/i6;
float vbn6 = b13/i6;
float vcn6 = b14/i6;
float POT6 = b4/i6;
float AVISO6 = b8/i6;
float freq6 = b3/i6;
float iin6 = b7/i6;
float E6 = b15/i6;
float FP6 = b5/i6;

//PACOTES PARA ENVIO AO THINGSPEAK
 String PACOTE1 = "GET /update?api_key="+ API1+"&field1"+"="+String(POT1)+"&field2"+"="+String(I1)+"&field3"+"="+String(van1) +"&field4"+"="+String(AVISO1)+"&field5"+"="+String(POT2)+"&field6"+"="+String(I2)+ "&field7"+"="+String(van2)+"&field8"+"="+String(AVISO2);//enviando pacote com dados do termopar, dht11, gps
 sendCommand("AT+CIPMUX=1",5,"OK");
 sendCommand("AT+CIPSTART=0,\"TCP\",\""+ HOST +"\","+ PORT,15,"OK");
 sendCommand("AT+CIPSEND=0," +String(PACOTE1.length()+4),4,">");
 Serial1.println(PACOTE1);
 delay(150);
 countTrueCommand++;
 sendCommand("AT+CIPCLOSE=0",5,"OK");
//parte 2
```

```
String PACOTE2 = "GET /update?api_key="+ API2+"&field1"+"="+String(POT3)+"&field2"+"="+String(I3)+"&field3"+"="+String(van3) +"&field4"+"="+String(AVISO3)+"&field5"+"="+String(POT4)+"&field6"+"="+String(I4)+ "&field7"+"="+String(van4)+"&field8"+"="+String(AVISO4);//enviando pacote com dados do termopar, dht11, gps
 //String getData = "GET /update?api_key="+ API +"&"+"field1"+"="+getSensortermopar();
 sendCommand("AT+CIPMUX=1",5,"OK");
 sendCommand("AT+CIPSTART=0,\"TCP\",\""+ HOST +"\","+ PORT,15,"OK");
 sendCommand("AT+CIPSEND=0," +String(PACOTE2.length()+4),4,">");
 Serial1.println(PACOTE2);
 delay(150);
 countTrueCommand++;
 sendCommand("AT+CIPCLOSE=0",5,"OK");
String PACOTE3 = "GET /update?api_key="+ API3+"&field1"+"="+String(POT5)+"&field2"+"="+String(I5)+"&field3"+"="+String(van5) +"&field4"+"="+String(AVISO5)+"&field5"+"="+String(POT6)+"&field6"+"="+String(I6)+ "&field7"+"="+String(van6)+"&field8"+"="+String(AVISO6);//enviando pacote com dados do termopar, dht11, gps
 //String getData = "GET /update?api_key="+ API +"&"+"field1"+"="+getSensortermopar();
 sendCommand("AT+CIPMUX=1",5,"OK");
 sendCommand("AT+CIPSTART=0,\"TCP\",\""+ HOST +"\","+ PORT,15,"OK");
 sendCommand("AT+CIPSEND=0," +String(PACOTE3.length()+4),4,">");
 Serial1.println(PACOTE3);
 delay(150);
 countTrueCommand++;
 sendCommand("AT+CIPCLOSE=0",5,"OK");
  delay(100);

 //inclusão na planilha
DateTime now = rtc.now();
   usina = SD.open("DATA.csv", FILE_WRITE);
  if (usina) {
    usina.print(now.year(), DEC);
```

```
    usina.print('/');
    usina.print(now.month(), DEC);
    usina.print('/');
    usina.print(now.day(), DEC);
    usina.print(',');
    usina.print(now.hour(), DEC);
    usina.print(':');
    usina.print(now.minute(), DEC);
    usina.print(':');
    usina.print(now.second(), DEC);
    usina.print(",");
  }
  Serial.print(now.year(), DEC);
  Serial.print('/');
  Serial.print(now.month(), DEC);
  Serial.print('/');
  Serial.println(now.day(), DEC);
  Serial.print(now.hour(), DEC);
  Serial.print(':');
  Serial.print(now.minute(), DEC);
  Serial.print(':');
  Serial.println(now.second(), DEC);
  usina.close();
  delay(100);

usina = SD.open("DATA.csv", FILE_WRITE);
  if (usina) {
    Serial.println("planilha aberta..");
    usina.println(VCC1);
    usina.print(",");
    usina.print(I1);
    usina.print(",");
    usina.print(freq1);
```

```
    usina.print(",");
    usina.print(P1);
    usina.print(",");
    usina.print(FP1);
    usina.print(",");
    usina.print(iin1);
    usina.print(",");
    usina.print(van1);
    usina.print(",");
    usina.print(vbn1);
    usina.print(",");
    usina.print(vcn1);
    usina.print(",");
    usina.print(vab1);
    usina.print(",");
    usina.print(vbc1);
    usina.print(",");
    usina.print(vca1);

    //INVERSOR 2
  usina.println(VCC2);
    usina.print(",");
    usina.print(I2);
    usina.print(",");
    usina.print(freq2);
    usina.print(",");
    usina.print(P2);
    usina.print(",");
    usina.print(FP2);
    usina.print(",");
    usina.print(iin2);
    usina.print(",");
    usina.print(van2);
```

```
usina.print(",");
usina.print(vbn2);
usina.print(",");
usina.print(vcn2);
usina.print(",");
usina.print(vab2);
usina.print(",");
usina.print(vbc2);
usina.print(",");
usina.print(vca2);
//INVERSOR 3
usina.println(VCC1);
usina.print(",");
usina.print(I1);
usina.print(",");
usina.print(freq1);
usina.print(",");
usina.print(P1);
usina.print(",");
usina.print(FP1);
usina.print(",");
usina.print(iin1);
usina.print(",");
usina.print(van1);
usina.print(",");
usina.print(vbn1);
usina.print(",");
usina.print(vcn1);
usina.print(",");
usina.print(vab1);
usina.print(",");
usina.print(vbc1);
usina.print(",");
```

```
usina.print(vca1);

usina.println(VCC1);
usina.print(",");
usina.print(I1);
usina.print(",");
usina.print(freq1);
usina.print(",");
usina.print(P1);
usina.print(",");
usina.print(FP1);
usina.print(",");
usina.print(iin1);
usina.print(",");
usina.print(van1);
usina.print(",");
usina.print(vbn1);
usina.print(",");
usina.print(vcn1);
usina.print(",");
usina.print(vab1);
usina.print(",");
usina.print(vbc1);
usina.print(",");
usina.print(vca1);

usina.println(VCC1);
usina.print(",");
usina.print(I1);
usina.print(",");
usina.print(freq1);
usina.print(",");
usina.print(P1);
```

```
usina.print(",");
usina.print(FP1);
usina.print(",");
usina.print(iin1);
usina.print(",");
usina.print(van1);
usina.print(",");
usina.print(vbn1);
usina.print(",");
usina.print(vcn1);
usina.print(",");
usina.print(vab1);
usina.print(",");
usina.print(vbc1);
usina.print(",");
usina.print(vca1);

usina.println(VCC1);
usina.print(",");
usina.print(I1);
usina.print(",");
usina.print(freq1);
usina.print(",");
usina.print(P1);
usina.print(",");
usina.print(FP1);
usina.print(",");
usina.print(iin1);
usina.print(",");
usina.print(van1);
usina.print(",");
usina.print(vbn1);
usina.print(",");
```

```
    usina.print(vcn1);
    usina.print(",");
    usina.print(vab1);
    usina.print(",");
    usina.print(vbc1);
    usina.print(",");
    usina.print(vca1);
      }
  usina.close();}
void sendCommand(String command, int maxTime, char readReplay[]) {
  Serial.print(countTrueCommand);
  Serial.print(". at command => ");
  Serial.print(command);
  Serial.print(" ");
  while(countTimeCommand < (maxTime*1))
  {  Serial1.println(command);//at+cipsend
  if(Serial1.find(readReplay))//ok
  {    found = true;
   break;}
  countTimeCommand++; }
  if(found == true)
  {  Serial.println("Ok");
  countTrueCommand++;
  countTimeCommand = 0; }
  if(found == false)
  {  Serial.println("Falha");
  countTrueCommand = 0;
  countTimeCommand = 0;
  }
  found = false;
 }
```