UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

GUILHERME SANT ANNA STRESSER

PROTÓTIPO DE SEGUIDOR SOLAR PARA USO DIDÁTICO COM MONITORAMENTO VIA APLICATIVO

MEDIANEIRA

2022

GUILHERME SANT ANNA STRESSER

PROTÓTIPO DE SEGUIDOR SOLAR PARA USO DIDÁTICO COM MONITORAMENTO VIA APLICATIVO

Sola tracker prototype for didactic use with app monitoring

Trabalho de conclusão de curso de graduação apresentada como requisito para obtenção do título de Bacharel em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientador: Filipe Marangoni

MEDIANEIRA

2022

GUILHERME SANT ANNA STRESSER

PROTÓTIPO DE SEGUIDOR SOLAR PARA USO DIDÁTICO COM MONITORAMENTO VIA APLICATIVO

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 20/junho/2022

_____

Filipe Marangoni
Doutorado
Universidade Tecnlógica Federal do Paraná

_____

Alberto Noboru Miyadaira
Doutorado
Universidade Tecnológica Federal do Paraná

_____

Alexandre Victor Casella
Mestrado
Universidade Tecnológica Federal do Paraná

MEDIANEIRA

2022

RESUMO

Com o aumento da procura pela geração de energia solar fotovoltaica é necessário o estudo e o desenvolvimento de tecnologias que possam aperfeiçoar e tornar essa área cada vez mais rentável. Um seguidor solar é um dispositivo que tem a função de movimentar (direcionar) o painel fotovoltaico para posicioná-lo sempre com o melhor ângulo em relação ao sol para obter a maior incidência de radiação solar melhorando sua eficiência e rendimento. No curso de Engenharia Elétrica, em algumas disciplinas (como as da área de Controle), muitas vezes acaba não sendo possível realizar uma abordagem prática, e a utilização de um protótipo construído para fins didáticos acaba sendo uma boa opção. Este presente trabalho tem o intuito de desenvolver um estudo sobre seguidores solares, microcontroladores, linguagem de programação C e HTML, comunicação entre dispositivos via IP e noções de eletrônica, para que seja possível o desenvolvimento de um protótipo de rastreador solar. Com o desenvolvimento deste protótipo futuros alunos das disciplinas da área de controle poderão desenvolver e implementar algoritmos e analisar o comportamento de uma planta real, para aprofundar e aperfeiçoar seus conhecimentos na área.

Palavras-chave: solar; protótipo; arduíno.

ABSTRACT

With the increase in demand for photovoltaic solar energy generation, it is necessary to study and develop technologies that can improve and make this area increasingly profitable. A solar tracker is a device that has the function of moving (directing) the photovoltaic panel to always position it with the best angle in relation to the sun to obtain the highest incidence of solar radiation, improving its efficiency and yield. In the Electrical Engineering course, in some subjects (such as those in the Control area), it is often not possible to carry out a practical approach, and the use of a prototype built for didactic purposes ends up being a good option. This present work aims to develop a study on solar trackers, microcontrollers, C and HTML programming language, communication between devices via IP and notions of electronics, so that it is possible to develop a prototype of a solar tracker. With the development of this prototype, future students of the disciplines of the control area will be able to develop and implement algorithms and analyze the behavior of a real plant, to deepen and improve their knowledge in the area.

Keywords: solar; prototype; arduino.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

%	Porcentagem
µA	Micro amperes
µA	Micro amperes
A	Ampères
kg	Quilo gramas
km/h	Quilômetros por hora
kW	Quilo Watts
KΩ	Quilos ohms
m	Metros
mA	Mili amperes
MB	Megabit
MHz	Mega hertz
º	Graus
ºC	Graus Celsius
V	Volts
ABNT	Associação Brasileira de Normas Técnicas
Coef.	Coeficiente
IBGE	Instituto Brasileiro de Geografia e Estatística
NBR	Normas Brasileiras
UTFPR	Universidade Tecnológica Federal do Paraná
ABNT	Associação Brasileira de Normas Técnicas
Coef.	Coeficiente
IBGE	Instituto Brasileiro de Geografia e Estatística
NBR	Normas Brasileiras
UTFPR	Universidade Tecnológica Federal do Paraná

SUMÁRIO

1 INTRODUÇÃO

Com o aumento da procura pela geração de energia solar fotovoltaica é necessário o estudo e o desenvolvimento de tecnologias que possam aperfeiçoar e tornar essa área cada vez mais rentável. Uma dessas tecnologias é chamada de Seguidor Solar, de acordo com Alexandru & Pozna (2008) um seguidor solar é um dispositivo que tem a função de movimentar (direcionar) o painel fotovoltaico para posicioná-lo sempre com o melhor ângulo em relação ao sol para obter a maior incidência de radiação solar melhorando sua eficiência e rendimento.

No curso de Engenharia Elétrica, em algumas disciplinas (como as da área de Controle), muitas vezes acaba não sendo possível realizar uma abordagem prática devido à falta de componentes ou recursos financeiros, falta de experiência dos alunos em “construir” protótipos ou por falta de tempo. Com isso acaba-se optando pelo uso somente de softwares para o aprendizado, porém por se tratar de uma área complexa, o uso de softwares pode se tornar um tópico abstrato para muitos alunos. Sendo assim, a utilização de kits ou protótipos prontos para o uso didático pode ser uma opção bastante viável.

O presente trabalho tem o intuito de desenvolver um estudo sobre seguidores solares, microcontroladores, linguagem de programação C e HTML, comunicação entre dispositivos via IP (“Internet Protocol”), e noções de eletrônica, para que seja possível o desenvolvimento de um protótipo de rastreador solar. O protótipo terá atuação por sensores, movimentado por dois eixos de rotação, controlado através de um microcontrolador da família Arduino, que tenha a capacidade de comunicação com qualquer dispositivo eletrônico que possua a função de comunicação IP. O objetivo é monitorar os parâmetros de tensão, corrente, temperatura, radiação UV, potência, e também, fazer o posicionamento de forma manual do seguidor. Sendo assim, com o desenvolvimento deste protótipo futuros alunos das disciplinas da área de controle poderão desenvolver e implementar algoritmos e analisar o comportamento de uma planta real, para aprofundar e aperfeiçoar seus conhecimentos na área. Além disso, este trabalho contribui com o estudo e o desenvolvimento de tecnologias relacionadas aos sistemas fotovoltaicos.

1.1 Delimitação do tema

A partir do conteúdo relacionado com seguidores solares, programação em HTML, microcontroladores, desenvolvimento de apps, e comunicação IP, o tema deste trabalho será delimitado no estudo e no desenvolvimento de um protótipo de um seguidor solar com comunicação IP, com a possibilidade de acesso e monitoramento remoto.

1.2 Justificativa

Com o desenvolvimento deste protótipo, muitos estudantes poderão utilizá-lo para estudos relacionados às áreas de eletrônica, programação, controle e sistemas fotovoltaicos, melhorando sua experiência na área, pois conseguirão aplicar na prática o que antes só viam em softwares e simulações. Desta forma, não é necessário dedicar um tempo para o desenvolvimento do sistema (planta), e o foco poderia ser em desenvolver métodos de controle melhorando e adicionando novas funcionalidades ao protótipo.

2 OBJETIVO

O objetivo deste trabalho é projetar e construir um protótipo didático de um seguidor solar com a utilização do Arduino e de vários módulos de sensores, que possua comunicação IP para ser acessado remotamente.

2.1 Objetivos específicos

Para que seja possível atingir o objetivo principal deste trabalho, foram considerados os seguintes objetivos específicos:

- Estudar as funções e o funcionamento de seguidores solares comerciais;
- Identificar componentes, módulos e shields comerciais de fácil acesso relacionados a plataforma Arduino, que poderiam ser utilizados para o desenvolvimento do protótipo;
- Construir o protótipo com dois eixos de rotação para o direcionamento do painel;
- Programar e implementar no Arduino uma comunicação IP com acesso aos dados via linguagem HTML;
- Realizar testes para verificar o funcionamento do protótipo.

3 REVISÃO DA LITERATURA

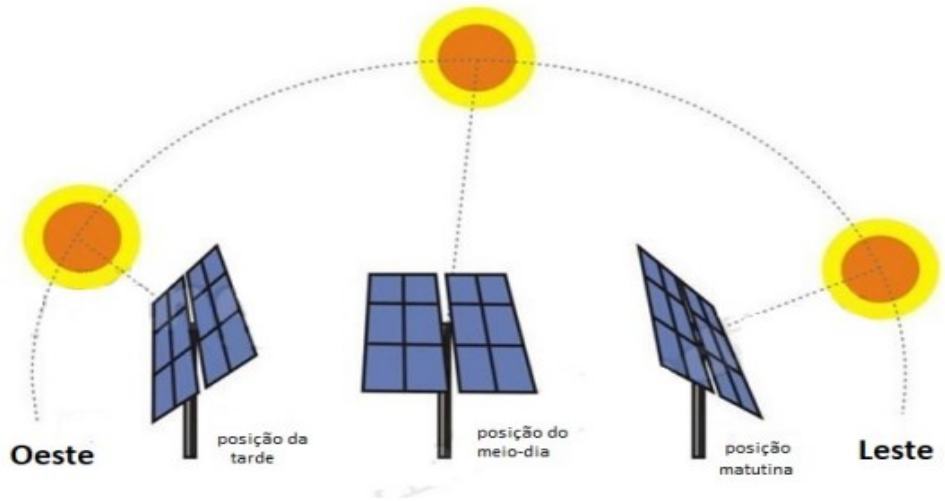
Nesta seção serão abordados os assuntos necessários para o entendimento, amadurecimento e desenvolvimento deste trabalho.

3.1 Seguidores solares

Em muitos sistemas fotovoltaicos é adotado o padrão de sistema fixo, ou seja, o painel fica na mesma posição todo o tempo. Desta forma os painéis não aproveitam 100% da incidência dos raios solares durante todo o dia, por mais que sejam calculados os melhores ângulos e posicionamento dos painéis, eles ainda irão apresentar redução na sua capacidade de geração. De acordo com Cresesb (2014), para que o sistema funcione perfeitamente e tenha a melhor eficiência é necessário realizar um estudo da região a qual o sistema será instalado, analisando a altitude, as características climáticas, o relevo, e possíveis eventos que possam de alguma maneira interferir no ângulo ideal do sistema.

Com a adição de um seguidor solar ao sistema é possível fazer com que o painel fotovoltaico mude sua posição no decorrer do dia, de acordo com a da direção do sol, conseguindo um aumento em sua eficiência e rendimento. Na Figura 1 mostra a variação da posição de um painel fotovoltaico que apresenta um rastreador solar em relação à posição do sol.

Figura 1 – Variação da posição de um painel fotovoltaico ao longo do dia



Fonte: Solar Motors (2015).

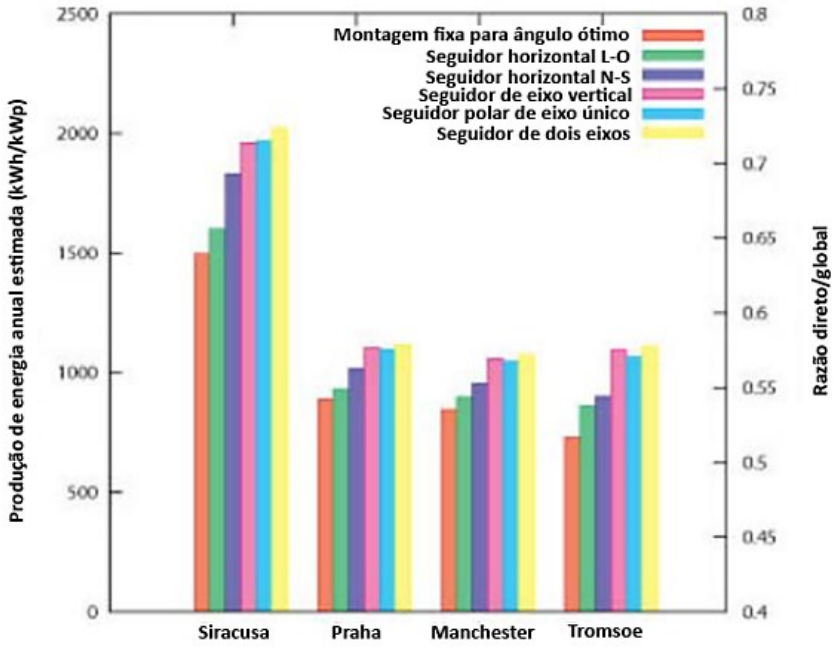
Atualmente existem vários tipos diferentes de seguidores solares no mercado, variando suas funções, seus custos e sua linha de montagem, porém para Cortez (2013) a melhor forma de classificar os seguidores solares é pela quantidade de eixos que lhe possuem, pois são os eixos responsáveis pelo seu movimento, e quanto mais flexível um rastreador é, melhor será sua precisão.

Seguidores solares podem ser separados em seguidores ativos e seguidores passivos, de acordo com Alexandru & Pozna (2008) os seguidores passivos tem sua movimentação realizada pela expansão térmica de um gás denominado Freon, a expansão desse gás causa o deslocamento do painel que é projeto para que esse movimento fique paralelo ao sol.

Os seguidores solares chamados de ativos possuem um sistema mecânico movido por motores responsáveis pela movimentação de seus eixos, prevendo seu movimento através de sensores ou de algoritmos pré-programados que fazem a análise da posição solar. Em Huld (2010) encontra-se que a principal diferente entre os seguidores ativos seria seu grau de liberdade de movimentação, como será apresentado nas próximas seções.

Huld (2010) apresenta uma comparação de produção de energia entre a utilização dos diferentes tipos de seguidores solares, ambos os seguidores foram usados junto à um sistema de 1 kWp dispostos em quatro regiões diferentes:

Figura 2 – Comparação entre o uso de diferentes seguidores solares



Fonte: Huld (2010).

3.1.1 Seguidores solares de eixo único

De acordo com Reis (2016) os seguidores de eixo único possuem sua movimentação apenas em uma angulação, tendo apenas um ponto de referência, ou se movem no eixo X, ou seja, esses seguidores só podem se mover seguindo o sol de leste a oeste (e vice-versa). Os seguidores solares de eixo único possuem duas categorias de disposição de seu eixo de movimentação, o “Eixo único horizontal”, nesta disposição o eixo de rotação é disposto horizontalmente em relação ao solo, esta disposição é mais usada quando se mostra necessário dispor vários eixos de rotação em paralelo entre si. O “Eixo único vertical” possui a disposição do eixo de rotação verticalmente em relação ao solo. Por norma eles têm o módulo inclinado com um ângulo relativo ao eixo de rotação fazendo assim uma rotação em “varredura” em cone simétrica ao seu eixo de rotação (REIS et. al. 2016).

Para Reis (2016) as principais características destes seguidores são, seguem o sol em um único movimento, aumentam seu rendimento em até 34%, tem sua construção simples, porém eficaz, necessitam de pouca manutenção, tem um custo financeiro mais baixo, dificilmente irá apresentar avarias no sistema.

A seguir encontram-se alguns exemplos de seguidores solares comerciais de eixo único fabricados e distribuídos pela empresa “SOLAR MOTORS”.

De acordo com o Datasheet do modelo ST40M2V3P disponibilizado pelo vendedor o eixo tem sua movimentação na base de 0,5º a cada movimento, seu movimento máximo consiste em 100º sendo 50º para leste e 50º para oeste. Na Figura 3 pode ser observado este modelo.

Figura 3 – Seguidor solar ST40M2V3P vista lateral e eixo de rotação

Fonte: Solar Motors (2020).

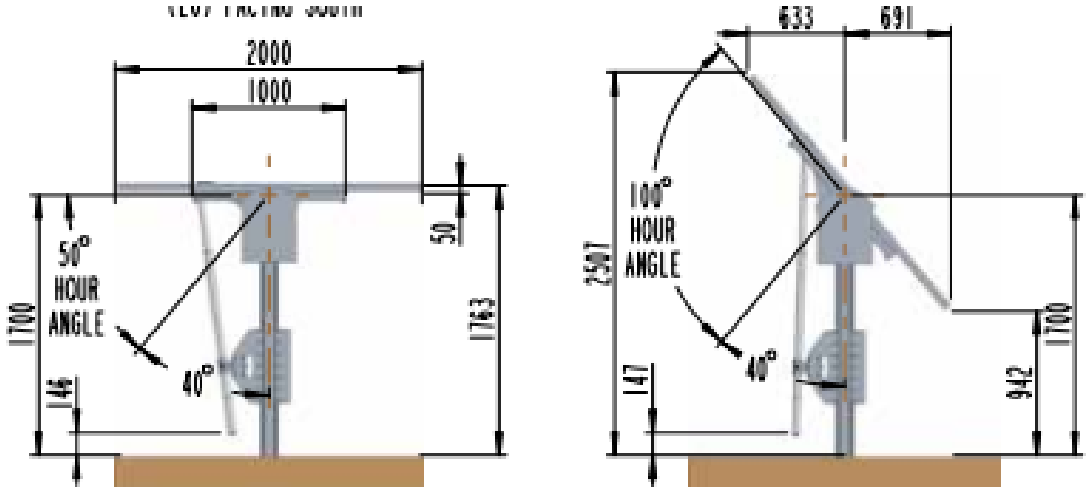
Para o movimento usa-se um motor DC de 24V com um consumo de 2,5A para funcionamento, o eixo suporta até três peças de painéis fotovoltaicos de dimensão 0,99m x 1,65m com no máximo 20kg cada, a estrutura suporta uma força de vento de até 144km/h.

No segundo exemplo é apresentado o seguidor solar de eixo único STL24. De acordo com o Datasheet disponibilizado pelo vendedor o eixo tem sua movimentação na base de 0,5º a cada movimento, seu movimento máximo consiste em 100º sendo 50º para leste e 50º para oeste, seu ângulo de elevação é horizontal, para o movimento.

Para este modelo é utilizado um motor DC de 24V com um consumo de 5A para funcionamento, o eixo aguenta até 25kg pra cada painel com dimensões de 12m x 0,5m que podem ser enfileirados em até 24 peças.

Na Figura 4 podem ser observadas as dimensões do modelo e a identificação de como ocorre a movimentação.

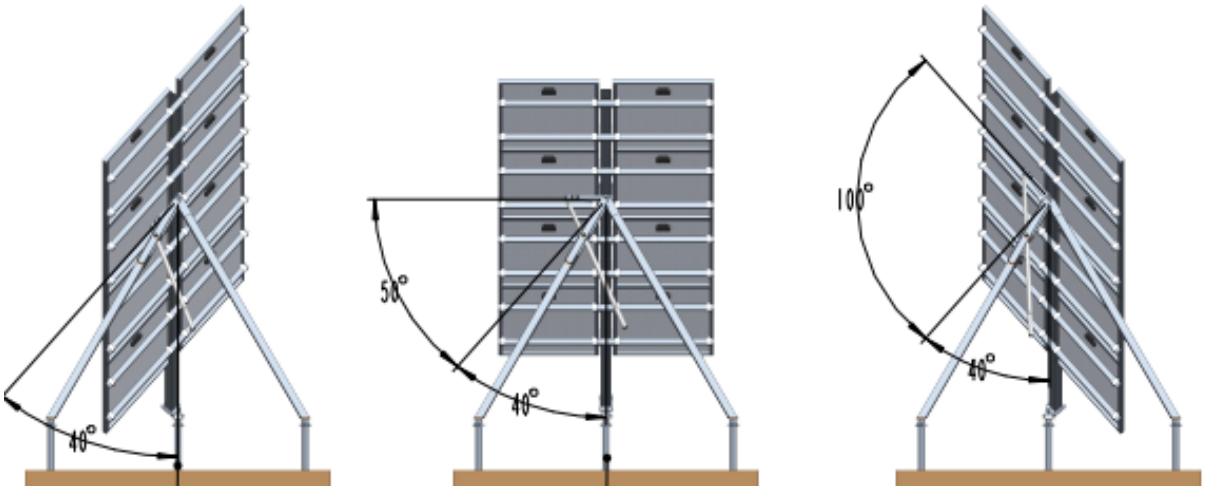
Figura 4 – Seguidor solar de eixo único STL24 movimentação



Fonte: Datasheet STL24 Solar Tracker (2020).

O terceiro exemplo, apresentado na Figura 5, é do seguidor solar de eixo único STSAT20S16. De acordo com o Datasheet disponibilizado pelo vendedor o eixo tem sua movimentação na base de 0,5º a cada movimento, seu movimento máximo consiste em 100º sendo 50º para leste e 50º para oeste. Ppara o movimento usa-se um motor DC de 25V com um consumo de 5A para funcionamento, o eixo aguenta até 25kg pra cada painel com dimensões de 1m x 2m com capacidade de controlar até oito painéis e tem uma resistência a ventos de até 144km/h.

Figura 5 – Seguidor Solar de eixo único STSAT20S16 movimentação



Fonte: Datasheet – STSAT20S16 Solar Tracker.

3.1.2 Seguidores solares de eixo duplo

De acordo com Reis (2016) seguidores solares de eixo duplo possuem um design mais complexo, apresentando sua movimentação em duas angulações devido aos seus dois eixos, podendo mudar sua posição verticalmente (norte a sul) e horizontalmente (leste a oeste). Estes seguidores são mais utilizados quando há a necessidade de maior rendimento na geração de energia, conseguindo até um aumento de 8% a mais do que seguidores de eixo único. Estes seguidores de eixo duplo usam também um “sensor olho” que segue visualmente o sol, enquanto que os seguidores de eixo único normalmente fazem o seguimento usando um padrão pré-estabelecido baseado no tempo e época do ano (REIS et. al. 2016).

Para Reis (2016) as principais características destes seguidores são: eles seguem o sol em dois movimentos, aumentam seu rendimento em até 37%; possuem uma construção complexa envolvendo motores e sensores; maior necessidade de manutenção; tem um custo financeiro mais elevado; por ser um sistema mais complexo pode apresentar mais pontos de falhas.

A seguir serão apresentados dois exemplos de seguidores solares comerciais de eixo duplo, fabricados e distribuídos pela empresa “SOLAR MOTORS”.

O primeiro modelo é o seguidor solar de eixo duplo ST54M3S30. De acordo com o Datasheet do fabricante, sua estrutura tem capacidade para até 15 peças de painéis com dimensões de 0,99m x 1,95m com peso máximo de 25kg para cada

peça, sua estrutura é desenvolvida para aguentar vendo de até 20 m/s, sua movimentação se dá através de um motor DC de 24V com um consumo de 5A de capacidade, sua angulação máxima de azimuth (movimento angular) é de 350º e sua angulação máxima de elevação é de 90º.

Na Figura 6 observa-se o sistema de movimentação deste modelo. Também é possível observar a complexidade da estrutura e o aumento na quantidade de componentes, em comparação com os modelos de eixo duplo, como o modelo apresentado na Figura 3.

Figura 6 – Seguidor Solar de eixo duplo ST54M3S30 sistema de movimentação

Fonte: Solar Motors (2020).

O segundo exemplo é do seguidor solar de eixo duplo ST44M2V4P. De acordo com o Datasheet do fabricante, sua estrutura tem capacidade para até 4 peças de painéis com dimensões de 0,99m x 1,65m com peso máximo de 20kg para cada peça, sua estrutura é desenvolvida para aguentar vendo de até 50 km/h.

Sua movimentação é realizada através de um motor DC de 24V com um consumo de 2,5A de capacidade, sua angulação máxima de azimuth (movimento angular) é de 100º e sua angulação máxima de elevação é de 15º à 90º.

Na Figura 7 podem ser observados alguns detalhes do sistema de movimentação deste modelo.

Figura 7 – Seguidor solar de eixo duplo ST44M2V4P sistema de movimentação

Fonte: Solar Motors (2020).

3.2 Microcontroladores

Microcontroladores são dispositivos eletrônicos programáveis afim de realizar algum controle de processos lógicos. De acordo com Souza & Souza (2012) o funcionamento de um microcontrolador é gerado através de um algoritmo desenvolvido pelo usuário que é gravado em sua memória, e sempre que o dispositivo for iniciado irá executar a ação contida no algoritmo.

São formados por Unidade Central de Processamento (CPU), pinos de entrada e saída, memória de dados e de programa, e periféricos como: timers, contadores, PWM, conversores analógico-digitais, comunicação serial, etc. O que diferencia um microcontrolador de um microprocessador é que o microcontrolador reúne o microprocessador e os periféricos necessários para seu funcionamento em um único chip (MIYADAIRA, 2013; NICOLOSI, 2013; SOUZA e SOUZA, 2012).

De acordo com Miyadaira (2013) microcontroladores podem ser classificados pela sua arquitetura, tendo a arquitetura Von-Neumann e a arquitetura Harvard, em Harvard existe um barramento responsável pelo acesso à memória de dados e outro barramento responsável pela memória de programa, em Von-Neumann existe apenas um barramento para ambas as memórias.

Miyadaira (2013) ainda cita que, a memória de programa é responsável pela função de armazenamento do algoritmo que irá definir a funcionamento do sistema, e essa memória é do tipo não volátil, ou seja, ao desligar o microcontrolador e

desenergiza-lo o mesmo não terá seus dados e seu algoritmo perdidos. Essa memória pode ser de quatro tipos, ROM, EPROM, OTP, e FLASH. Para Nicolosi (2013), a memória de dados é responsável por armazenar os dados de operação do sistema, porém ao se tratar de uma memória volátil, as memórias serão zeradas quando o sistema for desligado ou desenergizado diferenciando-se da memória de programa, essa memória é denominada de memória RAM.

A comunicação dos microcontroladores com elementos externos funciona através de pinos I e O, pinos esses que são denominados de pinos de entrada e saída respectivamente. Podem ser definidos como entrada, onde realizará a leitura do sinal do pino, ou como saída, onde definirá o nível de tensão do pino para controlar periféricos (MIYADAIRA, 2013).

3.3 Arduino

De acordo com Thomsen (2014), em meados de 2005 um grupo de pesquisadores conseguiram criar um dispositivo barato, funcional, de fácil programação para que até pessoas que não possuíssem um alto conhecimento sobre programação conseguissem utilizar, que possuísse também um hardware livre para que fosse usado para várias funcionais e utilidades apenas modificando e personalizando seu circuito integrado, assim deu-se a criação do Arduino.

Thomsen (2014) ainda explica que o Arduino possui vários modelos diferentes de Arduino, e cada modelo possui um microcontrolador diferente, como por exemplo, Arduino Uno com um microcontrolador de modelo ATMEL, Arduino Mega com um microcontrolador de modelo ATMEGA 2560, ou o Arduino Due com um microcontrolador ARM. Arduinos possuem circuitos responsáveis pelas entradas e saídas que são conectadas à um computador para assim fazer sua programação, programação essa que é escrita em linguagem C, após a gravação do algoritmo no seu microcontrolador basta liga-lo ao dispositivo ou sistema desejado e realizar seu funcionamento.

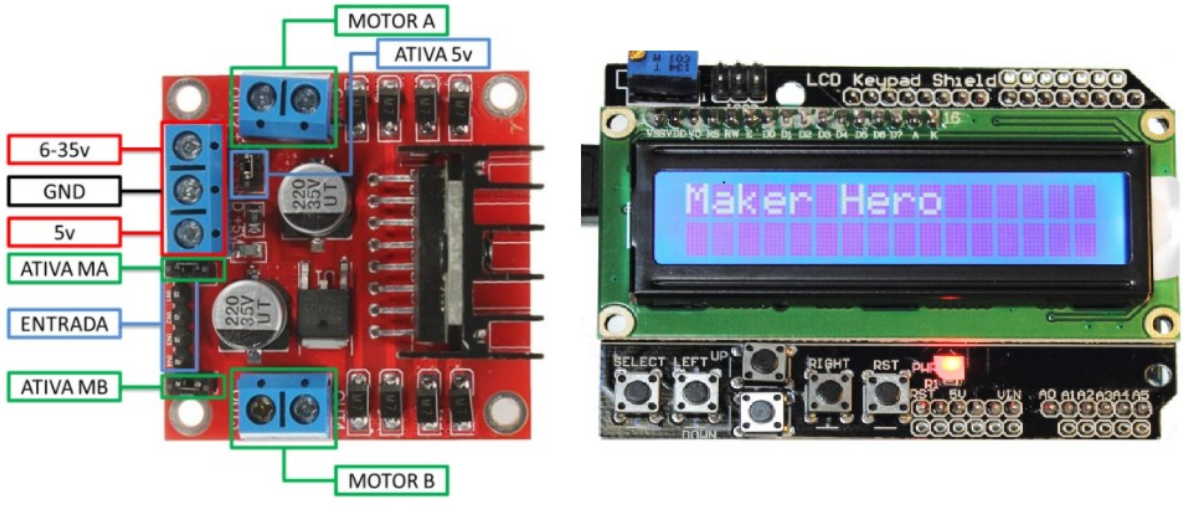
Um dos fatores determinantes para a enorme versatilidade e popularidade dos Arduinos são os chamados *Shields* e os módulos, *Shields* são placas de circuitos desenvolvidas especialmente para ter compatibilidade com o Arduino que dão a possibilidade de desempenhar enumeras funções, essas placas podem ser

desenvolvidas para ter displays, sensores, módulos de comunicação, microcontroladores, relés, entre outras coisas. Módulos são pequenas placas de circuitos que contém os sensores e outros componentes auxiliares como resistores, capacitores, leds, entre outros componentes eletrônicos.

Para melhor entendimento, usa-se como exemplo a ligação de dois motores DC ao Arduino, para o uso dos mesmos será necessária uma ponte H, logo será preciso construir um circuito com resistores, capacitores, soldas e afins, contudo com a facilidade dos *Shields* basta comprar um módulo de ponte H (Figura 8) e liga-los as portas I/O do Arduino e as conexões dos motores facilitando o processo. Outro exemplo seria a necessidade de utilização de um display LCD (Figura 8) para a criação de dado projeto, para fazer a ligação do display ao Arduino seria necessário resistores, *push-bottons*, capacitores, um PWM (*Pulse Width Modeulation*), e até a necessidade de um microcontrolador somente para o display, porem com a utilização do Display LCD *Shield*, todos esses componentes já estão soldados e conectados na placa de circuito.

Em resumo *Shields* apresentam componentes mais complexos, como microcontroladores, enquanto que módulos apresenta somente sensores e os componentes necessários para utilização do mesmo. *Shields* e módulos necessitam de bibliotecas de software específicas para funcionar corretamente, sendo necessário instalá-las no IDE do Arduino, baixando-as do website do fabricante do *Shield* (REIS et. al. 2015).

Figura 8 – Módulo ponte H L298N e Display LCD Shield



Fonte: FilipeFlop (2013), FilipeFlop (2020).

3.3.1 Módulo ESP8266

O módulo ESP8266, observado na Figura 9, é um shield produzido pela empresa Espressif System, esse módulo possui um sistema capaz de realizar uma comunicação Wifi própria, devido a essa função, este módulo é altamente utilizado como módulo Wifi em vários projetos, tanto sendo usado sozinho quanto sendo usado em conjunto com o Arduino para a criação de sistemas embarcados. Este módulo geralmente é o mais utilizado em projetos que visam ter uma conexão Wireless devido ao seu baixo custo e sua facilidade de programação e uso.

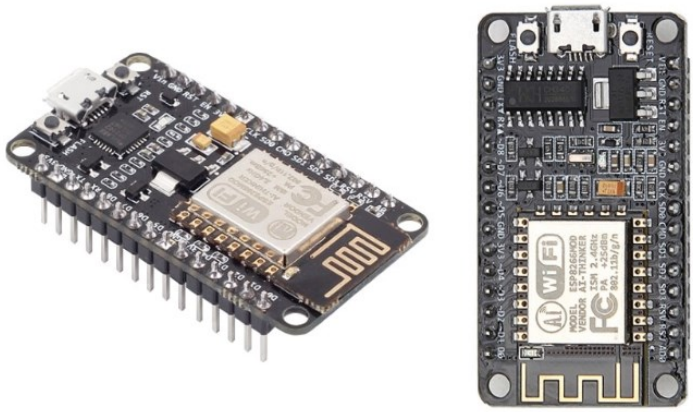
De acordo com Kolban (2016), o ESP8266 está no mercado desde 2014, e no passar do tempo até os dias de hoje foram desenvolvidos diversos tipos de modelos do módulo, os modelos variam do ESP-01 até o ESP-12, todos os modelos possuem o mesmo processador mudando apenas o número de pinos de entrada e saída (GPIO) disponíveis, memória disponível e o espaçamento entre os pinos. A seguir encontra-se uma tabela retirada de Kolban (2016) que mostra as especificações técnicas do ESP8266.

Tabela 1 – Especificações do ESP8266

Especificação	Característica
Tensão	3,3V
Consumo de corrente	10μA
Memória Flash	16MB max (512k normal)
Processador	Tensilica L106 32 bit
Velocidade do processador	80-160MHz
RAM	32K + 80K
GPIOs	17 (multiplexada com outras funções)
Analógico para digital	1 entrada com 1024 de resolução
Suporte 802.11	b/g/n/d/e/i/k/r
Máxima corrente de conexão TCP	5

Fonte: Kolban (2016).

Figura 9 – Módulo ESP8266

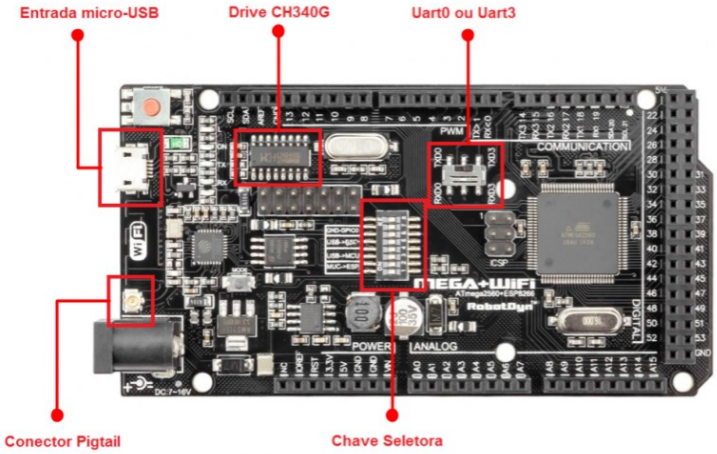


Fonte: FilipeFlop (2016).

3.3.2 Arduino Mega wifi

O Arduino Mega Wifi é uma versão modificada do Arduino Mega com módulo ESP8266 integrado em sua placa, possibilitando e facilitando a conexão entre ambos. Na sua placa possui uma entrada micro USB para transferência de dados, com um drive USB-Serial CH340G, uma chave seletora para escolher qual se deseja usar, ou seja, usar somente o ESP8266, ou somente o Arduino, ou usar ambos juntos, possui também uma chave seletora de UART (Serial) do Arduino Mega com o ESP8266 facilitando sua conexão, e por fim um conector Pigtail para conectar uma antena Wifi para melhorar a área de conectividade Wifi.

Figura 10 – Arduino Mega WIfi hardware



Fonte: Albuquerque (2020).

De acordo com seu Datasheet, o Arduino Mega Wifi possui uma memória flash conectada ao ESP8266 de 4 MB, sendo superior ao Arduino Mega normal, uma tensão de entrada de 7 a 16 V e uma corrente de saída de 1,6 A para 5 V e 1 A para 3,3 V, enquanto que o Arduino Mega normal só pode fornecer 500 mA para 5 V e 50 mA para 3,3 V.

3.3.3 Módulo de sensor LDR

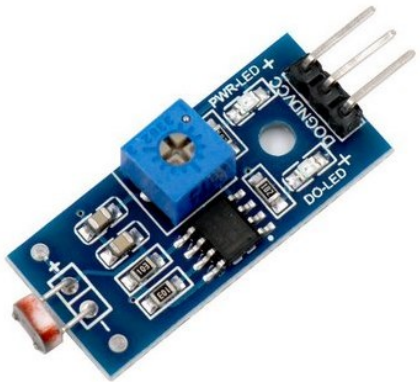
LDR é a sigla em inglês para *Light Dependent Resistor*, que traduzido significa resistor dependente de luz. O LDR também é conhecido como foto resistor, e ele é um tipo de resistor que tem a capacidade de variar a sua resistência em função da intensidade de luz que incide sobre ele (ALVES et. al. 2019).

Existem dois tipos de sensores LDR, os sensores usados para captar a luz visível e os sensores usados para captar a luz infravermelha, trata-se de um elemento passivo, ou seja, não possui polaridade definida. Alves (2019) explica que o funcionamento do LDR se dá pela incidência dos fótons na superfície do sensor fazendo a liberação de seus elétrons fazendo com que a condutividade aumente e a resistência diminua.

A variação do funcionamento de um LDR ocorre devido a incidência de luminosidade do ambiente, ou seja, ao apresentar uma incidência maior de luz menor será a resistência do LDR, por isso é recomendado para seguidores solares.

O módulo de sensor LDR pode ser observado na figura 11, possui três terminais: VCC; GND; e DATA, que é o sinal que será enviado ao Arduino. Também pode ser observado o trimpot para ajuste de ganho.

Figura 11 – Módulo de sensor LDR



Fonte: Datasheet Photosensitive Sensor Module (2006).

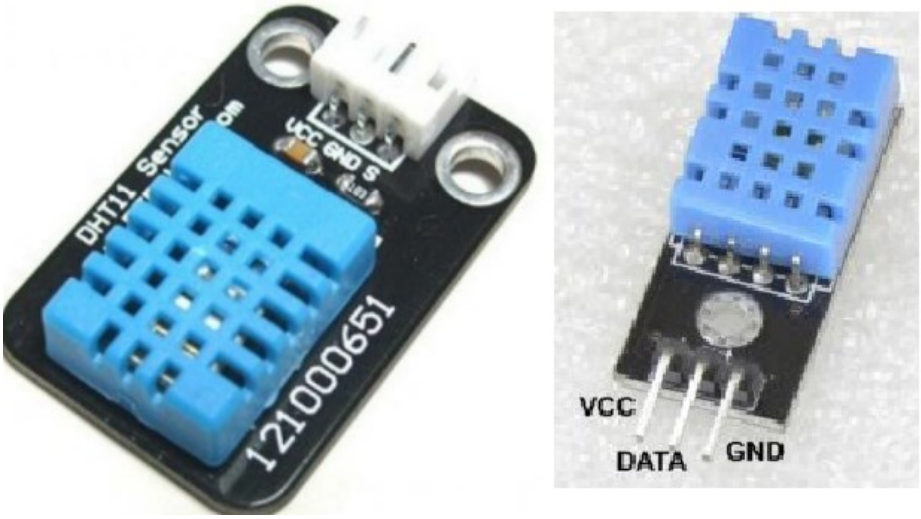
De acordo com seu Datasheet, ao fazer uso de sua biblioteca de funcionalidade, o módulo apresenta uma função denominada ADC (*Analog to Digital*) que tem a capacidade de realizar a conversão do sinal analógico captado pelo sensor em sinal digital mostrado no monitor serial do ambiente de programação do Arduino, facilitando muito processo de análise do parâmetro de medição.

3.3.4 Módulo de sensor de temperatura e umidade

O módulo de sensor de temperatura e umidade DHT11 (Figura 12) tem a funcionalidade de medir a temperatura e umidade do ambiente em que está presente. Em seu *datasheet* encontra-se que seu funcionamento se dá pelo seu termistor NTC, um resistor sensível a variações de temperatura, no módulo do sensor encontra-se um microcontrolador responsável por fazer as medições e transmitir os valores em formato digital para os pinos de saída.

Em seu *datasheet* ainda se encontra suas especificações, sua faixa de umidade relativa varia entre 20% à 80% com uma precisão de ±5%RH, a temperatura que o sensor é capaz de medir está na faixa de 0 ºC à 50 ºC com uma precisão de ±2%RH, a precisão acontece a cada dois segundos, sua tensão de alimentação é de 5 V com um consumo de corrente de 2,5 mA.

Figura 12 – Módulo de Sensor DHT11 e sua pinagem



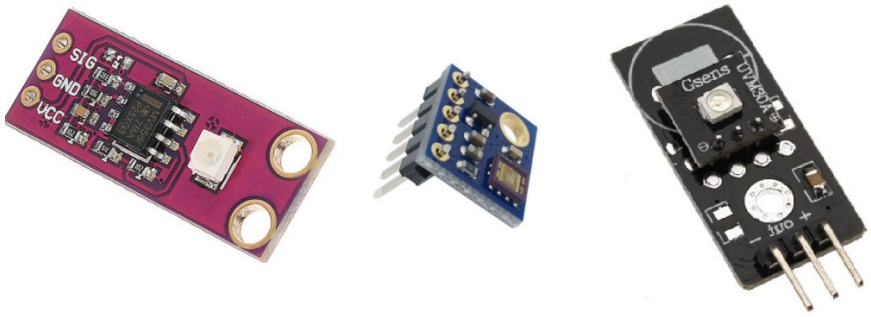
Fonte: D-Robotics (2010).

3.3.5 Módulo de sensor de radiação ultravioleta

De acordo com Pereira (2018), o módulo de sensor ultravioleta é uma placa de circuito para o Arduino que tem a capacidade de medição de raios ultravioletas, a maioria dos módulos de sensores UV disponibilizados para utilização em conjunto com Arduino somente funcionam na faixa luminosa de 240 nm à 370 nm, sua medição é dada através de um sinal analógico possuindo três saídas para pinagem, uma responsável pela transferência de dados, uma para alimentação e uma para o GND.

Os três exemplos de módulos que podem ser observados na Figura 13 e que são usados em conjunto com Arduino para medição de raios UV são: o módulo de Sensor de Raio Ultravioleta UV Guva-S12SD, de acordo com o seu *datasheet* possui uma tensão de operação de 5V, uma faixa de trabalho de temperatura de -30 ºC à 85 ºC e tem a capacidade de medição de ondas UV na faixa de 240-370 nm; o módulo de Sensor de Luz Ultravioleta UV ML8511, possui tensão de operação de 4,6 V, a faixa de trabalho de temperatura é de -20 ºC à 70 ºC e tem a capacidade de medição de ondas UV de 230-365nm; e o módulo Sensor de Raio Ultravioleta UV UVM-30 A, possui tensão de operação de 5 V, uma faixa de trabalho de temperatura de -20 ºC à 85 ºC e capacidade de medição de ondas UV na faixa de 200-370 nm.

Figura 13 – Módulos de Sensores UV S12SD, UV ML8511 e UV UVM-30A



Fonte: Thomsen (2015).

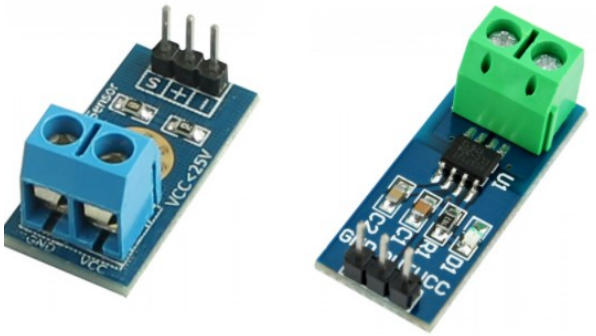
3.3.6 Sensor de tensão e sensor de corrente

Um dos objetivos deste rastreador solar será a obtenção da tensão e da corrente que a mini célula fotovoltaica será capaz de gerar. Para conseguir medir esse parâmetro é necessário o uso do módulo de Sensor de Tensão DC para

Arduino. De acordo com o seu *datasheet* tem a capacidade de medir tensões DC em uma faixa de 0 V à 25 V, possui dois resistores soldados em sua placa de circuito de valores 30 KΩ e 7,5 KΩ com tolerâncias de 1 %.

Para a medição da corrente elétrica gerada pelo mini painel fotovoltaico é necessário o uso do módulo de Sensor de Corrente ACS712. O *datasheet* informa que este sensor tem a capacidade de medir correntes AC e DC de até no máximo 30 A, possui uma tensão de alimentação de apenas 5 V devido a sua ligação com Arduino e possui um tempo de resposta de 5 us. Os módulos de tensão e corrente podem ser observados na Figura 14.

Figura 14 – Módulo Sensor de Tensão DC e módulo de Sensor de Corrente ACS712



Fonte: UsinaInfo (2021).

3.3.7 Servos motores

De acordo com Eletrograte (2017) um servo motor trata-se de um motor eletromecânico que em sua composição tem acoplado um encoder e um controlador, por causa da presença deste controlador, os servos motores se diferenciam de motores normais pois podem ser programados para atualizar sua posição inicial até dada posição especifica desejada. Um servo basicamente funciona como um atuador rotativo para controle de posição, que atua com precisão e velocidade controlada em malha fechada (ELETROGATE et. al. 2017).

Os modelos mais utilizados em projetos envolvendo Arduinos são o Micro servo 9G SG90 e o Micro servo Motor MG995, apresentados na Figura 15. Ambos possuem a mesma capacidade de atuação diferenciando-se apenas na tensão de operação e na força de torque. O SG90 trata-se de um modelo mais simples, por isso possui um torque de 1,5 kg.cm e uma tensão de operação de 6 V, já o MG995 trata-se de um modelo mais robusto possuindo um torque de 9,4 kg.cm e um tensão de operação de 7,2 V (ROBOCORE et.al 2021; FILIPEFLOP et. al 2021).

Figura 15 – Servo motores SG90 e MG995 respectivamente

Fonte: Robocore (2021), Eletrogate (2017).

3.3.8 Mini painel fotovoltaico

O protótipo de seguidor solar desenvolvido neste trabalho será para uso didático e será construído com uma escala bem reduzida em comparação com os modelos comerciais. Desta forma, será utilizado um mini painel fotovoltaico.

O mini painel fotovoltaico observado na Figura 16, tem a capacidade de gerar energia elétrica através da conversão dos raios solares. De acordo com a UsinaInfo (2020), este mini painel consegue alcançar uma potência máxima de 1 W, e dependendo da variação da incidência dos raios solares, consegue alcançar uma corrente máxima de 180 mA e a tensão de 6 V, suas dimensões são 112x84x3 mm e tem um peso de 35 gramas.

Figura 16 – Mini painel fotovoltaico vista frontal e vista traseira

Fonte: UsinaInfo (2020).

3.3.9 Módulo display LCD

Os módulos Liquid Crystal Display (LCD) são dispositivos que possuem uma interface gráfica elétrica padronizada e recursos internos de softwares que permitem enviar textos, números e símbolos para sua tela criando assim uma interface visual para os usuários, em projetos que envolvem microcontroladores os displays mais usados são os de 16x2, 16 colunas e 2 linhas. De acordo com Murta (2022), o chip controlador de LCD mais utilizado atualmente é o HD44780 desenvolvido pela empresa Hitachi, sua comunicação com o Arduino pode ser paralela ou via Serial. Em muitos projetos será visto a comunicação via Serial pois a mesma oferece a facilidade de economizar portas digitais e a facilidade de implementar funções prontas através da biblioteca ***"I2C"***.

A Tabela 2 foi retirada do Datasheet do display 16x2 e mostra sua pinagem e suas funções.

Tabela 2 – Pinagem display LCD 16x2

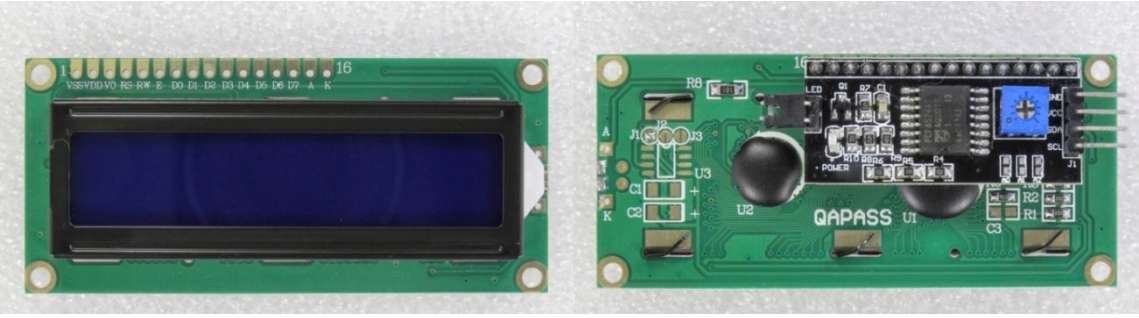
Pino	Nomenclatura	Função
Pino 1	VSS	Pino GND
Pino 2	VDD	Pino de alimentação 5V
Pino 3	VO	Pino de ajuste de contraste do LCD
Pino 4	RS	Seleção de comandos
Pino 5	R/W	Read/Write
Pino 6	E	Enable
Pino 7	D0	Data bit 0
Pino 8	D1	Data bit 1
Pino 9	D2	Data bit 2
Pino 10	D3	Data bit 3
Pino 11	D4	Data bit 4
Pino 12	D5	Data bit 5
Pino 13	D6	Data bit 6
Pino 14	D7	Data bit 7
Pino 15	A	Anodo do LED de iluminação
Pino 16	K	Catodo do LED de iluminação

Fonte: Datasheet RT162-7 (2020).

De acordo com Murta (2022), a interface I2C foi desenvolvida para resolver o problema de limitação de portas nos microcontroladores usados em conjunto com o display, em especial o Arduino, nesses displays que possuem a interface I2C possuem um chip PCF8574, o mesmo é um expansor de portas paralelas podendo controlar até 8 bits tanto como entrada ou como saída, a velocidade da interface é de 100 KHz e tem sua alimentação de 5V.

Para a ligação do display com Arduino só é necessário o uso de quatro pinos, GND, Vcc, SDA (serial data) e o SCL (serial clock).

Figura 17 – Módulo display LCD 16x2 e display LCD 16x2 com interface I2C



Fonte: Murta (2022).

3.4 Linguagens de programação

Nesta seção serão apresentadas as linguagens de programação necessárias para o desenvolvimento do projeto, tanto para o *back-end* (linguagem responsável pelo funcionamento interno de um software, tudo aquilo que está por trás da interface da aplicação), quanto para o *front-end* (linguagem responsável pela conexão do usuário ao software através de uma interface gráfica, tudo aquilo que está a frente do software) do sistema.

3.4.1 Linguagem C

De acordo com Evans & Noble (2013) a linguagem usada para criação de projetos e algoritmos no Arduino é a linguagem C, é usada essa linguagem devido ao seu método de *Processing*, método esse que tem o intuito de desenvolver fundamentos de programação dentro de um contexto visual.

Graças à alta popularidade que o Arduino alcançou, diversos entusiastas e programadores desenvolveram uma ampla gama de bibliotecas que possuem variadas funções e recursos para serem utilizados em algoritmos, por isso o Arduino tornou-se o microcontrolador perfeito para pequenos projetos. A linguagem C segue o padrão de que os elementos do código e as bibliotecas tornam-se a estrutura do algoritmo, as variáveis são os conjuntos de dados coletados e as funções desenvolvem o controle do sistema.

3.4.2 Linguagem HTML

Ramalho (1996) mostra que a linguagem HTML, ou *HyperText Markup Language*, é uma linguagem usada para desenvolver layouts de páginas da Web. A simplicidade dessa linguagem é baseada na finalidade de formatar o texto mostrado e criar relações entre outras homepages, com isso, são criados documentos com o conceito de hipertextos (FREITAS et. al. 2014).

Freitas (2014) afirma que quando um navegador faz o acesso a uma página na web, a linguagem HTML faz a análise do texto através de tags. As tags podem ser denominadas como caracteres especiais que tem a função de armazenar a informação do algoritmo e mostrar o texto na tela para que o usuário possa acessar. Por isso a linguagem HTML é muito usada para algoritmos *Front-end*, pois ela tem a função de fazer a conexão que o usuário terá com o sistema ou o programa utilizando um layout.

A linguagem HTML será usada para este projeto para desenvolver o layout do app de controle do seguidor solar, e fazer com que o mesmo possa ser acessado em qualquer dispositivo que tenha acesso a conexão IP, o layout terá as funções de mostrar e escolher qual parâmetro o seguidor estará analisando no momento.

4 DESENVOLVIMENTO

Nesta seção serão apresentados todos os procedimentos utilizados para realizar o desenvolvimento do protótipo.

4.1 Definição de modos de uso do protótipo

O protótipo foi desenvolvimento para que possa ser utilizado de duas maneiras diferentes: 1) em uso de mesa, o protótipo deverá estar conectado a um computador, tendo sua alimentação e sua transmissão de dados via USB, os dados medidos pelos sensores poderão ser informados através do monitor serial do IDE "***Arduino"***, software desenvolvimento pelos próprios criadores do Arduino para realizar o desenvolvimento de algoritmos e transferência de dados entre o computador e a placa Arduino; 2) em uso portátil, o protótipo deverá estar conectado a uma fonte de alimentação externa e os dados medidos pelos sensores serão informados através de um painel LCD presente na estrutura do protótipo.

Em ambos os usos o protótipo terá duas opções de modo de uso, o modo manual, o qual o usuário deverá realizar a movimentação dos servos através de um joystick e poderá escolher qual a melhor angulação para os sensores. E o modo automático, o qual será controlado através de seis sensores LDR dispostos em lugares estratégicos na estrutura do protótipo, os sensores ficaram constantemente captando a luz e calculando qual a melhor angulação para que o painel solar tenha a melhor captação possível.

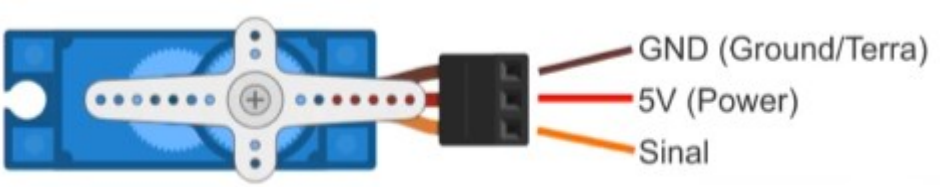
Os modos manual e automático se diferenciaram apenas pelo método de controle dos servos motores, os demais sensores funcionaram da mesma maneira para ambos os modos.

4.2 Controle dos servo-motores para modo manual

Para realizar o movimento de angulação do protótipo foram usados dois servo-motores SG90 (Figura 15), um responsável pela movimentação no eixo X e outro responsável pela movimentação no eixo Y. A movimentação do protótipo será

necessária para que o mesmo consiga captar o máximo de medições possíveis pelos sensores, logo os dois servos simulam a movimentação de um painel solar de eixo duplo. Os servos SG90 possuem uma conexão simples, tendo três saídas, uma para o Vcc, uma para o GND, e uma para a conexão com uma porta digital do Arduino (Figura 18).

Figura 18 – Conexões para o funcionando dos servos



Fonte: FilipeFlop (2013).

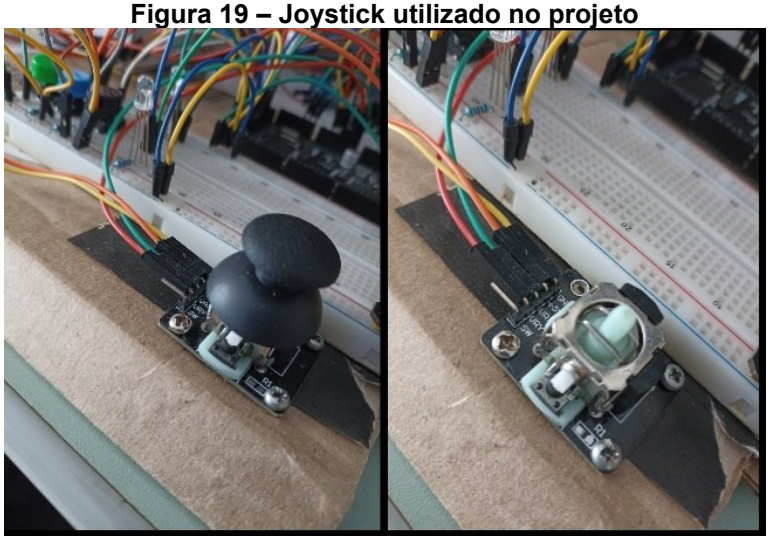
Existem várias bibliotecas prontas disponíveis para controle de diversos servos-motores conectados ao Arduino disponibilizadas no próprio site dos desenvolvedores do Arduino, como mencionado no tópico 3.3, para realizar o desenvolvimento do código responsável pelo controle dos servos foi utilizado a biblioteca de programação ***"Servo"***, tal biblioteca possui todas as funções que serão necessárias para o controle dos servos.

Para o controle manual foi utilizado um joystick, que pode ser observado na Figura 19. Este joystick possui dois potenciômetros ligados a uma alimentação de 5 V e duas saídas analógicas ligadas ao Arduino, uma para cada potenciômetro. O sinal analógico dos potenciômetros varia entre 0 à 1100, em repouso o joystick tem um sinal no valor de 550, empurrando o joystick para um lado este valor irá aumentar até 1100, consequentemente empurrando para o lado oposto o joystick irá diminuir até o valor 0.

Para realizar este controle foi criada duas variáveis que irão receber os sinais analógicos do joystick, e mais duas variáveis que irão converter esses sinais analógicos em sinais digitais para passar a informação para os servos. As variáveis digitais irão começar com um valor de 55, consequentemente os servos irão iniciar na posição de 55º, sendo assim caso o valor de leitura do potenciômetro seja menor que 550 a variável de conversão digital irá diminui seu valor de 1 em 1 até alcançar o valor 0 (valores negativos não serão lidos pois a angulação do servo para em 0º), em contra partida caso a leitura analógica do potenciômetro seja maior que 550 a variável de conversão digital irá aumentar o seu valor de 1 em 1 até alcançar o valor

de 180 (valor de angulação máxima alcançado pelo servo) e no momento que a variável digital tem seu valor alterado através da função ***"write"***, disponibilizada pela biblioteca ***"Servo"***, esse valor será convertido para a angulação do servo realizando seu movimento (código disponibilizados nos apêndices A, B e C).

Figura 19 – Joystick utilizado no projeto

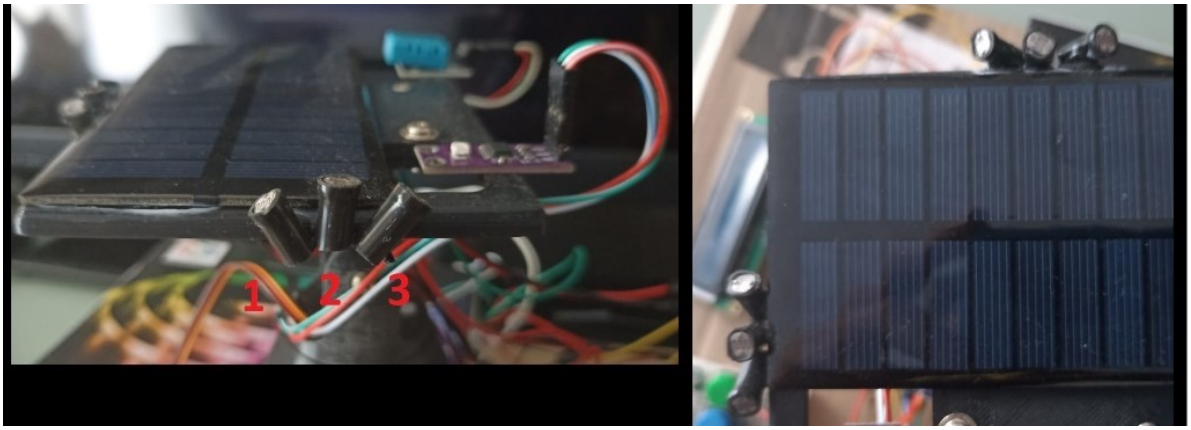


Fonte: Autoria própria.

4.3 Controle dos servo-motores para modo automático

De mesmo modo que o desenvolvimento do modo manual, para realizar o desenvolvimento do algoritmo de controle dos servos foi usado a biblioteca ***"Servo"***, diferenciando apenas que o controle do movimento de angulação dos servos é feito por seis sensores LDR, os sensores estão dispostos em dois grupos contendo três sensores arranjados da maneira apresentada na Figura 20, para o uso dos sensores LDR não é necessário usar alguma biblioteca especifica, sendo somente necessário declarar uma variável analógica para receber o sinal de leitura de cada sensor separadamente.

Figura 20 – Disposição dos sensores LDR

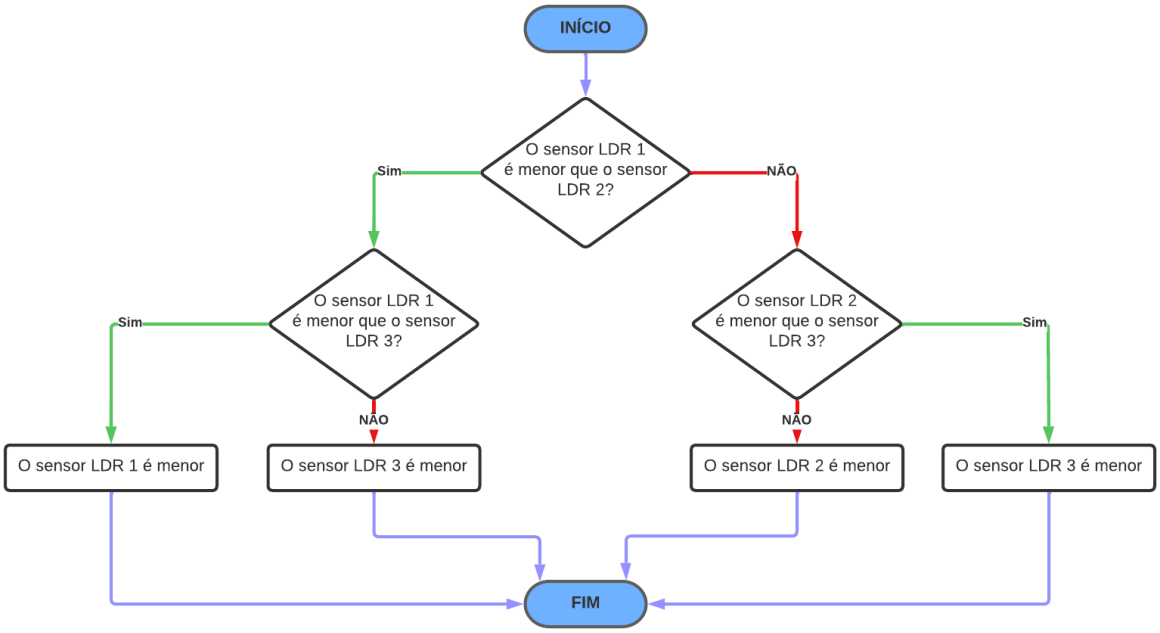


Fonte: Autoria própria.

O sinal do sensor LDR varia conforme sua captação de luz, ou seja, qual sensor apresentar a maior captação de luz consequentemente será o sensor que irá ditar a movimentação da angulação do servo. De maneira simples, pela figura 20, o protótipo sempre irá buscar com que o sensor do meio seja o que apresenta o maior índice de luminosidade, pois assim o painel solar irá obter uma incidência em toda a sua área.

Os sensores LDR são divididos em dois grupos, um para controle da movimentação no eixo X e um para o controle da movimentação no eixo Y, os dados obtidos pelas leituras analógicas dos LDRs passa por uma função simples a qual realiza o cálculo de comparação para distinguir qual sensor terá o menor valor pois quanto mais próximo do valor 0 maior será a incidente luminosa que o sensor irá captar, sendo assim qual sensor tiver seu menor valor será armazenado na variável **“menor”.** Agora para a movimentação dos eixos será aplicado a seguinte regra, olhando a Figura 20 observa-se a disposição dos sensores, caso o sensor 1 esteja captando a maior incidência luminosa o servo irá deslocar seu eixo até o ponto em que o sensor 2 estiver com a maior incidência, pois assim o painel estará na melhor posição para captar os raios solares. Em contrapartida, caso o sensor 3 esteja com a maior incidência o servo irá se movimentar da mesma maneira (Código disponibilizados nos apêndices A, B e C).

Figura 21 – Fluxograma para interpretar o funcionamento dos sensores LDR.



Fonte: Autoria própria.

4.4 Controle dos sensores

Na sequência serão apresentados os procedimentos utilizados para a configuração dos sensores de tensão, corrente, temperatura e umidade, e medidor ultravioleta . Todo o código de programação desenvolvido está disponibilizado nos apêndices A, B e C.

4.4.1 Sensor medidor de tensão

O sensor medidor de tensão foi usado para realizar a medição da tensão contínua produzida pelo mini painel solar. Para o uso deste sensor não será necessário o uso de uma biblioteca especifica, pois o mesmo tem seu funcionamento baseado em divisores de tensão.

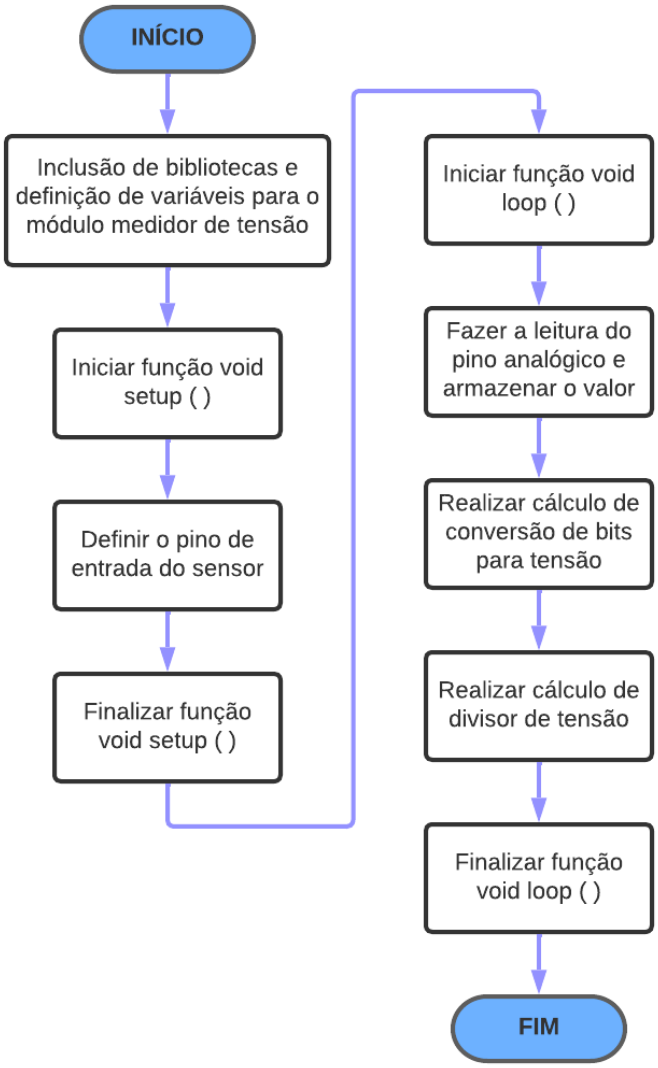
De acordo com o seu Datasheet, o sensor tem a capacidade de ser conectado a um valor de tensão DC de entrada de até cinco vezes maior que o VCC de sua porta analógica, como está sendo usando um Arduino mega, pode ser conectado até 25 V. A resolução do conversor analógico do Arduino é de 10 bits, sendo assim a resolução do sensor de tensão será de 0,00489 V, assim sendo o sensor só irá captar valores de tensão acima de 0,02445 V.

A conexão do sensor é simples, possui duas entradas, uma para o positivo e outra para o negativo da carga e três saídas, uma para o VCC, uma para o GND e uma saída analógica para a transferência de dados com o Arduino, como pode ser visto na Figura 12.

O desenvolvimento do algoritmo responsável por controlar o sensor consiste em, primeiramente declarar uma variável analógica que irá armazenar a leitura do sensor, depois definir duas variáveis, uma para armazenar a tensão final após o divisor de tensão e uma para armazenar a conversão da leitura do sensor em tensão de entrada, criar duas variáveis referentes aos valores dos resistores para o cálculo de divisor de tensão.

Nessa parte é necessário consultar o Datasheet para conferir qual o valor dos resistores soldados no módulo, que no caso deste módulo sãio usados um resistor de 20 kΩ e outro de 7,5 kΩ. Por fim, ao realizar a leitura do sensor, é feito a conversão dos bits em valores de tensão, e depois aplicado o divisor de tensão recebendo assim o valor de tensão de saída do sensor

Figura 22 – Fluxograma para interpretar o funcionamento do sensor medidor de tensão



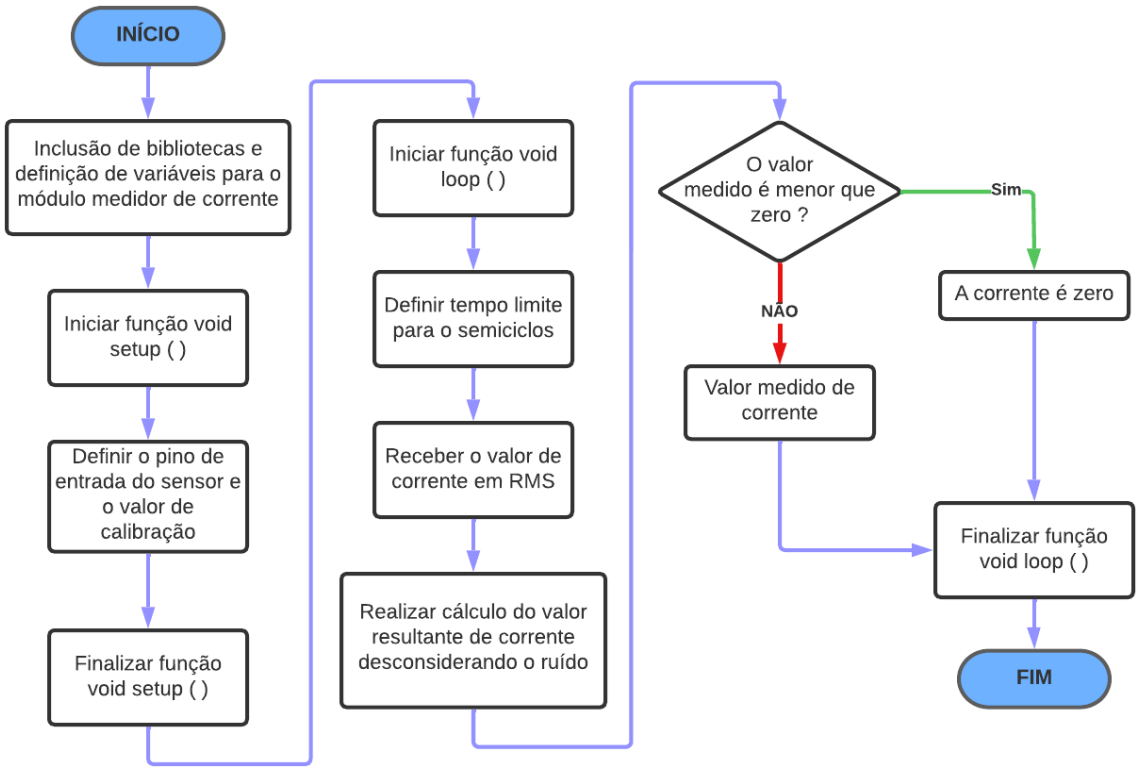
Fonte: Autoria própria.

4.4.2 Sensor medidor de corrente

O sensor medidor de corrente foi usado para realizar a medição da corrente contínua produzida pelo mini painel solar. De acordo com seu Datasheet, este sensor utiliza o efeito hall para fazer a detecção do campo magnético que é gerado pela passagem da corrente. O sensor usado para o projeto foi o ACS712 (figura 12), o sensor possui duas entradas para serem conectadas entre as fases da carga, e três saídas, uma para o VCC, uma para o GND e uma analógica para a conexão de dados com o Arduino. Para o uso deste sensor será necessário aplicar a biblioteca ***"EmonLib"*** e suas funções no desenvolvimento do algoritmo.

O algoritmo desenvolvido consiste em definir uma variável analógica para receber a leitura do sensor, definir uma variável denominada ***"CURRENT_CAL"***, está variável faz parte das funções da biblioteca e tem como função fazer a compensação no cálculo de forma que o sensor fique calibrado e consiga efetuar as medições o mais próximas do valor real. Para encontrar o valor desta variável foi necessário realizar a medida da carga com o auxílio de um Amperímetro, o valor encontrado foi de 2,2 mA, depois é necessário usar a função ***"calcVI"*** responsável por delimitar os ciclos e os tempo de leitura dos ciclos antes de informar o valor final, para realizar a conversão do valor de corrente RMS obtido para um valor de corrente padrão é necessário usar a função ***"Irms"***, conseguindo o valor de corrente mais aproximado medido pelo sensor (Código disponibilizados nos apêndices A, B e C).

Figura 23 – Fluxograma para interpretar o funcionamento do sensor medidor de corrente



Fonte: Autoria própria.

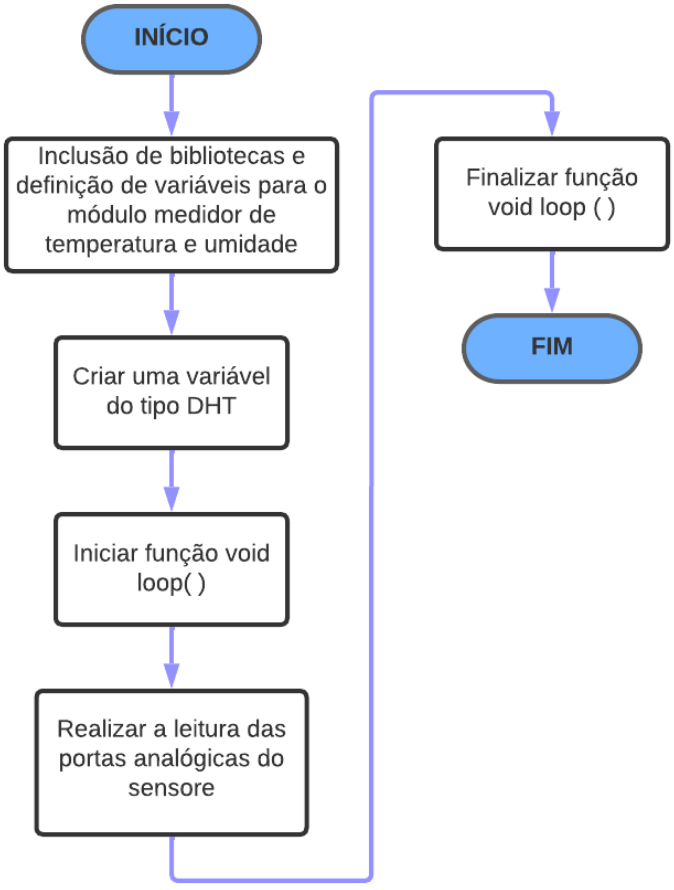
4.4.3 Sensor de temperatura e umidade

O módulo de sensor de temperatura e umidade escolhido para ser usado neste projeto foi o módulo DHT11 (figura 12), de acordo com seu Datasheet, este

sensor faz medições de temperatura a uma faixa de 0º até 50º celsius e mede a umidade do ambiente na faixa de 20 % a 90 %, tendo uma margem de erro de 2 ºC para a temperatura e 5 % para a umidade. A ligação do módulo segue o padrão de todos os módulos já apresentados anteriormente, tendo três saídas, uma para o VCC, uma para o GND, uma saída analógica para a com Arduino.

Para o desenvolvimento do algoritmo responsável pelo controle do sensor é necessário o uso da biblioteca ***"dht.h"***, primeira é declarada uma variável analógica responsável por receber os valores obtidos pela leitura do sensor, depois é necessário informar qual o tipo da variável será usado para realizar a conversão dos valores de leitura para valores de informação. E por fim para informar os valores obtidos pelo sensor basta usar as funções ***"DHT.humidity"*** para os valores de umidade e ***"DHT.temperature"*** para os valores de temperatura fornecidas pela biblioteca (Código disponibilizados nos apêndices A, B e C).

Figura 24 – Fluxograma para interpretar o funcionamento do sensor medidor de temperatura e umidade.

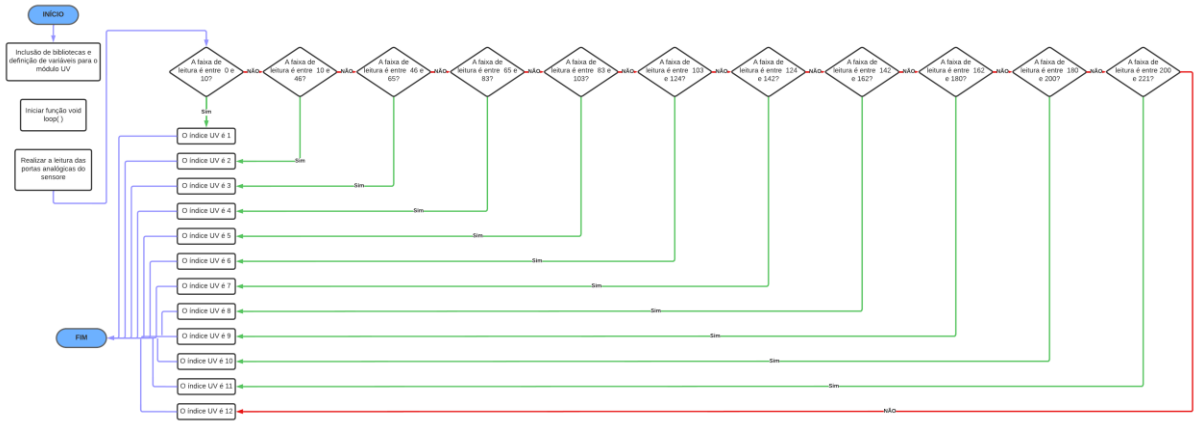


Fonte: Autoria própria.

4.4.4 Sensor de medição ultravioleta

O módulo sensor de medição ultravioleta escolhido para ser usado neste projeto será o sensor UV S12SD (Figura 13). O módulo possui três saídas, uma para o VCC, uma para o GND, e uma saída analógica para a conexão de dados com o Arduino. Para o desenvolvimento do algoritmo responsável pelo controle do sensor não será necessário o uso de nenhuma biblioteca específica, primeiramente declara-se uma variável analógica para receber o valor do sinal obtido pela leitura do sensor, o valor de leitura do sinal varia de 0 a 256, e de acordo com o Datasheet do sensor, cada período de leitura é equivalente à um valor de índice UV, por exemplo, o valor entre 10 e 46, é equivalente ao índice 1, o valor de 46 à 65 é equivalente ao índice 2, e assim sucessivamente. Logo só é necessário criar um laço IF ELSE para encaixar todos os intervalos equivalentes aos seus respectivos índices (Código disponibilizados nos apêndices A, B e C).

Figura 25 – Fluxograma para interpretar o funcionamento do sensor medidor de UV



Fonte: Autoria própria.

4.5 Comunicação wireless

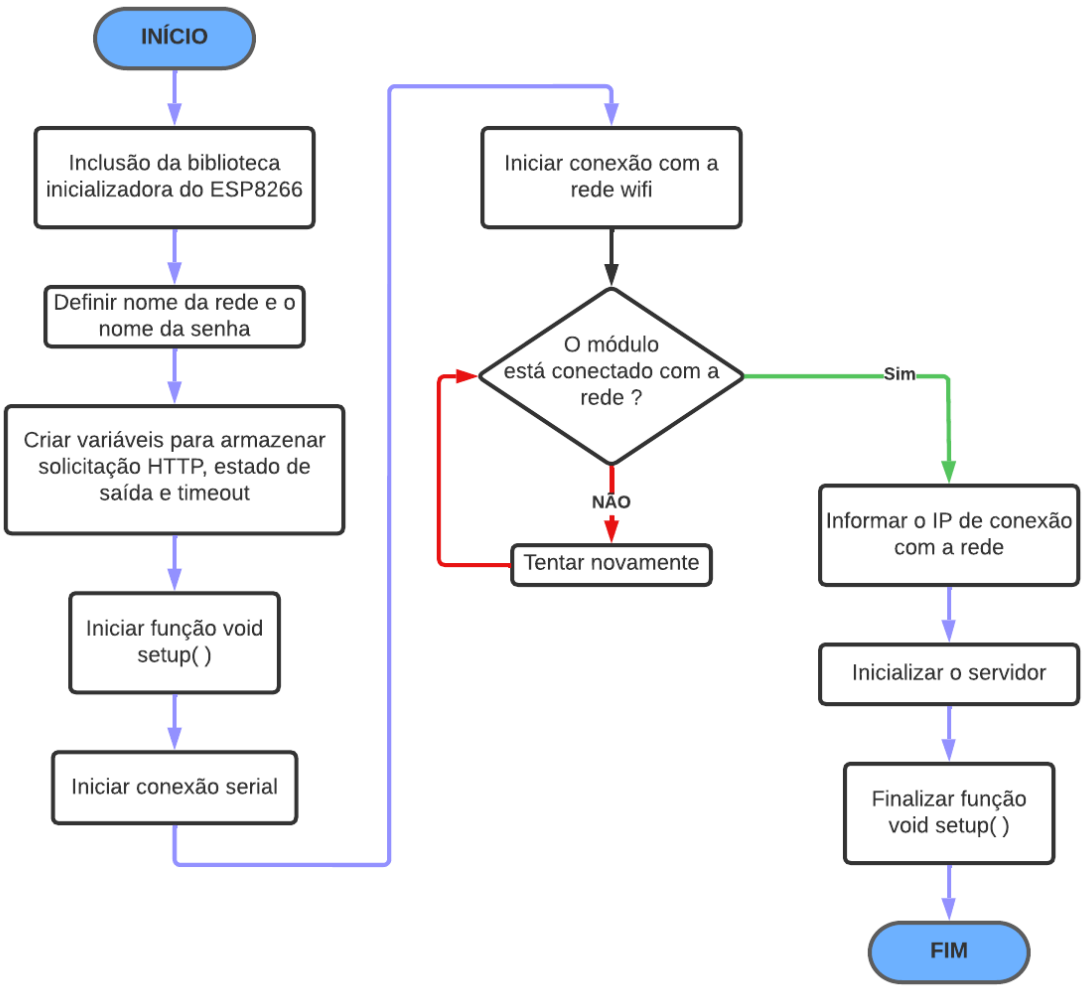
Para realizar a comunicação foi necessário desenvolver dois algoritmos separadamente, porém ambos devem se comunicar via conexão serial, um algoritmo para controle do módulo ESP8266 e para envio de dados, e um algoritmo para controle do Arduino Meda e para recebimento de dados.

A comunicação Wireless será feita pelo módulo ESP8266 embutido no Arduino Mega, para este projeto foi escolhido a comunicação serial no canal 3,

primeiramente importa-se a biblioteca ***"ESP8266Wifi"*** responsável por acionar as funções necessárias para o controle do módulo, depois deve-se escolher qual a porta usada para a conexão com o servidor Web, para este projeto será a porta padrão 80, dentro da função ***"setup()"*** será configurado a porta serial para 115200 de velocidade de baudrate, essa é a velocidade recomendada para realizar o envio de informações do ESP para o Arduino Mega.

É necessário criar duas variáveis do tipo char para receber o nome da rede wifi que o módulo deve se conectar e a senha de entrada (caso tenha senha). Em seguida é chamada a função ***"Wifi.begin"*** para realizar a conexão com a rede, caso a rede se conecte com sucesso, no monitor serial será apresentando o endereço IP para a conexão com o ESP através da função ***"Wifi.localIP"***, conseguindo assim iniciar o servidor web (Código disponibilizados nos apêndices A, B e C).

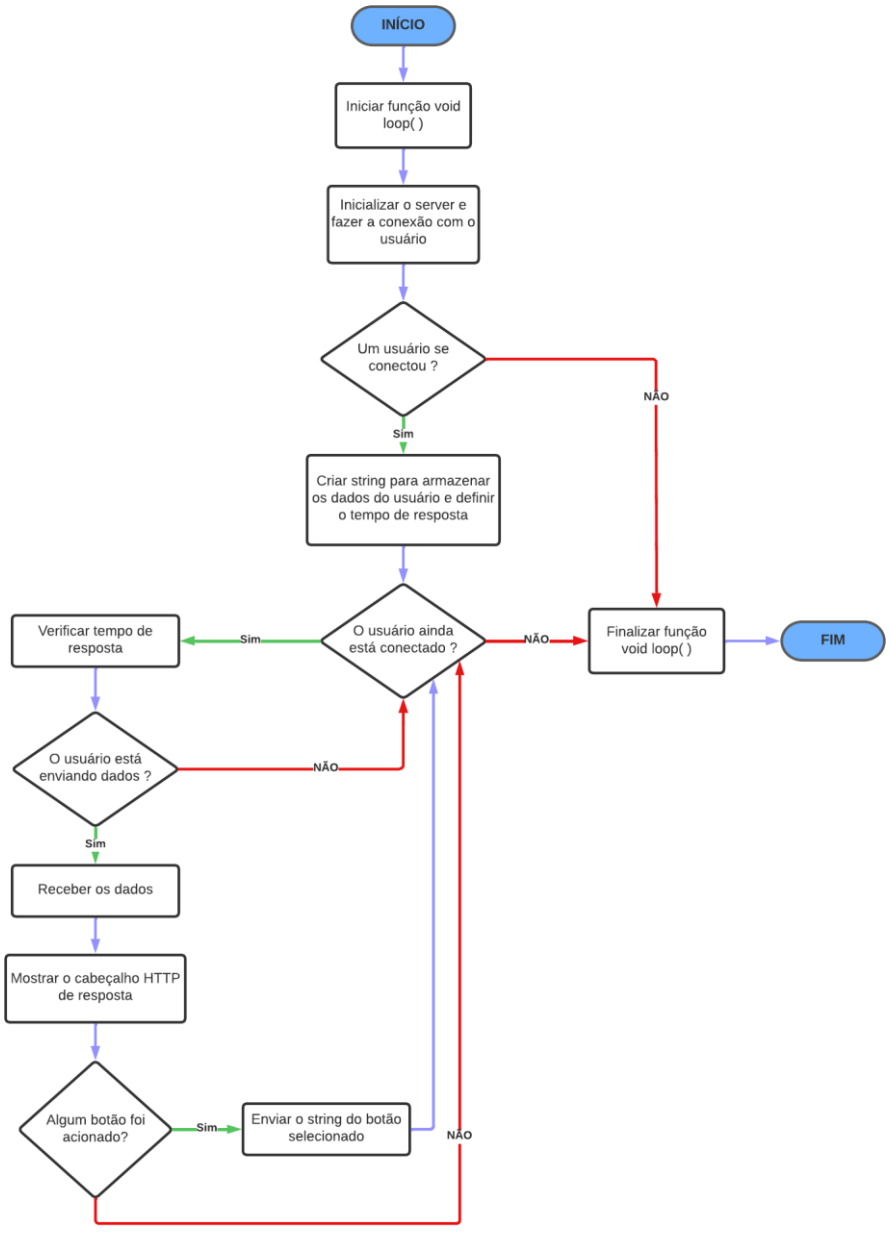
Figura 26 – Fluxograma para interpretar o funcionamento da primeira parte do controle do ESP8266.



Fonte: Autoria própria.

Dentro da função ***"loop()"*** foi criado as solicitações necessárias para os usuários se conectarem ao servidor Web do ESP, através da função ***"server.available()"*** o ESP8266 estará sempre disponível para se conectar com usuários e manterá seu servidor ativo, quando o ESP8266 detecta uma solicitação de um usuário é preciso realizar o armazenamento desses dados, assim usando o laço de repetição while juntando com as funções ***"client.connect, currentTime, previousTime, timeoutTime, client.available, client.read"*** foi criada uma função que faça com que o usuário permaneça conectado ao servidor mesmo após inúmeras solicitações de dados.

Figura 27 – Fluxograma do funcionamento da segunda parte do controle do ESP8266

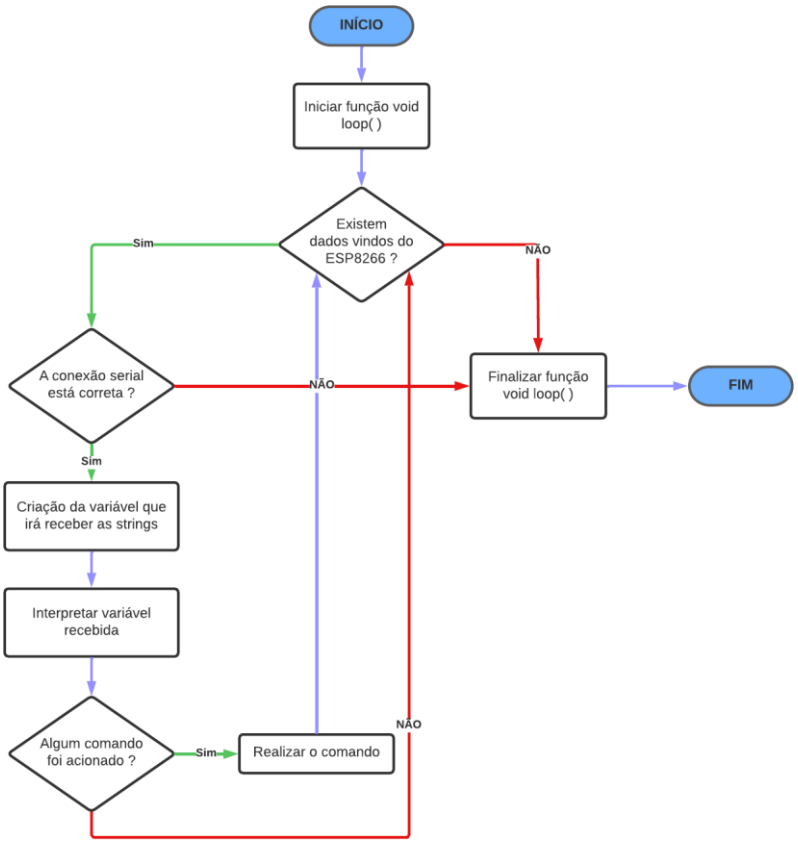


Fonte: Autoria própria.

Após realizar essa leitura da solicitações escolhidas pelo usuário foi criado URLs para fazer a conexão com os comandos controlados pelo Arduino mega, cada URL é acionada ao selecionar cada botão criado na página do servidor Web, ou seja, caso o usuário selecione o botão responsável por realizar o comando de mover o painel para cima, irá acionar a URL “GET /14/up”, ao acionar essa URL será printado no serial de comunicação entre o ESP e o Arduino a string “SERVO_up”, o Arduino irá ler essa string e irá realizar a ação de mover o painel para cima (Código disponibilizados nos apêndices A, B e C).

Para o algoritmo desenvolvido no Arduino mega para a comunicação Wireless tem-se, a declaração de uma variável que irá receber a string de comunicação serial, depois é preciso inicializar as duas comunicações seriais do projeto, a comunicação serial do Arduino, chamada “Serial”, e a comunicação entre o Arduino e o ESP chamada de “Serial3” pois está sendo o canal 3 de comunicação. Dentro da função ***“loop()”*** é necessário fazer com que o Arduino esteja sempre verificando buffer da UART3, e caso existe algum dado solicitado, seja replicado na UART0 e lido pelo Arduino (Código disponibilizados nos apêndices A, B e C).

Figura 28 – Fluxograma d o funcionamento da comunicação do ESP8266 com o Arduino

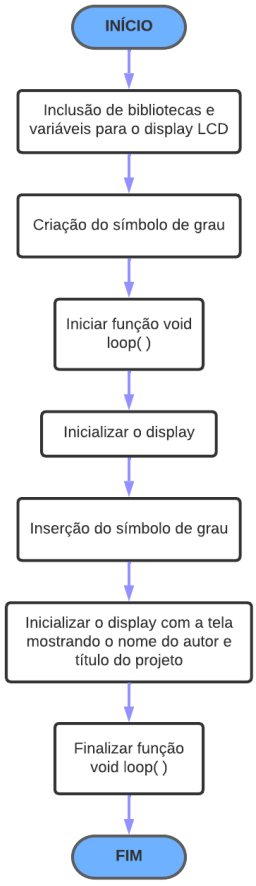


Fonte: Autoria própria.

4.6 Implementação do display LCD e variáveis para parâmetros obtidos

O display LCD escolhido para ser usado no projeto foi um display 16x2, para realizar o desenvolvimento do algoritmo responsável pelo controle do display é necessário o uso das bibliotecas ***"Wire"*** e ***"LiquidCrystal_I2C"***, depois é necessário definir o endereço responsável pela comunicação do display com o barramento, de acordo com o Datasheet será usado o endereço padrão 0x27, também definisse o número de linhas e colunas existentes no display, como o display é 16x2, logo o mesmo terá 16 colunas e 2 linhas. É necessário usar a função ***"LiquidCrystal_I2C"*** para estanciar o endereço, as linhas e as colunas, para que a biblioteca intenda qual o tipo de display será usado. Um pequeno detalhe que foi usado no desenvolvimento é que como será usado o display para informar os valores dos parâmetros, e alguns parâmetros precisam do caractere especial " º ", foi criado a função ***"byte"*** e criado um vetor que será interpretado pelo display e mostrado o caractere (Código disponibilizados nos apêndices A, B e C).

Figura 29 – Fluxograma do funcionamento da primeira parte do controle do display LCD

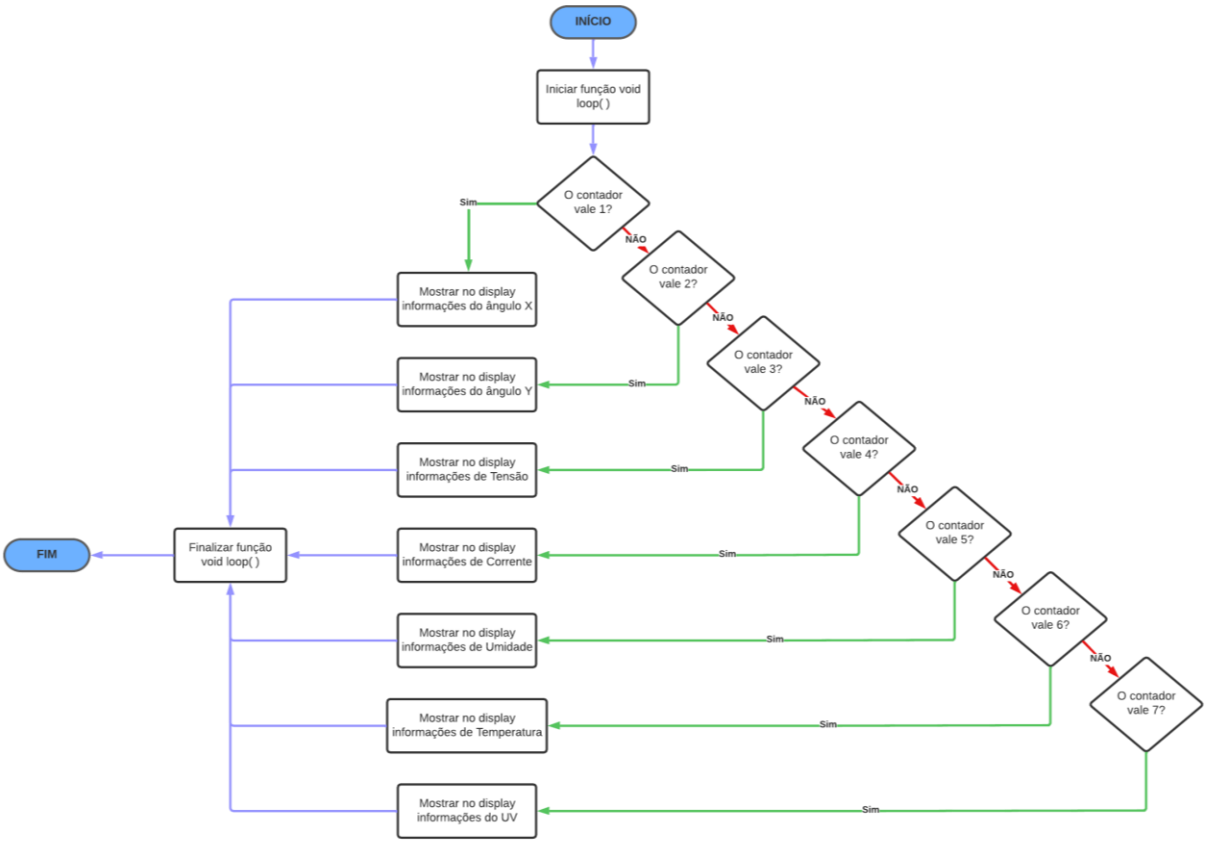


Fonte: Autoria própria.

Dentro da função ***"void()"*** é necessário fazer a inicialização do display, ativar o backlight do visor e a criação do caractere especial "º", usando as funções ***"lcd.init()", "lcd.backlight()"*** e ***"lcd.createdChar()"*** respectivamente.

Dentro da função ***"loop()"*** foi criado uma variável denominada contador, e cada vez que o botão responsável por mostrar o valores dos parâmetros no display for acionado essa variável irá mudar seu valor, com isso cada valor será relacionado com um valor de parâmetro diferente que será mostrado no visor, por exemplo, se o contador estiver com o valor igual a 3 será mostrado no visor o valor da tensão medido pelo módulo sensor medidor de tensão (códigos disponibilizados nos apêndices A, B e C).

Figura 30 – Fluxograma para interpretar o funcionamento da segunda parte do controle do display LCD

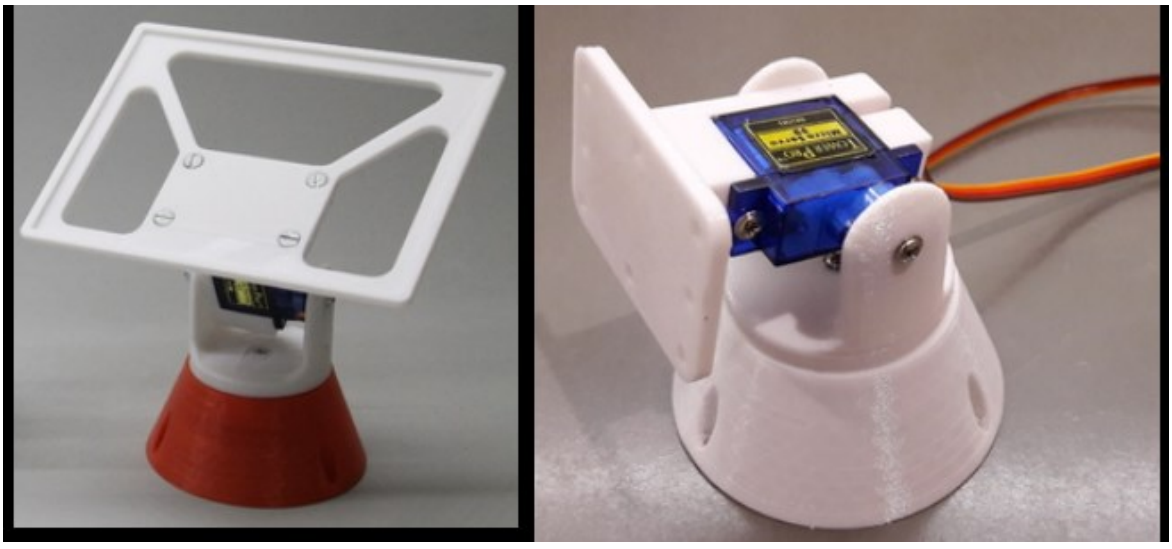


Fonte: Autoria própria.

4.7 Construção da estrutura

Para decidir qual estrutura usar foi realizado acesso no website ***"www.thingiverse.com"***, esse site serve para que pessoas compartilhem seus projetos feitos usando uma impressora 3D, as pessoas disponibilizam os desenhos prontos para serem baixados e imprimidos. Com uma rápida pesquisa foi encontrado o projeto do autor Elitomanolo, um projeto de um desenho de uma base com um suporte para um mini painel solar, a base contém dois encaixes para servo-motores para realizar os movimentos no eixo X e no eixo Y, como mostrado na figura 22, além do uso da base do projeto pelo Elitomanolo foi realizado algumas adições para usar os demais sensores, nas laterais do encaixe do mini painel solar foi posicionado os sensores LDR, para que o sensores não interfiram na medição um do outro e para que não captem reflexos de luz das laterais os sensores foram envoltos com um pedaço de cano preto para isolar e fazer com que eles captem apenas a luz que incide por cima, enquanto que na base foi posicionado na parte inferior os sensores de medição de temperatura e humidade, e o sensor UV, para que fiquem em contato com o ambiente externo sem nenhuma interferência a todo momento. Já os sensores de medição de corrente e tensão foram posicionados por dentro da estrutura do protótipo conectados diretamente com os polos positivos e negativo do painel solar, para a carga foi usado um resistor de potência, a estrutura dos sensores está demonstrado na Figura 32.

Figura 31 – Estrutura responsável pela movimentação do painel solar



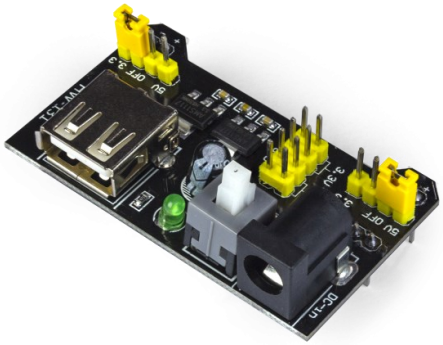
Fonte: Autoria própria.

Figura 32 – Disposição dos sensores na estrutura

Fonte: Autoria própria.

Como apresentado anteriormente, o Arduino escolhido para o uso neste projeto foi o Arduino mega wifi, por apresentar mais portar analógicas que um Arduino comum, e também já possuir o módulo ESP8266 já integrado em seu shield. Foi escolhido usar duas protoboards em vez de criar um PCB shield, pois um dos intuitos deste projeto é que o mesmo seja usado para uso didático, logo com a protoboard os alunos poderão estudar e visualizar todas as ligações e ainda poderão retirar ou implementar novos sensores e novas ligações para adquirir conhecimento na área. Foi necessário o uso de um módulo de fonte ajustável (figura 33) para fazer a alimentação do display LCD e do joystick, sendo que somente com a alimentação disponibilizada pela saída do Arduino não era o suficiente para o controle de todos os módulos usados no projeto, sendo assim para o uso total do projeto é necessárias duas fontes de alimentação, ou uma fonte com duas saídas.

Figura 33 – Módulo de fonte ajustável utilizado na estrutura



Fonte: Robocore (2022).

Na estrutura estão presentes quatro LEDs indicativos, dois LEDs vermelhos para indicar qual o modo que está sendo usado no momento (o modo de mesa ou o modo web portátil), um LED verde para indicar se o modo de posicionamento manual está ativado, e um LED azul para indicar se o modo de posicionamento automático está ativado.

Por fim foram adicionados cinco botões na estrutura: um botão para ativar o modo mesa; um botão para ativar o modo web portátil; um botão para ativar o modo posicionamento automático; um botão para ativar o modo posicionamento manual; e um botão para mostrar os valores obtidos pelos sensores no monitor serial do computador.

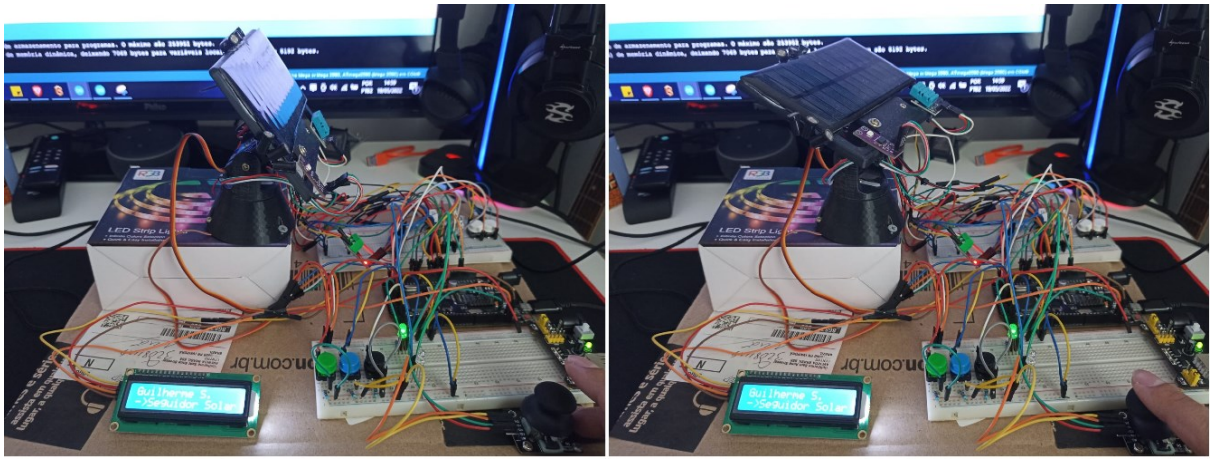
5 RESULTADOS E DISCUSSÕES

Nesta seção serão apresentados todos os modos projetados para o funcionamento do protótipo, enfatizando os resultados obtidos e os erros que surgiram durante o desenvolvimento.

5.1 Modo de uso de mesa

Como ilustrado na Figura 35, tem-se o funcionamento do modo manual, com o indicativo do LED verde, após ser pressionado o botão verde, juntamente com alguns resultados obtidos pelos sensores.

Figura 34 – Modo manual em funcionamento



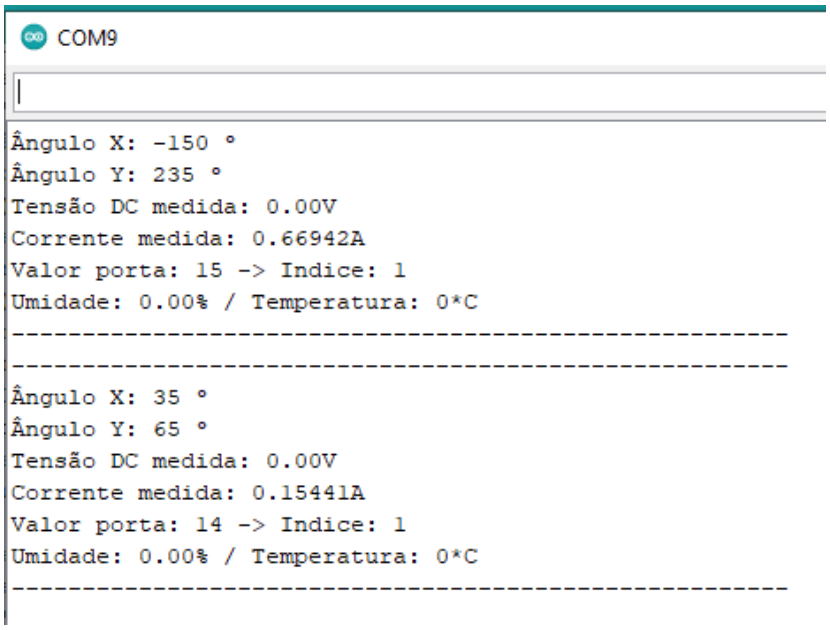
Fonte: Autoria própria.

Na Figura 35 podem ser observados os parâmetros obtidos usando o modo manual. Esses parâmetros serão informados no monitor serial da plataforma Arduino quando pressionado o botão da cor preta presente ao lado dos botões de ativação do modo manual e do modo automático, sendo possível realizar o monitoramento dos parâmetros a qualquer momento.

Esses valores são lidos pelos sensores armazenados nas variáveis de declaração e enviados através da conexão serial RTX 0, a conexão padrão que o

Arduino faz com a entrada USB do computador/notebook. Os parâmetros estão dispostos e organizados como observado na Figura 35.

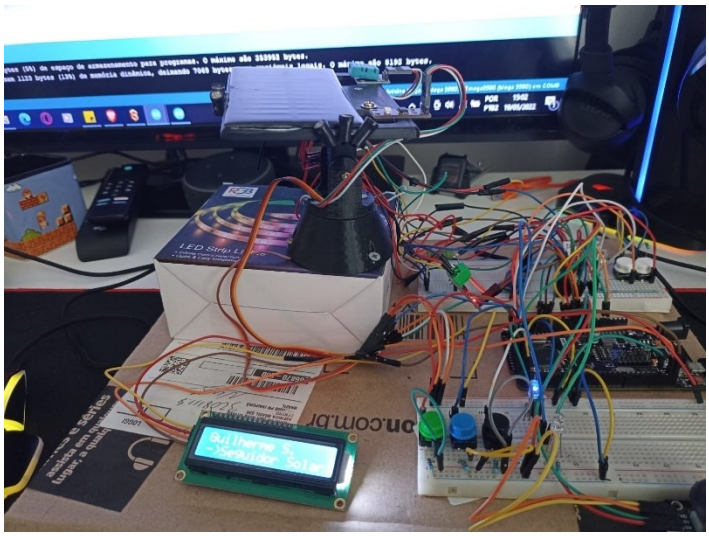
Figura 35 – Parâmetros obtidos usando o modo manual



Fonte: Autoria própria.

Na Figura 36 pode ser observado o funcionamento do modo automático se apresenta após pressionar o botão azul e ter o LED azul ligado como indicativo, as figuras abaixo demonstrar seu funcionamento e alguns resultados obtidos pelos sensores.

Figura 36 – Modo automático em funcionamento

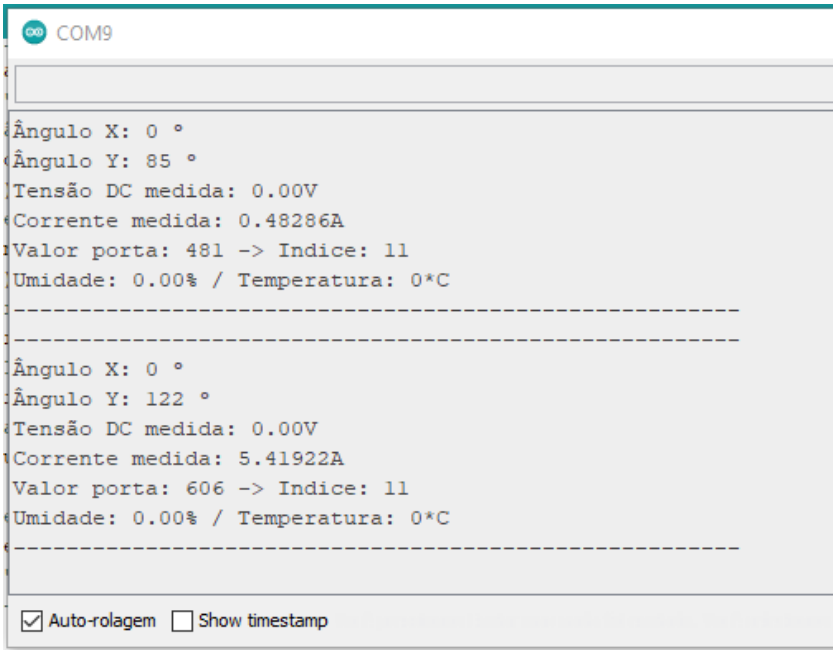


Fonte: Autoria própria.

Da mesma forma apresentada anteriormente, na Figura 38 pode ser observado os parâmetros obtidos usando o modo automático. O mesmo botão usado no modo manual pode ser pressionado no modo automático que também irá mostrar os valores obtidos pelos sensores. O mesmo botão foi usado para economizar espaço e entradas, o botão não apresentará interferência entre os valores dos modos, pois foi declarado de forma diferente, sendo ela ativada dependendo de qual modo foi selecionado.

Os valores são lidos pelos sensores e são enviados também com a utilização da conexão serial RTX 0, os parâmetros estão dispostos e organizados seguindo o mesmo padrão do modo manual.

Figura 37 – Parâmetros obtidos usando o modo automático



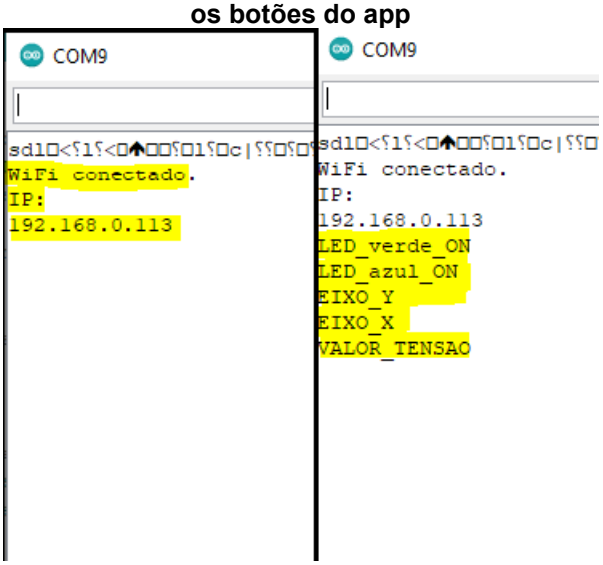
Fonte: Autoria própria.

5.2 Modo de uso web portátil

Para o uso deste modo é necessário o uso de algum dispositivo que realize a conexão via IP, o layout desenvolvido é compatível com dispositivos celulares, notebooks e computadores. E do mesmo jeito que no modo de uso de mesa, ao selecionar os modos automático ou manual via app, os LEDs indicativos verde e azul irão acionar. A seguir é demonstrado o funcionamento do layout do app

desenvolvido e os resultados obtidos pelos sensores mostrados no painel do display LCD, como explicado no Capítulo 4.

Figura 38 – Realização da conexão do protótipo ao wifi e transferência de dados ao selecionar os botões do app



Fonte: Autoria própria.

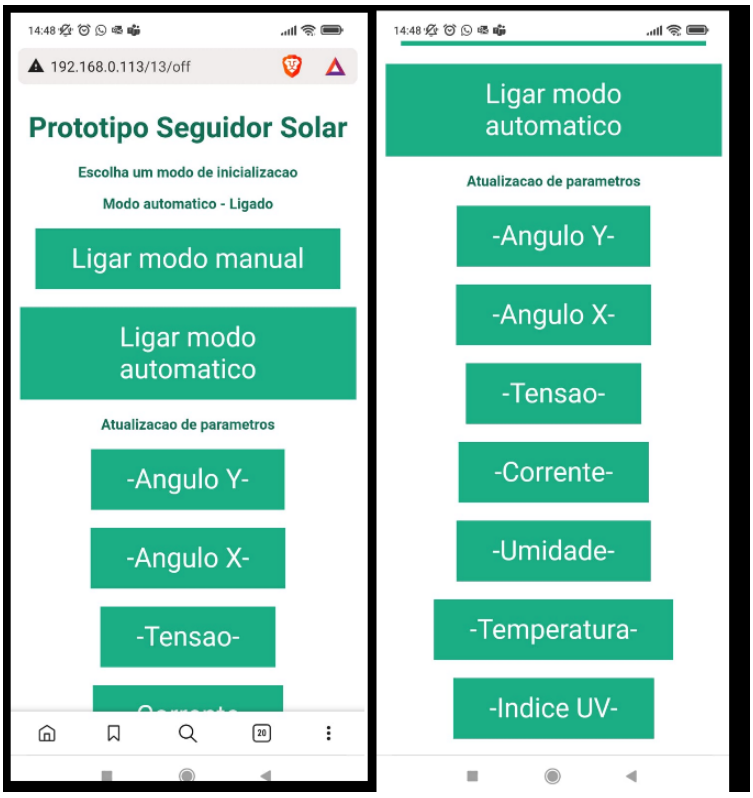
Na Figura 39 pode ser observado o APP quando é acessado pelo celular, o APP foi desenvolvido usando o zoom e a centralização automáticos da tela do dispositivo que está sendo usado para acessar o IP do ESP8266, eliminando os problemas que poderiam causar a desorganização dos layouts dos botões. O usuário pode ativar o protótipo em modo manual ou automático, a diferença entre essas 2 opções será mostrada nas Figuras 40 e 41.

Figura 39 – Apresentação inicial do app ao acessar via celular

Fonte: Autoria própria.

Na Figura 40 é observado o APP com o modo automático selecionado. O APP irá exibir qual modo está acionado para não confundir o usuário, no caso do modo automático irão ser habilitadas as opções de monitorar todos os parâmetros dos sensores do protótipo, ao selecionar algum sensor sua informação será exibida no display LCD (Figura 42). Para atualizar os valores dos parâmetros basta clicar no botão novamente.

Figura 40 – Layout do app com o modo automático selecionado

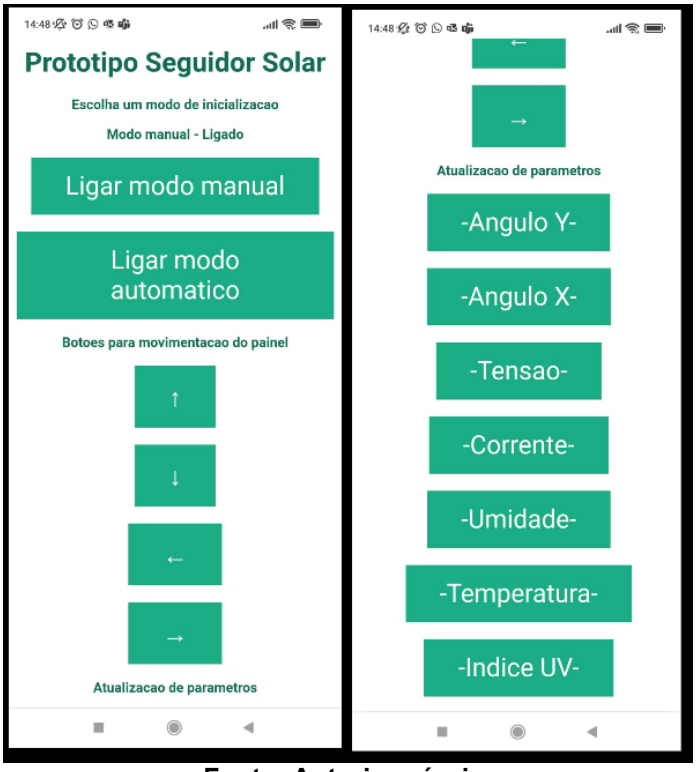


Fonte: Autoria própria.

Na Figura 41 é observado o APP com o modo manual selecionado. Novamente o APP irá exibir qual modo está acionado para não confundir o usuário, no caso do modo manual além de ser habilitadas as opções de monitorar todos os parâmetros dos sensores do protótipo, será habilitado os botões de controle de movimentação dos servos motores da estrutura.

Os botões de movimentação só funcionam com um movimento por vez, caso será pressionado e segurado o botão só irá contar com um movimento. De mesma forma que para o modo automático os parâmetros dos sensores do protótipo serão exibidos no display LCD (Figura 42).

Figura 41 – Layout do app com o modo manual selecionado



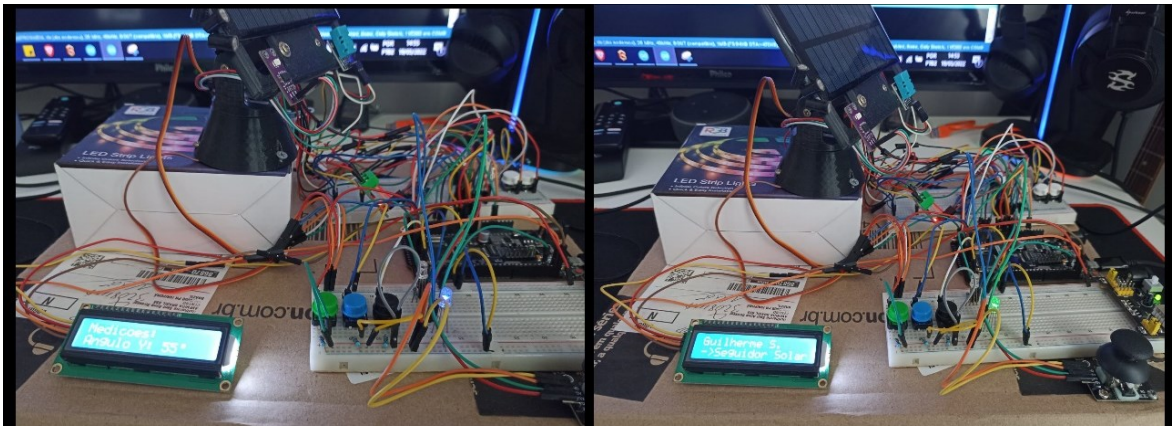
Fonte: Autoria própria.

Na Figura 42 podem ser observadas como são apresentadas as informações no display LCD do protótipo. Ao clicar no botão de -Corrente- o painel irá mostrar o valor medido pelo sensor medidor de corrente no momento em que o botão foi pressionado, como os valores não são fixos, e ficam variando com o passar do tempo, é necessário pressionar o botão novamente para observar o valor mais atualizado. Não foi possível implementar uma função que realizasse a atualização dos parâmetros constantemente em tempo real com apenas um click do botão.

Figura 42 – Layout das medições apresentadas no display LCD

Fonte: Autoria própria

Figura 43 – Funcionamento dos modos manual e automático com pelo app



Fonte: Autoria própria

5.3 Dificuldades encontradas

A falta de documentação e de informações de maneira formal acerca do ESP8266 e do Arduino Mega Wifi dificultaram bastante o aprendizado de suas funções, conexões e aplicações das bibliotecas. Ambos os dispositivos merecem uma exploração maior de aplicação para desenvolvimento de projetos, com ênfase no Arduino Mega Wifi pela sua facilidade em ter mais portas GPIOs e possui um módulo wifi integrado em seu shield.

Ao longo do desenvolvimento deste projeto a maior dificuldade se deu ao tentar estabelecer uma conexão serial entre o Arduino e o ESP8266, sendo que a passagem de dados do Arduino para o ESP8266 apresentava muito erros e falhas ao tentar fazer com que os parâmetros obtidos pelos sensores fossem representados diretamente no layout do app, e para contornar a situação foi necessário o uso de um display LCD para demonstrar os parâmetros obtidos. Outra dificuldade se deu pela instabilidade nas conexões devido ao uso de protoboards para realizar as mesmas, mas como já dito anteriormente, por se tratar de um protótipo didático preferiu-se o uso das mesmas.

5.3.1 Erros que mais ocorreram

Muito comum ao desenvolver algoritmos é aprender o uso correto das bibliotecas para cada sensor, logo, erros ao importar bibliotecas e funções erradas tornam-se bastante presentes. Outro erro que ocorreu com frequência foi a interpretação de dados do potenciômetro pelos servos-motores, com um toque

demorado e sem cuidado nos potenciômetros pode fazer com que o ângulo dos servos altere seu valor muito rapidamente causando instabilidade nos servos e deixando-os perdidos, tento que ser obrigado a realizar o reset do protótipo.

Entretanto com o uso contínuo do protótipo alguns erros podem ocorrer, erros já esperados no desenvolvimento, como se está usando protoboards alguns sensores podem apresentar mal contato entre seus GPIOs, erro facilmente percebido ao analisar os valores apresentados no monitor serial, para contornar o erro basta reajustar as ligações com mais firmeza.

5.4 Conclusão e trabalhos futuros

Os resultados finais obtidos foram bastante satisfatórios, pois todos os resultados esperados foram alcançados. Foram realizados vários testes no modo de mesa e no modo web portátil, em vários horários diferentes no decorrer do dia para testar como os sensores e o painel iriam responder aos estímulos. Em todos os testes o sistema respondeu de forma satisfatória, enfatizando somente os problemas já apresentados anteriormente, com o uso contínuo pode ocorrer interferência nas entradas GPIOs devido a utilização das protoboards, e em algumas vezes acabou apresentando falta de conexão com o ESP8266, porém em questão de segundos a conexão era restaurada.

Apesar das dificuldades de encontrar estudos mais formais e complexos da utilização do módulo ESP8266 e do Arduino Mega Wifi, são dois microcontroladores que valem o investimento no campo da eletrônica e da automação, com base nesses tópicos, ainda pode ser feitos diversos projetos acadêmicos para explorar todas as possibilidades que esses microcontroladores podem nos apresentar.

Esse protótipo pode ser utilizado como base para outros projetos, e como sugestão de trabalhos futuros pode ser citado:

- Abordar melhorias na implementação dos sensores, e melhorias na aplicação das funções usadas para desenvolver o algoritmo;
- Testar outras funcionalidades dos sensores, podem realizar a aplicação de outros sensores, no caso do ESP8266, pode ser implementado controles por infra vermelho, controles analógicos de motores mais potentes que o servo SG90, sistema de segurança, implementação de uma comunicação bluetooth;

- Para eliminar os problemas de má conexão e interferência, pode ser desenvolvido placas de circuito impresso para realizar a conexão dos sensores com solda;
- Como a plataforma Arduino possui integração com o Matlab, pode ser desenvolvido um controle PID para melhorar a movimentação e a resposta dos servos motores;
- Realizar a implementação do módulo ESP32 substituindo o módulo ESP8266 para obter funções e aplicações Wireless mais especificas e evoluídas;
- Aperfeiçoar a comunicação TCP/IP de tal modo que seja possível acessar e controlar o protótipo de qualquer lugar que tenha acesso à um ponto de conexão de rede.

REFERÊNCIAS

ALBUQUERQUE, Yure. **Primeiros passos com o Arduino Mega Wifi.** 2020. Disponível em: < https://blog.smartkits.com.br/primeiros-passos-com-o-arduino-mega-wifi/>. Acesso em: 10 fev 2022.

ALEXANDRU, C.; POZNA, C. Different tracking strategies for optimizing the energetic efficiency of a photovoltaic system. In: IEEE. **Automation, Quality and Testing, Robotics, 2008. AQTR 2008. IEEE International Conference on**. 2008. v. 3, p. 434–439.

ALVES, Pedro. **LDR – O que é e como funciona!**. 2019. Disponível em: https://www.manualdaeletronica.com.br/ldr-o-que-e-como-funciona/. Acesso em: 02 nov. 2021.

ARDUINO. **Escudo Arduino Ethernet V1.** 2006. Disponível em: https://www.arduino.cc/en/Main/ArduinoEthernetShieldV1. Acesso em: 22 nov. 2021.

BRAGA, R. P.; **Energia Solar Fotovoltaica: Fundamentos e Aplicações** 2008. Monografia apresentada ao Curso de Engenheiro Eletricista da Universidade Federal do Rio de Janeiro.

CORTEZ, R. J. M. **Sistema de seguimento solar em produção de energia fotovoltaica**. Dissertação (Mestrado) — Faculdade de Engenharia Universidade do Porto, 2013.

CRESESB. **Manual de Engenharia para Sistemas Fotovoltaicos**. 2014. Disponível em: < http://www.cresesb.cepel.br/index.php?section=publicacoes&task=livro&cid=481>. Acesso em: 03 nov. 2021.

CURVELLO, André. **Apresentando o modulo ESP8266.** 2015. Disponível em: < https://www.embarcados.com.br/modulo-esp8266/>. Acesso em: 20 abril 2022.

DOS REIS, Fábio. **Arduino – Conhecendo os Shields.** 2015. Disponível em: http://www.bosontreinamentos.com.br/eletronica/arduino/arduino-conhecendo-os-shields/. Acesso em 24 nov. 2021.

ELETROGATE. **Servo motor com Arduino: Conheça aplicações e aprenda a usar.** 2017. Disponível em: https://blog.eletrogate.com/servo-motor-para-aplicacoes-com-arduino/. Acesso em: 28 nov. 2021.

ELETRONIC, Snhenzhen Ruite. **Datasheet RT162-7**. 2020. Disponível em: <https://www.filipeflop.com/img/files/download/Datasheet_Display_16x2.pdf>. Acesso em: 15 fev 2022.

EVANS, Martin; NOBLE, Joshua; HACHENBAUM, Jordan. **Arduino: Em ação**. 3.

ed. São Paulo: Novatec, 2013. 423 p.

FREITAS, Roberto Augusto. **Automação Residencial utilizando Arduino e aplicação WEB.** 2014. 178 f. Tese (Graduação em Engenharia da computação) – Centro Universitário de Brasília, Brasília 2014.

HULD, T. et al. Analysis of one-axis tracking strategies for pv systems in europe. **Progress in Photovoltaics: Research and Applications**. Wiley Online Library, v. 18, n. 3, p. 183–194, 2010

KOLBAN, N. **Kolban's Book on ESP8266**. 2016. Disponível em: <https://leanpub.com/ESP8266_ESP32>. Acesso em: 19 abril 2022.

LAPIS. **Datasheet: ML8511.** 2013. Disponível em: https://datasheetspdf.com/pdf/1188772/LAPIS/ML8511/1. Acesso em 22 nov. 2021.

MACEDO. W. N. **Análise do fator de dimensionamento do inversor aplicado a sistemas fotovoltaicos conectados à rede**. São Paulo, Brasil: Tese de Doutorado, Universidade de São Paulo. Brasil, 2006. 183 p.

MIYADAIRA, Alberto Noboru. **Microcontroladores PIC18: aprenda e programe em linguagem C**. 4. ed. São Paulo: Érica, 2013.

MURTA, José Gustavo Abreu. **Guia completo do display LCD – Arduino.** 2022. Disponível em: <https://blog.eletrogate.com/guia-completo-do-display-lcd-arduino/>. Acesso em: 15 fev 2022.

NICOLOSI, Denys Emílio Campion. **Microcontrolador 8051 Detalhado**. 9. ed. São Paulo: Érica, 2013. Disponível em: https://integrada.minhabiblioteca.com.br/books/9788536519876. Acesso em: 30 out. 2021.

PEREIRA, Tiago Quartiero. **Aplicação da metodologia de taxonomia de bloom revisada no ensino de física a parte da análise de dados de estações meteorológicas.** 2018. 178 f. Tese (Pós-Graduação em Física) – Universidade Federal de Santa Catarina, Araranguá 2018.

PUHLMANN, Henrique Frank Werner. **Módulo de Display LCD.** 2015. Disponível em: < https://www.embarcados.com.br/modulo-de-display-lcd/>. Acesso em: 15 fev 2022.

RAMALHO, J. A. **Iniciando em HTML.** 1996. M.M. Filho, Ed. São Paulo: Makron Books.

Reis, Pedro. **Diferenças entre seguidores solares de eixo único e eixo duplo.** 2017. Disponível em: https://www.portal-energia.com/diferencas-seguidores-solares-eixo-unico-eixo-duplo/. Acesso em 24 nov. 2021.

Reis, Pedro. **Em que consiste um sistema de seguidor solar fotovoltaico.** 2016. Disponível em: https://www.portal-energia.com/em-que-consiste-sistema-seguidor-

solar-fotovoltaico/. Acesso em: 24 nov. 2021.

ROBOCORE. **Micro Servo 9g SG90.** 2020. Disponível em: https://www.robocore.net/servo-motor/micro-servo-9g-sg90-towerpro. Acesso em: 28 nov 2021.

ROITHNER. **Datasheet: GUVA-S12SD.** 2011. Disponível em: https://datasheetspdf.com/pdf/1093061/ROITHNER/GUVA-S12SD/1. Acesso em: 22 nov. 2021.

SOARES, G. F. W. ; VIEIRA, L. S. R. ; GALDINO, M. A. ; OLIVIERI, M. M. A. ; BORGES, E. L. P. ; CARVALHO, C. M. ; LIMA, A. A. N. . Avaliação Técnico-Econômica da Aplicação de Sistemas Fotovoltaicos Individuais e de Centrais com Minirredsr na Eletrificação Rural. **Revista Brasileira de Energia Solar** , v. II, p. 117-128, 2010.

SOUZA, Daniel Rodrigues de; SOUZA, David José de. **Desbravando o PIC18: ensino didático**. 1. ed. São Paulo: Érica, 2012. Disponível em: https://integrada.minhabiblioteca.com.br/books/9788536518329. Acesso em: 30 out. 2021.

SUPPORT, Tech. **Datasheet: W5100 Ethernet Shield.** 2012. Disponível em: https://www.curtocircuito.com.br/datasheet/modulo/Modulo_Ethernet_Shield.pdf. Acesso em: 22 nov. 2021.

THOMSEN, Adilson. **Medidor de índice com Arduino.** 2015. Disponível em: https://www.filipeflop.com/blog/medidor-de-indice-uv-com-arduino/. Acesso em: 10 nov. 2021.

THONSEM, Adilson. **O que é Arduino ?**. 2014. Disponível em: https://www.filipeflop.com/blog/o-que-e-arduino/. Acesso em: 02 nov. 2021.

UK, D-Robotics. **Datasheet: DHT11 Humidity & Temperature Sensor.** 2010. Disponível em: https://datasheetspdf.com/pdf/785590/D-Robotics/DHT11/1. Acesso em: 22 nov. 2021.

APÊNDICE A - VARIÁVEIS E BIBLIOTECAS

Código ESP8266 – variáveis e bibliotecas

```
#include <ESP8266WiFi.h>
// Substitua pelas suas credenciais de rede.
const char* ssid     = "Nome rede";
const char* password = "senha";
// Defina um servidor com a porta 80.
WiFiServer server(80);
// Variável para armazenar a solicitação HTTP.
String header;
// Variável para armazenar o estado de saída atual.
String outputState = "Desligado";
String outputState2 = "Desligado";
unsigned long currentTime = millis();
unsigned long previousTime = 0;
// Defina um timeout de 2000 milisegundos.
const long timeoutTime = 2000;
```

Código Arduino Mega – variáveis e bibliotecas

```
//DEFINIÇÃO DOS LEDS, BOTÕES E VARIÁVEIS NECESSÁRIAS
String msg;//String para armazenar a mensagem recebida pela porta serial 3.
#define LedVerdeManual 33
#define LedAzulAutomatico 32
#define botaoVerde 35
#define botaoAzul 34
#define botaoResultado 30
#define botaoIniciarWEB 47
#define botaoIniciarMESA 46
#define ledWEB 45
#define ledMESA 44
#define LedVerdeManualWEB 42
#define LedAzulAutomaticoWEB 43
int web = 0;
int contador = 0;
int modo = 0;
int teste = 0;
```

```
//INCLUSÃO BIBILIOTECAS E VARIAVEIS DOS SERVO MOTORES
#include <Servo.h>
Servo ServoY;
Servo ServoX;
int posicaoServoY = 50;
int posicaoServoX = 50;
int contadorX = 0;
int contadorY = 0;
#define controleX A0
#define controleY A1
// INCLUSÃO DE BIBLIOTECAS E VARIÁVEIS do LCD
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
// DEFINIÇÕES
#define endereco  0x27 // Endereços comuns: 0x27, 0x3F
#define colunas   16
#define linhas    2
// INSTANCIANDO OBJETOS
LiquidCrystal_I2C lcd(endereco, colunas, linhas);
//Array simbolo grau
byte grau[8]={B00111,B00101,B00111,B00000,B00000,B00000,B00000,B00000};
int thisChar = 0 ;
//INCLUSÃO DE BIBLIOTECAS VARIÁVEIS PARA O MÓDULO DE MEDIDOR DE TENSÃO
#define pinoSensor A2 //Pino de entrada analógico para o sensor
float tensaoEntrada = 0.0; //Variavel que irá armazenar o valor da tensão de entrada no sensor
float tensaoMedida = 0.0; //Variavel que irá armazenar o valor da tensão medida no sensor
float valorR1 = 30000.0; //Valor do resistor 1 do divisor de tensão do módulo
float valorR2 = 7500.0; //Valor do resistor 2 do divisor de tensão do módulo
int leituraSensor = 0; //Variavel responsável por amazenar os valores de leitura da entrada analógica
//INCLUSÃO DE BIBLIOTECAS E VARIÁVEIS PARA O MÓDULO DE MEDIDOR DE CORRENTE
#include "EmonLib.h" //INCLUSÃO DE BIBLIOTECA
#define CURRENT_CAL 18.40 //VALOR DE CALIBRAÇÃO (DEVE SER AJUSTADO EM PARALELO
COM UM MULTÍMETRO MEDINDO A CORRENTE DA CARGA)
#define pinoSensor2 A3 //PINO ANALÓGICO EM QUE O SENSOR ESTÁ CONECTADO
//float ruido = 0.08; //RUÍDO PRODUZIDO NA SAÍDA DO SENSOR (DEVE SER AJUSTADO COM A
CARGA DESLIGADA APÓS CARREGAMENTO DO CÓDIGO NO ARDUINO)
float ruido = 0.18;
EnergyMonitor emon1; //CRIA UMA INSTÂNCIA
//INCLUSÃO DE BIBLIOTECAS E VARIÁVEIS PARA O MÓDULO DE ÍNDICE UV
int indiceUV;
```

```
//INCLUSÃO DE BIBLIOTECAS E VARIÁVEIS PARA O MÓDULO DE MEDIDOR DE TEMPERATURA E HUMIDADE
#include "dht.h" //INCLUSÃO DE BIBLIOTECA
#define pinoDHT11 A5 //PINO ANALÓGICO UTILIZADO PELO DHT11
dht DHT; //VARIÁVEL DO TIPO DHT
//INCLUSÃO DOS SENSORES E VARIÁVEIS LDR
#define pinLDR1 A10 //-> DEFINIÇÃO DA PORTA ANALÓGICA PARA O LDR Y_1
#define pinLDR2 A9 //-> DEFINIÇÃO DA PORTA ANALÓGICA PARA O LDR Y_2
#define pinLDR3 A8 //-> DEFINIÇÃO DA PORTA ANALÓGICA PARA O LDR Y_3
#define pinLDR4 A11 //-> DEFINIÇÃO DA PORTA ANALÓGICA PARA O LDR X_1
#define pinLDR5 A12 //-> DEFINIÇÃO DA PORTA ANALÓGICA PARA O LDR X_2
#define pinLDR6 A13 //-> DEFINIÇÃO DA PORTA ANALÓGICA PARA O LDR X_3
int menorY = 0; //-> VARIÁVEL RESPONSÁVEL POR RECEBER A MAIOR INCIDENCIA LUMINOSA NO EIXO Y
int menorX = 0; //-> VARIÁVEL RESPONSÁVEL POR RECEBER A MAIOR INCIDENCIA LUMINOSA NO EIXO X
int PosicaoInicialAutomaticoY = 50;
int PosicaoInicialAutomaticoX = 50;
```

APÊNDICE B FUNÇÃO SETUP()

Código ESP8266 – função Setup()

```
Serial.begin(115200);
delay(5000);
// Conecte com a rede Wi-Fi com SSID e senha.
//Serial.print("Conectando a rede ");
//Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
 }
// Print local IP address and start web server
Serial.println("");
Serial.println("WiFi conectado.");
Serial.println("IP: ");
Serial.println(WiFi.localIP());
//Inicie o servidor.
server.begin();
```

Código Arduino Mega – função Setup()

```
//Definindo as portas digitais DOS BOTÕES E DOS LEDS
pinMode(LedVerdeManual, OUTPUT);
pinMode(LedAzulAutomatico, OUTPUT);
pinMode(LedVerdeManualWEB, OUTPUT);
pinMode(LedAzulAutomaticoWEB, OUTPUT);
pinMode(ledWEB, OUTPUT);
pinMode(ledMESA, OUTPUT);
pinMode(botaoVerde, INPUT);
pinMode(botaoAzul, INPUT);
pinMode(botaoResultado, INPUT);
pinMode(botaoIniciarWEB, INPUT);
pinMode(botaoIniciarMESA, INPUT);
//Definindo os servos
ServoY.attach(49);
ServoY.write(posicaoServoY);
```

```
ServoX.attach(51);
ServoX.write(posicaoServoX);
//DEFININDO SENSOR MEDIDOR DE TENSÃO
pinMode(pinoSensor, INPUT); //definindo o pino de entrada do sensor
//DEFININDO SENSOR DE MEDIDOR DE CORRENTE
emon1.current(pinoSensor2, CURRENT_CAL); //PASSA PARA A FUNÇÃO OS PARÂMETROS (PINO ANALÓGIO / VALOR DE CALIBRAÇÃO)
//INICIALIZAÇÃO DO LCD
lcd.init();                  // inicializa LCD
lcd.backlight();             // ativa led de backlight
lcd.clear();
//Cria o caractere customizado com o simbolo do grau
lcd.createChar(1, grau);
lcd.setCursor(0, 0);         // selecionando coluna 0 e linha 0
lcd.print("Guilherme S.");   // print da mensagem
lcd.setCursor(0,1);
lcd.print("->Seguidor Solar");
//Defina a porta serial para comunicação usb.
Serial.begin(115200);
//Defina a porta serial para comunicação com ESP8266.
Serial3.begin(115200);
```

APÊNDICE C – FUNÇÃO LOOP()

Código ESP8266 – função loop()

```
WiFiClient client = server.available();   // Escute os clientes conectados
  if (client) {                           // Se um novo cliente se conectar
    String currentLine = "";              // Faça uma String para armazenar dados recebidos do cliente
    currentTime = millis();
    previousTime = currentTime;
    while (client.connected() && currentTime - previousTime <= timeoutTime) { // loop enquanto cliente
estiver conectado.
      currentTime = millis();
      if (client.available()) {           // Se holver bytes para ler do cliente,
        char c = client.read();           // faça a leitura.
        header += c;
        if (c == '\n') {                  // Se o byte é um caractere de nova linha,
          // é o fim da solicitação HTML,entao envie uma resposta.
          if (currentLine.length() == 0) {
            //Envie um cabeçalho HTTP de resposta.
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();

            // Procure o trecho "GET /13/on" dentro da solicitação do recebida do cliente.
            if (header.indexOf("GET /13/on") >= 0) {
              //Envie um comando para Mega2560 via serial.
              Serial.println("LED_verde_ON");
              //Altere a variavel de estado.
              outputState = "VerdeLigado";
            } else if (header.indexOf("GET /13/off") >= 0) {
              //Envie um comando para Mega2560 via serial.
              Serial.println("LED_azul_ON");
              //Altere a variavel de estado.
              outputState = "AzulLigado";
            }
            // Comandos para controles servos manual
            if (header.indexOf("GET /14/up") >= 0) {
              Serial.println("SERVO_up");
            }
            if (header.indexOf("GET /14/down") >= 0) {
```

```
            Serial.println("SERVO_down");
          }
          if (header.indexOf("GET /14/left") >= 0) {
            Serial.println("SERVO_left");
          }
          if (header.indexOf("GET /14/right") >= 0) {
            Serial.println("SERVO_right");
          }
          //Atualização de parametros
          if (header.indexOf("GET /15/Y") >= 0) {
            Serial.println("EIXO_Y");
          }
          if (header.indexOf("GET /15/X") >= 0) {
            Serial.println("EIXO_X");
          }
          if (header.indexOf("GET /15/TENSAO") >= 0) {
            Serial.println("VALOR_TENSAO");
          }
          if (header.indexOf("GET /15/CORRENTE") >= 0) {
            Serial.println("VALOR_CORRENTE");
          }
          if (header.indexOf("GET /15/UMIDADE") >= 0) {
            Serial.println("VALOR_UMIDADE");
          }
          if (header.indexOf("GET /15/TEMPERATURA") >= 0) {
            Serial.println("VALOR_TEMPERATURA");
          }
          if (header.indexOf("GET /15/UV") >= 0) {
            Serial.println("VALOR_UV");
          }

          // Pagina HTML
          client.println("<!DOCTYPE html><html>");
          client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
          client.println("<link rel=\"icon\" href=\"data:,\">");
          // CSS para estilizar a pagina
          client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}");
          client.println("h1,p {font-weight: bold;color: #126e54; font-size: 32px;}");
```

```
            client.println("p {font-size: 16px;}");
            client.println(".button { background-color: #1BAE85; border: none; color: white; padding: 16px
40px;");
            client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");
            client.println("</style></head>");
            client.println("<body><h1>Prototipo Seguidor Solar</h1>");
            client.println("<p>Escolha um modo de inicializacao</p>");
// Mostre o estado atual do pino 13, aqui representado pela variavel de estado outputState.
            if (outputState == "VerdeLigado") {
              client.println("<p>Modo manual - Ligado</p>");
            }
            if (outputState == "AzulLigado") {
              client.println("<p>Modo automatico - Ligado</p>");
            }
            client.println("<p><a href=\"/13/on\"><button class=\"button\">Ligar modo
manual</button></a></p>");
            client.println("<p><a href=\"/13/off\"><button class=\"button\">Ligar modo
automatico</button></a></p>");
            if (outputState == "VerdeLigado") {
              client.println("<p>Botoes para movimentacao do painel</p>");
              client.print("<p><a href=\"/14/up\"><button class=\"button\">&#8593</button></a></p>");
              client.print("<a href=\"/14/down\"><button class=\"button\">&#8595</button></a>");
              client.print("<p><a href=\"/14/left\"><button class=\"button\">&#8592</button></a></p>");
              client.print("<a href=\"/14/right\"><button class=\"button\">&#8594</button></a>");
            }
            client.println("<p>Atualizacao de parametros</p>");
            client.print("<p><a href=\"/15/Y\"><button class=\"button\">-Angulo Y-</button></a></p>");
            client.print("<p><a href=\"/15/X\"><button class=\"button\">-Angulo X-</button></a></p>");
            client.print("<p><a href=\"/15/TENSAO\"><button class=\"button\">-Tensao-
</button></a></p>");
            client.print("<p><a href=\"/15/CORRENTE\"><button class=\"button\">-Corrente-
</button></a></p>");
            client.print("<p><a href=\"/15/UMIDADE\"><button class=\"button\">-Umidade-
</button></a></p>");
            client.print("<p><a href=\"/15/TEMPERATURA\"><button class=\"button\">-Temperatura-
</button></a></p>");
            client.print("<p><a href=\"/15/UV\"><button class=\"button\">-Indice UV-</button></a></p>");
            client.println("</body></html>");
            // A resposta HTTP termina com outra linha em branco.
            client.println();
```

```
          break;
        } else { // Se você recebeu uma nova linha, limpe currentLine
          currentLine = "";
        }
      } else if (c != '\r') {  // Se você tiver mais alguma coisa além de um caractere de retorno de carro,
        currentLine += c;      // Adicione-o ao final do currentLine.
      }
    }
  }
  // Limpe a variável de cabeçalho
  header = "";
  // Feche a conexão.
  client.stop();
}
```

Código Arduino Mega – função loop()

```
if(digitalRead(botaoIniciarWEB) == HIGH){
  web = 1;
  digitalWrite(ledWEB, HIGH);
  digitalWrite(ledMESA, LOW);
  digitalWrite(LedVerdeManualWEB, LOW);
  digitalWrite(LedAzulAutomaticoWEB, LOW);
  digitalWrite(LedVerdeManual, LOW);
  digitalWrite(LedAzulAutomatico, LOW);
  }
if(digitalRead(botaoIniciarMESA) == HIGH){
  web = 2;
  digitalWrite(ledWEB, LOW);
  digitalWrite(ledMESA, HIGH);
  digitalWrite(LedVerdeManualWEB, LOW);
  digitalWrite(LedAzulAutomaticoWEB, LOW);
  digitalWrite(LedVerdeManual, LOW);
  digitalWrite(LedAzulAutomatico, LOW);
  }
//INICIALIZAÇÃO DO CONTROLE PELO WEB SERVER
if(web == 1){  //Aguarde dados vindos do ESP8266.
  if (Serial3.available()) {
```

```
    //Leitura de um byte.
    char data = Serial3.read();
    //Imprima o mesmo dado pela porta usb.
    Serial.print(data);
    //Acrescente o caractere recebido a string de mensagem.
    msg += data;
     if (data == 13) {//Limpa a string ao receber caractere CR(Carriage return).
      msg = "";
    }
//DECLARANDO O STRINGS RESPONSÁVEIS PELOS CONTROLES DO SEGUIDOR
    if (msg.indexOf("LED_verde_ON") > 0) {
    //Ativa luz modo manual.
     digitalWrite(LedVerdeManualWEB, HIGH);
     digitalWrite(LedAzulAutomaticoWEB, LOW);
     digitalWrite(LedVerdeManual, LOW);
     digitalWrite(LedAzulAutomatico, LOW);
     teste = 0;
   }
  if (msg.indexOf("SERVO_up") > 0) {
      posicaoServoY = posicaoServoY + 5;
      ServoY.write(posicaoServoY);
    }
    if (msg.indexOf("SERVO_down") > 0) {
      posicaoServoY = posicaoServoY - 5;
      ServoY.write(posicaoServoY);
    }
    if (msg.indexOf("SERVO_left") > 0) {
      posicaoServoX = posicaoServoX - 5;
      ServoX.write(posicaoServoX);
    }
    if (msg.indexOf("SERVO_right") > 0) {
      posicaoServoX = posicaoServoX + 5;
      ServoX.write(posicaoServoX);
    }
    if (msg.indexOf("LED_azul_ON")> 0) {
      //Ativa luz modo automatico
      digitalWrite(LedVerdeManualWEB, LOW);
      digitalWrite(LedAzulAutomaticoWEB, HIGH);
      digitalWrite(LedVerdeManual, LOW);
      digitalWrite(LedAzulAutomatico, LOW);
```

```
  teste = 1;
 }
 if(digitalRead(LedAzulAutomaticoWEB) == HIGH){
  int valorLDR1_1 = analogRead(pinLDR1);
  int valorLDR1_2 = analogRead(pinLDR1);
  int valorLDR1_3 = analogRead(pinLDR1);
  int valorLDR1 = (valorLDR1_1 + valorLDR1_2 + valorLDR1_3)/3;

  int valorLDR2_1 = analogRead(pinLDR2);
  int valorLDR2_2 = analogRead(pinLDR2);
  int valorLDR2_3 = analogRead(pinLDR2);
  int valorLDR2 = (valorLDR2_1 + valorLDR2_2 + valorLDR2_3)/3;

  int valorLDR3_1 = analogRead(pinLDR3);
  int valorLDR3_2 = analogRead(pinLDR3);
  int valorLDR3_3 = analogRead(pinLDR3);
  int valorLDR3 = (valorLDR3_1 + valorLDR3_2 + valorLDR3_3)/3;
  //MOVIMENTO EM Y
  if (valorLDR1 < valorLDR2){
    if(valorLDR1 < valorLDR3){
      menorY = valorLDR1;
      }else{
        menorY = valorLDR3;
        }
    }else{
      if(valorLDR2 < valorLDR3){
        menorY = valorLDR2;
        }else{
          menorY = valorLDR3;
          }
      }
  delay(100);
  if(menorY == valorLDR1){
    //servoY.write(20);
    posicaoServoY = posicaoServoY - 1;
    ServoY.write(posicaoServoY);
    }
  if(menorY == valorLDR2){
    ServoY.write(posicaoServoY);
    }
```

```
    if(menorY == valorLDR3){
      posicaoServoY = posicaoServoY + 1;
      ServoY.write(posicaoServoY);
      }
    if(posicaoServoY <= 0){
      posicaoServoY = posicaoServoY + 1;
      ServoY.write(posicaoServoY);
      }
    if(posicaoServoY >= 180){
      posicaoServoY = posicaoServoY - 1;
      ServoY.write(posicaoServoY);
      }
  }
  if (msg.indexOf("EIXO_Y") > 0) {
    contador = 2;
  }
  if (msg.indexOf("EIXO_X") > 0) {
    contador = 1;
  }
  if (msg.indexOf("VALOR_TENSAO") > 0) {
    contador = 3;
  }
  if (msg.indexOf("VALOR_CORRENTE") > 0) {
    contador = 4;
  }
  if (msg.indexOf("VALOR_UMIDADE") > 0) {
    contador = 5;
  }
  if (msg.indexOf("VALOR_TEMPERATURA") > 0) {
    contador = 6;
  }
  if (msg.indexOf("VALOR_UV") > 0) {
    contador = 7;
  }
//MEDIDOR DE TENSÃO
        leituraSensor = analogRead(pinoSensor); //FAZ A LEITURA DO PINO ANALÓGICO E ARMAZENA NA VARIÁVEL O VALOR LIDO
        tensaoEntrada = (leituraSensor * 5.0) / 1024.0; //VARIÁVEL RECEBE O RESULTADO DO CÁLCULO
```

```
 tensaoMedida = tensaoEntrada / (valorR2/(valorR1+valorR2)); //VARIÁVEL RECEBE O VALOR DE TENSÃO DC MEDIDA PELO SENSOR
//---------------------------------------------------------------------------
//MEDIDOR DE CORRENTE
  emon1.calcVI(17,100); //FUNÇÃO DE CÁLCULO (17 SEMICICLOS / TEMPO LIMITE PARA FAZERA MEDIÇÃO)
  double currentDraw = emon1.Irms; //VARIÁVEL RECEBE O VALOR DE CORRENTE RMS OBTIDO
  currentDraw = currentDraw-ruido; //VARIÁVEL RECEBE O VALOR RESULTANTE DA CORRENTERMS MENOS O RUÍDO
if(currentDraw < 0){ //SE O VALOR DA VARIÁVEL FOR MENOR QUE 0, FAZ
    currentDraw = 0; //VARIÁVEL RECEBE 0
  }
//---------------------------------------------------------------------------
//SENSOR UV
  //Leitura dos dados do sensor
  int leitura_porta = analogRead(A4);
  //De acordo com a leitura define o indice UV corrrspondente
  if (leitura_porta <= 10) {
    indiceUV = 0;
  } else if (leitura_porta > 10 && leitura_porta <= 46) {
    indiceUV = 1;
  } else if (leitura_porta > 46 && leitura_porta <= 65) {
    indiceUV = 2;
  } else if (leitura_porta > 65 && leitura_porta <= 83) {
    indiceUV = 3;
  } else if (leitura_porta > 83 && leitura_porta <= 103) {
    indiceUV = 4;
  } else if (leitura_porta > 103 && leitura_porta <= 124) {
    indiceUV = 5;
  } else if (leitura_porta > 124 && leitura_porta <= 142) {
    indiceUV = 6;
  } else if (leitura_porta > 142 && leitura_porta <= 162) {
    indiceUV = 7;
  } else if (leitura_porta > 162 && leitura_porta <= 180) {
    indiceUV = 8;
  } else if (leitura_porta > 180 && leitura_porta <= 200) {
    indiceUV = 9;
  } else if (leitura_porta > 200 && leitura_porta <= 221) {
    indiceUV = 10;
```

```
    } else {
      indiceUV = 11;
    }
//---------------------------------------------------------------------
//MEDIDOR TEMPERATURA E HUMIDADE
    DHT.read11(pinoDHT11); //LÊ AS INFORMAÇÕES DO SENSOR
//---------------------------------------------------------------------
//CONTROLE DO LCD PELO IP
    if(contador == 1){
      lcd.setCursor(0,0);
      lcd.print("                ");
      lcd.setCursor(0,0);
      lcd.print("Medicoes:");
      lcd.setCursor(0,1);
      lcd.print("                ");
      lcd.setCursor(0,1);
      lcd.print("Angulo X: ");
      if(posicaoServoX<10){
        lcd.print(posicaoServoX);
        lcd.setCursor(11,1);
        lcd.write(1);
      }
      if(posicaoServoX >=10 and posicaoServoX < 100){
        lcd.print(posicaoServoX);
        lcd.setCursor(12,1);
        lcd.write(1);
      }
      if(posicaoServoX>=100){
        lcd.print(posicaoServoX);
        lcd.setCursor(13,1);
        lcd.write(1);
      }
      }
    if(contador == 2){
      lcd.setCursor(0,1);
      lcd.print("                ");
      lcd.setCursor(0,1);
      lcd.print("Angulo Y: ");
      if(posicaoServoY<10){
        lcd.print(posicaoServoY);
```

```
    lcd.setCursor(11,1);
    lcd.write(1);
  }
  if(posicaoServoY>=10 and posicaoServoY < 100){
    lcd.print(posicaoServoY);
    lcd.setCursor(12,1);
    lcd.write(1);
  }
  if(posicaoServoY>=100){
    lcd.print(posicaoServoY);
    lcd.setCursor(13,1);
    lcd.write(1);
  }
  }
 if(contador == 3){
  lcd.setCursor(0,1);
  lcd.print("                ");
  lcd.setCursor(0,1);
  lcd.print("Tensao: ");
  if(tensaoMedida < 10){
    lcd.print(tensaoMedida,2);
    lcd.setCursor(12,1);
    lcd.print("V");
    }
  if(tensaoMedida >= 10){
    lcd.print(tensaoMedida,2);
    lcd.setCursor(13,1);
    lcd.print("V");
    }
 }
 if(contador == 4){
  lcd.setCursor(0,1);
  lcd.print("                ");
  lcd.setCursor(0,1);
  lcd.print("Corrente: ");
  if(currentDraw < 10){
    lcd.print(currentDraw,3);
    lcd.setCursor(15,1);
    lcd.print("A");
    }
```

```
      if(currentDraw >= 10){
        lcd.print(currentDraw,3);
        lcd.setCursor(16,1);
        lcd.print("A");
        }
    }
    if(contador == 5){
      lcd.setCursor(0,1);
      lcd.print("                ");
      lcd.setCursor(0,1);
      lcd.print("Umidade: ");
      lcd.print(DHT.humidity);
      lcd.print("%");
      }
    if(contador == 6){
      lcd.setCursor(0,1);
      lcd.print("                ");
      lcd.setCursor(0,1);
      lcd.print("Temperatura:");
      lcd.print(DHT.temperature, 0);
      lcd.setCursor(14,1);
      lcd.write(1);
      lcd.print("C");
      }
    if(contador == 7){
      lcd.setCursor(0,1);
      lcd.print("                ");
      lcd.setCursor(0,1);
      lcd.print("Indice UV: ");
      lcd.print(indiceUV);
      }
//--------------------------------------------------------------------------
  }
}
  delay(50);
//--------------------------------------------------------------------------
//MODO CONTROLE PADRÃO
if(web == 2){
  if(digitalRead(botaoVerde) == HIGH){
    modo = 1;
```

```
  }
 if(modo == 1){
  digitalWrite(LedVerdeManual, HIGH);
  digitalWrite(LedAzulAutomatico, LOW);
  digitalWrite(LedVerdeManualWEB, LOW);
  digitalWrite(LedAzulAutomaticoWEB, LOW);
  contadorX = analogRead(controleX); //-> VARIAVEL QUE IRÁ RECEBER OS DADOSANALOGICOS DO POTENCIOMETRO X
contadorY = analogRead(controleY); //-> VARIAVEL QUE IRÁ RECEBER OS DADOS ANALOGICOS DO POTENCIOMETRO Y
  //MOVIMENTO EM X
  if(contadorX > 550){ //-> CASO O POTENCIOMETRO X AUMENTA O VALOR ACIMA DE 550 O SERVO X IRÁ SE MOVIMENTO PARA A DIREITA
    //movimentoX = movimentoX + 1;
    posicaoServoX = posicaoServoX + 5;
    ServoX.write(posicaoServoX);
  }
  if(contadorX < 490){ //-> CASO O POTENCIOMETRO X DIMINUA O VALOR ABAIXO DE 490 O SERVO X IRÁ SE MOVIMENTAR PARA A ESQUERDA
    //movimentoX = movimentoX - 1;
    posicaoServoX = posicaoServoX - 5;
    ServoX.write(posicaoServoX);
  }
  if(posicaoServoX >= 180){ //-> LIMITA O SERVO EM 180º
    ServoX.write(180);
  }
  if(posicaoServoX <= 0){ //-> LIMITA O SERVO EM 0º
    ServoX.write(0);
  }
  //MOVIMENTO EM Y
  if(contadorY > 550){ //-> CASO O POTENCIOMETRO Y AUMENTA O VALOR ACIMA DE 550 OSERVO Y IRÁ SE MOVIMENTAR PARA BAIXO
    posicaoServoY = posicaoServoY - 5;
    ServoY.write(posicaoServoY);
  }
  if(contadorY < 490){ //-> CASO O POTENCIOMETRO Y DIMINUA O VALOR ABAIXO DE 490 OSERVO Y IRÁ SE MOVIMENTAR PARA CIMA
    //movimentoY = movimentoY + 1;
    posicaoServoY = posicaoServoY + 5;
    ServoY.write(posicaoServoY);
```

```
  }
  if(posicaoServoY >= 180){ //-> LIMITA O SERVO EM 180º
    ServoY.write(180);
  }
  if(posicaoServoY <= 0){ //-> LIMITA O SERVO EM 0º
    ServoY.write(0);
  }
  delay(10);

  leituraSensor = analogRead(pinoSensor); //FAZ A LEITURA DO PINO ANALÓGICO E ARMAZENA NA VARIÁVEL O VALOR LIDO
  tensaoEntrada = (leituraSensor * 5.0) / 1024.0; //VARIÁVEL RECEBE O RESULTADO DO CÁLCULO
  tensaoMedida = tensaoEntrada / (valorR2/(valorR1+valorR2)); //VARIÁVEL RECEBE O VALOR DE TENSÃO DC MEDIDA PELO SENSOR
 emon1.calcVI(17,100); //FUNÇÃO DE CÁLCULO (17 SEMICICLOS / TEMPO LIMITE PARA FAZER A MEDIÇÃO)
  double currentDraw = emon1.Irms; //VARIÁVEL RECEBE O VALOR DE CORRENTE RMS OBTIDO
  currentDraw = currentDraw-ruido; //VARIÁVEL RECEBE O VALOR RESULTANTE DA CORRENTE RMS MENOS O RUÍDO
  if(currentDraw < 0){ //SE O VALOR DA VARIÁVEL FOR MENOR QUE 0, FAZ
    currentDraw = 0; //VARIÁVEL RECEBE 0
  }
  //Leitura dos dados do sensor
  int leitura_porta = analogRead(A4);
  //De acordo com a leitura define o indice UV corrrspondente
  if (leitura_porta <= 10) {
    indiceUV = 0;
  } else if (leitura_porta > 10 && leitura_porta <= 46) {
    indiceUV = 1;
  } else if (leitura_porta > 46 && leitura_porta <= 65) {
    indiceUV = 2;
  } else if (leitura_porta > 65 && leitura_porta <= 83) {
    indiceUV = 3;
  } else if (leitura_porta > 83 && leitura_porta <= 103) {
    indiceUV = 4;
  } else if (leitura_porta > 103 && leitura_porta <= 124) {
    indiceUV = 5;
  } else if (leitura_porta > 124 && leitura_porta <= 142) {
```

```
    indiceUV = 6;
  } else if (leitura_porta > 142 && leitura_porta <= 162) {
    indiceUV = 7;
  } else if (leitura_porta > 162 && leitura_porta <= 180) {
    indiceUV = 8;
  } else if (leitura_porta > 180 && leitura_porta <= 200) {
    indiceUV = 9;
  } else if (leitura_porta > 200 && leitura_porta <= 221) {
    indiceUV = 10;
  } else {
    indiceUV = 11;
  }

  DHT.read11(pinoDHT11); //LÊ AS INFORMAÇÕES DO SENSOR

  if(digitalRead(botaoResultado) == HIGH){
    Serial.println("-----------------------------------------------------");
    Serial.print("Ângulo X: ");
    Serial.print(posicaoServoX);
    Serial.println(" º");
    Serial.print("Ângulo Y: ");
    Serial.print(posicaoServoY);
    Serial.println(" º");
    Serial.print("Tensão DC medida: ");
    Serial.print(tensaoMedida,2); //IMPRIME NA SERIAL O VALOR DE TENSÃO DC MEDIDA E
LIMITA O VALOR A 2 CASAS DECIMAIS
                Serial.println("V");
                Serial.print("Corrente medida: "); //IMPRIME O TEXTO NA SERIAL
                Serial.print(currentDraw,5); //IMPRIME NA SERIAL O VALOR DE CORRENTE MEDIDA
                Serial.println("A"); //IMPRIME O TEXTO NA SERIAL
                Serial.print("Valor porta: ");
                Serial.print(leitura_porta);
                Serial.print(" -> Indice: ");
                Serial.println(indiceUV);
                Serial.print("Umidade: "); //IMPRIME O TEXTO NA SERIAL
                Serial.print(DHT.humidity); //IMPRIME NA SERIAL O VALOR DE UMIDADE MEDIDO
                Serial.print("%"); //ESCREVE O TEXTO EM SEGUIDA
                Serial.print(" / Temperatura: "); //IMPRIME O TEXTO NA SERIAL
                Serial.print(DHT.temperature, 0); //IMPRIME NA SERIAL O VALOR DE UMIDADE
MEDIDO E REMOVE A PARTE DECIMAL
```

```
    Serial.println("*C"); //IMPRIME O TEXTO NA SERIAL
    Serial.println("-----------------------------------------------------");
    delay(3000);
  }
}

if(digitalRead(botaoAzul) == HIGH){
  modo = 2;
  }
if(modo == 2){
  digitalWrite(LedVerdeManual, LOW);
  digitalWrite(LedAzulAutomatico, HIGH);
  digitalWrite(LedVerdeManualWEB, LOW);
  digitalWrite(LedAzulAutomaticoWEB, LOW);
  int valorLDR1_1 = analogRead(pinLDR1);
  int valorLDR1_2 = analogRead(pinLDR1);
  int valorLDR1_3 = analogRead(pinLDR1);
  int valorLDR1 = (valorLDR1_1 + valorLDR1_2 + valorLDR1_3)/3;

  int valorLDR2_1 = analogRead(pinLDR2);
  int valorLDR2_2 = analogRead(pinLDR2);
  int valorLDR2_3 = analogRead(pinLDR2);
  int valorLDR2 = (valorLDR2_1 + valorLDR2_2 + valorLDR2_3)/3;

  int valorLDR3_1 = analogRead(pinLDR3);
  int valorLDR3_2 = analogRead(pinLDR3);
  int valorLDR3_3 = analogRead(pinLDR3);
  int valorLDR3 = (valorLDR3_1 + valorLDR3_2 + valorLDR3_3)/3;

  //MOVIMENTO EM Y
  if (valorLDR1 < valorLDR2){
    if(valorLDR1 < valorLDR3){
      menorY = valorLDR1;
      }else{
        menorY = valorLDR3;
        }
    }else{
      if(valorLDR2 < valorLDR3){
        menorY = valorLDR2;
        }else{
```

```
        menorY = valorLDR3;
        }
     }
  delay(100);
  if(menorY == valorLDR1){
   //servoY.write(20);
   PosicaoInicialAutomaticoY = PosicaoInicialAutomaticoY - 1;
   ServoY.write(PosicaoInicialAutomaticoY);
   }
  if(menorY == valorLDR2){
    ServoY.write(PosicaoInicialAutomaticoY);
   }
  if(menorY == valorLDR3){
   PosicaoInicialAutomaticoY = PosicaoInicialAutomaticoY + 1;
   ServoY.write(PosicaoInicialAutomaticoY);
    }
  if(PosicaoInicialAutomaticoY <= 0){
   PosicaoInicialAutomaticoY = PosicaoInicialAutomaticoY + 1;
   ServoY.write(PosicaoInicialAutomaticoY);
   }
  if(PosicaoInicialAutomaticoY >= 180){
   PosicaoInicialAutomaticoY = PosicaoInicialAutomaticoY - 1;
   ServoY.write(PosicaoInicialAutomaticoY);
   }
```

```
leituraSensor = analogRead(pinoSensor); //FAZ A LEITURA DO PINO ANALÓGICO E ARMAZENA NA VARIÁVEL O VALOR LIDO
tensaoEntrada = (leituraSensor * 5.0) / 1024.0; //VARIÁVEL RECEBE O RESULTADO DO CÁLCULO
tensaoMedida = tensaoEntrada / (valorR2/(valorR1+valorR2)); //VARIÁVEL RECEBE O VALOR DE TENSÃO DC MEDIDA PELO SENSOR

emon1.calcVI(17,100); //FUNÇÃO DE CÁLCULO (17 SEMICICLOS / TEMPO LIMITE PARA FAZER A MEDIÇÃO)
double currentDraw = emon1.Irms; //VARIÁVEL RECEBE O VALOR DE CORRENTE RMS OBTIDO
currentDraw = currentDraw-ruido; //VARIÁVEL RECEBE O VALOR RESULTANTE DA CORRENTE RMS MENOS O RUÍDO
```

```
if(currentDraw < 0){ //SE O VALOR DA VARIÁVEL FOR MENOR QUE 0, FAZ
  currentDraw = 0; //VARIÁVEL RECEBE 0
}

//Leitura dos dados do sensor
int leitura_porta = analogRead(A3);
//De acordo com a leitura define o indice UV corrrspondente
if (leitura_porta <= 10) {
  indiceUV = 0;
} else if (leitura_porta > 10 && leitura_porta <= 46) {
  indiceUV = 1;
} else if (leitura_porta > 46 && leitura_porta <= 65) {
  indiceUV = 2;
} else if (leitura_porta > 65 && leitura_porta <= 83) {
  indiceUV = 3;
} else if (leitura_porta > 83 && leitura_porta <= 103) {
  indiceUV = 4;
} else if (leitura_porta > 103 && leitura_porta <= 124) {
  indiceUV = 5;
} else if (leitura_porta > 124 && leitura_porta <= 142) {
  indiceUV = 6;
} else if (leitura_porta > 142 && leitura_porta <= 162) {
  indiceUV = 7;
} else if (leitura_porta > 162 && leitura_porta <= 180) {
  indiceUV = 8;
} else if (leitura_porta > 180 && leitura_porta <= 200) {
  indiceUV = 9;
} else if (leitura_porta > 200 && leitura_porta <= 221) {
  indiceUV = 10;
} else {
  indiceUV = 11;
}

DHT.read11(pinoDHT11); //LÊ AS INFORMAÇÕES DO SENSOR


if(digitalRead(botaoResultado) == HIGH){
  Serial.println("-----------------------------------------------------");
  Serial.print("Ângulo X: ");
  Serial.print(0);
```

```
                Serial.println(" º");
                Serial.print("Ângulo Y: ");
                Serial.print(PosicaoInicialAutomaticoY);
                Serial.println(" º");
                Serial.print("Tensão DC medida: ");
                Serial.print(tensaoMedida,2); //IMPRIME NA SERIAL O VALOR DE TENSÃO DC MEDIDA E LIMITA O VALOR A 2 CASAS DECIMAIS
                Serial.println("V");
                Serial.print("Corrente medida: "); //IMPRIME O TEXTO NA SERIAL
                Serial.print(currentDraw,5); //IMPRIME NA SERIAL O VALOR DE CORRENTE MEDIDA
                Serial.println("A"); //IMPRIME O TEXTO NA SERIAL
                Serial.print("Valor porta: ");
                Serial.print(leitura_porta);
                Serial.print(" -> Indice: ");
                Serial.println(indiceUV);
                Serial.print("Umidade: "); //IMPRIME O TEXTO NA SERIAL
                Serial.print(DHT.humidity); //IMPRIME NA SERIAL O VALOR DE UMIDADE MEDIDO
                Serial.print("%"); //ESCREVE O TEXTO EM SEGUIDA
                Serial.print(" / Temperatura: "); //IMPRIME O TEXTO NA SERIAL
                Serial.print(DHT.temperature, 0); //IMPRIME NA SERIAL O VALOR DE UMIDADE MEDIDO E REMOVE A PARTE DECIMAL
                Serial.println("*C"); //IMPRIME O TEXTO NA SERIAL
                Serial.println("---------------------------------------------------");
                delay(3000);
              }

            }
}
```