UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Ilha Solteira

UNIVERSIDADE ESTADUAL PAULISTA – UNESP

FACULDADE DE ENGENHARIA CÂMPUS DE ILHA SOLTEIRA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Matheus Rogério Bocchi da Silva

Sistema de controle por navegador para quarto de hospital

Ilha Solteira

2023

Faculdade de Engenharia de Ilha Solteira
Cursos: Agronomia, Ciências Biológicas, Eng. Civil, Eng. Elétrica, Eng. Mecânica, Física, Matemática e Zootecnia.
Avenida Brasil Centro, 56 Caixa Postal 31 CEP 15385-000 Ilha Solteira São Paulo Brasil
pabx (18) 3743 1000 fax (18) 3742 2735 scom@adm.feis.unesp.br www.feis.unesp.br

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Ilha Solteira

Sistema de controle por navegador para quarto de hospital

Trabalho de Graduação apresentado à Faculdade de Engenharia de Ilha Solteira-UNESP, como requisito parcial à obtenção do título de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Marcelo Augusto Assunção Sanches

Ilha Solteira

2023

Faculdade de Engenharia de Ilha Solteira
Cursos: Agronomia, Ciências Biológicas, Eng. Civil, Eng. Elétrica, Eng. Mecânica, Física, Matemática e Zootecnia.
Avenida Brasil Centro, 56 Caixa Postal 31 CEP 15385-000 Ilha Solteira São Paulo Brasil
pabx (18) 3743 1000 fax (18) 3742 2735 scom@adm.feis.unesp.br www.feis.unesp.br

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

Silva, Matheus Rogério Bocchi da.
S586s Sistema de controle por navegador para quarto de hospital / Matheus Rogério Bocchi da Silva. -- Ilha Solteira: [s.n.], 2023
88 f. : il.

Trabalho de conclusão de curso (Graduação em Engenharia Elétrica) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira, 2023

Orientador: Marcelo Augusto Assunção Sanches

Inclui bibliografia

1. Hospital. 2. Esp-32. 3. Gerenciamento inteligente.

Raiane da Silva Santos
Supervisora Técnica de Seção
Seção Técnica de Referência, Atendimento ao usuário e Documentação
Diretoria Técnica de Biblioteca e Documentação
CRB/8 - 9999

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira

ATA DE DEFESA DE TRABALHO DE GRADUAÇÃO

Aos vinte e nove dias do mês de junho do ano de dois mil e vinte e três, o discente ***Matheus Rogério Bocchi da Silva,*** matriculado sob o nº 151051127, tendo como banca examinadora o seu orientador, *o Prof. Dr. Marcelo Augusto Assunção Sanches, o Mestrando. Leonardo Xavier do Nascimento e Mestranda. Sara Moreira Macedo,* apresentou o Trabalho de Graduação intitulado **"Sistema de controle por navegador para quarto de hospital",** obtendo a nota 10,0 (dez) e conceito Aprovado.

Prof. Dr. Marcelo Augusto Assunção Sanches
- orientador -

Matheus Rogério Bocchi da Silva
- discente -

Leonardo Xavier do Nascimento
Mestrando Leonardo Xavier do Nascimento
- Membro da Banca -

Sara Moreira Macedo
Mestranda. Sara Moreira Macedo
- Membro da Banca -

Faculdade de Engenharia – Departamento de Engenharia Elétrica
Avenida Professor José Carlos Rossi, 1370 - CEP 15.385-000, Ilha Solteira - SP - Brasil
Tel (18) 37431150 - Fax (18) 3742-2735 - www.feis.unesp.br

EVE

A mim mesmo, pelos anos de dedicação e aos meus pais, por sempre me ajudarem

Faculdade de Engenharia de Ilha Solteira
Cursos: Agronomia, Ciências Biológicas, Eng. Civil, Eng. Elétrica, Eng. Mecânica, Física, Matemática e Zootecnia.
Avenida Brasil Centro, 56 Caixa Postal 31 CEP 15385-000 Ilha Solteira São Paulo Brasil
pabx (18) 3743 1000 fax (18) 3742 2735 scom@adm.feis.unesp.br www.feis.unesp.br

Agradecimentos

Primeiramente agradeço ao Professor Marcelo Augusto Assunção Sanches por toda a sua orientação na realização deste trabalho e sua eterna dedicação e disponibilidade para todos os alunos que precisam e o procuram.

Juntamente agradeço ao meu amigo Luiz Feltrim, pelos anos de estudos juntos e pela grande ajuda na compreensão de HTML e *Javascript* no início deste trabalho.

Por último, agradeço aos meus pais, principalmente minha mãe, por sempre ajudar ao longo de meus anos neste curso, incentivando minha embarcada no mundo dos eletrônicos e tendo paciência em me ouvir conversar sobre a área incessantemente.

Resumo

A internação de uma pessoa em hospital requer bastante cuidado e atenção e devido ao grande número de internações e o número limitado de colaboradores, é importante ter meios que auxiliem nestes cuidados e que deem mais autonomia ao paciente. O presente trabalho refere-se ao projeto de um sistema inteligente que permite um maior controle do paciente sobre o quarto que está internado, podendo controlar a luminosidade do ambiente, monitorar o nível de soro e avisar quando o mesmo acaba, selecionar a numeração do presente quarto em conjunto com uma URL (*Uniform Resource Locator*) customizada, utilizar 3 interruptores inteligentes, ter contato direto com os funcionários de recinto, salvar e carregar fichas dos pacientes em um cartão de memória, utilizando apenas um dispositivo digital, ou smartphone, com acesso à internet e a um navegador. A realização deste controle foi feita graças a o microcontrolador ESP-32, responsável por todo o processamento necessário, em conjunto de páginas web escritas em HTML e *Javascript*, criando assim a interface humana necessária. O trabalho, depois de testes preliminares em *protoboards*, foi consolidado em uma PCB (*printed circuit board*), com todos os seus circuitos integrados na mesma, fazendo com que a placa seja facilmente integrada no ambiente desejado. Testes no projeto final demonstraram performance satisfatória, atingindo todos os seus objetivos e executando todas as suas funcionalidades de acordo com o esperado.

Palavras chaves: Hospital; ESP-32; Gerenciamento inteligente.

Abstract

Hospitalizing a person requires a lot of care and attention due to a high number of patients and a limited number of caretakers, it's vital the existence of means that assist these needs and give more autonomy to the patient. This present work refers to a project of a smart system that gives more control over a hospital room to a hospitalized person, which gives control over the brightness of the ambient light, monitors the amount of IV and warns when it's empty, assigns the room a number with a custom URL (Uniform Resource Locator), controls 3 smart relays, gives direct contact with the health workers, saves and loads the patient data to an SD card, all of this needing only a digital device with internet access and a browser. This system was done using the ESP-32 microcontroller, responsible for all the processing needed in addition with the web pages written in HTML and Javascript, creating a human interface. With preliminary tests done in breadboards, the project was consolidated on a PCB (Printed Circuit Board), incorporating all the circuits necessary, making this board very easy to integrated in a desired location. Tests were made on the final stage of this project, with the assembled board, which gave positive results, reach all the goals set in the beginning of this work.

Key words: Hospital; ESP-32; Smart managing.

Lista de Figuras

Lista de Tabelas

Lista de abreviaturas e siglas

ADC	*Analog to Digital Converter*
CDN	*Content Delivery Network*
CI	Circuito Integrado
CS	*Chip Select*
CSS	*Cascading Style Sheets*
DIP	*Dual In-line Package*
DNS	*Domain Name System*
EN	*Enable*
FTDI	*Future Technology Devices International Limited*
GND	*Ground*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
I/O	*Input/Output*
I2C	*Inter-Integrated Circuit*
I2S	*Inter-IC Sound*
IP	*Internet Protocol*
mDNS	*Multicast Domain Name System*
MISO	*Master In Slave Out*
MOSI	*Master Out Slave In*
OLED	*Organic Light-emitting Diode*
SCLK	*Serial Clock*
SD	*Secure Digital*
SDMMC	*Secure Digital Multi Media Card*
SPI	*Serial Peripheral Interface*
SPIFFS	*Serial Peripheral Interface Flash File System*
TH	*Through Hole*
UART	*Universal Asynchronous Receiver-Transmitter*
URL,	*Uniform Resource Locator*

Sumário

1. Introdução

O processo de hospitalização de uma pessoa contém diversos procedimentos e protocolos visando a melhoria da saúde deste paciente ao longo de sua estadia no hospital. No entanto, algumas vezes a saúde mental do mesmo acaba ficando em segundo plano, focando-se apenas na sua melhora física. O mesmo se vê em um lugar estranho, sem rotinas comuns e com poucos estímulos, o que acaba desorientando o paciente em sua hospitalização prolongada (PISTORIA, 2021).

Com o avanço da era tecnológica, a quantidade de *smartphones* nas mãos das pessoas está cada vez mais aumentando. Segundo a pesquisa de *Global* Mobile Market Report (SOUZA, 2021), o Brasil é o quinto país com o maior número de celulares, contando com 109 milhões de usuários. Estima-se, em média, uma proporção de 4 celulares por 1 aparelho de tv e, levando em conta outros dispositivos digitais como computadores, *tablets* e *laptops*, são 2 por habitante (Portal FGV, 2021).

Levando em consideração o grande número de dispositivos digitais, a probabilidade do paciente a ser hospitalizado ter um destes aparelhos é alta, sendo assim o melhor ponto de entrada para encontrar uma solução ao problema descrito anteriormente.

Proporcionar um maior controle do ambiente de sua estadia pode prevenir o declínio mental do paciente. Tendo isso em mente, foi desenvolvido um projeto utilizando um ESP32 em conjunto com um *dimmer*, 3 relés, uma *Load Cell* com um amplificador HX711.

1.1. Objetivo

Este trabalho tem como objetivo criar um sistema de controle para quartos de hospitais, com uma interface para controle de luminosidade, ativação de interruptores, monitoramento de soro, aviso para enfermeiros(as) em casos de emergência e arquivamento de fichas dentro de um cartão de memória, utilizando um ESP32 funcionando conectado à rede WiFi disponível no quarto de hospital, permitindo o uso pelo paciente utilizando um smartphone.

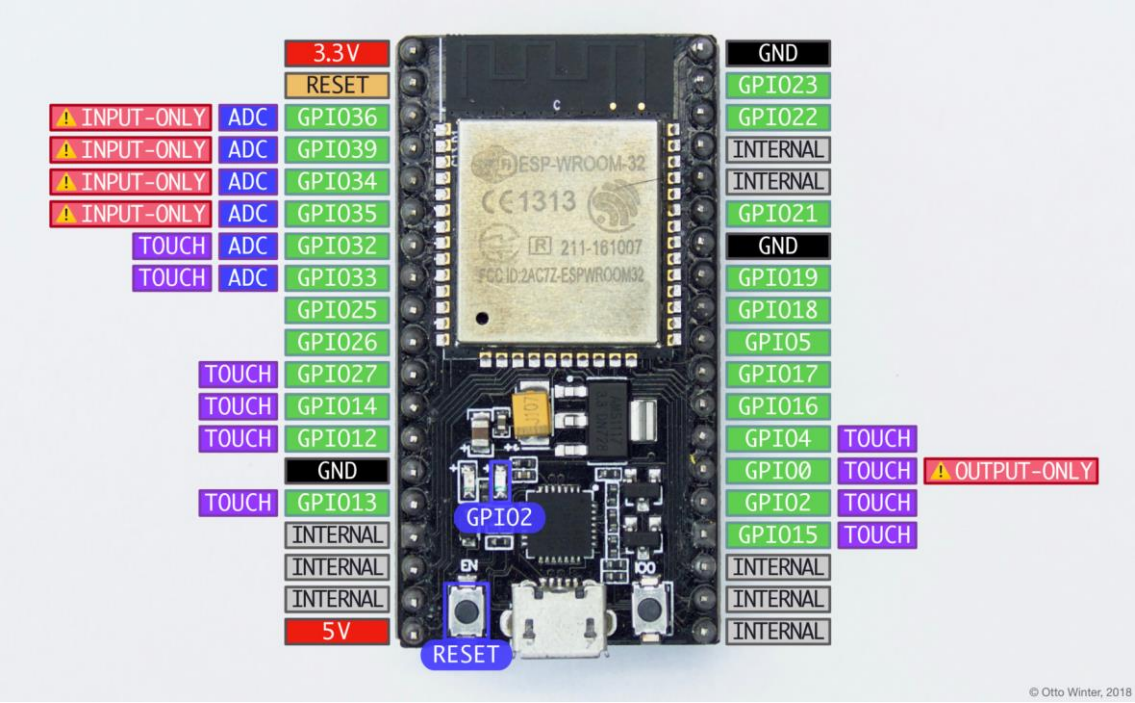
2. Fundamentos teóricos

2.1. ESP-32

Criado pela Expressif, o ESP-32 abrange uma série de microcontroladores com características distintas, sendo o ESP-WROOM-32 escolhido para a realização deste projeto. O mesmo contém dois núcleos de processamento ("*cores*") com frequência ajustável de 80MHz a 240MHz, acesso a *WiFi* e *Bluetooth*, 15 portas I/O disponíveis e mais 4 como só entrada, além de ser compatível com SPI, UART, I2S, I2C e cartões SD (SYSTEMS, 2023).

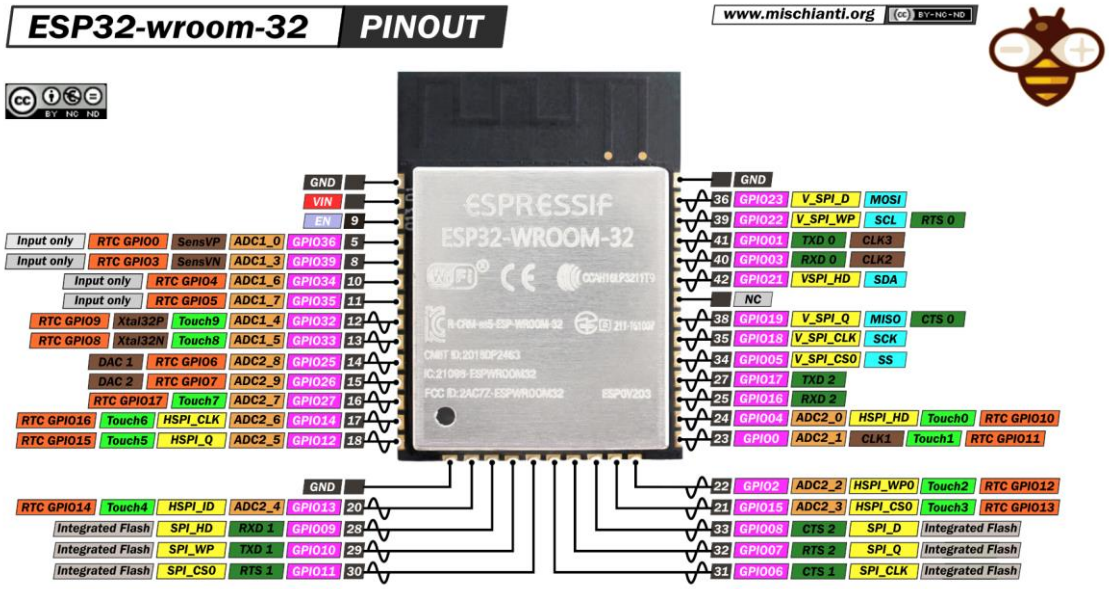
No plano original deste projeto se planejava a utilização do ESP8266, microcontrolador criado pela a mesma empresa, com menos funcionalidades, mas de valor mais baixo. Após serem testadas a implementação das funcionalidades desejadas, no entanto, foi determinado que o ESP8266 não seria adequado principalmente devido à falta de I/Os disponíveis necessárias para o escopo deste projeto, além de seu funcionamento em apenas um núcleo de processamento (*single core*) apresentar problemas com a operação do *dimmer* em conjunto com o programa geral. Logo, foi utilizado o módulo NodeMCU para a prototipagem do trabalho (**Figura 1**), enquanto o ESP-WROOM-32 (**Figura 2**) foi utilizado para sua consolidação.

Figura 1 – NodeMCU utilizado para desenvolver o projeto.



Fonte: (ESPHOME, 2023).

Figura 2 – ESP-WROOM-32 utilizado no produto final.



Fonte: (MISCHIANTI, 2021).

2.2. HTML, CSS, *Javascript* e HTTP

Como a forma de implementação de interface humana teve como objetivo ser abrangente, não sendo restrita a dispositivos específicos, o uso de navegadores de internet foi escolhido. Logo, foi necessário a criação de páginas web, o qual é feito com base nos seguintes componentes:

- HTML (*HyperText Markup Language*), ou "linguagem de marcação de hypertexto", é uma linguagem que define a estrutura de texto e insere elementos e funções para seu funcionamento, sendo a base para a criação de páginas da web (MDN WEB DOCS, 2022).
- CSS (*Cascading Style Sheets*), ou "folhas de estilo em cascata", é uma linguagem focada na estilização do descrito em um texto HTML, sendo responsável, por exemplo, em mudar fontes, cores, alinhamento, posição, dentre outros (COMPUTER HOPE, 2019).
- *Javascript* consiste em uma linguagem de programação destinada a implementar itens complexos a páginas web sem a necessidade de recarrega-lás, como gerar gráficos 2D/3D, controlar multimídias, imagens animadas, dentre outros. Neste projeto, seu principal uso foi na criação de

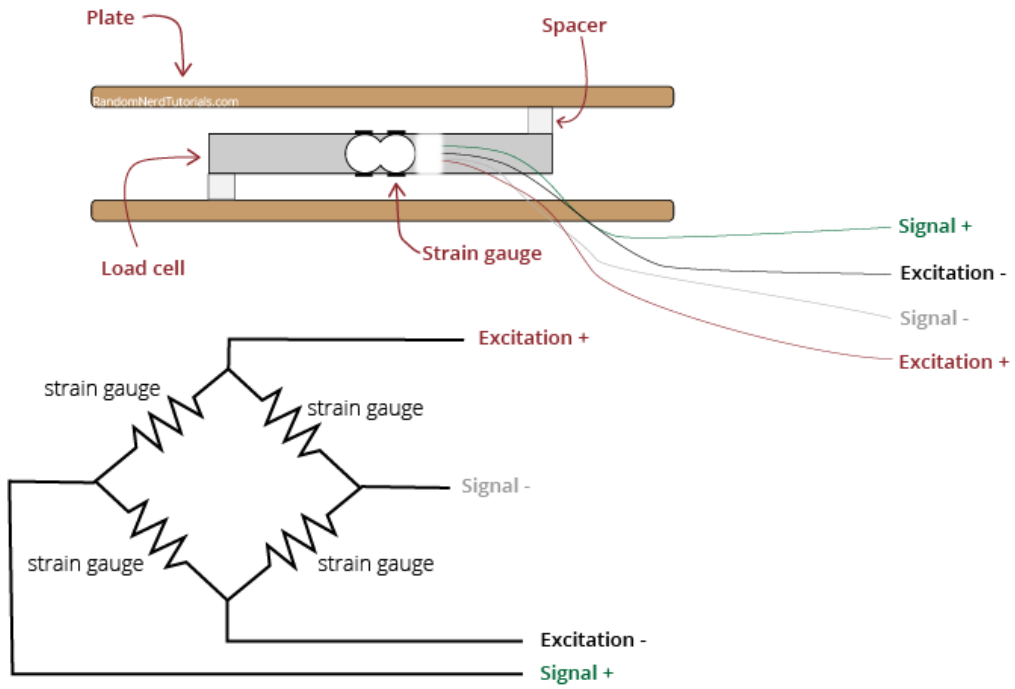
funções que permitiram a comunicação entre o microcontrolador e o dispositivo digital conectado a mesma rede, utilizando as rotas de HTTP (*Hypertext Transfer Protocol*) (MDN WEB DOCS, 2023a).

- HTTP (*Hypertext Transfer Protocol*), ou "protocolo de transferência de hypertexto", é um protocolo de comunicação entre clientes (navegadores) e servidores, baseado em diversos métodos de aquisição, como GET, POST, PUT, dentre outros (MDN WEB DOCS, 2023b). Sua utilização nesse projeto foi de extrema importância, pois foi responsável por toda a comunicação entre o ESP32 e o navegador, sendo assim a base geral do funcionamento do trabalho. Em todas as suas instâncias dentro do projeto, o método de aquisição utilizado foi o GET, consistindo em uma requisição de dados que pode ser armazenada no *cache* do navegador, o que se provou útil na depuração do programa, ser salva no histórico do navegador e em seus "*bookmarks*", além de ser totalmente visível na URL (W3SCHOOLS, 2023).

2.3. Load cell

A *Load Cell*, ou célula de carga, é um sensor eletromecânico responsável por medir a atuação de força mecânica aplicada a seu corpo. Para realizar essa medição, vários extensores ("*strain gauges*") são distribuídos ao longo de sua estrutura, fazendo com que qualquer deformação na mesma cause alteração na resistência apresentada por estes extensores (FLINTEC, 2023). O funcionamento de uma célula de carga de ponto único, a utilizada neste projeto, se dá em base a uma ponte de Wheatstone (ELECTRONICS TUTORIALS, 2023), uma ponte de 4 resistores com valores conhecidos e tensão medida entre seus pontos centrais, como mostrado na **Figura 3**. Deformações ao longo da barra causam mudanças nas resistências dos extensores, causando variação na tensão medida entre os pontos. Esta variação de tensão é então convertida por um conversor analógico para digital (ADC, ou *Analog to Digital Converter*), sendo assim melhor adequada para o uso em conjunto com microcontroladores (RANDOM NERD TUTORIALS, 2023).

Figura 3 – *Load Cell* e sua ponte de *Wheatstone*.



Fonte: (RANDOM NERD TUTORIALS, 2023).

2.4. DNS e mDNS

O funcionamento da internet se dá no acesso de sites por meio de navegadores. Para acessar estes sites, é necessário saber o IP destinado ao mesmo, pois todos os dispositivos conectados à rede recebem seu próprio IP para identificação. Dessa forma, para que não seja necessário digitar o número IP de todo o site desejado a ser acesso, são usados servidores DNS (*Domain Name System*).

Servidores DNS consistem em computadores com um banco de dados que contém endereços IP públicos, que estão associados a sites. Portanto, ao escrever o nome do site desejado, o servidor DNS encontra o número IP correspondente que é passado para o navegador, que, por sua vez, envia os dados para a CDN (*Content Delivery Network*) ou o servidor de origem. Portanto é possível acessar sites, sem a necessidade de escrever seu IP, com o servidor DNS funcionando como uma "lista telefônica" (FORTINET, 2023).

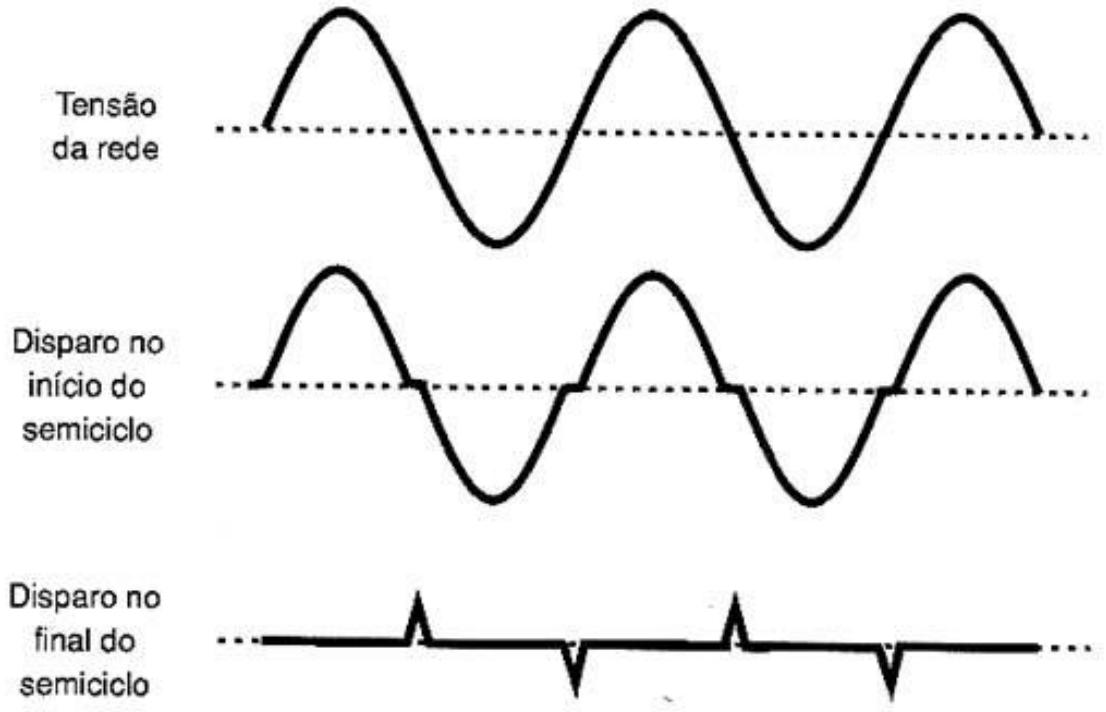
Agora mDNS, ou *multicast* DNS, consiste na mesma funcionalidade descrita para servidores DNS, porém visando uma escala menor. Seu funcionamento consiste no cliente enviar um "*multicast*", protocolo de comunicação entre dois

IPs dentro de uma mesma conexão, na rede que está conectado e verificando qual dispositivo conectado a mesma contém tal nome. Desta forma, não a necessidade de utilizar um servidor externo, mas sim identificações únicas dentro de uma mesma rede WIFI. Para o uso deste protocolo no projeto, foram criados URLs (*Uniform Resource Locato*), com sufixo ".local/" para denominar um endereço mDNS (IONOS, 2020).

2.5. Dimmer

O funcionamento de um *dimmer* é necessário controlar a tensão que alimenta a lâmpada responsável pela iluminação. Tal feito pode ser alcançado a partir do uso de um Triac, um componente formado por dois tiristores em paralelo, mas em sentidos opostos, o que possibilita o controle de sua ativação por meio do ângulo de disparo como um tiristor, porém permitindo a passagem de corrente em ambos os sentidos do circuito. Variando os ângulos de disparo, a carga recebe diferentes potências, como mostrado na **Figura 4** (INSTITUTO NCB, 2023).

Figura 4 - Variação de potência utilizando um Triac.

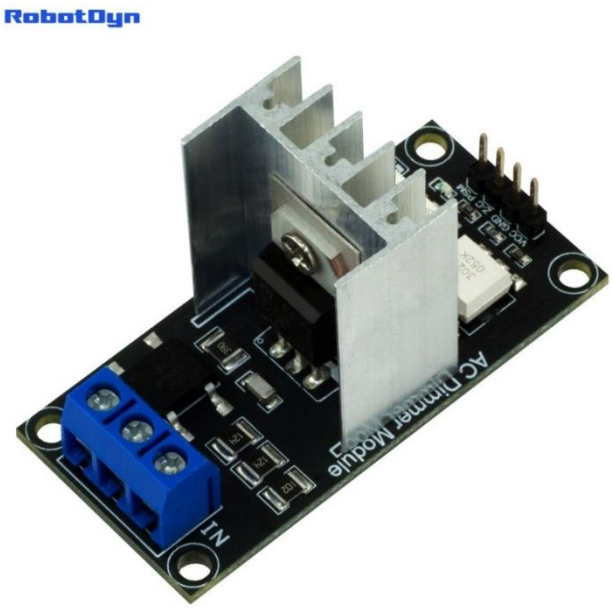


Fonte: (INSTITUTO NCB, 2023).

No projeto, este processo de variação de potência é feito com um módulo *dimmer*, mostrado na **Figura 5**, que funciona a partir do controle do microcontrolador. Este controle só consegue ser feito porque o módulo utiliza o

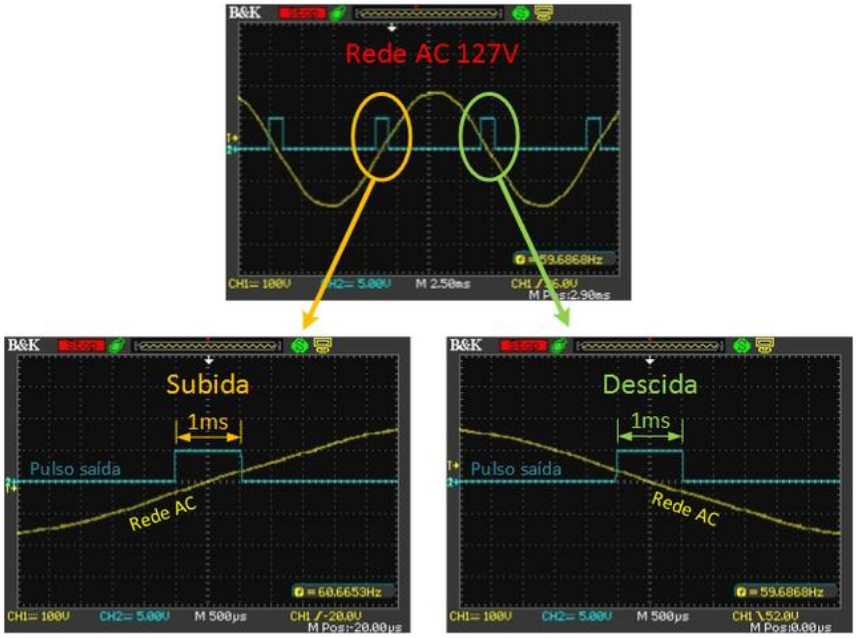
processo de zero crossing para determinar o ângulo de disparo do triac. Zero crossing consiste no ponto onde a função muda de sinal, no caso, onde o eixo (valor 0) é interceptado (exemplo encontrado na **Figura 6**). Assim, é possível sincronizar o microcontrolador com os pontos em que o valor de y é igual a 0 e permitir o controle sobre o triac.

Figura 5 - *AC Dimmer Module.*



Fonte: (ROBOTDYN, 2022).

Figura 6 – Exemplo de sincronização com *zero crossing*.



Fonte: (CIRCUITAR, 2023).

3. Metodologia

3.1. Materiais

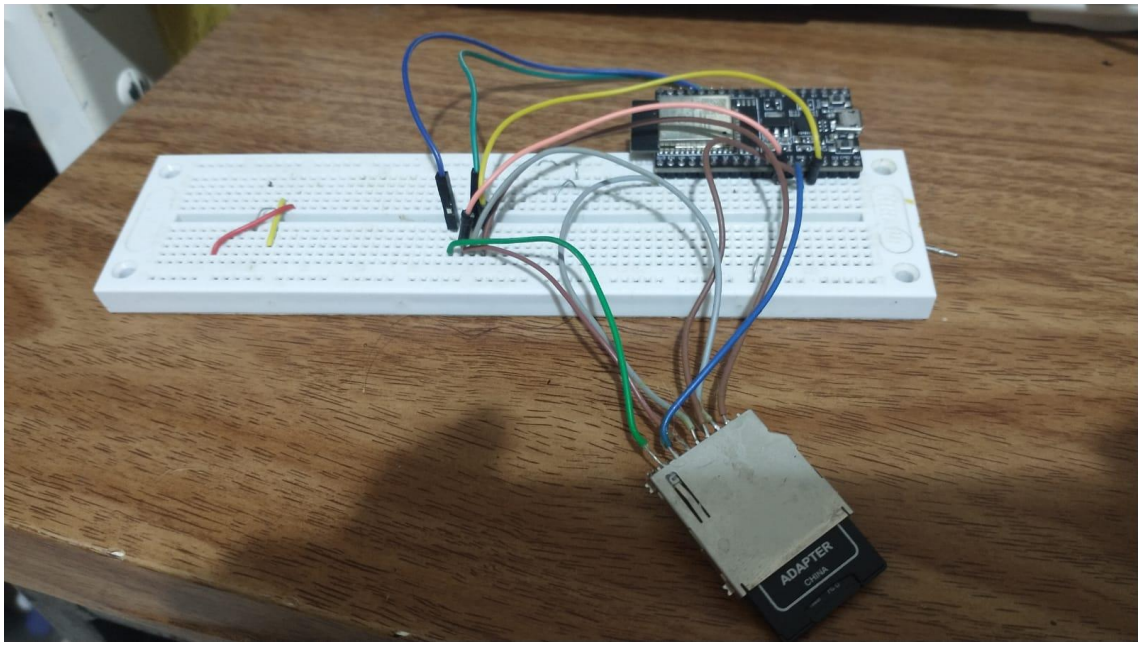
Os materiais utilizados ao longo deste projeto e na consolidação da placa PCB consistem em:

- Resistores de 1kΩ, 1.2kΩ, 8.2kΩ, 10kΩ e 20kΩ de tamanho 1206;
- Capacitores de 100nF, 1µF e 10µF cerâmicos com tamanho 1206, 22µF de tântalo e 2200µF eletrolítico;
- Diodos 1N4007;
- Botões SMD KFC-A06;
- Transistores NPN 2N3904;
- Relés 5V SRD-05VDC-SL-C;
- Regulador linear de tensão de 3.3V AMS1117-3.3S;
- Circuito integrado HX711;
- DIP *switch* com 4 pinos de seleção;
- *Bornes* duplos com 5.08mm;
- Transistor PNP BC807 com tamanho SOT-23;
- Conector AC bipolar tipo 8;
- Conector de cartão de memória SDMMC;
- ESP-WROOM-32;
- Conector Mini Din 4;
- Módulo *dimmer Robotdyn*;
- Módulo conversor AC/DC (110v AC – 5V DC) Hi-Link HLK-10M05
- Programador USB FTDI 3.3V.

Como parte central deste projeto, o ESP32 foi escolhido ao invés do ESP8266 devido a seu maior número de portas *I/Os*, processador de maior velocidade e dois núcleos de processamento. Testes iniciais foram feitos utilizando o ESP8266 inserido em uma *protoboard*, porém logo seu uso demonstrou inadequado no começo da programação.

Para garantir um melhor funcionamento da programação e facilitar o seu *debug*, o código foi dividido em partes de acordo com cada função desejada, sendo testadas isoladamente e depois adicionadas ao código principal. Um exemplo disso pode ser observado na **Figura 7**, onde o módulo *NodeMCU* do ESP32 está pareado a um conector de cartão de memória, o qual toda a função de utilização do mesmo presente no código principal foi programada e testada separadamente nesta *protoboard*.

Figura 7 - Teste *NodeMCU* Cartão SD



Fonte: Próprio do autor.

Os testes na *protoboard* foram feitos para: funcionamento da *load cell* em conjunto com amplificador HX711, controle de intensidade de luz com o *dimmer Robotdyn*, acionamento dos relés, ler e salvar arquivos utilizando um cartão SD, funcionamento geral do *display* OLED.

Assim, a programação deste projeto foi finalizada, estando disponível no Apêndice A. Sua explicação em detalhes se encontra abaixo, focando em suas funções específicas.

3.2. Programação

3.2.1. SPIFFS

SPIFFS (*Serial Peripheral Interface Flash File System*) consiste em um sistema de arquivo para utilização em um ESP8266/32, fazendo com que sua memória *flash* funcione como uma unidade de armazenamento, sem suporte para diretórios complexos, apenas sua raiz. Portanto é possível inserir arquivos completos nesta memória e referenciar os mesmos na programação.

A comunicação entre internet e microcontrolador se dá por *web pages*, as mesmas presentes dentro do código do programa. Inserir toda a programação

de uma página HTML dentro do programa contido no ESP32 não só aumenta as oportunidades de erro, mas como não permite a organização de uma lista de arquivos com conteúdo separados, como um arquivo CSS e mais páginas HTML.

Logo, com o SPIFFS foi possível utilizar duas páginas HTML, uma com o monitoramento geral ("monitoramento.html") e outra com a ficha do paciente ("ficha.html"), além de um arquivo separado com as informações para a estética do site ("style.css"), todas salvas dentro da memória *flash* do ESP32 e sendo referenciadas no código.

Foi necessário a adição da biblioteca responsável por habilitar esta funcionalidade, chamada "SPIFFS.h" (**Figura 8**), além de incorporar um arquivo responsável por adicionar um botão à interface do Arduino IDE que permite fazer o *upload* de arquivos presentes na pasta "*data*" do projeto à memória do ESP. No código, foi iniciado a biblioteca e indicado o diretório de cada página em cada uma de suas rotas HTTP criadas no servidor assíncrono (**Figura 9**).

Figura 8 – Inicialização da biblioteca SPIFFS.h.

```
//Inicialização do SPIFFS
if(!SPIFFS.begin(true)){
  Serial.println("");
  Serial.println("Erro ao iniciar SPIFFS");
  display.println("Erro ao iniciar SPIFFS");
  display.display();
  return;
}
```

Fonte: Próprio do autor.

Figura 9 – Exemplo de rotas HTTP com diretório graças ao SPIFFS.

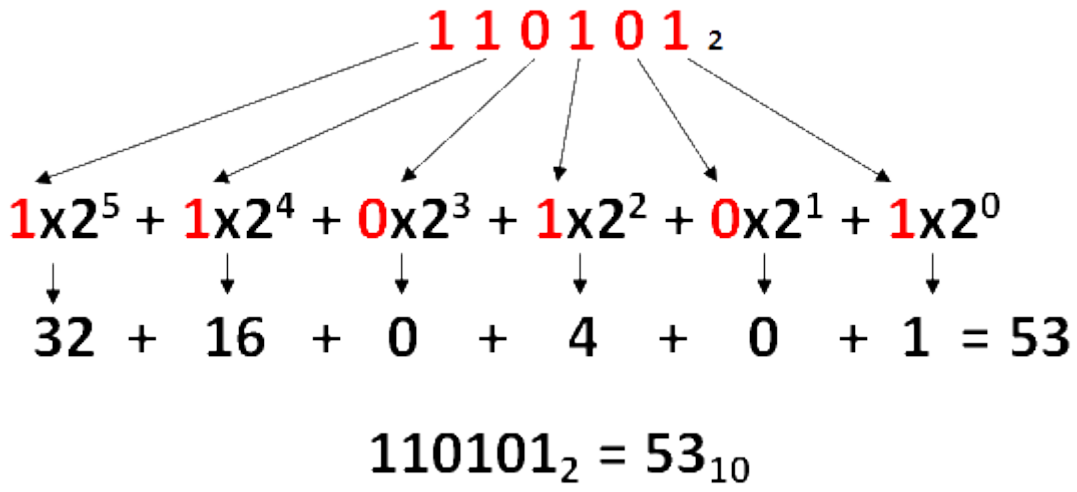
```
server.on("/ficha-paciente", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send(SPIFFS, "/ficha.html");
});
server.on("/monitoramento", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send(SPIFFS, "/monitoramento.html");
});
```

Fonte: Próprio do autor.

3.2.2. DIP *switch*

A existência de um DIP *switch* de 4 pinos no circuito do projeto visou a possibilidade de selecionar qual o número do quarto que a placa se apresenta, gerando URLs específicas para cada número. A função "selecao_quartos()" (**Figura 11** e **Figura 12**) faz a leitura de quatro portas I/Os previamente configuradas como entrada ("*input*"), as quais são interpretadas como dígitos individuais de um número binário de 4 bits. Assim, este número é convertido para decimal, seguindo a fórmula representada na **Figura 10**, com um intervalo de 0 a 15. Para facilitar a nomenclatura e evitar a existência de um "quarto 0", o resultado da conversão foi adicionado a 1, existindo então "quarto 1" a "quarto 16".

Figura 10 – Conversão binário para decimal.



Fonte: (SILVA, 2018).

Com o número decimal adquirido, é convertido para "*string*", para poder ser adicionado ao prefixo "quarto-" e o sufixo ".local/", gerando a URL específica para aquele quarto, como mostrado na **Figura 13**.

Figura 11 – DIP *switch* interpretado como entrada.

```
//Entrada dip switch para número do quarto
pinMode(39, INPUT);
pinMode(36, INPUT);
pinMode(35, INPUT);
pinMode(34, INPUT);
```

Fonte: Próprio do autor.

Figura 12 – Função "selecao_quartos()".

```
//Responsável por ler a entrada binaria fornecida pelo DIP switch, convertendo em decimal para a URL fornecida
void selecao_quartos(){
  switch_pin1 = 1 - digitalRead(39);
  switch_pin2 = 1 - digitalRead(36);
  switch_pin3 = 1 - digitalRead(35);
  switch_pin4 = 1 - digitalRead(34);
  Serial.println("");
  Serial.print("Número Binário: ");
  Serial.print(switch_pin4);
  Serial.print(switch_pin3);
  Serial.print(switch_pin2);
  Serial.print(switch_pin1);
  bin_number = (((switch_pin4) * 8) + ((switch_pin3) * 4) + ((switch_pin2)*2) + (switch_pin1) + 1);
  str_number = String(bin_number);
  url = "quarto-" + str_number;
  url_char = url.c_str();

}
```

Fonte: Próprio do autor.

Figura 13 – Adição do ".local/" na função mDNS.

```
//Geração do mDNS (URL)
if (!MDNS.begin(url_char)) {
  Serial.println("");
  Serial.println("Erro ao iniciar mDNS");
  display.println("Erro mDNS");
  display.display();

}
Serial.println("");
Serial.println("mDNS gerado com sucesso");
Serial.println("");
Serial.print("URL gerado: ");

Serial.print(url);
Serial.print(".local/");
display.println("");
display.print("URL: ");
display.print(url);
display.print(".local/");
display.display();
```

Fonte: Próprio do autor.

3.2.3. Comunicação entre a *web page* e o ESP-32

O fundamento básico do funcionamento deste projeto se dá na comunicação entre as páginas webs e o microcontrolador, havendo a troca de dados em ambas as direções, tanto no envio quanto no recebimento de dados. Isso foi alcançado graças a marcadores dentro do código HTML e funções em *Javascript* que solicitam informações para esses marcadores.

Os marcadores se apresentam como "*placeholders*" (**Figura 14**), palavras temporárias que demarcam qual posição dentro do código será substituída pelo o que realmente se deseja, enquanto uma função *javascript* (**Figura 15**) faz a solicitação HTTP e informa qual o marcador a ser substituído. Como exemplo, para a visualização da quantidade de soro, a página HTML faz a solicitação HTTP pela função *javascript* para substituir o marcador "%SORO%" (repetindo a cada 10 segundos), que é respondida por uma função "*processor*" no código do ESP, que determina qual *placeholder* está sendo solicitado e, assim, qual a resposta certa para o servidor.

Os marcadores utilizados foram: SORO, INTERRUPTORES, SLIDER-VALUE, AJUDA-SORO, NOME, IDADE, QUEIXAS E DIAGNOSTICO. Todos seguem o mesmo princípio discutido anteriormente, com ressalva ao "INTERRUPTORES" que envia três linhas de código inteiras com o estado das saídas para os relés ao invés de somente um único dado.

Figura 14 – Marcadores ("*placeholders*") na página de monitoramento.

```
18  <div class="left">
19      <p>Nivel de soro: <span id="SORO"> %SORO%</span></p>
20  </div>
21
22  <div class="left_ajuda" id="AJUDA-SORO">
23      %AJUDA-SORO%
24  </div>
25
26  <div class="left_ajuda_geral" id="AJUDA-GERAL">
27      %AJUDA-GERAL%
28  </div>
29
30  <div class="right">
31      %INTERRUPTORES%
32  </div>
```

Fonte: Próprio do autor.

Figura 15 – Exemplo *Javascript* para adquirir nível de Soro.

```
setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("SORO").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/SORO", true);
  xhttp.send();
},10000) ;
```

Fonte: Próprio do autor.

3.2.4. Aviso ao responsável da saúde

Para manter o profissional de saúde a par da situação do paciente, a página de monitoramento apresenta dois avisos: um sobre o fim do soro sendo medido e outro para qualquer tipo de ajuda que o paciente necessite, sendo o último ativado por um botão. Como o paciente e o profissional estarão conectados ao mesmo servidor, porém com clientes diferentes, é necessário que a variável responsável por ativar estes avisos esteja no controlador ao invés da página *web*, evitando que o mesmo apareça apenas para um dos usuários.

Dessa forma, foi adicionado ao código uma checagem a estes avisos, chamada de "flag_ajuda1" (**Figura 16**) e "flag_ajuda2" (**Figura 17**), em conjunto com uma solicitação HTTP feita pela página de monitoramento (**Figura 18 e Figura 19**), que pede o estado destas variáveis periodicamente ao ESP e atualizando o HTML com sua resposta.

Figura 16 – a) "flag_ajuda1" – b) resposta ao servidor.

```
if ((peso<=0) || (percent<=0)){
  peso=0;
  percent=0;
  flag_ajudal = 1;
}
else{
  flag_ajudal = 0;
}
```

(a)

```
String text_ajuda(){
  if (flag_ajudal == 1){
    return "<p class=\"blink\"> Necessario trocar soro </p>";
    }
    if (flag_ajudal == 0){
    return " ";
    }
}
```

(b)

Fonte: Próprio do autor.

Figura 17 – "flag_ajuda2" com sua checagem.

```
String text_ajuda_geral(){
  flag_ajuda2 = flag_ajuda2+1;
  if (flag_ajuda2 == 2){
    flag_ajuda2 = 0;
    }

  if (flag_ajuda2 == 1){
    return "<p class=\"blink\"> ASSISTENCIA NECESSARIA </p>";
    }
    if (flag_ajuda2 == 0){
    return " ";
    }
}

String check_ajuda_geral(){
  if (flag_ajuda2 == 1){
    return "<p class=\"blink\"> ASSISTENCIA NECESSARIA </p>";
    }
    if (flag_ajuda2 == 0){
    return " ";
    }
}
```

Fonte: Próprio do autor.

Figura 18 – Rotas HTTP para as flags.

```
server.on("/AJUDASORO", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send_P(200, "text/plain",text_ajuda().c_str());
});

server.on("/AJUDAGERAL", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send_P(200, "text/plain",text_ajuda_geral().c_str());
});

server.on("/AJUDAGERAL-CHECK", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send_P(200, "text/plain",check_ajuda_geral().c_str());
});
```

Fonte: Próprio do autor.

Figura 19 – Funções *Javascript* para que solicitam as rotas.

```
setInterval(function ( ) {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        document.getElementById("AJUDA-SORO").innerHTML = this.responseText;
      }
    };
    xhttp.open("GET", "/AJUDASORO", true);
    xhttp.send();
  },10000) ;

  setInterval(function ( ) {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        document.getElementById("AJUDA-GERAL").innerHTML = this.responseText;
      }
    };
    xhttp.open("GET", "/AJUDAGERAL-CHECK", true);
    xhttp.send();
  },10000) ;


  function generate_ajuda(element){
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        document.getElementById("AJUDA-GERAL").innerHTML = this.responseText;
      }
    };
     xhttp.open("GET", "/AJUDAGERAL", true);
    xhttp.send();
  }
```

Fonte: Próprio do autor.

3.2.5. Sistema de uso do Cartão SD

Uma das funcionalidades pensadas para este projeto que apresentou grande versatilidade foi a integração de uma entrada para cartões de memória. Para tal funcionalidade ser executada, foi necessário o uso da biblioteca "SD,h", que contém diversos exemplos de como utilizá-la, o que foi usado como base para todo o funcionamento deste sistema. Os exemplos em questão contêm comandos gerais em como controlar o cartão de memória, com funções de leitura e criação de diretórios e arquivos, as quais foram modificadas para o código geral do projeto.

Em primeiro lugar o programa verifica se há algum cartão inserido na placa, avisando se houve algum erro na sua identificação. Clicando na página HTML

de ficha do paciente para salvar os dados faz com que o programa ative 3 funções:

- listDir_1: lista o diretório raiz do cartão, verificando se as pastas "Ficha" e "Variáveis" e existem. Se as mesmas não existem, são criadas pela função "createDir", que é chamada dentro da "listDir_1". Caso já tenham sido criadas, nada é feito. O código desta função pode ser visto na **Figura 20** e **Figura 21**;
- listDir_2: lista o diretório "Ficha" e verifica qual o nome das pastas dentro do mesmo. A estrutura utilizada tanto no diretório "Ficha" quanto no "Variáveis" consiste em pastas enumeradas indicando qual a ordem de uso da função. Portanto, por exemplo, se já existem 3 pastas, nomeadas "1", "2" e "3", a pasta "4" é criada e a ficha com as informações de nome, idade, queixa e diagnóstico de cada paciente é salva dentro deste diretório, sendo providas pela página HTML através das rotas HTTP criadas anteriormente (**Figura 22** e **Figura 23**);
- listDir_3: faz o mesmo processo que a função descrita anteriormente, porém no diretório "Variáveis", checando o contador interno e criando pastas enumeradas também, se diferenciando em salvar arquivos separados para cada item da ficha ao invés de um único arquivo de texto, como na função anterior (**Figura 24** e **Figura 25**).

Agora, clicando no botão "Carregar" contido na página HTML, ativa a seguinte função:

- listDir_4: lista o diretório "Variáveis", carregando o conteúdo de cada um dos arquivos contendo as informações da ficha que estão dentro do último diretório enumerado criado, carregando em variáveis internas do programa e enviadas ao servidor utilizando as rotas HTTP feitas previamente (**Figura 26** e **Figura 27**).

Para ativar as funções descritas anteriormente, duas funções gerais foram criadas: "salvar_cartao()" e "carregar_cartao()", as quais podem ser vistas na **Figura 28**.

Figura 20 – listDir_1 primeira parte

```
void listDir_1(fs::FS &fs, const char * dirname, uint8_t levels){
  counter = 0;
  Serial.printf("Listando diretorio: %s\n", dirname);

  File root = fs.open(dirname);
  if(!root){
    Serial.println("Falha ao abrir diretorio");
    erro_display1 = 10;
    return;
  }
  if(!root.isDirectory()){
    Serial.println("Diretorio nao encontrado");
    erro_display1 = 10;
    return;
  }

  File file = root.openNextFile();
  while(file){
    if(file.isDirectory()){
      counter = counter + 1;
      Serial.print("  DIR : ");
      Serial.println(file.name());
      if (counter == 2){
        dir_name1=String(file.name());
        Serial.print(dir_name1);
      }
```

Fonte: Próprio do autor.

Figura 21 – listDir_1 segunda parte

```
        if (counter == 3){
          dir_name2=String(file.name());
          Serial.print(dir_name2);
        }


      }
      file = root.openNextFile();
    }

    if (dir_name1 != "Ficha") {
       createDir(SD, "/Ficha");
    }

    if (dir_name2 != "Variaveis") {
       createDir(SD, "/Variaveis");
    }
  }
```

Fonte: Próprio do autor.

Figura 22 – listDir_2 primeira parte.

```
void listDir_2(fs::FS &fs, const char * dirname, uint8_t levels){
  counter_2 = 0;
  Serial.printf("Listando diretorio: %s\n", dirname);

  File root = fs.open(dirname);
  if(!root){
    Serial.println("Falha ao abrir diretorio");
    erro_display1 = 10;
    return;
  }
  if(!root.isDirectory()){
    Serial.println("Diretorio nao encontrado");
    erro_display1 = 10;
    return;
  }

  File file = root.openNextFile();
  while(file){
    if(file.isDirectory()){
      Serial.print("  DIR : ");
      Serial.println(file.name());
      counter_2 = String(file.name()).toInt();
    }

    file = root.openNextFile();
  }
```

Fonte: Próprio do autor.

Figura 23 – listDir_2 segunda parte.

```
  Serial.print("  Contador: ");
  Serial.println(counter_2);

    int next_folder = counter_2 + 1;
    String dir_next_folder = "/Ficha/" + String(next_folder);

    String var1 = dir_next_folder + "/ficha.txt";
    const char* char_var1 = var1.c_str();

    const char* dir_next_folder_char = dir_next_folder.c_str();
    createDir(SD, dir_next_folder_char);
    writeFile_2(SD, char_var1, "Ficha do Paciente:");

}
```

Fonte: Próprio do autor.

Figura 24 – listDir_3 primeira parte.

```
void listDir_3(fs::FS &fs, const char * dirname, uint8_t levels){
  counter_2 = 0;
  Serial.printf("Listando diretorio: %s\n", dirname);

  File root = fs.open(dirname);
  if(!root){
    Serial.println("Falha ao abrir diretorio");
    erro_display1 = 10;
    return;
  }
  if(!root.isDirectory()){
    Serial.println("Diretorio nao encontrado");
    erro_display1 = 10;
    return;
  }

  File file = root.openNextFile();
  while(file){
    if(file.isDirectory()){
      Serial.print("  DIR : ");
      Serial.println(file.name());
      counter_2 = String(file.name()).toInt();
    }

    file = root.openNextFile();
  }
```

Fonte: Próprio do autor.

Figura 25 – listDir_3 segunda parte.

```
    Serial.print("  Contador: ");
    Serial.println(counter_2);
       int next_folder = counter_2 + 1;
       String dir_next_folder = "/Variaveis/" + String(next_folder);

       String var1 = dir_next_folder + "/nome.txt";
       String var2 = dir_next_folder + "/idade.txt";
       String var3 = dir_next_folder + "/queixas.txt";
       String var4 = dir_next_folder + "/diagnostico.txt";

       const char* char_var1 = var1.c_str();
       const char* char_var2 = var2.c_str();
       const char* char_var3 = var3.c_str();
       const char* char_var4 = var4.c_str();

       const char* char_inputMessage4 = inputMessage4.c_str();
       const char* char_inputMessage5 = inputMessage5.c_str();
       const char* char_inputMessage6 = inputMessage6.c_str();
       const char* char_inputMessage7 = inputMessage7.c_str();

       const char* dir_next_folder_char = dir_next_folder.c_str();
       createDir(SD, dir_next_folder_char);
       writeFile_1(SD, char_var1, char_inputMessage4);
       writeFile_1(SD, char_var2, char_inputMessage5);
       writeFile_1(SD, char_var3, char_inputMessage6);
       writeFile_1(SD, char_var4, char_inputMessage7);

}
```

Fonte: Próprio do autor.

Figura 26 – listDir_4 primeira parte.

```
void listDir_4(fs::FS &fs, const char * dirname, uint8_t levels){
  counter_2 = 0;
  Serial.printf("Listando diretorio: %s\n", dirname);

  File root = fs.open(dirname);
  if(!root){
    Serial.println("Falha ao abrir diretorio");
    erro_display2 = 100;
    return;
  }
  if(!root.isDirectory()){
    Serial.println("Diretorio nao encontrado");
    erro_display2 = 100;
    return;
  }

  File file = root.openNextFile();
  while(file){
    if(file.isDirectory()){
      Serial.print("  DIR : ");
      Serial.println(file.name());
      counter_2 = String(file.name()).toInt();
    }

    file = root.openNextFile();
  }
  Serial.print("  Contador: ");
  Serial.println(counter_2);
```

Fonte: Próprio do autor.

Figura 27 – listDir_4 segunda parte.

```
    String dir_next_folder = "/Variaveis/" + String(counter_2);

    String var1 = dir_next_folder + "/nome.txt";
    String var2 = dir_next_folder + "/idade.txt";
    String var3 = dir_next_folder + "/queixas.txt";
    String var4 = dir_next_folder + "/diagnostico.txt";

    const char* char_var1 = var1.c_str();
    const char* char_var2 = var2.c_str();
    const char* char_var3 = var3.c_str();
    const char* char_var4 = var4.c_str();
    readFile(SD, char_var1);
    load_SD1 = load_SD;
    Serial.print(load_SD1);
    readFile(SD, char_var2);
    load_SD2 = load_SD;
    Serial.print(load_SD2);
    readFile(SD, char_var3);
    load_SD3 = load_SD;
    Serial.print(load_SD3);
    readFile(SD, char_var4);
    load_SD4 = load_SD;
    Serial.print(load_SD4);

}
```

Fonte: Próprio do autor.

Figura 28 – Funções para salvar e carregar arquivos do cartão.

```
//Função que salva os dados da ficha no cartão, contendo sub funções responsáveis por verificar se os diretórios já existem ou se já existem arquivos nos mesmos
//Sub funções: listDir_1, listDir_2, listDir_3, creatDir, writeFile_1 e writeFile_2
void salvar_cartao(){
   listDir_1(SD, "/", 0);
   listDir_2(SD, "/Ficha", 0);
   listDir_3(SD, "/Variaveis", 0);
   clean_screen();
   error_check();

}

//Função para carregar os dados da ficha que estão no cartão SD
//Sub função: listDir_4, readFile
void carregar_cartao(){
  listDir_4(SD, "/Variaveis", 0);
  clean_screen();
  error_check();
```

Fonte: Próprio do autor.

3.2.6. Ativação dos Relés

Uma das formas de interação do paciente com o ambiente foi alcançada a partir do controle de 3 relés conectados à rede (110V), funcionando como "tomadas inteligentes", as quais fornecem a alimentação da rede doméstica a partir de um controle remoto. A programação para obter tal funcionalidade pode ser contextualizada em apenas na mudança de estado de portas I/Os do ESP32, utilizando o comando básico de "pinMode" (configura se uma porta se comporta como entrada ou saída, como mostrado na **Figura 29**) e "digitalWrite" (muda o estado para o desejado em uma porta configurada como saída). Porém, a necessidade de fazer este controle por meio de um navegador fez com que tal comando de "digitalWrite" seja vinculado à resposta obtida pela rota HTTP (**Figura 30)** ao se clicar no botão dedicado a ativar ou desativar tais relés, sendo os mesmos criados com código do HTML dentro do código principal do controlador que é enviado a página e inserido no lugar de um marcador, como demonstrado na **Figura 31** e **Figura 32**.

Figura 29 – Configuração das portas como saída.

```
//Saida para os relés
pinMode(12, OUTPUT);
digitalWrite(12, LOW);
pinMode(16, OUTPUT);
digitalWrite(16, LOW);
pinMode(17, OUTPUT);
digitalWrite(17, LOW);
```

Fonte: Próprio do autor.

Figura 30 – Rota HTTP para mudar o estado das saídas.

```
server.on("/update", HTTP_GET, [] (AsyncWebServerRequest *request) {
  String inputMessage1;
  String inputMessage2;
  //"url/update?output=<inputMessage1>&state=<inputMessage2>"
  if (request->hasParam(PARAM_INPUT_1) && request->hasParam(PARAM_INPUT_2)) {
    inputMessage1 = request->getParam(PARAM_INPUT_1)->value();
    inputMessage2 = request->getParam(PARAM_INPUT_2)->value();
    digitalWrite(inputMessage1.toInt(), inputMessage2.toInt());
  }
  Serial.println("");
  Serial.print("Rele: ");
  Serial.print(inputMessage1);
  Serial.print(" - Setado em: ");
  Serial.println(inputMessage2);
  request->send(200, "text/plain", "OK");
});
```

Fonte: Próprio do autor.

Figura 31 – Resposta ao marcador com código HTML.

```
if(var == "INTERRUPTORES"){
  String buttons = "";
  buttons += "<h4>Interruptor 1</h4><label class=\"button_switch\"><input type=\"checkbox\" onchange=\"toggle_button_switch(this)\" id=\"17\" " + outputState(17) + "><span class=\"button_sli
  buttons += "<h4>Interruptor 2</h4><label class=\"button_switch\"><input type=\"checkbox\" onchange=\"toggle_button_switch(this)\" id=\"16\" " + outputState(16) + "><span class=\"button_sli
  buttons += "<h4>Interruptor 3</h4><label class=\"button_switch\"><input type=\"checkbox\" onchange=\"toggle_button_switch(this)\" id=\"12\" " + outputState(12) + "><span class=\"button_sli
  return buttons;
}
```

Fonte: Próprio do autor.

Figura 32 – Função para mudar o estado visual do botão.

```
String outputState(int output){
  if(digitalRead(output)){
    return "checked";
  }
  else {
    return "";
  }
}
```

Fonte: Próprio do autor.

3.2.7. Utilização do *Robotdyn dimmer*

A programação do *dimmer* só foi alcançada devido a biblioteca "RBDdimmer.h" e seus exemplos que abrangem uma diversa gama de microcontroladores. Seguindo o que é descrito nestes exemplos, foi necessário apenas fornecer as informações de quais portas seriam responsáveis pelo pino de reconhecimento *zero crossing* e qual recebe o sinal de controle do triac, recebendo a porcentagem de luminosidade do *slider* contido na página de monitoramento (**Figura 34** e **Figura 35**). No entanto, testes preliminares demonstraram que, apesar de controlar o brilho da lâmpada como previsto, a mesma ficava piscando intermitentemente. Isso ocorre devido a sensibilidade do módulo *dimmer*, fazendo com que todo o processamento do resto do código interferisse com a execução desta tarefa. A solução para este problema foi de utilizar a função de fixar uma tarefa em um único núcleo de processamento do ESP32 e aumentando seu nível de importância, fazendo com que tal tarefa fosse executada separadamente sem interrupções da execução do código principal, como mostrado na **Figura 33**.

Figura 33 – Configuração da utilização de um *core* para o *dimmer*.

```
//Configura um core para somente realizar o funcionamento do dimmer
xTaskCreatePinnedToCore(
                  dimmer_core,
                  "dimer_core",
                  10000,
                  NULL,
                  1,
                  NULL,
                  1);

}
```

Fonte: Próprio do autor.

Figura 34 – Função que controla o *dimmer*.

```
//Funcionamento do dimmer em um core separado
void dimmer_core( void * pvParameters ){

    while(true){
      if (sliderValue2 != slider){
      Serial.print(sliderValue2);
      dimmer.setPower(sliderValue2);
      slider = sliderValue2;
      }
      else{
      dimmer.getPower();
      }

    }
}
```

Fonte: Próprio do autor.

Figura 35 – Código *javascript* para o *slider* determinar a luminosidade.

```
63  function updateSliderPWM(element) {
64    var slider_Value = document.getElementById("pwmSlider").value;
65    document.getElementById("light_slider_value").innerHTML = slider_Value;
66    console.log(slider_Value);
67    var xhr = new XMLHttpRequest();
68    xhr.open("GET", "/slider?value="+slider_Value, true);
69    xhr.send();
70  }
```

Fonte: Próprio do autor.

3.2.8. Checagem de Erro e tela OLED

Com o intuito de manter o usuário informado de possíveis problemas ao longo do código, foi feito um sistema de verificação, que enumera e identifica qual parte do programa apresentou erros. Para fazer isso, foram criadas 3 variáveis:

- "erro_display": variável inserida na inicialização do cartão de memória, sendo dado o valor "1" caso não seja possível inicializar o mesmo, informação determinada pela biblioteca de interface com o cartão;
- "erro_display1": variável inserida logo após qualquer leitura de diretório feita ao tentar carregar a ficha pela página, sendo dado o valor "10" caso não seja possível listar o mesmo, também fornecido pela mesma biblioteca citada anteriormente;
- "erro_display2": variável inserida logo após qualquer leitura de diretório ou arquivo feita ao tentar salvar a ficha pela página, sendo dado o valor "100" caso não seja possível listar o mesmo, fornecido pela biblioteca do cartão.

Dessa forma, estes valores são somados e são analisados apenas 4 das possíveis combinações, gerando respostas de erro presentes na **Tabela 1**.

Tabela 1 - Combinações de avisos

Valor da variável	Erro mostrado na tela
001	"SD não reconhecido"
010	"Falha ler arquivos"
100	"Falha salvar arquivos"
110	"Falha ler arquivos" "Falha salvar arquivos"

Fonte: Próprio do autor.

Como é possível observar na **Tabela 1**, o valor "001" demonstra falha ao reconhecer o cartão de memória, o que faz com que qualquer combinação com este número seja redundante, pois a leitura e gravação de dados em um cartão terá falha se o mesmo não for reconhecido. Porém, é possível um cartão ser reconhecido e apresentar problemas de gravação, de leitura ou ambos os erros, fazendo com que os números "010", "100" e "110" sejam analisados. Esta análise é feita dentro de um "*switch – case*", uma declaração lógica de várias condições que permite a saída rápida do mesmo caso o número analisado seja igual a condição dada, observada na **Figura 36** e **Figura 37**.

Essas informações são apresentadas pela tela OLED de 1.3 polegadas, controlada pelas bibliotecas "Adafruit_GFX.h" e "Adafruit_SSD1306.h". Inicialmente, caso a conexão com a rede WiFi funcione, é apresentado o número IP e URL específica do quarto fornecido, sendo a parte inferior deste display focada para avisar os erros mencionados anteriormente. Para que seja possível atualizar esta parte de erros sem a necessidade limpar a tela inteira, foi criada a função "*clean_screen()*", que desenha um retângulo preto em cima desta mesma parte, efetivamente apagando o que estava sendo mostrado. De qualquer forma, os erros citados anteriormente são mostrados no painel *serial* do Arduino IDE, mostrando em detalhes outros erros que não aparecem na tela OLED.

Figura 36 – "error_check()" primeira parte.

```
//Função responsável por chegar por erros e mostrar no display OLED
void error_check(){
  erro_display = erro_display + erro_display1 + erro_display2;
  switch (erro_display) {
  case 001:
    display.setCursor(0,25);
    display.println("");
    display.print("Erro:");
    display.println("");
    display.print("SD nao reconhecido");
    break;
  case 010:
    display.setCursor(0,25);
    display.println("");
    display.print("Erro:");
    display.println("");
    display.print("Falha ler arquivos");
    break;
```

Fonte: Próprio do autor.

Figura 37 – "error_check()" segunda parte.

```
 case 100:
   display.setCursor(0,25);
   display.println("");
   display.print("Erro:");
   display.println("");
   display.print("Falha salvar arquivos");
   break;
 case 110:
   display.setCursor(0,25);
   display.println("");
   display.print("Erro:");
   display.println("");
   display.print("Falha ler arquivos");
   display.println("");
   display.print("Falha salvar arquivos");
   break;

}
display.display();
}
```

Fonte: Próprio do autor.

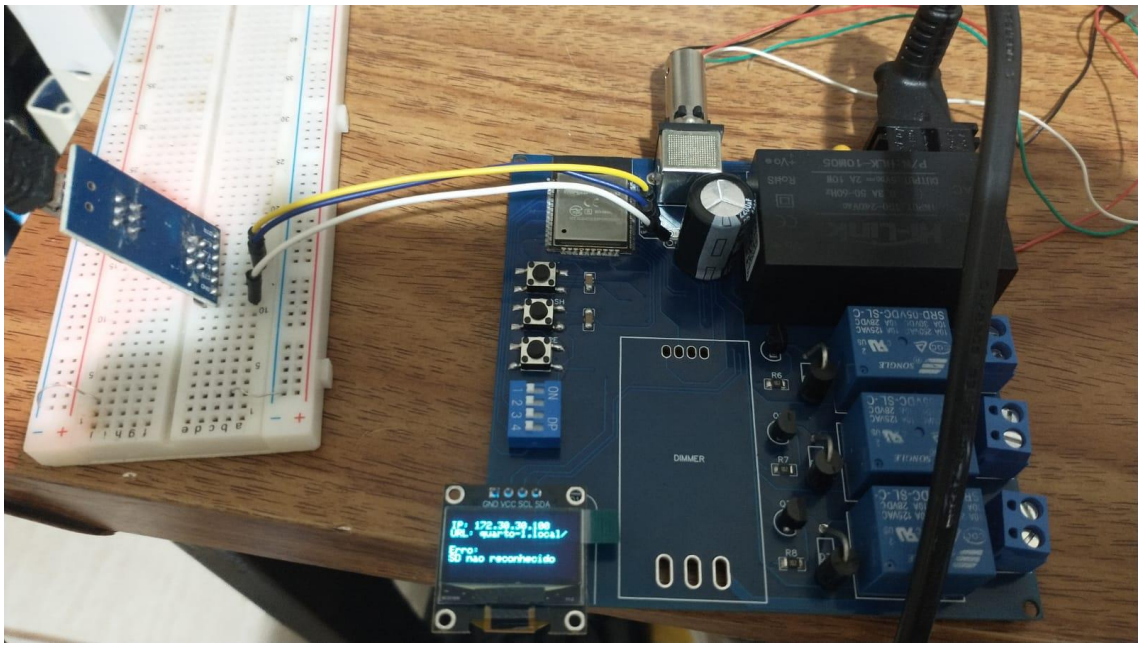
3.3. Criação do circuito geral

Com todos os testes feitos previamente utilizando a *protoboard* e a programação funcionando como previsto, foi executado a próxima etapa deste projeto que consiste na consolidação do mesmo em uma placa de circuito impresso. Para realizar tal feito, foi utilizado como guia o modelo referência fornecido pela Espressif (**Figura 40**) para a construção de kit de desenvolvimento baseado no ESP-WROOM-32. Com ele foi possível entender como a construção de um kit é feita e qual a configuração necessária para deixar um ESP funcional.

Seguindo este guia, o pino EN (3) do controlador foi ligado a um capacitor cerâmico de 100nF conectado ao *ground* e a um resistor *pull-up* de 10kΩ, além de ser conectado a um botão em paralelo a outro capacitor de mesmo valor ligados ao *ground*. Em seguida, foi feita a mesma configuração de capacitor e botão paralelos ligados ao GND, só que desta vez conectados ao pino FLASH (25 ou ID0). Estes dois botões permitem colocar o controlador em modo de gravação, sendo possível enviar o código pelo Arduino IDE. Para isso, primeiro se segura o botão EN (que reseta o ESP), depois segura-se o botão FLASH sem soltar o EN, depois solta-se o EN continuando a pressionar o FLASH. Assim, enquanto o botão FLASH é pressionado, o ESP está em modo de gravação, utilizando um gravador FTDI para realizar a transferência de dados como

mostrado na **Figura 38**, sendo necessário resetá-lo quando o *upload* do código é terminado.

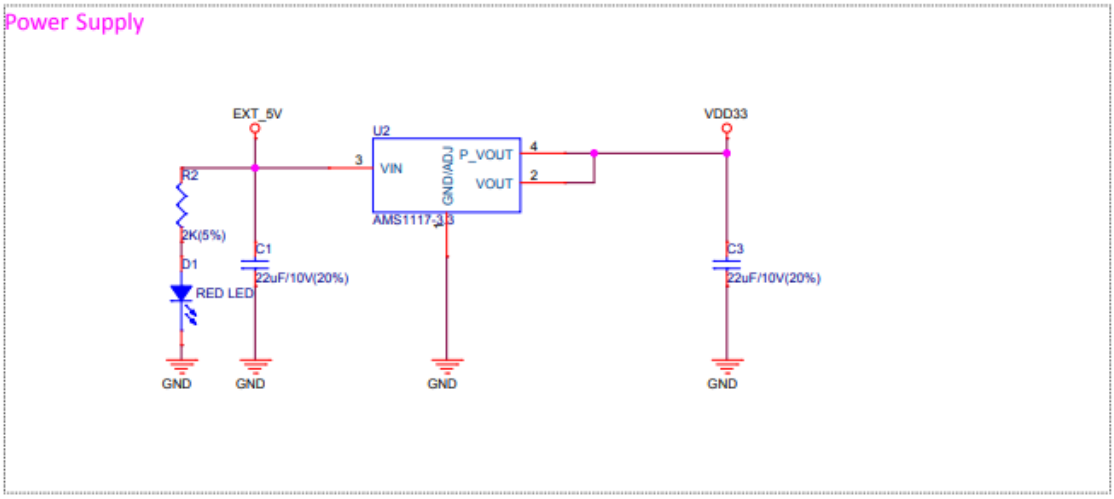
Figura 38 – Upload utilizando o FTDI.



Fonte: Próprio do autor.

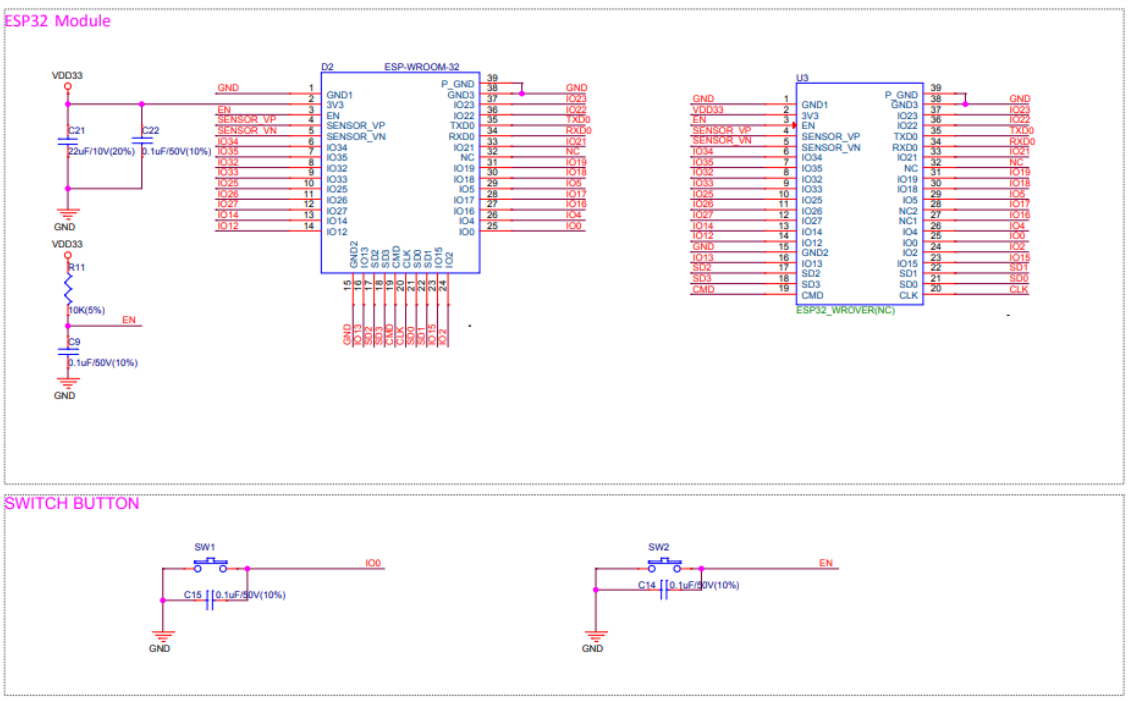
Para permitir uma operação sem erros, foi necessário conectar os pinos 1, 15, 38 e 39 ao *ground* (GND1, GND2, GND3 e P_GND, respectivamente, demonstrado na **Figura 42**) e realizar a alimentação do ESP a partir de um regulador de tensão de 3.3V, AMS1117, com dois capacitores de tântalo de 22uF, um na entrada e outro na saída, para filtrar as tensões, como mostrado na **Figura 39** e **Figura 41**.

Figura 39 – Alimentação recomendada pelo *DevKit*.



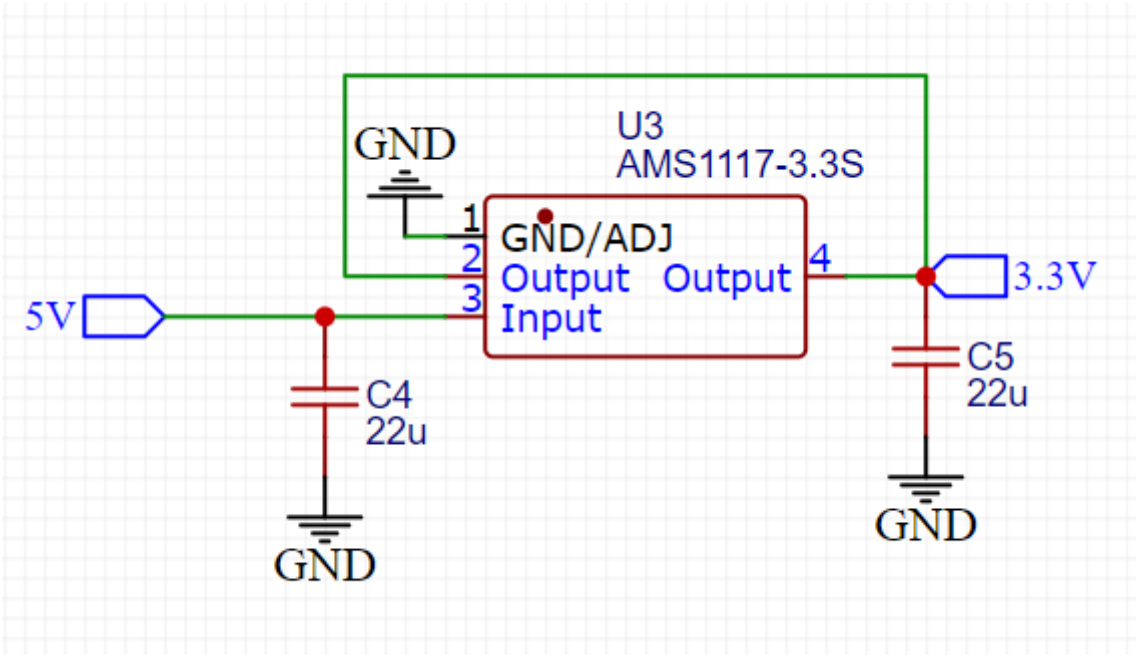
Fonte: (ESPRESSIF, 2017).

Figura 40 – Esquemático *DevKit* ESP-WROOM-32.



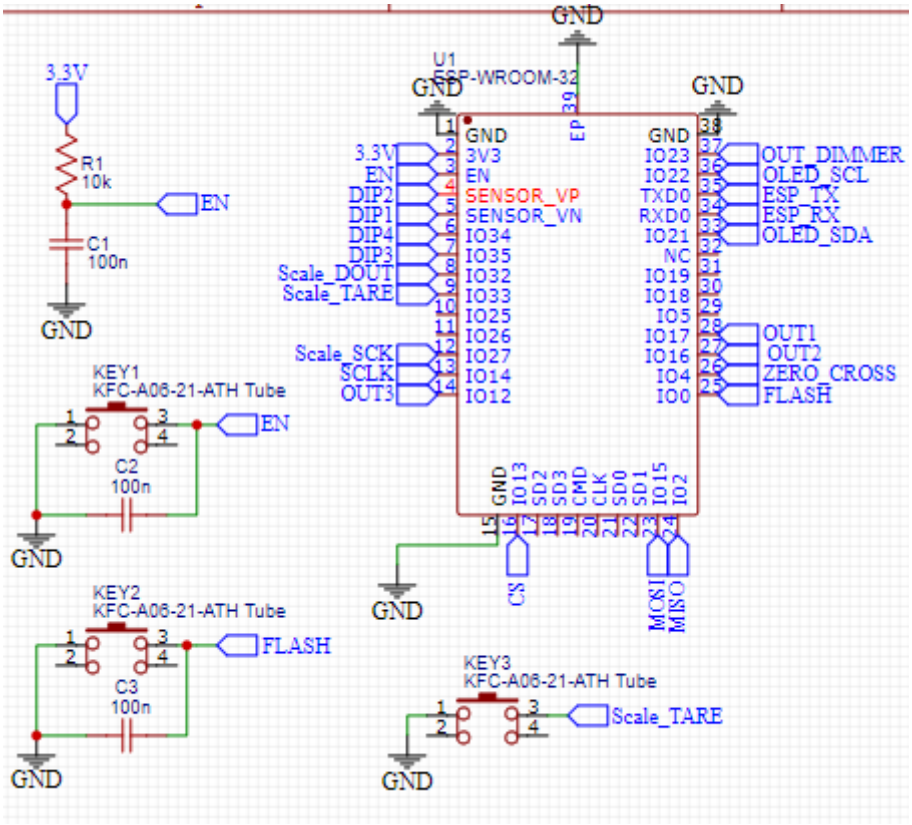
Fonte: (ESPRESSIF, 2017).

Figura 41 – Alimentação do projeto.



Fonte: Próprio do autor.

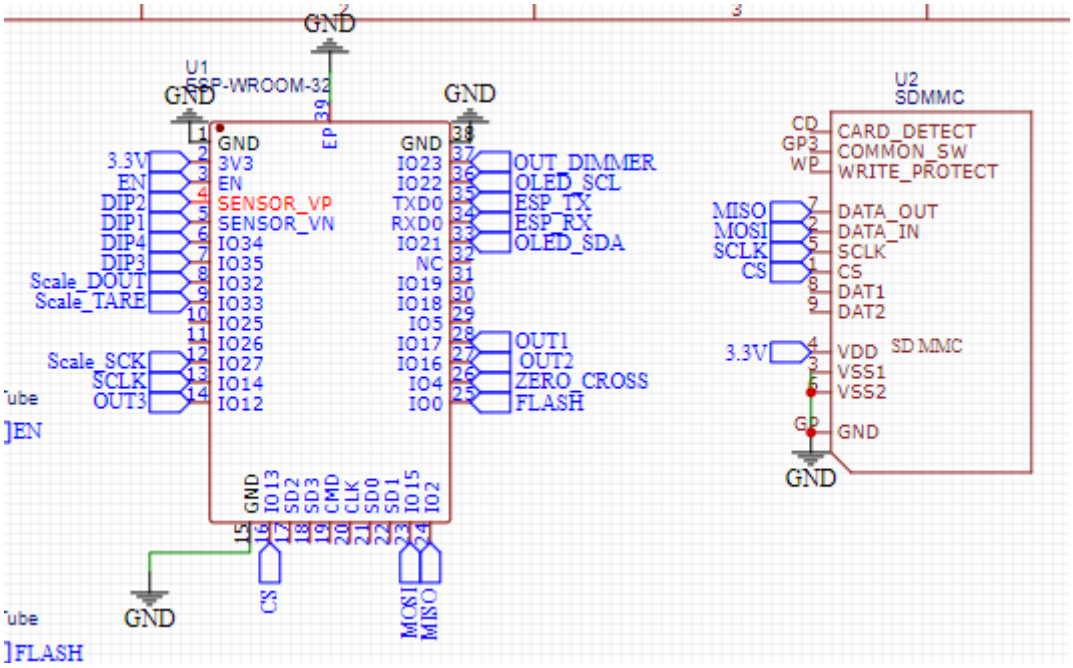
Figura 42 – Circuito ESP.



Fonte: Próprio do autor.

A conexão do conector de cartão de memória foi feita seguindo os pinos indicados pela Espressif, de acordo com a interface SPI os pinos MOSI (*Master Out Slave In*) e MISO (*Master In Slave Out*) foram alocados para a porta IO15 e IO2 respectivamente, CS para a IO13 e SCLK para IO14 (**Figura 43**).

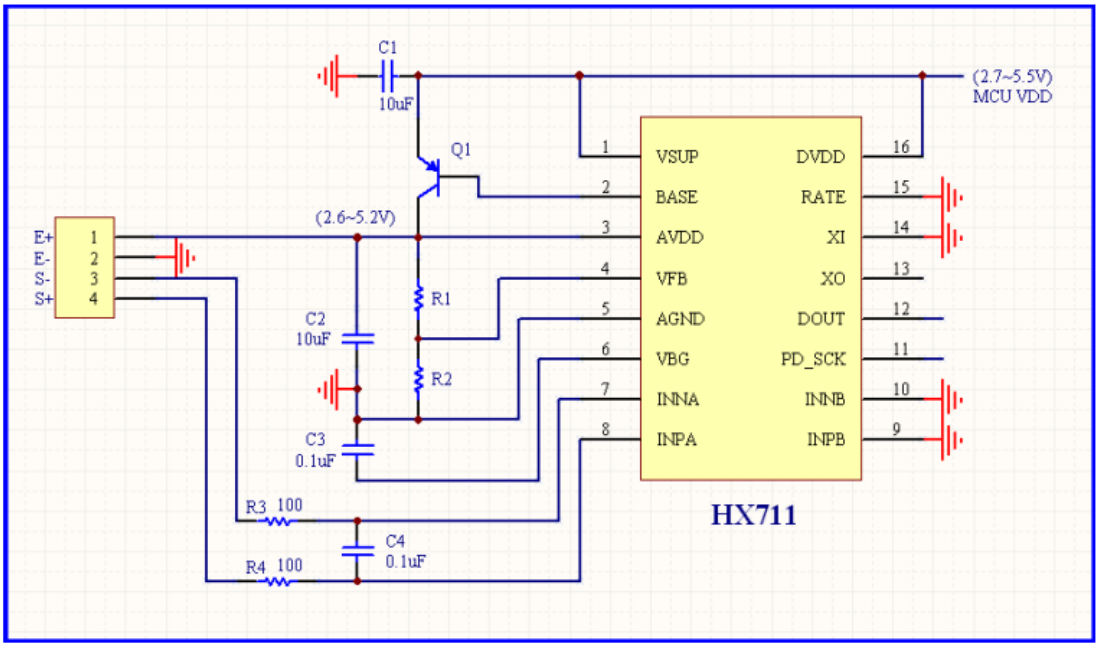
Figura 43 – Circuito cartão de memória.



Fonte: Próprio do autor.

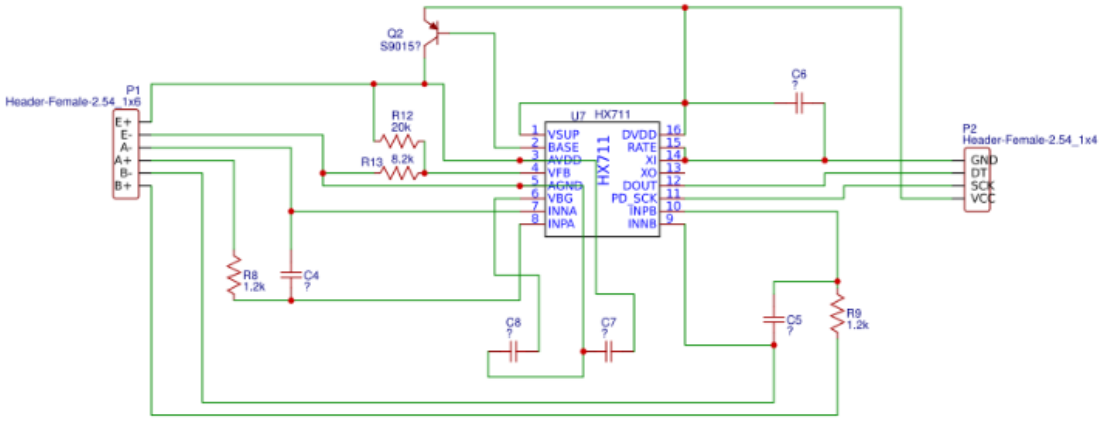
Como descrito anteriormente, a operação da *load cell* só foi possível graças ao módulo conversor analógico para digital baseado no CI HX711, sendo assim necessário a recriação do circuito presente no módulo utilizado nos testes, integrando-o na PCB. A realização de tal feito foi possível graças ao *datasheet* do HX711 que apresenta um circuito exemplo (**Figura 44**) de como operá-lo, além de um circuito encontrado na internet que recria o módulo (**Figura 45**). Ambos foram comparados com o módulo em mãos para determinar se seus componentes e ligações estava de acordo com o existente e que a operação do circuito criado seria plausível, demonstrado na **Figura 46**. Para permitir que a *load cell* seja conectada e desconectada facilmente, sem haver a necessidade de ficar presa permanentemente na placa, foi usado um conector fêmea para PCB Mini Din de 4 pinos, seguindo o *pinout* mostrado na **Figura 47**, enquanto um conector macho foi utilizado na *load cell*.

Figura 44 – Circuito exemplo datasheet HX711.



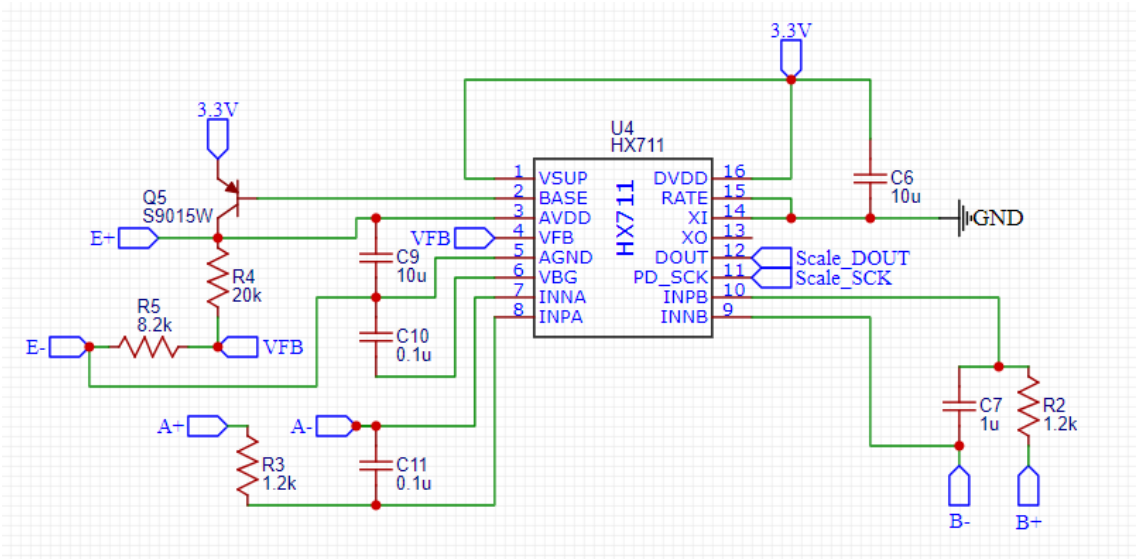
Fonte: (AVIA SEMICONDUCTOR, 2023).

Figura 45 – Réplica do circuito encontrado na internet.



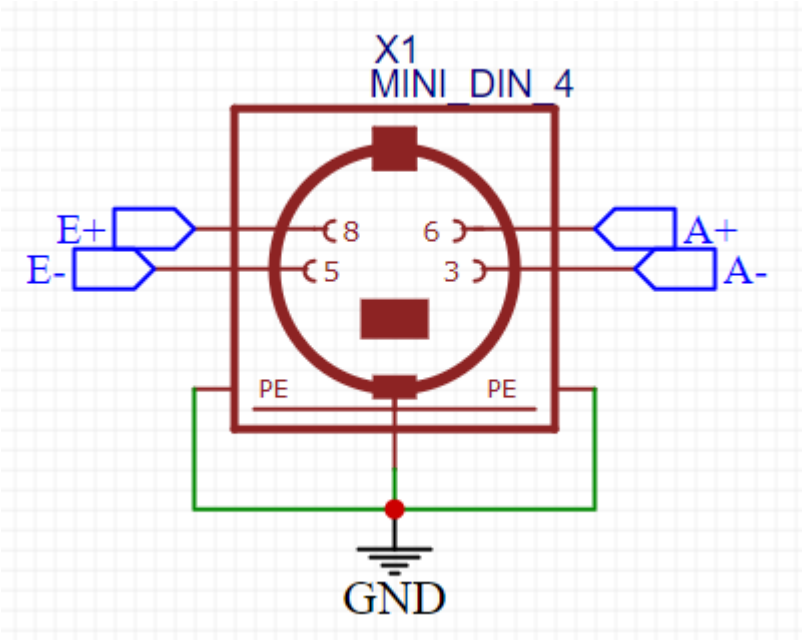
Fonte: (ADAMFK, 2019).

Figura 46 – Circuito criado para o projeto.



Fonte: Próprio do autor.

Figura 47 – *Pinout* Mini Din 4 fêmea.

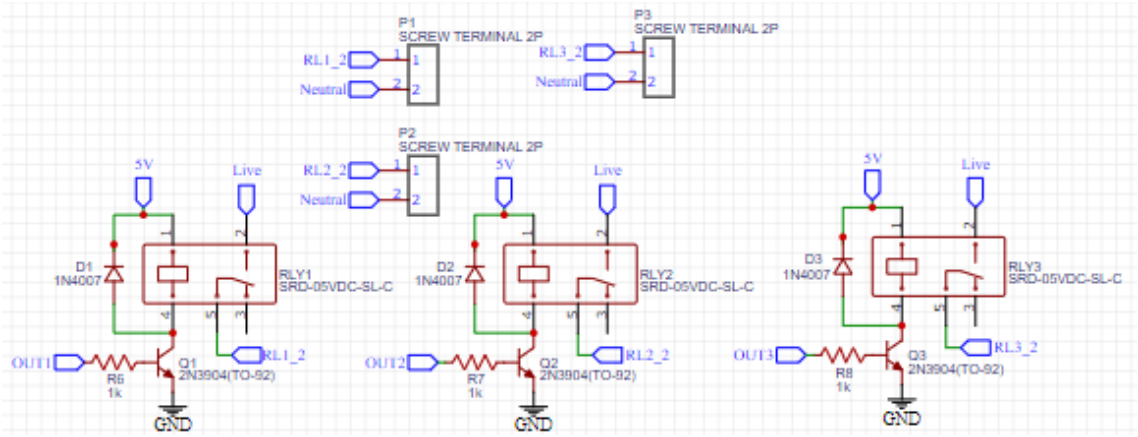


Fonte: Próprio do autor.

Por fim, foram feitos os circuitos para os relés (**Figura 48**), consistindo em 3 transistores em funcionamento de chave, com um resistor de 1KΩ na base gerar uma corrente de ativação, tendo a bobina do relé entre o coletor e a alimentação (5V). Para evitar picos de tensão ao se desligar o relé foi colocado

um diodo flyback em paralelo com a bobina, em sentido contrário a alimentação, permitindo um caminho para dissipar a corrente. Os relés ficam responsáveis como interruptores para o positivo da rede, com o neutro já conectado direto ao *borne* externo.

Figura 48 – Circuito relés.



Fonte: Próprio do autor.

A criação deste circuito geral foi feita pela plataforma *EasyEDA*, que permite não só a montagem do circuito, mas também a criação da PCB. Tal circuito, com suas duas páginas, está disponível no Apêndice E.

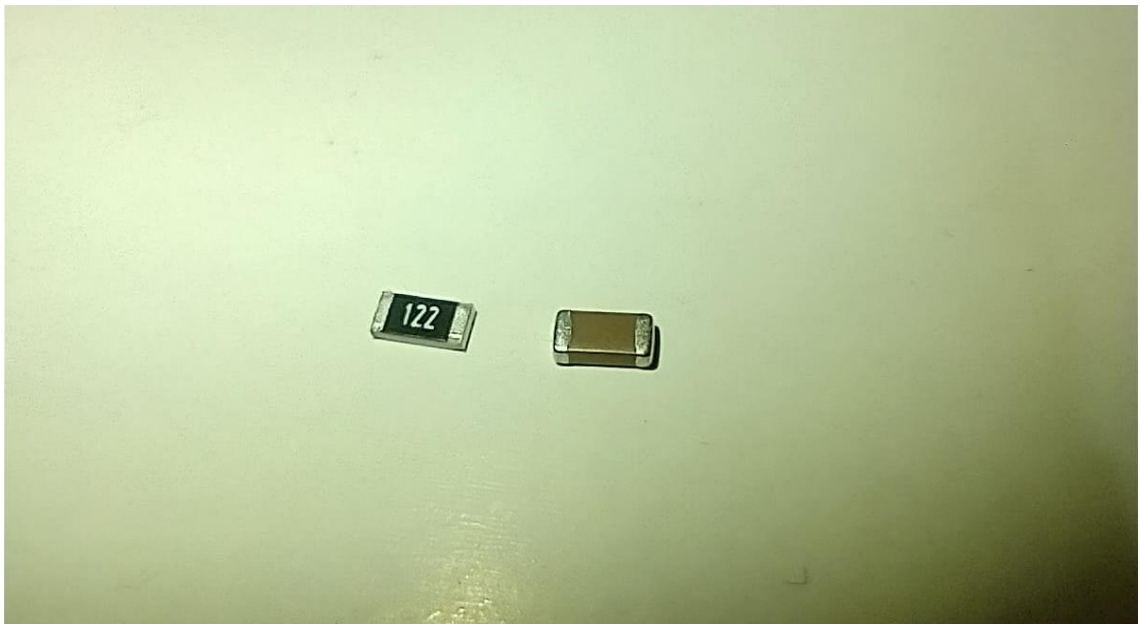
3.4. Criação da PCB

Com o circuito geral finalizado, foi criado a placa para contê-lo. O serviço de fabricação de placas é fornecido pelo site "jlcpcb", feito a partir do envio de arquivos contendo o design das PCBs. Para utilizar o valor mais baixo (2 dólares) de produção, é necessário que a placa tenha uma área de 10x10cm. Portanto, gerenciamento de espaço se tornou o objetivo principal nesse desenvolvimento, visando economizar o máximo de espaço possível ainda mantendo coerência e facilidade no momento de soldagem dos componentes.

Para não ultrapassar este limite de área estabelecido pelo site que fabrica as placas, foi necessário a escolha de componentes SMDs (*surface mount device*), com o tamanho escolhido (*footprint*) 1206 (**Figura 49**) para os componentes passivos, devido ao seu menor tamanho, porém com dimensões grandes o suficiente para facilitar o processo de solda. A maioria dos componentes passivos, CIs e botões foram SMD enquanto os componentes TH (*through hole*) consistem em: relés com seus transistores de ativação e diodos *flyback*,

capacitor de filtragem para 5V, DIP *switch*, conector mini din 4, conector AC, *bornes* e o *display* OLED.

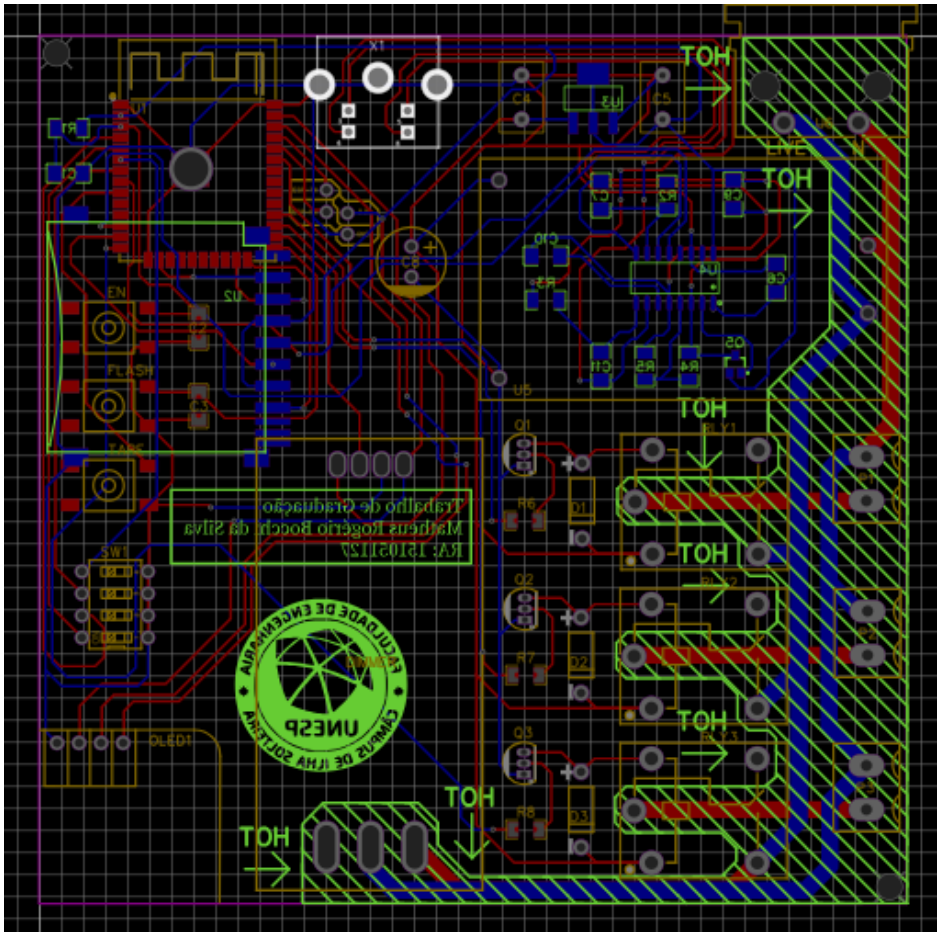
Figura 49 – Resistor e capacitor 1206.



Fonte: Próprio do autor.

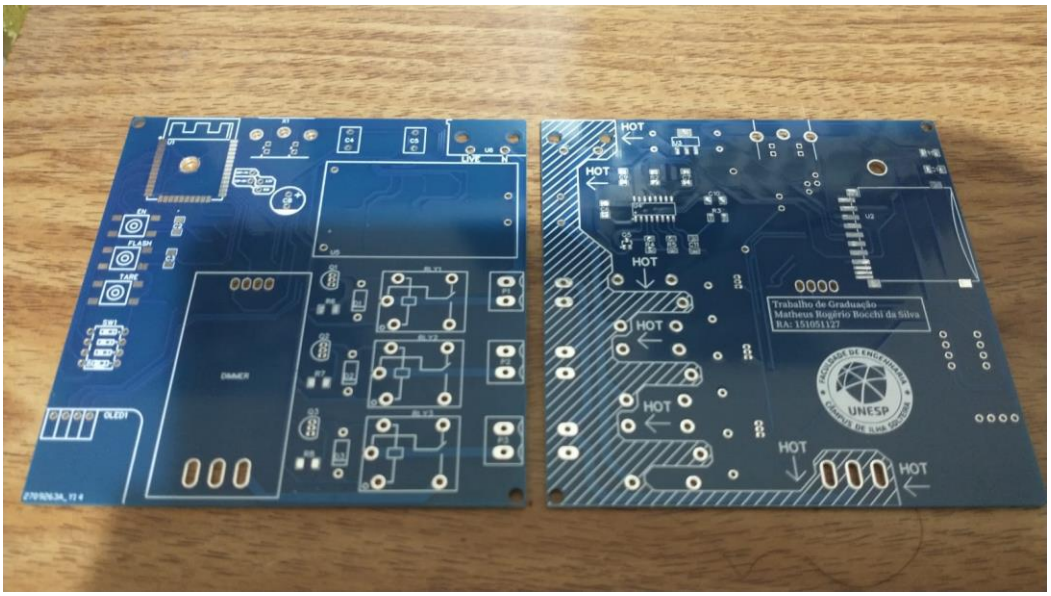
Por último no processo de criação da placa, foi demarcado quais trilhas conduzem corrente alternada, focando em deixá-las o mais próximo possível, para isolar uma única área da placa como parte perigosa. A alimentação AC entra pelo conector de tomada, alimenta o módulo Hi-link, vai para os relés e seus *bornes* e, por fim, chega ao *dimmer*, seguindo assim duas trilhas (positivo e neutro) pela a extremidade direita e inferior da placa. Todos os pontos de contato que passam na placa e ficam expostos, como os pinos dos relés, são incluídos nessa demarcação, com o aviso de "HOT" para demonstrar perigo. A **Figura 50** demonstra o design final da placa, enquanto a **Figura 51** e **Figura 52** demonstram o produto final.

Figura 50 – *Design* da PCB.



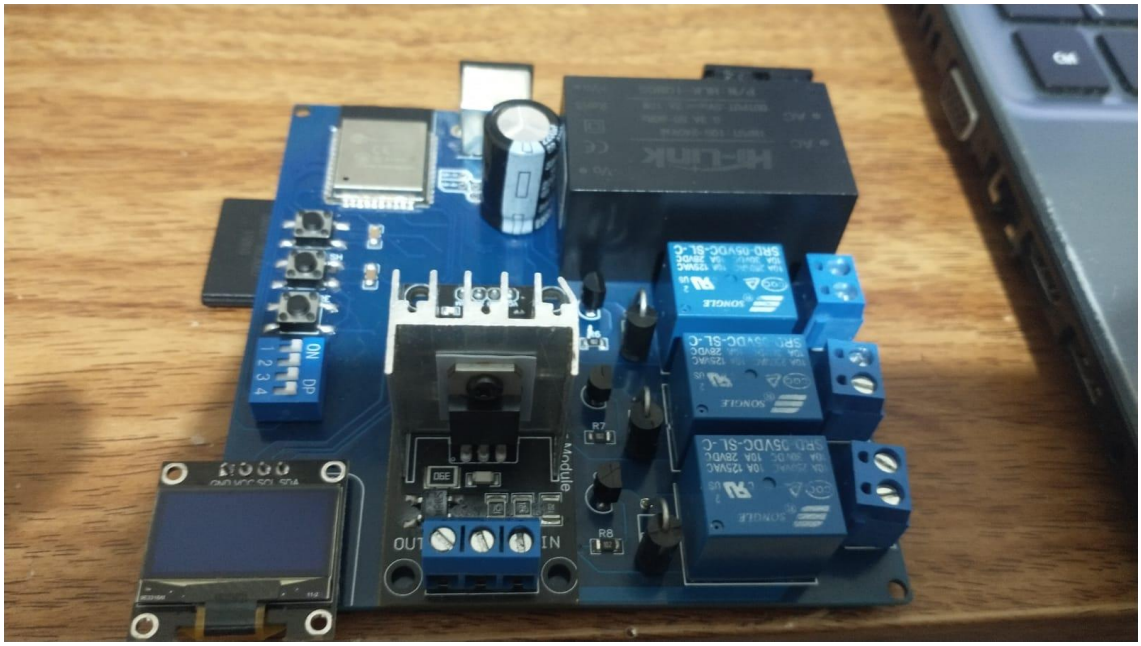
Fonte: Próprio do autor.

Figura 51 – PCB sem os componentes frente e verso.



Fonte: Próprio do autor.

Figura 52 – PCB com componentes soldados.

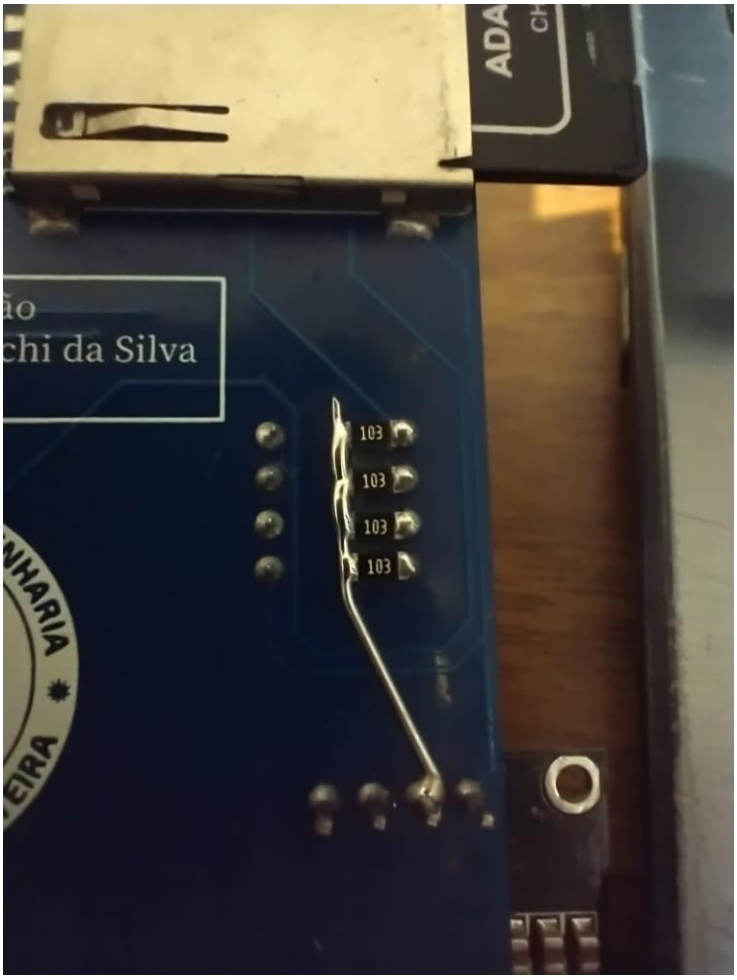


Fonte: Próprio do autor.

4. Resultados

Na sua primeira ativação foi constatado que a funcionalidade de selecionar qual a numeração do quarto em que a placa está presente não estava funcionando, mostrando números esporádicos que não condiziam com o colocado no DIP *switch*. Após inspeções e experimentações com um código separado visando exclusivamente esta funcionalidade, descobriu-se que tal erro ocorreu por falta de resistores *pull-ups* na entrada das portas I/Os. Outras entradas utilizadas na programação foram configuradas com *pull-ups* fornecidos por software, opção disponível em microcontroladores ESP-32, porém, ao utilizar as portas I/O 34, 35, 36 e 39, portas que somente podem ser usadas como entrada, esta opção não estava disponível. Portanto para consertar tal problema foi soldado resistores *pull-ups* de 10kΩ direto nos pinos do DIP *switch* (**Figura 53**), utilizando a alimentação da tela OLED como tensão de *pull-up* devido a sua localização mais próxima ao DIP *switch*.

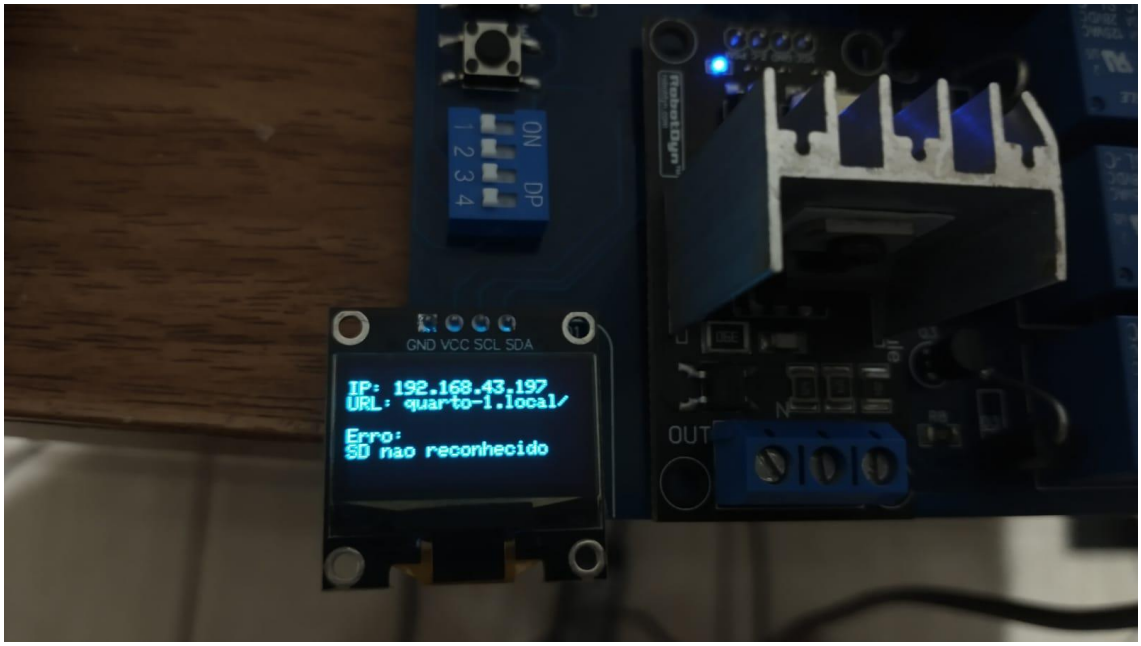
Figura 53 - Resistores *pull-up*.



Fonte: Próprio do autor.

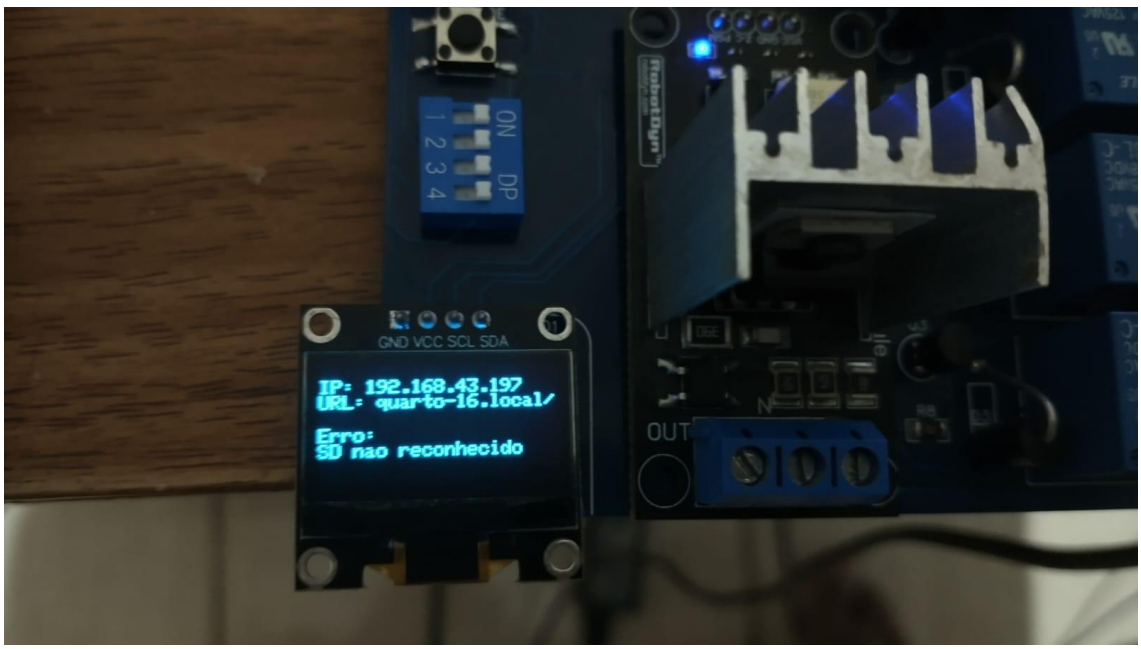
Consertando este problema, o processo de seleção dos quartos funcionou como o previsto. A exemplo disso, o DIP *switch* foi colocado em "000", selecionando o primeiro quarto, como mostrado na **Figura 54**. Com o DIP *switch* em "1111", o último quarto foi selecionado (**Figura 55**), demonstrando toda a faixa de possíveis quartos disponíveis para seleção.

Figura 54 - DIP *switch* quarto 1.



Fonte: Próprio do autor.

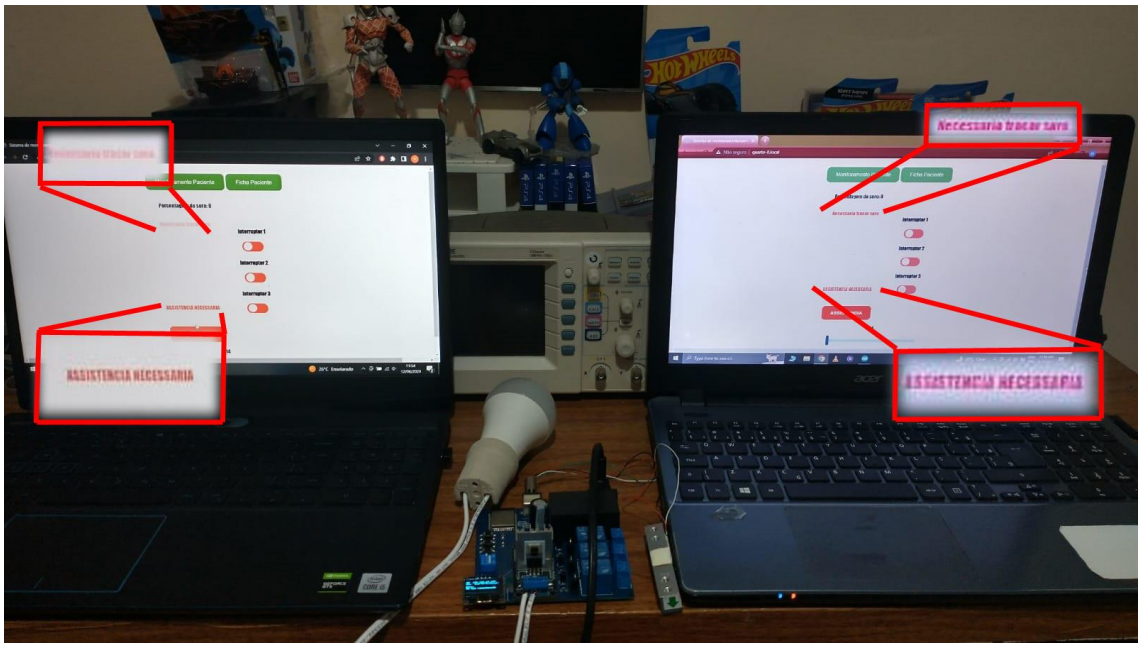
Figura 55 - DIP *switch* quarto 16.



Fonte: Próprio do autor.

Desta forma, a execução das outras funções da placa se apresentou como previsto e desejado. Na **Figura 56** é possível ver o aviso da falta de soro funcionar em dois clientes separados ao mesmo tempo, simulando a utilização pelo paciente e um funcionário do hospital responsável por cuidar do mesmo, além da mesma porcentagem de nível de soro estar sendo mostrada em ambos os navegadores. Não apenas o aviso da falta de soro, na mesma figura é possível observar dois clientes separados mostrando o aviso de emergência, supostamente clicado pelo paciente a espera de ajuda.

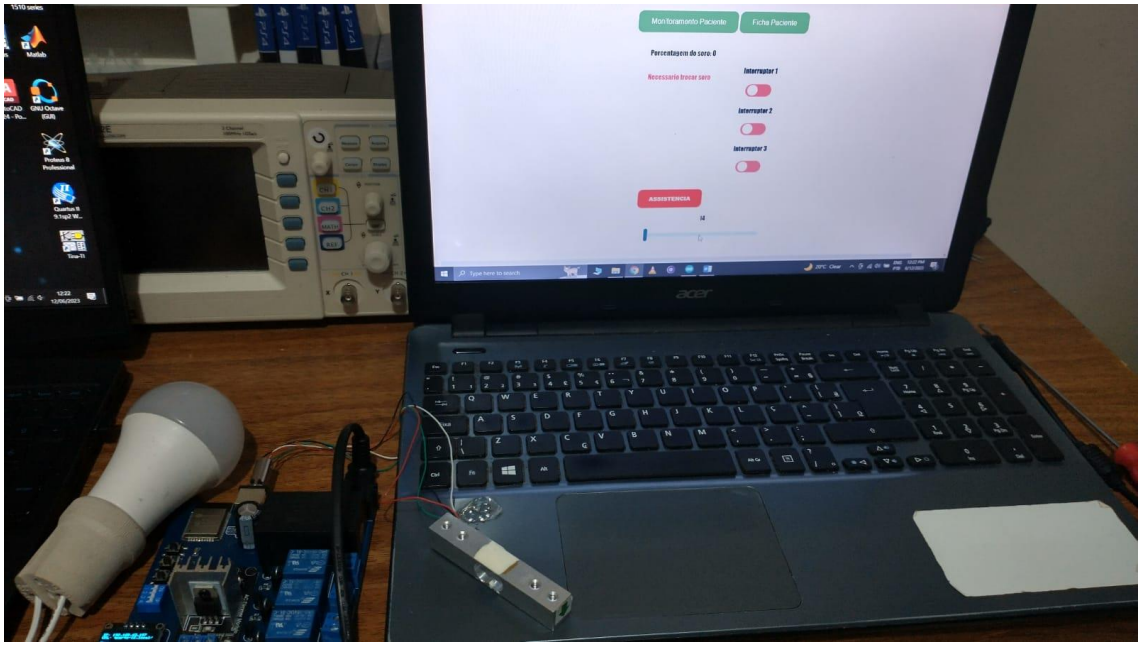
Figura 56 – Dois clientes demonstrando o aviso de falta de soro e emergência.



Fonte: Próprio do autor.

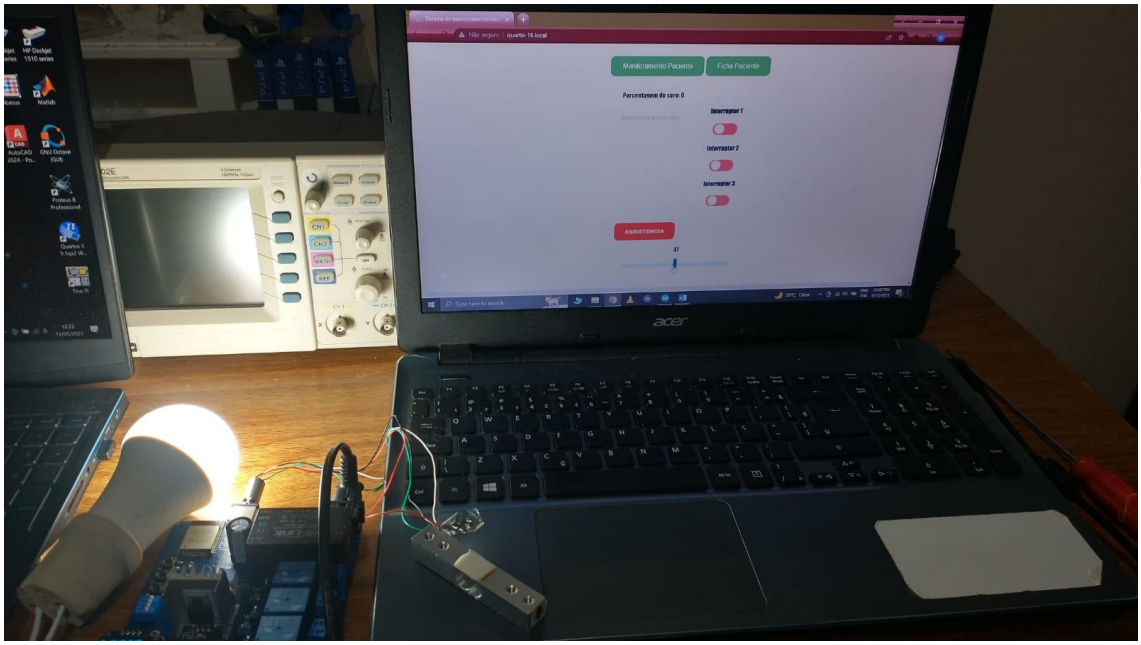
A variação da intensidade de luz também funcionou como previsto, utilizando um núcleo de processamento separado do restante da programação. Na **Figura 57** é possível visualizar a lâmpada de teste simulando a iluminação do quarto com o *slider* da página de monitoramento totalmente a sua esquerda, indicando que a luz esteja desligada. **Figura 58** mostra o *slider* na metade, com a lâmpada funcionando em 50%. Por fim, a **Figura 59** demonstra o *slider* totalmente para a direita, com a luminosidade da lâmpada em 100%.

Figura 57 – *Slider* para esquerda – lâmpada em 0%.



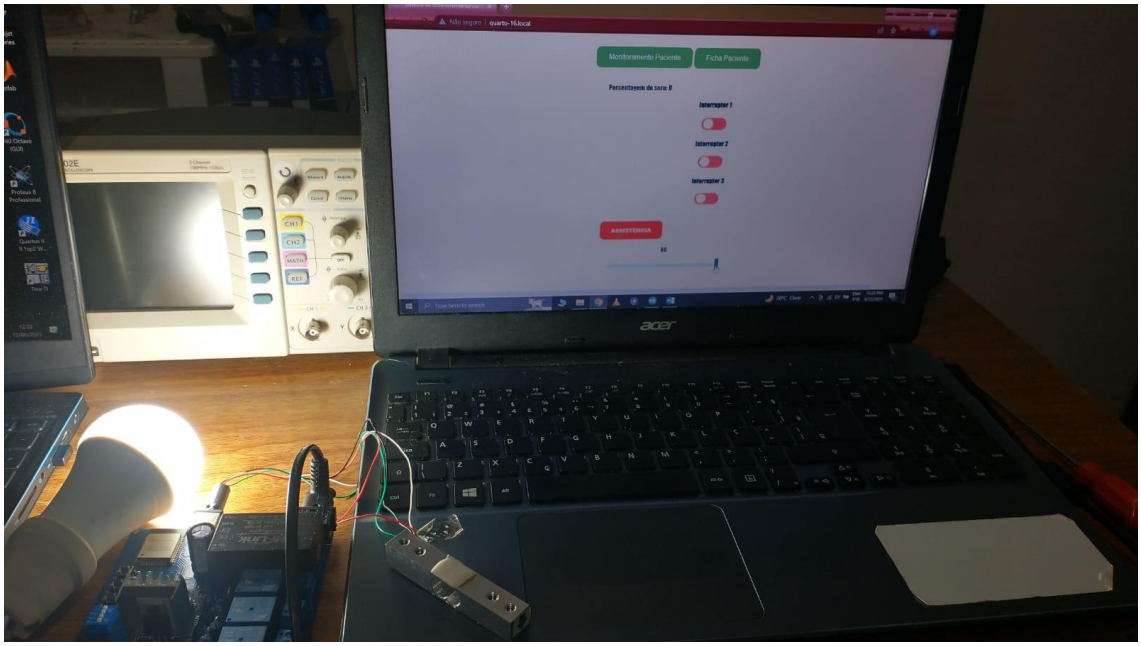
Fonte: Próprio do autor.

Figura 58 – *Slider* na metade – lâmpada em 50%.



Fonte: Próprio do autor.

Figura 59 – *Slider* para direita – lâmpada em 100%.

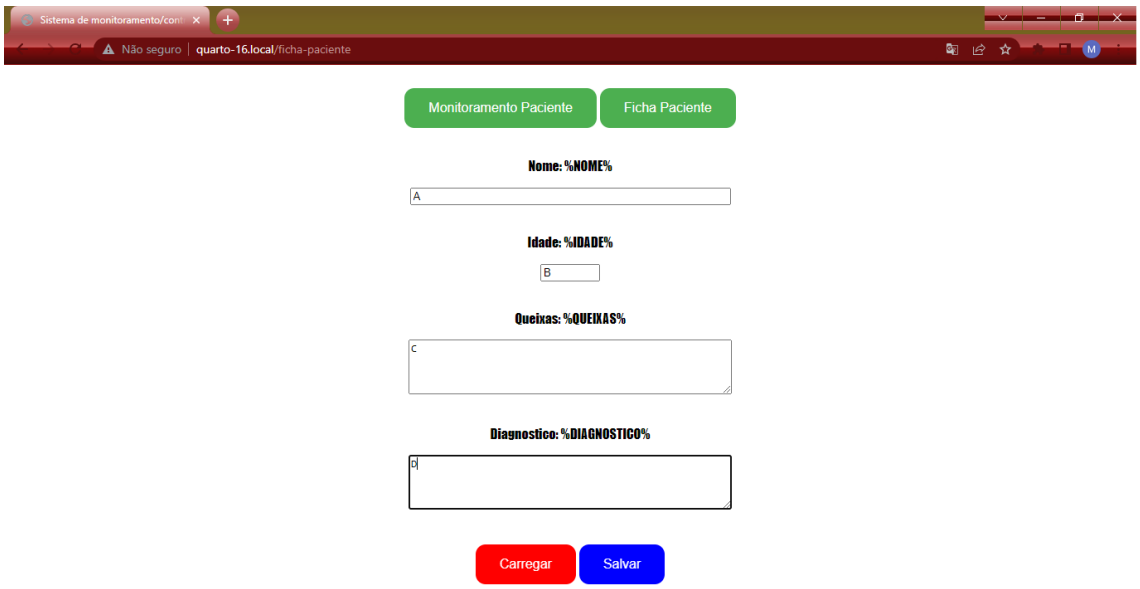


Fonte: Próprio do autor.

Para testar o funcionamento do cartão de memória primeiro foi ligado a placa sem se ter inserido o mesmo, causando um erro no seu reconhecimento que pode ser observado na **Figura 54** e **Figura 55** no teste dos DIP *switches*. Tal erro é mostrado no *display* OLED, provando o seu funcionamento e reconhecimento de erros. Como exemplo (**Figura 60**), a ficha foi completada com as seguintes informações:

- Nome: A;
- Idade: B;
- Queixas: C;
- Diagnóstico: D.

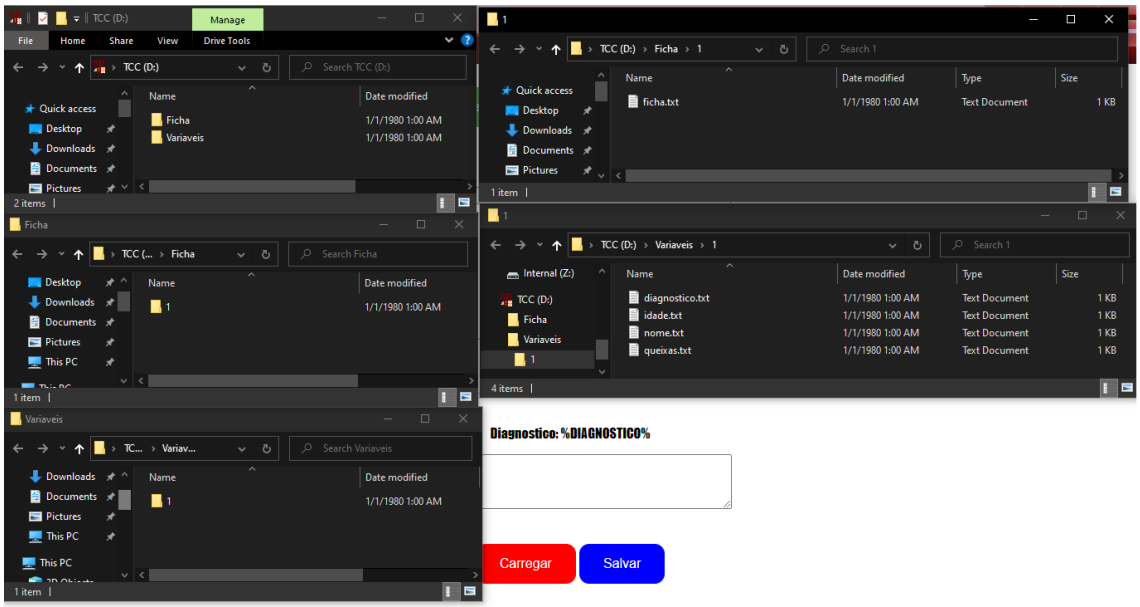
Figura 60 – Primeira ficha sendo salva.



Fonte: Próprio do autor.

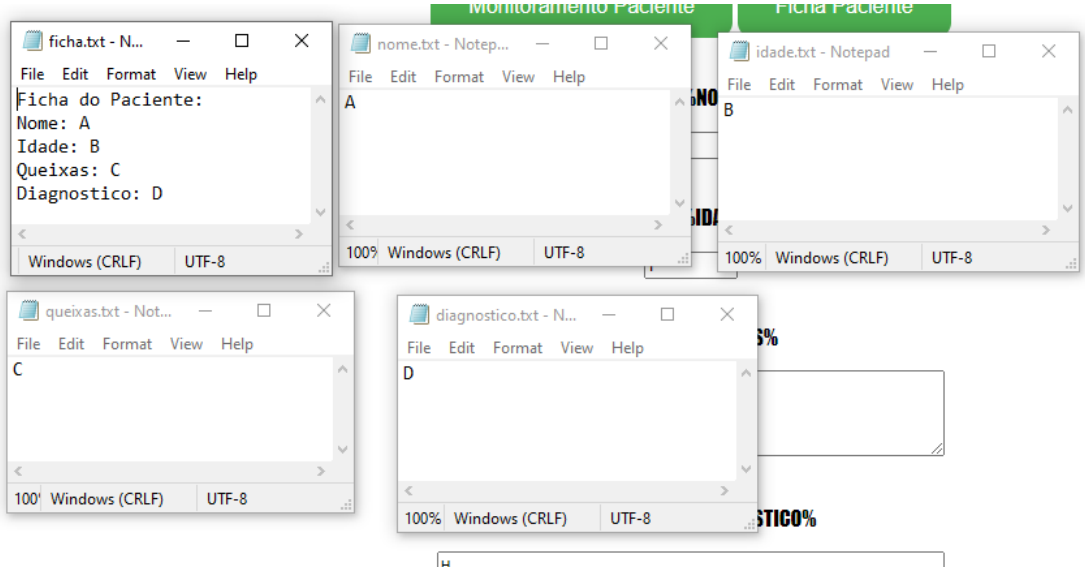
Com a ficha salva, foram criadas duas pastas na raiz do cartão, que estava vazio: uma de nome "Ficha" e outra de nome "Variáveis" (**Figura 61**). Dentro delas foi criada a pasta "1", com as informações fornecidas anteriormente (**Figura 62**).

Figura 61 – Pastas criadas no cartão.



Fonte: Próprio do autor.

Figura 62 – Arquivos criados no cartão.

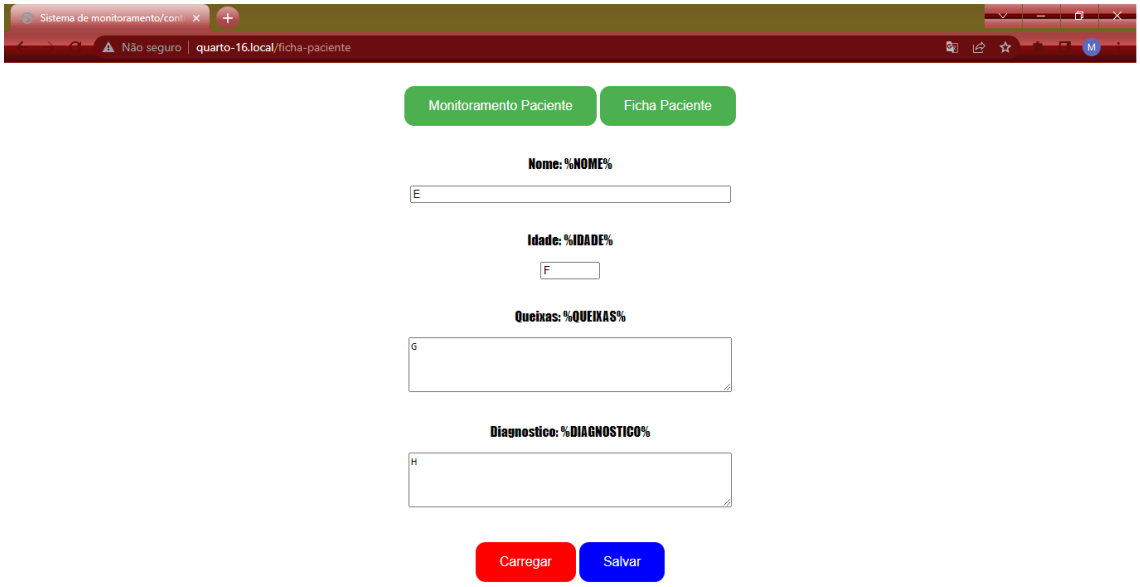


Fonte: Próprio do autor.

Salvando a ficha (**Figura 63**) com as seguintes informações, uma nova pasta "2" é criada em ambos os diretórios (**Figura 64**), mantendo o que foi salvo anteriormente. Os arquivos salvos podem ser vistos na **Figura 65.**

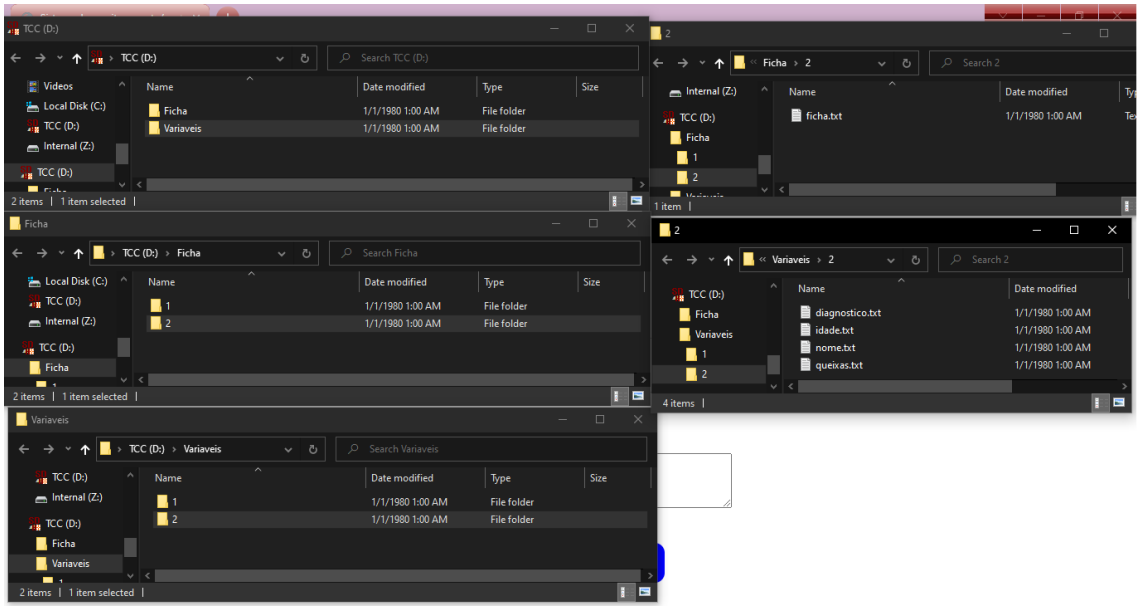
- Nome: E;
- Idade: F;
- Queixas: G;
- Diagnóstico: H.

Figura 63 – Segunda ficha sendo salva.



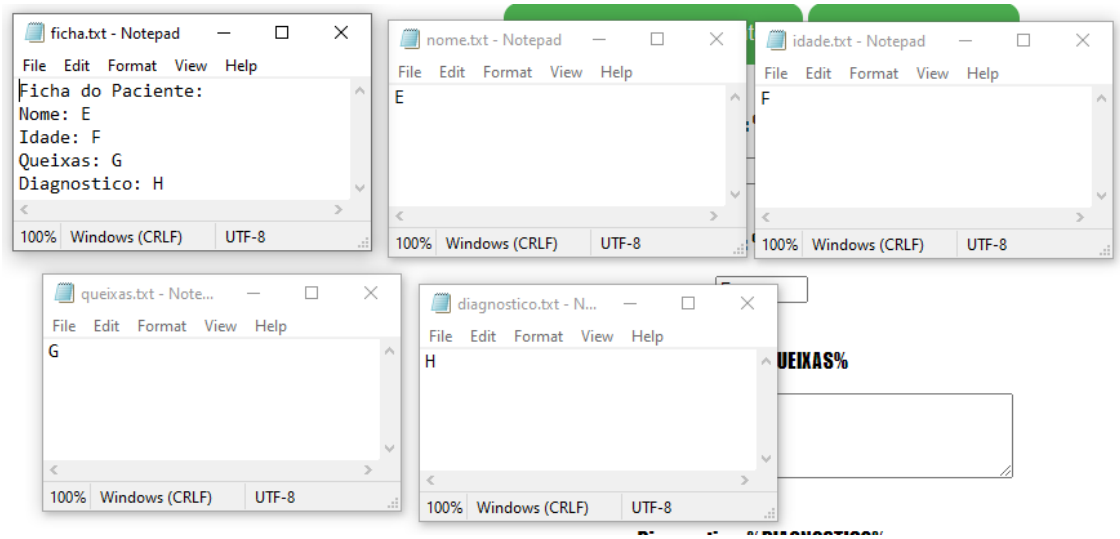
Fonte: Próprio do autor.

Figura 64 – Pastas criadas no cartão.



Fonte: Próprio do autor.

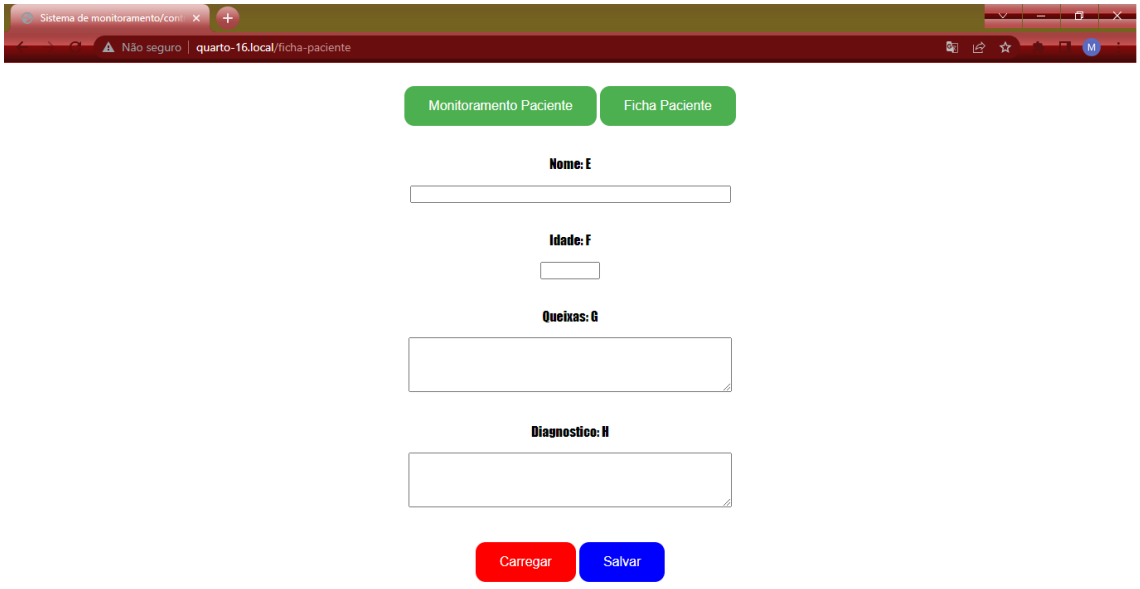
Figura 65 – Arquivos criados no cartão.



Fonte: Próprio do autor.

Ao clicar em "Carregar" na página de criação da ficha do paciente, a última ficha salva será carregada e mostrada na tela, como demonstrado na **Figura 66**, com a ficha "EFGH" carregada.

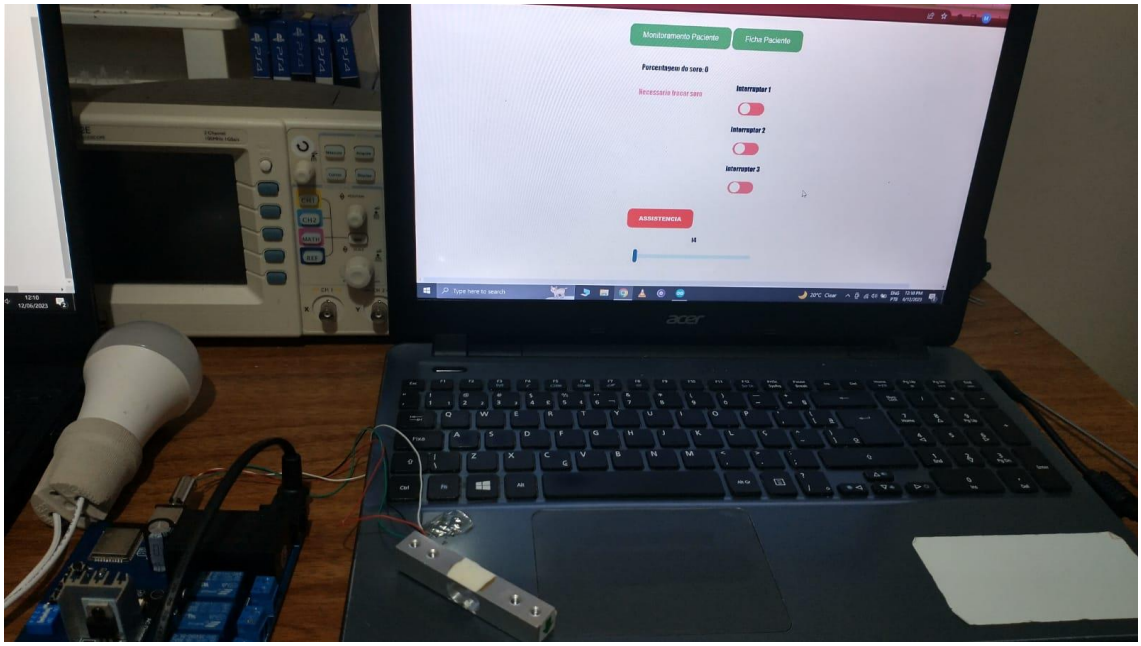
Figura 66 - Segunda ficha carregada.



Fonte: Próprio do autor.

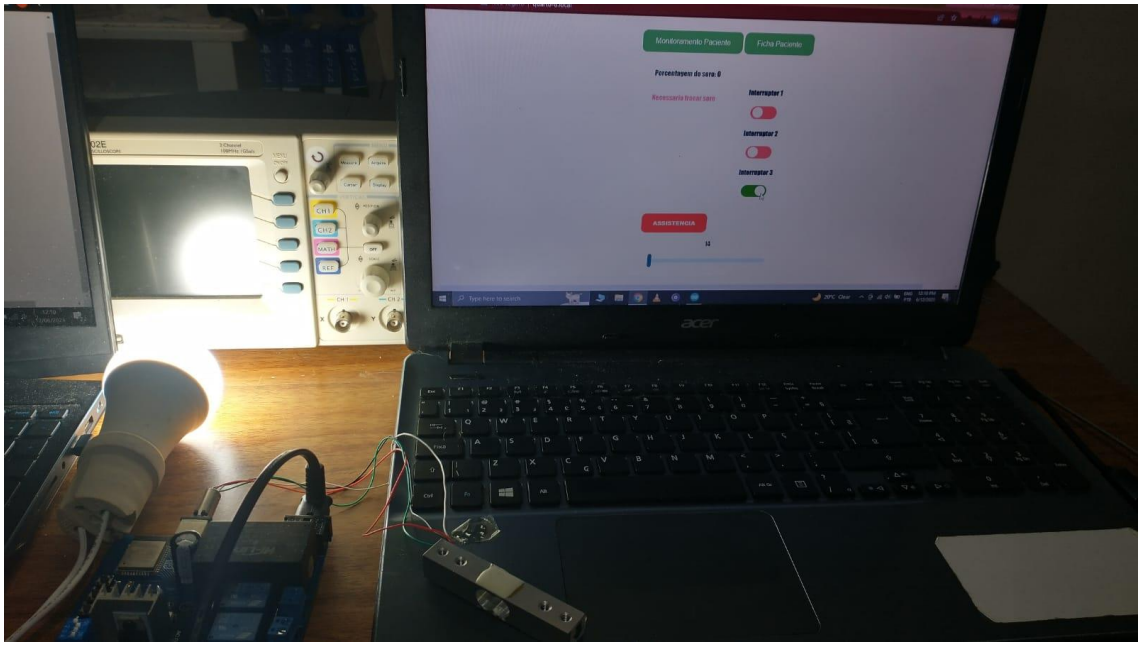
A ativação dos relés se comportou corretamente, com um exemplo de sua ativação na **Figura 67** e **Figura 68**, onde a lâmpada de teste foi conectada ao terceiro borne controlada pela página de monitoramento.

Figura 67 – Lâmpada conectada ao relé desativado.



Fonte: Próprio do autor.

Figura 68 – Lâmpada conectada ao relé ativado.



Fonte: Próprio do autor.

5. Conclusão

Após analisar os resultados obtidos com o que foi originalmente proposto, conclui-se que o trabalho conseguiu alcançar o desejado. O desenvolvimento de uma placa controlada por um ESP-32 dedicada a otimização e controle de um quarto de hospital, que permite um maior controle do ambiente ao paciente que ali está, foi atingido com sucesso.

Neste projeto foi criado uma interface entre o paciente e o funcionário de saúde responsável para poder monitorar o nível de soro e visualizar pedidos de ajuda, com seleção de número de quartos, ativação e desativação de 3 interruptores conectados à rede, controle de intensidade de luz e possibilidade de salvar e carregar informações sobre o paciente em um cartão de memória, tudo isso funcionando sem a necessidade de uma fonte externa, tornando o projeto extremamente fácil de ser integrado no ambiente.

Se apresentando como um trabalho multifacetado, foram necessários conhecimentos de diversas disciplinas ministradas ao decorrer do curso para sua execução, como por exemplo: Eletrônica, Instrumentação Eletrônica e Circuitos Digitais para o desenvolvimento do hardware, e Introdução a Ciência da Computação para o desenvolvimento do software. Muito da teoria necessária, como o entendimento de *Javascript* e HTML, foi aprendida por meios externos, entendimento esse que só foi possível graças a base teórica fornecidas por estas disciplinas citadas anteriormente, no entanto.

5.1. Implementações futuras

Tendo atingido seu objetivo, o projeto apresenta diversos pontos que podem ser melhorados ou mudados de acordo com a necessidade encontrada, como por exemplo:

- Integrar o circuito do módulo *dimmer* na placa, o que evitaria a necessidade de se comprar um módulo específico;
- Dependendo do uso destinado, não utilizar o módulo *dimmer*, devido a sua demanda de processamento ao ESP;
- Criar uma versão "*Lite*" do projeto, utilizando um ESP8266, devido ao seu baixo custo, com apenas a interface de monitoramento para medir o nível de soro e aviso de emergência, pois apresenta poucas I/Os disponíveis.
- Integrar o circuito de programação do ESP-32, aumentando a complexidade da placa, porém a deixando mais versátil, seguindo por completo o esquemático do *devkit* fornecido pela *Espressif*.

6. Referências

ADAMFK. **HX711 module**. 2019. Disponível em: https://oshwlab.com/adamfk/hx711-module. Acesso em: 05 jun. 2023.

Avia Semiconductor. **24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales**. 2023. Disponível em: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf. Acesso em: 05 jun. 2023.

CIRCUITAR. **Zero Cross**: Detecção de zero da rede elétrica. Detecção de zero da rede elétrica. 2023. Disponível em: https://www.circuitar.com.br/nanoshields/modulos/zero-cross/index.html. Acesso em: 20 maio 2023.

Electronics Tutorials. **Wheatstone Bridge**. 2023. Disponível em: https://www.electronics-tutorials.ws/blog/wheatstone-bridge.html. Acesso em: 19 maio 2023.

ESPHOME. **NodeMCU ESP32**. 2023. Disponível em: https://esphome.io/devices/nodemcu_esp32.html. Acesso em: 17 maio 2023.

ESPRESSIF. **ESP32_DevKitc_V4**. 2017. Disponível em: https://dl.espressif.com/dl/schematics/esp32_devkitc_v4-sch.pdf. Acesso em: 23 maio 2023.

FLINTEC. **What Is A Load Cell And How Does It Work?** 2023. Disponível em: https://www.flintec.com/weight-sensors/load-cells/what-is-a-load-cell. Acesso em: 08 jul. 2023.

FORTINET. **What Is DNS (Domain Name System)?** 2023. Disponível em: https://www.fortinet.com/resources/cyberglossary/what-is-dns. Acesso em: 19 maio 2023.

Instituto NCB. **Dimmer com TRIAC (ART294)**. Disponível em: https://newtoncbraga.com.br/index.php/artigos/54-dicas/2025-art294.html. Acesso em: 20 maio 2023.

IONOS. **Multicast DNS: alternative name resolution on a small scale**. 2020. Disponível em: https://www.ionos.com/digitalguide/server/know-how/multicast-dns/. Acesso em: 20 maio 2023.

MDN Web Docs. **HTML basics**. 2022. Disponível em: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics. Acesso em: 07 set. 2022.

MDN Web Docs (a). **What is JavaScript?** 2023. Disponível em: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript. Acesso em: 17 maio 2023.

MDN Web Docs (b). **HTTP request methods**. 2023. Disponível em: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods. Acesso em: 17 maio 2023.

MISCHIANTI, Renzo. **ESP32-wroom-32 high resolution pinout and specs**. 2021. Disponível em: https://mischianti.org/2021/05/26/esp32-wroom-32-high-resolution-pinout-and-specs/. Acesso em: 17 maio 2023.

PISTORIA, Michael Joseph. **Confusão e declínio mental devido à hospitalização**. 2021. Disponível em: https://www.msdmanuals.com/pt-br/casa/assuntos-especiais/cuidados-hospitalares/confusão-e-declínio-mental-devido-à-hospitalização. Acesso em: 09 jul. 2022.

Portal FGV. **Retrospectiva 2021: Brasil tem dois dispositivos digitais por habitante, revela pesquisa da FGV**. 2021. Disponível em: https://portal.fgv.br/noticias/retrospectiva-2021-brasil-tem-dois-dispositivos-digitais-habitante-revela-pesquisa-fgv. Acesso em: 09 jul. 2022.

Random Nerd Tutorials. **ESP32 with Load Cell and HX711 Amplifier (Digital Scale)**. 2023. Disponível em: https://randomnerdtutorials.com/esp32-load-cell-hx711/. Acesso em: 19 maio 2023.

ROBOTDYN. **AC Dimmer Module, 1 Channel, 3.3V/5V logic, AC 50/60hz, 110V~400V**. Disponível em: https://robotdyn.com/ac-light-dimmer-module-1-channel-3-3v-5v-logic-ac-50-60hz-220v-110v.html. Acesso em: 09 jul. 2022.

SILVA, Adriano. **Converter Binário para Decimal: Conversor Bin to Dec online**. 2018. Disponível em: https://calculareconverter.com.br/binario-para-decimal/. Acesso em: 22 maio 2023.

SOUZA, Karina. **Brasil é um dos cinco países com maior número de celulares, mostra ranking**. 2021. Disponível em: https://exame.com/pop/brasil-e-um-dos-cinco-paises-com-maior-numero-de-celulares-mostra-ranking/. Acesso em: 09 jul. 2022.

SYSTEMS, Espressif. **ESP32-WROOM-32 Datasheet**. 2023. Disponível em: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf. Acesso em: 17 maio 2023.

W3SCHOOLS. **HTTP Request Methods**. 2023. Disponível em: https://www.w3schools.com/tags/ref_httpmethods.asp. Acesso em: 17 maio 2023.

Apêndice A – Programação do ESP-32

```
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <Wire.h>
#include <ESPmDNS.h>
#include <SPIFFS.h>
#include "HX711.h"
#include "soc/rtc.h"
#include "FS.h"
#include "SD.h"
#include "SPI.h"

#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

#include <RBDdimmer.h>
#define outputPin 23
#define zerocross  4

#define SD_MISO     2
#define SD_MOSI     15
#define SD_SCLK     14
#define SD_CS       13
SPIClass sdSPI(VSPI);

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

int switch_pin1, switch_pin2, switch_pin3,switch_pin4, bin_number;
int flag_ajuda1;
int flag_ajuda2;
int calibrar;
int calibrar2 = 1;
int primeira_leitura = 0;
int percent;
int total;
int counter, counter_2;
int slider;
int sliderValue2;
int erro_display, erro_display1, erro_display2;
int wifi_check;

String str_number, url;
String sliderValue = "14";
String dir_name1 = "Sem_diretorio";
String dir_name2 = "Sem_diretorio";

const char* url_char;
const char* ssid = "Teste TCC";
const char* password = "12345678";
const char* PARAM_INPUT_1 = "output";
const char* PARAM_INPUT_2 = "state";
const char* PARAM_INPUT_3 = "value";
const char* PARAM_INPUT_4 = "nome";
const char* PARAM_INPUT_5 = "idade";
```

```
const char* PARAM_INPUT_6 = "queixas";
const char* PARAM_INPUT_7 = "diagnostico";

String inputMessage3;
String inputMessage4;
String inputMessage5;
String inputMessage6;
String inputMessage7;

String load_SD;
String load_SD1 = "";
String load_SD2 = "";
String load_SD3 = "";
String load_SD4 = "";

const int DOUT_PIN = 32;
const int SCK_PIN = 27;

dimmerLamp dimmer(outputPin, zerocross);

float cal = (((110601+109860+110161+110937)/4)/294);
HX711 scale;



AsyncWebServer server(80);

int peso = 0;
int i = 0;

//Respostas ao servidor para substituir os marcadores entre "%%" no HTML (Ex: %SORO%,
%INTERRUPTORES%...)
String processor(const String& var){
  if(var == "SORO"){
    return getPeso();
  }

  if(var == "INTERRUPTORES"){
    String buttons = "";
    buttons += "<h4>Interruptor 1</h4><label class=\"button_switch\"><input type=\"checkbox\"
onchange=\"toggle_button_switch(this)\" id=\"17\" " + outputState(17) + "><span
class=\"button_slider\"></span></label>";
    buttons += "<h4>Interruptor 2</h4><label class=\"button_switch\"><input type=\"checkbox\"
onchange=\"toggle_button_switch(this)\" id=\"16\" " + outputState(16) + "><span
class=\"button_slider\"></span></label>";
    buttons += "<h4>Interruptor 3</h4><label class=\"button_switch\"><input type=\"checkbox\"
onchange=\"toggle_button_switch(this)\" id=\"12\" " + outputState(12) + "><span
class=\"button_slider\"></span></label>";
    return buttons;
  }

  if (var == "SLIDERVALUE"){
    return sliderValue;
  }

  if (var == "AJUDA-SORO"){
    Serial.println(flag_ajuda1);
    return text_ajuda();
    }
```

```
  if (var == "NOME"){
    return getNome();
    Serial.print(load_SD1);
  }

  if (var == "IDADE"){
    return getIdade();
    Serial.print(load_SD2);
  }

  if (var == "QUEIXAS"){
    return getQueixas();
    Serial.print(load_SD3);
  }

  if (var == "DIAGNOSTICO"){
    return getDiag();
    Serial.print(load_SD4);
  }
  return String();
}

String outputState(int output){
  if(digitalRead(output)){
    return "checked";
  }
  else {
    return "";
  }
}


String text_ajuda(){
  if (flag_ajuda1 == 1){
    return "<p class=\"blink\"> Necessario trocar soro </p>";
    }
    if (flag_ajuda1 == 0){
    return " ";
    }
}

String text_ajuda_geral(){
  flag_ajuda2 = flag_ajuda2+1;
  if (flag_ajuda2 == 2){
    flag_ajuda2 = 0;
    }

  if (flag_ajuda2 == 1){
    return "<p class=\"blink\"> ASSISTENCIA NECESSARIA </p>";
    }
    if (flag_ajuda2 == 0){
    return " ";
    }
}

String check_ajuda_geral(){
  if (flag_ajuda2 == 1){
    return "<p class=\"blink\"> ASSISTENCIA NECESSARIA </p>";
```

```
    }
    if (flag_ajuda2 == 0){
    return " ";
    }
}


void setup(){

  Serial.begin(9600);

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("Erro ao iniciar o Display"));
  }
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);

  //Inicialização do Cartão SD
  sdSPI.begin(SD_SCLK, SD_MISO, SD_MOSI, SD_CS);
  if(!SD.begin(SD_CS, sdSPI)) {
    Serial.println("Falha ao abrir o Cartão SD");
    erro_display = 1;
  }

  //Entrada dip switch para número do quarto
  pinMode(39, INPUT);
  pinMode(36, INPUT);
  pinMode(35, INPUT);
  pinMode(34, INPUT);

  //Botão para calibrar a balança
  pinMode(33, INPUT_PULLUP);

  //Saida para os relés
  pinMode(12, OUTPUT);
  digitalWrite(12, LOW);
  pinMode(16, OUTPUT);
  digitalWrite(16, LOW);
  pinMode(17, OUTPUT);
  digitalWrite(17, LOW);

  //Função dedicada a converter o número binário geradio pelo DIP switch em decimal para
gerar uma URL única a cada quarto
  selecao_quartos();

  //Inicialização do dimmer
  dimmer.begin(NORMAL_MODE, ON);
  dimmer.getPower();
  dimmer.setPower(14);

  //Inicialização da balança
  scale.begin(DOUT_PIN, SCK_PIN);
  scale.set_scale(cal);
  scale.tare();
```

```
//Inicialização do SPIFFS
if(!SPIFFS.begin(true)){
  Serial.println("");
  Serial.println("Erro ao iniciar SPIFFS");
  display.println("Erro ao iniciar SPIFFS");
  display.display();
  return;
}

//Conexão Wi-Fi
WiFi.begin(ssid, password);
wifi_check = 0;
while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  display.setCursor(0, 0);
  Serial.println("");
  Serial.println("Conectando a rede...");
  display.println("");
  display.println("Conectando a rede...");
  display.display();
  wifi_check = wifi_check + 1;
  if (wifi_check == 10){
    display.clearDisplay();
    display.setCursor(0, 0);
    display.println("");
    display.println("Rede nao encontrada");
    display.display();
    return;

  }
}

display.clearDisplay();
//Endereço IP fornecido ao ESP32
display.setCursor(0, 0);
Serial.println("");
Serial.print("Endereço IP: ");
Serial.print(WiFi.localIP());
display.println("");
display.print("IP: ");
display.print(WiFi.localIP());

//Geração do mDNS (URL)
if (!MDNS.begin(url_char)) {
  Serial.println("");
  Serial.println("Erro ao iniciar mDNS");
  display.println("Erro mDNS");
  display.display();

}
Serial.println("");
Serial.println("mDNS gerado com sucesso");
Serial.println("");
Serial.print("URL gerado: ");

Serial.print(url);
Serial.print(".local/");
display.println("");
display.print("URL: ");
```

```
display.print(url);
display.print(".local/");
display.display();



//Caminhos (rotas, routes) de comunicação entre o servidor e o ESP32
server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send(SPIFFS, "/monitoramento.html", String(), false, processor);
});

server.on("/style.css", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send(SPIFFS, "/style.css", "text/css");
});

server.on("/SORO", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send_P(200, "text/plain", getPeso().c_str());
});

server.on("/AJUDASORO", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send_P(200, "text/plain",text_ajuda().c_str());
});

server.on("/AJUDAGERAL", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send_P(200, "text/plain",text_ajuda_geral().c_str());
});

server.on("/AJUDAGERAL-CHECK", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send_P(200, "text/plain",check_ajuda_geral().c_str());
});

server.on("/ficha-paciente", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send(SPIFFS, "/ficha.html");
});
server.on("/monitoramento", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send(SPIFFS, "/monitoramento.html");
});


server.on("/update", HTTP_GET, [] (AsyncWebServerRequest *request) {
  String inputMessage1;
  String inputMessage2;
  //"url/update?output=<inputMessage1>&state=<inputMessage2>"
  if (request->hasParam(PARAM_INPUT_1) && request->hasParam(PARAM_INPUT_2)) {
    inputMessage1 = request->getParam(PARAM_INPUT_1)->value();
    inputMessage2 = request->getParam(PARAM_INPUT_2)->value();
    digitalWrite(inputMessage1.toInt(), inputMessage2.toInt());
  }
  Serial.println("");
  Serial.print("Rele: ");
  Serial.print(inputMessage1);
  Serial.print(" - Setado em: ");
  Serial.println(inputMessage2);
  request->send(200, "text/plain", "OK");
});

server.on("/slider", HTTP_GET, [] (AsyncWebServerRequest *request) {
  String inputMessage3;
```

```
  //"url/slider?value=<inputMessage3>"
  if (request->hasParam(PARAM_INPUT_3)) {
    inputMessage3 = request->getParam(PARAM_INPUT_3)->value();
    sliderValue = inputMessage3;
    sliderValue2 = sliderValue.toInt();

  }
  Serial.println("");
  Serial.print("Slider: ");
  Serial.print(inputMessage3);
  request->send(200, "text/plain", "OK");
 });

  server.on("/ficha", HTTP_GET, [] (AsyncWebServerRequest *request) {

  //"url/slider?value=<inputMessage3>"----------------------------------------------------------------------------
--
  if (request->hasParam(PARAM_INPUT_4) && request->hasParam(PARAM_INPUT_5) &&
request->hasParam(PARAM_INPUT_6) && request->hasParam(PARAM_INPUT_7)) {
    inputMessage4 = request->getParam(PARAM_INPUT_4)->value();
    inputMessage5 = request->getParam(PARAM_INPUT_5)->value();
    inputMessage6 = request->getParam(PARAM_INPUT_6)->value();
    inputMessage7 = request->getParam(PARAM_INPUT_7)->value();
  }


  Serial.println("");
  Serial.print("Nome: ");
  Serial.print(inputMessage4);
  Serial.println("");
  Serial.print(" Idade: ");
  Serial.print(inputMessage5);
  Serial.println("");
  Serial.print(" Queixas: ");
  Serial.print(inputMessage6);
  Serial.println("");
  Serial.print(" Diagnostico: ");
  Serial.print(inputMessage7);
  request->send(200, "text/plain", "OK");
  salvar_cartao();
 });

 server.on("/LER-FICHA", HTTP_GET, [](AsyncWebServerRequest *request){
  carregar_cartao();
  request->send_P(200, "text/plain", getNome().c_str());
 });

 server.on("/NOME", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send_P(200, "text/plain", getNome().c_str());
 });

 server.on("/IDADE", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send_P(200, "text/plain", getIdade().c_str());
 });

 server.on("/QUEIXAS", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send_P(200, "text/plain", getQueixas().c_str());
 });
```

```
  server.on("/DIAGNOSTICO", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", getDiag().c_str());
  });


  //Iniciar o servidor
  server.begin();

  //Setup da tela OLED (Inicializa, limpa e checa por erros no boot do sistema
  display.display();
  clean_screen();
  error_check();

  //Configura um core para somente realizar o funcionamento do dimmer
  xTaskCreatePinnedToCore(
                dimmer_core,
                "dimer_core",
                10000,
                NULL,
                1,
                NULL,
                1);

}

void loop(){
  //Verifica se "tare" está sendo pressionado (zerar a balança)
  calibrar = digitalRead(33);
  if (calibrar == 0){
    calibrar2 = 0;
  }
}


//Responsável por ler a entrada binaria fornecida pelo DIP switch, convertendo em decimal
para a URL fornecida
void selecao_quartos(){
  switch_pin1 = 1 - digitalRead(39);
  switch_pin2 = 1 - digitalRead(36);
  switch_pin3 = 1 - digitalRead(35);
  switch_pin4 = 1 - digitalRead(34);
  Serial.println("");
  Serial.print("Número Binário: ");
  Serial.print(switch_pin4);
  Serial.print(switch_pin3);
  Serial.print(switch_pin2);
  Serial.print(switch_pin1);
  bin_number = (((switch_pin4) * 8) + ((switch_pin3) * 4) + ((switch_pin2)*2) + (switch_pin1) +
1);
  str_number = String(bin_number);
  url = "quarto-" + str_number;
  url_char = url.c_str();

}

//Funçao geral da balança, adquirindo o peso do soro
String getPeso() {
  scale.power_up();
  peso = scale.get_units(5);
```

```
  if (primeira_leitura==0){
    total = peso;
    primeira_leitura = 1;
  }
  if (calibrar2==0){
    total = peso;
    calibrar2 = 1;
  }

  scale.power_down();
  if (total == 0){
    percent = 0;
  }
  else{
    percent = ((peso * 100) / total);
  }

  if ((peso<=0) || (percent<=0)){
    peso=0;
    percent=0;
    flag_ajuda1 = 1;
  }
  else{
    flag_ajuda1 = 0;
  }
  Serial.println("");
  Serial.print("Peso do soro:");
  Serial.print(peso);
  Serial.println("");
  Serial.print("Total:");
  Serial.print(total);
  return String(percent);
}
//Funções que adquirem os itens da ficha para o servidor
String getNome(){
  return String(load_SD1);
}

String getIdade(){
  load_SD2 = load_SD2;
  return String(load_SD2);
}

String getQueixas(){
  load_SD3 = load_SD3;
  return String(load_SD3);
}

String getDiag(){
  load_SD4 = load_SD4;
  return String(load_SD4);
}

//Função que salva os dados da ficha no cartão, contendo sub funções responsáveis por
verificar se os diretórios já existem ou se já existem arquivos nos mesmos
//Sub funções: listDir_1, listDir_2, listDir_3, creatDir, writeFile_1 e writeFile_2
void salvar_cartao(){
   listDir_1(SD, "/", 0);
```

```
  listDir_2(SD, "/Ficha", 0);
  listDir_3(SD, "/Variaveis", 0);
  clean_screen();
  error_check();

}

//Função para carregar os dados da ficha que estão no cartão SD
//Sub função: listDir_4, readFile
void carregar_cartao(){
 listDir_4(SD, "/Variaveis", 0);
 clean_screen();
 error_check();

}



void listDir_1(fs::FS &fs, const char * dirname, uint8_t levels){
 counter = 0;
 Serial.printf("Listando diretorio: %s\n", dirname);

 File root = fs.open(dirname);
 if(!root){
  Serial.println("Falha ao abrir diretorio");
  erro_display1 = 10;
  return;
 }
 if(!root.isDirectory()){
  Serial.println("Diretorio nao encontrado");
  erro_display1 = 10;
  return;
 }

 File file = root.openNextFile();
 while(file){
  if(file.isDirectory()){
   counter = counter + 1;
   Serial.print("  DIR : ");
   Serial.println(file.name());
   if (counter == 2){
    dir_name1=String(file.name());
    Serial.print(dir_name1);
   }

   if (counter == 3){
    dir_name2=String(file.name());
    Serial.print(dir_name2);
   }


  }
  file = root.openNextFile();
 }

 if (dir_name1 != "Ficha") {
   createDir(SD, "/Ficha");
 }
```

```
  if (dir_name2 != "Variaveis") {
    createDir(SD, "/Variaveis");
  }
}


void listDir_2(fs::FS &fs, const char * dirname, uint8_t levels){
  counter_2 = 0;
  Serial.printf("Listando diretorio: %s\n", dirname);

  File root = fs.open(dirname);
  if(!root){
    Serial.println("Falha ao abrir diretorio");
    erro_display1 = 10;
    return;
  }
  if(!root.isDirectory()){
    Serial.println("Diretorio nao encontrado");
    erro_display1 = 10;
    return;
  }

  File file = root.openNextFile();
  while(file){
    if(file.isDirectory()){
      Serial.print("  DIR : ");
      Serial.println(file.name());
      counter_2 = String(file.name()).toInt();
    }

    file = root.openNextFile();
  }
  Serial.print("  Contador: ");
  Serial.println(counter_2);

     int next_folder = counter_2 + 1;
     String dir_next_folder = "/Ficha/" + String(next_folder);

     String var1 = dir_next_folder + "/ficha.txt";
     const char* char_var1 = var1.c_str();

     const char* dir_next_folder_char = dir_next_folder.c_str();
     createDir(SD, dir_next_folder_char);
     writeFile_2(SD, char_var1, "Ficha do Paciente:");

}

void listDir_3(fs::FS &fs, const char * dirname, uint8_t levels){
  counter_2 = 0;
  Serial.printf("Listando diretorio: %s\n", dirname);

  File root = fs.open(dirname);
  if(!root){
    Serial.println("Falha ao abrir diretorio");
    erro_display1 = 10;
    return;
  }
  if(!root.isDirectory()){
```

```
  Serial.println("Diretorio nao encontrado");
  erro_display1 = 10;
  return;
 }

 File file = root.openNextFile();
 while(file){
  if(file.isDirectory()){
   Serial.print("  DIR : ");
   Serial.println(file.name());
   counter_2 = String(file.name()).toInt();
  }

  file = root.openNextFile();
 }
 Serial.print("  Contador: ");
 Serial.println(counter_2);

   int next_folder = counter_2 + 1;
   String dir_next_folder = "/Variaveis/" + String(next_folder);

   String var1 = dir_next_folder + "/nome.txt";
   String var2 = dir_next_folder + "/idade.txt";
   String var3 = dir_next_folder + "/queixas.txt";
   String var4 = dir_next_folder + "/diagnostico.txt";

   const char* char_var1 = var1.c_str();
   const char* char_var2 = var2.c_str();
   const char* char_var3 = var3.c_str();
   const char* char_var4 = var4.c_str();

   const char* char_inputMessage4 = inputMessage4.c_str();
   const char* char_inputMessage5 = inputMessage5.c_str();
   const char* char_inputMessage6 = inputMessage6.c_str();
   const char* char_inputMessage7 = inputMessage7.c_str();

   const char* dir_next_folder_char = dir_next_folder.c_str();
   createDir(SD, dir_next_folder_char);

   writeFile_1(SD, char_var1, char_inputMessage4);
   writeFile_1(SD, char_var2, char_inputMessage5);
   writeFile_1(SD, char_var3, char_inputMessage6);
   writeFile_1(SD, char_var4, char_inputMessage7);

}

void listDir_4(fs::FS &fs, const char * dirname, uint8_t levels){
 counter_2 = 0;
 Serial.printf("Listando diretorio: %s\n", dirname);

 File root = fs.open(dirname);
 if(!root){
  Serial.println("Falha ao abrir diretorio");
  erro_display2 = 100;
  return;
 }
 if(!root.isDirectory()){
  Serial.println("Diretorio nao encontrado");
  erro_display2 = 100;
```

```
  return;
 }

 File file = root.openNextFile();
 while(file){
  if(file.isDirectory()){
   Serial.print("  DIR : ");
   Serial.println(file.name());
   counter_2 = String(file.name()).toInt();
  }

  file = root.openNextFile();
 }
 Serial.print("  Contador: ");
 Serial.println(counter_2);


   String dir_next_folder = "/Variaveis/" + String(counter_2);

   String var1 = dir_next_folder + "/nome.txt";
   String var2 = dir_next_folder + "/idade.txt";
   String var3 = dir_next_folder + "/queixas.txt";
   String var4 = dir_next_folder + "/diagnostico.txt";

   const char* char_var1 = var1.c_str();
   const char* char_var2 = var2.c_str();
   const char* char_var3 = var3.c_str();
   const char* char_var4 = var4.c_str();
   readFile(SD, char_var1);
   load_SD1 = load_SD;
   Serial.print(load_SD1);
   readFile(SD, char_var2);
   load_SD2 = load_SD;
   Serial.print(load_SD2);
   readFile(SD, char_var3);
   load_SD3 = load_SD;
   Serial.print(load_SD3);
   readFile(SD, char_var4);
   load_SD4 = load_SD;
   Serial.print(load_SD4);




}


void createDir(fs::FS &fs, const char * path){
 Serial.printf("Criando diretório: %s\n", path);
 if(fs.mkdir(path)){
  Serial.println("Diretório criado");
 } else {
  Serial.println("Falha ao crar diretório");
  erro_display1 = 10;
 }
}
```

```
void writeFile_1(fs::FS &fs, const char * path, const char * message){
  Serial.printf("Salvando arquivo: %s\n", path);

  File file = fs.open(path, FILE_WRITE);
  if(!file){
    Serial.println("Falha ao abrir arquivo para gravação");
    erro_display1 = 10;
    return;
  }
  if(file.print(message)){
    Serial.println("Arquivo salvado");
  } else {
    Serial.println("Falha ao salvar");
    erro_display1 = 10;
  }
  file.close();
}

void writeFile_2(fs::FS &fs, const char * path, const char * message){
  Serial.printf("Salvando arquivo: %s\n", path);

  File file = fs.open(path, FILE_WRITE);
  if(!file){
    Serial.println("Falha ao abrir arquivo para gravação");
    erro_display1 = 10;
    return;
  }
  file.print(message);
  file.println("");
  file.print("Nome: ");
  file.print(inputMessage4);
  file.println("");
  file.print("Idade: ");
  file.print(inputMessage5);
  file.println("");
  file.print("Queixas: ");
  file.print(inputMessage6);
  file.println("");
  file.print("Diagnostico: ");
  file.print(inputMessage7);

  file.close();
}


void readFile(fs::FS &fs, const char * path){
  load_SD = "";
  Serial.printf("Lendo arquivo: %s\n", path);

  File file = fs.open(path);
  if(!file){
    Serial.println("Falha ao abrir arquivo");
    erro_display2 = 100;
    return;
  }

  Serial.print("Lido do arquivo: ");
  while(file.available()){
    load_SD += (char)file.read();
```

```
  }
  file.close();


}

//Função responsável por chegar por erros e mostrar no display OLED
void error_check(){
  erro_display = erro_display + erro_display1 + erro_display2;
  switch (erro_display) {
  case 001:
    display.setCursor(0,25);
    display.println("");
    display.print("Erro:");
    display.println("");
    display.print("SD nao reconhecido");
    break;
  case 010:
    display.setCursor(0,25);
    display.println("");
    display.print("Erro:");
    display.println("");
    display.print("Falha ler arquivos");
    break;
  case 100:
    display.setCursor(0,25);
    display.println("");
    display.print("Erro:");
    display.println("");
    display.print("Falha salvar arquivos");
    break;
  case 110:
    display.setCursor(0,25);
    display.println("");
    display.print("Erro:");
    display.println("");
    display.print("Falha ler arquivos");
    display.println("");
    display.print("Falha salvar arquivos");
    break;

}
display.display();
}

//Limpa a tela a partir do ponto (x,y) (0,25), limpando apenas o que está abaixo do IP e URL
void clean_screen(){
  display.setCursor(0,25);
  display.fillRect(0, 25, 128, 50, BLACK);
  display.display();
}

//Funcionamento do dimmer em um core separado
void dimmer_core( void * pvParameters ){

    while(true){
      if (sliderValue2 != slider){
      Serial.print(sliderValue2);
```

```
    dimmer.setPower(sliderValue2);
    slider = sliderValue2;
    }
    else{
    dimmer.getPower();
    }

  }
}
```

Apêndice B – Programação da página de monitoramento

```
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" href="data:,">
  <link rel="stylesheet" type="text/css" href="style.css">
  <title>Sistema de monitoramento/controle quarto de hospital</title>
<style>
</style>
</head>
<body>

 <div class="centralizar">
        <p><a href="/"><button class="page_button">Monitoramento
Paciente</button></a></p>
        <p><a href="/ficha-paciente"><button class="page_button">Ficha
Paciente</button></a></p>
 </div>

 <div class="left">
        <p>Porcentagem do soro: <span id="SORO"> %SORO%</span></p>
 </div>

 <div class="left_ajuda" id="AJUDA-SORO">
   %AJUDA-SORO%
 </div>

 <div class="left_ajuda_geral" id="AJUDA-GERAL">
   %AJUDA-GERAL%
 </div>

 <div class="right">
        %INTERRUPTORES%
 </div>

 <div class="left">
   <button class="help_button" onclick="generate_ajuda(this)">ASSISTENCIA</button>
 </div>

 <div>
        <p><span id="light_slider_value">%SLIDERVALUE%</span></p>
        <p><input type="range" oninput="updateSliderPWM(this)" id="pwmSlider" min="14"
max="80" value="14" class="light_slider"></p>
 </div>

</body>
<script>

  setInterval(function ( ) {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        document.getElementById("SORO").innerHTML = this.responseText;
      }
    };
```

```
  xhttp.open("GET", "/SORO", true);
  xhttp.send();
 },10000) ;

 function toggle_button_switch(element) {
  var xhr = new XMLHttpRequest();
   if(element.checked){ xhr.open("GET", "/update?output="+element.id+"&state=1", true); }
   else { xhr.open("GET", "/update?output="+element.id+"&state=0", true); }
  xhr.send();
}
 function updateSliderPWM(element) {
         var slider_Value = document.getElementById("pwmSlider").value;
         document.getElementById("light_slider_value").innerHTML = slider_Value;
         console.log(slider_Value);
         var xhr = new XMLHttpRequest();
         xhr.open("GET", "/slider?value="+slider_Value, true);
         xhr.send();
}
setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
   if (this.readyState == 4 && this.status == 200) {
    document.getElementById("AJUDA-SORO").innerHTML = this.responseText;
   }
  };
  xhttp.open("GET", "/AJUDASORO", true);
  xhttp.send();
 },10000) ;

 setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
   if (this.readyState == 4 && this.status == 200) {
    document.getElementById("AJUDA-GERAL").innerHTML = this.responseText;
   }
  };
  xhttp.open("GET", "/AJUDAGERAL-CHECK", true);
  xhttp.send();
 },10000) ;




 function generate_ajuda(element){
         var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
   if (this.readyState == 4 && this.status == 200) {
    document.getElementById("AJUDA-GERAL").innerHTML = this.responseText;
   }
  };
          xhttp.open("GET", "/AJUDAGERAL", true);
  xhttp.send();
 }

</script>
</html>
```

Apêndice C – Programação da página de ficha

```
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" href="data:,">
  <link rel="stylesheet" type="text/css" href="style.css">
  <title>Sistema de monitoramento/controle quarto de hospital</title>
<style>
</style>
</head>
<body>

 <div class="centralizar">
          <p><a href="/"><button class="page_button">Monitoramento
Paciente</button></a></p>
          <p><a href="/ficha-paciente"><button class="page_button" onclick="idle(this)">Ficha
Paciente</button></a></p>
 </div>

  <div class="left_ficha">
  <p>Nome: <span id="text_nome_1"> %NOME%</span></p>
  <p><input <input type="text" id="input_nome" name="input_nome" size="50"><br><br></p>
  <p> Idade: <span id="text_idade_1">%IDADE%</span></p>
  <p><input <input type="text" id="input_idade" name="input_idade" size="5"><br><br></p>
  <p>Queixas: <span id="text_queixas_1">%QUEIXAS%</span></p>
  <p><textarea id="input_queixas" name="input_queixas" rows="4"
cols="50"></textarea><br><br></p>
  <p>Diagnostico: <span id="text_diagnostico_1">%DIAGNOSTICO%</span></p>
  <p><textarea id="input_diagnostico" name="input_diagnostico" rows="4"
cols="50"></textarea><br><br></p>
  </div>

  <div class="centralizar">
  <button class="load_button" onclick="read_ficha(this)">Carregar</button>
  <button class="save_button" onclick="generate_ficha(this)">Salvar</button>
  </div>

</body>
<script>

function idle(elemet){
   var xhttp = new XMLHttpRequest();
   xhttp.open("GET", "/NOME", true);
   xhttp.send();
   var xhttp = new XMLHttpRequest();
   xhttp.open("GET", "/IDADE", true);
   xhttp.send();
}


function generate_ficha(element) {
   var nome = document.getElementById("input_nome").value;

   console.log(nome);
   var idade = document.getElementById("input_idade").value;
```

```
  console.log(idade);
  var queixas = document.getElementById("input_queixas").value;

  console.log(queixas);
  var diagnostico = document.getElementById("input_diagnostico").value;

  console.log(diagnostico);
  var xhr = new XMLHttpRequest();
  xhr.open("GET",
"/ficha?nome="+nome+"&idade="+idade+"&queixas="+queixas+"&diagnostico="+diagnostico,
true);
  xhr.send();
}

function read_ficha(element) {
   var xhttp = new XMLHttpRequest();
   xhttp.open("GET", "/LER-FICHA", true);
   xhttp.send();
   var xhttp = new XMLHttpRequest();
   xhttp.onreadystatechange = function() {
     if (this.readyState == 4 && this.status == 200) {
       document.getElementById("text_nome_1").innerHTML = this.responseText;
     }
   };
          xhttp.open("GET", "/NOME", true);
   xhttp.send();

          var xhttp = new XMLHttpRequest();
   xhttp.onreadystatechange = function() {
     if (this.readyState == 4 && this.status == 200) {
       document.getElementById("text_idade_1").innerHTML = this.responseText;
     }
   };
          xhttp.open("GET", "/IDADE", true);
   xhttp.send();

          var xhttp = new XMLHttpRequest();
   xhttp.onreadystatechange = function() {
     if (this.readyState == 4 && this.status == 200) {
       document.getElementById("text_queixas_1").innerHTML = this.responseText;
     }
   };
          xhttp.open("GET", "/QUEIXAS", true);
   xhttp.send();

          var xhttp = new XMLHttpRequest();
   xhttp.onreadystatechange = function() {
     if (this.readyState == 4 && this.status == 200) {
       document.getElementById("text_diagnostico_1").innerHTML = this.responseText;
     }
   };
          xhttp.open("GET", "/DIAGNOSTICO", true);
   xhttp.send();
}

</script>
</html>
```

Apêndice D – Programação do arquivo CSS

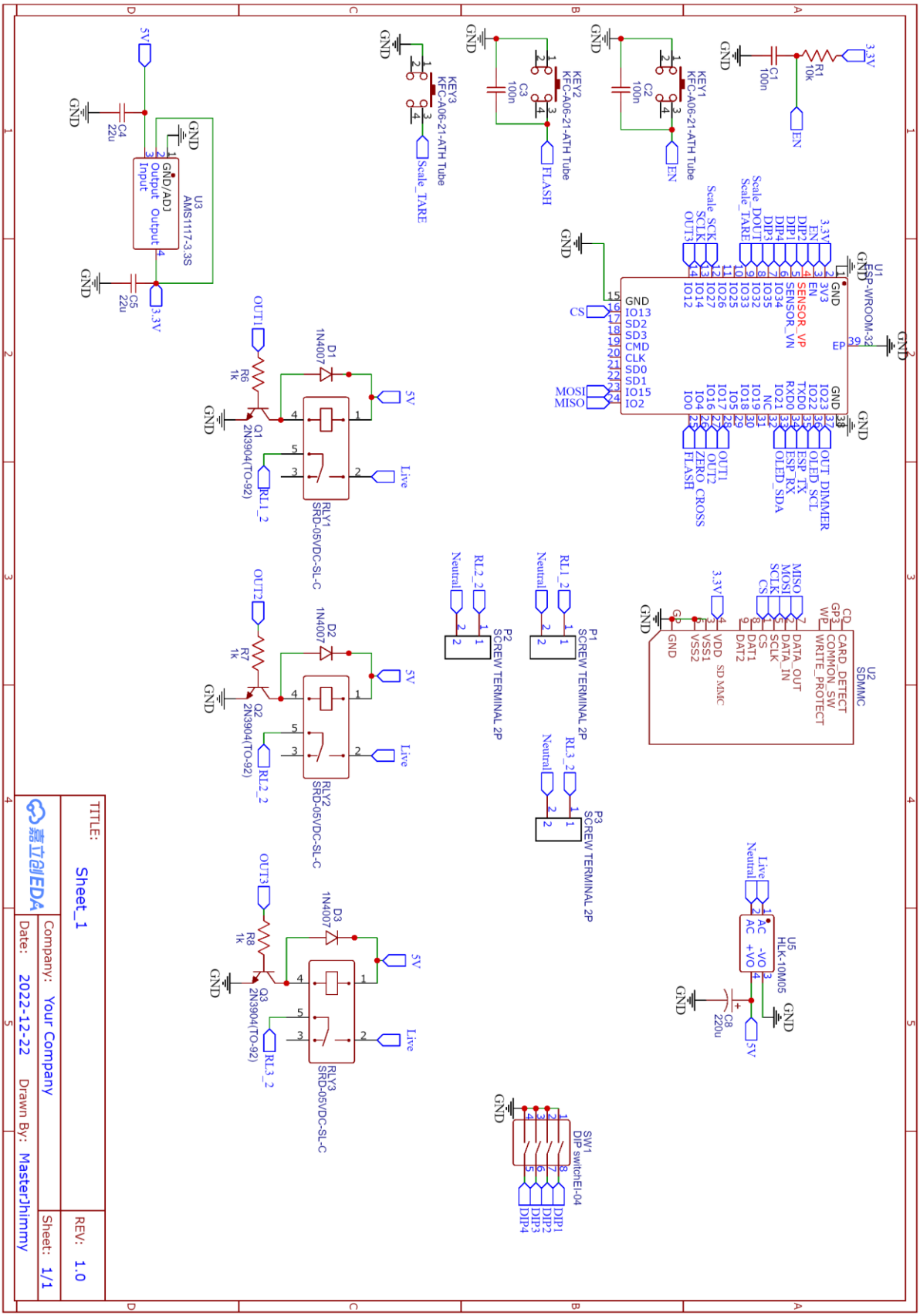
```
html {
  font-family: Impact, fantasy;
  display: inline-block;
  margin: 0px auto;
  text-align: center;
}
h1{
  color: dimgrey;
  padding: 2vh;
}

.centralizar {
  display: flex;
  justify-content: center;
}

.left{
  position: relative;
  right: 90px;
}

.right{
  position: relative;
  left: 100px;
  bottom: 35px;
}

.left_ficha{
        justify-content: flex-start;
}

.left_ajuda{
  position: relative;
  left: -95px;
  top: -20px;
}

.left_ajuda_geral{
  position: relative;
  left: -95px;
  top: 200px;
}

.page_button {
  background-color: #4CAF50;
  border: none;
  border-radius: 12px;
  color: white;
  padding: 15px 29px;
  text-align: center;
  text-decoration: none;
  font-family: Arial;
  font-size: 16px;
  margin: 4px 2px;
  cursor: pointer;
```

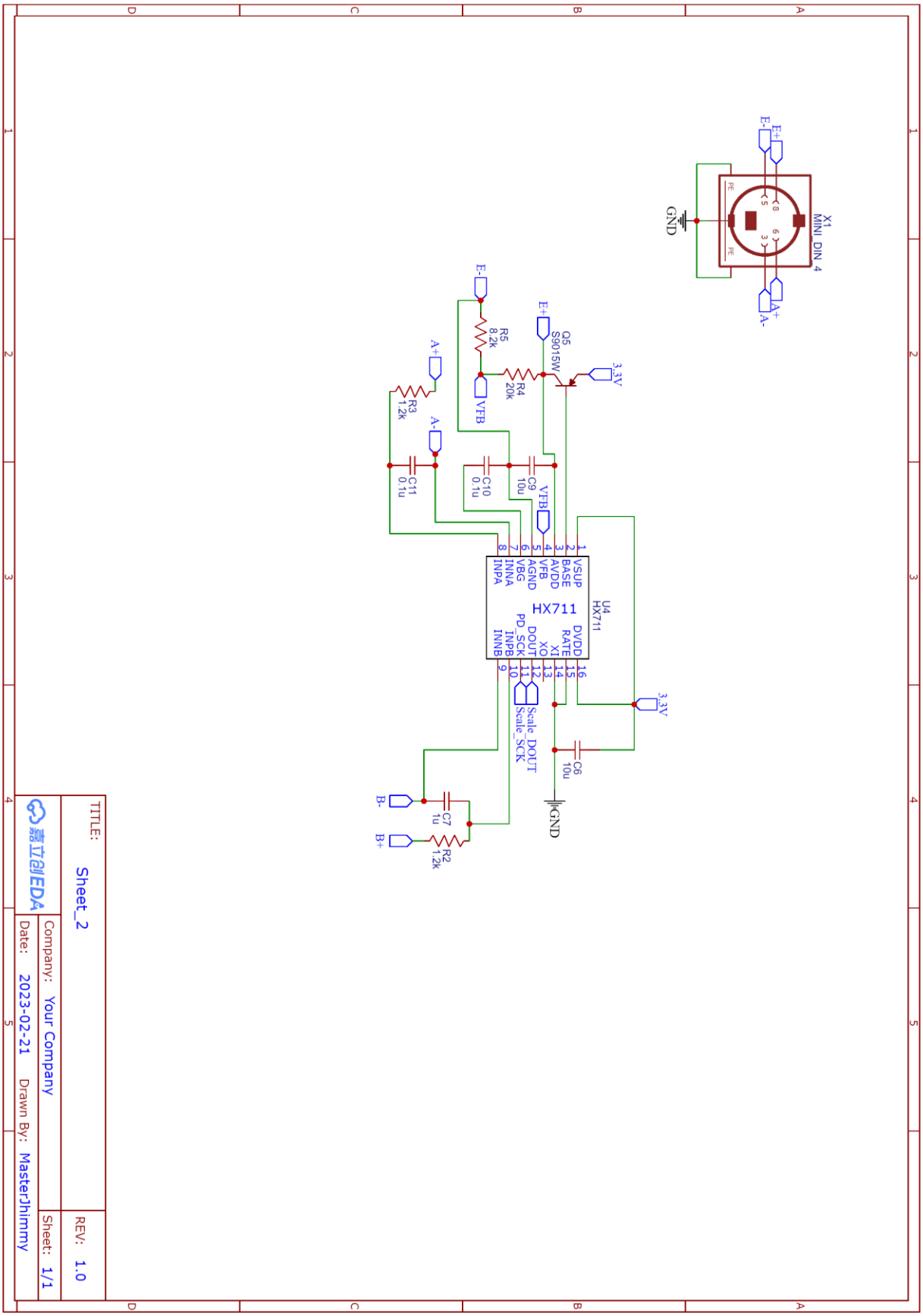
```
}

.help_button {
  background-color: red;
  border: none;
  border-radius: 12px;
  color: white;
  padding: 15px 29px;
  text-align: center;
  text-decoration: none;
  font-family: Arial Black;
  font-size: 14px;
  font-weight: bold;
  margin: 4px 2px;
  cursor: pointer;

}

.save_button {
  background-color: blue;
  border: none;
  border-radius: 12px;
  color: white;
  padding: 15px 29px;
  text-align: center;
  text-decoration: none;
  font-family: Arial;
  font-size: 16px;
  margin: 4px 2px;
  cursor: pointer;

}

.load_button {
  background-color: red;
  border: none;
  border-radius: 12px;
  color: white;
  padding: 15px 29px;
  text-align: center;
  text-decoration: none;
  font-family: Arial;
  font-size: 16px;
  margin: 4px 2px;
  cursor: pointer;

}

.button_switch {
   position: relative;
   display: inline-block;
   width: 64px;
   height: 30px
}

.button_switch input {
   display: none
}
```

```
.button_slider {
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background-color: #ff3333;
  border-radius: 34px;
}

.button_slider:before {
  position: absolute;
  content: "";
  height: 25px;
  width: 25px;
  left: 5px;
  bottom: 3px;
  background-color: #fff;
  -webkit-transition: .4s;
  transition: .4s;
  border-radius: 34px
}


input:checked+.button_slider {
  background-color: green
}

input:checked+.button_slider:before {
  -webkit-transform: translateX(30px);
  -ms-transform: translateX(30px);
  transform: translateX(30px)
}

.light_slider {
        -webkit-appearance: none;
        margin: 14px;
        width: 300px;
        height: 10px;
        background: LightBlue;
        outline: none;
        -webkit-transition: .2s;
        transition: opacity .2s;
        border-radius: 10px;
}

.light_slider::-webkit-slider-thumb {
        -webkit-appearance: none;
        appearance: none;
        width: 10px;
        height: 35px;
        background: #003249;
        cursor: pointer;
        border-radius: 10px;
}
.light_slider::-moz-range-thumb {
        width: 10px;
```

```
        height: 35px;
        background: #003249;
        cursor: pointer;
        border-radius: 10px;
}

.blink {
        position: relative;
        bottom: -40px;
   animation: blinker 0.5s linear infinite;
   color: red;
   }
   @keyframes blinker {
      50% {
         opacity: 0;
         }
      }
```

Apêndice E – Circuito geral




X1
MINI_DIN_4
GND
E+
E-
A+
A-
Q5
S9015W
R5
8.2k
R4
20k
R3
1.2k
R2
1.2k
C11
0.1u
C10
0.1u
C9
10u
C7
1u
C6
10u
3.3V
VFB
B+
B-
U4
HX711
VSUP
BASE
AVDD
VFB
AGND
VBG
INNA
INPA
DVDD
RATE
XI
XO
DOUT
PD_SCK
INPB
INNB
Scale_DOUT
Scale_SCK
TITLE:
Sheet_2
Company: Your Company
Date: 2023-02-21
Drawn By: MasterJhimmy
Sheet: 1/1
REV: 1.0