UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE ENGENHARIA ELETRÔNICA

ALISSON DE SOUZA CASTRO

AUTOMAÇÃO RESIDENCIAL CENTRALIZADA UTILIZANDO ESP32 EM CONJUNTO COM FPGA E COMUNICAÇÃO WI-FI

TRABALHO DE CONCLUSÃO DE CURSO

CAMPO MOURÃO
2019

ALISSON DE SOUZA CASTRO

AUTOMAÇÃO RESIDENCIAL CENTRALIZADA UTILIZANDO ESP32 EM CONJUNTO COM FPGA E COMUNICAÇÃO WI-FI

Trabalho de Conclusão de Curso de Graduação, apresentado à disciplina de TCC 2, do curso Superior de Engenharia Eletrônica do Departamento Acadêmico de Eletrônica - DAELN - da Universidade Tecnológica Federal do Paraná - UTFPR, como requisito parcial para obtenção do título de Engenheiro em Eletrônica.

Orientador: Prof. Dr. Marcio Rodrigues da Cunha

CAMPO MOURÃO
2019

TERMO DE APROVAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO INTITULADO

AUTOMAÇÃO RESIDENCIAL CENTRALIZADA UTILIZANDO ESP32 EM CONJUNTO COM FPGA E COMUNICAÇÃO WI-FI

DO DISCENTE

ALISSON DE SOUZA CASTRO

Trabalho de Conclusão de Curso apresentado no dia 25 de novembro de 2019 ao Curso Superior de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná, Campus Campo Mourão. O discente foi arguido pela Comissão Examinadora composta pelos professores abaixo assinados. Após deliberação, a comissão considerou o trabalho aprovado com alterações (aprovado, aprovado com alterações ou reprovado).

Prof. Gilson Junior Schiavon
Avaliador(a) 1
UTFPR

Prof. Lucas Ricken Garcia
Avaliador(a) 2
UTFPR

Prof. Marcio Rodrigues da Cunha
Orientador
UTFPR

RESUMO

O propósito deste trabalho é desenvolver um sistema de automação residencial centralizado, utilizando o microcontrolador ESP32 em conjunto com FPGA como central de automação, e uma rede Wi-Fi para a comunicação entre os dispositivos. Foram implementadas duas plantas semelhantes para representar dois sistemas sendo controlados pela central de automação, as plantas contêm sensores de temperatura, luminosidade e presença, quanto aos atuadores foram utilizados dois reles um para controle do ventilador e outro para lâmpada, e um servomotor para simular uma cortina. O sensor de temperatura foi utilizado para controle do ventilador, já o de luminosidade para controle do servomotor e o de presença para acionar a lâmpada. O objetivo foi verificar o desempenho da FPGA trabalhando em conjunto com o ESP32 neste tipo de aplicação. Os resultados foram obtidos através de testes nos programas e painel desenvolvidos, onde o painel contem a central de automação e as plantas, foi possível notar que os sensores e atuadores tiveram um bom desempenho e que a utilização do ESP32 + FPGA não apresentou um desempenho satisfatório, onde o ESP32 sendo utilizado individualmente proporcionaria um desempenho semelhante, isso pode estar relaciona ao fato dos cálculos implementados serem relativamente simples. Foi desenvolvido uma página HTML para ser utilizada como interface com o usuário.

PALAVRAS CHAVE: ESP32. FPGA. Automação Residencial. Sensores. Atuadores. HTML. Wi-Fi.

ABSTRACT

The objective of this work is to develop a centralized home automation system, using the ESP32 microcontroller in conjunction with FPGA as the automation center and a Wi-Fi network for communication between devices. Two similar plants were implemented to use two systems controlled by the automation center, such as temperature, light and presence sensors, while the actuators were used two relays for fan control and one for pressure, and one servomotor to simulate a curtain. . The temperature sensor was used for fan control, while the brightness for servomotor control and the presence to drive a lamp. The goal was to verify the performance of FPGA working in conjunction with ESP32 in this type of application. The results were performed through tests in the programs and activated panel, where the panel contemplates the automation center and plants, it was possible to notice that the sensors and actuators performed well and the use of ESP32 + FPGA was not shown as a performance Satisfactory, where ESP32 is used to provide similar performance, this may be related to the fact that the implemented calculations are relatively simple. An HTML page was developed to be used as a user interface.

KEYWORD: ESP32. FPGA. Home automation. Sensors. Actuators. HTML. Wi-Fi.

LISTA DE FIGURAS

Figura 1 – Arquitetura Centralizada. 17
Figura 2 – Sensor PIR produzido pela Adafruit. 18
Figura 3 – Sensor LDR. 19
Figura 4 – Módulo LDR. 19
Figura 5 – Esquemático do Módulo LDR. 20
Figura 6 – Gráfico de curvas do PTC e NTC 21
Figura 7 – Módulo NTC Arduino 22
Figura 8 – Esquemático do Módulo NTC. 22
Figura 9 – Exemplo acionamento relé. 24
Figura 10 – Esquema ativação e exemplo modulo relé. 24
Figura 11 – Servomotor desmontado. 25
Figura 12 – PWM Servomotor. 25
Figura 13 – Texto entre tags e como aparece no navegador. 28
Figura 14 – Arquitetura FPGA. 29
Figura 15 – Kit Cyclone EP4CE15F17C8 29
Figura 16 – Diagrama de Blocos ESP32. 30
Figura 17 – ESP32 DevKitC. 31
Figura 18 – Diagrama de blocos projeto. 33
Figura 19 – Vista frontal painel. 34
Figura 20 – Vista lateral painel. 34
Figura 21 – Diagrama componentes no painel. 35
Figura 22 – Esquema elétrico plantas. 36
Figura 23 – Esquema elétrico central automação. 37
Figura 24 – Conteúdo atuadores e sensores. 39
Figura 25 – Implementação recebimento de dados página HTML. 41
Figura 26 – Diagrama de blocos código cliente 42
Figura 27 – Diagrama de blocos código central de automação 43
Figura 28 – Digrama de blocos código VHDL. 45
Figura 29 – Diagrama Inicial página HTML. 46
Figura 30 – Diagrama blocos ESP32 e ESP32+FPGA 47
Figura 31 – Estrutura montada. 49
Figura 32 – Painel montado com componentes. 49

Figura 33 – Valores de corrente elétrica e tensão. 50
Figura 34 – Teste HTML Mozilla e Chrome 51
Figura 35 – Desempenho código plantas. 51
Figura 36 – Teste temperatura. 52
Figura 37 – Tempo execução etapa 1 FPGA. 55
Figura 38 – Elementos lógicos FPGA. 55
Figura 39 – Página HTML aberta no smartphone. 62
Figura 40 – Página HTML. 63
Figura 41 – Cálculo coeficientes A, B e C. 99

LISTA DE QUADROS

Quadro 1 – Protocolos IEEE 802.11. 26
Quadro 2 – Relaciona os pinos do ESP32 DevKit com os da FPGA e sua função. ..37
Quadro 3 – Metodologia de ensaios ESP32 e FPGA. 47

LISTA DE TABELAS

Tabela 1 – Tempos de envio e recebimento WiFi central de automação. 53
Tabela 2 – Comparação ESP+FPGA e ESP. 54

LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

FPGA	*Field Programmable Gate Arrays* (Arranjo de Portas Programáveis em Campo)
DAELN	Departamento Acadêmico de Eletrônica
CI	Circuito Integrado
RAM	*Random Access Memory* (Memória de Acesso Aleatório)
WiFi	*Wireless Fidelity* (Fidelidade Sem Fio)
IEEE	*Institute of Electrical and Electronic Engineers* (Instituto de Engenheiros Eletricistas e Eletrônicos)
VHSIC	*Very High Speed Integrated Circuits* (Circuito Integrado de Velocidade Muito Alta)
VHDL	VHSIC *Hardware Description Language* (Linguagem de Descrição de Hardware)
IP	*Internet Protocol* (Protocolo de Internet)
HTML	*Hypertext Markup Language* (Linguagem de Marcação de Hipertexto)
PIR	*Passive Infrared* (Infravermelho Passivo)
CC	Corrente Continua
CA	Corrente Alternada
PWM	*Pulse Width Modulation* (Modulação de Largura de Pulso)
LAN	*Local Área Network* (Rede de Área Local)
TCP	*Transmission Control Protocol* (Protocolo de Controle de Transmissão)
RTT	*Round Trip Time* (Tempo de Ida e Volta)
IoT	*Internet of Things* (Internet das Coisas)
ADC	*Analog to Digital Converter* (Conversor de Analógico para Digital)
LDR	*Light Dependent Resistor* (Resistor Dependente de Luz)
PTC	*Positive Temperature Coefficient* (Coeficiente de Temperatura Positivo)
NTC	*Negative Temperature Coefficient* (Coeficiente de Temperatura Negativo
SoC	*System On Chip* (Sistema em um Chip)
GPIO	*General Purpose Input/Output* (Entradas e Saídas de Proposito Geral)

SUMÁRIO

1 INTRODUÇÃO 12
1.1 OBJETIVOS 14
1.1.1 OBJETIVO GERAL 14
1.1.2 OBJETIVOS ESPECÍFICOS 14
1.1.3 JUSTIFICATIVA 15
2 FUNDAMENTAÇÃO TEÓRICA 16
2.1 AUTOMAÇÃO RESIDENCIAL 16
2.1.1 FORMA DE CONTROLE 16
2.1.1.1 CENTRALIZADA 17
2.2 SENSORES E TRANSDUTORES 17
2.2.1 SENSOR INFRAVERMELHO 17
2.2.2 LDR 18
2.2.3 TERMISTORES 20
2.3 ATUADORES 23
2.3.1 RELÉ 23
2.3.2 SERVOMOTOR 24
2.4 LAN 26
2.4.1 IEEE 802.11 26
2.4.2 TCP/IP 27
2.5 HTML 28
2.6 FPGA 28
2.7 ESP32 30
3 METODOLOGIA 32
3.1 MONTAGEM PAINEL 33
3.1.1 ESTRUTURA 34
3.1.2 SISTEMA ELÉTRICO. 35
3.1.2.1 SISTEMA ELÉTRICO PLANTAS 35
3.1.2.2 SISTEMA ELÉTRICO CENTRAL DE AUTOMACÃO 36
3.2 CÓDIGOS DA CENTRAL DE AUTOMAÇÂO E PLANTAS 38
3.2.1 IMPLEMENTAÇÃO DA COMUNICAÇÃO WIFI 38
3.2.2 IMPLEMENTAÇÃO DO CÓDIGO CLIENTE/PLANTAS 39

3.2.3 IMPLEMENTAÇÃO DO CÓDIGO SERVIDOR/CENTRAL AUTOMAÇÃO 40
3.2.4 IMPLEMENTAÇÂO DO CÓDIGO VHDL 44
3.2.5 IMPLEMENTAÇÃO DO CÓDIGO HTML PÁGINA LOCAL.....46
3.3 METODOLOGIA DE TESTES 47
4 RESULTADOS.....49
4.1 RESULTADOS ESTRUTURA E SISTEMA ELÉTRICO 49
4.2 PÁGINA HTML 50
4.3 RESULTADO PLANTAS.....51
4.4 RESULTADOS CENTRAL DE AUTOMAÇÃO 52
5 CONCLUSÃO 56
REFERÊNCIAS.....58
APÊNDICE A – IMAGENS PÁGINA HTML. 62
APÊNDICE B – CÓDIGO CENTRAL DE AUTOMAÇÂO 64
APÊNDICE C – CÓDIGO UTILIZADO PELAS PLANTAS 81
APÊNDICE D – CÓDIGO HTML.....92
APÊNDICE E – CÓDIGO VHDL 96
ANEXO A – CÁLCULO COEFICIENTES STEINHART & HART 99

1 INTRODUÇÃO

A automação residencial é uma tecnologia relativamente nova com seu surgimento datado em 1970 nos Estados Unidos. Tendo por objetivo tornar as residências mais cômodas para seus moradores, por meio da automatização de determinadas tarefas e atividades ou proporcionar um controle mais simples para os seus usuários (MURATORI e DAL BÓ, 2011).

Com o desenvolvimento tecnológico dos últimos anos, principalmente dos computadores pessoais e da internet a automação está cada vez mais acessível às pessoas (MURATORI e DAL BÓ, 2011).

Por trás da automação residencial existem diversos dispositivos envolvidos como controladores, sensores, atuadores, barramento e interface, que trabalham de maneira centralizada ou descentralizada (ACCARDI e DODONOV, 2012; TERUEL, 2008).

Na centralizada há um único controlador que recebe e envia as informações para sensores, atuadores e interface. Descentralizada existem diversos controladores interconectados, os sensores e atuadores podem estar conectados a todos os controladores ou somente a um, ou seja, a proposta desse sistema é dividir as tarefas. Ambos utilizam de meios de comunicação que podem ser barramentos/cabos ou radiofrequência (ACCARDI e DODONOV, 2012; TERUEL, 2008).

Existem diversos dispositivos programáveis capazes de realizar o controle de uma residência sendo um desses os microcontroladores (OLIVEIRA e PETREK, 2014).

O microcontrolador basicamente é um dispositivo processador, com memória RAM, barramentos, comunicação serial e outras tecnologias presentes no mesmo microchip (OLIVEIRA e PETREK, 2014). Existem uma gama de microcontroladores com diferentes aplicações e com dispositivos internos diferentes (AURELIANO, 2017). Alguns exemplos de microcontroladores que podem ser utilizados na domótica são os 8051, ATMega328 e PIC (OLIVEIRA e PETREK, 2014).

Diferente dos dispositivos programáveis já mencionados que possuem hardware é fixo, existem dispositivos que são programados em nível de *hardware*, proporcionando assim a capacidade de ser programado o *hardware* para uma especificação particular (PEDRONI, 2010).

Sendo um desses a FPGA. As aplicações da FPGA se dão em projetos complexos e de elevado desempenho (PEDRONI, 2010).

A FPGA, utiliza de linguagem de descrição de hardware, como VHDL ou Verilog, permitindo assim que o circuito seja sintetizado e simulado antes da implementação física e também a incorporação de códigos previamente projetados e códigos IP (*Intelectual Property*) como o Nios II da Altera (PEDRONI, 2010; TOCCI, WIDMER e MOSS, 2011).

Os sensores e atuadores são os dispositivos que proporcionam a interação com o meio físico.

Os sensores reagem eletricamente a alguma alteração de energia do ambiente que pode ser luminosa, térmica, cinética, entre outras. Sendo possível assim medir grandezas como temperatura, umidade, corrente elétrica, posição, velocidade e aceleração (THOMAZINI e ALBUQUERQUE, 2010; BOLZANI, 2004).

Os atuadores são dispositivos que alteram seu estado através de comandos elétricos recebidos. Assim é possível fazer a alteração em uma variável do ambiente. Podem ser conectados diretamente ao controlador ou possuir uma pequena interface que possibilita que outros dispositivos possam atuar no mesmo. Existem diversos tipos de atuadores como Válvulas, Relés, Cilindros, Motores e etc. (THOMAZINI e ALBUQUERQUE, 2010; BOLZANI, 2004).

Dentro deste contexto, existem diversas tecnologias desenvolvidas para a comunicação entre controladores, atuadores e sensores, sendo uma dessas os protocolos de comunicação.

Os protocolos de comunicação de uso geral como o IEEE 802.11, popularmente conhecido como WiFi (*Wireless Fidelity*). O WiFi surgiu em 1997, e foi sendo aprimorado ao longo dos anos surgindo variações deste como o 802.11a e 802.11b que surgiram em 1999, 802.11g em 2003, 802.11n em 2009. Cada um destes utilizam uma tecnologia diferente da radiofrequência. A velocidade pode variar de 1 a 600 Mbps e a radiofrequência pode ser de 2,4 ou 5 GHz, dependendo da variação utilizada (TANENBAUM, 2011).

Atualmente existem microcontroladores que já possuem suporte Wi-Fi integrado ao seu sistema como o ESP32 desenvolvido pela empresa Espressif, este possui dois microprocessadores Xtensa® 32-bit LX6, podendo trabalhar com um *clock* de até 240Mhz, tem suporte para conexões Wi-Fi e Bluetooth, o que o torna uma ótima escolhe quando se trata de projetos móveis (ESPRESSIF, 2019a, 2019b).

A interface tem como finalidade fazer a interação entre o usuário e o dispositivo a ser utilizado. O interfaceamento pode ser feito de diversas formas como navegadores de internet, smartphone, painéis e controles. A interface permite ao usuário visualizar informações e interagir com o sistema (CASADOMO, 2010 apud ACCARDI e DODONOV, 2012).

1.1 OBJETIVOS

1.1.1 OBJETIVO GERAL

Implementar um sistema de automação residencial, onde sera feito a leitura da temperatura, luminosidade e presença, será automatizado o acionamento da iluminação, ventilação e cortina, onde estes serão controlados de maneira centralizada utilizando ESP32 em conjunto com a FPGA como central de controle, protocolo de comunicação IEEE 802.11 para comunicação entre plantas e central e como interface com o usuário uma página HTML na rede local. Serão desenvolvidas duas plantas semelhantes para representar dois sistemas sendo controlados.

1.1.2 OBJETIVOS ESPECÍFICOS

Aqui serão apresentados os objetivos específicos do trabalho:

- Levantar os requisitos das seguintes tecnologias: automação residencial, microcontroladores, FPGA, protocolo IEEE 802.11, sensores, atuadores e interfaceamento.
- Implementar um painel contendo, central de automação e plantas com seus devidos componentes.
- Desenvolver um programa para comunicação entre a FPGA e o módulo ESP32.
- Desenvolver um programa para ser utilizado pelas plantas com os sensores e atuadores.
- Desenvolver uma página na rede local em HTML para ser utilizado como interface para o sistema.
- Verificar a viabilidade do sistema e seu funcionamento.

1.1.3 JUSTIFICATIVA

A automação residencial é uma tecnologia relativamente nova que poucos tem conhecimento, sendo interessante estudar meios de torná-la mais acessível. Tendo em vista que esta proporciona uma maior comodidade, conforto e segurança para as pessoas que a utilizam. Portanto, neste trabalho será abordado a utilização de uma FPGA em conjunto com o microcontrolador ESP32 como central de automação, pois estes possuem uma versatilidade que possibilita a implementação de diferentes sistemas e verificar se é viável a utilização destes em conjunto para esta aplicação.

2 FUNDAMENTAÇÃO TEÓRICA

Será abordada a revisão bibliográfica dos temas que serão necessários ter conhecimento para desenvolvimento do projeto, tais como automação residencial, protocolo IEEE 802.11, sensores, transdutores, atuadores, HTML, FPGA e módulo WiFi ESP32.

2.1 AUTOMAÇÃO RESIDENCIAL

A definição básica de Automação residencial é a de união de diversas tecnologias, visando o máximo de qualidade de vida dentro da residência. No Brasil o nome Automação residencial teve origem a partir do nome americano *Home Automation***.** Isso se deve ao fato de os primeiros sistemas de automação terem sua origem nos Estados Unidos. Já na Europa a Automação residencial é mais conhecida como Domótica, resultado da união da palavra *Domus* (casa) com Robótica (controle automatizado) (MURATORI e DAL BÓ, 2014; ROVERI, 2012).

Em meados da década de 2000 a domótica era tida como um luxo. No entanto, isso está mudando pouco a pouco, principalmente devido ao avanço da tecnologia de um modo geral. A automação residencial está cada vez mais presente nas residências. Hoje é comum encontrar residências com o portão automatizado ou com sistema de monitoramento via câmeras (MURATORI e DAL BÓ, 2014).

O objetivo da automação residencial é tornar o dia a dia melhor em sua residência, aumentando o conforto, segurança e também a sustentabilidade. Dentro desses tópicos podemos citar alguns itens que podem ser automatizados como, áudio, imagem (TV e projetores), climatização, alarmes, câmeras de segurança, travas, fechaduras eletrônicas, iluminação, aquecimento, entre outros. (MURATORI e DAL BÓ, 2014; ROVERI, 2012).

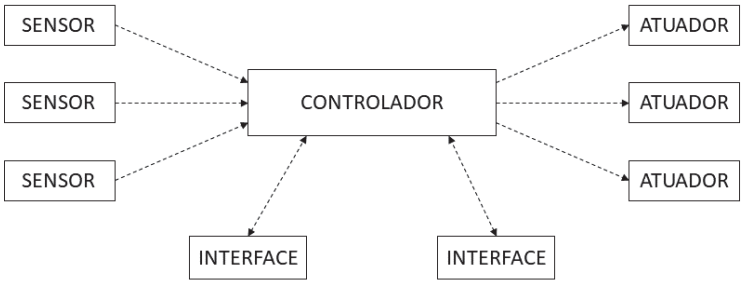
2.1.1 FORMA DE CONTROLE

A topologia de controle utilizada é a centralizada.

2.1.1.1 CENTRALIZADA

Todos os dispositivos automatizados estão conectados a um controlador, que recebe os dados processando-os conforme foi programado e enviando os comandos. Basicamente, o controlador faz a ligação entre sensor, atuador e interface, conforme observado na topologia ilustrada na Figura 1 (ALMEIDA, 2009 apud ACCARDI e DODONOV, 2012).

Figura 1 – Arquitetura Centralizada.



Fonte: CASADOMO, 2010 apud ACARDI, DODONOV, 2012.

2.2 SENSORES E TRANSDUTORES

Os sensores são capazes de detectar diversos tipos de energia do ambiente, tais como luminosa, térmica e cinética. Já os transdutores são dispositivos completos, ou seja, possuem sensores e circuitos internos, portanto, sendo capazes de medir grandezas físicas e com isso gerar um sinal elétrico proporcional (THOMAZINI e ALBUQUERQUE, 2010; PAZOS, 2002).

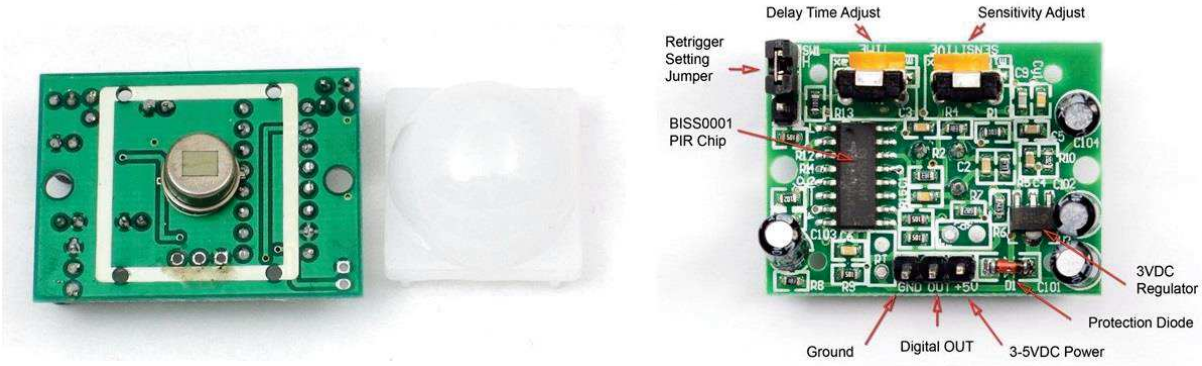
2.2.1 SENSOR INFRAVERMELHO

Os sensores infravermelhos basicamente se dividem em dois tipos, ativo e passivo. Ambos são amplamente utilizados em sistemas de segurança e iluminação automática, pois têm a capacidade de detectar a presença de pessoas (THOMAZINI e ALBUQUERQUE, 2010). Na automação residencial é mais comum a utilização de sensores passivos.

O sensor passivo trabalha apenas como receptor, pois este tem a capacidade de detectar o calor humano a distância. O elemento que torna isso possível é o sensor pirotérmico (termopilha) (THOMAZINI e ALBUQUERQUE, 2010).

Os sensores passivos são comumente chamados de PIR (*Passive Infrared*), a empresa Adafruit®, desenvolveu um modelo de sensor, pequeno, de baixa potência e custo. Os dois elementos principais, que possibilitam o funcionamento são o sensor pirotérmico e o CI BISS0001, que trabalham em conjunto com uma gama de resistores e capacitores, na Figura 2 é possível observar o sensor PIR da Adafruit (ADAFRUIT INDUSTRIES, 2014).

Figura 2 – Sensor PIR produzido pela Adafruit.



Fonte: Adaptado de ADAFRUIT INDUSTRIES.

Esse modelo de PIR, tem a possibilidade de ajuste de tempo de delay, sensibilidade, e configuração de trigger, tornando assim fácil o seu uso em projetos básicos que necessitam verificar a presença humana (ADAFRUIT INDUSTRIES, 2014).

2.2.2 LDR

O LDR (*Light Dependent Resistor*) ou fotorresistor é um sensor de luz, utilizado em inúmeras aplicações. A construção desse se dá a partir do sulfeto de cádmio, que altera sua resistência conforme o nível de intensidade luminosa. Em ambientes escuros a sua resistência é extremante elevada na casa de milhões de ohms. Já em exposição direta a luz, esta resistência cai para alguns milhares de ohms (THOMAZINI e ALBUQUERQUE, 2010).

A superfície do sensor é composta de sulfato de cádmio onde há pequenas trilhas de um material condutor, conseguindo assim uma maior capacidade de corrente e maior sensibilidade. Normalmente, o encapsulamento se dá com um material transparente. O LDR não é um componente polarizado, portando pode haver

fluxo de corrente nos dois sentidos dos terminais do sensor (THOMAZINI e ALBUQUERQUE, 2010).

Se comparado com outros sensores de luz, como fotodiodos ou fototransistores, o LDR tem um tempo de resposta mais lento, isso ocorre devido ao fato do LDR ter um tempo de recuperação muito longo, ou seja, quando o LDR está no escuro e operando com resistência máxima e uma luz é acesa a sua resistência diminui rapidamente, e em seguida a luz é ligada novamente, o tempo que o LDR demora para voltar a resistência máxima é bem mais lento . (THOMAZINI e ALBUQUERQUE, 2010). Na Figura 3 é possível observar um LDR.

Figura 3 – Sensor LDR.

Fonte: ROSSI, 2018.

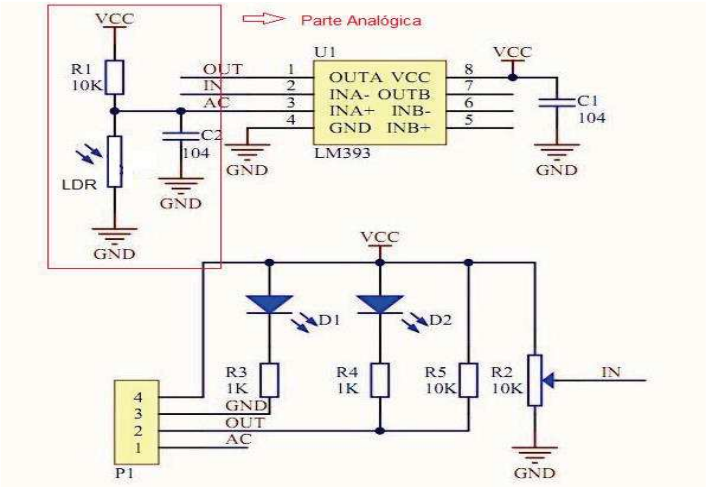
Para utilizar o LDR é necessário, um resistor trabalhando em conjunto, a união deste dois forma um divisor de tensão, onde a entrada é em luminosidade e a saída em tensão. Há dispositivos que fazem o circuito para utilização do LDR, como o módulo LDR para Arduino, este possui 4 pinos, sendo eles saída analógica (AC), saída digital (D0), alimentação (VCC) e o terra (GND) é possível observar este na Figura 4 (ROSSI, 2018).

Figura 4 – Módulo LDR.

Fonte: ROSSI, 2018.

O módulo, possui uma parte analógica e uma parte digital, a parte digital contém um comparador LM393 e um potenciômetro para regular a tensão de referência, a parte analógica contém um divisor de tensão, entre o resistor R1 e o sensor LDR (ROSSI, 2018). A tensão aplicada ao LDR é utilizada pelo comparador, ou pode ser feita sua leitura diretamente através um dispositivo próprio captação de valores de tensão elétrica. A Figura 5 mostra o esquemático deste módulo, é importante ressaltar que esse circuito pode ser utilizado por outros modelos de sensores que trabalham com a variação da resistência.

Figura 5 – Esquemático do Módulo LDR.



Fonte: Adaptado de SHOP1817254STORE, 2019.

A saída do módulo é em tensão elétrica como já foi dito, esta pode ser obtida através da equação 1.

$$AC = \frac{Rldr \times VCC}{Rldr + R1} \qquad (1)$$

A equação 1 é um divisor de tensão entre R1 e Rldr, onde AC é a saída do módulo em tensão elétrica, Rldr é o valor da resistência do LDR, R1 o valor da resistência em serie com o LDR e VCC a tensão utilizada para alimentar o módulo.

2.2.3 TERMISTORES

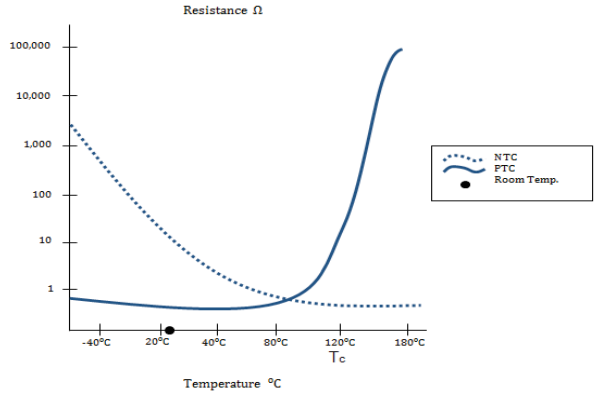
Estes sensores são semicondutores que a resistência varia com a temperatura. Os termistores possuem uma extrema sensibilidade a mudanças de temperatura. Os

elementos resistivos podem ser de óxidos de metais, como manganês, níquel, cobre, cobalto, titânio e ferro. Existem duas variações básicas de termistores, o PTC (*Positive Temperature Coefficient*) e o NTC (*Negative Temperature Coefficient*) (THOMAZINI e ALBUQUERQUE, 2010).

O PTC apresenta coeficiente térmico positivo, ou seja, a resistência aumenta com a temperatura, este apresenta coeficiente de temperatura positivo apenas dentro de uma faixa de temperatura, fora dessa limitação, o coeficiente é negativo ou nulo é possível observar este comportamento na Figura 6. (THOMAZINI e ALBUQUERQUE, 2010).

Já o NTC possui coeficiente negativo de temperatura, portando a sua resistência diminui conforme aumenta a temperatura. Este é mais utilizado em medição e controle de temperatura, porém em processos industriais este não é muito usado, provavelmente por não haver uma padronização dos fabricantes. Entre os sensores de temperatura, o NTC é um dos que fornece maior variação da saída por variação de temperatura, mas sua relação não é linear (THOMAZINI e ALBUQUERQUE, 2010).

Figura 6 – Gráfico de curvas do PTC e NTC



Fonte: STROSKI, 2017.

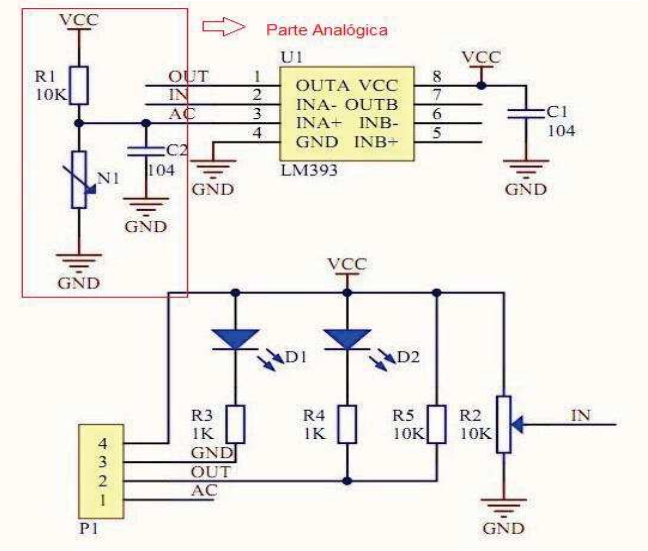
Semelhante ao LDR, o NTC precisa de um circuito auxiliar com um resistor para sua utilização, é possível encontrar módulos para Arduino iguais ao do LDR, trocando somente o sensor é possível observar isso nas Figuras 4 e 7. Observando o esquemático da Figura 8, podemos ver a parte analógica grifada em vermelho que é o que nos interessa.

É possível obter o valor de temperatura fornecido pelo NTC, utilizando a equação de Steinhart & Hart, onde esta relaciona, a temperatura com a resistência do NTC (THOMAZINI e ALBUQUERQUER,2010).

Figura 7 – Módulo NTC Arduino

Fonte: ROSSI, 2018.

Figura 8 – Esquemático do Módulo NTC.



Fonte: Adaptado de SHOP1817254STORE, 2019.

$$T = \frac{1}{A + B\ln(R) + C[\ln(R)]^3} \qquad (2)$$

Onde T é o valor da temperatura lida pelo sensor em Kelvins, e o R é o valor da resistência do NTC chamado de N1 na Figura 8, já A, B e C, são constantes que podem ser obtidas através do NTC, realizando testes no termistores ou através do *datasheet* (ROSSI, 2018). O Anexo A contém as fórmulas necessárias para obter as constantes A, B e C.

É importante notar que a saída AC do módulo Arduino acontece em tensão elétrica, portanto é necessário obter a resistência do NTC. A equação 1 pode ser

aplicada aqui, isolando a resistência no divisor de tensão entre o resistor R1 e Rn1, é possível obter a equação 2 (ROSSI, 2018).

$$Rn1 = \frac{R1 \times AC}{VCC - AC} \qquad (3)$$

Sendo Rn1 a resistência do NTC, R1 a resistência do resistor em serie com o NTC, AC a saída do modulo NTC que é em tensão elétrica e VCC a tensão elétrica utilizada para alimentar o modulo.

Utilizando o valor de Rn1 obtido e aplicando na equação 2 no lugar de R, é possível obter o valor de temperatura do ambiente em Kelvins.

2.3 ATUADORES

Os atuadores são capazes de alterar uma variável controlada, a partir de um sinal recebido de um controlador, agindo assim no sistema a ser controlado. Normalmente, estes trabalham com potência elevada (THOMAZINI e ALBUQUERQUE, 2010).

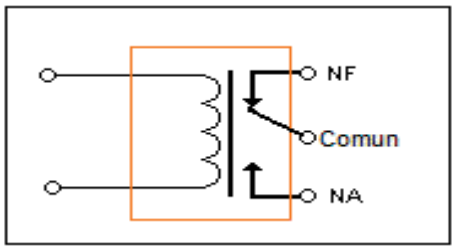
2.3.1 RELÉ

O relé ou *relay* é um dispositivo, eletromecânico que permite acionar circuitos de corrente elevada, alternada ou contínua, a partir de um sinal elétrico de baixa corrente. É muito utilizado em projetos de automação residencial, tendo em vista que este permite o acionamento de dispositivos de corrente alternada como lâmpadas e ventiladores etc (ATHOS ELECTRONICS, 2019; OLIVEIRA, 2019).

O relé normalmente possui cinco entradas, sendo duas para o sinal de acionamento e três para a carga externa a ser ativada, os pinos da carga externa são constituídos de NF (Normalmente fechado), NA (Normalmente Aberto) e comum, o funcionamento deste acontece da seguinte maneira, o pino comum é onde está o contato central, esse se mantém conectado ao contato NF até que seja aplicado uma tensão de acionamento, quando isso acontece a bobina interna é energizada formando um campo eletromagnético que se transforma em um imã, atraindo o

contato central para o contato NA é possível observar este comportamento na Figura 9 (ATHOS ELECTRONICS, 2019) .

Figura 9 – Exemplo acionamento relé.

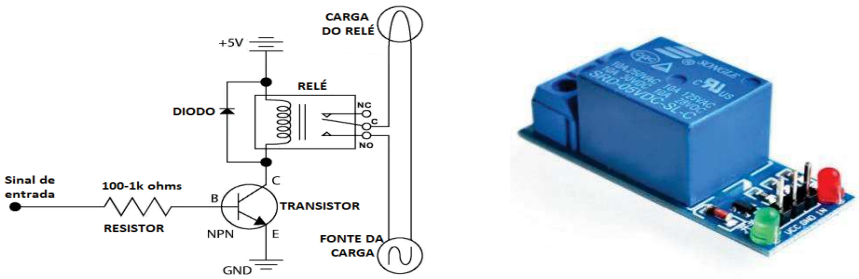


Fonte: Adaptado de ATHOS ELECTRONICS, 2019.

`

Semelhante aos sensores LDR e NTC, existem módulos prontos para reles, tendo em vista que é necessário um circuito de proteção para ativação da bobina, este circuito também permite o acionamento da bobina utilizando microcontroladores, a Figura 10, mostra um exemplo de circuito utilizado para ativação de um relé e um modulo relé. É possível observar a presença de um diodo em paralelo com a bobina, isso é utilizado pelo fato da bobina depois que é energizada, gera uma corrente reversa a qual sem a utilização desse diodo pode danificar o circuito de ativação (ATHOS ELECTRONICS, 2019).

Figura 10 – Esquema ativação e exemplo modulo relé.



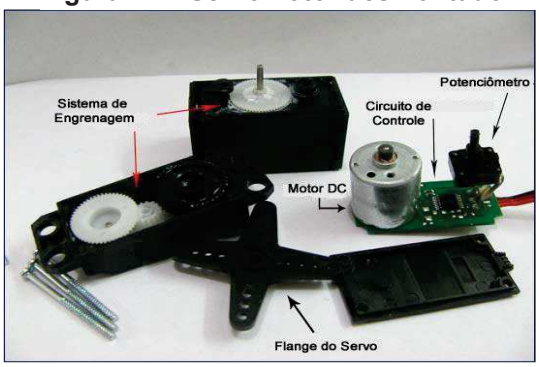
Fonte: ATHOS ELETRONICS, 2019; OLIVEIRA, 2019.

2.3.2 SERVOMOTOR

O servomotor também conhecido como servo, basicamente é um atuador eletromecânico, que mantém uma posição fixa a partir de um sinal de entrada, a sua rotação pode ser limitada ou não dependendo da sua construção. Amplamente utilizado na robótica e automação industrial, este pode ser tanto de corrente contínua ou alternada (SILVEIRA, 2019; MOTA, 2017).

Quando se fala no servo de corrente contínua e sua construção, é a junção de um motor CC (Corrente continua), engrenagens, dispositivo de detecção de posição e um circuito de controle, o servomotor possui três fios, alimentação, terra e sinal, sendo esse último utilizado para controle de posição. O seu funcionamento acontece da seguinte maneira o dispositivo de detecção de posição é um potenciômetro, onde esse produz uma tensão relativa a posição atual que é enviada para o circuito de controle, sendo assim comparada com o sinal de entrada recebido externamente, consequentemente, quando esses dois sinais são iguais o motor CC mantem a posição, quando esses diferem, o motor rotaciona proporcionalmente a diferença destes alterando assim sua posição, a Figura 11 mostra um servomotor desmontado (SILVEIRA, 2019).

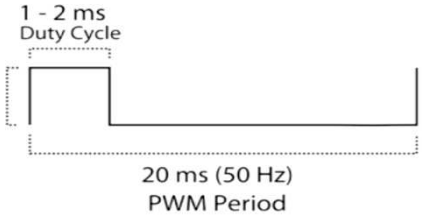
Figura 11 – Servomotor desmontado.



Fonte: SILVEIRA, 2019.

Um exemplo de servomotor é o MG90S, onde este é pequeno e leve, utilizado para projetos de modelismo ou de automação que necessitam movimentos em 180°, onde este trabalha com uma tensão de 4,8V a 6V, produzindo um torque proporcional a tensão de entrada, respectivamente sendo 1.8kg/cm a 2.2kg/cm, este trabalha com um sinal de entrada em PWM (*Pulse Width Modulation*) com período de 20ms e com um *dutycicle* entre 5% a 10%, sendo 5%(1ms) a posição -90° e 10%(2ms) a posição +90°, a Figura 12 exemplifica o descrito (COMPONENTS101, 2019).

Figura 12 – PWM



Fonte: COMPONENTS101, 2019.

2.4 LAN

A LAN (*Local Área Network*), também chamada de rede local, é um conjunto de dispositivos, conectados em uma mesma rede contidos em um único edifico pequeno que pode ir até um campus universitário com uma ampla área. Permitindo assim a troca de informação e compartilhamento de recursos. O tamanho limitado da LAN, permite a implementação de determinados projetos, além de tornar o gerenciamento de rede mais simples. A tecnologia mais utilizada para transmissão é a por cabos, mas as redes sem fios têm se tornado cada vez mais comuns, como o protocolo IEEE 802.11 (TANENBAUM, 2011).

2.4.1 IEEE 802.11

Existe uma variedade de tecnologias disponíveis para se montar uma rede sem fio, mas o padrão IEEE 802.11 é o mais utilizado, que popularmente é conhecido como Wi-Fi. Este padrão é usado para a criação de redes locais sem fio. Sua transmissão de dados é feita a partir de ondas de radiofrequência. A velocidade de transferência e o alcance dependem do protocolo utilizado. O Quadro 1 mostra detalhadamente estes protocolos, elencando a taxa de transferência máxima, frequência utilizada, modo de transmissão e o ano de lançamento oficial (TORRES, 2014).

Quadro 1 – Protocolos IEEE 802.11.

Protocolo	Taxa de transferências máxima	Frequência	Modo de Transmissão	Ano Lançamento Oficial
802.11-1997	2 Mbps	2,4 GHz	FHSS ou DSSS	1997
802.11b	11 Mbps	2,4 GHz	DSSS	1999
802.11a	54 Mbps	5 GHz	OFDM	1999
802.11g	54 Mbps	2,4 GHz	OFDM	2003
802.11n	600 Mbps	2,4 ou 5 GHz	MIMO-OFDM	2009
802.11ac	7 Gbps	5 GHz	MIMO-OFDM	2014
802.11ad	7 Gbps	60 GHz	MIMO-OFDM	--

Fonte: Adaptado de TORRES, 2014.

2.4.2 TCP/IP

O nome TCP/IP vem da junção de TCP (*Transmission Control Protocol*) e IP (*Internet Protocol*). Com a popularização da internet se tornou o protocolo mais utilizado em redes locais, tenso em visto que esse foi criado para ser utilizado na internet. Uma das vantagens desse protocolo, é que ele é roteável, ou seja, foi criado para grandes redes e para ser utilizado a uma longa distância, podendo haver vários caminhos para a informação atingir o receptor. Outra vantagem é o fato deste possuir arquitetura aberta e onde qualquer um pode utilizar. Todos os fabricantes de sistemas operacionais acabaram utilizando o TCP/IP, tornando o um protocolo universal, assim tornou-se possível que todos os sistemas possam se comunicar entre si (TORRES, 2014).

O protocolo TCP/IP, utiliza um sistema de endereçamento lógico chamado endereçamento IP, portanto cada dispositivo conectado a rede possui um endereço IP, onde essa possibilita identificar o dispositivo e a rede que este se localiza. Os dados enviados pela rede, podem ser entregues facilmente, pelo fato de cada um deste possui consigo o endereço IP do dispositivo de destino e a rede onde este se localiza. Este é o princípio básico de funcionamento de uma rede TCP/IP (TORRES, 2014).

O endereço IP, possui 32 bits, sendo formado por quatro conjuntos de 8 bits, separados por um ponto. Portanto o menor número endereço é 0.0.0.0 e o maior 255.255.255.255 (TORRES, 2014).

Já o TCP, é responsável por organizar os dados recebidos e verificar se chegaram corretamente, este tambem adiciona as portas de origem e destino a informação no envio e no recebimento identifica a aplicação que irá receber a informação através da porta atribuída a este no envio. Resumindo o protocolo IP, adiciona o endereço IP e o protocolo TCP adiciona as portas de origem e destino, tambem organiza os dados recebidos. O protocolo TCP também envia uma mensagem de confirmação de recebimento chamada de *acknowledge* a quem transmitiu a informação, caso o dispositivo de transmissão não receba essa mensagem em um determinado tempo, chamado de RTT (*Round Trip Time*), ele reenvia os dados. O TCP tambem é responsável por manter ou fechar uma conexão entre dois dispositivos em redes diferentes (TORRES, 2014).

O protocolo TCP, também permite a utilização de *socket* em cada porta. Isso permite várias conexões na mesma porta, um exemplo disso é quando se tem aberto dois navegadores no mesmo computador, através do socket atribuído a cada navegador, a informação chega ao navegador de destino pois esta tambem possui a informação de socket atribuída pelo transmissor (TORRES, 2014).

2.5 HTML

O HTML (*Hypertext Markup Language*) foi desenvolvido inicialmente por Tim Bernes-Lee é a principal linguagem utilizada na *World Wide Web* tambem conhecida como WWW, inicialmente o HTML foi projetado para descrever documentos científicos, no entanto seu design geral permitiu que fosse adaptado para ser utilizada por outros tipos de documentos e aplicativos, o HTML tem como objetivo ser uma linguagem universal, que seja entendida pelos diversos meios de acesso (WHATWG, 2019).

O conceito do HTML é a marcação. Os elementos são marcados para mostrar quais informações são exibidas na página e como são exibidas. Exemplo disso é o título de um artigo ou manchete de um site, o texto é marcado utilizando a tag chamada H1, a Figura 13 mostra como o texto é descrito e como é exibido no navegador. É possível observar que o texto se encontra entre as duas tags de marcação. Utilizando as tag torna possível dizer ao navegador o que é cada conteúdo, título, parágrafo, botão e tabelas etc. (EIS, 2011).

Figura 13 – Texto entre tags e como aparece no navegador.

Fonte: Autoria própria.

2.6 FPGA

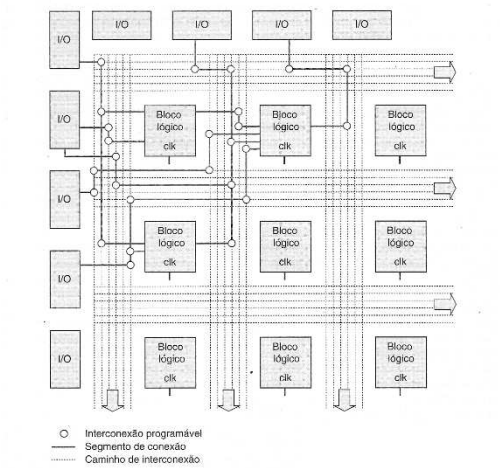
A tecnologia FPGA (*Field Programmable Gate Array*), foi criada em 1983 pela Xilinx. Diferentes dos microcontroladores que os blocos internos já vêm prontos e com uma arquitetura definida, sendo programados para determinadas funções, na FPGA

os blocos estão desconectados, onde estes são conectados por programação para aplicações especificas (BRAGA, 2012).

É possível observar na Figura 14 que os caminhos de interconexão são conectados pela interconexão programável, formando assim segmentos de conexão.

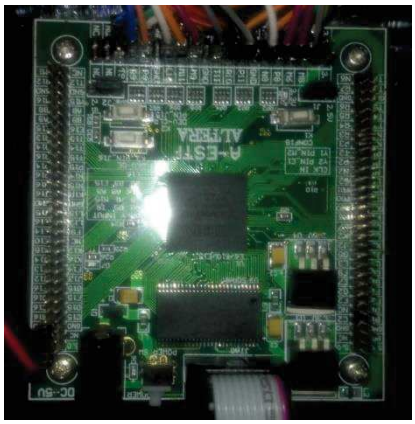
Os dispositivos lógicos programáveis, possui a vantagem de conseguir com um único CI (Circuito Integrado) implementar circuitos que antes eram necessários vários CI individuais. Assim, diminui-se o espaço nas placas de circuito impresso, menor consumo de energia e maior confiabilidade (TOCCI, WIDMER e MOSS, 2011).

Figura 14 – Arquitetura FPGA.



Fonte: TOCCI, WIDMER e MOSS, 2011.

Figura 15 – Kit Cyclone EP4CE15F17C8



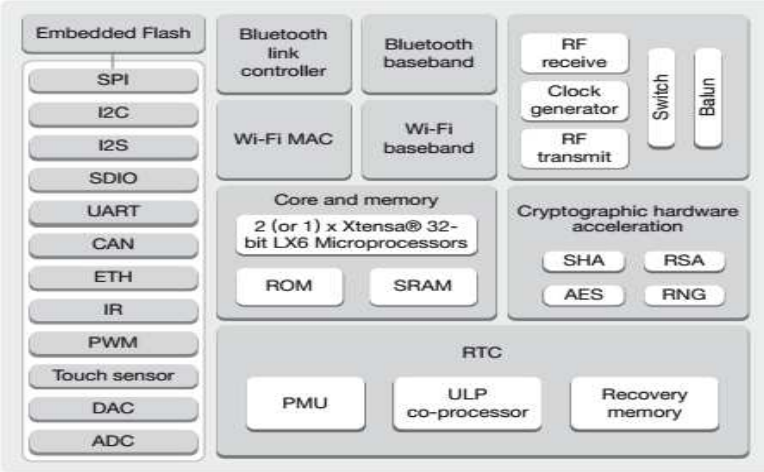
Fonte: Autoria própria.

Os chips de FPGA possuem kits de desenvolvimentos, com o objetivo de tonar o uso dessa mais simples. Como o modelo da Figura 15, onde esse possui em seu kit uma FPGA Cyclone IV EP4CE15F17C8 de 15.408 elementos lógicos, possui uma entrada para 5V, um oscilador de 50 MHz para ser utilizado como *clock*.

2.7 ESP32

O ESP32 é microcontrolador, lançado em 2016 pela empresa Espressif, este possui dois microprocessadores Xtensa® 32-bit LX6, podendo trabalhar com um *clock* de até 240Mhz, tem suporte para conexões Wi-Fi e Bluetooth, tornando assim uma excelentes escolha quando se fala em aplicações moveis ou de IoT(*Internet Of Things*), tendo em vista que esta tem como conceito a interconexão digital de objetos cotiados com a Internet. O ESP32 tem uma gama de recursos além da conectividade Wi-Fi e Blueetooth, como memória RAM de 520K bytes, memória ROM de 448k bytes, 34 portas GPIOS, 2 conversores analógico para digital onde somado estes possuem 18 canais, e saídas PWM etc. A Figura 16 ilustra o diagrama de blocos do ESP32 (ESPRESSIF, 2019a, 2019b; VILLARINO, 2016).

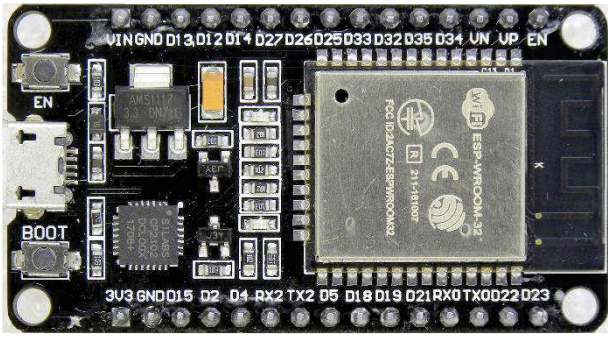
Figura 16 – Diagrama de Blocos ESP32.



Fonte: ESPRESSIF, 2019a.

A Espressif e outras empresas, tendo em vista facilitar o uso do ESP32, lançaram placas de desenvolvimento para este, sendo umas dessas o ESP32-DevKitC, onde este já possui um regulador de tensão de 3,3V modelo AMS1117 para alimentação do ESP, um chip de interface Serial-USB modelo CP2102, com suporte a USB 2.0, conector micro-USB e botões de RESET e LOAD. O DevKit pode ser usada similarmente como um Arduino. O DevKitC possui alguns modelos, as diferenças entre elas ocorrem na localização e quantidade dos pinos. É possível observar um DevKitC de 30 pinos na Figura 17 (MURTA, 2018).

Figura 17 – ESP32 DevKitC.



Fonte: MURTA, 2018.

3 METODOLOGIA

A finalidade desse projeto é fazer um sistema de automação residencial centralizado com sistema de comunicação Wi-Fi entre cliente e servidor, utilizando ESP32 e FPGA em conjunto como central de automação e verificar se isto é viável, para isto foi desenvolvido um painel de demonstração, onde este contém duas plantas semelhantes com sensores e atuadores e um ESP32 sendo utilizado como cliente e uma central de automação com ESP32 e FPGA Altera Cyclone IV.

Os dispositivos utilizados na central de automação.

- 1 ESP32 DevKit V1;
- 1 FPGA Altera Cyclone IV;

Os dispositivos utilizados em cada umas das plantas:

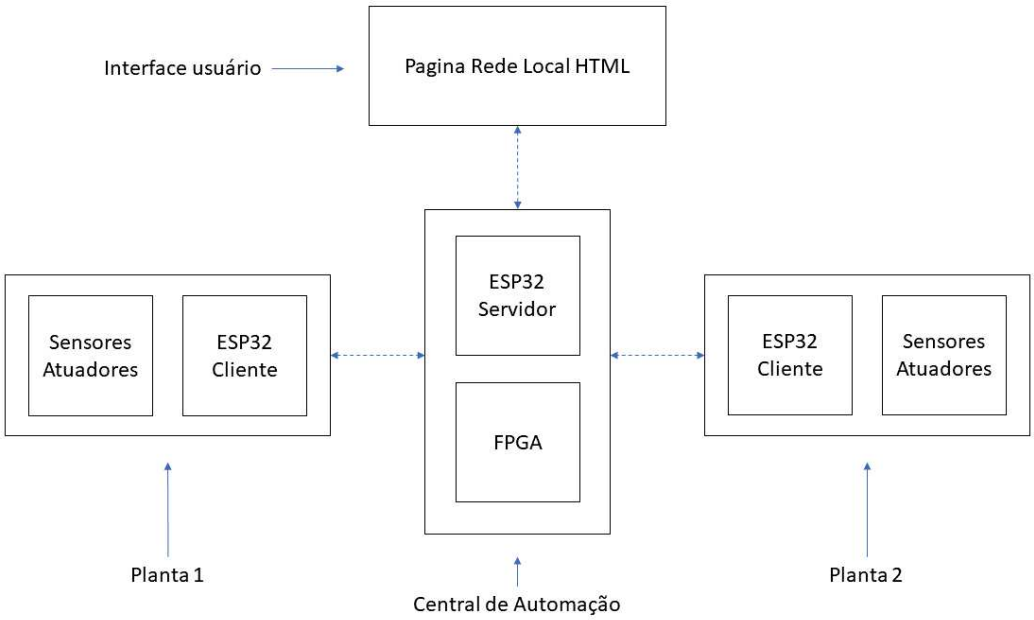
- 1 ESP32 DevKit V1;
- 2 Módulos Relé;
- 1 Modulo PIR;
- 1 Modulo NTC Arduino;
- 1 Modulo LDR Arduino;
- 1 Servomotor MG90S;
- 1 Lâmpada de Halogênio 70W
- 1 Ventilador.

A Figura 18 mostra o sistema de automação em modo centralizado, em que a FPGA em conjunto com o módulo ESP32 servidor formam a central de automação, conectando as duas plantas e a página local HTML, permitindo assim que o usuário faça as mudanças desejadas no sistema.

Foi fabricado um painel de demonstração utilizando uma estrutura de metalon e uma placa de policarbonato material semelhante ao acrílico, este foi utilizado como suporte para fixação dos componentes.

Foi desenvolvido uma página HTML para ser utilizado como interface para o usuário, onde esta se encontra armazenada no ESP32 Servidor e disponível na rede local, podendo ser conectada através de um navegador no computador ou smartphone.

Figura 18 – Diagrama de blocos projeto.



Fonte: Autoria própria.

Também foi desenvolvido um programa utilizando a IDE Arduino para o ESP32 Servidor, ele pode trabalhar individualmente como central de automação ou em conjunto com a FPGA, podendo assim comparar o desempenho da FPGA em relação ao ESP32 nesta aplicação. Foi desenvolvido um programa para o ESP32 Cliente também utilizando a IDE Arduino.

As plantas foram desenvolvidas para coleta de dados pelos sensores PIR, LDR e NTC e atuando no sistema através do servomotor, relé-lâmpada e relé-ventilador. Os dados são enviados para a central e com base nestes a central envia quais as ações devem ser feitas pelos atuadores.

3.1 MONTAGEM PAINEL

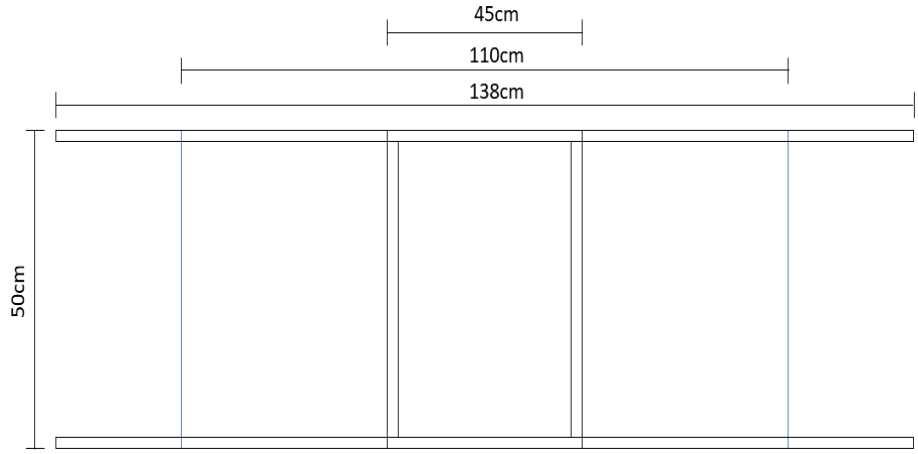
Neste tópico é descrito a montagem estrutural e elétrica do painel de demonstração.

3.1.1 ESTRUTURA

A parte estrutural do painel foi montada a partir de um suporte de metalon, uma placa de policarbonato e duas barras de cantoneira, as Figuras 19 e 20, mostram respectivamente a vista frontal e lateral.

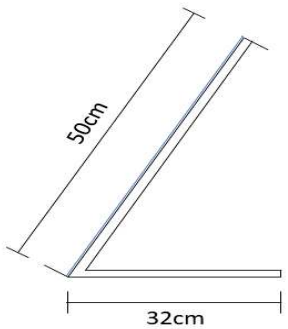
Com base nesses dois modelos foi montado o painel para fixação dos dispositivos.

Figura 19 – Vista frontal painel.



Fonte: Autoria própria.

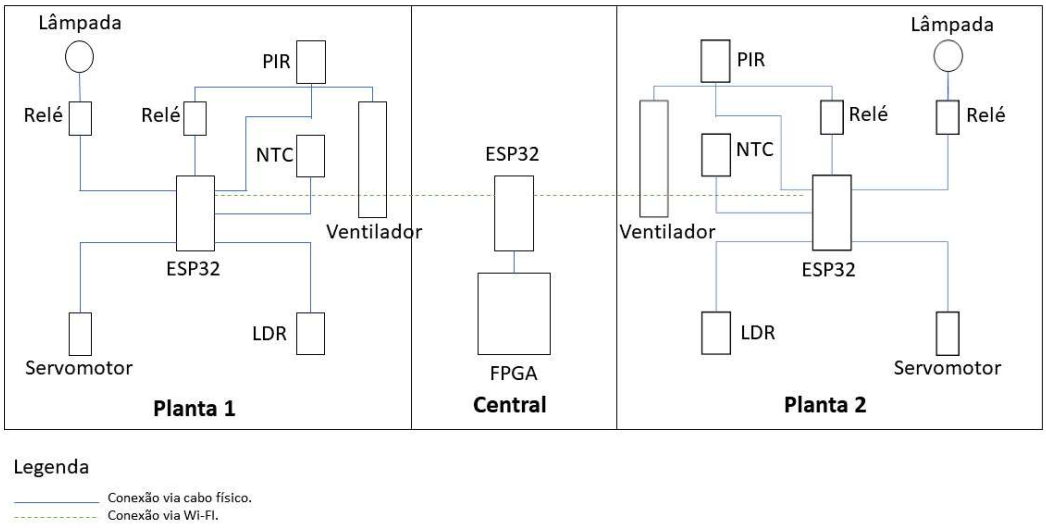
Figura 20 – Vista lateral painel.



Fonte: Autoria própria.

A partir disso foi fixado os componentes no painel de policarbonato de maneira semelhante ao diagrama de blocos da Figura 21.

Figura 21 – Diagrama componentes no painel.



Fonte: Autoria própria.

3.1.2 SISTEMA ELÉTRICO.

Aqui é descrito a montagem do sistema elétrico da planta e da central da automação.

3.1.2.1 SISTEMA ELÉTRICO PLANTAS

A conexão dos sensores nas plantas, foi feita da seguinte maneira:

Os módulos NTC e LDR, o pino A0 da saída analógica destes, foram conectados respectivamente ao pino 34 e 35 do ESP32, onde o 34 é o ADC 1 canal 6 e o 35 ADC 1 canal 7.

O módulo PIR, o seu pino OUT foi conectado ao pino 27 do ESP, onde este trabalha como entrada digital.

Os atuadores foram da seguinte forma:

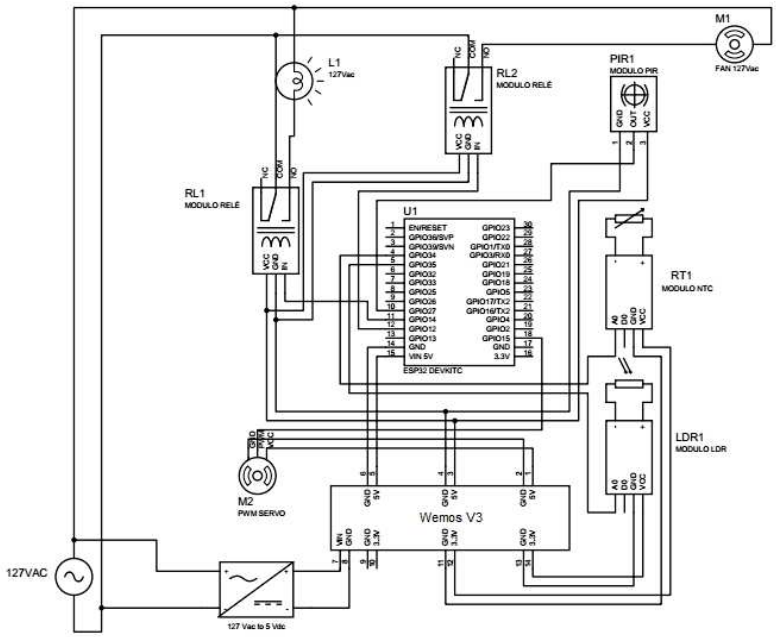
Em relação ao servomotor, o seu pino de entrada de sinal (PWM), foi conectado ao pino 15 do ESP, onde este trabalha como PWM, podendo assim alterar a posição do servo.

Nos relé 1 e relé 2, que correspondem, respectivamente a lâmpada e ao ventilador, onde o pino IN de cada, foi conectado aos pinos 14 e 12 do ESP, estes trabalham como saída digital, o pino comum (COM), foi conectado a uma das fases da rede elétrica alternada e o pino normalmente aberto (NO) conectado a lâmpada para o relé 1 e o ventilador para o relé 2.

A alimentação desses componentes foi através de um dispositivo chamado de Wemos V3, onde este fornece tensões de 5 Vcc e 3,3 Vcc, essas tensões são geradas a partir de uma bateria 18650 que é carregada utilizando uma fonte de smartphone. O ESP 32, servomotor, módulo PIR e relés, são alimentados com 5 Vcc, já os módulos NTC e LDR são alimentados com 3,3 Vcc, para que a tensão máxima no pino A0, não ultrapasse o limite permitido na entrada do ADC do ESP32 que é 3,9 Vcc. A lâmpada e o ventilador utilizam da tensão da rede para sua alimentação, sendo esta 127 Vac.

A Figura 22 mostra o diagrama elétrico das plantas 1 e 2, tendo em vista que estas são semelhantes, mudando somente a posição dos componentes.

Figura 22 – Esquema elétrico



Fonte: Autoria própria.

As conexões foram feitas utilizando fios com conectores tipo JR, vermelho e preto de 0,35mm² para alimentação e fio verde de 0,20mm² para as vias de envio e recebimento de dados/comandos.

3.1.2.2 SISTEMA ELÉTRICO CENTRAL DE AUTOMACÃO

A Comunicação entre ESP32 Servidor e a FPGA acontece de forma paralela, tendo isso em vista foram necessárias as conexões de 22 pinos do ESP com a FPGA,

conforme observado no Quadro 2 e suas respectivas funções, que serão abordadas na explicação dos códigos.

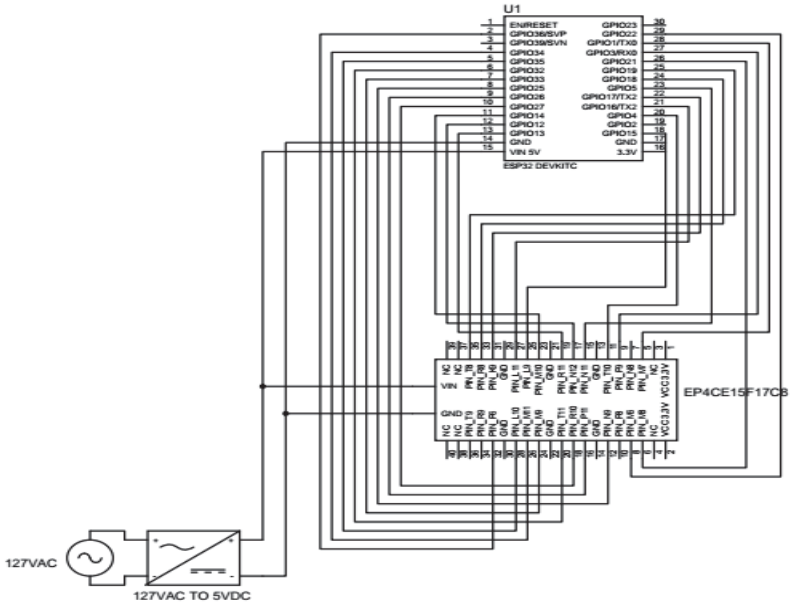
Como a FPGA e ESP32, trabalham no mesmo nível de tensão elétrica nos pinos 3,3Vcc, não foi necessário um circuito extra para esta comunicação, podendo ser feita direta através de fios com conector tipo JR. A Figura 23 ilustra o esquema elétrico das conexões entre o ESP32 e a FPGA, como a placa de desenvolvimento apresenta 160 pinos, foram representados somente os 40 disponíveis no borne utilizado.

Quadro 2 – Relaciona os pinos do ESP32 DevKit com os da FPGA e sua função.

ESP32	FPGA	FUN.	ESP32	FPGA	FUN.	ESP32	FPGA	FUN.
GPIO12	PIN_N12		GPIO25	PIN_N9		GPIO1	PIN_M7	ESC.
GPIO13	PIN_R11	Para o	GPIO26	PIN_P11	Para o	GPIO21	PIN_M8	LIGA
GPIO14	PIN_M10	ESP32	GPIO27	PIN_R10	ESP32	GPIO22	PIN_M6	LIBE.
GPIO15	PIN_L9	Receber	GPIO32	PIN_T11	Enviar	GPIO3	PIN_P9	
GPIO16	PIN_L11	Dados	GPIO33	PIN_M9	Dados	GPIO4	PIN_T10	Seletor
GPIO17	PIN_K9	da	GPI34	PIN_M11	a	GPIO5	PIN_N11	
GPIO18	PIN_R8	FPGA	GPI35	PIN_L10	FPGA			
GPIO19	PIN_T8		GPI36	PIN_P6				

Fonte: Autoria própria.

Figura 23 – Esquema elétrico central



Fonte: Autoria própria.

3.2 CÓDIGOS DA CENTRAL DE AUTOMAÇÂO E PLANTAS

Para o desenvolvimento do projeto, foi necessário a implementação de quatro códigos, sendo dois em C/C++ utilizando a IDE do Arduino para o ESP32 Central e Planta, um em VHDL utilizando o programa Quartus 18.1 Lite Edition para a FPGA e o último em HTML para ser utilizado como interface com o usuário, onde esse foi implementado dentro do código em C feito para o ESP32 Servidor.

A escolha IDE do Arduino para implementação dos códigos do ESP32, se deu pelo fato desta ser mais simples de utilizar e tem uma disponibilidade maior de conteúdo na internet. Esta já possui boa parte das funções do ESP32, já implementadas.

3.2.1 IMPLEMENTAÇÃO DA COMUNICAÇÃO WIFI

Serão apresentados os pontos principais referentes a metodologia de implementação do código de comunicação entre a central e planta.

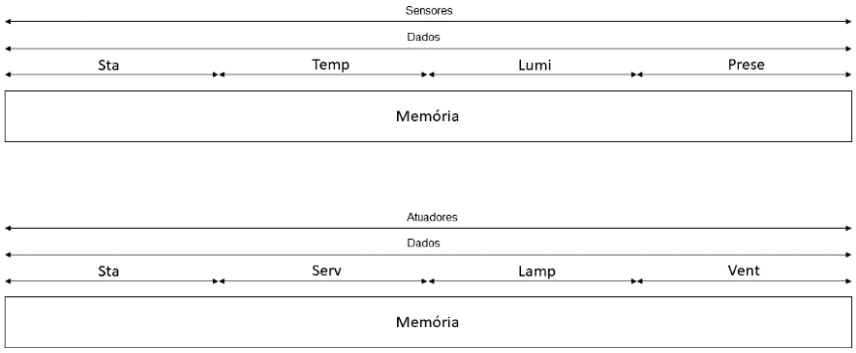
A biblioteca WiFi.h, é responsável pela configuração do WiFi do ESP e, possui variáveis do tipo WiFiServidor e WiFiClient, onde essas possibilitam criar um servidor e fazer conexões na rede local do tipo TCP/IP descrito na secção 3.4.2, utilizando um IP e uma Porta. Tambem possui funções como a Read() e Write(), essas recebem e enviam dados via Wi-Fi cujo tamanho de bytes seja conhecido (ARDUINO, 2019).

Para utilização destas funções foram criadas duas variáveis globais, uma chamada de Atuadores e a outra de Sensores, a primeira é enviada do Servidor para as plantas, contendo as informações necessárias para atualizações dos atuadores no sistema. Já a variável Sensores é enviada das plantas para o servidor, contendo os valores obtidos através dos sensores. Essas variáveis foram criadas tanto no servidor quanto no cliente. A Figura 24 mostra o conteúdo destas variáveis.

O envio e recebimento acontece utilizando a variável Dados, pois esta compartilha o mesmo espaço de memória que as variáveis *Sta*, *Temp*, *Lumi* e *Prese* para os Sensores, *Sta*, *Serv*, *Lamp* e *Vent* para os Atuadores. Isso acontece pelo fato das variáveis Sensores e Atuadores, serem respectivamente dos tipos Datar e Datae, onde esses são *unions* criadas, as *unions* compartilham o mesmo espaço de memória entre suas variáveis semelhante a ponteiros. O que torna possível manipular uma

variável utilizando um tipo diferente. É possível observar essa implementação tanto no código do servidor quanto cliente, respectivamente nos apêndices B e apêndice C.

Figura 24 – Conteúdo atuadores e sensores.



Fonte: Autoria própria.

3.2.2 IMPLEMENTAÇÃO DO CÓDIGO CLIENTE/PLANTAS

O ESP32 Cliente, focou na coleta de dados através dos sensores e agir no sistema através dos atuadores, este se comunica com o servidor via Wi-Fi, utilizando o método descrito no tópico 3.2.1.

O mesmo código foi utilizado para as duas plantas, tendo em vista que essas são iguais, alterando somente a posição dos componentes, a única diferença entre elas é a presença de uma variável de identificação.

O ESP32 faz leitura dos sensores, LDR e NTC utilizando o ADC interno, portanto foi necessário utilizar a biblioteca esp_adc_cal.h, a empresa Espressif, disponibilizou esta biblioteca para calibrar o ADC interno.

Quanto ao tratamento feito em relação ao NTC, foi implementado as equações 3 e 2 descritas no tópico 2.2.3, sendo a primeira utilizada para obter o valor de resistência do NTC a partir da leitura de tensão, a segunda para converter essa resistência para temperatura. Os coeficientes A, B e C utilizados pela equação 2, foram obtidos conforme o Anexo A, os valores de temperatura e resistência para aplicação nas fórmulas de coeficiente foram obtidos através de Spark (2018).

Para o LDR, foi feita a normalização dos valores de tensão obtidos utilizando a função map() da Arduino IDE, para ser mostrados em porcentagem de luminosidade.

Já o sensor PIR, seu valor é lido diretamente, devido ao fato desse apresentar uma saída digital.

Os atuadores, o relé ventilador e relé lâmpada, podem ser acionados diretamente através de um pino de saída digital.

Para o servomotor que simula uma cortina foi necessário a utilização de uma saída PWM, onde este foi configurado com uma resolução de 8 bits, portanto o *duty cycle* em 100% acontece quando é escrito o valor 255 utilizando a função ledcWrite (), esta função é própria do ESP32. Como descrito no tópico 2.3.2, o servomotor trabalha de 5% a 10% do *duty cycle*, sendo esses valores na resolução de 8 bits, respectivamente próximo a 12 e 25. A Figura 26 na página 41 mostra o diagrama de blocos do código do cliente, o código do cliente/planta completo está disponível no apêndice C.

3.2.3 IMPLEMENTAÇÃO DO CÓDIGO SERVIDOR/CENTRAL AUTOMAÇÃO

Para a central de automação foi desenvolvido, um código onde é possível escolher se o microcontrolador trabalhará sozinho ou em conjunto com a FPGA, o objetivo é verificar a eficiência de cada um dos dois.

O Sistema de comunicação entre Servidor e Cliente ESP conforme descrito no tópico 3.2.1.

Além da biblioteca WiFi.h também foi utilizada a biblioteca esp32-hal-cpu.h, com o uso desta consegue-se alterar a frequência de execução do ESP32, o que torna possível realizar testes em 80, 160 e 240 MHz.

Para alteração do sistema de controle ESP32 e para ESP32 com a FPGA, foi criado uma variável, alterando esta é possível trabalhar com a FPGA ou sem.

A comunicação entre ESP32 e FPGA aconteceu de forma paralela, utilizando cerca de 22 conexões, onde 8 foram para envio de dados, 8 para recebimento, 3 para seleção da etapa na FPGA, 1 para salvar os dados enviados na FPGA, 1 para ativar a saída da FPGA e o último para verificar se o ventilador deve ser ligado. Os pinos onde essas conexões aconteceram estão disponíveis no Quadro 2 disponível no tópico 3.1.2.2.

A comunicação foi feita em oito etapas, sendo quatro para o cliente 1 e quatro para o cliente 2. Onde em cada etapa são enviados dados diferentes, a escolha desse método de envio aconteceu pelo motivo de fazer uma comunicação paralela para teste com a FPGA, mas como a há uma limitação de pinos no ESP32, foi necessário ser feita em etapas. A primeira etapa envia o valor de temperatura desejada para FPGA

recebido via página HTML, a segunda etapa envia o valor de temperatura obtido através da planta conectada ao cliente 1 e recebe a média dos últimos 150 valores enviados, a etapa três envia o valor de luminosidade em % e recebe a média dos últimos 150 valores enviados, a quarta etapa recebe a conversão da média dos valores de luminosidade em posição do servomotor. As 4 últimas etapas são iguais só que feitas para o cliente 2.

Para comparação foi montado esses mesmos processos para ESP32 trabalhando individualmente.

Para comunicação com a página local HTML, a conexão acontece de forma semelhante aos ESP32 Cliente, mudando somente a forma como os dados são recebidos e enviados. No recebimento de dados estes são salvos em uma variável do tipo *string* e em seguida esta é comparada com vários ifs, onde em seus parâmetros se encontra outras *strings*, caso as *strings* forem iguais os ifs são executados conforme foram programados, alterando variáveis globais que manipulam os atuadores nos clientes ou os dados presentes na página HTML. Isso é possível pois na página HTML foram implementados botões que utilizam o método GET, onde este envia os parâmetros configurados em conjunto com a *url* da página para o destino. A Figura 25 mostra o descrito grifado em vermelho.

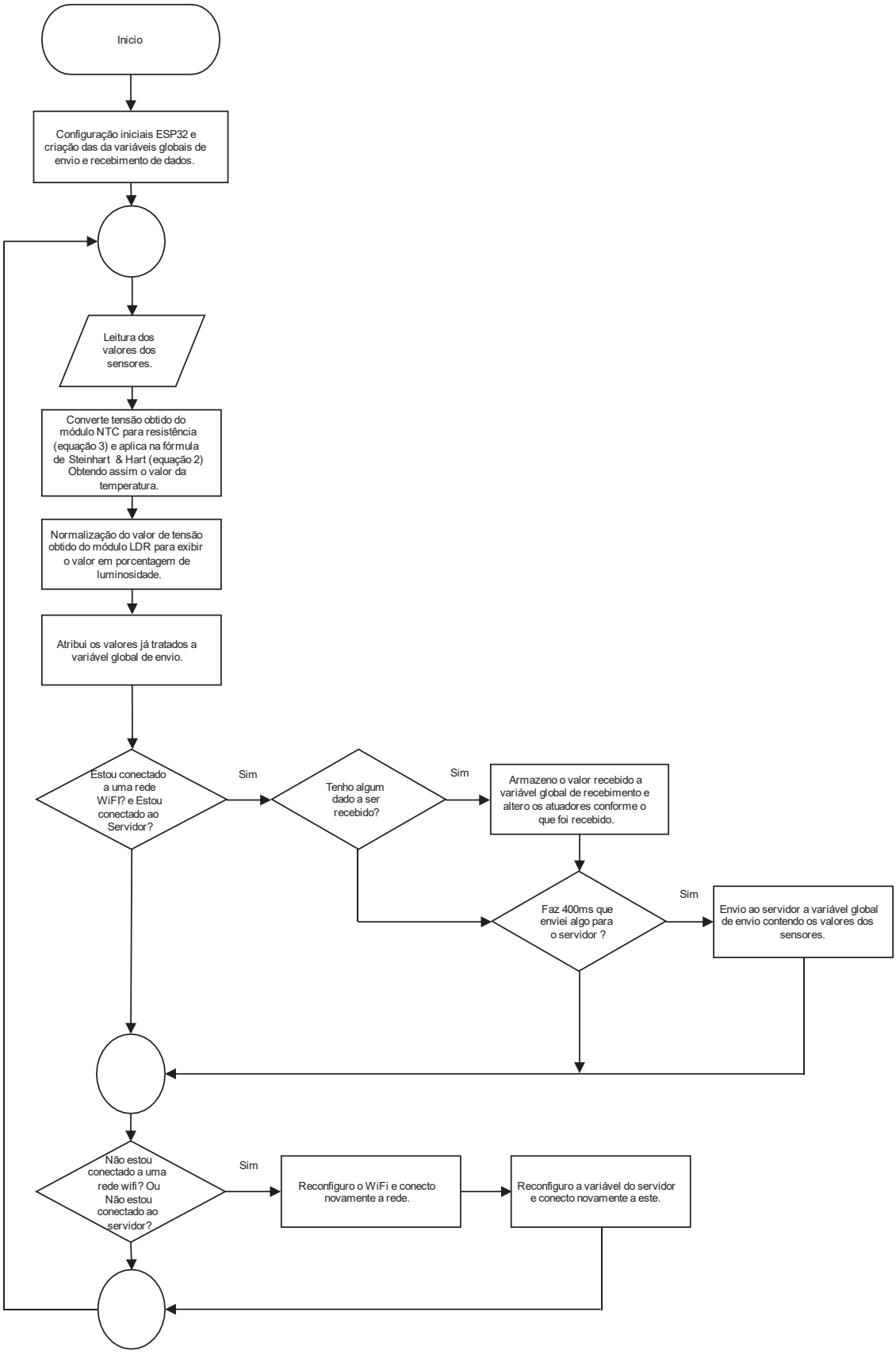
O código HTML é enviado para o navegador depois que todos os ifs são verificados, o método de envio é o Client.println(), onde esse envia a página como se fosse uma *string*.

A Figura 27, mostra um digrama de blocos do código da central. O código completo da central da automação se encontra no apêndice B.

Figura 25 – Implementação recebimento de dados página HTML.

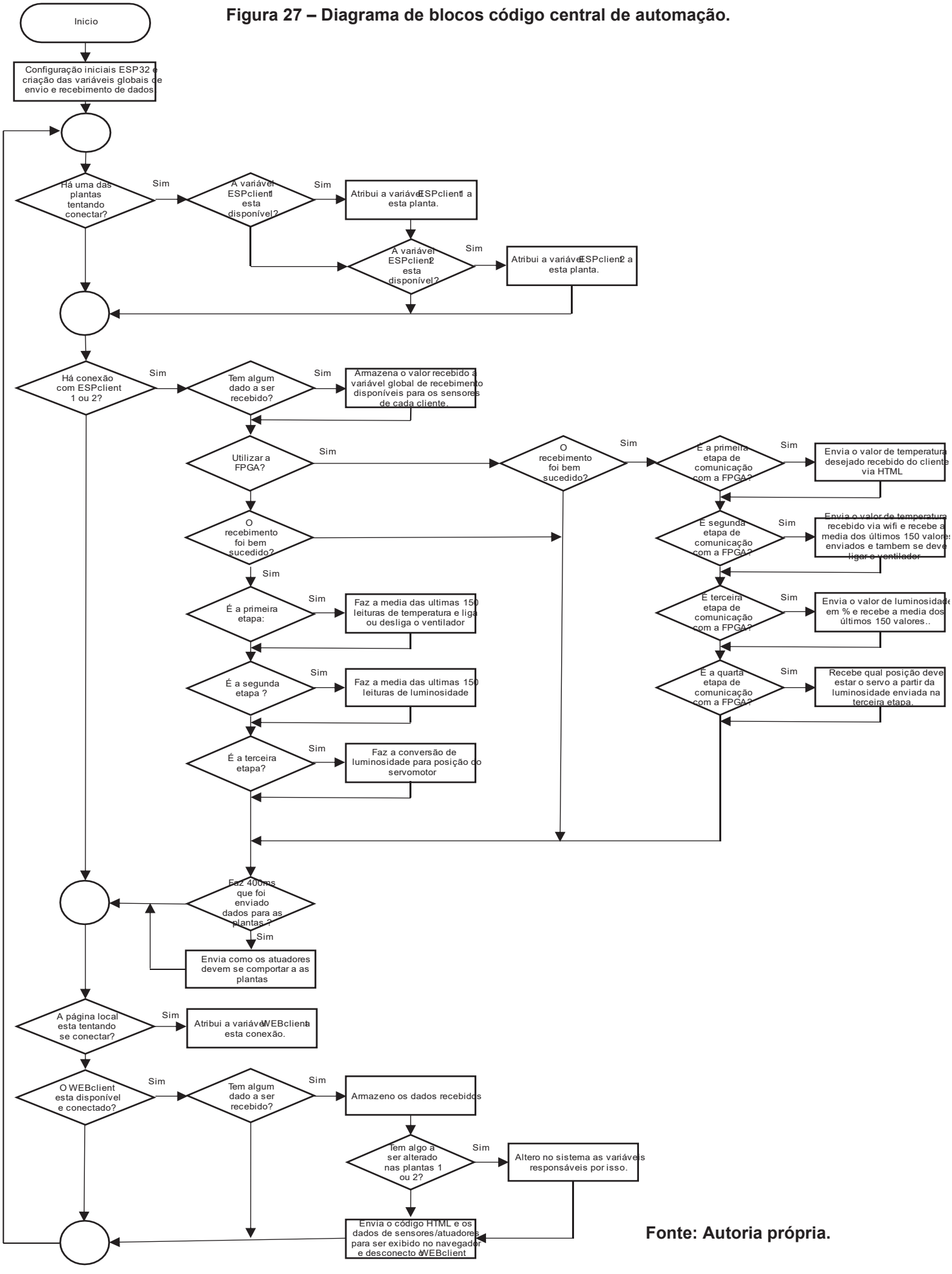
Fonte: Autoria própria.

Figura 26 – Diagrama de blocos código cliente.



Fonte: Autoria própria.

Figura 27 – Diagrama de blocos código central de automação.



Fonte: Autoria própria.

3.2.4 IMPLEMENTAÇÂO DO CÓDIGO VHDL

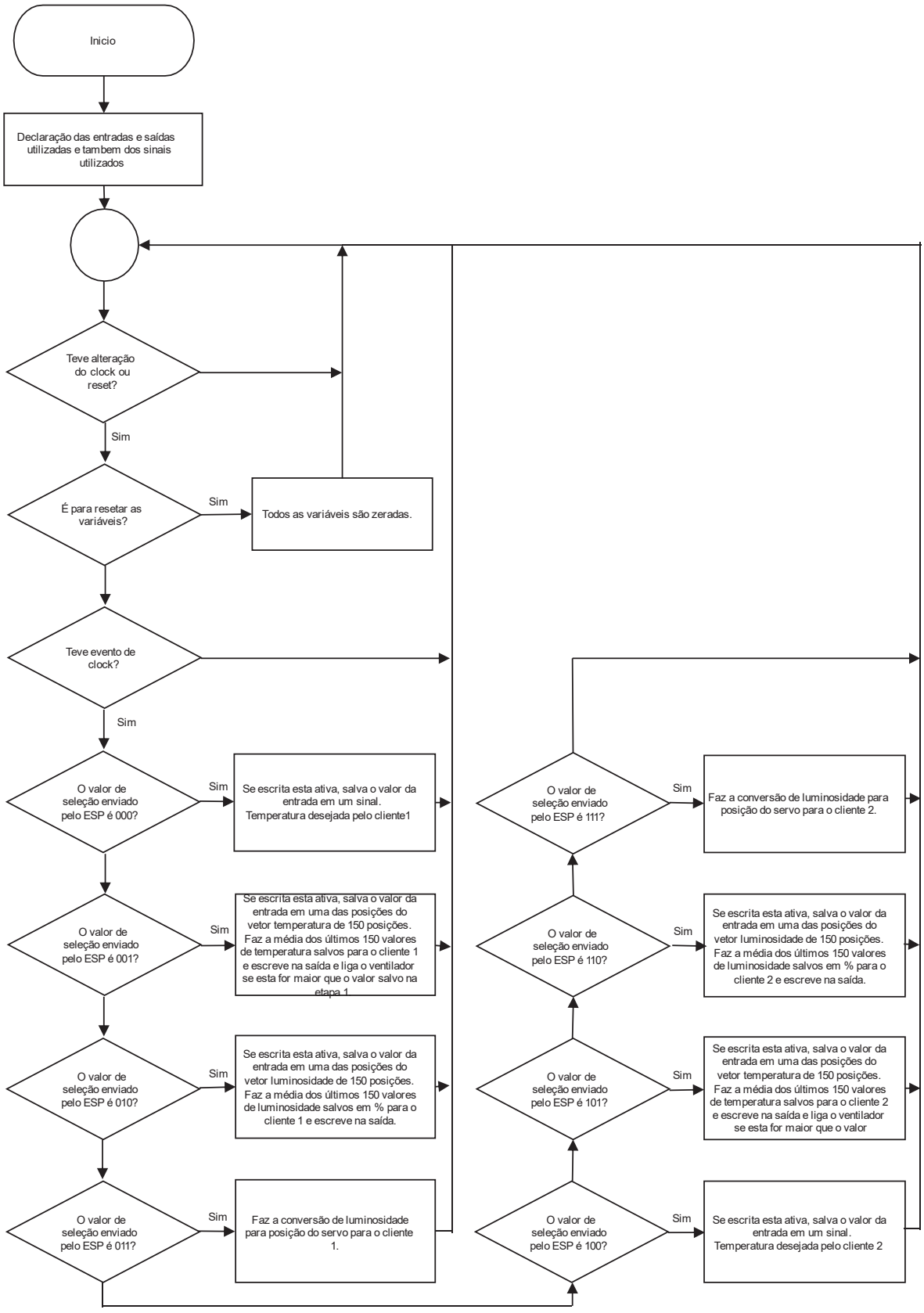
O código VHDL, foi implementando três blocos, sendo entrada, saída e controle. Na entrada e a saída, não foi implementado nenhum tratamento. O bloco de controle fica responsável pelo processamento dos valores de sensores recebidos do ESP32. E faz o controle automático do ventilador e do servomotor.

O bloco de controle dispõe de oito variáveis, sendo essas de entrada ou saída. Destas variáveis uma é o CLK responsável por manter o código em execução e a outra é o RESET, utilizada para reiniciar as variáveis. As outras variáveis são as conexões disponíveis no quadro 2, onde essas são responsáveis pela comunicação com o ESP32, sendo as de entrada de dados ESCRITA, LIBERA, NUM (Vetor de 3 posições) e DADOSI (Vetor de 8 posições), e as de saída de dados LIGA e DADOSO (Vetor de 8 posições).

O código foi implementando utilizando a metodologia de código sequencial para as 8 etapas presentes na comunicação. Onde essas são selecionadas a partir da variável NUM, onde essa assume valores de 0 a 111, a primeira etapa (000) é onde o valor desejado de temperatura é salvo na FPGA, segunda etapa (001) é salvo o valor de temperatura do sensor e retorna a média dos últimos 150 valores salvos e liga o ventilador caso a temperatura for superior a temperatura salva na etapa 1, etapa três (010) salva o valor da luminosidade em % e retorna a média dos últimos 150 valores salvos, etapa quatro (011) é feito a conversão da média de luminosidade em posição para o servomotor. As quatro etapas restantes são 100, 101, 110 e 111. Essas são iguais as 4 primeiras, so que para o cliente 2.

Para fazer a média dos valores dos sensores foram utilizadas funções para sua implementação acontecer de forma paralela. Também foi utilizada uma função para converter luminosidade em posição do servomotor, tendo em vista que este ira simular uma cortina que trabalha com a luminosidade natural. A Figura 28 mostra o digrama de blocos do código implementado e é possível encontrar o código implementado no apêndice E.

Figura 28 – Digrama de blocos código VHDL.



Fonte: Autoria própria.

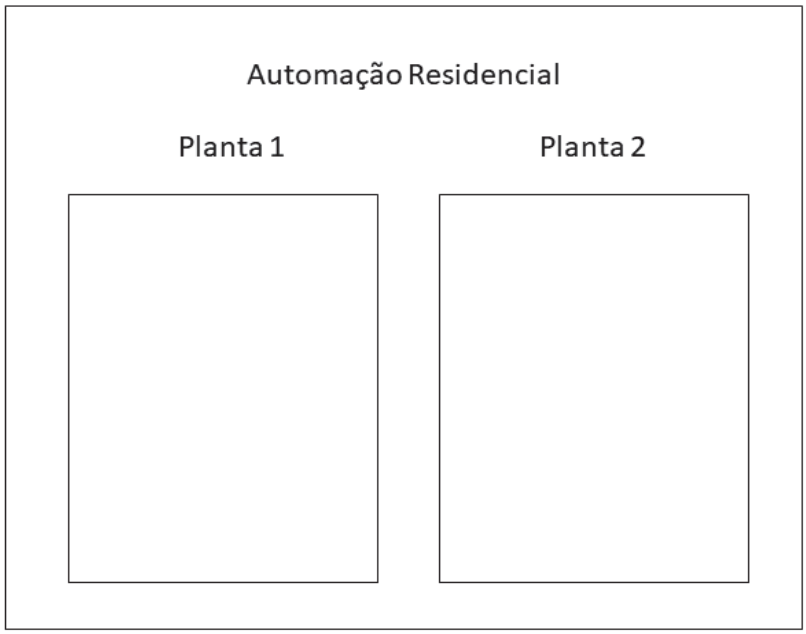
3.2.5 IMPLEMENTAÇÃO DO CÓDIGO HTML PÁGINA LOCAL

A página HTML, foi desenvolvida para servir de interface com o usuário, esta foi desenvolvida para mostrar os valores dos sensores e também para tornar possível a interação com o sistema. Essa mostra os valores dos sensores PIR, NTC e LDR. Torna possível ativar ou desligar a lâmpada e o ventilador e também alterar a posição do servomotor. Possui a opção de ativar o modo automático também.

Para o seu desenvolvimento foi utilizado o sistema de tabelas e botões com o método GET, esse descrito no tópico 3.2.3. O seu código foi implementado dentro do código C da central de automação.

A página foi programada para atualizar a cada 5s, para isso foi utilizado um pequeno script Java. O objetivo com isso é atualizar os valores mostrados. A Figura 29 mostra um diagrama da ideia inicial utilizada em sua implementação. É possível observar o código HTML desenvolvido em conjunto com o código C da central no apêndice B e somente o seu código no Apêndice E.

Figura 29 – Diagrama Inicial página HTML.



Fonte: Autoria própria.

3.3 METODOLOGIA DE TESTES

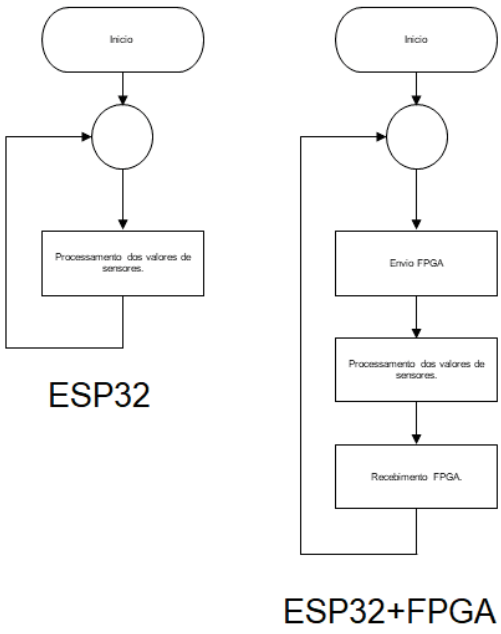
Para realizar a comparação entre ESP32 e FPGA, serão realizados seis ensaios, onde três desses são somente com o ESP32 e o restante com ESP32 + FPGA, serão realizados testes variando a frequência de execução do ESP32. Com o objetivo de obter o tempo de execução total das etapas descritas nos tópicos 3.2.3 e 3.2.4. O quadro 3, apresenta a metodologia dos ensaios.

Quadro 3 – Metodologia de ensaios ESP32 e FPGA.

	Arquitetura	Frequência
Ensaio 1	ESP32	80Mhz
Ensaio 2	ESP32	160Mhz
Ensaio 3	ESP32	240Mhz
Ensaio 4	ESP32 + FPGA	80Mhz
Ensaio 5	ESP32 + FPGA	160Mhz
Ensaio 6	ESP32 + FPGA	240Mhz

Fonte: Autoria própria,

Figura 30 – Diagrama blocos ESP32 e ESP32+FPGA



Fonte: Autoria própria.

A Figura 30, mostra o processo de tratamento dos dados no ESP32 e ESP32+FPGA, é possível ver que ao utilizar ESP32+FPGA é adicionado ao processo o tempo de envio e recebimento, tendo isso em vista também será feito a análise do código VHDL utilizando a análise de tempo disponível no Quartus, mostrando assim realmente o tempo de processamento da FPGA.

4 RESULTADOS

Serão abortados os resultados obtidos em relação a parte física e elétrica do painel, o funcionamento dos códigos e o desempenho da FPGA em conjunto com o ESP32.

4.1 RESULTADOS ESTRUTURA E SISTEMA ELÉTRICO

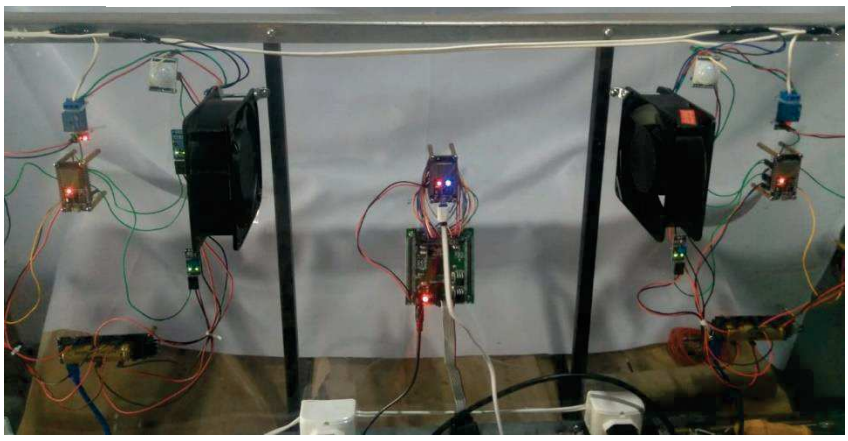
Com base no que foi descrito no tópico 3.1.1 é possível observar a estrutura já montada na Figura 31.

Figura 31 – Estrutura montada.

Fonte: Autoria

A Figura 32 mostra o painel montado já com os componentes.

Figura 32 – Painel montado com componentes.



Fonte: Autoria própria.

A parte elétrica do painel apresentou um funcionamento correto. Foi medido a corrente alternada elétrica, com os ventiladores e lâmpadas ligadas e também com esses desligados é possível observar esses valores na Figura 33.

Utilizando desses valores é possível calcular a potência elétrica necessária para alimentar o painel. A potência quando ventiladores e lâmpadas estão ligados foi de cerca de 199,32W e a potência quando so os dispositivos menores estão ativos foi de aproximadamente 3,72W. O valor elevado quando as lâmpadas e ventiladores estão ligados, acontece pelo fato de ter sido utilizados duas lâmpadas de 70W cada, que representa 70% da potência total.

Figura 33 – Valores de corrente elétrica e tensão.

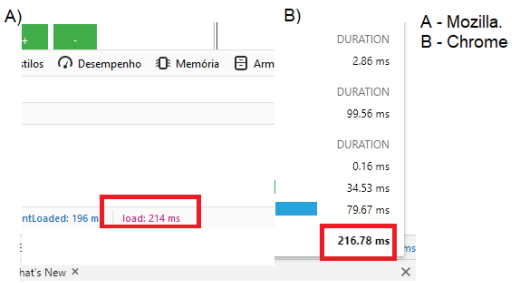
Fonte: Autoria própria.

4.2 PÁGINA HTML

A página HTML apresentou uma aparência simples e agradável e demostrou um bom desempenho nos navegadores utilizados para testes. Sendo esses o Mozilla Firefox e Google Chrome.

O tempo de carregamento da página foi próximo tanto no Mozilla quanto no Chrome, sendo 214ms no Mozilla e 216,78ms no Chrome. É possível observar esses valores na Figura 34.

Figura 34 – Teste HTML Mozilla e Chrome



Fonte: Autoria própria.

O Apêndice A apresenta imagens de teste da página HTML no navegador Mozilla e Mozilla Mobile.

Todas as funções implementadas no HTML funcionaram de maneira correta. Apesar de ter o mesmo tempo de carregamento, a página apresentou um desempenho melhor quando utilizado o navegador Mozilla, tanto quanto para computador e mobile.

4.3 RESULTADO PLANTAS

O desempenho do código foi verificado utilizando a função micros () onde essa retorna o tempo de execução do código em µs, foi verificado o tempo de envio e recebimento de dados via WiFi e o tempo de execução do código completo. A Figura 35 mostra os dois processos que foram verificados e onde foram colocados os pontos de coleta de tempo em vermelho, para verificar o tempo de execução total do código foi colocado um ponto no início e outro no final.

A Figura 35 também apresenta o tempo de execução em µs, é possível ver que o tempo de recebimento foi de 206 µs e o de envio de 3583 µs, o código inteiro apresentou desempenho próximo de 4000 µs, quando não ocorre um evento de envio

Figura 35 – Desempenho código plantas.

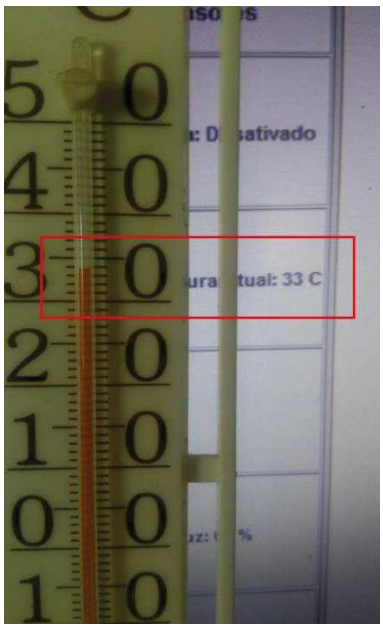
```
if(ESPClient.available() > 0){ //Verifica se tem dados disponiveis p  3981 :TEMPO EXECUÇÃO CODIGO
  t2 = micros();
  statu = ESPClient.read(Atuadores.Dados, sizeof(Atuadores.Dados));/  206 :TEMPO RECEBIMENTO
  t2 = micros() - t2;                                                 4232 :TEMPO EXECUÇÃO CODIGO
  h1 = 1;
}                                                                     4009 :TEMPO EXECUÇÃO CODIGO

if (millis() - te >= 400){//Envia os dados para o Servidor de 40      3940 :TEMPO EXECUÇÃO CODIGO
  t3 = micros();
  EnviarTudo(Sensores.Dados,sizeof(Sensores.Dados),ESPClient);        3583 :TEMPO ENVIO
  t3 = micros() - t3;                                                 7568 :TEMPO EXECUÇÃO CODIGO
  h2 = 1;
  te = millis();                                                      4045 :TEMPO EXECUÇÃO CODIGO
}
```

Fonte: Autoria própria.

ou recebimento, quando um ocorre o tempo de execução total foi de 4232 µs para recebimento e 7568 µs para envio.

Quanto aos valores de sensores obtidos pelas plantas, estas apresentaram um desempenho satisfatório, tanto para os valores de temperatura quanto para luminosidade e presença. A Figura 36 mostra a comparação da temperatura mostrada na interface HTML com um termômetro.

Figura 36 – Teste temperatura.



Fonte: Autoria própria.

Os testes foram realizados somente em uma planta, tendo em vista que essas são iguais.

4.4 RESULTADOS CENTRAL DE AUTOMAÇÃO

Para a central de automação foram realizados teste de tempo de envio e recebimento de dados via Wi-Fi para as plantas e também para a página HTML, e também de desempenho do ESP32 sendo utilizado com a FPGA. Também foi obtido a quantidade de elementos lógicos utilizados na FPGA.

Os testes de conexão WiFi, foram executados de forma similar ao da planta, marcando o tempo de início e final da tarefa. É possível observar os tempos de execução obtidos na Tabela 1 sendo esses em µs, onde é possível notar que é necessário um tempo relativamente maior para enviar os dados do que receber. O

tempo de envio e recebimento da planta 1 é maior que o da planta 2. Sendo o tempo de envio para a planta 1 próximo a 1700 µs e para a planta 2 1400 µs, o tempo de recebimento da planta 1 ficou próximo a 300 µs e para planta 2 próximo a 200 µs.

Tabela 1 – Tempos de envio e recebimento WiFi central de automação.

Página HTML	Envio	90841us	91456us	90985us
	Recebimento	13us	13us	13us
Cliente 1	Envio	1715us	1953us	1660us
	Recebimento	327us	319us	200us
Cliente 2	Envio	1451us	1409us	1306us
	Recebimento	212us	201us	394us

Fonte: Autoria própria.

O tempo de envio para a página HTML é de cerca de 91000 µs, um valor elevado, comparados com a planta 1 e 2, mas a quantidade de dados enviados para a página local é consideravelmente maior em cada interação.

Para comparação do desempenho da ESP32 + FPGA em relação ao ESP32, foram montadas etapas, onde foi marcado o tempo de execução de cada uma utilizando a função micros (). Onde a frequência de operação foi alterada entre 80Mhz, 160Mhz e 240MHz. Foram descritas nos tópicos 3.2.3 e 3.2.4 as etapas de comunicação e o que cada uma faz e no tópico 3.3 a metodologia de ensaios.

Através dos valores obtidos através da interface de comunicação serial da IDE do Arduino, foi montado a Tabela 2.

Observando o quadro é possível notar, que o ESP32+FPGA apresentou um desempenho relativamente igual ou superior nos casos que exigiam fazer a média dos valores dos sensores. Estes estão grifados em azul. Já na conversão de luminosidade para posição do servo a ESP+FPGA apresentou um desempenho inferior grifado em amarelo. Os casos grifados em vermelhos dos valores obtidos do cliente 2 apresentaram um valor discrepante quando se comparados ao cliente 1. Portanto foram ignorados, tendo o fato que foram implementados de maneira igual sendo esse um resultado não esperado.

Também é possível observar que o aumento da frequência de execução do código na ESP32, faz uma redução proporcionalmente inversa, no tempo de execução das instruções.

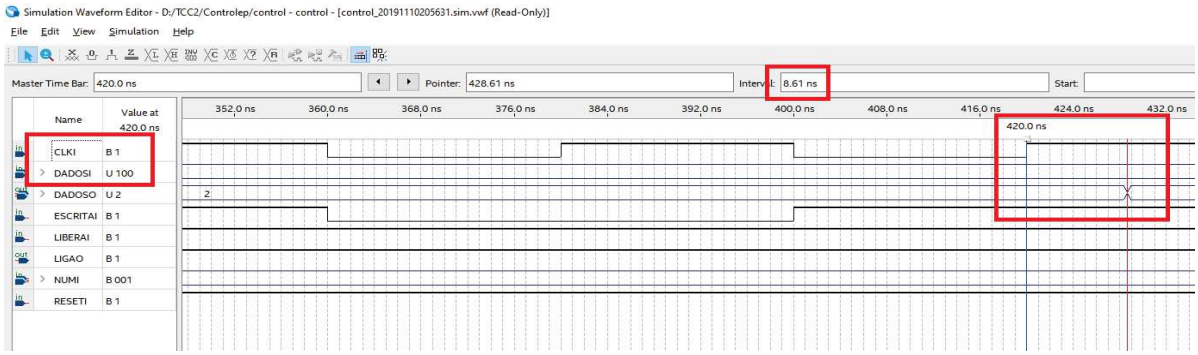
Tabela 2 – Comparação ESP+FPGA e ESP.

		ESP32			ESP32 + FPGA		
Frequência	Cliente	80MHz	160MHz	240MHz	80MHz	160MHz	240MHz
Temp.	1	12us	6us	4us	11us	6us	4us
		12us	6us	4us	10us	6us	3us
		12us	6us	4us	10us	5us	4us
	2	12us	6us	4us	22us	17us	11us
		12us	6us	4us	27us	17us	12us
		12us	6us	5us	23us	17us	11us
Lumi.	1	12us	6us	4us	10us	6us	3us
		12us	6us	4us	10us	5us	3us
		12us	5us	4us	10us	5us	3us
	2	12us	6us	4us	10us	5us	4us
		12us	6us	3us	10us	5us	3us
		12us	6us	4us	10us	5us	3us
Servo	1	3us	1us	1us	6us	3us	2us
		3us	2us	1us	7us	3us	2us
		2us	2us	1us	7us	3us	2us
	2	7us	9us	4us	7us	4us	2us
		7us	9us	5us	7us	3us	2us
		7us	9us	5us	7us	3us	3us
Azul	Resultado ESP+FPGA Melhor ou Igual ao ESP						
Amarelo	Resultado ESP+FPGA Inferior ao ESP						
Vermelho	Valores desconsiderados pela discrepância.						

Fonte: Autoria própria.

Quanto a implementação do código em VHDL. A Figura 39, apresenta a análise de tempo.

Figura 37 – Tempo execução etapa 1 FPGA.



Fonte: Autoria própria.

A partir da Figura 39 é possível obter o tempo aproximado da execução da segunda etapa, onde essa retorna a média dos últimos 150 valores de temperatura para o cliente 1. Sendo esse 8,61 ns após a borda de subida do *clock* de execução da FPGA. Comparando com os valores obtidos do ESP32 e ESP32+FPGA em 240MHz disponíveis no quadro 4 para temperatura do cliente 1. Pode-se observar que a FPGA apresenta um desempenho superior ao ESP32 é que o valor elevado da ESP32+FPGA ocorre devido a atraso de comunicação.

A Figura 40 apresenta a quantidade de elementos lógicos utilizados, cerca de 12629.

É possível dizer, que para este caso, utilizar a FPGA em conjunto com o ESP32 não apresenta um desempenho significante. Tendo em vista o alto tempo para efetuar a comunicação entre ESP32 e FPGA é também pelo fato das operações utilizadas na comparação serem relativamente simples.

Figura 38 – Elementos lógicos

Flow Status	Successful - Mon Nov 04 20:43:17 2019
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	control
Top-level Entity Name	media
Family	Cyclone IV E
Device	EP4CE15F17C8
Timing Models	Final
Total logic elements	12,629 / 15,408 (82 %)
Total registers	4927
Total pins	24 / 166 (14 %)
Total virtual pins	0
Total memory bits	0 / 516,096 (0 %)
Embedded Multiplier 9-bit elements	0 / 112 (0 %)
Total PLLs	0 / 4 (0 %)

Fonte: Autoria própria.

5 CONCLUSÃO

O objetivo deste projeto foi implementar um protótipo de automação residencial centralizado utilizando ESP32 em conjunto com a FPGA como central de automação e rede Wi-Fi de maneira genérica e comparar o desempenho do ESP32 e da FPGA.

A fundamentação teórica forneceu o conhecimento necessário para implementação desse projeto tendo em vista que conforme era pesquisado foi descobrindo novos meios de implementar do sistema, como os módulos de sensores já disponíveis, tendo em vista que é necessário um circuito para utilização destes.

A página HTML apresentou um resultado satisfatório, sendo utilizada como interface com o usuário. Tendo em vista que essa proporciona o acesso ao usuário para qualquer dispositivo que possui acesso a um navegador, tornando assim o sistema mais acessível.

As plantas apresentaram um resultado satisfatório na obtenção dos dados de temperatura e luminosidade. Também proporcionaram um ótimo desempenho no acionamento dos atuadores, principalmente o servomotor, tendo em vista que esse utiliza PWM para alterar sua posição.

Analisando os resultados é possível observar que o ESP32 sozinho consegue desempenhar o papel de central de automação, tendo que em vista que o desempenho utilizando a ESP32+FPGA em relação ao ESP32 trabalhando individualmente foi na casa de µs, sendo assim imperceptível para qualquer aplicação simples.

É importante notar que o conceito de computação heterogênea foi aplicado aqui, utilizando a FPGA como acelerador. O desempenho da FPGA foi superior ao ESP32, mas como o atraso de comunicação foi relativamente alto o processo final acabou obtendo um desempenho parecido. Utilizando uma aplicação que exija um desempenho elevado, o conjunto ESP32+FPGA pode ser tornar viável, como aplicações de criptografia de dados, um ponto muito importante quando se trata de comunicação a distância.

O ESP32 é um microcontrolador versátil e relativamente fácil de trabalhar, tendo em vista que este possui suporte para a IDE do Arduino, possuindo assim bastante conteúdo disponível para ser encontrado na internet. Além de um alto poder de processamento.

O sistema pode ser implementado, não somente na automação residencial, este pode ser modificado para ser utilizado em sistemas de monitoramento a distância via rede LAN ou se desenvolvido um meio de alimentação utilizando energia solar, pode ser utilizado como central de captação de dados climáticos. Pode ser uma boa opção tendo em vista que o Brasil é um país agrícola, sendo esse um dos setores que mais se desenvolvem a cada ano. Tendo em vista que o ESP32 possui módulos com capacidade de conexões a longas distância utilizando tecnologias como o Lora ou o ESPNOW, permitindo assim cobrir uma grande área diferente do WiFi.

É possível tornar o acesso da página HTML disponível em toda rede não somente a rede local, isso pode ser feito utilizando tecnologias como o DNSS ou utilizando um servidor para armazenar os dados dos sensores, sendo essa última opção um pouco mais complexa a implementação.

REFERÊNCIAS

ACCARDI, A.; DODONOV, E. Automação Residencial: Elementos Básicos, Arquiteturas, Setores, Aplicações e Protocolos. **Revista T.I.S**, São Carlos, v. 1, n. 2, p. 156-166, 2012. ISSN 2316-2872.

ADAFRUIT INDUSTRIES. PIR Motion Sensor. **Adafruit**, 2014. Disponivel em: <https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/>. Acesso em: 25 Outubro 2019.

AMETHERM. NTC Thermistors Steinhart and Hart Equation. **Ametherm**, 2013. Disponivel em: <https://www.ametherm.com/thermistor/ntc-thermistors-steinhart-and-hart-equation>. Acesso em: 2 Novembro 2019.

ARDUINO. WiFi library. **Arduino**, 2019. Disponivel em: <https://www.arduino.cc/en/Reference/WiFi>. Acesso em: 29 Outubro 2019.

ATHOS ELECTRONICS. Relé – O que é e como funciona. **Athos Electronics**, 2019. Disponivel em: <https://athoselectronics.com/rele/>. Acesso em: 25 Outubro 2019.

AURELIANO, A. Microcontroladores. **Fiozera**, 2017. Disponivel em: <https://fiozera.com.br/microcontroladores-914a59cbf7de>. Acesso em: 11 Maio 2017.

BOLZANI, C. A. M. **Desenvolvimento de um Simulador de Controle de Dispositivos Residenciais Inteligentes: Uma Introdução aos Sistemas Domóticos**. 2004. 131 f. Dissertação (Mestre em Engenharia) - Escola Politécnica da Universidade de São Paulo. São Paulo. 2004.

BRAGA, N. C. **Curso de Eletrônica:** Eletrônica Digital II. 1ª. ed. São Paulo: Instituto Newton C. Braga, v. IV, 2012.

COMPONENTS101. MG90S – Metal Gear Micro Servo Motor. **Components101**, 2019. Disponivel em: <https://components101.com/motors/mg90s-metal-gear-servo-motor>. Acesso em: 26 Outubro 2019.

EIS, D. O básico: O que é HTML?. **Tableless**, 2011. Disponivel em: <https://tableless.com.br/o-que-html-basico/>. Acesso em: 27 Outubro 2019.

ESPRESSIF. ESP32 SeriesDatasheet. **Espressif**, 2019a. Disponivel em: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>. Acesso em: 25 Outubro 2019.

ESPRESSIF. ESP32-WROOM-32Datasheet. **Espressif**, 2019b. Disponivel em: <https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf>. Acesso em: 25 Outubro 2019.

MOTA, A. O que é Servomotor? | Controlando um Servo com Arduino. **Vida de Silicio**, 2017. Disponivel em: <https://portal.vidadesilicio.com.br/o-que-e-servomotor/>. Acesso em: 25 Outubro 2019.

MURATORI, J. R.; DAL BÓ, P. H. Automação residencial: historico, definições e conceitos. **O Setor elétrico**, São Paulo, n. 62, p. 70, Março 2011.

MURATORI, J. R.; DAL BÓ, P. H. **Automação Residencial:** Conceitos e Aplicações. 2ª. ed. Belo Horizonte: Educere, 2014.

MURTA, J. G. A. Conhecendo o ESP32 – Introdução (1). **Eletrogate**, 2018. Disponivel em: <https://blog.eletrogate.com/conhecendo-o-esp32-introducao-1/>. Acesso em: 26 Outubro 2019.

OLIVEIRA, D. V. G.; PETREK, F. J. **Sistema de automação residencial controlado via web**. 2014. 170. Monografia (Graduação Engenharia Industrial Elétrica) - Universidade Tecnológica Federal do Paraná. Curitiba. 2014.

OLIVEIRA, E. Como usar com Arduino - Módulo Relé 5V 1 Canal. **MasterWalker Eletronic Shop**, 2019. Disponivel em: <https://blogmasterwalkershop.com.br/arduino/como-usar-com-arduino-modulo-rele-5v-1-canal/>. Acesso em: 25 Outubro 2019.

PAZOS, F. **Automação de Sistemas & Robótica**. Rio de Janeiro: Axcel Books, 2002.

PEDRONI, V. A. **Eletrônica digital moderna e VHDL**. Rio de Janeiro: Elsevier, 2010.

ROSSI, G. P. **Módulos de Hardware Para Aplicação em Arquiteturas Descentralizadas de Automação Residencial**. 2018. 130. Monografia (Graduação Engenharia Eletrônica) - Universidade Tecnológica Federal do Paraná. Campo Mourão. 2018.

ROVERI, M. R. **Automação Residencial**. 2012. 87 f. Monografia (Graduação Tecnólogo em Redes de Computadores) - Faculdade Politec. Santa Bárbara d'Oeste. 2012.

SHOP1817254 STORE. 5 pçs módulo de sensor térmico termistor critesistor módulo interruptor temperatura para diy carro eletrônico. **Aliexpress**, 2019. Disponivel em: <https://pt.aliexpress.com/item/32841441401.html>. Acesso em: 25 Outubro 2019.

SILVEIRA, C. B. Servo Motor: Veja como Funciona e Quais os Tipos. **Citisystems**, 2019. Disponivel em: <https://www.citisystems.com.br/servo-motor/>. Acesso em: 25 Outubro 2019.

SPARK. NTC Thermistors and You - Converting an ADC (analogRead) value into a Temperature. **Spark**, 2018. Disponivel em: <http://sparks.gogo.co.nz/ntc_thermistor.html>. Acesso em: 3 Novembro 2019.

STROSKI, P. N. NTC e PTC. **Electrical e Library**, 2017. Disponivel em: <http://www.electricalelibrary.com/2017/08/14/ntc-e-ptc/>. Acesso em: 24 Outubro 2019.

TANENBAUM, A. S. **Redes de Computadores**. 5ª. ed. [S.l.]: Pearson, 2011.

TERUEL, E. C. **Uma proposta de framework para sistemas de automação residencial com interface para WEB**. 2008. 158 f. Dissertação (Mestre em Tecnologia) - Centro Estadual de Educação Tecnológica Paula Souza. São Paulo. 2008.

THOMAZINI, D.; ALBUQUERQUE, P. U. B. **Sensores industriais:** fundamentos e aplicações. 7ª. ed. São Paulo: Érica, 2010.

TOCCI, R. J.; WIDMER, N. S.; MOSS, G. L. **Sistemas Digitais:** Princípios e aplicações. 11ª. ed. São Paulo: Pearson Prentice Hall, 2011.

TORRES, G. **Redes de Computadores**. 2ª. ed. Rio de Janeiro: Novaterra, 2014.

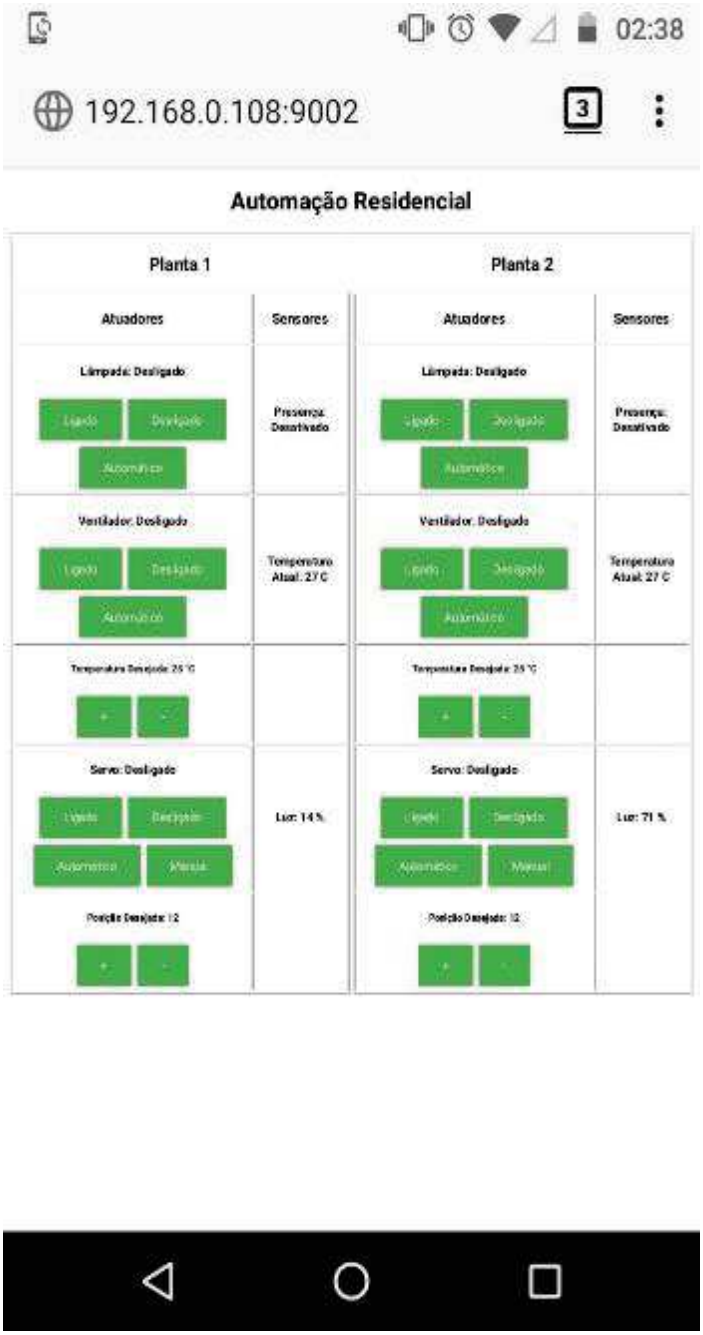
VILLARINO, J. Internet das Coisas: Um Desenho do Futuro. **Proof**, 2016. Disponivel em: <https://www.proof.com.br/blog/internet-das-coisas/>. Acesso em: 25 Outubro 2019.

WHATWG. HTML. **Whatwg**, 2019. Disponivel em: <https://html.spec.whatwg.org/multipage/introduction.html>. Acesso em: 27 Outubro 2019.

APÊNDICE A – IMAGENS PÁGINA HTML.

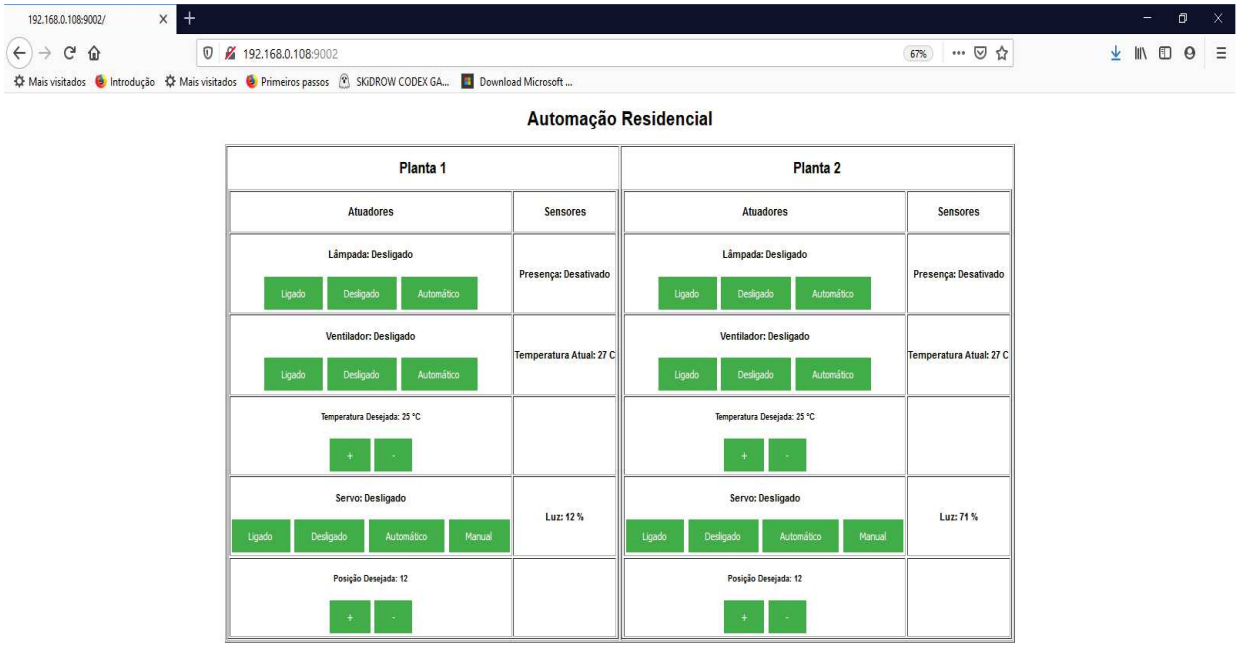
Aqui é apresentado as Figuras 39 e 40, sendo essas respectivamente a página HTML no navegador Mozilla e Mozilla Mobile.

Figura 39 – Página HTML aberta no smartphone.



Fonte: Autoria própria.

Figura 40 – Página HTML.



Fonte: Autoria própria.

APÊNDICE B – CÓDIGO CENTRAL DE AUTOMAÇÂO

Aqui é apresentado o código para a central de automação.

```
//-- Bibliotecas Utilizadas -------------------------------------------------------------
  #include <WiFi.h> //biblioteca responsavel pela comunicação Wi-Fi
  #include <esp32-hal-cpu.h> //biblioteca utilizada para alterar a velocidade de execução do
codigo
//---------------------------------------------------------------------------------------
  // Define Pinos
  #define     LED0      2           //Led Azul Disponivel no nodeMCU
//---------------------------------------------------------------------------------------

  WiFiServer  ESPServer(9001);    // Servidor criado para as plantas 1 e 2 na porta 9001
  WiFiServer  WEBServer(9002);    // Servidor criado para o codigo HTML da pagina local na
porta 9002.

  WiFiClient  ESPClient1;          // Cliente atribuidos as plantas 1 e 2, qual planta
conectada ao ESPClent depende de qual se conectar primeiro ao ESPServidor.
  WiFiClient  ESPClient2;          // Cliente atribuidos as plantas 1 e 2
  WiFiClient  WEBClient;           // Cliente atribuido aos dispositivos conectados ao site


//=======================================================================================
// Protocolo criado para os atuadores utilizando struct
  typedef struct {
    uint8_t Sta;
    uint8_t Serv;
    uint8_t Lamp;
    uint8_t Vent;
  }ProtocoloAtu;

// Protocolo criado para os sensores utilizando struct
  typedef struct {
    uint8_t Sta;
    uint8_t Temp;
    uint8_t Lumi;
    uint8_t Prese;
  }ProtocoloSen;

//Para facilitar o envio foi criado uma union utilizado o protocolo dos Atudores
  typedef union {
    ProtocoloAtu Protocolo;
    uint8_t Dados[4];
  }Datae ;

//Para facilitar o envio foi criado uma union utilizado o protocolo dos Sensores
  typedef union {
    ProtocoloSen Protocolo;
    uint8_t Dados[4];
  }Datar;


  Datae Atuadores1;//Criado uma variavel do tipo Datae que se refere a union criada para o
protocolo dos Atuadores
  Datae Atuadores2;//Criado uma variavel do tipo Datae que se refere a union criada para o
protocolo dos Atuadores
  Datar Sensores1;// Criado uma variavel do tipo Datar que se refere a union criada para o
protocolo dos Sensores
```

```
  Datar Sensores2;// Criado uma variavel do tipo Datar que se refere a union criada para o
protocolo dos Sensores


  uint32_t te1 = 0;//Variavel criada para marcar o tempo de envio de dados para o Cliente 1
  uint32_t te2 = 0;//Variavel criada para marcar o tempo de envio de dados para o Cliente 2

  uint32_t T1FPGA = 1;
  uint32_t T2FPGA = 1;
  uint8_t fp1[4] = {0,0,0,0};
  uint8_t fp2[4] = {0,0,0,0};

  uint32_t tl = 0;//Variavel criada para marcar o tempo de acionamento do Led Azul presente no
DevKitC V1.
  bool ali = true;//Variavel criada para marcar o ultimo estado do Led Azul

  //Aqui todas as variaveis foram criadas em pares, sendo uma para cada planta/cliente
  uint8_t C1VTemp = 25;//Variavel criada para salvar a temperatura desejada do cliente
recebida da pagina HTML
  uint8_t C2VTemp = 25;
  uint8_t C1VServ = 12;//Variavel criada para salvar a posição desejada do servo do cliente
recebida da pagina HTML
  uint8_t C2VServ = 12;

  uint8_t FPGA = 0;

  uint8_t C1TFPGA = 0;
  uint8_t C2TFPGA = 0;
  uint8_t C1LFPGA = 0;
  uint8_t C2LFPGA = 0;
  uint8_t C1SFPGA = 0;
  uint8_t C2SFPGA = 0;
  uint8_t C1VFPGA = 0;
  uint8_t C2VFPGA = 0;

  String C1SLamp = "Desligado";//Variavel responsavel por salvar o estado da lampada, essa
tambem é utilizado para imprimir na pagina HTML
  String C2SLamp = "Desligado";
  String C1SPres = "Desativado";//Variavel responsavel por salvar o valor do sensor PIR, essa
tambem é utilizado para imprimir na pagina HTML
  String C2SPres = "Desativado";
  String C1SVent = "Desligado";//Variavel utilizada para salvar o estado do ventilador, essa
tambem é utilizado para imprimir na pagina HTML
  String C2SVent = "Desligado";
  String C1SServ = "Desligado";//Variavel utilizada para salvar o estado do servomotor, essa
tambem é utilizado para imprimir na pagina HTML
  String C2SServ = "Desligado";
  String C1SPlanta = "Desconectado";//Variavel utilizada para marcar qual planta se conectou
ao cliente 1, essa tambem é utilizado para imprimir na pagina HTML
  String C2SPlanta = "Desconectado";

  String header;//variavel responsavel por armazem os dados recebidos da pagina local

  uint8_t t1sensor[150];
  uint8_t t2sensor[150];
  uint8_t l1sensor[150];
  uint8_t l2sensor[150];
  uint8_t w1=0;
  uint8_t w2=0;

//================================================================================
  void setup()
```

```
  {
    //Setando a velocidade de comunicação serial
    Serial.begin(115200);           // Cumunicação Serial.

    //Função que configura o ESP como roteador e ponto de acesso WiFI e cria a rede conforme o
enviado pelos 2 primeiros parametros, os outros 2 sao os da rede a ser conectada
    ConfigWifi("Alisson01", "12345678","Alisson","alisson29");

    pinMode(LED0, OUTPUT);//Configuração led azul DevKitC
    pinMode(25,INPUT);
    pinMode(26,INPUT);
    pinMode(27,INPUT);
    pinMode(32,INPUT);
    pinMode(33,INPUT);
    pinMode(34,INPUT);
    pinMode(35,INPUT);
    pinMode(36,INPUT);
    pinMode(21,INPUT);

    pinMode(1,OUTPUT);
    pinMode(3,OUTPUT);
    pinMode(4,OUTPUT);
    pinMode(5,OUTPUT);
    pinMode(12,OUTPUT);
    pinMode(13,OUTPUT);
    pinMode(14,OUTPUT);
    pinMode(15,OUTPUT);
    pinMode(16,OUTPUT);
    pinMode(17,OUTPUT);
    pinMode(18,OUTPUT);
    pinMode(19,OUTPUT);
    pinMode(22,OUTPUT);
    digitalWrite(22,HIGH);

    Atuadores1.Protocolo.Serv = 7; // Posição inicial servo.
    Atuadores2.Protocolo.Serv = 7; // Posição inicial servo.

    Serial.println("Frequencia");
    setCpuFrequencyMhz(160);
    Serial.println(getCpuFrequencyMhz());
    Serial.println(getXtalFrequencyMhz());
    Serial.println(getApbFrequency());


  }
//================================================================================
  void loop(){
    ClienteDisponivel(); //Essa função verificar se tem algum cliente ESP32 tentando se
conectar
    int t1 = 0 ;
    //int t2 = 0 ;
    //int t3 = 0 ;
    int statu1 = 0; //Variavel utilizada para verificar se o recebimento da msg do cliente 1
foi correto
    int statu2 = 0; //Variavel utilizada para verificar se o recebimento da msg do cliente 2
foi correto
//================================================================================
    if (ESPClient1 && ESPClient1.connected()){ //Verifica se tem algo atribuido a variavel
ESPCliente1 e se esta conectado.
      if(ESPClient1.available() > 0){  //Verifica se o ESPCliente1 enviou algo ao servidor.
        //t1 = micros();
        statu1 =  ESPClient1.read(Sensores1.Dados,sizeof(Sensores1.Dados));
```

```
        //t1 = micros() - t1;
        //Serial.print(t1);
        //Serial.println(" :TEMPO RECEBIMENTO CLIENTE 1");
      }
      if (statu1){ //Se o recebimento dos dados ocorreu de forma correta executa as alteraçoes
necessarias.

        fp1[0] = 1; fp1[1] = 1; fp1[2] = 1; fp1[3] = 1; //Variaveis utilizadas para
comunicação com a FPGA

        //Verifica qual planta se conectou ao ESPCliente 1 utilizando a variavel Sta
        if(Sensores1.Protocolo.Sta == 1){
          C1SPlanta = "Planta 1";
        }else if(Sensores1.Protocolo.Sta == 2){
          C1SPlanta = "Planta 2";
        }else{
          C1SPlanta = "Desconectado";
        }

        //Verifica se o PIR dectou algo, se sim salva "Ativo" na variavel utilizada pela
pagina HTML
        if(Sensores1.Protocolo.Prese == 1){
          C1SPres = "Ativo";
        }else{
          C1SPres = "Desativado";
        }

        //Se a lampada estiver configurada como automatico, verifica se sensor de presença
esta ativo e salva o status
        //de liga para ser enviado ao cliente.
        if(C1SLamp == "Automatico"){
          if(Sensores1.Protocolo.Prese == 1){
            Atuadores1.Protocolo.Lamp = 1;
          }else{
            Atuadores1.Protocolo.Lamp = 0;
          }
        }

        //Verifica se o ventilador esta no modo automatic, se estiver envia o valor obtido
atraves da FPGA ou ESP
        if(C1SVent == "Automatico"){
          Atuadores1.Protocolo.Vent = C1VFPGA;
        }

        //Verifica se o servo esta no modo automatico ou manual, se estiver no modo
automatico, envia o valor de 7 a 27,
        //conforme o obtido atraves da FPGA ou ESP, no modo manual envia direto o valor obtido
atraves da interface HTML.
        if(C1SServ == "Manual"){
          Atuadores1.Protocolo.Serv = C1VServ;
        }

        if(C1SServ == "Automatico"){
          Atuadores1.Protocolo.Serv = C1SFPGA;
        }
      }
         if(FPGA == 1){ //Verifica se é para utilizar a FPGA ou o ESP32
          if(fp1[0] == 1){ // Verifica se recebeu os dados corretamente
            if(T1FPGA == 1){ //Aqui verifica em qual etapa esta o processo de comunicação com
a FPGA

              uint32_t t = micros();//aqui é feito o envio da temperatura desejada para a FPGA
```

```
              PORTAW(C1VTemp);
              NUMW(0b000);
              (*((volatile uint32_t *) (0x3ff44008 )))|=(0x2 & (1 << 1));
              t = micros() - t;
              Serial.println(t);
              Serial.println("FPGA Tempd 1");
              (*((volatile uint32_t *) (0x3ff4400c )))|=(0x2 & (1 << 1));
              T1FPGA++;

            }else if(T1FPGA == 2){

              uint32_t t = micros();//aqui é feito o envio do valor do sensor de temperatura
para a FPGA e onde é lido a media dos ultimos 150 valores
              PORTAW(Sensores1.Protocolo.Temp);
              NUMW(0b001);
              (*((volatile uint32_t *) (0x3ff44008 )))|=(0x2 & (1 << 1));
              C1TFPGA = PORTAR();
              C1VFPGA = digitalRead(21);
              (*((volatile uint32_t *) (0x3ff4400c )))|=(0x2 & (1 << 1));
              t = micros() - t;
              Serial.println(t);
              Serial.println("FPGA Temp 1");
              T1FPGA++;

            }else if(T1FPGA == 3){

              uint32_t t = micros();//aqui é feito o envio do valor do sensor de luminosidade
para a FPGA e onde é lida a media dos ultimos 150 valores
              PORTAW(Sensores1.Protocolo.Lumi);
              NUMW(0b010);
              (*((volatile uint32_t *) (0x3ff44008 )))|=(0x2 & (1 << 1));
              C1LFPGA = PORTAR();
              (*((volatile uint32_t *) (0x3ff4400c )))|=(0x2 & (1 << 1));
              t = micros() - t;
              Serial.println(t);
              Serial.println("FPGA Lumi 1");
              T1FPGA++;

            }else if(T1FPGA == 4){

              uint32_t t = micros();//recebe a posição que o servo motor deve estabelecer
              NUMW(0b011);
              C1SFPGA = PORTAR();
              t = micros() - t;
              Serial.println(t);
              Serial.println("FPGA Serv 1");
              T1FPGA = 1;
              fp1[0] = 0;

            }
          }
        }else{
          if(fp1[0] == 1){//aqui é feito o meso processo descrito para a FPGA, so que no esp32
            t1sensor[w1] = Sensores1.Protocolo.Temp;
            l1sensor[w1] = Sensores1.Protocolo.Lumi;
            if(T1FPGA == 1){

              uint32_t t = micros();
              uint32_t soma1 = 0;
              for(int x = 0; x <=149; x++){
                soma1 = soma1 + t1sensor[x];
              }
```

```
              C1TFPGA = soma1/150;
              if(C1TFPGA > C1VTemp){
                C1VFPGA = 1;
              }else{
                C1VFPGA = 0;
              }
              t = micros() - t;
              Serial.println(t);
              Serial.println("ESP32 Temp 1");
              T1FPGA++;

            }else if(T1FPGA == 2){

              uint32_t t = micros();
              uint32_t soma2 = 0;
              for(int y = 0; y <=149; y++){
                soma2 = soma2 + l1sensor[y];
              }
              C1LFPGA = soma2/150;
              t = micros() - t;
              Serial.println(t);
              Serial.println("ESP32 Lumi 1");
              T1FPGA++;

            }else if(T1FPGA == 3){

              uint32_t t = micros();
              C1SFPGA = map(C1LFPGA, 100, 0, 7, 27);
              t = micros() - t;
              Serial.println(t);
              Serial.println("ESP32 Serv 1");
              T1FPGA = 1;
              fp1[0] = 0;

            }

          w1++;
          if(w1 == 150){
            w1 = 0;
          }
        }
      }
      if (millis() - te1 >= 400){
        //t1 = micros();
        EnviarTudo(Atuadores1.Dados,sizeof(Datae),ESPClient1);
        te1 = millis();
        //t1 = micros() - t1;
        //Serial.print(t1);
        //Serial.println(" :TEMPO ENVIO CLIENTE 1");
       }
    }
//=============================================================================================
    //O comportamento aqui é o mesmo descrito para o Cliente1, alterando somente o nome das
variaveis.
    if (ESPClient2 && ESPClient2.connected()){
      if(ESPClient2.available() > 0){
        //t1 = micros();
        statu2 =  ESPClient2.read(Sensores2.Dados,sizeof(Sensores2.Dados));
        //t1 = micros() - t1;
        //Serial.print(t1);
       //Serial.println(" :TEMPO RECEBIMENTO CLIENTE 2");
      }
```

```
if (statu2){
  fp2[0] = 1; fp2[1] = 1; fp2[2] = 1; fp2[3] = 1;

  if(Sensores2.Protocolo.Sta == 1){
    C2SPlanta = "Planta 1";
  }else if(Sensores2.Protocolo.Sta == 2){
    C2SPlanta = "Planta 2";
  }else{
    C2SPlanta = "Desconectado";
  }

  if(Sensores2.Protocolo.Prese == 1){
    C2SPres = "Ativo";
  }else{
    C2SPres = "Desativado";
  }

  if(C2SLamp == "Automatico"){
    if(Sensores2.Protocolo.Prese == 1){
      Atuadores2.Protocolo.Lamp = 1;
    }else{
      Atuadores2.Protocolo.Lamp = 0;
    }
  }

  if(C2SVent == "Automatico"){
    Atuadores2.Protocolo.Vent = C2VFPGA;
  }
  if(C2SServ == "Manual"){
    Atuadores2.Protocolo.Serv = C2VServ;
  }

  if(C2SServ == "Automatico"){
    Atuadores2.Protocolo.Serv = C2SFPGA;
  }
}

if(FPGA == 1){
  if(fp2[0] == 1){
    if(T2FPGA == 1){

      uint32_t t = micros();
      PORTAW(C2VTemp);
      NUMW(0b100);
      (*((volatile uint32_t *) (0x3ff44008 )))|=(0x2 & (1 << 1));
      t = micros() - t;
      Serial.println(t);
      Serial.println("FPGA Tempd 2");
      (*((volatile uint32_t *) (0x3ff4400c )))|=(0x2 & (1 << 1));
      T2FPGA++;

    }else if(T2FPGA == 2){

      uint32_t t = micros();
      PORTAW(Sensores2.Protocolo.Temp);
      NUMW(0b101);
      (*((volatile uint32_t *) (0x3ff44008 )))|=(0x2 & (1 << 1));
      C2TFPGA = PORTAR();
      C2VFPGA = digitalRead(21);
      (*((volatile uint32_t *) (0x3ff4400c )))|=(0x2 & (1 << 1));
      t = micros() - t;
      Serial.println(t);
```

```
          Serial.println("FPGA Temp 2");
          T2FPGA++;

        }else if(T2FPGA == 3){

          uint32_t t = micros();
          PORTAW(Sensores2.Protocolo.Lumi);
          NUMW(0b110);
          (*((volatile uint32_t *) (0x3ff44008 )))|=(0x2 & (1 << 1));
          C2LFPGA = PORTAR();
          (*((volatile uint32_t *) (0x3ff4400c )))|=(0x2 & (1 << 1));
          t = micros() - t;
          Serial.println(t);
          Serial.println("FPGA Lumi 2");
          T2FPGA++;

        }else if(T2FPGA == 4){

          uint32_t t = micros();
          NUMW(0b111);
          C2SFPGA = PORTAR();
          t = micros() - t;
          Serial.println(t);
          Serial.println("FPGA Serv 2");
          T2FPGA = 1;
          fp2[0] = 0;

        }
      }
    }else {
        if(fp2[0] == 1){
          t2sensor[w2] = Sensores2.Protocolo.Temp;
          l2sensor[w2] = Sensores2.Protocolo.Lumi;
          if(T2FPGA == 1){

            uint32_t t = micros();
            uint32_t soma3 = 0;
            for(int x = 0; x <=149; x++){
              soma3 = soma3 + t2sensor[x];
            }
            C2TFPGA = soma3/150;
            if(C2TFPGA > C2VTemp){
              C2VFPGA = 1;
            }else{
              C2VFPGA = 0;
            }
            t = micros() - t;
            Serial.println(t);
            Serial.println("ESP32 Temp 2");
            T2FPGA++;

          }else if(T2FPGA == 2){

            uint32_t t = micros();
            uint32_t soma4 = 0;
            for(int y = 0; y <=149; y++){
              soma4 = soma4 + l2sensor[y];
            }
            C2LFPGA = soma4/150;
            t = micros() - t;
            Serial.println(t);
            Serial.println("ESP32 Lumi 2");
```

```
              T2FPGA++;

            }else if(T2FPGA == 3){

              uint32_t t = micros();
              C2SFPGA = map(C2LFPGA, 100, 0, 7, 27);
              t = micros() - t;
              Serial.println(t);
              Serial.println("ESP32 Serv 2");
              T2FPGA = 1;
              fp2[0] = 0;

            }

          w2++;
          if(w2 == 150){
            w2 = 0;
          }
        }
      }
      if (millis() - te2 >= 400){
        //t1 = micros();
        EnviarTudo(Atuadores2.Dados,sizeof(Datae),ESPClient2);
        te2 = millis();
        //t1 = micros() - t1;
        //Serial.print(t1);
        //Serial.println(" :TEMPO ENVIO CLIENTE 2");
      }
    }

//*****************************************************************************************************
******
//WEBCliente
//Verifica se tem alguem tentando se conectar a pagina local.
  if (WEBServer.hasClient())
  {
    Serial.println(WEBServer.hasClient());
    //Se tem algo atribuido a veriavel WEBCliente, limpo esta.
    if (WEBClient) {
      WEBClient.stop();
    }
    //Atribui o novo cliente a variavel WEBClient
    WEBClient = WEBServer.available();
  }
  //Verifica se tem algo atribuido a variavel WEBClient
  if (WEBClient) {
    //Verifica se esta conectado
    if (WEBClient.connected()) {
      //Verifica se enviou algo ao servidor
      if (WEBClient.available() > 0) {
        //t1 = micros();
        char new_byte = WEBClient.read();//Recebe os dados enviados da pagina local
        //t1 = micros() - t1;
        //Serial.print(t1);
        //Serial.println(" :TEMPO RECEBIMENTO PAGINA HTML");
        Serial.write(new_byte);//imprimi eles na saida serial.
        header += new_byte;//salva eles na variavel header.
        if (new_byte == '\n') {//Se o byte recebido é uma quebra de linha executa uma
verificação
//=============================================================================================
=======
```

```
            //Utilizando a função indexOf, verifica se aquela string esta presença na string
header
            //se estiver executa  o if conforme foi configurado, a função indexOf retorna -1
            //quando a string não é encontrada.
//Lampada
            //Configuração da lampada
            if (header.indexOf("C1LAMP=ON") != -1)
             {
               Serial.println("Lampada Planta 1 on");
               Atuadores1.Protocolo.Lamp = 1;
               C1SLamp = "Ligado";
             }
             if (header.indexOf("C1LAMP=OFF") != -1)
             {
               Serial.println("Lampada Planta 1 off");
               Atuadores1.Protocolo.Lamp = 0;
               C1SLamp = "Desligado";
             }
             if (header.indexOf("C1LAMP=AUTO") != -1)
             {
               Serial.println("Lampada Planta 1 Automatico");
               Atuadores1.Protocolo.Lamp = 0;
               C1SLamp = "Automatico";
             }
//-------------------------------------------------------------------------------------------------
----
//Ventilador
             //Configuraçao do ventilador
             if (header.indexOf("C1VENT=ON") != -1)
             {
               Serial.println("Ventilador Planta 1 on");
               Atuadores1.Protocolo.Vent = 1;
               C1SVent = "Ligado";
             }
             if (header.indexOf("C1VENT=OFF") != -1)
             {
               Serial.println("Ventilador Planta 1 off");
               Atuadores1.Protocolo.Vent = 0;
               C1SVent = "Desligado";
             }
             if (header.indexOf("C1VENT=AUTO") != -1)
             {
               Serial.println("Ventilador Planta 1 automatico");
               Atuadores1.Protocolo.Vent = 0;
               C1SVent = "Automatico";
             }
             if (header.indexOf("C1VENT=VADD") != -1)
             {
               C1VTemp++;
             }
             if (header.indexOf("C1VENT=VSUB") != -1)
             {
               C1VTemp--;
             }
//-------------------------------------------------------------------------------------------------
----
//Servo
             //Configuração do servomotor
             if (header.indexOf("C1SERV=ON") != -1)
             {
               Serial.println("Servo ON");
               Atuadores1.Protocolo.Serv = 27;
```

```
              C1SServ = "Ligado";
            }
            if(header.indexOf("C1SERV=OFF") != -1) {
              Serial.println("Servo OFF");
              Atuadores1.Protocolo.Serv = 7;
              C1SServ = "Desligado";
            }
            if(header.indexOf("C1SERV=AUTO") != -1) {
              Serial.println("Servo 1 Automatico");
              Atuadores1.Protocolo.Serv = 7;
              C1SServ = "Automatico";
            }
            if(header.indexOf("C1SERV=MAN") != -1) {
              Serial.println("Servo 1 Manual");
              Atuadores1.Protocolo.Serv = 7;
              C1SServ = "Manual";
            }
            if(header.indexOf("C1SERV=VADD") != -1) {
              C1VServ++;
            }
            if(header.indexOf("C1SERV=VSUB") != -1) {
              C1VServ--;
            }
//=============================================================================================
//Foi feito a mesma coisa so que para o cliente 2
//Cliente 2
//Lampada
           if (header.indexOf("C2LAMP=ON") != -1)
            {
              Serial.println("Lampada Planta 2 on");
              Atuadores2.Protocolo.Lamp = 1;
              C2SLamp = "Ligado";
            }
            if (header.indexOf("C2LAMP=OFF") != -1)
            {
              Serial.println("Lampada Planta 2 off");
              Atuadores2.Protocolo.Lamp = 0;
              C2SLamp = "Desligado";
            }
            if (header.indexOf("C2LAMP=AUTO") != -1)
            {
              Serial.println("Lampada Planta 2 Automatico");
              Atuadores2.Protocolo.Lamp = 0;
              C2SLamp = "Automatico";
            }
//---------------------------------------------------------------------------------------------
----
//Ventilador
            if (header.indexOf("C2VENT=ON") != -1)
            {
              Serial.println("Ventilador Planta 2 on");
              Atuadores2.Protocolo.Vent = 1;
              C2SVent = "Ligado";
            }
            if (header.indexOf("C2VENT=OFF") != -1)
            {
              Serial.println("Ventilador Planta 2 off");
              Atuadores2.Protocolo.Vent = 0;
              C2SVent = "Desligado";
            }

            if (header.indexOf("C2VENT=AUTO") != -1)
```

```
            {
              Serial.println("Ventilador Planta 2 automatico");
              Atuadores2.Protocolo.Vent = 0;
              C2SVent = "Automatico";
            }

            if (header.indexOf("C2VENT=VADD") != -1)
            {
              C2VTemp++;
            }
            if (header.indexOf("C2VENT=VSUB") != -1)
            {
              C2VTemp--;
            }
//-------------------------------------------------------------------------------------------------
----
//Servo
            if (header.indexOf("C2SERV=ON") != -1)
            {
              Serial.println("Servo 2 ON");
              Atuadores2.Protocolo.Serv = 27;
              C2SServ = "Ligado";
            }
            if(header.indexOf("C2SERV=OFF") != -1) {
              Serial.println("Servo 2 OFF");
              Atuadores2.Protocolo.Serv = 7;
              C2SServ = "Desligado";
            }
            if(header.indexOf("C2SERV=AUTO") != -1) {
              Serial.println("Servo 2 Automatico");
              Atuadores2.Protocolo.Serv = 7;
              C2SServ = "Automatico";
            }
            if(header.indexOf("C2SERV=MAN") != -1) {
              Serial.println("Servo 2 Manual");
              Atuadores2.Protocolo.Serv = 7;
              C2SServ = "Manual";
            }
            if(header.indexOf("C2SERV=VADD") != -1) {
              C2VServ++;
            }
            if(header.indexOf("C2SERV=VSUB") != -1) {
              C2VServ--;
            }
//
==========================================================================================
====================

            header = "";//Aqui limpa a variavel header
            //t1 = micros();
            Site();//Chama a função Site(), onde esta contem a programação HTML da pagina
local
            WEBClient.stop();//Desconecta o cliente
            //t1 = micros() - t1;
            //Serial.print(t1);
            //Serial.println(" :TEMPO ENVIO PAGINA HTML");

            Serial.println("Client desconectado.");
            Serial.println("");
        }
      }
    }
```

```
  }
  //if utilizado para fazer o led azul do DevKitC piscar com um periodo de 500ms
  if(millis() - tl >= 250){
    ali = !ali;
    digitalWrite(LED0, ali);
    tl = millis();
  }

  }

//================================================================================
//Função responsavel por configurar o Servidor
  void ConfigWifi(char* Nome1, char* Senha1, char* Nome2, char* Senha2)
  {
    //Desconectada de qualquer ligação WiFI
    WiFi.disconnect();

    //Configura o modo do WiFI como ponto de acesso e estação.
    WiFi.mode(WIFI_AP_STA);
    Serial.println("WiFi.mode : AP_STA");

    //Nome da rede e senha, a ser criada.
    char* ESPssid1      = Nome1;
    char* ESPpassword1  = Senha1;

    //Iniciando o ponto de acesso
    WiFi.softAP(ESPssid1, ESPpassword1);
    Serial.println("Rede WiFi < " + String(ESPssid1) + " > Iniciada");

    //Espera 500ms
    delay(500);

    //Pegando o IP de Ponto de Acesso
    IPAddress IP = WiFi.softAPIP();

    //Imprimindo na saida serial o IP de Ponto de Acesso
    Serial.print("Endereço IP Ponto de Acesso: ");
    Serial.println(IP);

    //Imprimindo o Endereço MAC
    Serial.print("Ponto de Acesso MAC : ");
    Serial.println(String(WiFi.softAPmacAddress()));

    //Iniciando o Server
    ESPServer.begin();
    ESPServer.setNoDelay(true);
    Serial.println("ESPServer Iniciado");

    //Nome da rede e senha, a ser conectada.
    char* ESPssid2      = Nome2;
    char* ESPpassword2  = Senha2;

    WiFi.begin(ESPssid2, ESPpassword2);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("");
    Serial.println("WiFi Conectado.");
    Serial.println("Endereço IP: ");
```

```
    Serial.println(WiFi.localIP());

    WEBServer.begin();
    //WEBServer.setNoDelay(true);
    Serial.println("WEBServer Iniciado");
  }

//=============================================================================
//Função responsavel por conectar as plantas aos clientes 1 e 2
  void ClienteDisponivel()
  {
    if (ESPServer.hasClient())
    {
      if (!ESPClient1.connected() || !ESPClient1)
      {
        if(ESPClient1)
        {
          ESPClient1.stop();
        }
        if(ESPClient1 = ESPServer.available())
        {
          Serial.println("Novo Cliente: 1");
        }
      }

      if (!ESPClient2.connected() || !ESPClient2)
      {
        if(ESPClient2)
        {
          ESPClient2.stop();
        }
        if(ESPClient2 = ESPServer.available())
        {
          Serial.println("Novo Cliente: 2");
        }
      }
    }
  }

//=============================================================================
//Função responsavel por enviar os dados aos clientes.
  void EnviarTudo(uint8_t buffer[] , uint16_t tamanho , WiFiClient Envio){
    Envio.write(buffer,tamanho);
    Envio.flush();
  }
//=============================================================================
//Função responsavel por enviar a programação HTML a pagina da rede local
  void Site(){
    WEBClient.println("HTTP/1.1 200 OK");
    WEBClient.println("Content-type:text/html");
    WEBClient.println("Connection: close");
    WEBClient.println();
    WEBClient.println("<!DOCTYPE html><html>");
    WEBClient.println("<head><meta charset=\"UTF-8\" name=\"viewport\" content=\"width=device-
width, initial-scale=1\">");
    WEBClient.println("<link rel=\"icon\" href=\"data:,\">");
    WEBClient.println("<style>html {scroll-behavior: smooth; font-family: Helvetica; display:
inline-block; margin: 0px auto; text-align: center;}");
    WEBClient.println(".button { background-color: #4CAF50; border: 2px solid #4CAF50;; color:
white; padding: 15px 32px; text-align: center; text-decoration: none; display: inline-block;
font-size: 16px; margin: 4px 2px; cursor: pointer; }");
```

```
    WEBClient.println("text-decoration: none; font-size: 30px; margin: 2px; cursor:
pointer;}");

    WEBClient.println("</style></head>");
    WEBClient.println("<body><center><h1>Automação Residencial</h1></center>");

    WEBClient.println("<form><center><table border=1><tr><td><center><h2>" + C1SPlanta +
"</h2></center>");
    //WEBClient.println("<center><h2>Selecione o que deseja</h2></center>");

    WEBClient.println("<table border=1><tr><td><center><h3>Atuadores</h3></center></td>");
    WEBClient.println("<td><center><h3>Sensores</h3></center></td></tr>");

    WEBClient.println("<tr><td><center><p><h4 id=\"c1lamp\"> Lâmpada: " + C1SLamp +
"</h4></p>");
    WEBClient.println("<button onclick=\"location.href='#c1lamp';window.location.reload();\"
class=\"button\" name=\"C1LAMP\" value=\"ON\" type=\"submit\">Ligado</button>") ;
    WEBClient.println("<button onclick=\"location.href='#c1lamp';window.location.reload();\"
class=\"button\" name=\"C1LAMP\" value=\"OFF\" type=\"submit\">Desligado</button>");
    WEBClient.println("<button onclick=\"location.href='#c1lamp';window.location.reload();\"
class=\"button\" name=\"C1LAMP\" value=\"AUTO\"
type=\"submit\">Automático</button></center></td>");
    WEBClient.println("<td><center><p><h4>Presença: " + C1SPres +
"</h4></p></center></td></tr>");

    WEBClient.println("<tr><td><center><p><h4 id=\"c1vent\"> Ventilador: " + C1SVent +
"</h4></p>");
    WEBClient.println("<button onclick=\"location.href='#c1vent';window.location.reload();\"
class=\"button\" name=\"C1VENT\" value=\"ON\" type=\"submit\">Ligado</button>");
    WEBClient.println("<button onclick=\"location.href='#c1vent';window.location.reload();\"
class=\"button\" name=\"C1VENT\" value=\"OFF\" type=\"submit\">Desligado</button>");
    WEBClient.println("<button onclick=\"location.href='#c1vent';window.location.reload();\"
class=\"button\" name=\"C1VENT\" value=\"AUTO\"
type=\"submit\">Automático</button></center></td>");
    WEBClient.println((String)"<td><center><p><h4>Temperatura Atual: " + C1TFPGA + "
C</h4></p></center></td></tr>");

    WEBClient.println((String)"<tr><td><center><p><h5 id=\"c1tempd\">Temperatura Desejada: " +
C1VTemp + " °C</h5></p>");
    WEBClient.println("<button onclick=\"location.href='#c1tempd';window.location.reload();\"
class=\"button\" name=\"C1VENT\" value=\"VADD\" type=\"submit\">+</button>");
    WEBClient.println("<button onclick=\"location.href='#c1tempd';window.location.reload();\"
class=\"button\" name=\"C1VENT\" value=\"VSUB\" type=\"submit\">-</button></center></td>");
    WEBClient.println("<td></td></tr>");

    WEBClient.println("<tr><td><center><p><h4 id=\"c1servo\"> Servo: " + C1SServ +
"</h4></p>");
    WEBClient.println("<button onclick=\"location.href='#c1servo';window.location.reload();\"
class=\"button\" name=\"C1SERV\" value=\"ON\" type=\"submit\">Ligado</button>");
    WEBClient.println("<button onclick=\"location.href='#c1servo';window.location.reload();\"
class=\"button\" name=\"C1SERV\" value=\"OFF\" type=\"submit\">Desligado</button>");
    WEBClient.println("<button onclick=\"location.href='#c1servo';window.location.reload();\"
class=\"button\" name=\"C1SERV\" value=\"AUTO\" type=\"submit\">Automático</button>");
    WEBClient.println("<button onclick=\"location.href='#c1servo';window.location.reload();\"
class=\"button\" name=\"C1SERV\" value=\"MAN\"
type=\"submit\">Manual</button></center></td>");
    WEBClient.println((String)"<td><center><p><h4>Luz: " + C1LFPGA + " %
</h4></p></center></td></tr>");

    WEBClient.println((String)"<tr><td><center><p><h5 id=\"c1posid\">Posição Desejada: " +
C1VServ + "</h5></p>");
```

```
    WEBClient.println("<button onclick=\"location.href='#c1posid';window.location.reload();\"
class=\"button\" name=\"C1SERV\" value=\"VADD\" type=\"submit\">+</button>");
    WEBClient.println("<button onclick=\"location.href='#c1posid';window.location.reload();\"
class=\"button\" name=\"C1SERV\" value=\"VSUB\" type=\"submit\">-</button></center></td>");
    WEBClient.println("<td></td></tr>");

    WEBClient.println("</table></td>");

    WEBClient.println("<td><center><h2>" + C2SPlanta + "</h2></center>");
    //WEBClient.println("<center><h2>Selecione o que deseja</h2></center>");

    WEBClient.println("<table border=1><tr><td><center><h3>Atuadores</h3></center></td>");
    WEBClient.println("<td><center><h3>Sensores</h3></center></td></tr>");

    WEBClient.println("<tr><td><center><p><h4 id=\"c2lamp\"> Lâmpada: " + C2SLamp +
"</h4></p>");
    WEBClient.println("<button onclick=\"location.href='#c2lamp';window.location.reload();\"
class=\"button\" name=\"C2LAMP\" value=\"ON\" type=\"submit\">Ligado</button>") ;
    WEBClient.println("<button onclick=\"location.href='#c2lamp';window.location.reload();\"
class=\"button\" name=\"C2LAMP\" value=\"OFF\" type=\"submit\">Desligado</button>");
    WEBClient.println("<button onclick=\"location.href='#c2lamp';window.location.reload();\"
class=\"button\" name=\"C2LAMP\" value=\"AUTO\"
type=\"submit\">Automático</button></center></td>");
    WEBClient.println("<td><center><p><h4>Presença: " + C2SPres +
"</h4></p></center></td></tr>");

    WEBClient.println("<tr><td><center><p><h4 id=\"c2vent\"> Ventilador: " + C2SVent +
"</h4></p>");
    WEBClient.println("<button onclick=\"location.href='#c2vent';window.location.reload();\"
class=\"button\" name=\"C2VENT\" value=\"ON\" type=\"submit\">Ligado</button>");
    WEBClient.println("<button onclick=\"location.href='#c2vent';window.location.reload();\"
class=\"button\" name=\"C2VENT\" value=\"OFF\" type=\"submit\">Desligado</button>");
    WEBClient.println("<button onclick=\"location.href='#c2vent';window.location.reload();\"
class=\"button\" name=\"C2VENT\" value=\"AUTO\"
type=\"submit\">Automático</button></center></td>");
    WEBClient.println((String)"<td><center><p><h4>Temperatura Atual: " + C2TFPGA + "
C</h4></p></center></td></tr>");

    WEBClient.println((String)"<tr><td><center><p><h5 id=\"c2tempd\">Temperatura Desejada: " +
C2VTemp + " °C</h5></p>");
    WEBClient.println("<button onclick=\"location.href='#c2tempd';window.location.reload();\"
class=\"button\" name=\"C2VENT\" value=\"VADD\" type=\"submit\">+</button>");
    WEBClient.println("<button onclick=\"location.href='#c2tempd';window.location.reload();\"
class=\"button\" name=\"C2VENT\" value=\"VSUB\" type=\"submit\">-</button></center></td>");
    WEBClient.println("<td></td></tr>");

    WEBClient.println("<tr><td><center><p><h4 id=\"c2servo\"> Servo: " + C2SServ +
"</h4></p>");
    WEBClient.println("<button onclick=\"location.href='#c2tempd';window.location.reload();\"
class=\"button\" name=\"C2SERV\" value=\"ON\" type=\"submit\">Ligado</button>");
    WEBClient.println("<button onclick=\"location.href='#c2tempd';window.location.reload();\"
class=\"button\" name=\"C2SERV\" value=\"OFF\" type=\"submit\">Desligado</button>");
    WEBClient.println("<button onclick=\"location.href='#c2tempd';window.location.reload();\"
class=\"button\" name=\"C2SERV\" value=\"AUTO\" type=\"submit\">Automático</button>");
    WEBClient.println("<button onclick=\"location.href='#c2tempd';window.location.reload();\"
class=\"button\" name=\"C2SERV\" value=\"MAN\"
type=\"submit\">Manual</button></center></td>");
    WEBClient.println((String)"<td><center><p><h4>Luz: " + C2LFPGA + " %
</h4></p></center></td></tr>");

    WEBClient.println((String)"<tr><td><center><p><h5 id=\"c2posid\">Posição Desejada: " +
C2VServ + "</h5></p>");
```

```
    WEBClient.println("<button onclick=\"location.href='#c2posid';window.location.reload();\"
class=\"button\" name=\"C2SERV\" value=\"VADD\" type=\"submit\">+</button>");
    WEBClient.println("<button onclick=\"location.href='#c2posid';window.location.reload();\"
class=\"button\" name=\"C2SERV\" value=\"VSUB\" type=\"submit\">-</button></center></td>");
    WEBClient.println("<td></td></tr>");

    WEBClient.println("</table></td></tr></table></center></form>");

WEBClient.println("<script>setTimeout(function(){window.location.reload();window.location=['//
', location.host, location.hash].join('');},5000);</script>");
    WEBClient.println("</body></html>");
    WEBClient.println();
  }

 void NUMW(uint8_t  dados){
    digitalWrite(3,(0b00000001 & dados));
    digitalWrite(4,(0b00000010 & dados) >> 1);
    digitalWrite(5,(0b00000100 & dados) >> 2);
 }

 void PORTAW(uint8_t  dados){
    digitalWrite(12,(0b00000001 & dados));
    digitalWrite(13,(0b00000010 & dados) >> 1);
    digitalWrite(14,(0b00000100 & dados) >> 2);
    digitalWrite(15,(0b00001000 & dados) >> 3);
    digitalWrite(16,(0b00010000 & dados) >> 4);
    digitalWrite(17,(0b00100000 & dados) >> 5);
    digitalWrite(18,(0b01000000 & dados) >> 6);
    digitalWrite(19,(0b10000000 & dados) >> 7);
 }
  uint8_t PORTAR(){
    uint8_t aux = 0;
    aux = aux + digitalRead(25);
    aux = aux + (2 * digitalRead(26));
    aux = aux + (4 * digitalRead(27));
    aux = aux + (8 * digitalRead(32));
    aux = aux + (16 * digitalRead(33));
    aux = aux + (32 * digitalRead(34));
    aux = aux + (64 * digitalRead(35));
    aux = aux + (128 * digitalRead(36));
}
```

APÊNDICE C – CÓDIGO UTILIZADO PELAS PLANTAS

Aqui é apresentado o código para as plantas 1 e 2.

```
//--------------------------------------------------------------------------------------
// Bibliotecas utilizadas
//--------------------------------------------------------------------------------------

  #include <WiFi.h>    // Biblioteca utilizada pelo WiFi
  #include <esp_adc_cal.h> // Biblioteca utilizada para calibrar o ADC

//--------------------------------------------------------------------------------------
// Definindo os pinos I/O
//--------------------------------------------------------------------------------------
  #define       LED0      2         //Led presente na placa
  #define       TP        34        //Pino utilizado pelo sensor de temperatura
  #define       LUZ       35        //Pino utilizado pelo sensor de luz
  #define       PWM       15        //Pino utilizado pelo servo
  #define       PRE       27        //Pino utilizado pelo sensor de presença
  #define       RLAMP     14        //Pino utilizado para ligar o rele da lampada
  #define       RVENT     12        //Pino utilizado para ligar o rele do ventilador
//--------------------------------------------------------------------------------------
//Variaveis da rede WIFI
//--------------------------------------------------------------------------------------
  char ESPssid[] = "Alisson01";        // Nome do Wifi para se conectar
  char ESPsenha[] = "12345678";        // Senha da Rede
//--------------------------------------------------------------------------------------
// Ip e porta utilizados para se conectar ao servidor ip 192.168.4.1 e porta 9001
//--------------------------------------------------------------------------------------
  int             ESPServerPort  = 9001;
  IPAddress       ESPServer(192,168,4,1);
  WiFiClient      ESPClient; // Variavel para atribuir a conexao
//======================================================================================
// Protocolo criado para os atuadores utilizando struct
  typedef struct {
    uint8_t Sta;
    uint8_t Serv;
    uint8_t Lamp;
    uint8_t Vent;
  }ProtocoloAtu;

// Protocolo criado para os sensores utilizando struct
  typedef struct {
    uint8_t Sta;
    uint8_t Temp;
    uint8_t Lumi;
    uint8_t Prese;
  }ProtocoloSen;

//Para facilitar o envio foi criado uma union utilizado o protocolo dos Atudores
  typedef union {
    ProtocoloAtu Protocolo;
    uint8_t Dados[4];
  }Datae ;

//Para facilitar o envio foi criado uma union utilizado o protocolo dos Sensores
  typedef union {
    ProtocoloSen Protocolo;
    uint8_t Dados[4];
```

```
  }Datar;

  Datae Atuadores;//Criado uma variavel do tipo Datae que se refere a union criada para o
protocolo dos Atuadores
  Datar Sensores;// Criado uma variavel do tipo Datar que se refere a union criada para o
protocolo dos Sensores

  uint32_t te = 0;
  uint32_t tr = 0;
  uint32_t tl = 0;
  bool ali = true;

//Thermistor
//Coeficientes A,B e C para o metodo de Steinhart e Hart
  float A = 0.0006681896472440;
  float B = 0.0002918466107810;
  float C = -0.0000000027899501;

//definindo a voltagem de referencia como 1121 e criando uma variavel do tipo
esp_adc_cal_characteristics_t para salvar as configuraçoes do adc
  #define REF_VOLTAGE 1121
  esp_adc_cal_characteristics_t adc_chars;

//Variaveis de configuração do pwm
  uint8_t freq = 50;
  uint8_t canal = 0;
  uint8_t resol = 8;

//================================================================================

  void setup()
  {
    //Configuração Serial ------------------------------------------------------------
    Serial.begin(115200);

    //PWM Configuração ---------------------------------------------------------------
    ledcSetup(canal , freq, resol);
    ledcAttachPin(PWM, canal);
    ledcWrite(canal, 10);

    //Configurando o modo de operaçao dos pinos --------------------------------------
    pinMode(PRE, INPUT); //pino sensor de prensença configurado como entrada
    pinMode(LED0, OUTPUT);  //pino do led presente da placa como saida
    pinMode(RLAMP, OUTPUT);  //pino do rele da lampanda como saida
    pinMode(RVENT, OUTPUT);  //pino do rele do ventilador como saida
    digitalWrite(LED0, !LOW);  //desliga o led interno da placa

    //Configuração ADC ---------------------------------------------------------------
    adc1_config_width(ADC_WIDTH_BIT_12);//definindo a resoluçao de leitura para 12bits
    adc1_config_channel_atten(ADC1_CHANNEL_6,ADC_ATTEN_DB_11);//definindo a atenuação do ADC 1
Canal 6 para 11db podendo fazer leitura até 3.9 V, mas a melhor a faixa é 150 a 2450 mV
    adc1_config_channel_atten(ADC1_CHANNEL_7,ADC_ATTEN_DB_11);//definindo a atenuação do ADC 1
Canal 6 para 11db podendo fazer leitura até 3.9 V, mas a melhor a faixa é 150 a 2450 mV
    esp_adc_cal_value_t adc_type = esp_adc_cal_characterize(ADC_UNIT_1, ADC_ATTEN_DB_11,
ADC_WIDTH_BIT_12, REF_VOLTAGE, &adc_chars);//Salva as Configurações na variavel adc_chars
    if (adc_type == ESP_ADC_CAL_VAL_EFUSE_VREF)// esse if verifica se o efuse ja veio
configurado de fabrica ou foi utilizado algum outro metodo de configuração
    {
      Serial.println("ADC CAL ref eFuse encontrado: ");
      Serial.println(adc_chars.vref);
    }
     else if (adc_type == ESP_ADC_CAL_VAL_EFUSE_TP)
```

```
    {
      Serial.println("ADC CAL Two Point eFuse encontrado");
    }
     else
    {
      Serial.println("ADC CAL Nada encontrado, utilizando Vref padrao: ");
      Serial.println(adc_chars.vref);
    }

    //Fazendo a conexão ------------------------------------------------
    if(WiFi.status() == WL_CONNECTED) //Pergunta se está conectado em uma rede wifi, se sim
disconecta e desliga o wifi e aguarda 50 ms
    {
      WiFi.disconnect();
      WiFi.mode(WIFI_OFF);
      delay(50);
    }

    WiFi.mode(WIFI_STA);              //Configura o modo de trabalho do WiFI para
Station(Estação)

    //Conectando ao Wifi-----------------------------------------------------
    ConectarWifi();

    //Conectando ao Servidor -------------------------------------
    ESPConectar();
    Sensores.Protocolo.Sta = 2; //1 para planta 1 e 2 para a planta 2
  }

//=============================================================================================

  void loop()
  {
    uint32_t t = micros();
    uint32_t t2, t3;
    int h1 = 0;
    int h2 = 0;
    float V = avgAnalogRead(TP,50);//O valor retornado é a media das ultimas 50 leituras do
ADC do pino Ac do NTC.
    float R = (10000.00*V)/(3300.00 - V);//Convertendo o valor de tensao obtido em resistencia
para ser utilizado na formula de Steinhart & Hart.
    float temp = 1.00 / (A + (B*(log(R))) + (C*(pow(log(R),3)))); //Impremetação da função de
Steinhart & Hart.
    temp = temp - 273;//Converte de Kelvin para Celsius.

    Sensores.Protocolo.Temp = (uint8_t)temp; //Salva o valor de temperatura na variavel global
de envio.
    Sensores.Protocolo.Lumi = map(avgAnalogRead(LUZ,50), 350, 3100, 100 , 0); //Faz a leitura
do pino Ac do LDR e a media das 50 ultimas leituras
                                                                              //e sua
normalização para ser apresentado em % de luminosidade.
    Sensores.Protocolo.Prese = digitalRead(PRE);//Leitura do sinal enviado pelo sensor PIR.

    int statu;//Variavel utilizada para verificar se o recebimento foi bem sucedido.

    if (ESPClient && ESPClient.connected())//Verifica se esta conectado ao WiFi e conectado ao
Servidor, chamado aqui de ESPClient
    {
      if(ESPClient.available() > 0){ //Verifica se tem dados disponiveis para leitura no
buffer WiFi
        t2 = micros();
```

```
        statu = ESPClient.read(Atuadores.Dados, sizeof(Atuadores.Dados));//Recebe os dados dos
atuadores disponiveis, status recebe 1 se for bem sucedido.
        t2 = micros() - t2;
        h1 = 1;
      }
      if (statu){//Caso o recebimento seja bem sucedido altera os valores dos atuadores
conforme o recebido.
        if(Atuadores.Protocolo.Lamp > 0){ digitalWrite(RLAMP, HIGH);} else {
digitalWrite(RLAMP, LOW); }
        if(Atuadores.Protocolo.Vent > 0){ digitalWrite(RVENT, HIGH);} else {
digitalWrite(RVENT, LOW); }
        ledcWrite(canal, Atuadores.Protocolo.Serv);
      }

      if (millis() - te >= 400){//Envia os dados para o Servidor de 400 em 400 ms;
        t3 = micros();
        EnviarTudo(Sensores.Dados,sizeof(Sensores.Dados),ESPClient);
        t3 = micros() - t3;
        h2 = 1;
        te = millis();
      }
    }

    if(millis() - tl >= 250){//Faz o led azul disponivel no ESP32 piscar cin i periodo de
500ms(2hz)
      ali = !ali;
      digitalWrite(LED0, ali);
      tl = millis();
    }

   if(!ESPClient.connected() || !ESPClient){//Verificar se esta conectado ao WiFi e ao
Servidor, se não reinicia os dois e reconecta
      ConectarWifi();
      ESPConectar();
    }

   t = micros() - t;
  if(h2 == 1){
    Serial.print(t3);
    Serial.println(" :TEMPO ENVIO");
  }
  if(h1 == 1){
    Serial.print(t2);
    Serial.println(" :TEMPO RECEBIMENTO");
  }
  Serial.print(t);
  Serial.println(" :TEMPO EXECUÇÃO CODIGO");
  Serial.println("");
  }

//=============================================================================================

  void ConectarWifi() //Função responsavel por reiniciar o WiFi do ESP32 Client e reconectar
se necessario.
  {
    if(WiFi.status() == WL_CONNECTED)//Verifica se o WiFI esta ligado, se estiver desliga
    {
      WiFi.disconnect();
      WiFi.mode(WIFI_OFF);
    }

    WiFi.mode(WIFI_STA); //Liga o WiFI no modo client
```

```
    while(WiFi.status() != WL_CONNECTED) // Enquanto nao estiver conectado, ele fica tentando
se conectar a rede WiFi fornecida
    {
      WiFi.begin(ESPssid,ESPsenha);//Se conecta a rede WiFi fornecida, caso não esteja
disponivel, ele ficara preso nesse loop.
      for(int i=0; i < 10; i++)
      {
        digitalWrite(LED0, !HIGH);
        delay(250);
        digitalWrite(LED0, !LOW);
        delay(250);
        Serial.print(".");
      }
      Serial.println("");
      digitalWrite(LED0, !HIGH);
    }
    Serial.println("Conectado a rede Wifi: " + String(WiFi.SSID()));
    Serial.println("Intensidade do Sinal   : " + String(WiFi.RSSI()) + " dBm");
    Serial.print  ("Endereço Mac : ");
    Serial.println(String(WiFi.macAddress()));
    Serial.print  ("Endereço Ip Local : ");
    Serial.println(WiFi.localIP());
  }

  void ESPConectar()//Função responsavel em conectar o cliente ao servidor.
  {
    ESPClient.stop();//Desconecta a variavel tipo WiFi Cliente, caso essa esteja conectada.
    Serial.println("Desconectado");
    delay(500);
    if(ESPClient.connect(ESPServer, ESPServerPort))//Faz a conexao com o servidor via socket,
atraves do IP e Porta fornecidos.
      {
      Serial.println    ("<Conectado>");
      }
    Serial.print  ("Endereço IP Servidor : ");
    Serial.println(ESPServer);
    Serial.print  ("Numero da porta do Servidor  : ");
    Serial.println(ESPServerPort);
 }

//================================================================================
  void EnviarTudo(uint8_t buffer[] , uint16_t tamanho , WiFiClient Envio){//Função responsavel
por enviar os dados armazenados na variavel global de envio.
    Envio.write(buffer,tamanho);//Envia uma varivel do especificado via WiFi
    Envio.flush();//Limpa o buffer de envio
  }
//================================================================================

uint16_t avgAnalogRead(uint8_t pin, uint16_t samples) {//Função responsavel pela leitura do
ADC e fazer a media das ultimas 50 leituras
  adc1_channel_t chan;
  switch (pin) {
    case 32:
      chan = ADC1_CHANNEL_4;
    break;
    case 33:
      chan = ADC1_CHANNEL_5;
    break;
    case 34:
      chan = ADC1_CHANNEL_6;
    break;
```

```
    case 35:
      chan = ADC1_CHANNEL_7;
    break;
    case 36:
      chan = ADC1_CHANNEL_3;
    break;
    case 39:
      chan = ADC1_CHANNEL_0;
    break;
  }
  uint32_t sum = 0;
  for (int x=0; x<samples; x++) {
    sum += adc1_get_raw(chan);
  }
  sum /= samples;
  return esp_adc_cal_raw_to_voltage(sum, &adc_chars);//------------------------------------
-------------------------------------------
// Bibliotecas utilizadas
//--------------------------------------------------------------------------------------

  #include <WiFi.h>    // Biblioteca utilizada pelo WiFi
  #include <esp_adc_cal.h> // Biblioteca utilizada para calibrar o ADC

//--------------------------------------------------------------------------------------
// Definindo os pinos I/O
//--------------------------------------------------------------------------------------
  #define       LED0      2         //Led presente na placa
  #define       TP        34        //Pino utilizado pelo sensor de temperatura
  #define       LUZ       35        //Pino utilizado pelo sensor de luz
  #define       PWM       15        //Pino utilizado pelo servo
  #define       PRE       27        //Pino utilizado pelo sensor de presença
  #define       RLAMP     14        //Pino utilizado para ligar o rele da lampada
  #define       RVENT     12        //Pino utilizado para ligar o rele do ventilador
//--------------------------------------------------------------------------------------
//Variaveis da rede WIFI
//--------------------------------------------------------------------------------------
  char ESPssid[] = "Alisson01";        // Nome do Wifi para se conectar
  char ESPsenha[] = "12345678";        // Senha da Rede
//--------------------------------------------------------------------------------------
// Ip e porta utilizados para se conectar ao servidor ip 192.168.4.1 e porta 9001
//--------------------------------------------------------------------------------------
  int             ESPServerPort  = 9001;
  IPAddress       ESPServer(192,168,4,1);
  WiFiClient      ESPClient; // Variavel para atribuir a conexao
//======================================================================================
// Protocolo criado para os atuadores utilizando struct
  typedef struct {
    uint8_t Sta;
    uint8_t Serv;
    uint8_t Lamp;
    uint8_t Vent;
  }ProtocoloAtu;

// Protocolo criado para os sensores utilizando struct
  typedef struct {
    uint8_t Sta;
    uint8_t Temp;
    uint8_t Lumi;
    uint8_t Prese;
  }ProtocoloSen;

//Para facilitar o envio foi criado uma union utilizado o protocolo dos Atudores
```

```
  typedef union {
    ProtocoloAtu Protocolo;
    uint8_t Dados[4];
  }Datae ;

//Para facilitar o envio foi criado uma union utilizado o protocolo dos Sensores
  typedef union {
    ProtocoloSen Protocolo;
    uint8_t Dados[4];
  }Datar;

  Datae Atuadores;//Criado uma variavel do tipo Datae que se refere a union criada para o
protocolo dos Atuadores
  Datar Sensores;// Criado uma variavel do tipo Datar que se refere a union criada para o
protocolo dos Sensores

  uint32_t te = 0;
  uint32_t tr = 0;
  uint32_t tl = 0;
  bool ali = true;

//Thermistor
//Coeficientes A,B e C para o metodo de Steinhart e Hart
  float A = 0.0006681896472440;
  float B = 0.0002918466107810;
  float C = -0.0000000027899501;

//definindo a voltagem de referencia como 1121 e criando uma variavel do tipo
esp_adc_cal_characteristics_t para salvar as configuraçoes do adc
  #define REF_VOLTAGE 1121
  esp_adc_cal_characteristics_t adc_chars;

//Variaveis de configuração do pwm
  uint8_t freq = 50;
  uint8_t canal = 0;
  uint8_t resol = 8;

//============================================================================================

  void setup()
  {
    //Configuração Serial ------------------------------------------------------------------
    Serial.begin(115200);

    //PWM Configuração ---------------------------------------------------------------------
    ledcSetup(canal , freq, resol);
    ledcAttachPin(PWM, canal);
    ledcWrite(canal, 10);

    //Configurando o modo de operaçao dos pinos --------------------------------------------
    pinMode(PRE, INPUT); //pino sensor de prensença configurado como entrada
    pinMode(LED0, OUTPUT);  //pino do led presente da placa como saida
    pinMode(RLAMP, OUTPUT);  //pino do rele da lampanda como saida
    pinMode(RVENT, OUTPUT);  //pino do rele do ventilador como saida
    digitalWrite(LED0, !LOW);  //desliga o led interno da placa

    //Configuração ADC ---------------------------------------------------------------------
    adc1_config_width(ADC_WIDTH_BIT_12);//definindo a resoluçao de leitura para 12bits
    adc1_config_channel_atten(ADC1_CHANNEL_6,ADC_ATTEN_DB_11);//definindo a atenuação do ADC 1
Canal 6 para 11db podendo fazer leitura até 3.9 V, mas a melhor a faixa é 150 a 2450 mV
    adc1_config_channel_atten(ADC1_CHANNEL_7,ADC_ATTEN_DB_11);//definindo a atenuação do ADC 1
Canal 6 para 11db podendo fazer leitura até 3.9 V, mas a melhor a faixa é 150 a 2450 mV
```

```
    esp_adc_cal_value_t adc_type = esp_adc_cal_characterize(ADC_UNIT_1, ADC_ATTEN_DB_11,
ADC_WIDTH_BIT_12, REF_VOLTAGE, &adc_chars);//Salva as Configurações na variavel adc_chars
    if (adc_type == ESP_ADC_CAL_VAL_EFUSE_VREF)// esse if verifica se o efuse ja veio
configurado de fabrica ou foi utilizado algum outro metodo de configuração
    {
      Serial.println("ADC CAL ref eFuse encontrado: ");
      Serial.println(adc_chars.vref);
    }
     else if (adc_type == ESP_ADC_CAL_VAL_EFUSE_TP)
    {
      Serial.println("ADC CAL Two Point eFuse encontrado");
    }
     else
    {
      Serial.println("ADC CAL Nada encontrado, utilizando Vref padrao: ");
      Serial.println(adc_chars.vref);
    }

    //Fazendo a conexão ---------------------------------------------------
    if(WiFi.status() == WL_CONNECTED) //Pergunta se está conectado em uma rede wifi, se sim
disconecta e desliga o wifi e aguarda 50 ms
    {
      WiFi.disconnect();
      WiFi.mode(WIFI_OFF);
      delay(50);
    }

    WiFi.mode(WIFI_STA);              //Configura o modo de trabalho do WiFI para
Station(Estação)

    //Conectando ao Wifi---------------------------------------------------
    ConectarWifi();

    //Conectando ao Servidor ----------------------------------
    ESPConectar();
    Sensores.Protocolo.Sta = 2; //1 para planta 1 e 2 para a planta 2
  }

//============================================================================================

  void loop()
  {
    uint32_t t = micros();
    uint32_t t2, t3;
    int h1 = 0;
    int h2 = 0;
    float V = avgAnalogRead(TP,50);//O valor retornado é a media das ultimas 50 leituras do
ADC do pino Ac do NTC.
    float R = (10000.00*V)/(3300.00 - V);//Convertendo o valor de tensao obtido em resistencia
para ser utilizado na formula de Steinhart & Hart.
    float temp = 1.00 / (A + (B*(log(R))) + (C*(pow(log(R),3)))); //Impremetação da função de
Steinhart & Hart.
    temp = temp - 273;//Converte de Kelvin para Celsius.

    Sensores.Protocolo.Temp = (uint8_t)temp; //Salva o valor de temperatura na variavel global
de envio.
    Sensores.Protocolo.Lumi = map(avgAnalogRead(LUZ,50), 350, 3100, 100 , 0); //Faz a leitura
do pino Ac do LDR e a media das 50 ultimas leituras
                                                                              //e sua
normalização para ser apresentado em % de luminosidade.
    Sensores.Protocolo.Prese = digitalRead(PRE);//Leitura do sinal enviado pelo sensor PIR.
```

```
    int statu;//Variavel utilizada para verificar se o recebimento foi bem sucedido.

    if (ESPClient && ESPClient.connected())//Verifica se esta conectado ao WiFi e conectado ao
Servidor, chamado aqui de ESPClient
    {
      if(ESPClient.available() > 0){ //Verifica se tem dados disponiveis para leitura no
buffer WiFi
        t2 = micros();
        statu = ESPClient.read(Atuadores.Dados, sizeof(Atuadores.Dados));//Recebe os dados dos
atuadores disponiveis, status recebe 1 se for bem sucedido.
        t2 = micros() - t2;
        h1 = 1;
      }
      if (statu){//Caso o recebimento seja bem sucedido altera os valores dos atuadores
conforme o recebido.
        if(Atuadores.Protocolo.Lamp > 0){ digitalWrite(RLAMP, HIGH);} else {
digitalWrite(RLAMP, LOW); }
        if(Atuadores.Protocolo.Vent > 0){ digitalWrite(RVENT, HIGH);} else {
digitalWrite(RVENT, LOW); }
        ledcWrite(canal, Atuadores.Protocolo.Serv);
      }

      if (millis() - te >= 400){//Envia os dados para o Servidor de 400 em 400 ms;
        t3 = micros();
        EnviarTudo(Sensores.Dados,sizeof(Sensores.Dados),ESPClient);
        t3 = micros() - t3;
        h2 = 1;
        te = millis();
      }
    }

    if(millis() - tl >= 250){//Faz o led azul disponivel no ESP32 piscar cin i periodo de
500ms(2hz)
      ali = !ali;
      digitalWrite(LED0, ali);
      tl = millis();
    }

   if(!ESPClient.connected() || !ESPClient){//Verificar se esta conectado ao WiFi e ao
Servidor, se não reinicia os dois e reconecta
      ConectarWifi();
      ESPConectar();
    }

   t = micros() - t;
  if(h2 == 1){
    Serial.print(t3);
    Serial.println(" :TEMPO ENVIO");
  }
  if(h1 == 1){
    Serial.print(t2);
    Serial.println(" :TEMPO RECEBIMENTO");
  }
  Serial.print(t);
  Serial.println(" :TEMPO EXECUÇÃO CODIGO");
  Serial.println("");
  }

//=======================================================================================

  void ConectarWifi() //Função responsavel por reiniciar o WiFi do ESP32 Client e reconectar
se necessario.
```

```
  {
    if(WiFi.status() == WL_CONNECTED)//Verifica se o WiFI esta ligado, se estiver desliga
    {
      WiFi.disconnect();
      WiFi.mode(WIFI_OFF);
    }

    WiFi.mode(WIFI_STA); //Liga o WiFI no modo client

    while(WiFi.status() != WL_CONNECTED) // Enquanto nao estiver conectado, ele fica tentando
se conectar a rede WiFi fornecida
    {
      WiFi.begin(ESPssid,ESPsenha);//Se conecta a rede WiFi fornecida, caso não esteja
disponivel, ele ficara preso nesse loop.
      for(int i=0; i < 10; i++)
      {
        digitalWrite(LED0, !HIGH);
        delay(250);
        digitalWrite(LED0, !LOW);
        delay(250);
        Serial.print(".");
      }
      Serial.println("");
      digitalWrite(LED0, !HIGH);
    }
    Serial.println("Conectado a rede Wifi: " + String(WiFi.SSID()));
    Serial.println("Intensidade do Sinal   : " + String(WiFi.RSSI()) + " dBm");
    Serial.print  ("Endereço Mac : ");
    Serial.println(String(WiFi.macAddress()));
    Serial.print  ("Endereço Ip Local : ");
    Serial.println(WiFi.localIP());
  }

  void ESPConectar()//Função responsavel em conectar o cliente ao servidor.
  {
    ESPClient.stop();//Desconecta a variavel tipo WiFi Cliente, caso essa esteja conectada.
    Serial.println("Desconectado");
    delay(500);
    if(ESPClient.connect(ESPServer, ESPServerPort))//Faz a conexao com o servidor via socket,
atraves do IP e Porta fornecidos.
      {
      Serial.println    ("<Conectado>");
      }
    Serial.print  ("Endereço IP Servidor : ");
    Serial.println(ESPServer);
    Serial.print  ("Numero da porta do Servidor  : ");
    Serial.println(ESPServerPort);
 }

//================================================================================
  void EnviarTudo(uint8_t buffer[] , uint16_t tamanho , WiFiClient Envio){//Função responsavel
por enviar os dados armazenados na variavel global de envio.
    Envio.write(buffer,tamanho);//Envia uma varivel do especificado via WiFi
    Envio.flush();//Limpa o buffer de envio
  }
//================================================================================

uint16_t avgAnalogRead(uint8_t pin, uint16_t samples) {//Função responsavel pela leitura do
ADC e fazer a media das ultimas 50 leituras
  adc1_channel_t chan;
  switch (pin) {
    case 32:
```

```
      chan = ADC1_CHANNEL_4;
    break;
    case 33:
      chan = ADC1_CHANNEL_5;
    break;
    case 34:
      chan = ADC1_CHANNEL_6;
    break;
    case 35:
      chan = ADC1_CHANNEL_7;
    break;
    case 36:
      chan = ADC1_CHANNEL_3;
    break;
    case 39:
      chan = ADC1_CHANNEL_0;
    break;
  }
  uint32_t sum = 0;
  for (int x=0; x<samples; x++) {
    sum += adc1_get_raw(chan);
  }
  sum /= samples;
  return esp_adc_cal_raw_to_voltage(sum, &adc_chars);
}
```

APÊNDICE D – CÓDIGO HTML

Aqui é apresentado o código HTML da página local.

```
<!DOCTYPE html><html>
<head><meta charset="UTF-8" name="viewport" content="width=device-width,
initial-scale=1">
<link rel="icon" href="data:,">
<style>html {scroll-behavior: smooth; font-family: Helvetica; display:
inline-block; margin: 0px auto; text-align: center;}
.button { background-color: #4CAF50; border: 2px solid #4CAF50;; color:
white; padding: 15px 32px; text-align: center; text-decoration: none;
display: inline-block; font-size: 16px; margin: 4px 2px; cursor: pointer; }
text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}
</style></head>
<body><center><h1>Automação Residencial</h1></center>
<form><center><table border=1><tr><td><center><h2>Planta 2</h2></center>
<table border=1><tr><td><center><h3>Atuadores</h3></center></td>
<td><center><h3>Sensores</h3></center></td></tr>
<tr><td><center><p><h4 id="c1lamp"> Lâmpada: Desligado</h4></p>
<button onclick="location.href='#c1lamp';window.location.reload();"
class="button" name="C1LAMP" value="ON" type="submit">Ligado</button>
<button onclick="location.href='#c1lamp';window.location.reload();"
class="button" name="C1LAMP" value="OFF" type="submit">Desligado</button>
<button onclick="location.href='#c1lamp';window.location.reload();"
class="button" name="C1LAMP" value="AUTO"
type="submit">Automático</button></center></td>
<td><center><p><h4>Presença: Desativado</h4></p></center></td></tr>
<tr><td><center><p><h4 id="c1vent"> Ventilador: Desligado</h4></p>
<button onclick="location.href='#c1vent';window.location.reload();"
class="button" name="C1VENT" value="ON" type="submit">Ligado</button>
<button onclick="location.href='#c1vent';window.location.reload();"
class="button" name="C1VENT" value="OFF" type="submit">Desligado</button>
<button onclick="location.href='#c1vent';window.location.reload();"
class="button" name="C1VENT" value="AUTO"
type="submit">Automático</button></center></td>
<td><center><p><h4>Temperatura Atual: 29 C</h4></p></center></td></tr>
<tr><td><center><p><h5 id="c1tempd">Temperatura Desejada: 25 °C</h5></p>
<button onclick="location.href='#c1tempd';window.location.reload();"
class="button" name="C1VENT" value="VADD" type="submit">+</button>
<button onclick="location.href='#c1tempd';window.location.reload();"
class="button" name="C1VENT" value="VSUB" type="submit">-
</button></center></td>
<td></td></tr>
<tr><td><center><p><h4 id="c1servo"> Servo: Desligado</h4></p>
<button onclick="location.href='#c1servo';window.location.reload();" class="button"
name="C1SERV" value="ON" type="submit">Ligado</button>
<button onclick="location.href='#c1servo';window.location.reload();" class="button"
name="C1SERV" value="OFF" type="submit">Desligado</button>
```

```
<button onclick="location.href='#c1servo';window.location.reload();" class="button"
name="C1SERV" value="AUTO" type="submit">Automático</button>
<button onclick="location.href='#c1servo';window.location.reload();" class="button"
name="C1SERV" value="MAN" type="submit">Manual</button></center></td>
<td><center><p><h4>Luz: 71 % </h4></p></center></td></tr>
<tr><td><center><p><h5 id="c1posid">Posição Desejada: 12</h5></p>
<button onclick="location.href='#c1posid';window.location.reload();" class="button"
name="C1SERV" value="VADD" type="submit">+</button>
<button onclick="location.href='#c1posid';window.location.reload();" class="button"
name="C1SERV" value="VSUB" type="submit">-</button></center></td>
<td></td></tr>
</table></td>
<td><center><h2>Planta 1</h2></center>
<table border=1><tr><td><center><h3>Atuadores</h3></center></td>
<td><center><h3>Sensores</h3></center></td></tr>
<tr><td><center><p><h4 id="c2lamp"> Lâmpada: Desligado</h4></p>
<button onclick="location.href='#c2lamp';window.location.reload();" class="button"
name="C2LAMP" value="ON" type="submit">Ligado</button>
<button onclick="location.href='#c2lamp';window.location.reload();" class="button"
name="C2LAMP" value="OFF" type="submit">Desligado</button>
<button onclick="location.href='#c2lamp';window.location.reload();" class="button"
name="C2LAMP" value="AUTO" type="submit">Automático</button></center></td>
<td><center><p><h4>Presença: Desativado</h4></p></center></td></tr>
<tr><td><center><p><h4 id="c2vent"> Ventilador: Desligado</h4></p>
<button onclick="location.href='#c2vent';window.location.reload();" class="button"
name="C2VENT" value="ON" type="submit">Ligado</button>
<button onclick="location.href='#c2vent';window.location.reload();" class="button"
name="C2VENT" value="OFF" type="submit">Desligado</button>
<button onclick="location.href='#c2vent';window.location.reload();" class="button"
name="C2VENT" value="AUTO" type="submit">Automático</button></center></td>
<td><center><p><h4>Temperatura Atual: 28 C</h4></p></center></td></tr>
<tr><td><center><p><h5 id="c2tempd">Temperatura Desejada: 25 °C</h5></p>
<button onclick="location.href='#c2tempd';window.location.reload();" class="button"
name="C2VENT" value="VADD" type="submit">+</button>
<button onclick="location.href='#c2tempd';window.location.reload();" class="button"
name="C2VENT" value="VSUB" type="submit">-</button></center></td>
<td></td></tr>
<tr><td><center><p><h4 id="c2servo"> Servo: Desligado</h4></p>
<button onclick="location.href='#c2tempd';window.location.reload();" class="button"
name="C2SERV" value="ON" type="submit">Ligado</button>
<button onclick="location.href='#c2tempd';window.location.reload();" class="button"
name="C2SERV" value="OFF" type="submit">Desligado</button>
<button onclick="location.href='#c2tempd';window.location.reload();" class="button"
name="C2SERV" value="AUTO" type="submit">Automático</button>
<button onclick="location.href='#c2tempd';window.location.reload();" class="button"
name="C2SERV" value="MAN" type="submit">Manual</button></center></td>
<td><center><p><h4>Luz: 20 % </h4></p></center></td></tr>
<tr><td><center><p><h5 id="c2posid">Posição Desejada: 12</h5></p>
<button onclick="location.href='#c2posid';window.location.reload();" class="button"
name="C2SERV" value="VADD" type="submit">+</button>
<button onclick="location.href='#c2posid';window.location.reload();" class="button"
name="C2SERV" value="VSUB" type="submit">-</button></center></td>
<td></td></tr>
</table></td></tr></table></center></form>
<script>setTimeout(function(){window.location.reload();window.location=['//', location.host,
location.hash].join('');},5000);</script>
</body></html>

<!DOCTYPE html><html>
<head><meta charset="UTF-8" name="viewport" content="width=device-width, initial-scale=1">
<link rel="icon" href="data:,">
```

```
<style>html {scroll-behavior: smooth; font-family: Helvetica; display: inline-block; margin:
0px auto; text-align: center;}
.button { background-color: #4CAF50; border: 2px solid #4CAF50;; color: white; padding: 15px
32px; text-align: center; text-decoration: none; display: inline-block; font-size: 16px;
margin: 4px 2px; cursor: pointer; }
text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}
</style></head>
<body><center><h1>Automação Residencial</h1></center>
<form><center><table border=1><tr><td><center><h2>Planta 2</h2></center>
<table border=1><tr><td><center><h3>Atuadores</h3></center></td>
<td><center><h3>Sensores</h3></center></td></tr>
<tr><td><center><p><h4 id="c1lamp"> Lâmpada: Desligado</h4></p>
<button onclick="location.href='#c1lamp';window.location.reload();" class="button"
name="C1LAMP" value="ON" type="submit">Ligado</button>
<button onclick="location.href='#c1lamp';window.location.reload();" class="button"
name="C1LAMP" value="OFF" type="submit">Desligado</button>
<button onclick="location.href='#c1lamp';window.location.reload();" class="button"
name="C1LAMP" value="AUTO" type="submit">Automático</button></center></td>
<td><center><p><h4>Presença: Desativado</h4></p></center></td></tr>
<tr><td><center><p><h4 id="c1vent"> Ventilador: Desligado</h4></p>
<button onclick="location.href='#c1vent';window.location.reload();" class="button"
name="C1VENT" value="ON" type="submit">Ligado</button>
<button onclick="location.href='#c1vent';window.location.reload();" class="button"
name="C1VENT" value="OFF" type="submit">Desligado</button>
<button onclick="location.href='#c1vent';window.location.reload();" class="button"
name="C1VENT" value="AUTO" type="submit">Automático</button></center></td>
<td><center><p><h4>Temperatura Atual: 29 C</h4></p></center></td></tr>
<tr><td><center><p><h5 id="c1tempd">Temperatura Desejada: 25 °C</h5></p>
<button onclick="location.href='#c1tempd';window.location.reload();" class="button"
name="C1VENT" value="VADD" type="submit">+</button>
<button onclick="location.href='#c1tempd';window.location.reload();" class="button"
name="C1VENT" value="VSUB" type="submit">-</button></center></td>
<td></td></tr>
<tr><td><center><p><h4 id="c1servo"> Servo: Desligado</h4></p>
<button onclick="location.href='#c1servo';window.location.reload();" class="button"
name="C1SERV" value="ON" type="submit">Ligado</button>
<button onclick="location.href='#c1servo';window.location.reload();" class="button"
name="C1SERV" value="OFF" type="submit">Desligado</button>
<button onclick="location.href='#c1servo';window.location.reload();" class="button"
name="C1SERV" value="AUTO" type="submit">Automático</button>
<button onclick="location.href='#c1servo';window.location.reload();" class="button"
name="C1SERV" value="MAN" type="submit">Manual</button></center></td>
<td><center><p><h4>Luz: 71 % </h4></p></center></td></tr>
<tr><td><center><p><h5 id="c1posid">Posição Desejada: 12</h5></p>
<button onclick="location.href='#c1posid';window.location.reload();" class="button"
name="C1SERV" value="VADD" type="submit">+</button>
<button onclick="location.href='#c1posid';window.location.reload();" class="button"
name="C1SERV" value="VSUB" type="submit">-</button></center></td>
<td></td></tr>
</table></td>
<td><center><h2>Planta 1</h2></center>
<table border=1><tr><td><center><h3>Atuadores</h3></center></td>
<td><center><h3>Sensores</h3></center></td></tr>
<tr><td><center><p><h4 id="c2lamp"> Lâmpada: Desligado</h4></p>
<button onclick="location.href='#c2lamp';window.location.reload();" class="button"
name="C2LAMP" value="ON" type="submit">Ligado</button>
<button onclick="location.href='#c2lamp';window.location.reload();" class="button"
name="C2LAMP" value="OFF" type="submit">Desligado</button>
<button onclick="location.href='#c2lamp';window.location.reload();" class="button"
name="C2LAMP" value="AUTO" type="submit">Automático</button></center></td>
<td><center><p><h4>Presença: Desativado</h4></p></center></td></tr>
```

```
<tr><td><center><p><h4 id="c2vent"> Ventilador: Desligado</h4></p>
<button onclick="location.href='#c2vent';window.location.reload();" class="button"
name="C2VENT" value="ON" type="submit">Ligado</button>
<button onclick="location.href='#c2vent';window.location.reload();" class="button"
name="C2VENT" value="OFF" type="submit">Desligado</button>
<button onclick="location.href='#c2vent';window.location.reload();" class="button"
name="C2VENT" value="AUTO" type="submit">Automático</button></center></td>
<td><center><p><h4>Temperatura Atual: 28 C</h4></p></center></td></tr>
<tr><td><center><p><h5 id="c2tempd">Temperatura Desejada: 25 °C</h5></p>
<button onclick="location.href='#c2tempd';window.location.reload();" class="button"
name="C2VENT" value="VADD" type="submit">+</button>
<button onclick="location.href='#c2tempd';window.location.reload();" class="button"
name="C2VENT" value="VSUB" type="submit">-</button></center></td>
<td></td></tr>
<tr><td><center><p><h4 id="c2servo"> Servo: Desligado</h4></p>
<button onclick="location.href='#c2tempd';window.location.reload();" class="button"
name="C2SERV" value="ON" type="submit">Ligado</button>
<button onclick="location.href='#c2tempd';window.location.reload();" class="button"
name="C2SERV" value="OFF" type="submit">Desligado</button>
<button onclick="location.href='#c2tempd';window.location.reload();" class="button"
name="C2SERV" value="AUTO" type="submit">Automático</button>
<button onclick="location.href='#c2tempd';window.location.reload();" class="button"
name="C2SERV" value="MAN" type="submit">Manual</button></center></td>
<td><center><p><h4>Luz: 20 % </h4></p></center></td></tr>
<tr><td><center><p><h5 id="c2posid">Posição Desejada: 12</h5></p>
<button onclick="location.href='#c2posid';window.location.reload();" class="button"
name="C2SERV" value="VADD" type="submit">+</button>
<button onclick="location.href='#c2posid';window.location.reload();" class="button"
name="C2SERV" value="VSUB" type="submit">-</button></center></td>
<td></td></tr>
</table></td></tr></table></center></form>
<script>setTimeout(function(){window.location.reload();window.location=['//', location.host,
location.hash].join('');},5000);</script>
</body></html>
```

APÊNDICE E – CÓDIGO VHDL

Aqui é apresentado o código VHDL da página local.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_BIT.ALL;
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_BIT.ALL;
USE WORK.FUNCOES.ALL;


ENTITY operacao IS
          PORT(CLK : IN BIT;
                      RESET : IN BIT;
                ESCRITA : IN BIT;
                      LIBERA: IN BIT;
                      NUM : IN BIT_VECTOR(2 DOWNTO 0);
                      DADOSI : IN BIT_VECTOR(7 DOWNTO 0);
                      LIGA : OUT BIT;
                      DADOSO : OUT BIT_VECTOR(7 DOWNTO 0));
END ENTITY;

ARCHITECTURE operacao OF operacao IS
SIGNAL TEMP1, TEMP2, LUMI1, LUMI2: MEM;
SIGNAL AUXT1, AUXT2, AUXL1, AUXL2: BIT_VECTOR (7 DOWNTO 0);
SIGNAL TEMPD1, TEMPD2 : BIT_VECTOR(7 DOWNTO 0);
SIGNAL T1, T2, TD1, TD2, L1, L2 : BIT;
SIGNAL TP1, TP2, LU1, LU2 : INTEGER RANGE 0 TO 150;
BEGIN
  PROCESS(CLK, RESET)
  BEGIN
    IF(RESET = '0') THEN
      TP1 <= 0;
      TP2 <= 0;
                  LU1 <= 0;
                  LU2 <= 0;
                  T1 <= '0';
                  T2 <= '0';
                  L1 <= '0';
                  L2 <= '0';
                  TD1 <= '0';
                  TD2 <= '0';
    ELSIF(CLK'EVENT AND CLK = '1') THEN

                  IF(ESCRITA = '0') THEN
                            T1 <= '0';
                            T2 <= '0';
                            L1 <= '0';
                            L2 <= '0';
                            TD1 <= '0';
                            TD2 <= '0';
                  END IF;

      CASE NUM IS

        WHEN "000" =>
          IF(ESCRITA = '1' AND TD1 = '0') THEN
```

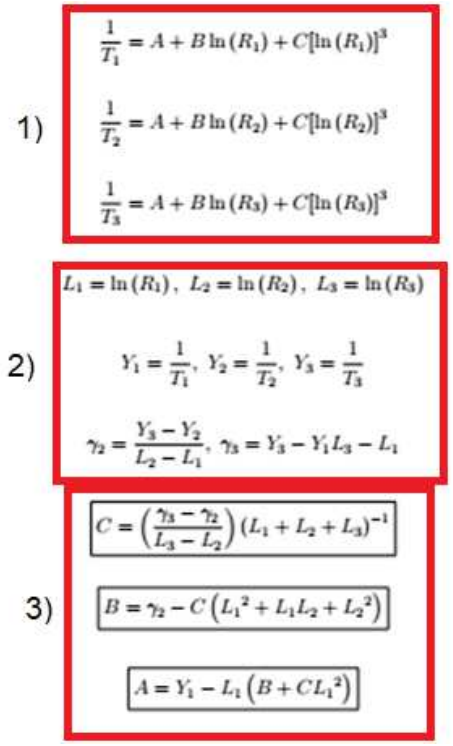
```
                                TD1 <= '1';
                                TEMPD1 <= DADOSI;
    END IF;

  WHEN "001" =>
    IF(ESCRITA = '1' AND T1 = '0') THEN
                                TEMP1(TP1) <= DADOSI;
                                TP1 <= TP1 + 1;
                                T1 <= '1';
                                IF(TP1 = 150) THEN
                                        TP1 <= 0;
                                END IF;
                        END IF;
    IF(LIBERA = '1') THEN
                                AUXT1 <= SOMA(TEMP1)(7 DOWNTO 0);
                                DADOSO <= AUXT1;
                                IF(AUXT1 > TEMPD1) THEN
                                        LIGA <= '1';
                                ELSE
                                        LIGA <= '0';
                                END IF;
    END IF;

  WHEN "010" =>
    IF(ESCRITA = '1' AND L1 = '0') THEN
                                LUMI1(LU1) <= DADOSI;
                                LU1 <= LU1 + 1;
                                L1 <= '1';
                                IF(LU1 = 150) THEN
                                        LU1 <= 0;
                                END IF;
                        END IF;
    IF(LIBERA= '1') THEN
                                AUXL1  <= SOMA(LUMI1)(7 DOWNTO 0);
                                DADOSO  <= AUXL1 ;
    END IF;

              WHEN "011" =>
                        IF(LIBERA = '1') THEN
                                DADOSO <= SERVO(AUXL1);
                        END IF;

  WHEN "100" =>
    IF(ESCRITA = '1' AND TD2 = '0') THEN
                                TD2 <= '1';
                                TEMPD2 <= DADOSI;
    END IF;

  WHEN "101" =>
    IF(ESCRITA = '1' AND T2 = '0') THEN
                                TEMP2(TP2) <= DADOSI;
                                TP2 <= TP2 + 1;
                                T2 <= '1';
                                IF(TP2 = 150) THEN
                                        TP2 <= 0;
                                END IF;
                        END IF;
    IF(LIBERA = '1') THEN
                                AUXT2 <= SOMA(TEMP2)(7 DOWNTO 0);
                                DADOSO <= AUXT2;
                                IF(AUXT2 > TEMPD2) THEN
                                        LIGA <= '1';
```

```
                                        ELSE
                                                LIGA <= '0';
                                        END IF;
          END IF;

         WHEN "110" =>
          IF(ESCRITA = '1' AND L2 = '0') THEN
                                        LUMI2(LU2) <= DADOSI;
                                        LU2 <= LU2 + 1;
                                        L2 <= '1';
                                        IF(LU2 = 150) THEN
                                                LU2 <= 0;
                                        END IF;
                                END IF;
          IF(LIBERA = '1') THEN
                                        AUXL2 <= SOMA(LUMI2)(7 DOWNTO 0);
                                        DADOSO <= AUXL2;
          END IF;

                                WHEN "111" =>
                                 IF(LIBERA = '1') THEN
                                        DADOSO <= SERVO(AUXL2);
                                 END IF;

      END CASE;
    END IF;
  END PROCESS;
END ARCHITECTURE;


LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_BIT.ALL;

PACKAGE FUNCOES IS
          TYPE MEM IS ARRAY(0 TO 150) OF BIT_VECTOR (7 DOWNTO 0);

          FUNCTION SOMA(SIGNAL SUM : MEM) RETURN BIT_VECTOR;
          FUNCTION SERVO(SIGNAL LUZ : BIT_VECTOR(7 DOWNTO 0)) RETURN BIT_VECTOR;

END PACKAGE;

PACKAGE BODY FUNCOES IS
          FUNCTION SOMA(SIGNAL SUM : MEM) RETURN BIT_VECTOR IS
          VARIABLE AUX : UNSIGNED (14 DOWNTO 0) := "000000000000000";
          BEGIN
                    I1 : FOR i IN 0 TO 150 LOOP
                              AUX := AUX + unsigned(SUM(i));
                    END LOOP;
                    RETURN bit_vector(AUX/"10010110");
          END SOMA;

          FUNCTION SERVO(SIGNAL LUZ : BIT_VECTOR(7 DOWNTO 0)) RETURN BIT_VECTOR IS
          VARIABLE AUX : UNSIGNED (7 DOWNTO 0) := "00000000";
          BEGIN
                    AUX := (unsigned(LUZ)/"0101") + "00000111";
                    RETURN bit_vector(AUX);
          END SERVO;
END PACKAGE BODY;
```

ANEXO A – CÁLCULO COEFICIENTES STEINHART & HART

Para utilizar a equação de Steinhart & Hart é necessário 3 valores de temperatura em Kelvin e as resistência correspondentes do termistor, os valores devem ser de 10 em 10 Kelvins de diferença (AMETHERM, 2013). Observando a Figura 41 é possível notar, os passos para se obter os coeficientes A, B e C. No passo 1 tem as três equações do sistema sem simplificação, no passo 2 faz a substituição de algumas variáveis para simplificar a resolução e no terceiro passo tem o sistema que deve ser resolvido para obter os coeficientes. T1, T2 e T3 são os três valores de temperatura e R1, R2 e R3 são os valores de resistência obtidos.

Figura 41 – Cálculo coeficientes A, B e C.



Fonte: Adaptado de AMETHERM, 2013