Universidade de Brasília – UnB
Campus Gama – FGA
Engenharia Eletrônica

Sistema IOT para controlar remotamente Ar Condicionados através de um Servidor WEB

Rodrigo Sousa Santos

Orientador: Dr. Prof. Dr. Sandro Augusto Pavlik Haddad

UnB – Universidade de Brasília

FGA – Faculdade Gama

Engenharia Eletrônica

"Sistema IOT para controlar remotamente ar condicionados através de um servidor WEB"

Rodrigo Sousa Santos

Orientador: Prof. Dr. Sandro Augusto Pavlik Haddad

Trabalho de Conclusão de Curso
Engenharia Eletrônica

Brasília/DF, Maio 2021

UnB – Universidade de Brasília

FGA – Faculdade Gama

Engenharia Eletrônica

"Sistema IOT para controlar remotamente ar condicionados através de um servidor WEB"

Rodrigo Sousa Santos

Trabalho de Conclusão de Curso submetido à Faculdade UnB Gama da Universidade de Brasília, como parte dos requisitos necessários para a obtenção do grau de bacharel em engenharia eletrônica

Aprovada por:

Prof. Dr. Sandro Augusto Pavlik Haddad

(Orientador)

Prof. Dr. Gilmar Silva Beserra

(Examinador interno)

Prof. Dr. Wellington Avelino do Amaral

(Examinador externo)

Brasília/DF, Maio 2021

Ficha Catalográfica

Santos, Rodrigo Sousa
Sistema IOT para controlar remotamente ar condicionados através de um servidor WEB, [Distrito Federal], 2021.
64p., 210 × 297 mm (FGA/UnB Gama, Bacharelado em Engenharia Eletrônica, 2018).
Trabalho de Conclusão de Curso, Faculdade UnB Gama, Engenharia Eletrônica

1. IOT	2. Internet das Coisas
3. Controle remoto	4. Servidor Web
I. FGA UnB/UnB.	II. Título (série)

Referência

Santos, Rodrigo Sousa (2021). Sistema IOT para controlar remotamente ar condicionados através de um servidor WEB. Trabalho de Conclusão de Curso, Engenharia Eletrônica, Faculdade UnB Gama, Universidade de Brasília, Brasília, DF, 64p.

Cessão de Direitos

Autor: Rodrigo Sousa Santos

Título: Sistema IOT para controlar remotamente ar condicionados através de um servidor WEB

Grau: Bacharel em Engenharia Eletrônica

Ano: 2021

---

rodrigo.sousa2711@gmail.com

Brasília, DF – Brasil

Resumo

Palavras-chave: Internet das coisas, IOT, controle remoto, ar condicionado, raspberry pi 3, esp8266, infravermelho, servidor *web*, cliente *web*, protocolos de comunicação.

Com o avanço da tecnologia, a internet das coisas está cada vez mais presente em nossas vidas. Visando facilitar e otimizar processos através da internet, o projeto consiste em disponibilizar um acesso na rede para que o professor consiga controlar o ar condicionado da sala no qual ele está dando aula, reduzindo o tempo perdido que ele leva até a secretaria para buscar e posteriormente deixar o controle remoto, além de preservar o dispositivo contra possíveis quedas e danos. Utilizando a Raspberry Pi 3 como computador central para subir o servidor na rede, o papel dos protocolos de comunicação é fundamental para que os professores tenham acesso à página com os botões de controle, assim como a ESP8266 como cliente consiga se comunicar e receber as *flags* do servidor para poder transmitir o código infravermelho correto para o ar condicionado selecionado realizar a função escolhida pelo usuário. O sistema geral foi separado em cinco subsistemas para inicialmente ser validado individualmente, para que posteriormente seja integrado por completo.

Abstract

Keywords: Internet of things, IOT, remote control, air conditioning, raspberry pi 3, esp8266, infrared, webserver, webclient, communication protocols.

With the advance of technology, the Internet of things is increasingly present in our lives. Aiming to facilitate and optimize processes through the Internet, the project consists in providing a network access so that the teacher can control the air conditioning of the room in which he is teaching, reducing the time he takes to the office to search and then leave the remote control, and preserve the device against possible falls and damage. Using Raspberry Pi 3 as the central computer to upload the server to the network, the role of communication protocols is essential for teachers to have access to the page with the control buttons, as well as the ESP8266 as a client can communicate and receive the flags from the server to be able to transmit the correct infrared code for the selected air conditioning to perform the function chosen by the user. The general system has been separated into five subsystems to be initially validated individually, so that it can later be fully integrated.

Sumário

Lista de Tabelas

Lista de Figuras

Nomenclaturas e Abreviações

IOT	–	*Internet das Coisas*
PIB	–	*Produto Interno Bruto*
RF	–	*Rádio Frequência*
IR	–	*Infravermelho*
LED	–	*Diodo Emissor de Luz*
WEB	–	*World Wide Web*
GPIO	–	*General Purpose Input/Output*
RAM	–	*Random Access Memory*
SDHC	–	*SD High Capacity*
USB	–	*Universal Serial Bus*
RISC	–	*Reduced Instruction Set Computer*
SPI	–	*Serial Peripheral Interface*
I2C	–	*Inter-Integrated Circuit*
I2S	–	*Inter-IC Sound*
UART	–	*Universal Asynchrounous Receiver/Transmiter*
SDK	–	*Kit de Desenvolvimento de Software*
OSI	–	*Interconexão de Sistemas Abertos*
ISO	–	*Organização Internacional de Normalização*
TCP/IP	–	*Transmisson Control Protocol/Internet Protocol*
UDP	–	*User Datagram Protocol*
HTTP	–	*Hypertext Transfer Protocol*
URL	–	*Uniform Resource Locator*
URG	–	*Urgent Pointer*
IPv4	–	*Internet Protocol versão 4*
API	–	*Interface de Programão de Aplicativos*
HTML	–	*Linguagem de Marcação de HiperTexto*
CSS	–	*Folhas de Estilo em Cascata SSH*
–	*Secure Socket Shell*	
GND	–	*Ground*
PCB	–	*Placa de Circuito Impresso*

Lista de Símbolos

m	–	mili (10^{-3})
μ	–	micro (10^{-6})
n	–	nano (10^{-9})
ρ	–	pico (10^{-12})
k	–	kilo (10^{3})
M	–	mega (10^{6})
G	–	mega (10^{9})
V	–	Tensão
Hz	–	Hertz
A	–	Ampére
$\%$	–	Porcentagem
λ	–	Comprimento de onda
$-$	–	Negativo
$+$	–	Positivo
$\ll$	–	Menor que
$\approx$	–	Aproximado
$=$	–	Igualdade
s	–	Segundo
bps	–	Bits por Segundo

1 Introdução

1.1 Contextualização

A Internet das Coisas, *Internet of Things* (IoT), está cada dia mais presente na vida das pessoas ao redor de todo o mundo. Com diversas definições sobre o que é IOT, pode-se dizer de modo geral que são sensores capazes de coletar e fazer o processamento de informações interconectados à objetos físicos através da internet criando um sistema computacional que tem como objeto facilitar a vida das pessoas, incrementando soluções funcionais nos processos do dia a dia [17][18].

De acordo com a Gardner, empresa de consultoria americana, até 2021 existe a estimativa de ter pelo menos 25 bilhões de coisas conectadas à internet em todo o mundo, e a empresa Mckinsey possui como estimativa que a IOT impactará na economia mundial obtendo um alcance de 4% a 11% do Produto Interno Bruto (PIB) [19].

Para o sucesso no avanço da IOT, é necessário que os custos de implementação das soluções tenham um custo compatível com os dispositivos a serem conectados. Por este motivo, é necessário sempre estabelecer uma relação custo benefício para a aplicação, principalmente na utilização de microcontroladores.

Em relação ao problema, para um local que possui uma área extensa e uma grande quantidade de ar condicionados, como por exemplo em uma universidade, disponibilizar controles para cada sala pode se tornar dificultoso quando se fala em organização, justamente pelos riscos onde o dispositivo pode acabar se perdendo ou até mesmo quebrando. Quando o controle do ar condicionado não é disponibilizado na sala e fica concentrado em um determinado local, o professor responsável teria que fazer um deslocamento para buscá-lo, perdendo tempo em que poderia estar aplicando a matéria.

Para facilitar este processo, utilizar um sistema IOT se torna bastante viável, principalmente quando toda a extensão da universidade possui uma rede local, contribuindo para utilização de microcontroladores com módulo *Wi-Fi* integrado, caso da ESP8266 12E, que possui um excelente custo benefício, além da utilização do sistema embarcado

Raspberry Pi 3, que pode ser utilizada como um computador central que que comunica a escolha do usuário com os dispositivos que controlam os ar condicionados.

1.2 Objetivos do Trabalho

Este trabalho possui como objetivo desenvolver um sistema IOT em que um responsável pela sala de aula possa ter acesso ao controle do ar condicionado para ligar, desligar e/ou ajustar a temperatura remotamente através da internet.

Os objetivos específicos são:

1. Criar uma interface *web* (Raspberry Pi 3) para acesso aos botões do controle do ar condicionado pelo usuário;
2. Realizar a comunicação entre a interface *web* e o dispositivo (NodeMCU Esp8266) que enviará os comandos infravermelhos para controle do ar condicionado.
3. Utilizar um sensor de temperatura (DHT11) e umidade para identificar que o ar condicionado está ligado ou desligado, apresentando esta informação na interface *web*.

1.3 Estrutura do Trabalho

A estrutura do trabalho é dividida em cinco capítulos. No capítulo 1 é apresentada a introdução, abordando a contextualização do problema e os objetivos propostos para realização do trabalho.

No capítulo 2 é apresentada a fundamentação teórica do trabalho em duas partes. Na primeira parte é abordado como funciona o controle remoto e suas formas de comunicação entre transmissor e receptor. Posteriormente é enfatizado sobre o infravermelho e como é sua utilização para recepção e transmissão de dados. Na segunda parte é apresentado o desenvolvimento *web* do sistema, abordando a Raspberry Pi 3 que será utilizada para hospedar o servidor, e o ES8266 que será um cliente *web* que fará conexão e transferência de dados com o servidor. Também são discutidos os protocolos de comunicação a serem utilizados para que a conexão ocorra com sucesso, e o *framework* utilizado para se comunicar com o usuário.

No capítulo 3 é abordada toda a metodologia utilizada para realização do projeto, descrevendo como o processo está sendo desenvolvido e as técnicas utilizadas para chegar

ao melhor resultado dos objetivos propostos.

No capítulo 4 são apresentados os resultados obtidos através de testes realizados por software, na comunicação, quanto testes realizados na prática, na recepção e transmissão do sinal infravermelho.

No capítulo 5 são feitas as conclusões em relação aos objetivos do projeto, e as propostas para trabalho futuro.

2 Fundamentação Teórica

2.1 Controle Remoto

O controle remoto é um dispositivo que teve suas primeiras utilizações na Primeira Guerra Mundial pela marinha alemã, com o intuito de direcionar navios a colidir com os barcos dos aliados, e também foi utilizado na Segunda Guerra Mundial para detonação de bombas. Comumente os controles remotos são desenvolvidos com a tecnologia de Rádio frequência (RF), ou infravermelho (IR) [20]. Nos dias atuais, os controles que utilizam RF são bastante utilizados em portões automáticos e alarmes, por exemplo, e controles infravermelhos são utilizados em aparelhos eletroeletrônicos, como TVs e ar condicionados. Pelo fato do dispositivo a ser controlado ser um ar condicionado no trabalho, então o foco será em controle que utiliza radiação infravermelha.

O processo de funcionamento de um controle remoto IR é dado pela utilização da luz para transportar sinais entre o emissor, controle remoto, e o receptor, dispositivo a ser controlado [20].

2.1.1 Radiação Infravermelha

O infravermelho é uma radiação eletromagnética, geralmente associada ao calor. A Figura 2.1 mostra o intervalo de todas as ondas eletromagnéticas, que são classificadas de acordo com seu comprimento de onda (λ). Também é possível notar o intervalo em que o espectro de luz é visível, indo do vermelho, onde $\lambda \approx 400$ ηm e o violeta, em que $\lambda \approx 700$ ηm [21].

Na Figura 2.1 também é possível identificar onde se encontra a faixa de comprimento e frequência da onda infravermelha. Fora da parte de espectro visível, ela possui início abaixo do espectro de luz vermelho, indo de $\lambda \approx 700$ ηm a $\lambda \approx 0{,}1$ mm. O sinal IR é gerado através de qualquer dispositivo que possa emitir calor, e um dos mais utilizados para este procedimento são alguns modelos de Diodos Emissores de Luz (LED), e *lasers*. No caso

de controle remoto de aparelhos eletroeletrônicos, o LED é usado predominantemente [22].

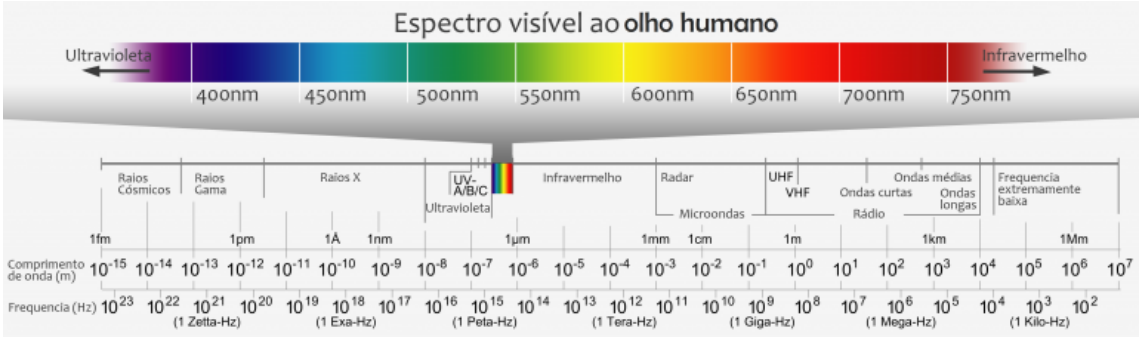

Figura 2.1. Espectro eletromagnético onde estão representadas todas as distribuições das ondas eletromagnéticas a partir de seu comprimento de onda e frequência. Fonte: [5]

O controle remoto IR, utiliza em sua estrutura um LED infravermelho emissor, como mostrado na Figura 2.2. O aparelho a ser controlado, possui um receptor IR, que decodifica este sinal para que o dispositivo possa entender e realize o comando respectivo. A forma de comunicação entre o sinal IR emissor e receptor é explicada a seguir.

Figura 2.2. LED emissor IR utilizado em um controle de ar condicionado para transmissão de dados. Fonte: [6]

2.1.2 Comunicação de Dados

A transmissão de informações entre o LED emissor e o receptor IR é feita através de códigos binários. Ao pressionar um botão no controle remoto, uma conexão é fechada

através de uma chave, que é detectada por um circuito integrado que gera um código binário específico daquela chave, transmitindo através do LED em uma série sistemática de pulsos infravermelhos ligando e desligando de acordo com o *bit*, onde o pulso possa ser representado por um bit igual a 1, e no caso de bit 0 o LED não emitiria nenhum pulso, como mostrado na Figura 2.3. Cada chave possui um código diferente para identificar cada função, e após a chave ser fechada o sinal passa por um transistor que amplifica o sinal a ser enviado pelo LED, que traduz os dados em luz infravermelha. [23][6].

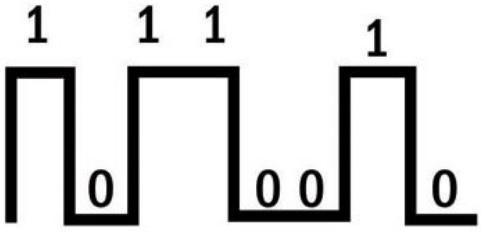

Figura 2.3. Pulsos em série onde o bit 1 é ligado, e o bit 0 é desligado. Fonte: [7] ADAPTADO.

Além de enviar os pulsos da função pressionada pelo botão, também é enviado juntamente em binário um código curto de identificação do produto a ser controlado, com a marca e modelo específico, garantindo que o controle remoto opere apenas no aparelho especificado, e não interfira em dispositivos que também possuem receptor infravermelho que estejam por perto [6].

A energia de transmissão do controle remoto IR é relativamente baixa, funcionando a uma distância do receptor até certa de 10 m de distância, além de ser passível de sofrer interferência de luz infravermelha como lâmpadas fluorescentes, raios solares e até mesmo o corpo humano.

Para que não ocorra essa interferência, os receptores IR respondem a sinais em um determinado comprimento de onda, geralmente a 980 ηm, além de também passar por um processo de modulação para outra frequência pelo fato dos raios solares também possuírem comprimentos de onda a 980 ηm. Não é um procedimento 100% eficiente, porém reduz drasticamente as interferências [6]. Por fim o sinal é decodificado e o aparelho a ser controlado responde de acordo com a função apertada no controle remoto.

2.1.3 Condicionamento de Sinal

Para recepção do sinal do controle remoto e transmissão, é necessário fazer o condicionamento do sinal para que ocorra uma comunicação dos dados a serem obtidos com

os microcontroladores, e vice-versa. Para recepção do sinal, é necessário a codificação do sinal advindo do controle remoto, e para transmissão, também é necessário a codificação e amplificação do sinal.

2.1.3.1 Recepção do Sinal Infravermelho

Para recepção de sinal do controle remoto, é utilizado o receptor infravermelho (VS1838), mostrado na Figura 2.4.

As características do receptor são:

- Tensão de operação: 2,7V a 5,5V;
- Distância de Recepção: 18m;
- Ângulo de Recepção: ± 45 graus;
- Baixo Nível de tensão: 0,4V;
- Alto Nível de tensão: 4.5V;
- Frequência da Portadora: 38KHz.

Figura 2.4. Receptor Infravermelho (VS1838) que transforma o sinal infravermelho recebido em saída digital para leitura do microcontrolador. Fonte: [8]

A pinagem do receptor IR é apresentada na Tabela 2.1.

Receptor IR (VS1838)	
1	Sinal
2	GND
3	VCC

Tabela 2.1. Pinagem do receptor infravermelho (VS1838) de 3 terminais, Sinal, VCC e GND.

O receptor (VS1838) além de ser um fotodiodo, também possui um invólucro metálico para fazer a filtragem de ruídos externos. Em seu circuito interno, o sinal recebido é amplificado, modulado e também filtrado, como mostrado na Figura 2.5, para que a saída do sensor seja digital, sendo 1 ou 0, para leitura correta do microcontrolador.

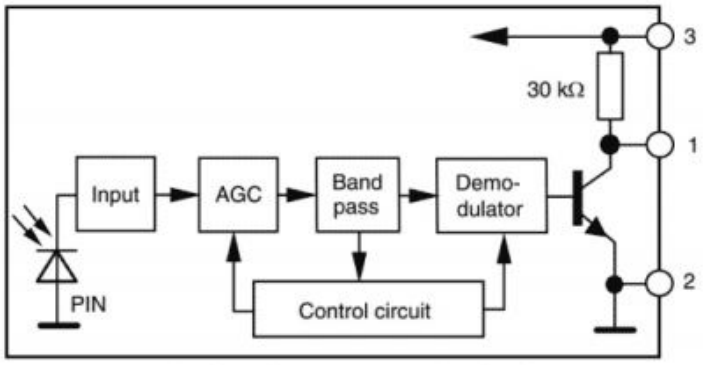

Figura 2.5. Diagrama de blocos que apresenta os processos realizados no sensor receptor infravermelho (VS1838), até sua saída para leitura do microcontrolador. Fonte: [9]

2.1.3.2 Transmissão do Sinal Infravermelho

- **Transistor BC547**

  Para transmissão do sinal é necessário primeiramente fazer a amplificação para que a área de alcance para controlar o dispositivo seja maior. Por isso é necessário a utilização de um transistor NPN BC547, mostrado na Figura 2.6. O BC547 foi escolhido por ser um transistor pequeno e de baixo consumo de energia.

Figura 2.6. Transistor NPN BC547 utilizado para realizar a amplificação do sinal IR de transmissão. Fonte: [10]

As características principais do transistor escolhido são:

- Máxima tensão de coletor: 45V;
- Máxima corrente de coletor: 100 mA;
- Ganho: 110 - 800.

A identificação dos pinos do transistor NPN BC547 é apresentada na Tabela 2.2.

Transistor NPN BC547	
1	Coletor
2	Base
3	Emissor

Tabela 2.2. Identificação dos pinos do transistor BC547, Coletor, Base e Emissor.

O transistor é um dispositivo semicondutor que possui três regiões de dopagem: base, coletor e emissor. Os transistores podem ser NPN ou PNP, sendo determinado se a região possui elétrons livres (N), ou lacunas, (P). O dispositivo NPN possui o coletor N, a base (P) e o emissor (N). A Figura 2.7 mostra o símbolo da direção convencional das correntes do transistor após a polarização, ou seja, após a conexão de uma fonte externa na região de dopagem da base. A Tabela 2.3 apresenta a legenda das correntes.

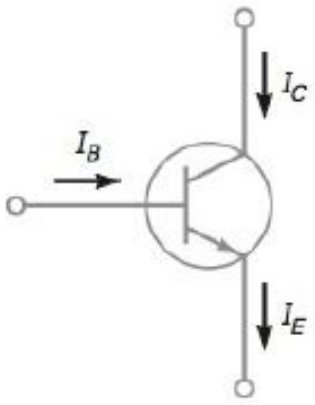

Figura 2.7. Símbolo do transistor NPN com a direção das correntes em fluxo convencional. Fonte: [11]

Corrente	Descrição
I_C	Corrente no coletor
I_B	Corrente na base
I_E	Corrente no emissor

Tabela 2.3. Identificação das correntes mostradas na Figura 2.7.

A região do coletor é onde mais se encontra elétrons livres, sendo a área mais dopada do transistor. Isso acontece pelo fato da região da base ser muito estreita e fracamente dopada, além de possuir uma distância muito curta para o coletor, passando para os mesmos elétrons injetados pelo emissor. Com a aplicação da Lei de Kirchhoff, temos a relação mostrada na Equação 2.1:

$$I_E = I_B + I_C \tag{2.1}$$

Pelo fato da corrente de base ser muito pequena, é possível concluir nas Equações 2.2 e 2.3:

$$I_E \approx I_C \tag{2.2}$$

$$I_B \ll I_C \tag{2.3}$$

O valor de ganho de corrente do transistor é mostrada na Equação 2.4, e a partir dela é possível determinar a corrente de coletor ou corrente de base a partir de uma fórmula derivada da Euqação2.4, mostrada na Equação 2.5e Equação 2.6.

$$\beta_{CC} = \frac{I_C}{I_B} \tag{2.4}$$

$$I_C = \beta_{CC} * I_B \tag{2.5}$$

$$I_B = \frac{I_C}{\beta_{CC}} \tag{2.6}$$

Uma característica importante do transistor é quando o emissor é conectado ao GND. Quando ocorre esta situação, a tensão (V_B) conectada à base polariza o diodo emissor de forma direta, fazendo com que o resistor(R_B) conectado entre eles consiga operar como limitador de corrente. Isso faz com que a tensão de saída no coletor seja controlada através da V_B e R_B, onde uma corrente baixa I_B controle uma corrente alta I_C [11].

- **LED Emissor Infravermelho**

  O componente utilizado para transmissão do sinal ao aparelho a ser controlado é o LED emissor de infravermelho, mostrado na Figura 2.8. Este dispositivo é capaz de converter sinal elétrico em sinal luminoso, realizando as funções escolhidas pelo usuário.

Figura 2.8. LED emissor de infravermelho utilizado para transmitir o sinal para o dispositivo receptor a ser controlado. Fonte: [1]

As características principais do LED são:

  – Comprimento de onda: 940ηm;

  – Tensão de operação: 1.2 à 1.4V;

  – Diâmetro: 5mm;

Alegenda dos pinos do LED emissor de IR é apresentada na Tabela 2.4.

LED Emissor de IR	
1	Catodo (-)
2	Anodo (+)

Tabela 2.4. Pinagem do LED emissor infravermelho de 5mm e 2 terminais. Fonte: [1]

2.1.3.3 Detecção de estado do Ar Condicionado

Para detectar o estado do ar condicionado, ligado ou desligado, é utilizado um sensor de temperatura e umidade em frente a saída de ar, fazendo a leitura e interpretação em relação aos dados. Para escolher o sensor correto, foram analisadas duas opções, DHT11 e DHT22, ambos de mesmo fabricante, mas com comportamentos diferentes, como é mostrado na Tabela 2.5.

Especificações	**DHT 11**	**DHT 22**
Tensão de Alimentação	3V - 5.5V	3.3V - 6V
Corrente Máxima de Captura	2.5 mA	1.5 mA
Faixa de Leitura (Umidade)	20 - 80%	0 - 100%
Precisão (Umidade)	5%	2%
Faixa de Leitura (Temperatura)	0 - 50 ºC	-40 ºC - 125 ºC
Precisão (Umidade)	+/-2 ºC	+/- 0.5 ºC
Intervalo entre Medições	1 s	2 s
Preço Médio	R$ 15,00	R$ 35,00

Tabela 2.5. Comparação das especificações dos sensores de temperatura e umidade DHT11 e DHT22.

Para a situação de apenas determinar se o ar condicionado está ligado ou desligado, é possível utilizar o DHT11, mostrado na Figura 2.9, pois ao usufruir de dados de umidade e temperatura em conjunto observando a suas variações, chegamos a valores distintos entre um estado e outro.

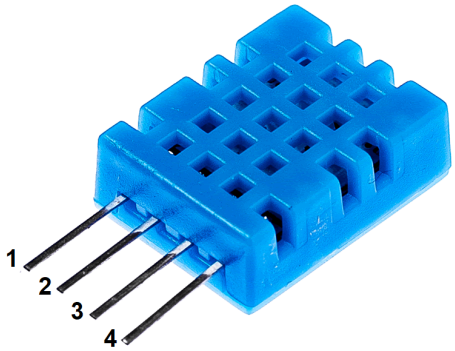

Figura 2.9. Sensor de Unidade e Temperatura DHT11 e a numeração dos seus pinos. Fonte: [12].

A Tabela 2.6 identifica a funcionalidade de cada pino do DHT11.

DHT 11	
1	VCC
2	DATA
3	N/A
3	GND

Tabela 2.6. Identificação dos pinos do sensor de temperatura e umidade DHT11.

2.2 Desenvolvimento WEB

O desenvolvimento World Wide Web (WEB) será feito entre o servidor e cliente. O servidor WEB será implementada pela placa Raspberry Pi 3, que servirá de interface para que o usuário consiga acessar os botões do controle remoto, e este comando interaja com o cliente WEB, que será acessado pelo microcontrolador ESP8266.

2.2.1 Raspberry Pi 3

A Raspberry Pi 3, mostrada na Figura 2.10 é um computador de baixo custo que possui uma placa contendo diversos componentes integrados. Além disso, também é

possível conectá-la a outros dispositivos externos através de um grupo de pinos chamados General Purpose Input/Output (GPIO), o que torna a placa ser compatível para se trabalhar com o IOT [24].

Figura 2.10. Placa Raspberry Pi 3 Modelo B que será utilizado como computador central com interface *web* entre o usuário e o cliente [13].

Dentre suas características, a Raspberry Pi 3 possui:

- Sistema operacional Raspbian, baseado na distribuição Debian do Linux.
- Processador de 64 *bits* Broadcom 2837 ARMv8;
- Clock de 1.2 GHz;
- Memória *Random Access Memory* (RAM) de 1 *GBytes*;
- Sistema de bluetooth integrado;
- Sistema de *Wi-Fi* integrado 2.4 GHz;
- 40 pinos GPIO;
- *Slot* para cartão *SD High Capacity* (SDHC);
- 1 saída *High-Definition Multimedia Interface* (HDMI);
- 4 portas *Universal Serial Bus* (USB);
- 1 conector RJ45;
- 1 soquete microUSB.

A escolha da Raspberry Pi 3 para o projeto se deu pelo fato dela ser a responsável por ser o elemento central que irá se comunicar com todos os microcontroladores baratos que serão clientes. Sua versão 3 de 64 *bits* possui um desempenho e recursos *wireless* aprimorados para que esta conexão conjunta tenha uma compatibilidade de software, desempenho e segurança muito maior que os 32 *bits* incorporados nas versões anteriores. Além disto, caso ocorra uma expansão do trabalho com adição de funções e módulos, a placa responderá prontamente sem qualquer problema de espaço e conexão [25].

2.2.2 NodeMCU ESP8266

A ESP32866, é um microcontrolador de 32 *bits* fabricada por uma empresa chinesa chamada Espressif. É amplamente utilizada em IOT por causa de suas interfaces de entrada e de saída e seu módulo integrado *Wi-Fi*. Dentre os diversos modelos fabricados, a placa escolhida para o projeto foi a do Módulo ESP8266 NodeMCU ESP-12E, mostrada na Figura 2.11. Ela será utilizada como cliente para se conectar ao servidor, resultando na ação gerada pelo usuário para controle do ar condicionado, através de um LED infravermelho.

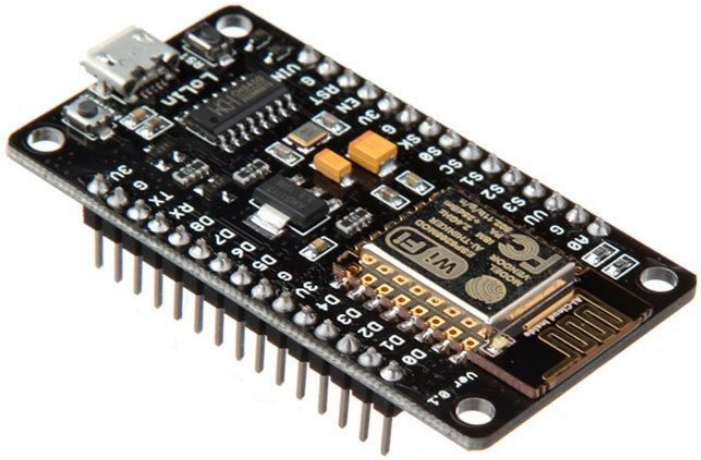

Figura 2.11. Módulo ESP8266 NodeMCU ESP-12E que será utilizado como cliente acessando o servidor na Raspberry Pi 3 e acionando o infravermelho para controle do ar condicionado [14].

Dentre suas características, a ESP8266 possui:

- Microcontrolador *Reduced Instruction Set Computer* (RISC) 32 *bits*;
- *Clock* de 80 MHz;
- Memória RAM de 60 KB;

- 16 pinos de GPIO;
- 1 entrada microUSB;
- Memória *flash* de 4 MB;
- Pinagens com interface *Serial Peripheral Interface* (SPI), *Inter-Integrated Circuit* (I2C), *Inter-IC Sound* (I2S);
- Pinagem *Universal Asynchrounous Receiver/Transmiter* (UART);
- Conversor analógico digital de 10 *bits*;
- Conectividade *Wi-Fi* compatível com o padrão IEEE 80.11 b/g/n, com autenticação WPA/WPA2 e WEP.

A ESP8266 foi escolhido pelo fato de possuir um baixo consumo de energia, por ser um microcontrolador barato com módulo *wireless* integrado, e pela compatibilidade com o *framework* da Espressif de desenvolvimento de software ESP-IDF, que fornece recursos básicos que ajudam o desenvolvedor com o Sistema Operacional em Tempo Real (RTOS) e com o Kit de Desenvolvimento de Software (SDK), que formam a estrutura mostrada na Figura 2.12.

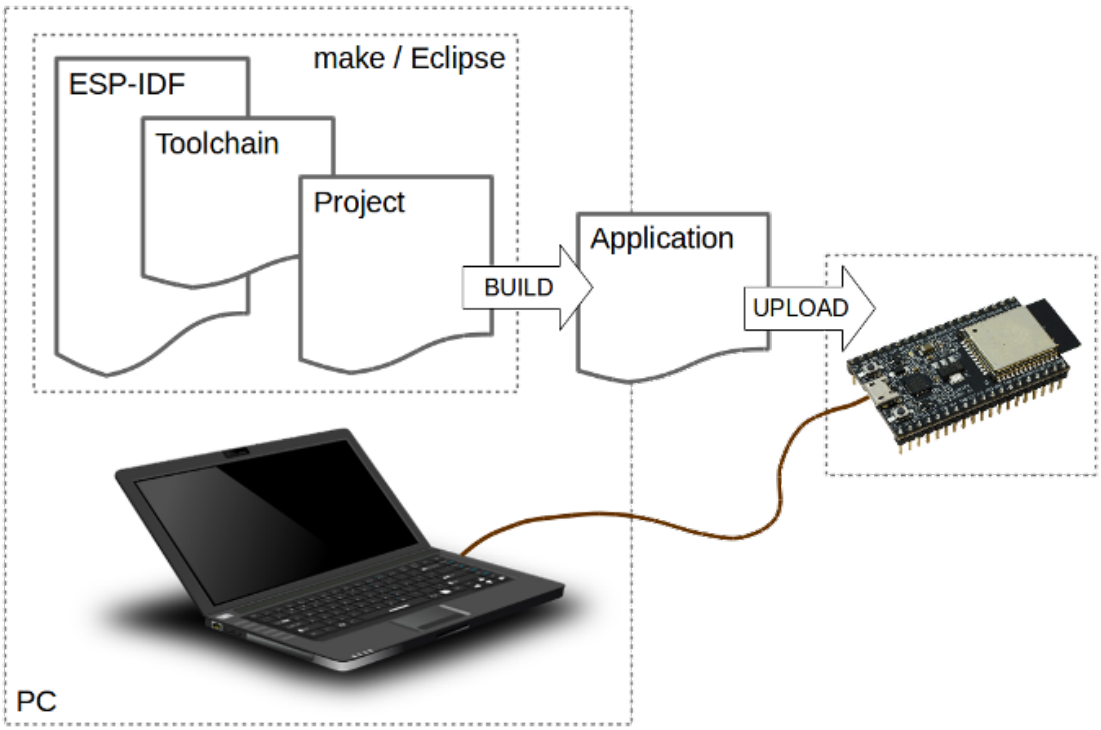

Figura 2.12. Estrutura de desenvolvimento de aplicativos para a ESP8266 da instalação até o carregamento na placa [15].

A figura 2.12 mostra que no computador o início ocorre na etapa *make*. Nesta etapa se encontra o *framework* ESP-IDF, o conjunto de programas para compilar e construir o aplicativo *toolchain*, a configuração do projeto e escrita do código em *project*, e a compilação do código para aplicação, no *build*. Para finalizar o processo, é feito o *upload* na placa a ser utilizada, conectada através do cabo USB [15].

2.2.3 Protocolos de Comunicação

Os protocolos de comunicação são definidos como um conjunto de normas e procedimentos para iniciar, manter e concluir uma comunicação entre dois pontos, garantindo que esta troca de informação ocorra de forma ordenada e sem erros. [26]. Existem vários tipos de protocolos que podem fazer esta comunicação entre o transmissor e o receptor, que irão se comunicar através do uso da mesma linguagem [26].

2.2.3.1 Modelo OSI

A arquitetura Interconexão de Sistemas Abertos (OSI), definida pela Organização Internacional de Normalização (ISO) e mostrada na Tabela 2.7, possui como definição a partição de funcionalidades de rede em sete camadas. Em cada nível, existem protocolos que implementando uma determinada função [4].

Modelo OSI	
Camada 1	Aplicação
Camada 2	Apresentação
Camada 3	Sessão
Camada 4	Transporte
Camada 5	Rede
Camada 6	Enlace
Camada 7	Física

Tabela 2.7. As sete camadas do modelo OSI e as espécies de protocolo correspondentes a cada nível [2].

2.2.3.2 Modelo TCP/IP

O modelo mais utilizado e conhecido é o *Transmission Control Protocol/Internet Protocol* (TCP/IP). Diferentemente do modelo OSI, o TCP/IP possui apenas 4 camadas, sendo este modelo aplicado à internet, como mostrado na Tabela 2.8. Uma das principais vantagens da utilização do TCP/IP é que ele pode ser utilizado tanto em redes de longas distâncias quanto em redes locais, além de se adaptar a sub-redes de diferentes velocidades de diferentes dispositivos.

Modelo TCP/IP	
Camada 1	Aplicação
Camada 2	Transporte
Camada 3	Internet
Camada 4	Host/Rede

Tabela 2.8. As quatro camadas do modelo TCP/IP e as espécies de protocolo correspondentes a cada nível [2].

- Host/Rede: Possui como função a conexão do host à rede utilizando algum protocolo para que seja possível enviar pacotes IP.
- Internet: Possui a função de permitir que os *hosts* injetem pacotes em qualquer rede garantindo que eles trafegarão de forma independente até o destino. O IP é o principal protocolo desta camada no modelo TCP/IP.

O IP é um protocolo que utiliza endereçamento para que o pacote de dados seja enviado para a estação de rede correta. Este endereçamento é formado por 32 *bits*, onde estes *bits* são separados entre a rede e o *host*. Também é utilizada a máscara de rede, que serve para determinar onde termina o endereço da rede e começa o endereço o *host*.

- Transporte: Possui a função de permitir que as entidades pares dos hosts de origem e de destino mantenham uma conexão. O TCP/IP possui dois protocolos para implementação do transporte, sendo o TCP e o *User Datagram Protocol* (UDP).

O TCP é utilizado para conexões confiáveis, correspondente a um controle na entrega de fluxo de *bytes* sem erros em uma determinada comunicação.

O UDP é utilizado para um serviço não orientado a conexão, ou seja, ele envia todos seus pacotes por acreditar que todos os dados serão recebidos sem erros e na sequência de envio.

- Aplicação: Possui a função de dar suporte às aplicações dos usuários. Sua comunicação é fornecida ao usuário através de conexão lógica, disponibilizando todos as ferramentas e recursos destinados a ele. Um dos principais protocolos desta camada é o *Hypertext Transfer Protocol* (HTTP).

2.2.3.3 Protocolo HTTP

O HTTP é um protocolo utilizado para realizar a comunicação entre navegadores e servidores *web*, permitindo também a transferência de arquivos usando serviços do TCP em uma única comunicação, onde os dados são transferidos entre cliente e servidor. Para comunicação de um cliente com o servidor, os comandos são enviados como uma mensagem de solicitação, e a resposta do servidor podem ser inseridos em uma mensagem contendo um conteúdo como um arquivo ou qualquer outra informação, como mostrado na Figura 2.13. O HTTP utiliza os serviços do TCP através da porta 80 [3] [4].

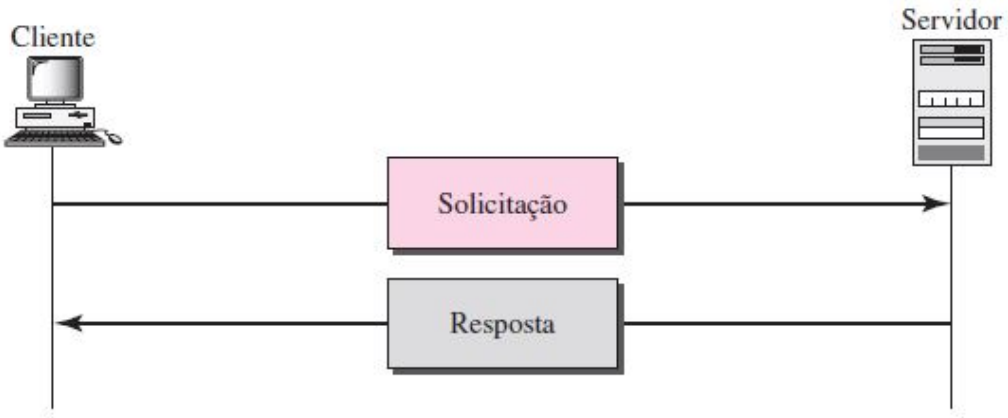

Figura 2.13. Protocolo HTTP em uma comunicação entre servidor e cliente, onde o cliente faz a solicitação para o servidor, e o servidor envia uma informação como resposta ao cliente [3].

A estrutura das mensagens de solicitação e respostas entre servidor e cliente são apresentados na Tabela 2.9.

Cliente	Servidor
Linha de Solicitação	Linha de Status
Cabeçalho	Cabeçalho
Linha em Branco	Linha em Branco
Corpo	Corpo

Tabela 2.9. Estrutura da mensagem de solicitação do cliente e de resposta do servidor no protocolo HTTP [3].

Na Tabela 2.9, a estrutura da mensagem do cliente é formada da seguinte maneira:

- Linha de Solicitação: É formada a partir do tipo de solicitação, URL, e a versão do HTTP, onde entre eles é dado um espaço em branco. Dentre os tipos de solicitações que podem ser utilizados, temos o método GET e o método POST. Ao utilizar o método GET, os parâmetros são passados através da *Uniform Resource Locator* (URL), ou seja, no cabeçalho. Já no método POST, os parâmetros são passados através do corpo da requisição [27]. A versão mais atual do HTTP é a 1.1.
- Cabeçalho: Permite que ocorra o envio de informações adicionais para o servidor. Pode ser formado por até duas linhas, e pode conter apenas cabeçalhos gerais, de solicitação e de entidade.
- Corpo: Onde se encontra o documento a ser transmitido.

Já a estrutura da mensagem do servidor possui algumas classificações diferentes:

- Linha de Status: É formada a partir da versão do HTTP, código do status e frase de status, onde entre eles é dado um espaço em branco. A Tabela 2.10 apresenta os tipos de códigos que podem ser retornados, sendo que o mesmo possui três dígitos.
- Cabeçalho: Permite que ocorra o recebimento e envio de informações adicionais para o cliente. Pode ser formado por até duas linhas, e pode conter apenas cabeçalhos gerais, de resposta e de entidade.
- Corpo: Onde se encontra o documento a ser recebido. Utilizado principalmente no método POST.

Código	Status	Descrição
1xx	Informação	Solicitação recebida, continuando processo.
2xx	Sucesso	Ação recebida com sucesso, entendida e aceita.
3xx	Redirecionamento	Uma ação adicional precisa ser tomada.
4xx	Erro do Cliente	Solicitação contém sintaxe errada ou não pode ser atendida.
5xx	Erro do Servidor	Servidor foi incapaz de processar.

Tabela 2.10. Tipos de códigos resultantes do processamento do protocolo HTTP na conexão entre servidor e cliente [4]

.

2.2.3.4 Protocolo TCP

Como mencionado no modelo TCP/IP na camada de transporte, o protocolo TCP possui uma conexão orientada e confiável, adquirindo vantagem em relação à conexão UDP. A conexão TCP possui três fases: estabelecimento de conexão, transferência de dados e por fim o encerramento da conexão.

A comunicação do protocolo TCP é feita através de uma conexão virtual de fim a fim, onde é criada virtualmente uma conexão entre dois processos através de uma porta, que possui a numeração 80 para comunicação HTTP. Essa porta funciona com um canal em que os dados são transportados entre o servidor e o cliente [3]. Estes dados transmitidos são escritos em *bytes*, e passam por um processo até chegar no transmissor, apresentado na Figura 2.14.

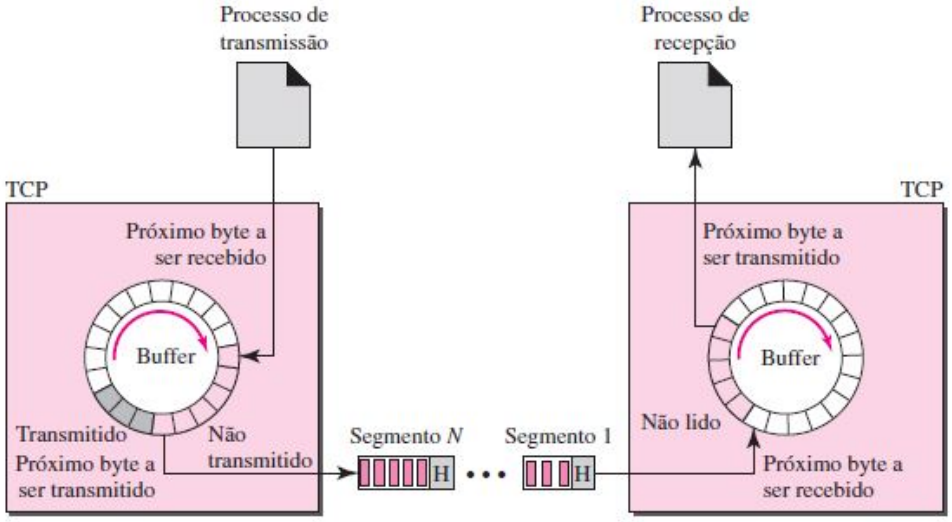

Figura 2.14. Gerenciamento de um fluxo de *bytes* do processo de transmissão até o processo de recepção dos dados [3].

O processo mostrado na Figura 2.14, apresenta que no *host* de origem o TCP mantém no *buffer*, *bytes* o suficiente advindos do processo de transmissão para preenchimento de um pacote. Posteriormente este pacote, chamado de segmento, é enviado para o *host* de destino onde o TCP transfere todo seu conteúdo para *buffer* de recepção, onde serão lidos [4].

Uma característica do protocolo TCP é pelo fato dele ter a opção de funcionar de maneira *full-duplex*, ou seja, a transferência de dados entre servidor e cliente podem ser encaminhados de ambos os lados simultaneamente, pelo fato de cada processo TCP possuir um *buffer* de transmissão e recepção. Na Figura 2.14, é mostrado o processo apenas com fluxo de dados em apenas um sentido.

O segmento é enviado contendo um cabeçalho para fins de controle, com campos no seguinte formato :

- Endereço da porta de origem: Possui 16 *bits* com o número da porta do *host* que está enviando o segmento.
- Endereço da porta de destino: Possui 16 *bits* com o número da porta do *host* que está recebendo o segmento.
- Número de seqüência: Possui 32 *bits* com o número atribuído ao primeiro *byte* de dados do segmento. Todos os *bytes* são numerados para garantir a confiabilidade na comunicação.

- Número de confirmação: Possui 32 *bits* com o número de *bytes* que o receptor está esperando receber do transmissor. No caso de sucesso na recepção, o número de confirmação é definido como x + 1, onde x é o número de *bytes* enviados.
- Comprimento do cabeçalho: Possui 4 *bits* com a quantidade de palavras de 4 *bytes* no cabeçalho, sendo esse valor entre 20 a 60 *bytes.*
- Reservado: Possui 6 *bits* reservados para utilização futura.
- Controle: Possui 6 *bits* para *flags* de controle. A Tabela 3.3 apresentam as *flags* que podem ser configuradas.
- Tamanho da janela: Possui 16 *bits* definindo o tamanho da janela em *bytes*, que o transmissor deve manter.
- *Checksum*: Possui 16 *bits* com o cálculo do *checksum.*
- *Urgent Pointer* (URG): Possui 16 *bits* quando o URG é acionado no campo de controle, aplicando urgência ao segmento transmitido.
- Opções: Até 40 *bytes* para informação opcional a serem adicionadas no cabeçalho.

Flag	**Descrição**
URG	Valor *Urgent Point* é válido.
ACK	Valor de confirmação é válido.
PSH	Empurra os dados.
RST	Reinicia a conexão.
SYN	Sincronização de números de sequência.
FIN	Encerra a conexão.

Tabela 2.11. Flags que possibilitam implementar o controle de fluxo no campo controle no cabeçalho do segmento [3]

.

2.2.3.5 Socket TCP

O *socket* TCP é um meio abstrato no qual um aplicativo pode enviar e receber dados. Um *socket* permite que um aplicativo seja conectado à rede e se comunique com outros aplicativos que estejam conectados à mesma rede. Os principais tipos de *sockets* no TCP/IP são de fluxo, TCP, e de datagrama, UDP. Os *sockets* de fluxo fornecem um serviço de fluxo de *bytes* confiável [28].

- Servidor Web

No caso de um servidor IP versão 4 (IPv4), sua função é configurar um *endpoint* de comunicação e aguardar passivamente por uma conexão do cliente. Para realização deste processo TCP, existem no geral seis etapas:

1. Criação do *socket*: É realizado através da função *socket(int domain, int type, int protocol)* com três parâmetros, determinando a família de protocolos a ser utilizada, a semântica de comunicação e o protocolo específico a ser utilizado. A Tabela 2.12 apresenta as opções de parâmetros.

Domínio	
PF_INET	Família Internet.
PF_UNIX	Funcionalidade de *pipe* do Unix.
PF_PACKET	Acesso direto à interface de rede.
Tipo	
SOCK_STREAM	Fluxo de *bytes*.
SOCK_DGRAM	Serviço orientado a mensagens.
Protocolo	
UNSPEC	Combinação PF_INET com SOCK_STREAM (TCP)

Tabela 2.12. Opções para configuração dos parâmetros domínio, tipo e protocolo na função *socket*() para criação do mesmo [4].

2. Atribuir um número de porta ao *socket* com a função *bind(int socket, struct sockaddr*address, int addr_len)*: O *socket* recém criado é vinculado a uma porta e um endereço especificado.

3. Conexões na porta com a função *listen(int socket, int backlog)*: Permite conexões de entrada de clientes para comunicação.

   A partir do próximo passo, o processo é feito repetidamente.

4. Obter um novo *socket* para cada cliente com a função *accept(int socket, struct sockaddr*address, int addr_len)*: Função que bloqueia e não retorna até que uma conexão seja estabelecida, e quando a conexão é completada, um novo *socket* é retornado e seu argumento *address* possui o de endereço do cliente.

5. Comunicação com o cliente através das funções *send(int socket, char *message, int buf_len, int flags)* e *recv(int socket, char *buffer, int msg_len, int flags)*: A função *send* envia a mensagem pelo *socket* quando passada como um parâmetro.

6. Encerrar a conexão com a função *close(int socket)*: Utilizado para finalizar a comunicação entre servidor e cliente, quando não é mais necessário a troca de informação [4][28].

- Cliente Web

No caso de um cliente IP versão 4 (IPv4), sua função é iniciar a comunicação com um servidor que está aguardando passivamente ser contatado. Para realização deste processo TCP, existem no geral quatro etapas, e a única diferença para o servidor é a exclusão da função *accept()* e a inclusão da função *connect()*:

- A função *connect(int socket, struct sockaddr*address, int addr_len)* estabelece uma conexão entre servidor e cliente para que ocorra a transferência de dados [4][28].

2.2.4 Flask

O *flask* é uma ferramenta útil para criação de aplicações *web*, pelo fato da possibilidade de flexibilização para criação de Interface de Programação de Aplicativos (API), com possibilidade de gerenciamento de dados enviados e recebidos [29]. Este *framework* é uma biblioteca do *python*.

Para utilização do *flask*, inicialmente ele deve ser instanciado através do $Flask(\_\_name\_\_)$. O argumento *name* é utilizado para determinar o caminho raiz da aplicação para que depois seja possível encontrar a localização dos arquivos de recursos relativos. O próximo passo para construção do servidor é fazer o mapeamento da URL para fazer o instanciamento quando o cliente realizar as solicitações ao servidor, criando uma rota, definida através do $@app.route('/')$ [30].

Após a definição da rota, é chamada uma função $index()$ para acionar a URL e a rota no navegador, retornando ao cliente uma resposta ao usuário, podendo ser usada uma página *Linguagem de Marcação de HiperTexto* (HTML), implementando a função $index()$ como uma função de visualização [30].

Uma grande vantagem deste *framework* é a utilização de rota que possui um nome dinâmico, fazendo que qualquer URL seja mapeada. Essa função é bastante utilizada para realizar uma ação de acordo com a rota que está sendo requisitada pelo cliente, sendo possível mapear suas necessidades, principalmente quando se tem as rotas pré estabelecidas. As rotas suportadas pelo *flask* podem ser do tipo *int, float* e *path* para rotas, porém o padrão utilizado é do tipo *string* [30].

Para iniciar e execução do servidor, a aplicação é instanciada de forma a garantir que o servidor seja iniciado apenas quando o script *python* for executado diretamente, através do $if\_\_name\_\_ ==' \_\_main\_\_' :$. Após este procedimento, é utilizado a função *app.run*, que podem ser utilizados diversos argumentos para configurar o modo de operação do servidor. Os dois argumentos principais são o *debug*, que pode ser habilitado recebendo o valor *True* para habilitar o modo de depuração, e o *host*, recebendo como valor o endereço e a porta que o servidor será executado [30].

Em relação à resposta ao cliente dentro da função $index()$, o *flask* procura por *templates* em uma subpasta *templates* localizada dentro da pasta da aplicação. Utilizando $returnrender_template('name.html')$, ele automaticamente procura na pasta o arquivo HTML para retornar ao cliente, otimizando a estrutura do código [30].

- **HTML**

  O HTML é uma linguagem que descreve tudo sobre o conteúdo e a estruturação de uma página web, e através da inserção em um servidor, qualquer navegador poderá ter acesso à pagina desenvolvida. A partir do momento em que o navegador acessa a página web, ele faz a leitura do HTML e interpreta as *tags*, que tem o objetivo de informar o significado e a estrutura do texto, separando títulos, parágrafos, imagens, e etc [31]. As principais *tags* são:

  (a) <html>: É a raiz de um arquivo HTML. Todo o conteúdo da página irá dentro desta *tag*.

  (b) <title>: Texto que define o título do documento a ser mostrado na bara de títulos, aba, do navegador.

  (c) <head>: Definição de dados, conjunto de caracteres, estilos, scripts e outras informações. Como por exemplo o endereçamento do arquivo Folhas de Estilo em Cascata (CSS).

  (d) <body>: Engloba todo conteúdo de um arquivo HTML, como títulos, imagens, parágrafos, hiperlinks, tabelas, entre outros [32].

- CSS

  O CSS é uma linguagem de marcação desenvolvida para estilizar as páginas web. As vantagens de se utilizar o CSS são: utilizar um arquivo separado do código HTML para formatação, deixando o arquivo principal organizado, ter a capacidade de alterar a aparência de um site inteiro apenas em um único arquivo, e possuir um controle maior sobre os elementos da página em HTML [33].

3 Metodologia

Para se obter a melhor metodologia a ser utilizada no trabalho, foi feito um estudo de duas abordagens, chamadas de *bottom up* e *top down*.

Na metodologia *bottom up*, de baixo para cima, primeiramente é feita uma análise e comportamento de partes e elementos mais básicos para formar um resultado e uma percepção mais abrangente. Já na metodologia *top down*, de cima para baixo, primeiramente é feita uma análise do problema como um todo, utilizando-se uma visão mais abrangente, e a partir disso este problema é dividido em partes, sendo quebrado em problemas menores.

Como o problema a ser trabalhado já é visto em um sistema como um todo, então a processo que se adequa ao projeto é a metodologia *top down*.

3.1 Metodologia *Top Down*

Após determinar a metodologia a ser utilizada, foi determinado o fluxo de projeto para o desenvolvimento do mesmo, mostrado na Figura 3.1. A partir da metodologia, após determinar o sistema geral do projeto, o mesmo é dividido em subsistemas, para que o resultado seja alcançado em partes, facilitando o desenvolvimento do trabalho.

3.1.1 Pesquisa bibliográfica

Inicialmente o problema geral foi analisado, e através de estudos bibliográficos por meio de livros, artigos e sites confiáveis foram determinados o sistema geral e todos os subsistemas que o compõem.

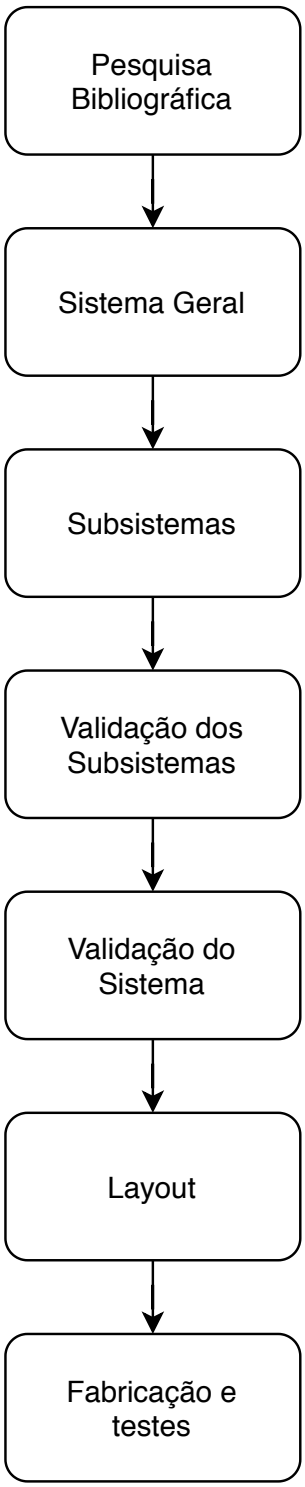

Figura 3.1. Fluxo de Projeto *Top Down*. Fonte: Elaborada pelo autor. Software DRAW.IO.

3.1.2 Sistema Geral

O sistema geral, mostrado na Figura 3.2, engloba todos os subsistemas determinados do projeto. A ideia do sistema geral é conseguir controlar ar condicionados através de um servidor *web*, e a partir da metodologia *top down*, criar subsistemas para facilitar o desenvolvimento do projeto.

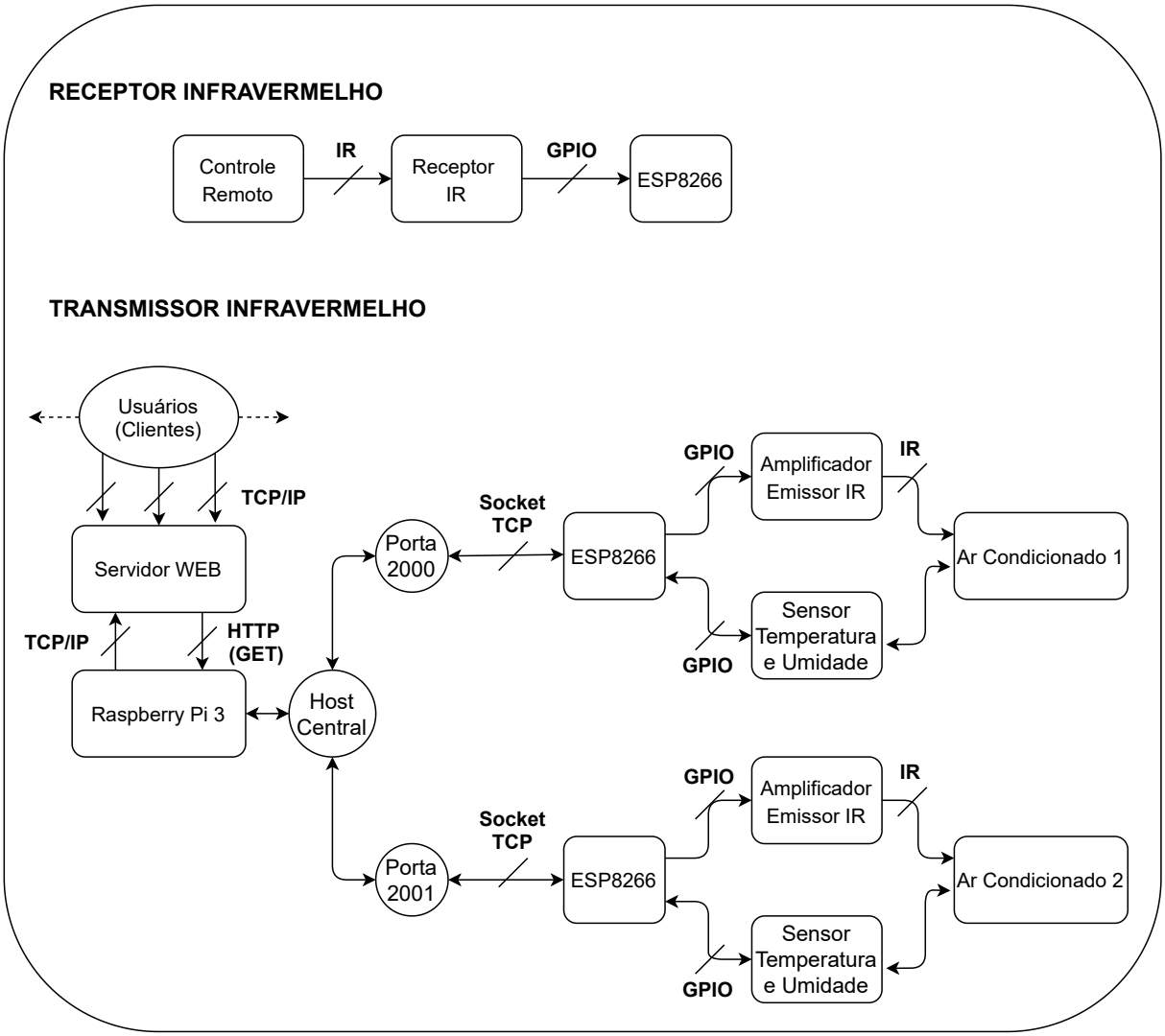

Figura 3.2. Sistema geral do projeto com protocolos de comunicação utilizados entre os módulos. Fonte: Elaborada pelo autor. Software DRAW.IO.

Na Figura 3.2, as setas pontilhadas indicam que existem uma quantidade indeterminada daqueles componentes, ficando a critério da aplicação.

Os subsistemas dentro do sistema geral são mostrados na Figura 3.3, para facilitar a visualização de como o projeto foi dividido.

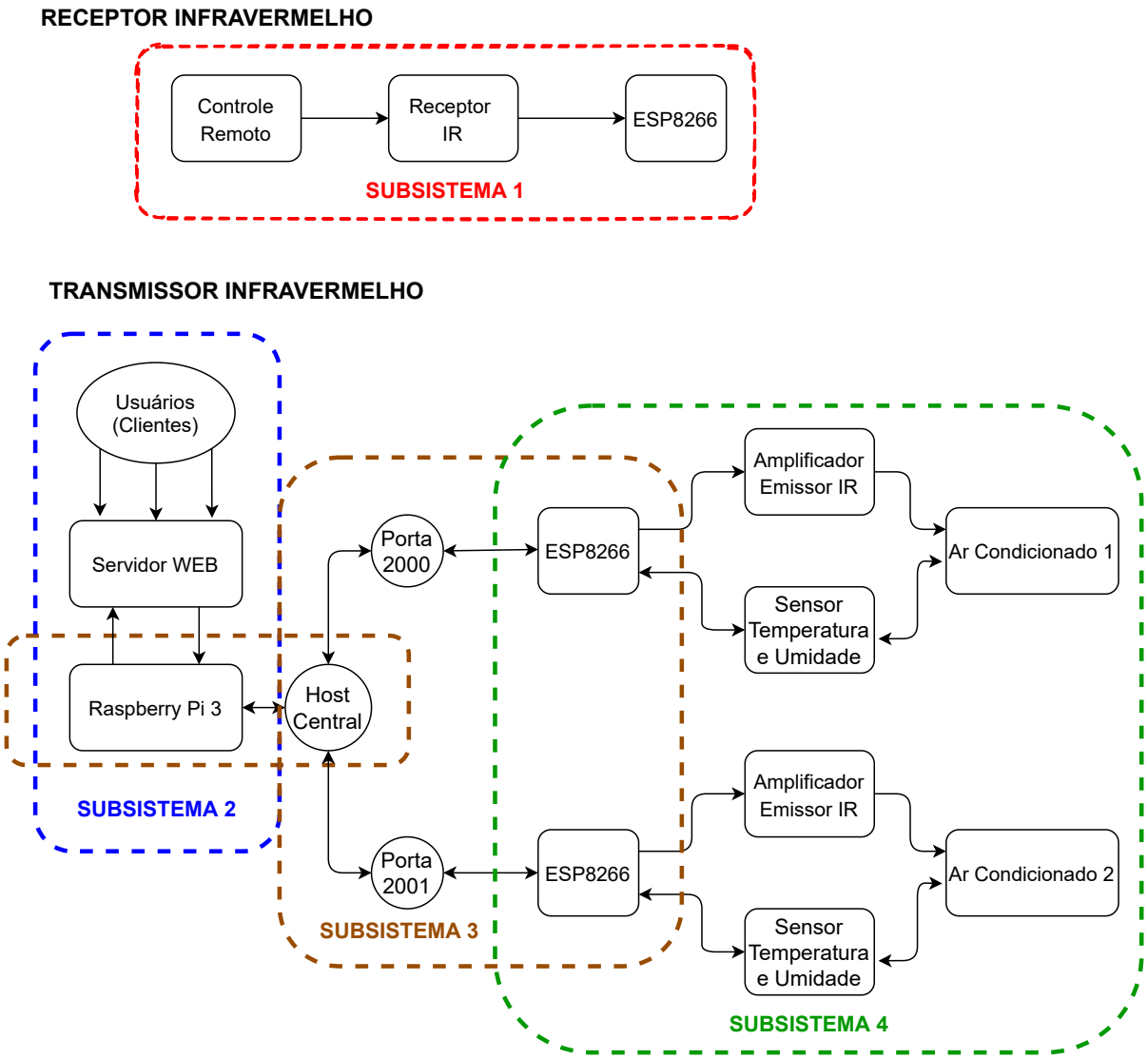

Figura 3.3. Subsistemas dentro do sistema geral para facilitar a implementação baseada na metodologia *Top Down*. Fonte: Elaborada pelo autor. Software DRAW.IO.

3.1.3 Subsistemas

O sistema geral foi dividido em quatro subsistemas, numerados de 1 a 4 apresentados na Figura 3.3

- Subsistema 1 - Recepção IR;
- Subsistema 2 - Servidor *Web*;
- Subsistema 3 - Comunicação Servidor - Cliente (ESP8266);
- Subsistema 4 - Transmissor IR e Leitor de Estado.

3.1.3.1 Subsistema 1 - Recepção IR

Neste subsistema, mostrado na Figura 3.4, ocorre a recepção do sinal infravermelho a partir do controle remoto. O dispositivo utilizado é o receptor IR VS1838, que transforma a luz infravermelha do controle remoto em sinal elétrico, sendo este lido e processado pelo ESP8266.

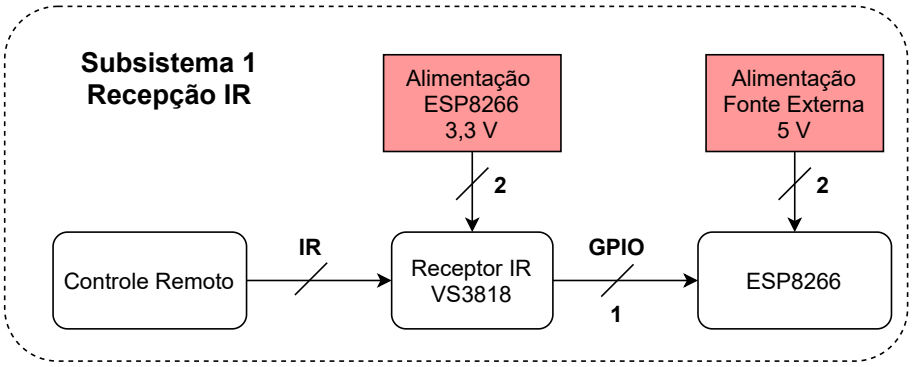

Figura 3.4. Subsistema de recepção do sinal infravermelho através do controle remoto para ser recebido na ESP8266. Fonte: Elaborada pelo autor. Software DRAW.IO.

O esquemático que apresenta a conexão do receptor IR com o ESP8266 é mostrado na Figura 3.5. Para montagem foi utilizado o Software *fritzing* e a Tabela 2.1 na subseção 2.1.3.1.

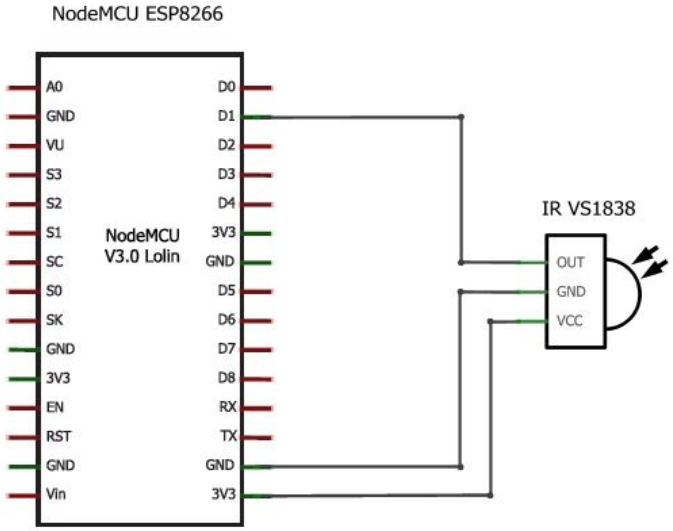

Figura 3.5. Esquemático que mostra as conexões entre o receptor do sinal infravemelho e a ESP8266. Fonte: Elaborada pelo autor. Software FRITZING.

3.1.3.2 Subsistema 2 - Servidor Web

Neste subsistema, mostrado na Figura 3.6, ocorre a hospedagem do servidor na internet através do IP da própria Raspberry Pi 3, que foi configurada como IP estático para que seu endereço não mudasse quando reiniciado. O servidor fica disponível para o usuário utilizando o *framwork flask*, que retorna uma página HTML para o cliente quando acessa a URL através de outro dispositivo, podendo selecionar qual ar condicionado quer controlar e a partir disso realizar ações nos botões apresentados.

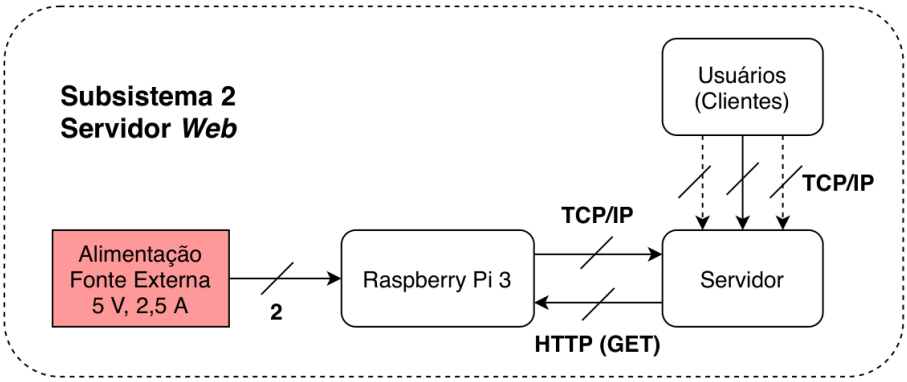

Figura 3.6. Subsistema 2 onde ocorre a hospedagem do servidor e a comunicação com o usuário através de outros dispositivos. Fonte: Elaborada pelo autor. Software DRAW.IO.

Utilizando o elemento <select>, foi criada uma lista com um menu de opções com a *tag* <option>, para determinar qual sala e qual ar condicionado da sala deseja controlar. Ao selecionar a opção desejada e apertar o botão para selecionar, é enviado um valor referente à opção escolhida através do método POST, mostrado na Tabela 3.1, sendo este recebido pela Raspberry Pi 3.

Sala	Ar Condicionado	Valor
S1	Ar Condicionado 1	s101
S1	Ar Condicionado 2	s102
S2	Ar Condicionado 1	s201

Tabela 3.1. Valores determinados para envio pelo método POST para identificar qual o ar condicionado escolhido pelo usuário.

Após escolher qual ar condicionado deseja-se controlar, é apresentado ao usuário um controle com 3 botões. Quando o usuário aperta um botão, o protocolo HTTP pelo método GET é acionado, identificando qual ação dever ser realizada através das rotas.

A Tabela 3.2 apresenta as rotas a serem utilizadas com apertos dos botões, que serão 3 opções: ligar/desligar, aumentar temperatura, e diminuir temperatura.

Ar Condicionado	
Função	**Rota**
Ligar/Desligar	/ar/onoff
Aumentar Temperatura	/ar/tempmais
Diminuir Temperatura	/ar/tempmenos

Tabela 3.2. Rotas geradas pelo cliente ao apertar botões utilizando o protocolo HTTP com o método GET para monitoramento através de rota dinâmica.

Com a utilização da rota dinâmica e a utilização o protocolo HTTP, é possível rastrear as ações do usuário, e qual ar condicionado selecionado para enviar as informações para a ESP8266 realizar a transmissão do sinal infravermelho correspondente, descrito no subsistema 3.1.3.3.

3.1.3.3 Subsistema 3 - Comunicação Servidor - Cliente (ESP8266)

Neste subsistema, apresentado na Figura 3.7, é feita a comunicação entre a Raspberry Pi 3 e a ESP8266 através de *socket* TCP, onde é enviada uma *flag* como mensagem de acordo com a função escolhida pelo usuário.

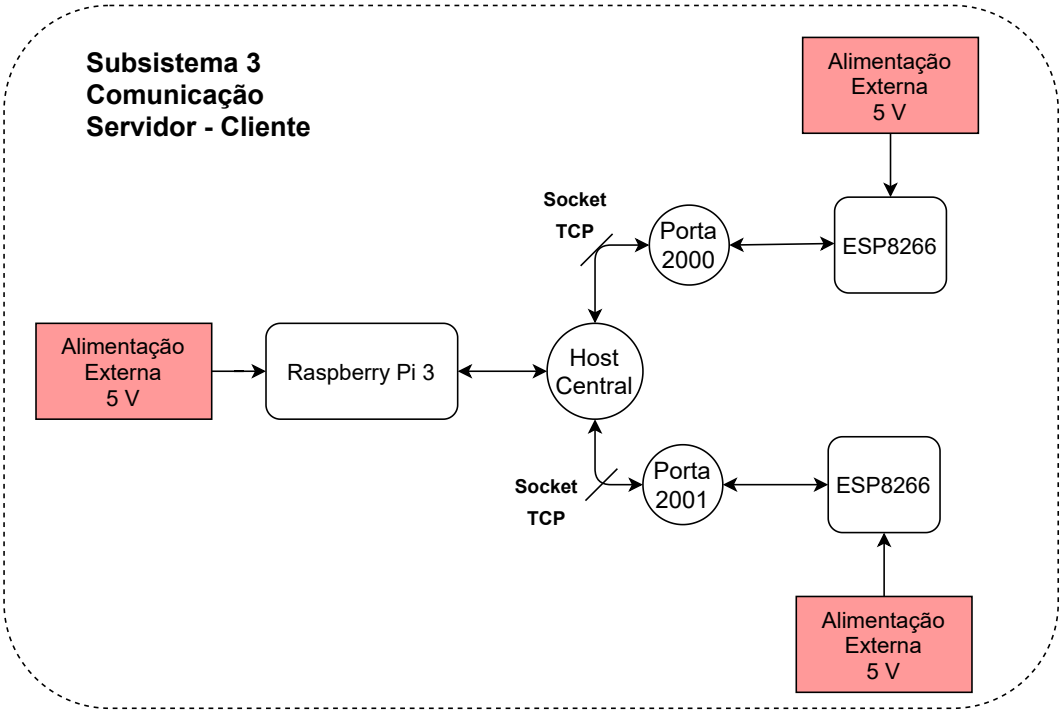

Figura 3.7. Subsistema 3 onde ocorre a comunicação entre servidor e cliente através de *socket* TCP e alimentação dos dispositivos. Fonte: Elaborada pelo autor. Software DRAW.IO.

O mapeamento das funções com as *flags* é mostrada na Tabela 3.3. As *flags* escolhidas foram definidas de forma a ser simplificada para que não ocorra erro na transmissão e na recepção.

Ar Condicionado 1	
Função	**Flag**
Ligar/Desligar	a1f1
Aumentar Temperatura	a1f2
Diminuir Temperatura	a1f3

Tabela 3.3. *Flags* enviadas do servidor Raspberry Pi 3 para o cliente ESP8266 de acordo com a função escolhida pelo usuário.

Para que o servidor tenha comunicação com vários clientes, foi definido um *host* central para que cada cliente não precisasse de uma rota diferente. Através do valor recebido por POST pelo *select*, mostrado na Tabela 3.1, a Raspberry Pi 3 consegue identificar para qual microcontrolador ela deve enviar a solicitação, pois é atribuído uma porta de comunicação diferente para cada ar condicionado. como mostrado na Tabela 3.4.

Sala	Ar Condicionado	Host	Porta
S1	Ar Condicionado 1	Host Central	2000
S1	Ar Condicionado 2	Host Central	2001
S2	Ar Condicionado 1	Host Central	2020

Tabela 3.4. Identificação das portas para comunicação entre Raspberry Pi 3 e ESP8266 com as variantes *sala*, e *arcondicionado*.

Além disto, o *host* central também evita que cada ESP8266 seja definida como IP estático, podendo nesse caso ser utilizado IP dinâmico, fazendo com que seus endereços sejam indiferentes, se importando apenas com o IP do próprio *host* central, sendo alterado apenas a porta para cada ESP8266. As setas de retorno são das informações do sensor DHT11, explanado no Subsistema 3.1.3.4

3.1.3.4 Subsistema 4 - Transmissor IR e Leitor de Estado

Neste subsistema, apresentado na Figura 3.8, a ESP8266 através da *flag* recebida no subsistema 3.1.3.3 transmite o sinal IR para o dispositivo a ser controlado. O código hexadecimal da função do controle remoto é inserido no código da ESP8266 depois de obtido através do receptor IR no subsistema 1, 3.1.3.1. Neste subsistema, também é feita a leitura da temperatura e umidade do ar condicionado através do sensor DHT11.

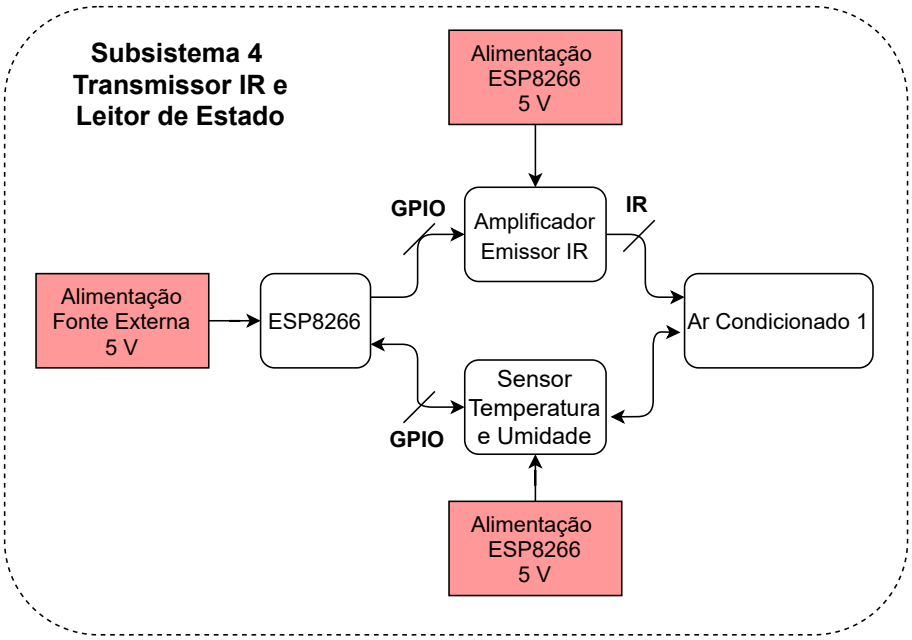

Figura 3.8. Subsistema 4 onde ocorre a amplificação e a transmissão do sinal para controle do ar condicionado, também é feita a leitura da temperatura e umidade do ar. Fonte: Elaborada pelo autor. Software DRAW.IO.

A amplificação do sinal enviado pela ESP8266 é realizada através de um transistor NPN BC547, para que o sinal alcance o dispositivo a uma distância considerável. Após a amplificação, o sinal é convertido de sinal elétrico para luminosidade através de um foto-diodo, que é apontado para o dispositivo a ser controlado. O sensor DHT11 é responsável por fazer a leitura de temperatura e umidade na boca do ar condicionado, e como a vane se movimenta para cima e para baixo, será contado um período para determinar se o aparelho foi realmente desligado ou não.

O esquemático que apresenta a conexão da ESP8266 com o sistema de amplificação e transmissão, e o sensor DHT11 é apresentado na Figura 3.9, montado através do software *fritzing*.

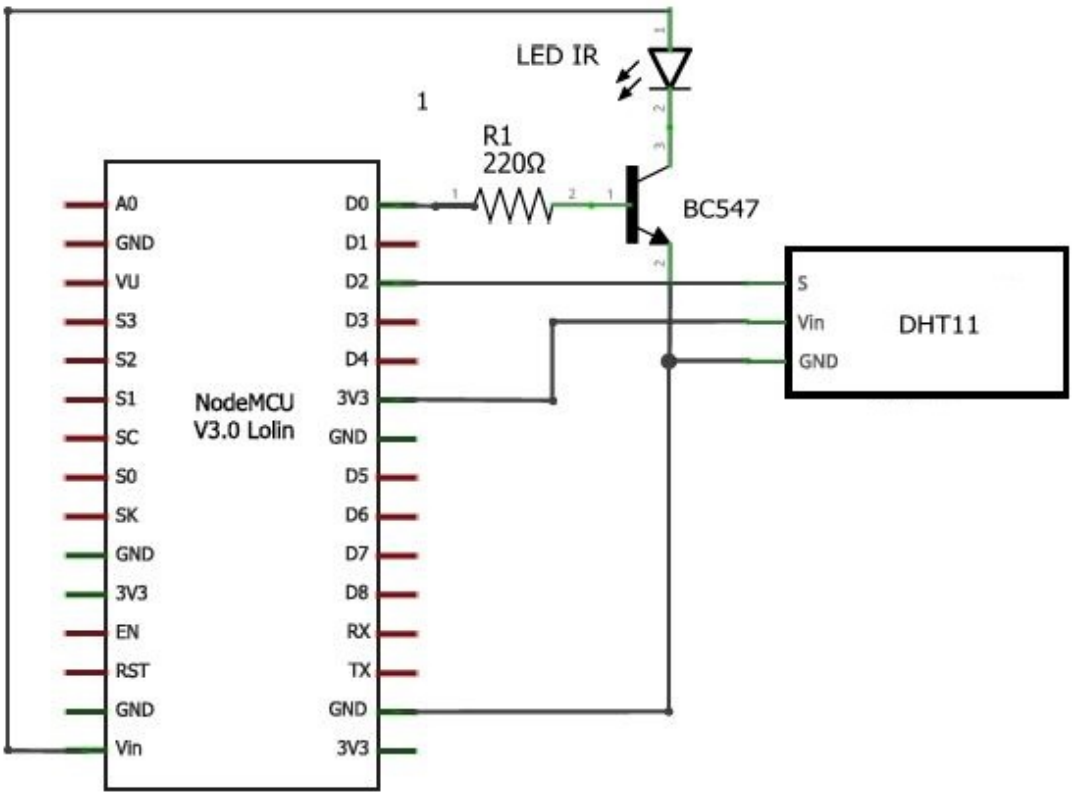

Figura 3.9. Esquemático que mostra as conexões entre a ESP8266, o sistema de amplificação e o emissor infravermelho, e o sensor de temperatura e umidade. Fonte: Elaborada pelo autor. Software FRITZING.

3.1.4 Layout

Para fazer o *layout* do circuito foi utilizado o software EAGLE. Foi adicionada a biblioteca do NodeMCU ESP8266 e os demais componentes. A Figura 3.10 mostra como ficou o *layout* e as trilhas que conectam um dispositivo ao outro.

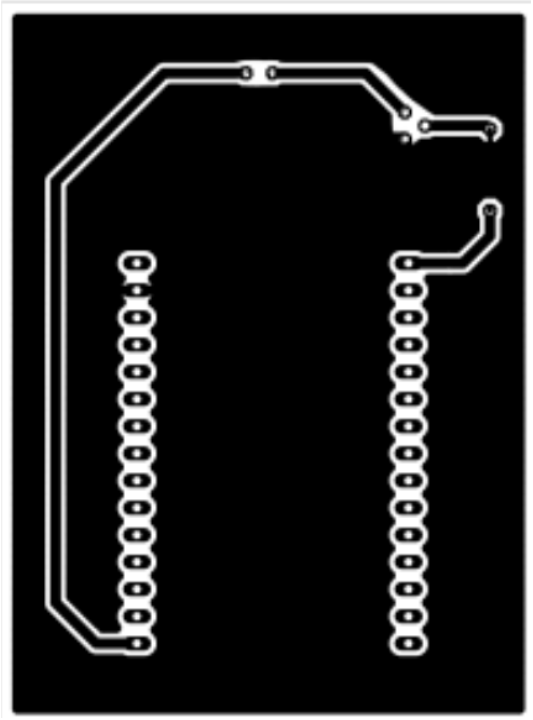

Figura 3.10. *Layout* do circuito para fabricação. Fonte: Elaborada pelo autor. Software EAGLE.

3.1.5 Fabricação e Testes

A fabricação da Placa de Circuito Impresso (PCB) será realizada de forma caseira, pelo fato de serem poucos componentes e consequentemente poucas trilhas.Também será produzida uma caixinha, mostrada na Figura 4.23, que será colada com fita dupla face 3M no ar condicionado na parte inferior para que não precise furar o dispositivo. A caixinha será impressa em uma impressora 3D, *FlashForge Finder*, projetada através do software SKETCHUP. Para comunicação com a impressora, foi utilizado o software FLASHPRINT. Suas dimensões são:

- Comprimento: 71 mm;
- Largura: 57 mm;
- Largura: 19 mm;
- Espessura: 1 mm.

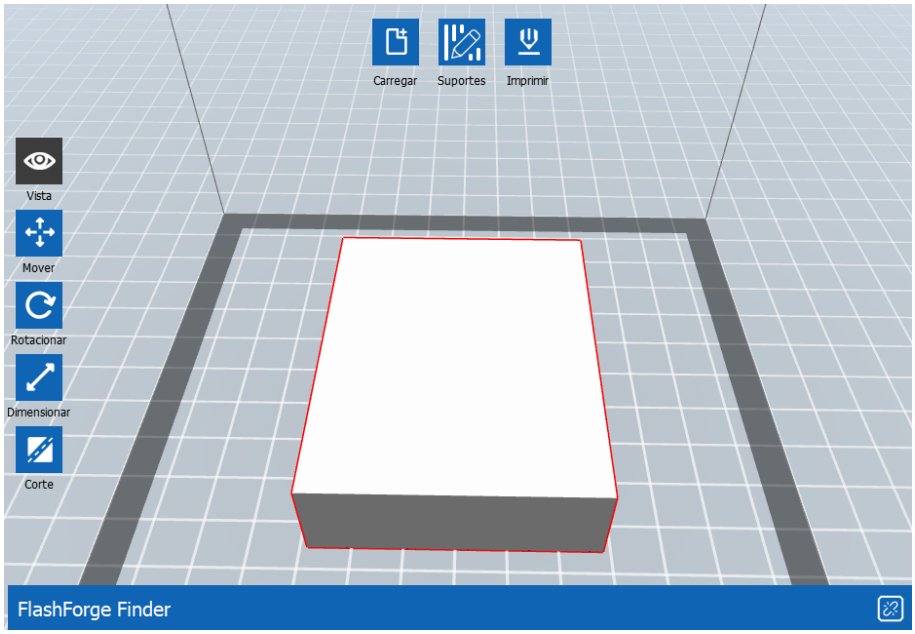

Figura 3.11. Design da caixinha utilizada para comportar a PCB. Fonte: Elaborada pelo autor. Software FLASHPRINT.

Os testes serão realizados no laboratório da empresa Delacroy Portaria Remota, local onde tenho fácil acesso a todos os materiais necessários e ao ar condicionado.

4 Resultados e Discussão

Para utilização da Raspberry Pi 3, foi utilizado o protocolo *Secure Socket Shell (SSH)* através do *software PUTTY* juntamente com a configuração de IP estático, para acesso ao terminal possibilitando a programação através de um outro computador. Os testes foram realizados em um ar condicionado da marca LG.

4.1 Subsistema 1 - Recepção IR

O controle remoto utilizado nos testes é de um ar condicionado *LG*, mostrado na Figura 4.1.

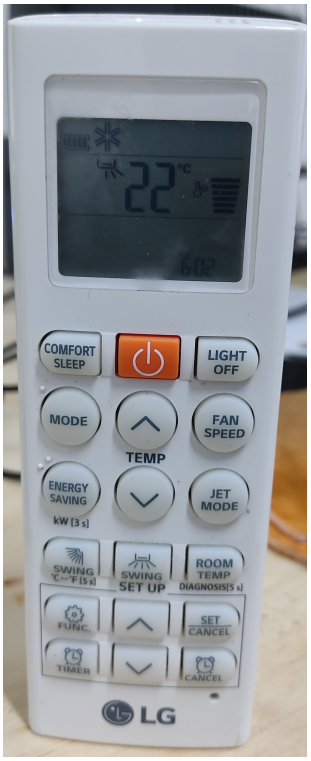

Figura 4.1. Controle *LG* de ar condicionado utilizado para validar o teste do subsistema 1.

Para recepção do sinal IR, foi utilizado o receptor infravermelho (VS1838) juntamente ao ESP8266, com sua conexão apresentada na Figura 3.5. Primeiramente foi definido o pino que faria a leitura dos dados do sensor, e ao fazer o *upload* pra placa, foram pressionados os botões de ligar, desligar, aumentar temperatura e diminuir temperatura. A Figura 4.2 apresenta o resultado obtido pelo terminal.

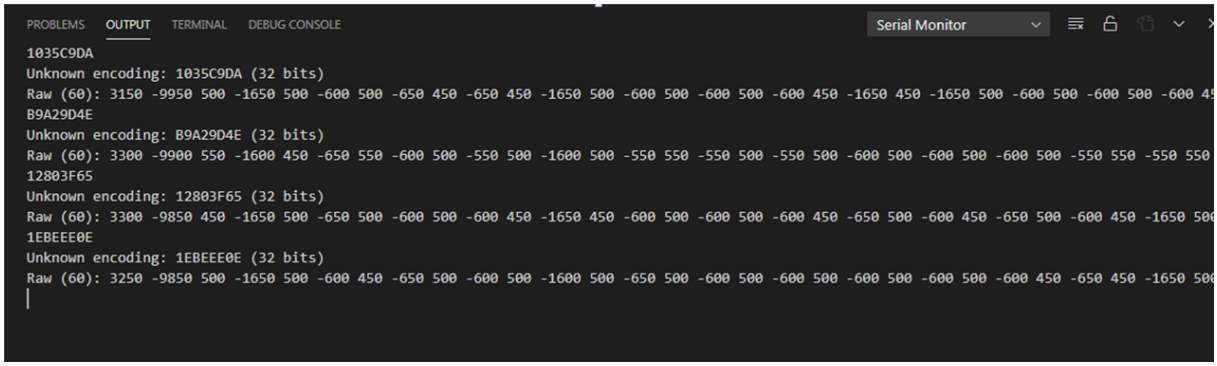

Figura 4.2. Dados decodificados do controle remoto recebido pelo receptor VS1838, no botão ligar/desligar, aumentar e diminuir temperatura.

Os dados de cada função para serem utilizados no código foram formatados no programa WORD, para que o pacote de dados não tenha traços e nem espaços, sendo cada valor separado por vírgula.

- Desligar: 3150,9950,500,1650,500,600,500,650,450,650,450,1650,500,600,500,600,500,600,450,1650,450,1650,500,600,500,600,500,600,450,650,500,600,500,600,500,600,450,650,500,600,450,650,500,600,500,1650,500,600,500,1650,450,600,500,650,450,650,500,1650,450
- Ligar: 3300,9900,550,1600,450,650,550,600,500,550,500,1600,500,550,550,550,500,550,500,600,500,600,500,600,500,550,550,550,550,600,500,550,500,600,500,600,500,1600,500,1600,500,1600,500,550,550,1600,500,550,500,600,500,1600,500,550,500,1600,500,1600,500
- Aumentar Temperatura: 3300,9850,450,1650,500,650,500,600,500,600,450,1650,450,600,500,600,500,600,450,650,500,600,450,650,500,600,450,1650,500,600,450,650,500,600,450,1650,500,600,500,600,500,600,500,600,500,1650,450,650,500,600,450,650,450,1650,500,600,500,600,500
- Diminuir Temperatura: 3250,9850,500,1650,500,600,450,650,500,600,500,1600,500,650,500,600,500,500,500,600,500,600,500,600,450,650,450,1650,500,600,450,650,450,

650,450,600,500,1650,450,1650,500,1650,500,600,500,1650,500,600,500,600,500,600,
500,600,450,1650,450,1650,450

Outra forma de descobrir o valor de cada botão de controle remoto, é pesquisar um arquivo pela internet com os dados de todos os botões, sendo possível encontrar diversos modelos de controle de todos os tipos de equipamentos.

Os dados formatados são inseridos na ESP8266 no subsistema 4.4, para controlar o equipamento.

4.2 Subsistema 2 - Servidor *Web*

Para saber sempre qual a URL do servidor, foi determinado um endereço IP estático em 192.168.0.37. O número da porta escolhida foi a porta 1000, pelo mesmo fator de não interferir com outras redes e os dispositivos que fazem conexão. Ao acessar o IP do servidor o usuário se conecta na página gerando uma requisição que retorna uma página HTML, mostrada na Figura 4.3 pelo computador, e na Figura 4.4 pelo celular, para que possa ser escolhido qual ar condicionado ele irá controlar, de acordo com seu número e sua sala correspondente.

Figura 4.3. Página visualizada pelo usuário ao acessar o servidor na Raspberry Pi 3 pelo computador. Fonte: Elaborada pelo autor.

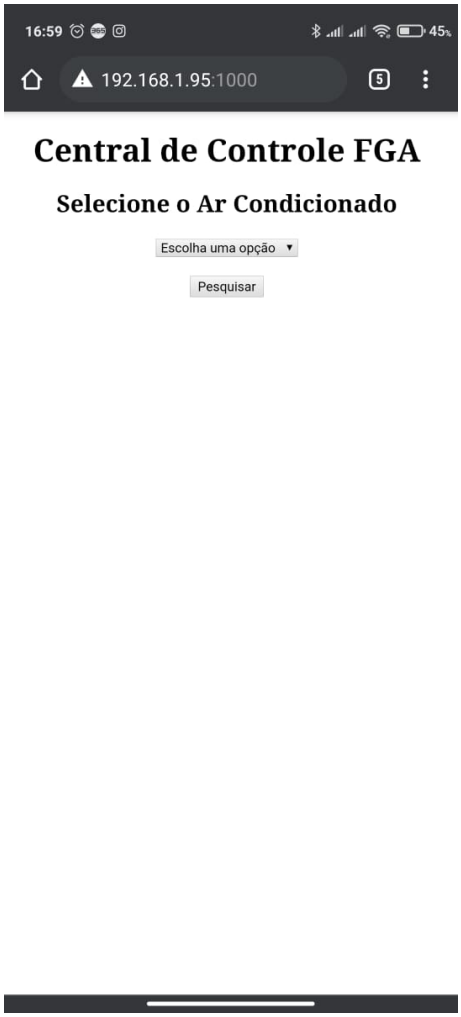

Figura 4.4. Página visualizada pelo usuário ao acessar o servidor na Raspberry Pi 3 pelo celular. Fonte: Elaborada pelo autor.

No momento em que o código em *python* está sendo executado, o servidor fica aguardando algum acesso de dispositivos. Quando um cliente acessa o servidor, aparece o IP do dispositivo no terminal, o método de requisição e o status de conexão. A Figura 4.5 mostra dados de acesso de um celular e de um computador.

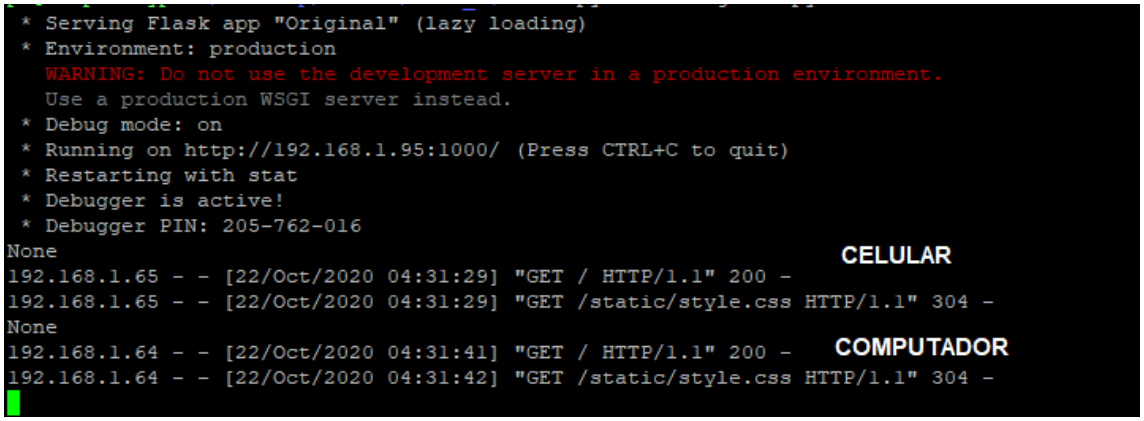

Figura 4.5. Terminal da Raspberry Pi onde se encontra o servidor e os dados de acesso de clientes. Fonte: Elaborada pelo autor.

Quando o usuário acessa a página do servidor, é possível escolher qual sala e qual ar condicionado ele quer controlar. A Figura 4.6 mostra algumas opções possíveis feitas através da *tag select*, juntamente com o elemento *optgroup*, capaz de colocar mais de uma opção em um determinado grupo.

Figura 4.6. Seleção com as opções de ar condicionados e salas para acesso remoto. Fonte: Elaborada pelo autor.

A Figura 4.7 mostra como foi montado o código em *python* do *template* HTML, mostrando os valores de cada ar condicionado, mostrado na Tabela 3.1, utilizados para cada opção que serão recebidos no *back-end*. Também é possível notar que após a escolha da opção, é necessário apertar no botão "Pesquisar", para que o valor da seleção seja enviada pelo método POST.

```
<!DOCTYPE html>
  <head>
     <title>Controle FGA</title>

     <link rel="stylesheet" href='../static/style.css'/>
  </head>
  <body>

  <center><h1> Central de Controle FGA</h1></center>
  <center><h2> Selecione o Ar Condicionado</h2></center>
  <h3></h3>
  <center>
  <form class="selecionarsala" method="POST" action="/">
  <select name="sala">
       <option value="none">Escolha uma opção</option>
       <optgroup label="Status Geral">
              <option value="status">Todas as Salas</option>
       </optgroup>
       <optgroup label="Sala S1">
              <option value="s101">Ar Condicionado 1</option>
       </optgroup>
       <optgroup label="Sala S2">
              <option value="s201">Ar Condicionado 1</option>
       </optgroup>
       <optgroup label="Sala S3">
              <option value="s301">Ar Condicionado 1</option>
       </optgroup>
       <optgroup label="Sala S4">
              <option value="s401">Ar Condicionado 1</option>
       </optgroup>
       <optgroup label="Sala S5">
              <option value="s501">Ar Condicionado 1</option>
       </optgroup>


    </select>
    <br><br>
    <button type="submit" class="btn btn-default">Pesquisar</button>
    </form>
```

Figura 4.7. Estrutura HTML para seleção de ar condicionados e envio de dados via POST. Fonte: Elaborada pelo autor.

Ao escolher o ar condicionado, o parâmetro de valor referente ao mesmo é recebido através do método POST com a função $request.from.get()$ no parâmetro *sala*, nome dado ao elemento *select* no HTML mostrado na Figura 4.7, para que a página HTML e os dados deste ar condicionado sejam selecionados, sendo este processo mostrado na Figura 4.8. Quando não há um valor passado como parâmetro, o usuário é direcionado à página principal, mostrada na Figura 4.3.

```python
from flask import Flask, render_template, request
import socket
import numpy as np
import os

HOST = '192.168.1.95'

app = Flask(__name__)

@app.route("/", methods=('GET','POST'))
def index():
        global select
        global retorno
        select = request.form.get('sala')
        print (select)
        if(select == 'none'):
                retorno = "teste3.html"
                return render_template(retorno)
        elif(select == "s101"):
                retorno = "s101.html"
                return render_template(retorno)
        elif(select == "s102"):
                retorno = "s102.html"
                return render_template(retorno)
        elif(select == 's201'):
                retorno = "s201.html"
                return render_template(retorno)
        elif(select == "s201"):
                retorno = "s301.html"
                return render_template(retorno)
        elif(select == 's401'):
                retorno = "s401.html"
                return render_template(retorno)
        elif(select == "s501"):
                retorno = "s501.html"
                return render_template(retorno)
        elif(select == 'status'):
                retorno = "status.html"
                return render_template(retorno)
```

Figura 4.8. Estrutura em *python* para recebimento do valor via POST e retorno do HTML respectivo à escolha. Fonte: Elaborada pelo autor.

A página mostrada após a escolha do ar condicionado é a do controle remoto, mostrada na Figura 4.9, padrão para todas as escolhas. A diferença visual entre elas são o número da sala e o número do ar condicionado.

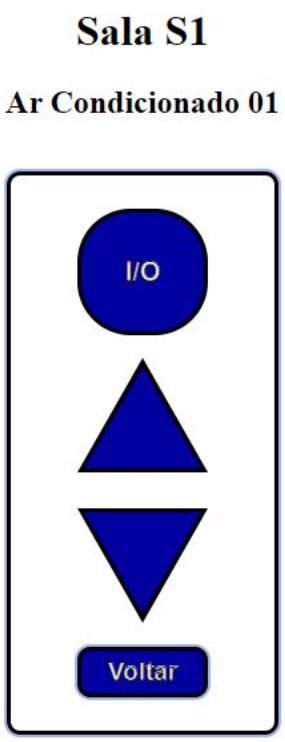

Figura 4.9. Controle remoto formado por 4 botões utilizados para envio de comandos ao respectivo microcontrolador. Fonte: Elaborada pelo autor.

O controle possui os botões ligar/desligar, aumentar temperatura, diminuir temperatura e voltar. Estes botões possuem *links* associados a eles, fazendo com que ocorra um redirecionamento de acordo com a Tabela 3.2. A Figura 4.10 mostra o arquivo HTML onde é possível observar as funções de cada botão e seu endereçamento.

```
<!DOCTYPE html>
   <head>
      <title>Sala S1</title>
      <link rel="stylesheet" href='../static/normalize.css'/>
      <link rel="stylesheet" href='../static/style.css'/>
   </head>
   <body>

        <center>

        <div class="nav">
        <h1> Sala S1 </h1>
        <h2> Ar Condicionado 01 </h2>
        <br>
        </div>
        <div class ="botoes">

        <a href="/ar1/onoff" class="button">I/O</a><br>
        <a href="/ar1/tempmais" class="buttonmais"></a>
        <br><br><br><br><br><br><br><br><br><br><br><br>
        <a href="/ar1/tempmenos" class="buttonmenos"></a>
        <br>
        <a href="/" class="buttonvoltar">Voltar</a>

        </center>
        </div>
   </body>
</html>
```

Figura 4.10. Estrutura HTML do controle remoto com endereçamento de cada botão. Fonte: Elaborada pelo autor.

Ao apertar o botão, o código em *python* busca o endereçamento para reconhecer qual foi a opção escolhida pelo usuário. Esse reconhecimento é feito através do método GET, sendo detalhada no código mostrado na Figura 4.11. Pelo *flask*, é buscado a partir da rota dinâmica na URL após o *host* sala/ação, recebendo estes valores e a partir deles determinar qual o microcontrolador correspondente e qual a ação a ser realizada. Em relação ao microcontrolador, eles se comunicarão por portas diferentes, sendo determinados após a escolha do *select* e o aperto do botão.

```
@app.route("/<sala>/<action>")
def action (deviceName, action):
        sala = select
        print (sala)
        a = 0
        if(sala == 'none'):
                PORT = 2000
        elif(sala == 's101'):
                PORT = 2000
                if (action == 'onoff'):
                        a = "alf1"
                if (action == 'tempmais'):
                        a = "alf2"
                if (action == 'tempmenos'):
                        a = "alf3"

        elif(sala == 's201'):
                PORT = 2001
                if (action == 'onoff'):
                        a = "alf1"
                if (action == 'tempmais'):
                        a = "alf2"
                if (action == 'tempmenos'):
                        a = "alf3"

        elif(sala == 's301'):
                PORT = 2003
                if (action == 'onoff'):
                        a = "alf1"
                if (action == 'tempmais'):
                        a = "alf2"
                if (action == 'tempmenos'):
                        a = "alf3"
```

Figura 4.11. Estrutura em *python* para leitura do endereço do botão com a sala e a ação. Fonte: Elaborada pelo autor.

4.3 Subsistema 3 - Comunicação Servidor - Cliente

4.3.1 Servidor

Para enviar a informação recebida pela rota dinâmica no subsistema 4.2, foi utilizado *socket* TCP para que a comunicação entre o servidor e o cliente ocorra no *host* central. Ainda observando a Figura 4.11, a porta de comunicação é definida para cada microcontrolador, para que não haja conflito entre usuários utilizando o sistema ao mesmo tempo, especificados na Tabela 3.4. O dado a ser enviado, mostrado na Tabela 3.3, depende do botão acionado, sendo este armazenado na variável a.

Para criar e configurar um *socket* é necessário definir alguns parâmetros, sendo o primeiro IPv4, descrito como *INET*, e TCP, como *STREAM*, mostrados na Figura 4.12. Também são definidos o *host* e porta para comunicação, e assim o *socket* aguarda uma requisição de um outro cliente para fazer a conexão.

```
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        print("Servidor Iniciado. Esperando pelo cliente..")
        s.bind((HOST, PORT))
        s.listen(1)
        conn, addr = s.accept()

with conn:
        print ("Conectado: ", addr)
        print ("Enviando dados")
        print (a)
        my_data = a
        x_encoded_data = my_data.encode('utf-8')
        conn.sendall(x_encoded_data)
        print (x_encoded_data)
        s.close()
        return render_template(retorno)
```

Figura 4.12. Estrutura em *python* para comunicação por *socket* TCP com o microcontrolador. Fonte: Elaborada pelo autor.

Quando a conexão com um outro cliente é estabelecida, os dados a serem enviados são codificados para o padrão utf-8, e após o envio, o *socket* é desconectado, para que ambas as partes finalizem a comunicação não interferindo na próxima comunicação. A Figura 4.13 mostra a partir do terminal do servidor como é apresentada esta conexão.

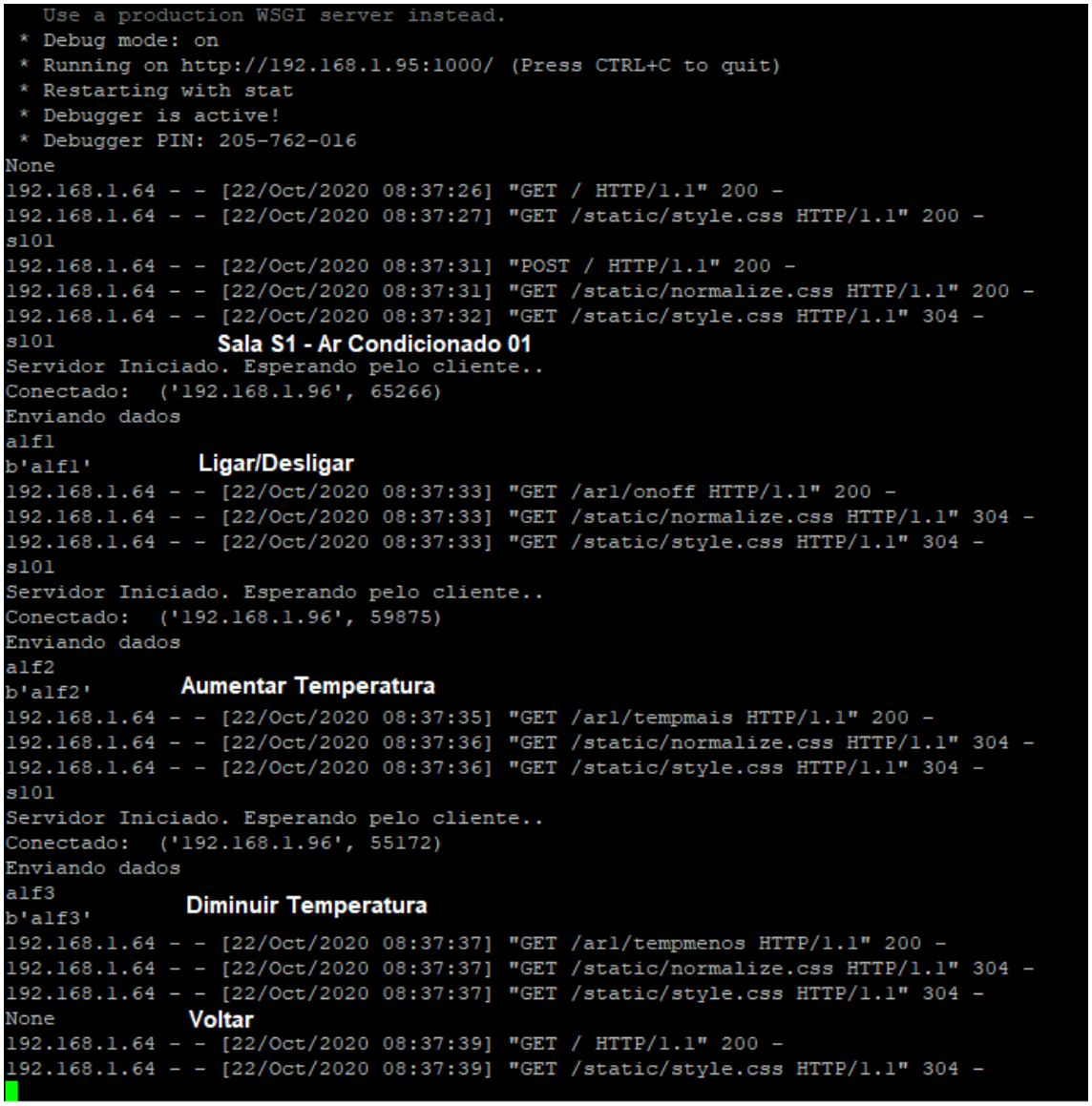

Figura 4.13. Envios realizados através do usuário na página web mostrados no terminal do servidor. Fonte: Elaborada pelo autor.

Primeiramente foi escolhida a sala S1 ar condicionado 01. Após a escolha, foram apertados os botões ligar/desligar, aumentar temperatura, diminuir temperatura e o botão voltar, respectivamente. É possível notar o código 200, apresentado na Tabela 2.10, indiciando que a conexão foi realizada com sucesso, em cada caso. Quando a conexão não é estabelecida com nenhum cliente, o servidor fica em *loop* aguardando um outro cliente para comunicação.

4.3.2 Cliente

Primeiramente o cliente acessa a rede *Wi-Fi*, recebendo o valor do seu IP como 192.168.1.96. Após a conexão na rede local, ele acessa o *host* central através do IP 192.168.1.95 e da porta definida por esse microcontrolador, 2000, mostrada na Figura 4.14, esperando por uma ação do usuário. Foi utilizado o *software VISUAL STUDIO CODE*.

```
const char* ssid     = "RODRIGO OI FIBRA 2G";
const char* password = "flamengo";

const char* host = "192.168.1.95";
const uint16_t port = 2000;
int i=0;
```

Figura 4.14. Estrutura de código do microcontrolador com acesso a rede *Wi-Fi* e endereçamento do IP do *host* e da porta para comunicação com o servidor. Fonte: Elaborada pelo autor.

Quando o usuário aperta um botão, ele recebe a *flag* do servidor que identifica a função, mostrada na Tabela 3.3, para poder fazer a transmissão IR, mostrada no próximo subsistema. Os casos da Figura 4.15 no terminal do cliente são as mensagens recebidas pelo servidor.

No terminal do cliente, mostrado na Figura 4.15, é possível também observar os dados durante a conexão e a transferência de dados, ao se apertar os botões ligar/desligar, aumentar temperatura e diminuir temperatura, respectivamente, como mostrado na Figura 4.13.

```
--- Miniterm on COM4  115200,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
..
WiFi conectado
Endereço IP:
192.168.1.96

Comunicação bem sucedida.
Dados Recebidos:
a1f1
Liga/Desliga
Encerrando Conexão.

Comunicação bem sucedida.
Dados Recebidos:
a1f2
Aumentar Temperatura
Encerrando Conexão.

Comunicação bem sucedida.
Dados Recebidos:
a1f3
Diminuir Temperatura
Encerrando Conexão.
```

Figura 4.15. Terminal do cliente contendo informações sobre a conexão e a transferência de dados com o servidor Fonte: Elaborada pelo autor.

Para que se possa ter total controle desta comunicação, é possível mapear quando a conexão é bem sucedida ou não. A Figura 4.16 apresenta a estrutura do código que mostra a espera de uma conexão até 5s, que caso não ocorra, é definido com uma falha. Caso a conexão seja feita com sucesso, o valor da *flag* é lida em um *loop*, que neste caso possui um total de quatro leituras pelo tamanho da *flag* ser de quatro letras.

```
unsigned long timeout = millis();
while (client.available() == 0) {
  if (millis() - timeout > 5000) {
    Serial.println(">>> Client Timeout !");
    client.stop();
    delay(60000);
    return;
  }
}
Serial.println("Comunicação bem sucedida.");
Serial.println("Dados Recebidos: ");
while (client.available()) {

  char flag = (client.read());
  Serial.print(flag);
  i=i+1;
```

Figura 4.16. Estrutura do código para comunicação *socket* TCP com o servidor através da porta 2000. Fonte: Elaborada pelo autor.

4.4 Subsistema 4 - Transmissor IR e Leitor de Estado

4.4.1 Transmissor IR

Como a definição do ar condicionado é feita através da porta, o único valor importante para o microcontrolador é a função a ser realizada, sendo definida na última letra para todos ar condicionados, como mostra a Tabela 3.3. Por este motivo, é feita uma comparação na última letra para definir a ação a ser tomada pelo ESP8266, mostrada na Figura 4.17.

```
char flag = (client.read());
i=i+1;

if(i==4){
    Serial.println();
    if(flag =='1'){
        a = a+1;
        if(a%2 == 0){
          irsend.sendRaw(liga,tamanho,frequencia);
           Serial.println("Liga");
        }
        else{
            irsend.sendRaw(desliga,tamanho,frequencia);
            Serial.println("Desliga");
            }
        delay(50);
    }
    else if(flag =='2'){
        Serial.print("Aumentar Temperatura");
        irsend.sendRaw(aum_volume,tamanho,frequencia);
        delay(50);
    }
    else if(flag =='3'){
        Serial.print("Diminuir Temperatura");
        irsend.sendRaw(dim_volume,tamanho,frequencia);
        delay(50);
    }
    i=0;
    }
```

Figura 4.17. Estrutura do código para detecção de função e envio de IR. Fonte: Elaborada pelo autor.

Após a definição da ação, é selecionada a variável que armazena o *buffer*, o tamanho do *buffer*, e a frequência em que opera o controle com o dispositivo, que no caso do ar condicionado é 38 KHz.

Quando o comando é enviado, o ar condicionado faz um barulho para avisar que recebeu o comando pelo IR. Neste teste foi observado que o controle remoto físico do ar condicionado não recebe informações do ar condicionado, pois ao desligar o aparelho remotamente, o controle ainda fica ligado. Outro ponto importante a ser observado, é que ao aumentar e diminuir a temperatura não se tem o retorno do valor em que a temperatura se encontra, aceitando apenas o barulho de recebimento do ar condicionado.

4.4.2 Leitor de Estado

Para detectar se o ar condicionado está ou não ligado, foi utilizado um sensor de temperatura DHT11 para obter através dos dados de temperatura e umidade o status do equipamento. Durante os testes notou-se que utilizar o sensor para determinar o estado não é um método eficaz, visto que quando o DHT fica muito gelado, ele demora bastante tempo para retornar um valor de limiar confortável para a decisão.

No teste, o sensor foi colocado diretamente na boca do ar condicionado ligado ao ESP8266, que estava conectado ao computador. Utilizando o software HYPERTERMINAL, foi gravado em um arquivo .txt os valores da porta serial, onde a partir destes dados, foi utilizado o software SCILAB para geração de gráficos. Os gráficos de temperatura e umidade recebidos pelo DHT11 são apresentados nas Figuras 4.18 e Figura 4.19.

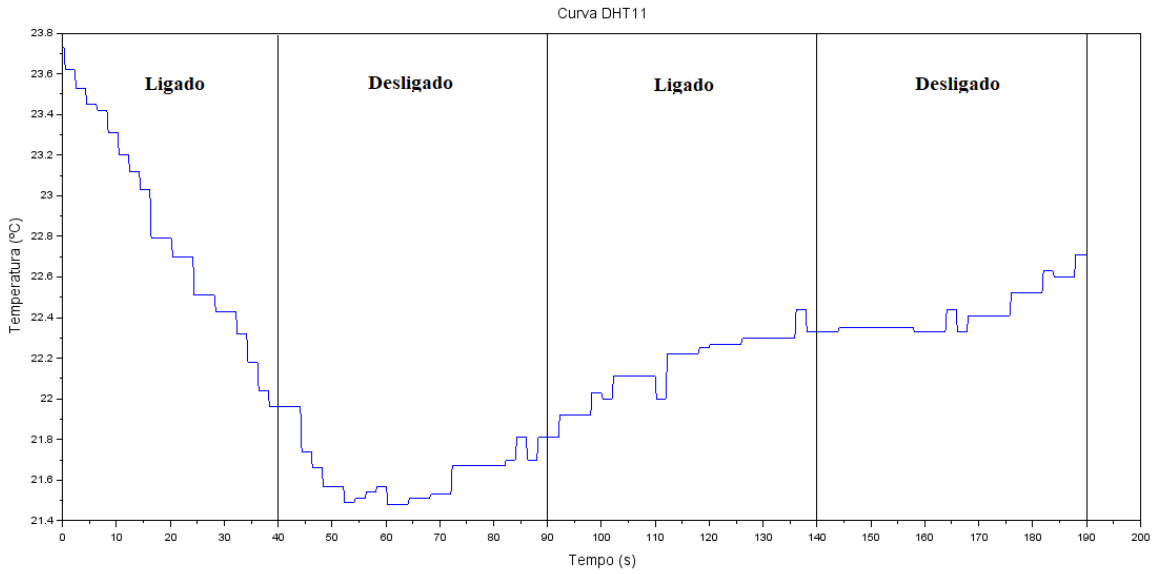

Figura 4.18. Gráfico de temperatura gerado pelo DHT11 em teste feito no ar condicionado. Fonte: Elaborada pelo autor. Software SCILAB.

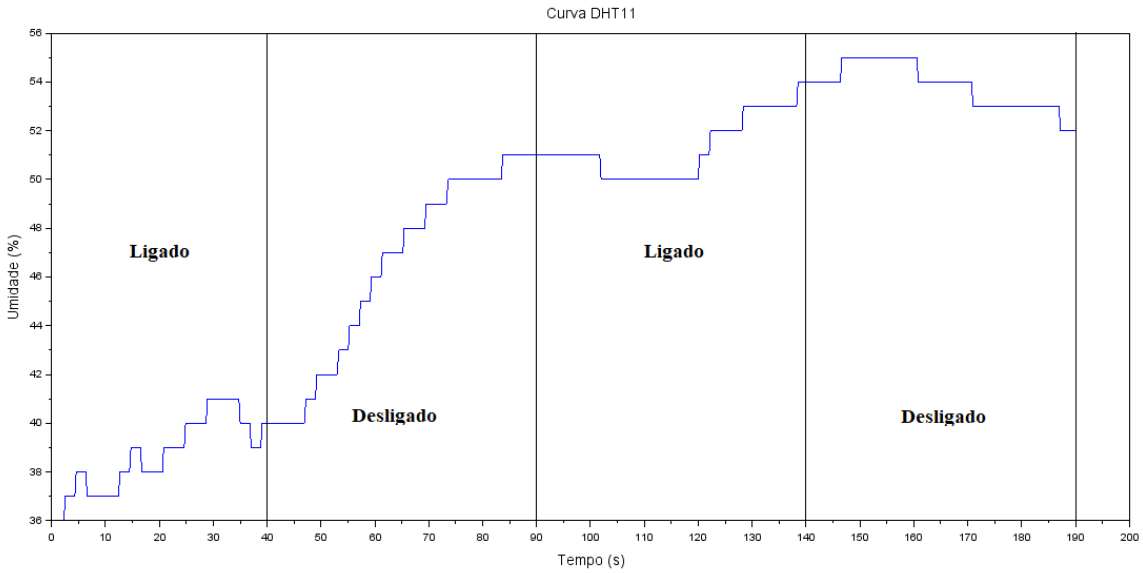

Figura 4.19. Gráfico de umidade gerado pelo DHT11 em teste feito no ar condicionado. Software SCILAB. Software SCILAB.

Nos testes feitos, a leitura ocorreu a cada 250 ms. No começo da leitura, o ar condicionado foi ligado na temperatura de 23ºC. Pode-se notar na Figura 4.18 que a temperatura começou a diminuir para bem mais abaixo do que a temperatura estabelecida, pelo fato do sensor estar na boca da saída do ar condicionado. Já a umidade, mostrada na Figura 4.19 começou a aumentar rapidamente. Após 40s, o ar condicionado foi desligado para observar a variação da temperatura e observar se seria possível chegar a uma definição se o ar condicionado estaria desligado ou não. Após 10s, a temperatura lida pelo sensor estabilizou por 20s e a temperatura começou a subir novamente. O ar condicionado foi novamente ligado aos 90s.

Neste primeiro experimento, foi possível notar que o ar condicionado demorou 20 s estabilizado e até os 90s, a temperatura se alterou menos de 1ºC. Portanto, em um período de alteração de 50s começando ao desligar até ligar novamente, a variação de temperatura é muito pequena para definir se o ar condicionado foi desligado ou se apenas a temperatura foi alterada. O processo foi repetido desligando entre 90s, ao ligar o equipamento, e ao desligar, em 140s. Neste segundo momento, mesmo sem alterar a temperatura do ar condicionado, a temperatura lida pelo sensor não abaixou mais, se mantendo constante, sendo mais difícil ainda tomar algum tipo de decisão.

Portanto, a partir das análises realizadas, foi descartada a utilização do sensor de temperatura para uma rápida resposta do status do ar condicionado. Pode ser que para uma definição que não precise ser momentânea, esses testes podem ser mais aprofundados

e o sensor tenha uma resposta melhor para este caso.

Com o descarte do DHT11, foi testado outro sensor, o piezoelétrico, mostrado na Figura 4.20. Este sensor foi utilizado para que fosse adquirido o status do ar condicionado através da vibração quando ligado, e a ausência, quando desligado. O piezoelétrico foi conectado na porta analógica A0 do ESP8266 como entrada para leitura em um *baud rate* de 115200bps. Foi utilizado um resistor de 1 MΩ em paralelo com o sensor para que o valor de tensão estivesse na escala de leitura do microcontrolador.

Figura 4.20. Piezoelétrico utilizado para detectar vibração do ar condicionado quando ligado. Fonte: Elaborada pelo autor. FONTE: [16]

Utilizando novamente o HYPERTERMINAL para leitura na porta serial, foi plotado um gráfico no SCILAB para análise dos dados obtidos, mostrado na Figura 4.21.

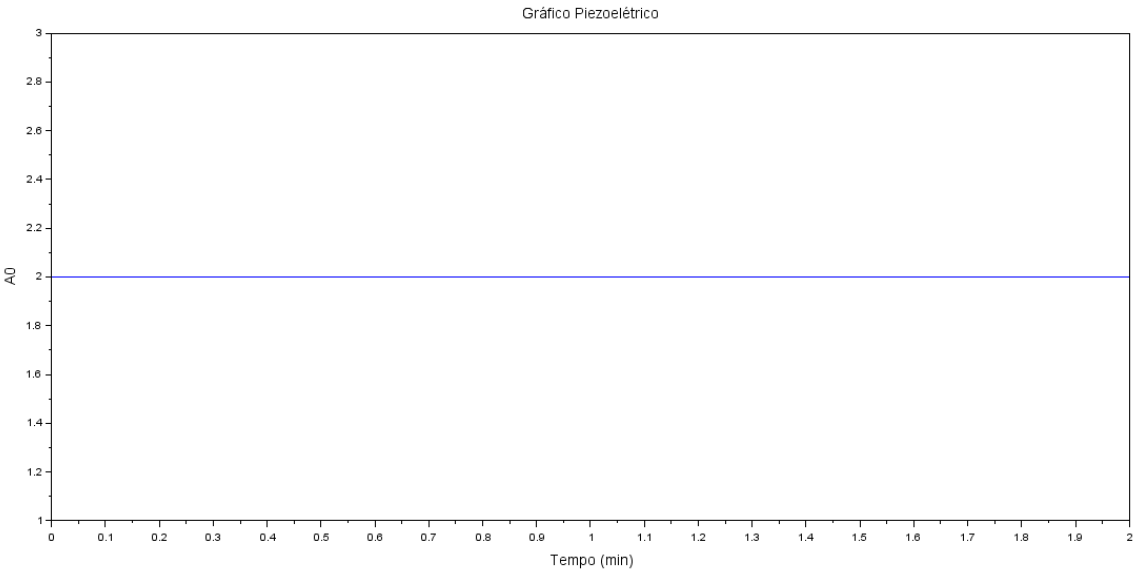

Figura 4.21. Gráfico dos valores do piezoelétrico lido pelo microcontrolador. Fonte: Elaborada pelo autor. Software SCILAB.

Analisando a Figura 4.21, pode se notar que o valor ficou constante. Como o teste foi feito em um ar condicionado inverter silencioso, com o piezoelétrico colado na parte inferior, não se notou pertubação entre ligar e desligar quando não se utilizou um período de leitura específico. Neste teste o ar condicionado foi ligado e desligado 4 vezes.

Entretanto, notou-se que ao alterar o tempo de leitura entre cada dado para um tempo em milissegundos, o comportamento do sensor variou bastante, de forma aleatória, indo de 2 até 9, independente de enviar o comando de ligar e desligar. Como não foi possível encontrar um limiar eficiente entre o estado ligado e desligado, o piezoelétrico também foi descartado.

Outro método estudado para implementação foi a utilização de um sensor de corrente não invasivo, sendo possível determinar com uma grande eficiência os estados de ligado e de *stand-by*. Entretanto, para realizar a leitura da corrente com este sensor, seria necessário utilizar apenas o fio positivo do ar condicionado, e como o equipamento vem de fábrica com um cabo grosso de energia que engloba tanto o fio positivo quanto o negativo, não seria possível fazer a leitura separada, eliminando a possibilidade de utilização do sensor de corrente.

4.4.3 Layout e Fabricação

Após fazer o *layout*, mostrado na Figura 3.10, se deu início à fabricação da PCB. Foi utilizado uma placa de cobre lisa, um furador de placa, ferro de solda, ferro de passar roupa, caneta esferográfica, percloreto de sódio e o *layout* impresso em um papel couchê.

Ao passar o ferro de passar roupa sem vapor em cima do papel na placa de cobre por volta de 8 minutos, por transferência térmica a tinta da placa no papel couchê foi transferida para o cobre. Foram feitos alguns retoques com caneta esferográfica demarcando melhor as partes em que a tinta não ficou de forma ideal para que as trilhas ficassem bem separadas. Após o retoque, a placa foi mergulhada no percloreto de sódio em torno de 30 minutos, corroendo o cobre nos locais demarcados.

Para finalizar o processo, foi realizada a solda dos componentes na placa. O resultado final da placa é mostrada na Figura 4.22.

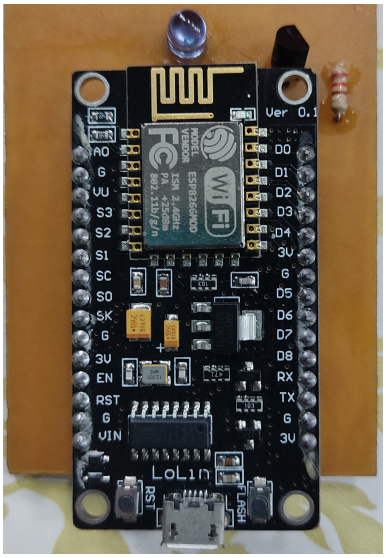

Figura 4.22. PCB produzida de forma caseira para transmissão do sinal infravermelho. Fonte: Elaborada pelo autor.

A caixinha em que a PCB é alocada foi impressa em uma impressora 3D, em uma duração de 50 minutos, gastando 18,78 gramas. O resultado final é mostrado na Figura 4.23.

Figura 4.23. Caixinha impressa na impressora 3D para ser colada com fita dupla face 3M no ar condicionado com a PCB dentro. Fonte: Elaborada pelo autor.

Todos os testes foram realizados em um laboratório com um ar condicionado *inverter* da LG, como mostra a Figura 4.24. Na imagem é possível notar a disposição da caixinha no ar condicionado. Para ficar uma instalação visualmente agradável, seria ideal ter uma tomada próxima ao ar condicionado, ou utilizar uma canaleta para que a fiação que chega na caixinha seja organizada e não fique exposta.

Figura 4.24. Ar condicionado utilizado para os testes realizados no projeto com a caixinha colada no canto direito inferior. Fonte: Elaborada pelo autor.

4.4.4 Custo

A tabela 4.1 mostra os valores gastos no projeto com equipamentos e confecção.

Equipamentos	Custo
Raspberry Pi 3	R$ 320,00
NodeMCU ESP8266	R$ 25,00
LED IR 5mm	R$ 0,81
Transistor BC547	R$ 0,18
Resistor 220 Ohms	R$ 0,05
Chapa de Cobre	R$ 2,00
Impressão Layout	R$ 4,00
Caixinha 3D	R$ 7,98

Tabela 4.1. Custos com equipamentos e materiais utilizados para desenvolvimento do projeto.

5 Conclusão

O desenvolvimento do projeto é de extrema importância por fatores de praticiade e otimização de tempo para os professores, como também por fatores sanitários em que vivemos atualmente no mundo inteiro. A utilização de um equipamento pessoal torna muito mais prática a tarefa de controlar o ar condicionado da sala em que está ocorrendo a aula sem a necessidade de buscar o controle físico na secretaria.

Ao acessar a interface *web* no IP em que se encontra o servidor, o usuário tem a visualização do controle de forma remota, com botões de liga/desliga, aumentar temperatura e diminuir temperatura, e a partir deste acesso, é possível definir qual a função que ele deseja enviar para o ar condicionado. A comunicação entre o servidor com a escolha do usuário pelo protocolo HTTP apresentou uma facilidade na forma de obter os dados escolhidos pelo mesmo, sendo este enviado para o microcontrolador cliente, utilizando-se *socket* TCP. Utilizando este método todos os comandos enviados do servidor para o cliente foram recebidos corretamente justificando a escolha do TCP, e não do UDP.

No desenvolvimento da conexão entre o infravermelho e o ar condicionado, é possível concluir que o equipamento consegue receber de forma eficaz o sinal, fazendo um barulho para avisar que a recepção foi feita com sucesso. O grande problema a ser debatido, é saber o status em que o ar condicionado se encontra, pois após as tentativas com o sensor de temperatura DHT11 e o piezoelétrico, não foi possível encontrar um limiar que garantisse que o equipamento estaria ligado ou desligado.

Para trabalhos futuros, é possível acrescentar mais dispositivos a serem controlados, facilitando ainda mais para os professores ao estarem na sala de aula, como por exemplo a televisão e o data show. Outro ponto a ter um estudo mais avançado é para se obter o retorno do estado do ar condicionado, fazendo com que tenha uma página para monitoramento possibilitando tomadas de decisão mais fáceis. Monitorar a temperatura do ar condicionado e a utilização da bateria externa são outros pontos a serem discutidos para melhor a aplicação do produto.

Referências Bibliográficas

[1] Led emissor infravermelho ir 5mm. https://www.filipeflop.com/produto/led-emissor-infravermelho-ir-5mm. (Acesso em 18/11/2020).

[2] Alexandre Fernandes de Moraes. *Redes de computadores*. 2014.

[3] Behrouz A. Forouzan. *Comunicação de Dados e Redes de Computadores*. 2007.

[4] Larry C Peterson. *Redes de computadores: Uma Abordagem de Sistemas*. 2013.

[5] Isabel; Gonzalez-Rubio Jesus Ramirez-Vasquez, Raquel; Escobar and Enrique Arribas. Comentário sobre radiação de baixa frequência e possível influência nociva a sistemas biológicos. *Revista Brasileira Ensino Física)*, 2020.

[6] Remote control. https://www.explainthatstuff.com/remotecontrol.html. (Acesso em 16/11/2020).

[7] Sinais analógicos e digitais de um sistema embarcado: Visão geral e análise dos sistemas automotivos. https://www.oficinabrasil.com.br/noticia/tecnicas/sinais-analogicos-e-digitais-de-um-sistema-embarcado-visao-geral-e-analise-dos-sistemas-automotivos. (Acesso em 16/11/2020).

[8] Vs1838 - receptor infravermelho universal. https://www.eletrodex.com.br/vs1838-receptor-infravermelho-universal.html. (Acesso em 18/11/2020).

[9] Diagrama vs1838 - receptor infravermelho universal. https://howtomechatronics.com/tutorials/arduino/control-any-electronics-with-a-tv-remote-arduino-ir-tutorial/. (Acesso em 05/04/2021).

[10] Transistor npn bc547. https://www.smartkits.com.br/transistor-npn-bc547. (Acesso em 18/11/2020).

[11] David J. Malvino, Albert. Bates. *Eletrônica*. 2011.

[12] Sensor de umidade e temperatura dht11. https://www.marinostore.com/sensores/dht11-sensor-temperatura-e-umidade. (Acesso em 07/04/2021).

[13] Raspberry pi 3b. https://cdn.shopify.com/s/files/1/0174/1800/products/Raspberry$_Pi_{3B1o}f_{11}024x1$ 1526041006. (*Acessoem*19/11/2020).

[14] Módulo esp8266 nodemcu esp-12e com wifi v3. https://www.baudaeletronica.com.br/modulo-wifi-esp8266-nodemcu-esp-12e.html. (Acesso em 19/11/2020).

[15] Esp8266 rtos sdk user manual. https://readthedocs.com/projects/espressif-esp8266-rtos-sdk/downloads/pdf/release-v3.1/. (Acesso em 19/11/2020).

[16] Transdutor piezoelétrico. https://imagineeletronica.lojavirtualnuvem.com.br/produtos/transdutor-piezoeletrico-12mm/. (Acesso em 15/04/2021).

[17] Eduardo Magrani. *A internet das coisas.* 2018.

[18] Sérgio de Oliveira. *Internet das Coisas com ESP8266, Arduino e Raspberry Pi.* 2017.

[19] Iot oferece uma nova perspectiva de negócios para o mercado de energia. https://inforchannel.com.br/iot-oferece-uma-nova-perspectiva-de-negocios-para-o-mercado-de-energia/. (Acesso em 16/11/2020).

[20] How remote controls work. https://electronics.howstuffworks.com/remote-control.htm. (Acesso em 16/11/2020).

[21] Ivan S Oliveira. *Física Moderna Para Iniciados, Interessados e Aficionados (Volume 1).* 2005.

[22] Louis E. Frenzel. *Fundamentos de Comunicação Eletrônica: Volume 1: Modulação, Demodulação e Recepção.* 2012.

[23] Inside a tv remote control. https://electronics.howstuffworks.com/inside-rc.htm. (Acesso em 18/11/2020).

[24] Claudio L V Oliveira, Cristina B M Nabarro, and Humberto A P Zanneti. *Raspberry Pi Descomplicado.* 2018.

[25] Eben Upton. *Raspberry Pi: Guia do Usuário.* 2017.

[26] Ademar Felipe Fey and Raul Ricardo Gaue. *Fundamentos De Telecomunicações E Comunicação De Dados.* 2020.

[27] Diferenças entre get e post. https://www.alura.com.br/artigos/diferencas-entre-get-e-post. (Acesso em 22/11/2020).

[28] Michael J. Donahoo. *TCP/IP Sockets in C: Practical Guide for Programmers.* 2011.

[29] Tiago Silva. *Flask de A a Z: Crie aplicações web mais completas e robustas em Python.* 2019.

[30] Miguel Grinberg. *Flask Web Development: Developing Advanced Web Applications with Python*. 2014.

[31] Elizabeth. Freeman Eric Freeman. *"Head First HTML with CSS XHTM"*. 2008.

[32] Html element reference. https://www.w3schools.com/TAgs/. (Acesso em 06/04/2021).

[33] David Powers. *"Getting Started with CSS"*. 2009.