



MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

**DESENVOLVIMENTO DE UM SISTEMA DE COLETA DE DADOS AMBIENTAIS
BASEADO EM PROCESSADORES DA FAMÍLIA ATMEGA COM TRANSMISSÃO
DE DADOS VIA WI-FI.**

Maiara Vitória Santos Pereira

Relatório de Iniciação Científica do programa
PIBIC, orientado pelo Dr. Waldeir Amaral Vilela
e coorientado por Dr. Ricardo Toshiyuki Irita

INPE

São José dos Campos

2022



RESUMO

As energias renováveis têm despertado grande interesse tanto comercial quanto científico no Brasil e no mundo, entre as fontes alternativas de geração de energia elétrica, as usinas fotovoltaicas são uma das alternativas que mais vem crescendo devido principalmente ao baixo impacto ambiental e o seu custo que vem reduzindo significativamente nos últimos anos. Dentro deste contexto, o Grupo de Dispositivos Fotovoltaicos (GDF) do Instituto Nacional de Pesquisas Espaciais (INPE) vem realizando estudos que requerem a medição de parâmetros ambientais tais como pressão, temperatura, umidade e radiação solar. A partir da problematização apresentada, este projeto de Iniciação Científica tem como objetivo principal, desenvolver uma solução que seja de baixo custo de um sistema de aquisição de dados ambientais que será utilizada em experimentos que visam estudar o potencial solar fotovoltaico. Durante a fase de testes foi utilizado o Arduino Uno, que utiliza o microprocessador de chip único, Atmega328 que é um módulo específico, que possui somente os circuitos e funções necessárias para o projeto, como as tarefas de coleta, transmissão e armazenamento dos dados. Outros módulos foram adicionados para possibilitar a comparação com o sistema de aquisição de dados comercial da Campbell Scientific (CR1000), que é frequentemente utilizado nas plataformas de coleta de dados do INPE.

Palavras-chave: Aquisição de Dados, Energia Solar Fotovoltaica. Dados Ambientais Plataformas de coleta de dados. Atmega328. Arduino.



LISTA DE FIGURAS

	Pág.
Figura 1: PCD.....	10
Figura 2: Rede Wireless.....	11
Figura 3: 2P2	12
Figura 4: Atmega328p.....	15
Figura 5: Arduino UNO	16
Figura 6: Montagem do módulo leitor de cartão SD	20
Figura 7: Montagem do módulo Real Time Clock RTC DS3231.....	21
Figura 8: Exemplo do comando no Browser	22
Figura 9: Esp respondendo a comandos AT via Serial do Arduino.....	23
Figura 10: ESP01.....	23
Figura 11: Fluxograma do circuito de dados analógicos com transmissão via Esp-01.....	24
Figura 12: Montagem a partir da integração dos componentes	25
Figura 13: ESP8266 NODEMCU.....	26
Figura 14: Fluxograma do circuito de dados analógicos com transmissão via Esp-01.....	27
Figura 15: Montagem do NODEMCU e Potênciometro.....	27
Figura 16: Fluxograma representando a leitura, exibição e armazenamento de dados.....	28
Figura 17: Montagem conexão serial.....	29
Figura 18: Módulo ESP32.....	30
Figura 19: Frente da Placa de Circuito Impresso desenvolvida no EasyEDA.....	35
Figura 20: Verso da Placa de Circuito Impresso desenvolvida no EasyEDA.....	35
Figura 21: Frente da Placa de Circuito Impresso desenvolvida no EasyEDA em 3D.....	36
Figura 22: Verso da Placa de Circuito Impresso desenvolvida no EasyEDA em 3D.....	36
Figura 23: Frente da Placa Impressa.....	37



Figura 24: Verso da Placa Impressa.....	37
Figura 25: Frente da Placa de Circuito Impresso desenvolvida no EasyEDA.....	40
Figura 26: Verso da Placa de Circuito Impresso desenvolvida no EasyEDA.....	40
Figura 27: Frente da Placa de Circuito Impresso desenvolvida no EasyEDA em 3D.....	41
Figura 28: Verso da Placa de Circuito Impresso desenvolvida no EasyEDA em 3D.....	41



LISTA DE TABELAS

Tabela 1: Comparativo de protocolos Wi-fi.....	13
Tabela 2: Comparativo de Modelos ESP.....	18
Tabela 3: Componentes da Placa ESP-01.....	34
Tabela 4: Componentes da Placa ESP 32.....	39



LISTA DE ABREVIATURAS E SIGLAS

GDF - Grupo de Dispositivos Fotovoltaicos
GPIO - General Purpose Input/Output
HTTP - Hypertext Transfer Protocol
IDE - Integrated Development Environment
INPE - Instituto Nacional de Pesquisas Espaciais
IP - Internet Protocol
PCD - Plataforma de coleta de dados
RTC - Real Time Clock
TCP - Protocolo de controle de transmissão
WI-FI - Wireless Fidelity
Kbps - Kilobits por segundo
m - Metros
Mbps - Megabits por segundo
ms - Milisegundos
IOT - internet of things



SUMÁRIO

1. INTRODUÇÃO	7
1.1 Objetivos	8
1.2 Objetivos específicos	8
2. INTRODUÇÃO TEÓRICA	9
2.1 Plataforma de Coleta de Dados	9
2.2 Tecnologia de comunicação sem fio - Rede Wireless	10
2.2.1 Rede peer-to-peer (2P2)	11
2.2.2 Protocolo Wi-fi	12
2.3 Microcontroladores	13
2.4 Softwares Computacionais	14
2.4.1 EasyEDA	14
2.4.2 Arduino IDE	14
3. DESENVOLVIMENTO	15
3.1 Materiais e Métodos	15
3.1.1 Atmega328	15
3.1.2 Processadores ESP8266 e ESP-32	17
3.1.3 Placa de Circuito Impresso	19
3.2 Testes e análises	20
3.2.1 Módulo leitor cartão SD	20
3.2.2 Módulo RTC DS3231	21
3.2.3 ESP-01	22
3.2.3.1 Transmissão de Dados Analógicos com comandos AT	24
3.2.3.1 Transmissão de Dados Analógicos através de comandos em C++ e Javascript	25
3.2.4 NODEMCU	26
3.2.4.1 Transmissão de Dados Analógicos - SoftAP	26
3.2.4.2 Memória Flash - SPIFFS	28
3.2.4.3 Teste conexão serial Nodemcu e Arduino - potenciômetro	29
3.2.4 ESP32	30
3.2.4.1 Transmissão de Dados Analógicos através de comandos em C++ e Javascript	31
4. RESULTADOS E DISCUSSÕES	32
4.1.1 Módulo de Aquisição utilizando o ESP01	33
4.1.2 Módulo de Aquisição utilizando o ESP32	38
5. CONCLUSÃO	42



1. INTRODUÇÃO

As atividades humanas necessitam, desde sempre, de uma análise e monitoramento dos dados meteorológicos e ambientais que tem forte impacto nas atividades humanas, variando de acordo com a atividade desenvolvida e com as tecnologias utilizadas. Nos dias atuais, isso se intensifica, pois, diversos outros serviços, além das atividades agrícolas e dos fenômenos que interferem diretamente na saúde humana, precisamos avaliar constantemente os dados e condições atmosféricas para geração de energia elétrica.

Os serviços como os de aviação em geral dependem da atualização recorrente dos dados meteorológicos. A partir da pluralidade de exigências, diversas frentes de pesquisas também utilizam como fonte de estudo as medições dos principais parâmetros ambientais. O Grupo de Dispositivos Fotovoltaicos - GDF, grupo de pesquisa do Instituto Nacional de Pesquisas Espaciais – INPE, desenvolve sensores ambientais, estudos da radiação solar e também estuda os dispositivos de aproveitamento deste tipo de potencial energético. Para o desenvolvimento de suas pesquisas, o GDF utiliza dispositivos para coleta de dados ambientais e analisa constantemente aspectos do meio ambiente, como por exemplo a medição de incidência da radiação solar.

Os dados são coletados ao longo do tempo por Plataformas de Coleta de Dados (PCD) localizadas em São José dos Campos - SP e em Cachoeira Paulista - SP. Nestas rotinas de trabalho são observadas algumas dificuldades, como o difícil acesso às plataformas para retirada dos dados. Normalmente os dispositivos de coleta de dados de baixo custo utilizam cartões de memória (SD) que são responsáveis por armazenar os dados. Além da dificuldade de acesso, o fato de retirar o cartão diversas vezes pode acarretar desgastes mecânicos no dispositivo, causando mal contato e, conseqüentemente, mal funcionamento a longo prazo. Outro método é a conexão direta de um cabo entre o microcontrolador e o computador que nem sempre é possível. Nesse caso é necessário inserir um código compilado no microcomputador para que ele execute a leitura dos dados. Procedimentos como esses atendem os experimentos, porém, não são soluções práticas para o dia a dia.

O GDF visando encontrar soluções que atendam às suas necessidades de pesquisa que não tenha custos elevados, propôs um trabalho de Iniciação Científica para o desenvolvimento de módulos e conexões para a armazenagem e transmissão de dados em nuvem, aprimorando



o trabalho já em desenvolvimento por outros alunos até o ano de 2021.

1.1 Objetivos

O objetivo principal deste trabalho consistiu no aprimoramento de um sistema de coleta de dados ambientais através da utilização de dispositivos capazes de estabelecer conexão sem fio para o envio de dados ambientais gerados em uma PCD a um computador remoto, além de armazenar os dados na nuvem.

1.2 Objetivos específicos

1. Estudar e melhorar uma unidade de coleta de dados ambientais de baixo custo em desenvolvimento no GDF/GPDMP/COPDT/INPE;
2. Desenvolver modos de transmissão de dados via Wi-Fi para uso com a unidade de coleta de dados ambientais de baixo custo;
3. Fazer o projeto básico do coletor e transmissor de dados utilizando Atmega;
4. Teste de desempenho dos dispositivos analisados;
5. Comparação entre as vantagens e desvantagens de cada módulo;
6. Montagem de um protótipo;
7. Testes do protótipo em laboratório.
8. Desenvolvimento de placas de circuito impresso para cada módulo desenvolvido.



2. INTRODUÇÃO TEÓRICA

2.1 Plataforma de Coleta de Dados

As estações de coletas de dados ambientais, também denominadas PCDs (Plataformas de Coleta de Dados) são normalmente estruturas munidas de torres metálicas, sistemas de aquisição de dados automatizados, e diversos tipos de sensores capazes de ler as grandezas ambientais, que são variáveis de interesse. Essas variáveis consistem nos mais diversos parâmetros, dependendo do contexto em que os dados são utilizados, podendo ser desde velocidade e direção dos ventos, índice pluviométricos (chuva), umidade relativa do ar, temperatura, umidade e temperatura do solo, pressão atmosférica, irradiação solar global, difusa e direta até parâmetros mais sofisticados como os espectros da radiação

As estações de coleta de dados ambientais podem ser divididas em diversas categorias conforme a sua aplicação, solarimétricas, meteorológicas, de aplicação agrárias, terrestres ou marinhas. As estações comerciais, normalmente utilizadas na maioria dos casos, fazem a coleta de dados e transmissão automática destes dados para os centros de recepção e análise dos dados. Esta transmissão automática dos dados pode ser feita por telemetria celular, satélite, wifi ou cabo de ethernet.

As plataformas terrestres, que são o objeto de estudo neste trabalho, geralmente possuem uma torre, como pode ser visto na Figura 1, onde seus sensores são instalados e conectados a um dispositivo coletor de dados. Estas plataformas são instaladas em pontos estratégicos para melhor fidelidade dos dados, o que geralmente pode significar sua instalação em locais remotos ou de difícil acesso.

Figura 1: PCD

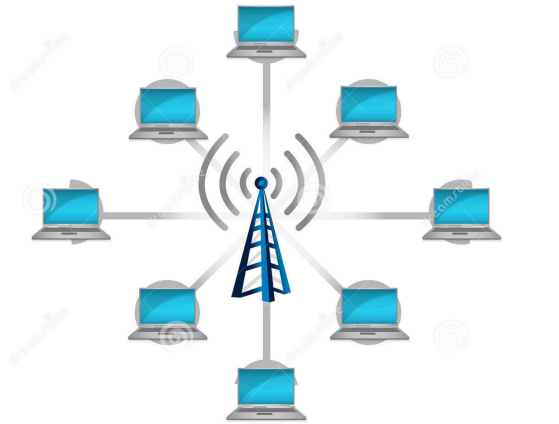


Fonte: GDF / [INPE](#)

2.2 Tecnologia de comunicação sem fio - Rede Wireless

A comunicação sem fio (wireless em inglês) consiste na transferência de dados/informações entre dispositivos eletrônicos sem a utilização de fios metálicos ou fibra óptica, como é representado na Figura 2. As distâncias envolvidas dependem a princípio do tipo de tecnologia utilizada na transmissão. No contexto mundial atual as comunicações sem fio são utilizadas em uma grande diversidade de dispositivos e áreas na rotina das pessoas, como o Wi-Fi, Bluetooth, GPS, o controle automático e até uma assistente pessoal digital, entre outras.

Figura 2: Rede Wireless

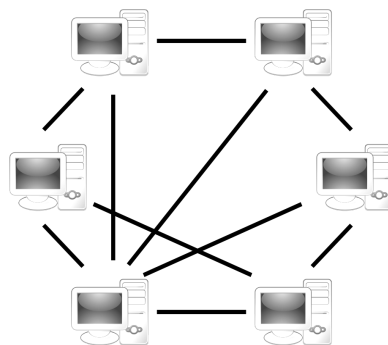


Fonte: Ilustração do conceito da conexão da rede wireless / [Dreamstime](#)

2.2.1 Rede peer-to-peer (2P2)

São consideradas as redes mais simples de serem entendidas, pois é uma arquitetura de redes em que cada nó coopera entre si para prover serviços um ao outro, sem a necessidade de um servidor central intermediando as conexões para troca de informações, sendo que todos os pares são clientes e servidores. Nesse modo de operação cada nó recebe uma denominação que varia de acordo com a sua função no momento, sendo emissores quando estão enviando dados, ou receptores quando estão recebendo dados. É importante pontuar essa topologia de rede pois na aquisição de dados de uma Plataforma de Coleta de Dados, a plataforma é considerada a emissora, já que coleta e envia os dados até os dispositivos externos que estão conectados a rede e são considerados os receptores, como é representado na Figura 3.

Figura 3: 2P2



Fonte: P2P / [Oficina da Net](#)

2.2.2 Protocolo Wi-fi

O Wi-fi é um conjunto de especificações técnicas para redes locais sem fio (WLAN — Wireless Local Area Network) baseado no padrão IEEE 802.11 e que será citado em muitos tópicos deste documento, é importante ressaltar que toda tecnologia Wi-Fi é Wireless, mas nem toda tecnologia Wireless é Wi-Fi mesmo que ambas as tecnologias são possíveis e proporcionadas por meio dos roteadores. A denominação "Wi-Fi" é uma abreviatura da expressão em inglês "Wireless Fidelity" e caracteriza uma rede que qualquer pessoa pode implementar para interconectar dispositivos que estejam próximos geograficamente, desde que os aparelhos sejam compatíveis com a tecnologia de transmissão de dados por meio de radiofrequência. Além de permitir que dispositivos se comuniquem dentro de uma rede wireless, a tecnologia Wi-Fi permite que os equipamentos conectados a ela tenham acesso à internet por meio de roteadores Wi-fi, sendo Modems internos ou externos. A partir dos dados métricos do Wi-fi é possível obter-se uma tabela comparativa dos diferentes protocolos Wi-fi e suas taxas de dados com a última atualização em 2021.



Tabela 1: Comparativo de protocolos Wi-fi

Protocolo	Frequência	Largura do canal	Taxa máxima de dados
802.11ax	2,4 ou 5 GHz	20, 40, 80, 160 MHz	2,4 Gbps
802.11ac wave2	5 GHz	20, 40, 80, 160 MHz	1,73 Gbps
802.11ac wave1	5 GHz	20, 40, 80 MHz	866,7 Mbps
802.11n	2,4 ou 5 GHz	20, 40 MHz	450 Mbps
802.11g	2,4 GHz	20 MHz	54 Mbps
802.11a	5 GHz	20 MHz	54 Mbps
802.11b	2,4 GHz	20 MHz	11 Mbps
Legacy 802.11	2,4 GHz	20 MHz	2 Mbps

Fonte: Diferentes protocolos Wi-Fi e taxas de dados / [Intel](#)

2.3 Microcontroladores

Os microcontroladores são elementos fundamentais no desenvolvimento desta pesquisa pois são o cérebro de dispositivos eletrônicos, seguindo a lógica dos processadores mas em uma escala de tamanho menor e também com uma capacidade de armazenamento e processamento reduzidas, os microcontroladores são indicados para prototipagem, a partir da facilidade no quesito programação, sendo versátil e prático devido o seu encapsulamento.

Geralmente os microcontroladores são acoplados a uma placa de circuito generalista que facilita o uso com diversas aplicações, entretanto, neste caso será desenvolvido uma placa



de circuito impresso para cada aplicação que irá depender dos módulos utilizados, atendendo apenas às necessidades observadas.

São diversas as plataformas de prototipagem baseadas em microcontroladores, a mais conhecida é o Arduino, um dispositivo open-source projetado em torno do Atmega328, cuja linguagem de programação é uma versão com poucas modificações do C++.

2.4 Softwares Computacionais

2.4.1 EasyEDA

A ferramenta EasyEda é um ecossistema online para desenvolvimento de circuitos com várias ferramentas, como a criação de esquemáticos e confecção de Placas de Circuito Impresso (PCB). O software foi escolhido para a criação de todas as Placas e Esquemáticos deste projeto, além disso, a confecção impressa das placas necessita de arquivos de fabricação Gerber, disponibilizado pela plataforma juntamente ao documento com as listas dos materiais específicos para o processo de soldagem.

2.4.2 Arduino IDE

A IDE do Arduino é um ambiente de desenvolvimento integrado de programações, onde o programador tem tudo que precisa para codificar sua placa, escrevendo seus códigos de maneira satisfatória, rápida e eficiente. Neste software o usuário tem acesso a várias funções, como por exemplo destaque de sintaxe, correção de erros, inclusão de bibliotecas, monitor serial e carregamento de código. Todas as codificações desse projeto foram desenvolvidas nesta plataforma, levando em consideração a facilidade e satisfação.

3. DESENVOLVIMENTO

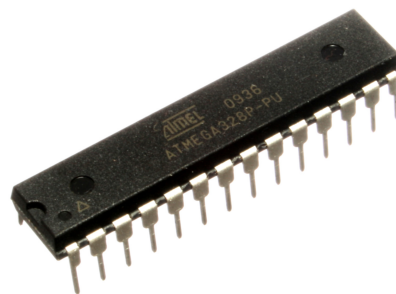
3.1 Materiais e Métodos

Durante o desenvolvimento do projeto foram utilizados alguns elementos com o objetivo de testar a melhor forma de transmitir os dados captados por rede sem fio, além disso, comparar com o desempenho do atual data logger utilizado, o CR-1000 da empresa Campbell Scientific. Os componentes foram submetidos a testes práticos para validar os valores nominais fornecidos pelos fabricantes, definindo qual componente se encaixaria melhor no projeto.

3.1.1 Atmega328

O microcontrolador faz parte da família lógica CMOS, baseando-se na arquitetura avançada AVR lançada pela ATMEL, o ATmega328 é comumente usado em muitos projetos e sistemas autônomos, os quais precisam de um microcontrolador simples, de baixa potência e baixo custo. O ATmega328 é utilizado durante os testes do projeto nas placas do Arduino UNO, com o objetivo de simplificar as diversas prototipagens. Além disso, é o cérebro das placas de circuito impresso dedicadas que serão citadas posteriormente, como é mostrado na Figura 4.

Figura 4: Atmega328p



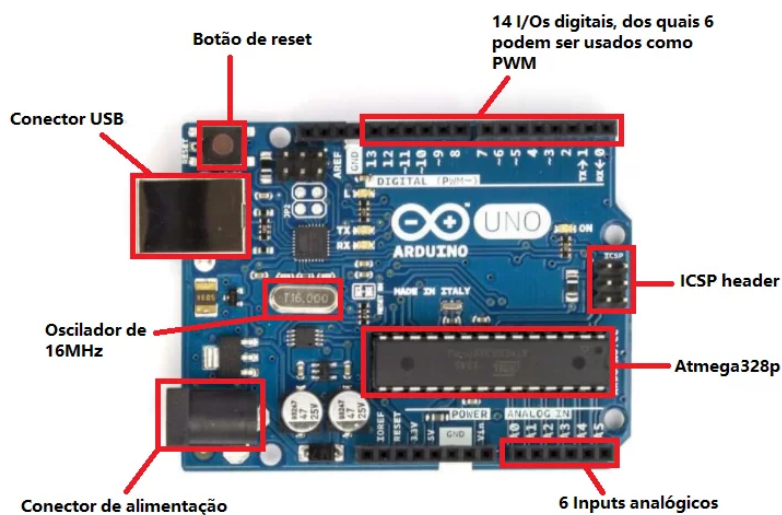
Fonte: Atmega328 / [Wikipedia](#)

3.1.1.1 Arduino UNO

O Arduino é uma plataforma de desenvolvimento de projetos eletrônicos, constituída tanto de hardware e software, possui diversos pontos positivos como o baixo custo de prototipagem, uma das motivações iniciais desse projeto de pesquisa, além disso, possui código aberto com fácil programação.

A placa possui diversos modelos, variando conforme a aplicação, o Arduino Uno, módulo exibido na Figura 5, foi o escolhido para representar o microcontrolador Atmega328 na fase de testes, pois têm o mesmo método de compilação de programação.

Figura 5: Arduino UNO



Fonte: Tipos de Arduino / [Blog Eletrogate](#)



3.1.2 Processadores ESP8266 e ESP-32

Os módulos projetados pela empresa especializada em IOT (Internet das Coisas) Espressif Systems se trata de plataformas com microcontroladores programáveis, como o Arduino, tendo o mesmo microcontrolador como base, o ESP8266 um procesador single-core que roda a 80MHz, variando apenas em seu tamanho e capacidade. O ESP-01 se trata de um dispositivo de tamanho reduzido, com duas entradas digitais para GPIO sendo geralmente utilizado em paralelo com um Arduino ou um ATmega328 através da comunicação serial. Já o NodeMCU se trata de um encapsulamento mais robusto do ESP8266 podendo até substituir o Arduino em seu uso, uma vez que o mesmo tem até mais pinos de GPIO do que alguns dispositivos da família portadora do ATmega, contando com entradas analógicas e digitais. Já o ESP32, é um processador dual-core de 160MHz, quando comparado com os outros módulos é o que possui Wi-fi mais rápido e têm mais GPIOs, além disso, suporta Bluetooth 4.2, possui pinos sensíveis ao toque, um sensor de efeito hall e um sensor de temperatura embutidos.

Levando em consideração as especificações da fabricante, os módulos da família ESP8266 têm uma limitação para a transferência de pacotes via rede que se dá devido ao fato do tamanho máximo de pacote ser relacionado ao MTU da rede em que está conectado, que geralmente é de 1500 bytes. Tratando-se agora de utilidade, o ESP8266 possibilita ao dispositivo se conectar à uma rede Wi-Fi padrão e trocar informações com qualquer outro dispositivo conectado nessa mesma rede já que tem a capacidade de funcionar como um servidor *Web* que executa o protocolo que permite a troca de arquivos de texto entre dispositivos via Internet, o HTTP.

A partir das informações disponibilizadas pelo fabricante, é possível a elaboração de uma tabela comparativa entres os processadores analisados neste projeto, com informações relevantes nas aplicações desejadas e citadas.



Tabela 2: Comparativo de Modelos ESP

	ESP8266	ESP32
MCU	Xtensa Single-core 32-bit L106	Xtensa Dual-Core 32-bit LX6 com 600 DMIPS
Comunicação Serial	Não	Sim
Bluetooth	Não	4.2 e BLE (Bluetooth Low Energy)
Frequência Típica	80MHz	160MHz
Memória RAM	36KB	520KB
Memória ROM	64KB	448KB
Valor Médio	R\$31,00	R\$38,90

Fonte: O Autor



3.1.3 Placa de Circuito Impresso

As placas de circuito impresso são componentes essenciais no desenvolvimento desse projeto, pois permite integrar em um único dispositivo todos os circuitos desenvolvidos em apenas uma placa com funções dedicadas às necessidades da pesquisa. Cada processador estudado possui um módulo de aquisição projetado para atender as demandas específicas.

As placas foram desenvolvidas com a ferramenta EasyEDA e foram confeccionadas no laboratório de circuito impresso do INPE, utilizando o processo de usinagem mecânica, feito por fresadoras e ferramentas de transferência de imagem. Neste trabalho não foi realizado o processo de montagem dos componentes na placa devido à falta de tempo hábil, ficando esta atividade para trabalhos futuros.

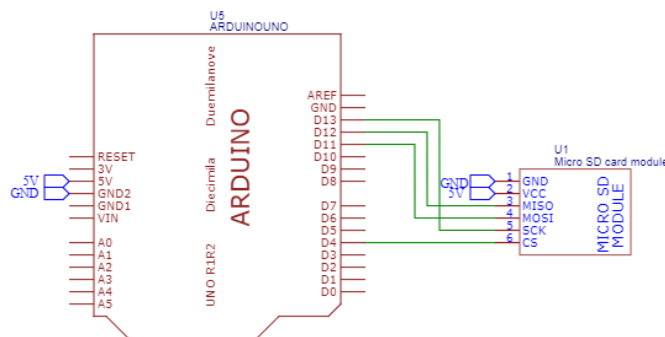
3.2 Testes e análises

Para a avaliação desse projeto e dos circuitos desenvolvidos foram feitos testes a fim de obter a melhor disposição dos componentes e circuitos na integração do dispositivo de coleta dos dados ambientais e o sistema de transmissão de dados via wifi.

3.2.1 Módulo leitor cartão SD

O módulo leitor cartão SD possui uma função importante, pois ele permite salvar em um micro cartão SD conectado ao módulo, os dados dos sensores quando o sistema não estiver conectado à alguma rede de internet. Nessa fase de testes, uma codificação foi desenvolvida com o objetivo de exibir na serial do arduino, os dados que já estavam salvos no cartão em um arquivo txt (Apêndice A), além disso, a montagem é exibida na Figura 6.

Figura 6: Montagem do módulo leitor de cartão SD.

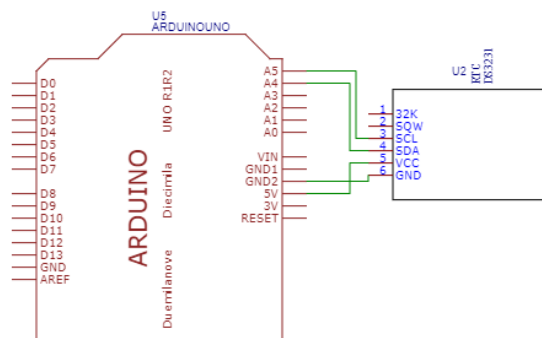


Fonte: O Autor

3.2.2 Módulo RTC DS3231

O módulo Real Time Clock RTC DS3231 tem como função essencial disponibilizar informações precisas de data e hora, possibilitando relacionar o dado com o horário que ele foi adquirido. Nesta fase de testes foi desenvolvida uma codificação para compreender o funcionamento do componente, e teve como objetivo principal exibir na serial do Arduino dados como ano, mês, dia, hora, minutos e segundos (Apêndice B), além disso, a montagem é exibida na Figura 7.

Figura 7: Montagem do módulo Real Time Clock RTC DS3231.

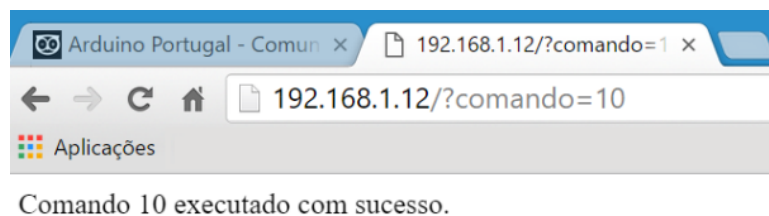


Fonte: O Autor.

3.2.3 ESP-01

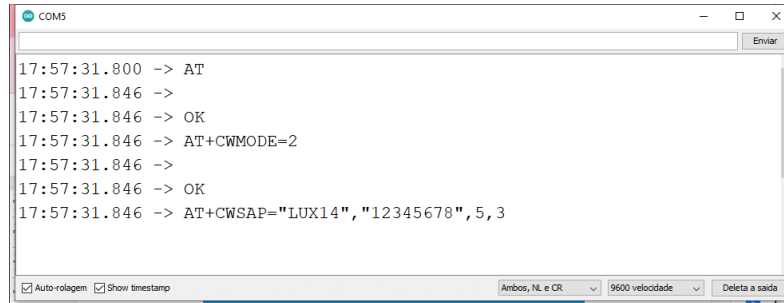
O ESP8266 é um módulo com microprocessador muito utilizado em projetos com Arduino para conectar-se à internet, sendo possível observá-lo na Figura 10. Ele é normalmente utilizado através de uma comunicação serial com uma placa da plataforma. Nesse projeto ele é utilizado para a transmissão dos dados coletados pelo Atmega328, enviando os dados para uma determinada página na internet. Nessa fase de testes foi desenvolvida uma codificação com o objetivo principal de controlar um led por comandos enviados através do browser com o IP do ESP, onde o comando 10 é responsável por acender e o comando 11 é responsável por apagar o led (Apêndice C). Na Figura 8 é mostrado como inserir o comando de teste no browser. Uma informação importante que deve ser considerada é o modo como o ESP8266 está operando, nesse projeto o componente está operando com o padrão de fábrica, respondendo assim a comandos AT, sendo necessário estar em conjunto com um microcontrolador, nesse projeto o Arduino, é responsável por enviar os comandos AT para o ESP via Serial como observado na Figura 9.

Figura 8: Exemplo do comando no Browser



Fonte:ESP8266 / [Arduino Portugal](#)

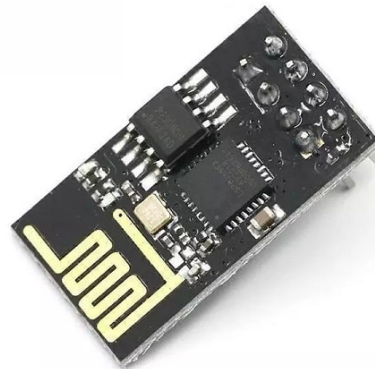
Figura 9: Esp respondendo a comandos AT via Serial do Arduino



```
COM5
17:57:31.800 -> AT
17:57:31.846 ->
17:57:31.846 -> OK
17:57:31.846 -> AT+CWMODE=2
17:57:31.846 ->
17:57:31.846 -> OK
17:57:31.846 -> AT+CWSAP="LUX14","12345678",5,3
```

Fonte: O Autor.

Figura 10: ESP01



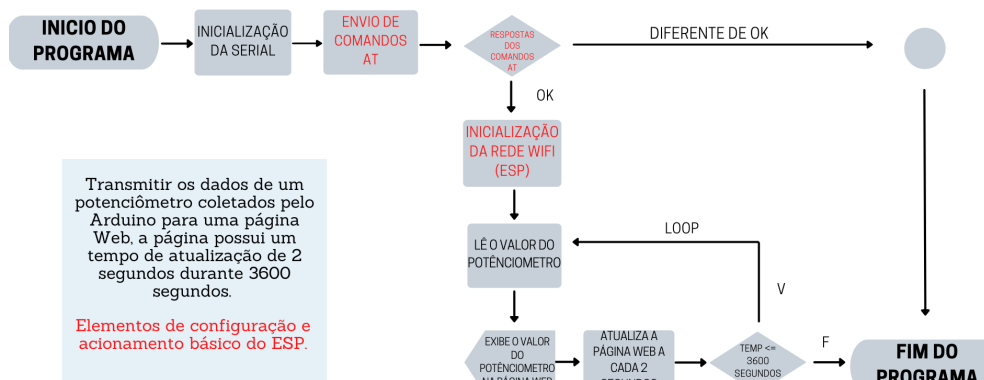
Fonte: Módulo WiFi ESP8266 ESP-01 / [Filipe Flop](#)

3.2.3.1 Transmissão de Dados Analógicos com comandos AT

Para complementar a análise funcional do dispositivo desenvolvido também foi testado a transmissão de dados analógicos, utilizando a mesma técnica abordado no tópico anterior, uma codificação foi desenvolvida com o objetivo principal de transmitir os dados coletados pelo Arduino de um potenciômetro para uma página Web, a página possui um tempo de atualização de dois segundos, representando assim a coleta dos dados analógicos dos sensores da PCD com transmissão via Wifi (Apêndice D).

Após a inicialização dos módulos através do cabo USB conectado ao CPU do computador, ocorre a transferência dos comandos AT a partir da comunicação serial entre os dois módulos com uma frequência pré-definida. Logo em seguida, obtém-se a resposta dos comandos, se caso for *OK* inicia-se uma rede WI-FI com as especificações citadas nas primeiras linhas da programação, conseqüentemente o sistema entra em um loop para a leitura e exibição do valor do potenciômetro em uma página WWW com o tempo de atualização de dois segundos durante 3600 segundos, se o valor apresentado na resposta dos comandos AT for diferente de *OK* o programa é automaticamente finalizado. O resumo do código na forma de fluxograma é apresentado na Figura 11.

Figura 11: Fluxograma do circuito de dados analógicos com transmissão via Esp-01

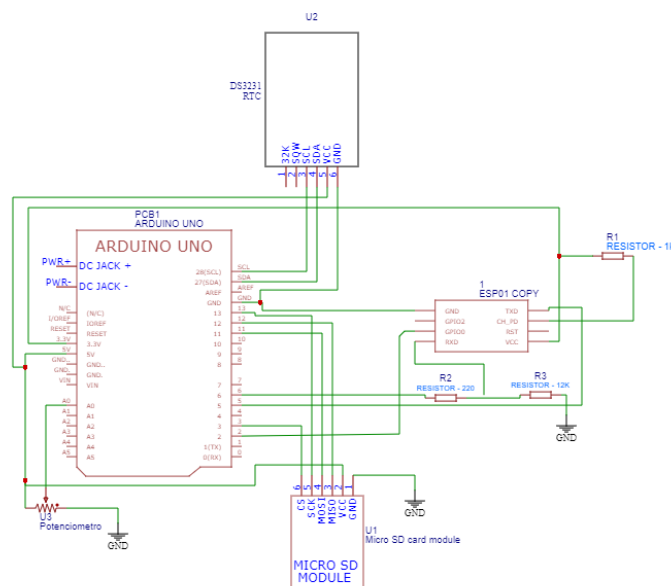


Fonte: O Autor.

3.2.3.1 Transmissão de Dados Analógicos através de comandos em C++ e Javascript

A partir dos dados que comprovam que quando o ESP8266 está operacional através do comando AT, há um aumento na utilização da memória volátil e da memória de programa por conta das instalações necessárias para atualização de firmware. Para analisar a flexibilidade do módulo através da codificação feita em uma linguagem de programação comum, foram escolhidas duas linguagens de programação em conjunto com o HTML, uma linguagem de marcação utilizada na construção de páginas na Web. Como é possível observar na Figura 12, não há diferenças no módulo de aquisição dos dados, apenas na codificação responsável pela integração dos componentes, neste teste há um código em C++ para programação do Arduino UNO e um código em C++, javascript e HTML para o ESP8266, possibilitando a navegação por botões ao invés de utilizar o browser (Apêndice E).

Figura 12: Montagem a partir da integração dos componentes



Fonte: O Autor.

3.2.4 NODEMCU

Nessa fase de testes, foram analisadas as propriedades do módulo ESP8266 NODEMCU, exibido na Figura 13, considerando que este dispositivo possui memória interna e portas analógicas que possibilitam a utilização sem um outro microcontrolador.

Figura 13: ESP8266 NODEMCU



Fonte: Módulo WiFi ESP8266 NodeMcu ESP-12E / [Smart Kits](#)

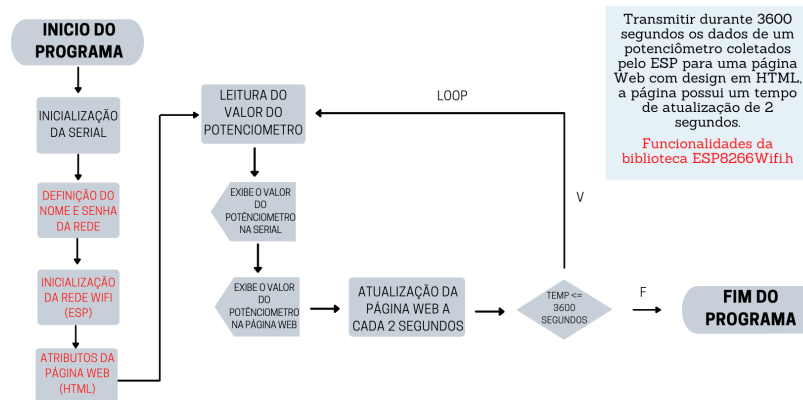
3.2.4.1 Transmissão de Dados Analógicos - SoftAP

Com o objetivo de testar a coleta e transmissão de dados analógicos com o módulo estudado, foi desenvolvida uma codificação que capta com precisão o valor analógico do potenciômetro e a partir das funcionalidades da biblioteca *ESP8266Wifi.h* o valor é exibido na página Web criada pelo componente (Apêndice F). A página possui um tempo de atualização pré-determinado de dois segundos, representando a coleta dos dados analógicos dos sensores da PCD com transmissão via Wifi. Além disso, é exibido na Figura 14 um Fluxograma responsável por representar os processos deste circuito, destacando em vermelho as funcionalidades da principal biblioteca utilizada, já na Figura 15 é exibido o esquemático de montagem para aquisição dos dados.

Após a inicialização do módulo através de uma fonte qualquer de tensão, nesse caso o cabo USB conectado ao CPU do computador utilizado, ocorre a inicialização da serial. Logo

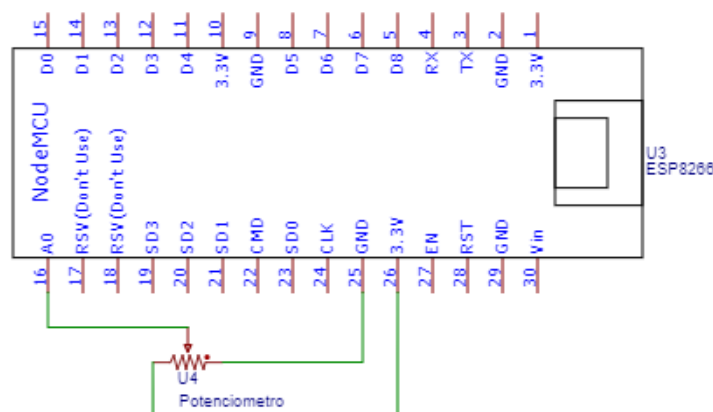
em seguida, há o Upload da codificação no processador do módulo, carregando as definições para a criação da rede WI-FI em C++ e Javascript, e os atributos para o design da página em HTML. O sistema entra em um loop para a leitura e exibição do valor do potenciômetro na serial, e através de funções específicas o módulo faz a leitura da serial e exibe o valor apresentado na página com o tempo de atualização de 2 segundos durante 3600 segundos, se o tempo passar do valor pré-definido o programa é automaticamente finalizado.

Figura 14: Fluxograma do circuito de dados analógicos com transmissão via Esp-01



Fonte: O Autor.

Figura 15: Montagem do NODEMCU e Potenciômetro



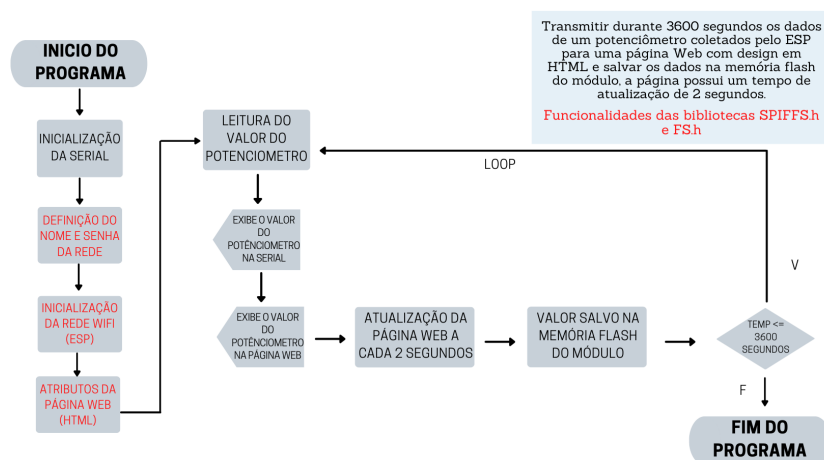
Fonte: O Autor.

3.2.4.2 Memória Flash - SPIFFS

A partir das especificações técnicas do NODEMCU, disponibilizadas pela fabricante, é possível observar que a memória flash do ESP é de 32MB, tem um sistema de arquivos denominado SPIFFS que permite que o usuário acesse de forma simples e limitada a memória flash do dispositivo mesmo após uma queda de energia. Nos outros testes, os dados foram armazenados em um módulo SD, entretanto, nesse o objetivo é desenvolver uma codificação que leia os dados, exiba na página web e armazene no próprio componente (Apêndice G). É importante ressaltar que o SPIFFS não suporta diretórios, logo, tudo deve ser salvo em um arquivo único e plano. Na Figura 9, é exibido um fluxograma para representar os processos deste circuito, destacando em vermelho as funcionalidades das duas bibliotecas utilizadas neste teste, *SPIFFS.h* e *FS.h*, ambas nativas do pacote ESP32.

Após a inicialização do módulo através do cabo USB conectado ao CPU do computador, ocorre a inicialização da serial. Logo em seguida, há o upload da codificação no processador, carregando as definições para a criação da rede WI-FI em C++ e Javascript, e os atributos para o design da página em HTML. O sistema entra em um loop para a leitura e exibição do valor do potenciômetro na serial, e através de funções específicas o módulo faz a leitura da serial, exibe o valor apresentado na página com o tempo de atualização de dois segundos durante 3600 segundos, além disso, o valor é salvo na memória flash do ESP em forma de TXT, se o tempo passar do valor pré-definido o programa é automaticamente finalizado. O resumo do código na forma de fluxograma é apresentado na Figura 16.

Figura 16: Fluxograma representando a leitura, exibição e armazenamento de dados

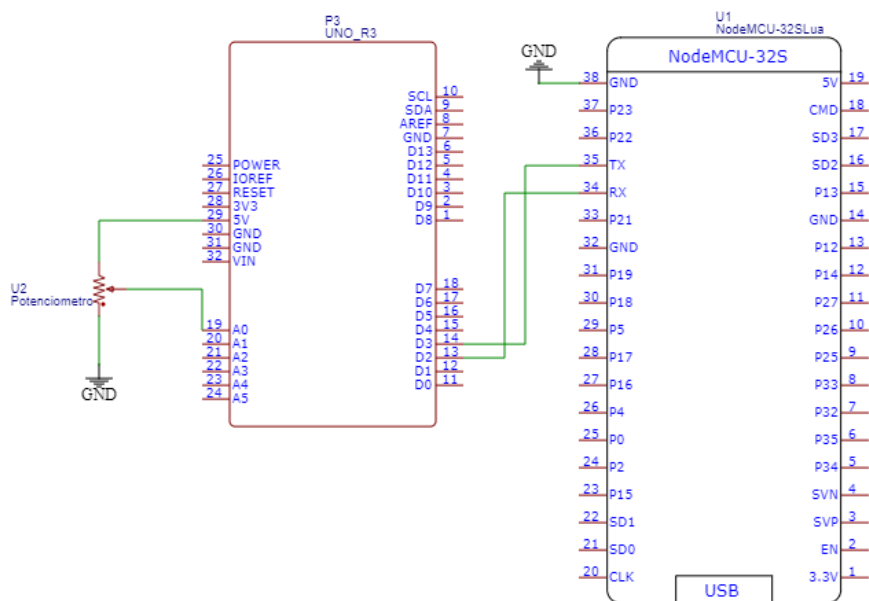


Fonte: O Autor.

3.2.4.3 Teste conexão serial Nodemcu e Arduino - potenciômetro

Com o objetivo de integrar o módulo ESP8266 Nodemcu com o Arduino para a leitura de um valor analógico, foram desenvolvidas duas codificações, uma para cada componente (Apêndice H). Inicialmente, o Arduino é responsável por ler o valor do potenciômetro e exibi-lo na serial com uma frequência pré-definida, paralelamente, o Nodemcu inicia a rede wi-fi com as especificações definidas na programação. Além disso, cria através dos parâmetros em HTML a página WEB, possibilitando que o valor transmitido pelo Arduino via serial seja lido pelo Nodemcu, configurando-o como uma variável que será exibida na página criada. É possível observar a montagem do circuito na figura 17.

Figura 17: Montagem conexão serial



Fonte: O Autor.

3.2.4 ESP32

O ESP32 é uma das variações dos módulos da família ESP, diferente do processador ESP8266, o chip principal do componente é o ESP-WROOM-32. A placa possui 34 portas GPIOs e 18 canais ADC, que fazem a conversão analógica-digital, como é mostrado na Figura 18. Algumas observações importantes sobre esse dispositivo são o baixo consumo de energia e o alto desempenho de potência, diferenciando dos demais. Além disso, o modelo possui uma função interessante que é a inclusão do Bluetooth Low Energy e Bluetooth Classic. Assim como os outros módulos, esta placa também pode ser programada em LUA ou pela IDE do Arduino. Durante a gravação na IDE do Arduino, o botão BOOT deve estar pressionado pelo programador para que ela comece a codificada.

Figura 18: Módulo ESP32



Fonte: [Mouser Eletronics](#)



3.2.4.1 Transmissão de Dados Analógicos através de comandos em C++ e Javascript

Com o objetivo de testar a coleta e transmissão de dados analógicos com o módulo analisado, foi desenvolvida uma codificação (Apêndice I) que capta com precisão o valor do potenciômetro e a partir das funções determinadas na codificação, o valor é salvo no Módulo SD CARD em conjunto com as informações disponibilizadas pelo RTC DS3231. É importante ressaltar que nesse teste não foi utilizado o microcontrolador Arduino UNO pois apenas o ESP32 em conjunto com os outros dois módulos citados neste tópico realizam todas as funções necessárias com alto desempenho. Assim como nos outros testes, o valor é exibido na página Web criada pelo componente com atualização de 2 segundos, representando a coleta dos dados analógicos dos sensores da PCD com transmissão via Wifi.

A formatação da programação nesse teste é um pouco diferente das observadas anteriormente pois segue a linha de codificação em blocos, sendo representada pelo documento *INDEX*, seguindo com as mesmas linguagens de programação utilizadas nesse projeto de pesquisa, C++, JavaScript e uma Linguagem de Marcação de HiperTexto, o HTML, o bloco de construção mais básico da web.



4. RESULTADOS E DISCUSSÕES

4.1 Desenvolvimento dos módulos de aquisição

Com o auxílio da plataforma anteriormente citada, EasyEDA, foram desenvolvidas as duas placas dedicadas, tornando a aquisição de dados mais fácil e diminuindo o custo final. Além das placas, todos os circuitos foram desenhados e projetados na mesma plataforma, possibilitando transformar automaticamente o esquemático em uma PCB, levando em consideração que o LAYOUT foi criado com todas as propriedades exatas dos componentes que seriam soldados na placa física.

Outro fator importante e que deve ser levado em consideração, foi a utilização dos serviços e equipamentos disponibilizados nos laboratórios do INPE, especificamente no Laboratório de Fabricação de Circuito Impresso. Infelizmente, só foi possível concluir a etapa de impressão do Módulo de Aquisição baseado no ESP01 devido a problemas relacionados ao tempo.

A finalização dessas placas é a resultante de todo o projeto de pesquisa, incluindo os estudos dos alunos anteriores.



4.1.1 Módulo de Aquisição utilizando o ESP01

Tendo como base, a placa desenvolvida na etapa anterior de pesquisa realizada pelo Aluno Andrew, foram integrados alguns componentes essenciais para a realização de todas as necessidades observadas. A placa dedicada utiliza o Atmega328p como núcleo central do sistema, além disso é importante ressaltar que o módulo escolhido para essa análise é o ESP01.

Outros componentes como o Micro SD CARD e o Real Time Clock DS3231 também foram implementados. A placa conta com barramento de pinos para a conexão dos sensores da Plataforma de Coleta de Dados, além de outros componentes que permitem a realização positiva das funções

Assim como foi testado anteriormente, a conexão entre os dispositivos da placa com os externos é feita através da rede Wi-Fi criada.

Os componentes foram testados separadamente na protoboard como é possível observar nos testes anteriormente citados, além disso, para testar o upload de codificações no processador Atmega foram compilados códigos simples como por exemplo o controle de leds.

A seguir é exibida uma tabela com todos os componentes necessários para a construção da placa dedicada, a lista é disponibilizada pela plataforma com todas as especificações exatas de todos os itens, como o valor da resistência dos resistores. Além disso, são exibidas imagens do Módulo de Aquisição utilizando o ESP01, duas delas desenvolvidas no EasyEDA (Figura 19 e Figura 20), duas delas no formato 3D na plataforma (Figura 21 e Figura 22) e as outras duas são as placas físicas impressas (Figura 23 e Figura 24).

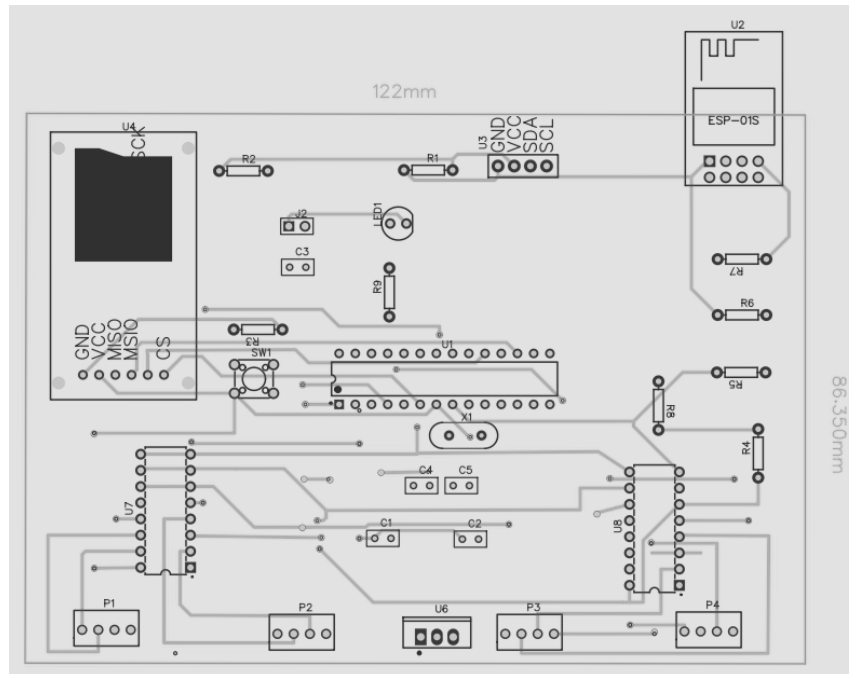


Tabela 3: Componentes da Placa ESP-01

ID	Nome	Quantidade
1	TO-220-3_L10.0-W4.5-P2.54-L	1
2	100n	3
3	450	1
4	320	1
5	350	2
6	1k	1
7	200	1
8	420	2
9	10k	1
10	ESP-01S	1
11	Button-6x6x6mm	1
12	ATMEGA328P-PU	1
13	SD MODULE	1
14	HDR-M-2.54_1x2	1
15	RTC	1
16	KF2510-4A	4
17	LED-TH-5mm_G	1
18	22p	2
19	CRYSTAL	1
20	PDIP-16_L19.7-W6.6-P2.54-LS7.8-BL	2

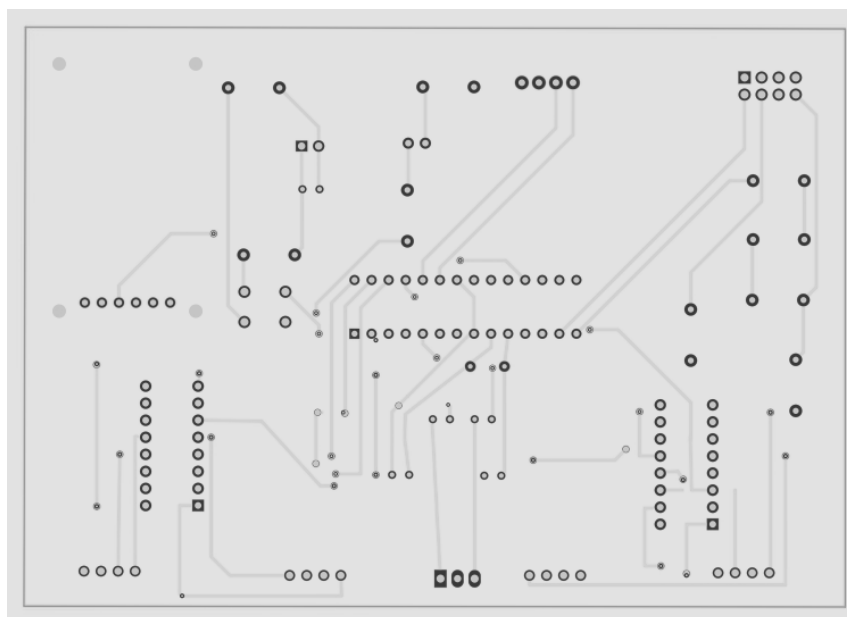
Fonte: O Autor.

Figura 19: Frente da Placa de Circuito Impresso desenvolvida no EasyEDA



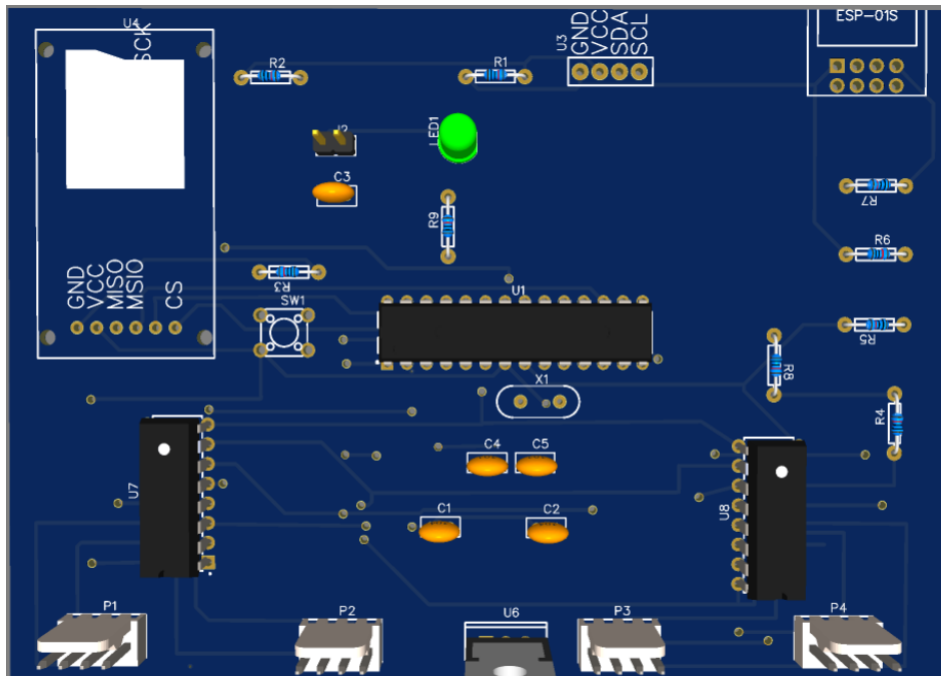
Fonte: O Autor

Figura 20: Verso da Placa de Circuito Impresso desenvolvida no EasyEDA



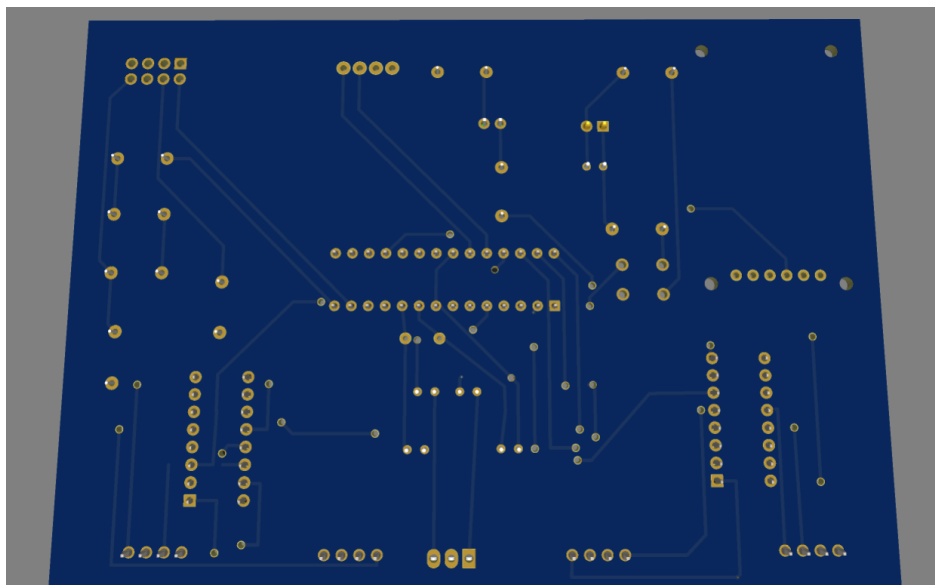
Fonte: O Autor.

Figura 21: Frente da Placa de Circuito Impresso desenvolvida no EasyEDA em 3D



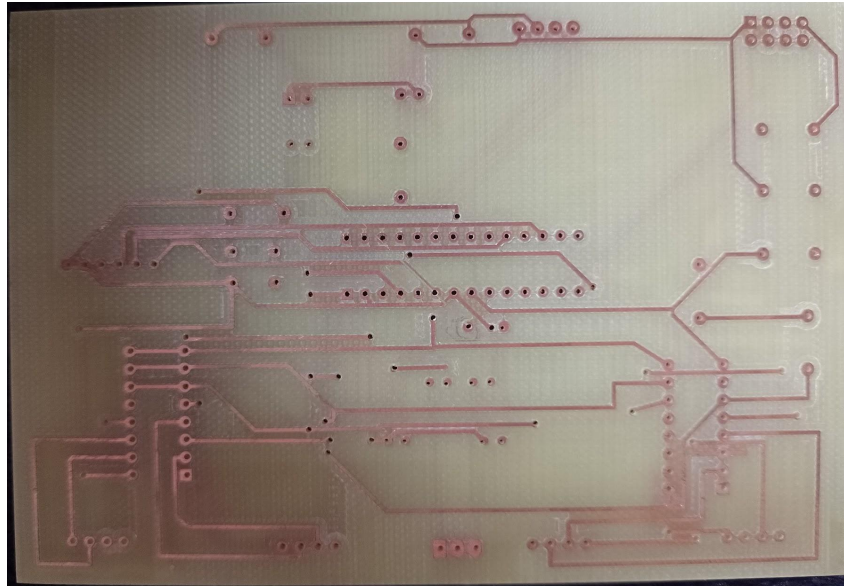
Fonte: O Autor.

Figura 22: Verso da Placa de Circuito Impresso desenvolvida no EasyEDA em 3D



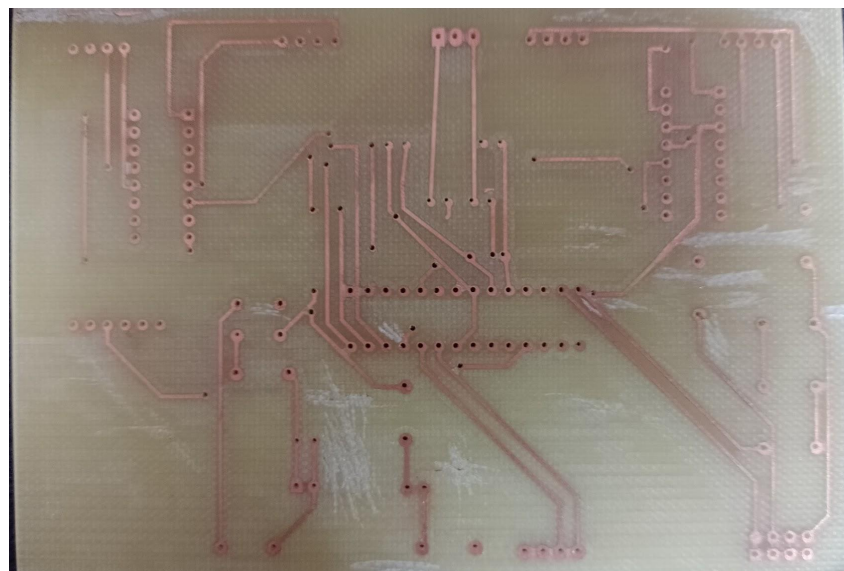
Fonte: O Autor.

Figura 23: Frente da Placa Impressa



Fonte: O Autor.

Figura 24: Verso da Placa Impressa



Fonte: O Autor.



4.1.2 Módulo de Aquisição utilizando o ESP32

O módulo de Aquisição para o ESP32 possui algumas diferenças em comparação com o anterior, pois o módulo analisado têm diversas funções e especificações que fazem com que o Atmega328p não seja necessário nessa aplicação. Neste caso, o núcleo é o processador Xtensa Dual-Core 32-bit LX6 com 600 DMIPS, que faz todas as funções anteriormente realizadas pelo Arduino/Atmega328p, os outros componentes também não são necessários, pois a capacidade do módulo ESP32 é satisfatória para o circuito.

A placa conta com barramento de pinos para a conexão dos sensores da Plataforma de Coleta de Dados, além de outros componentes que permitem a realização positiva das funções, a conexão entre os dispositivos da placa com os dispositivos externos é feita através da rede Wi-Fi criada. Os componentes foram testados na protoboard como é possível observar nos testes citados e analisados nos tópicos anteriores.

A seguir é exibida uma tabela com todos os componentes necessários para a construção da placa dedicada, a lista é disponibilizada pela plataforma com todas as especificações exatas de todos os itens. Além disso, são exibidas imagens do Módulo de Aquisição utilizando o ESP32, duas delas desenvolvidas no EasyEDA (Figura 25 e Figura 26) e duas delas no formato 3D na mesma plataforma (Figura 27 e Figura 28) .

Neste caso, não foi possível imprimir a placa devido a problemas relacionados ao tempo. Entretanto, mesmo com apenas a montagem virtual é possível comprovar a simplificação da placa em relação a montagem, no quesito programação dos módulos a diferença não é tão discrepante.

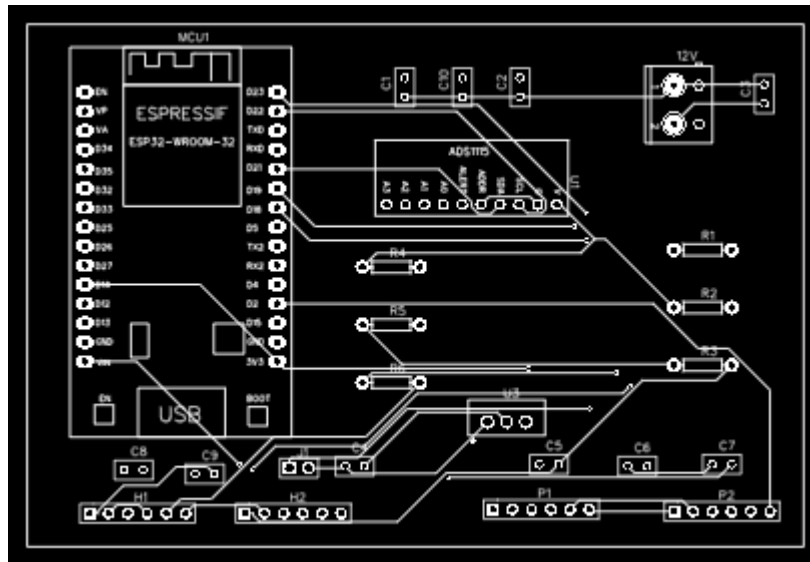


Tabela 4: Componentes da Placa ESP32

ID	Nome	Quantidade
1	Terminal	1
2	Pin	7
3	0.1u	3
4	Sensores	2
5	RTC3231	1
6	SDCARD	1
7	Vcc bat/usb	1
8	ESP32-WROOM-32-30Pin	1
9	10k	6
10	ADS1115 0x48	1
11	7805(TO-220)	1

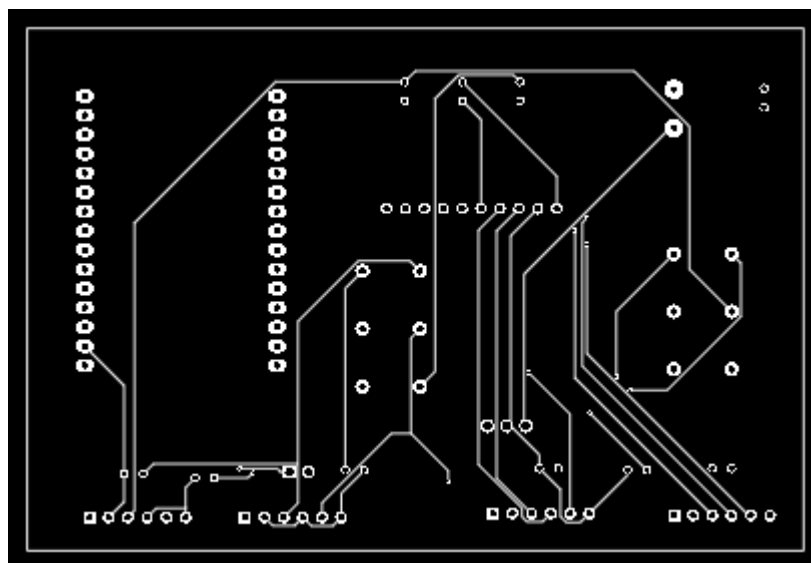
Fonte: O Autor.

Figura 25: Frente da Placa de Circuito Impresso desenvolvida no EasyEDA



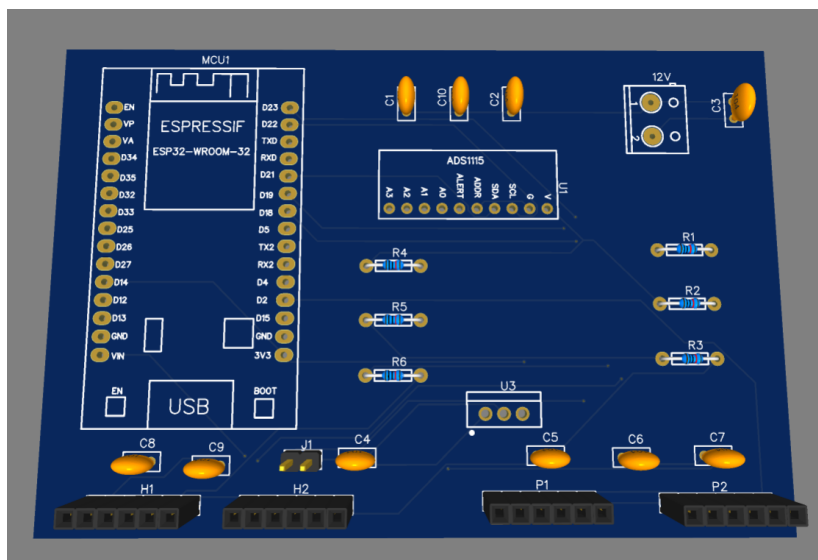
Fonte: O Autor.

Figura 26: Verso da Placa de Circuito Impresso desenvolvida no EasyEDA



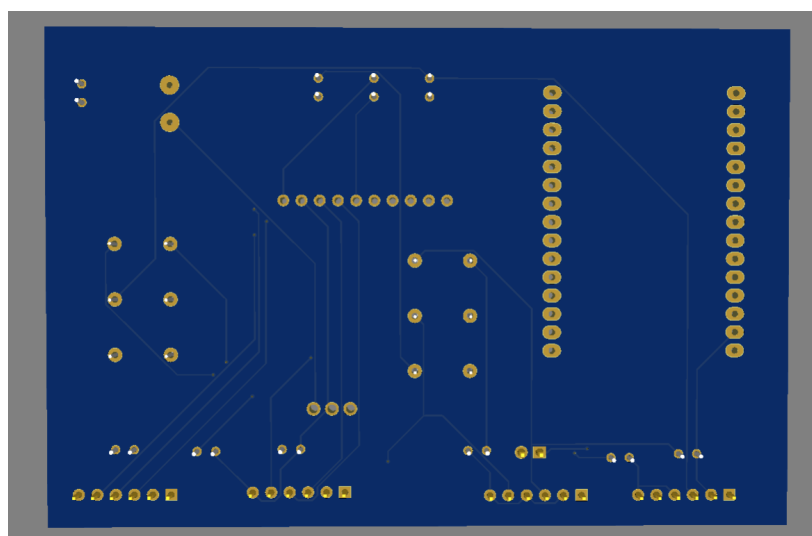
Fonte: O Autor.

Figura 27: Frente da Placa de Circuito Impresso desenvolvida no EasyEDA em 3D



Fonte: O Autor.

Figura 28: Verso da Placa de Circuito Impresso desenvolvida no EasyEDA em 3D



Fonte: O Autor.



5. CONCLUSÃO

Após os testes realizados em laboratório, concluiu-se que os módulos analisados atendem os requisitos necessários, e que substituem de forma satisfatória e com baixo custo o datalogger CR-1000 da empresa *Campbell Scientific*, apesar dessa etapa de testes não ter sido aplicada em campo devido à falta de tempo hábil.

Também foi possível comprovar que o módulo ESP01 é o mais adequado para essa aplicação, pois tem o menor custo comercial e atende todas as necessidades pré-determinadas. Além da facilidade de programação, ele possui o menor tamanho, necessitando de um pequeno espaço para ser instalado.

Considerando a importância dos dados coletados, é necessário otimizar e implementar técnicas de segurança de dados, a fim de evitar ataques cibernéticos de quaisquer tipos.

Como atividades futuras, é necessário finalizar o processo de integração dos circuitos desenvolvidos nas placas de circuito impresso e realizar os testes abordados nesse projeto de pesquisa. Para concluir o projeto, é importante a instalação dos sistema em campo em paralelo com um sistema comercial e comparar o seu comportamento por um período de tempo longo.



REFERÊNCIAS BIBLIOGRÁFICAS

ESPRESSIF SYSTEMS. ESP32 Series Datasheet. 2021. 65p. Disponível em: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf . Acesso em: 10 fev. 2022.

AI-THINKER CO. LTD. Datasheet ESP-01.v 2015 Disponível em: <https://pdf1.alldatasheet.com/datasheet-pdf/view/1179098/ETC2/ESP-01.html>. Acesso em 11 fev. 2022.

NodeMCU ESP8266 Disponível em:
<https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet> . Acesso em: 11 fev. 2022.

ESP32 LoRa WiFi SX1278 433MHZ de Longo Alcance com Display OLED e Bluetooth
Disponível em:
<https://www.usinainfo.com.br/lora/esp32-lora-wifi-sx1278-433mhz-de-longo-alcance-com-display-oled-e-bluetooth-5517.html>. Acesso em: 11 fev. 2022.

ARDUINO. Arduino Docs. 2021. Disponível em: <https://docs.arduino.cc/>. Acesso em: 14 fev. 2022.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS. Sistema de Coleta de Dados. 06 dez. 2019. Disponível em: http://www.cbers.inpe.br/sobre/coleta_dados.php. Acesso em 11 maio. 2022.



APÊNDICE A - TESTE MÓDULO CARTÃO SD

```
/* TESTE MÓDULO CARTÃO SD
 * Modificações: Maiara Vitória
 * NOME ARQUIVO: ESPINPE
 */

#include <SPI.h>
#include <SD.h>

const int chipSelect = 10;

void setup(){
  Serial.begin(9600);
  while(!Serial){
    ;
  }
  Serial.println("Inicializando o
cartão de memória...");

Serial.println("*****
*****");
  pinMode(SS, OUTPUT);

  if(!SD.begin(chipSelect)){
    Serial.println("Cartão de memória
falhou ou não está presente!");
    return;
  }
  Serial.println("Cartão de memória
inicializado com sucesso!");
  Serial.println("*****
*****");
  Serial.println("Mensagem do arquivo
de texto que está no cartão de
memória:");
  Serial.println();

  SDFile dataFile =
SD.open("ESPINPE.txt");

  if(dataFile){
    while(dataFile.available()){
      Serial.write(dataFile.read());
    }
    dataFile.close();
  }
  else{
    Serial.println("Erro ao abrir o
arquivo!");
  }
}

void loop(){
}
```

APÊNDICE B - TESTE MÓDULO RTC DS3231

```
RTC_DS3231 rtc;
char daysOfTheWeek[7][12] =
{"Domingo", "Segunda", "Terça",
"Quarta", "Quinta", "Sexta",
"Sábado"};

void setup(){
  Serial.begin(9600);
  if(! rtc.begin()) {
    Serial.println("*****
*****");
  }
}
```



```
Serial.println("DS3231 não encontrado");
while(1);
}
if(rtc.lostPower()){
Serial.println("DS3231 OK!");
//REMOVA O COMENTÁRIO DE UMA
DAS LINHAS ABAIXO PARA INSERIR AS
INFORMAÇÕES ATUALIZADAS EM SEU RTC

rtc.adjust(DateTime(F(__DATE__),
F(__TIME__)));
//rtc.adjust(DateTime(2022, 1,
19, 15, 00, 45));
}
delay(100);
}
```

```
void loop () {
DateTime now = rtc.now();
Serial.print("Data: ");
Serial.print(now.day(), DEC);
Serial.print('/');
Serial.print(now.month(),
DEC);
```

APÊNDICE C - TESTE ESP01

```
/* TESTE MÓDULO ESP01
*
* Modificações: Maiara Vitória
*
*/

#include <SoftwareSerial.h>
#define DEBUG true
#define ssid "LUX14" //nome da
rede
#define pass "12345678" //senha da
rede
#define debug true

SoftwareSerial myserial(2, 3);
//RX pino 2, TX pino 3

void setup()
{
```

```
Serial.print('/');
Serial.print(now.year(), DEC);
Serial.print(" / Dia: ");

Serial.print(daysOfTheWeek[now.day
OfTheWeek()]);
Serial.print(" / Horas: ");
Serial.print(now.hour(), DEC);
Serial.print(':');
Serial.print(now.minute(),
DEC);
Serial.print(':');
Serial.print(now.second(),
DEC);
Serial.println();
delay(1000);
}
```

```
Serial.begin(9600); //velocidade
de comunicacao por porta serie com
o PC por USB

myserial.begin(74880);
//velocidade de comunicacao por
porta serie com o ESP8266
pinMode(13, OUTPUT);

String response = "";
String cmd = "AT+CWSAP=\"";

myserial.write("AT\r\n");
delay(1000);
//myserial.println("AT+RST");
//delay(1000);
myserial.println("AT+CWMODE=2");
delay(1000);
cmd += ssid;
cmd += "\",\"";
cmd += pass;
cmd += "\",5,3\r\n";
```



```
myserial.println(cmd);
delay(3000);
myserial.println("AT+CIFSR");
delay(1000);
myserial.println("AT+CIPMUX=1");
delay(1000);

myserial.println("AT+CIPSERVER=1,80");
delay(1000);

while (myserial.available())
{
    char c = myserial.read(); //
    read the next character.
    //Serial.write (c);
    response += c;
}
Serial.println(response);

delay(3000);
}

void loop()
{
    // Verifica se o myserial esta
    enviando dados
    if (myserial.available())
    {
        if (myserial.find("+IPD, "))
        {
            delay(1000);

            unsigned int connectionId =
myserial.read() - 48;

            myserial.find("command=");
// advance cursor to "command="

            int command =
(myserial.read() - 48) * 10;

            command += (myserial.read()
- 48);

            Serial.print("recebido: ");

Serial.println(String(command));

            String content = "command "
+ String(command) + " executado
com sucesso.";

sendHTTPResponse(connectionId,
content);

            switch(command){
                case 10:
                    digitalWrite(13,HIGH);
                    break;
                case 11:
                    digitalWrite(13,LOW);
                    break;
            }
        }
    }
}

void sendHTTPResponse(unsigned int
connectionId, String content)
{
    // build HTTP response
    String httpHeader;
    // HTTP Header
    httpHeader = "HTTP/1.1 200
OK\r\nContent-Type: text/html;
charset=UTF-8\r\n";
    httpHeader += "Content-Length:
";
    httpHeader += content.length();
    httpHeader += "\r\n";
    httpHeader += "Connection:
close\r\n\r\n";
    httpHeader += content;
    httpHeader += " ";

    String cipSend = "AT+CIPSEND=";
    cipSend += connectionId;
    cipSend += ",";
    cipSend += httpHeader.length();
    //cipSend += "\r\n";
    ComandoAT(cipSend, 1000);
    ComandoAT(httpHeader, 1000);
}
```



```
}  
  
String ComandoAT(String command,          if (debug)  
const int timeout)                       {  
{                                         Serial.println(response);  
    String response = "";                }  
    myserial.println(command);           return response;  
  
    long int time = millis();            }  
  
    while ( (time + timeout) >  
millis())  
    {  
        while (myserial.available())  
        {  
            char c = myserial.read();  
            response += c;  
        }  
    }  
}
```

APÊNDICE D - TRANSMISSÃO DE DADOS ANALÓGICOS

APÊNDICE E - TRANSMISSÃO DE DADOS ANALÓGICOS ATRAVÉS DE COMANDOS EM C++ E JAVASCRIPT

E.1 - Unidade 1

```
#include <SoftwareSerial.h>              }  
                                          void loop()  
                                          {  
  
String str;                               const int analogpin = A0;  
SoftwareSerial Serial1(2, 3); //        int valor = 0;  
RX, TX                                    Serial.begin(115200);  
                                          valor = analogRead(analogpin);  
                                          Serial.println(valor);  
void setup(){                             str =String(valor);  
                                          Serial1.println(str);  
    Serial.begin(115200);                 delay(2000);  
    Serial1.begin(115200);                }  
  
    delay(2000);                          }  
                                          #define APSSID "ESPMAI"  
                                          #define APPSK "12345678"  
                                          #endif
```

E.2 - Unidade 12 /* MODIFICAÇÕES MAIARA VITÓRIA */

```
#include <SoftwareSerial.h>  
#include <ESP8266WiFi.h>  
#include <WiFiClient.h>  
#include <ESP8266WebServer.h>  
  
#ifndef APSSID  
const char *ssid = APSSID;  
const char *password = APPSK;  
  
//char valor ;
```




```
ESP8266WebServer server(80);

void handlebutton() {
    String myString;
    myString = Serial.readString();
    Serial.println(myString);
    delay(1000);

    String html;
    html = "<!DOCTYPE html>"

"<html>"
"<head>"

"<meta http-equiv='refresh'
content='5'>"

"<title> ESP8266</title>"

"</head>"

"<body>"

"<h1><center>TRANSMISSAO
WIFI!</center></h1>"

"<center><font size='5'>DADOS
AMBIENTAIS!</center>"

"<center><table
border='1'><center>"

"<colgroup>"

"<col span='3'
style='background-color:SkyBlue'>"

"</colgroup>"

"<tr>"

"<center><th VALOR DO
POTENCIOMETRO 1 </th></center>"

    "</tr>"
    "<tr>"
    "<td>"
    "<script type='text/javascript'>"
    //"const SerialPort =
    require('serialport')"
    //"const parsers =
    SerialPort.parsers"
    "document.write("+String(Serial.re
    adString()+")"
    "</script type='text/javascript'>"
    "</td>"
    "</table>"
    "</body>"
    "</html>";
    server.send(200, "text/html",
    html);
    }

void setup() {
    Serial.begin(115200);

    Serial.println();
    Serial.print("Configuring access
    point...\n");
    WiFi.softAP(ssid, password);
    IPAddress myIP =
    WiFi.softAPIP();
    Serial.print("AP IP address: ");
    Serial.println(myIP);

    server.on("/", handlebutton);

    server.begin();
    // iniciando server
```



```
Serial.println("Servidor HTTP
iniciado");

while (!Serial) {
    ; // wait for serial port to
connect. Needed for native USB
port only
}

}

void loop() {

    server.handleClient();

    /*if (Serial.available()) {
        valor = Serial.readString();
        //valor =
Serial.write(Serial.read());
        Serial.write(valor);
        delay(3000);
    } */

    delay(3000);
}

}
```

APÊNDICE F - TRANSMISSÃO DE DADOS ANALÓGICOS - SOFTAP

```
/* Create a WiFi access point and
provide a web server on it.
MODIFICAÇÕES MAIARA VITÓRIA
*/

#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>

#ifndef APSSID
#define APSSID "ESPMAI"
#define APPSK "12345678"
#endif

const char *ssid = APSSID;
const char *password = APPSK;

ESP8266WebServer server(80);

/* Just a little test message. Go
to http://192.168.4.1 in a web
browser
connected to this access point
to see it.
*/

void handlebutton() {
    String html;
    html = "HTTP/1.1 200
OK\r\nContent-Type: text/html;
charset=UTF-8\r\n";

    html = "<!DOCTYPE html>"
    "<html>"
    "<head><body>"
    "<title> ESP8266</title>"
    "<h1><center>TRANSMISSAO
WIFI!</center></h1>"
    "<center><font size='5'>DADOS
AMBIENTAIS!</center>"
    "<br>"
    "<center><button>MOSTRAR
DADOS</button></center>"
    "<br>"
    "<center><button>LER
DADOS</button></center>"
    "<br>"
    "<center><button>GRAVAR
DADOS</button></center>"
    "</body></html>";
    server.send(200, "text/html",
html);
}

/*void handleFrame() {
    String html;
    html = "HTTP/1.1 200
OK\r\nContent-Type: text/html;
charset=UTF-8\r\n";
    html = "<!DOCTYPE html>"
    "<html>"
    "<head><body>"
    "<title> ESP8266</title>"
    "<h1><center>TRANSMISSAO
WIFI!</center></h1>"
    "<center><font size='5'>DADOS
AMBIENTAIS!</center>"
    "<br>"
    "<center><button>MOSTRAR
DADOS</button></center>"
    "<br>"
    "<center><button>LER
DADOS</button></center>"
    "<br>"
    "<center><button>GRAVAR
DADOS</button></center>"
    "</body></html>";
    server.send(200, "text/html",
html);
}

*/
```



```
"<head><body>"
"<title>          ESP8266</title>"
"<h1><center>TRANSMISSAO
WIFI!</center></h1>"
"<center><font      size='5'>DADOS
AMBIENTAIS!</center>"
"<br>"
"<center><button>MOSTRAR
DADOS</button></center>"
"<br>"
"<center><button>LER
DADOS</button></center>"
"<br>"
"<center><button>GRAVAR
DADOS</button></center>"
"<br>"
"<br>"
"<table          border='1'>"
"<tr>"
"<th><center>      VALOR          DO
POTENCIOMETRO    </center></th>"
"</tr>"
"<tr>"
"<td>"
"valor          =      analogRead(A0)"
"cl.print(valor);          "
"</td>"
"</tr>"
"<table>"
"</body></html>";
    server.send(200, "text/html",
html);
}
*/

void frameDados(){
    const int analogpin = A0;
    int valor = 0;
    Serial.begin(115200);
    valor = analogRead(analogpin);
    Serial.println(valor);
    delay(100);

    String html;
        html = "HTTP/1.1 200
OK\r\nContent-Type:      text/html;
charset=UTF-8\r\n";
    html = "<!DOCTYPE html>"
```

```
"<html>"
"<head><body>"
"<title>          ESP8266</title>"
"<h1><center>TRANSMISSAO
WIFI!</center></h1>"
"<center><font      size='5'>DADOS
AMBIENTAIS!</center>"
"<br>"
"<center><button>MOSTRAR
DADOS</button></center>"
"<br>"
"<center><button>LER
DADOS</button></center>"
"<br>"
"<center><button>GRAVAR
DADOS</button></center>"
"<br>"
"<br>"
"<table          border='1'>"
"<tr>"
"<th><center>      VALOR          DO
POTENCIOMETRO    </center></th>"
"</tr>"
"<tr>"
"<td>"
//"valor          =      analogRead(A0)"
//"cl.print(valor)"
"</td>"
"</tr>"
"<table>"
"</body></html>";
    server.send(200, "text/html",
html);
}
```

```
void setup() {
    delay(1000);
    Serial.begin(115200);
    Serial.println();
    Serial.print("Configuring access
point...\n");
    /* You can remove the password
parameter if you want the AP to be
open. */
    WiFi.softAP(ssid, password);
```



```
        IPAddress      myIP      =          server.on("/1", frameDados);
WiFi.softAPIP();          server.begin();
Serial.print("AP IP address: ");      Serial.println("\n HTTP server
Serial.println(myIP);                started \n");
server.on("/", handlebutton);        }
server.begin();
Serial.println("\n HTTP server
started \n");
void loop() {
    server.handleClient();
}
```

APÊNDICE G - MEMÓRIA FLASH - SPIFFS

```
/*          File dataFile;          // Objeto
MODIFICAÇÕES MAIARA VITÓRIA      responsável por escrever/Ler do
10/03/2022                          cartão SD
*/
bool cartaoOk = true;
int a = analogRead(A0);
String ADCValue = String(a);

#include <SoftwareSerial.h>
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>

#include <SPI.h>          //
Biblioteca de comunicação SPI
#include <SD.h>          //
Biblioteca de comunicação com cartão
SD

#include "index.h" //Our HTML webpage
contents with javascripts

#define LED 2 //On board LED
#ifndef APSSID
#define APSSID "ESPMAI"
#define APPSK "12345678"
#endif

const char *ssid = APSSID;
const char *password = APPSK;
const int chipSelect = 4;

//char valor ;

ESP8266WebServer server(80);

void handlebutton() {

    String myString;

    myString = Serial.readString();

    Serial.println(myString);

    delay(1000);

}

void handleRoot() {
    String s = MAIN_page; //Read HTML
contents
    server.send(200, "text/html", s);
//Send web page
}

void handleADC() {
    int a = analogRead(A0);
    String ADCValue = String(a);

    server.send(200, "text/plane",
ADCValue); //Send ADC value only to
client ajax request
}
```



```
void handleLED() {
  String ledState = "OFF";
  String t_state =
server.arg("LEDstate"); //Refer
xhttp.open("GET",
"setLED?LEDstate="+led, true);
  Serial.println(t_state);
  if(t_state == "1")
  {
    digitalWrite(LED,LOW); //LED ON
    ledState = "ON"; //Feedback
parameter
  }
  else
  {
    digitalWrite(LED,HIGH); //LED OFF
    ledState = "OFF"; //Feedback
parameter
  }

  server.send(200, "text/plane",
ledState); //Send web page
}

void resetFile(){ // iniciar ou
reiniciar o arquivo
//create a new file and write data to
the file
  File f=SPIFFS.open(fname,"w");
  if(!f) { Serial.println("file
open failed"); }
  else {
    //Write data to file
    Serial.println("Escrevendo o
cabeçalho do arquivo");
    Serial.println("Gravando valores
de sensores a cada 5 segundos");
    f.print("Inicio do arquivo
sfile.txt \r");
    f.print("Valores de sensores a
cada 1s \n\r");
    f.print("data hora Sensor1
Sensor2 Sensor3 Sensor4 Sensor5\r");
    f.close();
  }
}
```

```
void getdata() // mostrar na pagina
html valores do arquivo contido no
arquivo fname
{
  File f=SPIFFS.open(fname,"r");
  int SDfileSz=f.size();
  Serial.print("SDfileSz: ");
Serial.println(SDfileSz);
  server.sendHeader("tamanho
conteudo", (String)(SDfileSz));
  server.sendHeader("controle de
conteudo", "max-age=2628000,
public");
  size_t fsizeSent =
server.streamFile(f,"text/plain");
  Serial.print("fsizeSent: ");
Serial.println(fsizeSent);
  f.close();
  delay(100);
}

void setup() {

  Serial.begin(115200);

  Serial.println();
  Serial.print("Configuring access
point...\n");
  WiFi.softAP(ssid, password);
  IPAddress myIP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(myIP);

  //Onboard LED port Direction output
pinMode(LED,OUTPUT);

  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.println(WiFi.softAPIP());

  server.on("/B", handlebutton);
  server.on("/R", handleRoot);
//Which routine to handle at root
location. This is display page
  server.on("/L", handleLED);
  server.on("/ADC", handleADC);
  server.on("/3", getdata);
```



```
server.on("/4", resetFile);

server.begin();
// iniciando server
Serial.println("Servidor HTTP
iniciado");

if (!SD.begin(chipSelect)) {
  Serial.println("Erro na leitura
do arquivo não existe um cartão SD ou
o módulo está conectado corretamente
?");
  cartaoOk = false;
  return;
}

if (cartaoOk){
  dataFile =
SD.open("datalog.csv", FILE_WRITE);
  Serial.println("Cartão SD
Inicializado para escrita :D ");
}

void loop(void) {

  server.handleClient();
//Handle client requests

  //Limpendo Variáveis
  String leitura = ""; // Limpo
campo contendo string que será
armazenada em arquivo CSV

  String ADCValue = "";

  leitura = String(millis()) + ";" +
String(ADCValue);

  if (dataFile) {
    dataFile.println(leitura); //
Escrevemos no arquivos e pulamos uma
linha
    dataFile.close(); //
Fechamos o arquivo
  }

  delay(3000); // Aguardamos 3
segundos para executar o loop
novamente
}

#include <SoftwareSerial.h>
String str;
SoftwareSerial Serial1(2, 3); // RX,
TX

void setup(){

  Serial.begin(115200);
  Serial1.begin(115200);

  delay(2000);

}

void loop()
{
  const int analogpin = A0;
  int valor = 0;
  Serial.begin(115200);
  valor = analogRead(analogpin);
  Serial.println(valor);

  str = String(valor);
  Serial1.println(str);
  delay(1000); }
}
```

APÊNDICE H - DADOS ANALÓGICOS NO MÓDULO SD

```
/*
* ESP8266 NodeMCU AJAX Demo
* Updates and Gets data from
webpage without page refresh
* https://circuits4you.com
```

```
*/
*
https://portal.vidadesilicio.com.br/modulo-cartao-micro-sd-nodemcu-d
atalogger/
```

adaptado por: MAIARA



```
data : 09-02-2022
*/

#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <SPI.h> //
Biblioteca de comunicação SPI
Nativa
#include <SD.h> //
Biblioteca de comunicação com
cartão SD Nativa

#include "index.h" //Our HTML
webpage contents with javascripts

#define LED 2 //On board LED

const char* ssid = "ESPMAI";
const char* password = "12345678";
const int chipSelect = 4;

ESP8266WebServer server(80);
//Server on port 80

File dataFile; //
Objeto responsável por
escrever/Ler do cartão SD
bool cartaoOk = true;
int a = analogRead(A0);
String ADCValue = String(a);

void handleRoot() {
  String s = MAIN_page; //Read HTML
  contents
  server.send(200, "text/html", s);
  //Send web page
}

void handleADC() {
  int a = analogRead(A0);
  String ADCValue = String(a);

  server.send(200, "text/plane",
  ADCValue); //Send ADC value only
  to client ajax request
}

void handleLED() {
  String ledState = "OFF";
  String t_state =
  server.arg("LEDstate"); //Refer
  xhttp.open("GET",
  "setLED?LEDstate="+led, true);
  Serial.println(t_state);
  if(t_state == "1")
  {
    digitalWrite(LED,LOW); //LED ON
    ledState = "ON"; //Feedback
    parameter
  }
  else
  {
    digitalWrite(LED,HIGH); //LED
    OFF
    ledState = "OFF"; //Feedback
    parameter
  }

  server.send(200, "text/plane",
  ledState); //Send web page
}

void setup(void){
  Serial.begin(115200);
  // WiFi.begin(ssid, password);
  //Connect to your WiFi router
  WiFi.softAP(ssid, password);
  Serial.println("");

  //Onboard LED port Direction
  output
  pinMode(LED,OUTPUT);

  //If connection successful show
  IP address in serial monitor
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  //
  Serial.println(WiFi.localIP());
  //IP address assigned to your ESP
  Serial.println(WiFi.softAPIP());
}
```



```
server.on("/", handleRoot);  
//Which routine to handle at root  
location. This is display page  
server.on("/setLED", handleLED);  
server.on("/readADC",  
handleADC);  
  
server.begin();  
//Start server  
Serial.println("HTTP server  
started");  
  
if (!SD.begin(chipSelect)) {  
Serial.println("Erro na  
leitura do arquivo não existe um  
cartão SD ou o módulo está  
conectado corretamente ?");  
cartaoOk = false;  
return;  
}  
  
if (cartaoOk){  
dataFile =  
SD.open("datalog.csv",  
FILE_WRITE);  
Serial.println("Cartão SD  
Inicializado para escrita :D ");  
}
```

H.1 – Unidade 1:

```
const char MAIN_page[] PROGMEM =  
R"=====  
<!DOCTYPE html>  
<html>  
<body>  
  
<div id="demo">  
<h1>The ESP8266 NodeMCU Update web  
page without refresh</h1>  
<button type="button"  
onclick="sendData(1)">LED  
ON</button>  
<button type="button"  
onclick="sendData(0)">LED  
OFF</button><BR>  
</div>
```

```
}  
  
void loop(void){  
server.handleClient();  
//Handle client requests  
  
//Limpendo Variáveis  
String leitura = ""; // Limpo  
campo contendo string que será  
armazenada em arquivo CSV  
  
String ADCValue = "";  
  
leitura = String(millis()) + ";"  
+ String(ADCValue);  
  
if (dataFile) {  
dataFile.println(leitura); //  
Escrevemos no arquivos e pulamos  
uma linha  
dataFile.close(); //  
Fechamos o arquivo  
}  
delay(1000); // Aguardamos 1  
segundos para executar o loop  
novamente  
}
```

<div>

ADC Value is : <span
id="ADCValue">0

LED State is : <span
id="LEDState">NA
</div>

```
<script>  
function sendData(led) {  
var xhttp = new  
XMLHttpRequest();  
xhttp.onreadystatechange =  
function() {
```




```
        if (this.readyState == 4 &&
this.status == 200) {
document.getElementById("LEDState"
).innerHTML =
        this.responseText;
    }
};
        xmlhttp.open("GET",
"setLED?LEDstate="+led, true);
        xmlhttp.send();
}

setInterval(function() {
    // Call a function repetatively
with 2 Second interval
    getData();
}, 1000); //2000mSeconds update
rate

function getData() {
    var xmlhttp = new
XMLHttpRequest();
    xmlhttp.onreadystatechange =
function() {
        if (this.readyState == 4 &&
this.status == 200) {

document.getElementById("ADCValue"
).innerHTML =
        this.responseText;
    }
};
    xmlhttp.open("GET", "readADC",
true);
    xmlhttp.send();
}
</script>
```

```
<br><br><a
href="https://www.gov.br/inpe/pt-b
r">INPE</a>
</body>
</html>
)=====";
```

APÊNDICE I - TRANSMISSÃO E DADOS ANALÓGICOS ATRAVÉS DE COMANDOS EM C++ E JAVASCRIPT ESP32

I.1 - Unidade 1

```
#include <SPI.h>
```

```
#include <WiFi.h>
```



```
#include <WiFiAP.h>
#include <Arduino.h>
#include <WebServer.h>
#include <WiFiClient.h>
#include <SoftwareSerial.h>

#define LED_BUILTIN 2
#define RXD2 16
#define TXD2 17
#define ANALOGPIN 14

#ifdef APSSID
#define APSSID "ESPMAIMAI"
#define APPSK "12345678"
#endif

const char *ssid = APSSID;
const char *password = APPSK;
int analogvalue = 0;

WebServer server(80);

WiFiClient client;

String header;
String valor;

void handleRoot() {
    String html;
    html = "<!DOCTYPE html>"

    "<html>"

    "<head>"

    "<meta http-equiv='refresh' content='5'>"

    "<title> ESP8266</title>"

    "</head>"

    "<body>"

    "<h1><center>ESP32 - TESTE POTENCIOMETRO!</center></h1>"

    "<center><font size='5'>DADOS INPE!</center><br>"

    "<a href =\"/1\"> <center><button onclick=\"alert('Dados Salvos!!!')\" style=\\\"/font-family:century gothic;color:White;font-size:20px;background-color:SteelBlue;\">GET DATA</button></center></a>\""

    "<a href =\"/2\"> <center><button onclick=\\\"alert('O LED esta on!!')\" style=\\\"/font-family:century gothic;color:White;font-size:20px;background-color:SeaGreen;\">LED ON</button></center></a>\""

    "<a href =\"/3\"> <center><button onclick=\\\"alert('O LED esta off!!')\" style=\\\"/font-family:century gothic;color:White;font-size:20px;background-color:FireBrick;\">LED OFF</button></center></a>\""

    "<center><table border='1'><center>"

    "<colgroup>"

    "<col span='3' style='background-color:Khaki'>"

    "</colgroup>"

    "<tr>"

    "<center><th> VALOR DO POTENCIOMETRO 1 </th></center>"

    "</tr>"

    "<tr>"

    "<td align='center'>"

    "<script type='text/javascript'>"
```



```
"document.write("+String(analogval
ue)+")"

"</script type='text/javascript'"

"</td>"

"</table>"

"</body>"

"</html>";

        server.send(200, "text/html",
html);

}

void ledOn()
{

    String ledState = "on";
    pinMode(LED_BUILTIN, OUTPUT);
    digitalWrite(LED_BUILTIN, HIGH);
    delay(1000);
    ledState = "ON";

    String html;
    html = "<!DOCTYPE html>"

"<html>"

"<head>"

"<meta        http-equiv='refresh'
content='5'">"

"<title> ESP8266</title>"

"</head>"

"<body>"

"<h1><center>ESP8266 MAI - TESTE
POTENCIOMETRO!</center></h1>"

"<center><font        size='5'>DADOS
INPE!</center><br>"

                                "<a
href        =\"/3\">        <center><button
onclick=\\\"alert('O LED esta off!')
\\\"style=\\\"/font-family:century
gothic;color:White;font-size:20px;
background-color:Red;\\\">LED
OFF</button></center></a>\\\""

"<br><a        href        =\"/\">
<center><button
style=\\\"/font-family:century
gothic;color:Black;font-size:20px;
background-color:CorndflowerBlue;\\\"
>PAGINA
INICIAL</button></center></a>"

"<br><center>LED State is : ON
</center>"

"</body>"

"</html>";

        server.send(200, "text/html",
html);

}

void ledOff()

{

    String ledState = "off";
    pinMode(LED_BUILTIN, OUTPUT);
    digitalWrite(LED_BUILTIN, LOW);
    delay(1000);
    ledState = "OFF";

    String html;
    html = "<!DOCTYPE html>"

"<html>"

"<head>"
```



```
"<meta http-equiv='refresh'
content='5'>"
```

```
"<title> ESP8266</title>"
```

```
"</head>"
```

```
"<body>"
```

```
"<h1><center>ESP8266 MAI - TESTE
POTENCIOMETRO!</center></h1>"
```

```
"<center><font size='5'>DADOS
INPE!</center><br>"
```

```
"<a
href =\"/2\"> <center><button
onclick=\"alert('O LED esta on!')
\\\"style=\\\"/font-family:century
gothic;color:Black;font-size:20px;
background-color:Green;\\\">LED
ON</button></center></a>\\\""
```

```
"<br><a href =\"/\">
<center><button
style=\\\"/font-family:century
gothic;color:Black;font-size:20px;
background-color:cornflowerblue;\\\"
>PAGINA
INICIAL</button></center></a>"
```

```
"<br><center>LED State is : OFF
</center>"
```

```
"</body>"
```

```
"</html>";
```

```
server.send(200, "text/html",
html);
```

```
}
```

```
void getdata()
```

```
{
digitalWrite(0, HIGH);
delay(100);
digitalWrite(0, LOW);
```

```
String html;
html = "<!DOCTYPE html>"
```

```
"<html>"
```

```
"<head>"
```

```
"<meta http-equiv='refresh'
content='5'>"
```

```
"<title> ESP8266</title>"
```

```
"</head>"
```

```
"<body>"
```

```
"<h1><center>EXIBICAO DE
DADOS</center></h1>"
```

```
"<center><font size='5'>DADOS
INPE!</center><br>"
```

```
"<center><span
id='document.write(\"+String('valor
')+\"')>0</span><br></center>"
```

```
"<br><a href =\"/\">
<center><button
style=\\\"/font-family:century
gothic;color:Black;font-size:20px;
background-color:cornflowerblue;\\\"
>PAGINA
INICIAL</button></center></a>"
```

```
"</body>"
```

```
"</html>";
```

```
server.send(200, "text/html",
html);
```

```
delay(1000);
```

```
}
```

```
void setup() {
```



```
pinMode(LED_BUILTIN, OUTPUT);
Serial.begin(115200);
Serial.println();
    Serial.println("Configuring
access point...\n");
    WiFi.softAP(ssid, password);
        IPAddress    myIP    =
WiFi.softAPIP();
    Serial.print("AP IP address: ");
    Serial.println(myIP);

    Serial.println("");
    Serial.print("Connected to ");
    Serial.println(ssid);
    Serial.println(WiFi.softAPIP());

    server.on("/", handleRoot);
    server.on("/1", getData);
    server.on("/2", ledOn);
    server.on("/3", ledOff);

server.begin();
    Serial.println("Servidor HTTP
iniciado");

    delay(500);
}

void loop() {

    server.handleClient();

        valor = Serial.readString();
            analogvalue =
analogRead(ANALOGPIN);
            Serial.println(valor);
            valor = analogvalue;

        delay(1000);
}
```

I.2 - Unidade 2

```
#include <SoftwareSerial.h>
#include <Wire.h>
#include <SPI.h> //
Biblioteca de comunicação SPI
#include <SD.h> //
Biblioteca de comunicação com
cartão SD
#include "RTClib.h"

String str;
String arquivo="DADOSIN.txt";
int cont=0;
int interruptPin = 2;
const int pinoSS = 3;

RTC_DS3231 RTC;

char    daysOfTheWeek[7][12]    =
{"Domingo", "Segunda", "Terça",
"Quarta", "Quinta", "Sexta",
"Sábado"}; //DECLARAÇÃO DOS DIAS
DA SEMANA
char buf[100];

SoftwareSerial Serial1(5, 6); //
RX, TX

File    myFile    =
SD.open("DADOSIN.txt");

void setup(){

    Serial1.begin(115200);
    Serial.begin(9600);

    RTC.begin();

    RTC_DS3231 RTC;
        char    daysOfTheWeek[7][12]    =
{"Domingo", "Segunda", "Terça",
"Quarta", "Quinta", "Sexta",
"Sábado"};

        if (SD.begin()) { // Inicializa
o SD Card
```



```
Serial.println("SD Card pronto
para uso."); // Imprime na tela
}

else {
    Serial.println("Falha na
inicializacao do SD Card.");
    return;
}

//REMOVA O COMENTÁRIO DE UMA
DAS LINHAS ABAIXO PARA INSERIR AS
INFORMAÇÕES ATUALIZADAS EM SEU RTC

//RTC.adjust(DateTime(F(__DATE__),
F(__TIME__)));
//RTC.adjust(DateTime(2022,
05, 05, 09, 57, 00));

delay(1000);
}

void loop()
{

const int analogpin = A5;
int valor = 0;

Serial.begin(9600);

valor = analogRead(A5);

Serial1.println(valor);
Serial.println(valor);

delay(1000);

        myFile =
SD.open(arquivo,FILE_WRITE);

digitalWrite(analogpin, valor);

    DateTime now = RTC.now(); //
obtendo tempo do RTC

myFile.print("Data: ");
myFile.print(now.day(), DEC);
myFile.print('/');

        myFile.print(now.month(),
DEC);
myFile.print('/');
myFile.print(now.year(), DEC);
myFile.print(" / Dia: ");

myFile.print(daysOfTheWeek[now.day
OfTheWeek()]);
myFile.print(" / Horas: ");
myFile.print(now.hour(), DEC);
myFile.print(':');
myFile.print(now.minute(),
DEC);
myFile.print(':');
myFile.print(now.second(),
DEC);
myFile.print(':');
myFile.print(" /
Potenciometro");
myFile.print(':');
myFile.print(valor);
myFile.println();
delay(100);
myFile.close();

    sprintf(buf, "%d/%d/%d %d:%d:%d
potenciometro:%i % \r",now.year(),
now.month(), now.day(),
now.hour(), now.minute(),
now.second(), (int)valor);
    Serial.println(buf);
    cont++;
    delay(1000);

attachInterrupt(digitalPinToInterr
upt(interruptPin), card, RISING);
        pinMode(interruptPin,
INPUT_PULLUP);

}

void card(){

myFile = SD.open(arquivo);

if (myFile) {
```



```
//Serial.println("conteudo do arquivo " + arquivo + ":");  
  
while (myFile.available()) {  
    Serial.println(myFile.read());  
    delay(100);  
}  
  
Serial.println("-----  
-----");  
  
myFile.close();  
}  
  
else {  
    Serial.println("erro ao abrir  
" + arquivo );  
}  
  
delay(1000);  
}
```