



PROJETO FINAL
MONITORAMENTO DE CORREDORES FRIO EM DATA CENTERS

VAGNER SANCHES VASCONCELOS

Projeto final da Fase de Capacitação Básica do Programa de Residência Tecnológica em Sistemas Embarcados – Embarcotech do Instituto Hardware BR – HBr.

Campinas
2025

SUMÁRIO

1 INTRODUÇÃO.....	2
2 ESCOPO DO PROJETO.....	2
3 COMPONENTES DE HARDWARE.....	7
4 COMPONENTES DE SOFTWARE.....	12
5 EXECUÇÃO DO PROJETO.....	15
REFERÊNCIAS.....	21

1 INTRODUÇÃO

Este trabalho descreve o processo de desenvolvimento de um projeto de sistemas embarcados, apresentando os passos seguidos desde o alinhamento estratégico do projeto com os direcionados de valor (*drivers*) de negócio, passando pela concepção, implementação, testes e validação do sistema.

De forma geral, a metodologia de projeto adotada está em linha com as propostas em [1] envolvendo as seguintes etapas iterativas: i) levantamento e elucidação dos requisitos; ii) especificação, que envolve a descrição em linguagem clara e a validação dos requisitos pelos interessados; iii) arquitetura, descrição funcional do sistema em diagramas de blocos, destacando os principais componentes e módulos do sistema; iv) componentes, seleção dos componentes de *hardware* e *software* que atendam as especificações; e integração de sistemas. Além das etapas citas, a metodologia adotada envolve a abordagem evolucionária, conforme Figura 1.

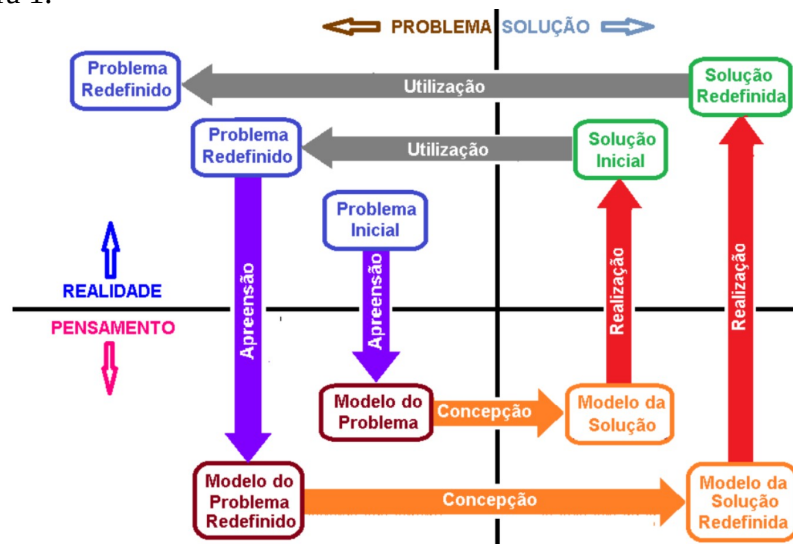


Figura 1: Espiral evolucionária.

Fonte: [1]

Essa abordagem, conforme relatado em [1], é um ciclo de vida dinâmico de desenvolvimento, envolvendo a prototipação, que após sucessivas iterações e modificações no protótipo obtêm-se os resultados desejados:

“[...] sendo que essa filosofia de desenvolvimento reflete o espírito da engenharia, que procura aplicar os conhecimentos científicos e os procedimentos empíricos à criação de equipamentos, sistemas, métodos e processos.”

2 ESCOPO DO PROJETO

2.1 DESCRIÇÃO DO PROJETO

O projeto em questão busca resolver um problema em corredores frios de data centers, conforme destacado em vermelho na Figura 2.

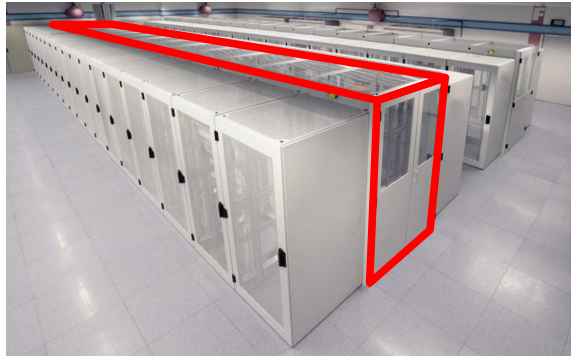


Figura 2: Gabinetes enclausurados em um Data Center.

Fonte: [9]

Nestes sistemas, o ar frio é normalmente injetado pelo piso, depois passa por dentro dos ativos de rede nos gabinetes e volta para a unidade de refrigeração, conforme ilustrado na Figura 3.

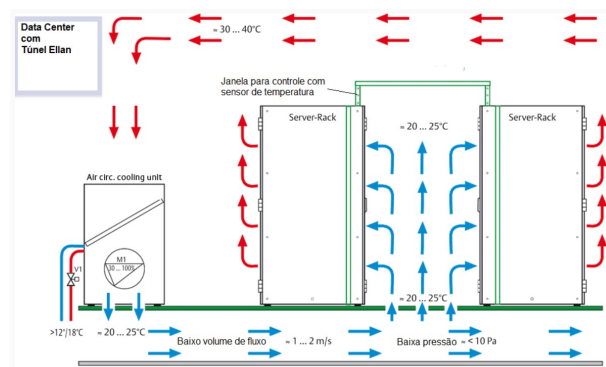


Figura 3: Fluxo de calor em um sistema de Túneis frio.

Fonte: [9]

Para preservar os ativos, é muito comum os data centers possuírem sistemas de extinção de incêndio utilizando um gás de extinção, tipo o FM-200. Em caso da detecção do incêndio, automaticamente o gás é acionado e inunda a sala apagando o fogo e preservando os computadores. Contudo, para ser eficaz é necessário que o teto do corredor frio abra para o gás extintor atingir os equipamentos.

Toda essa integração já é normalmente realizada, contudo, pode acontecer do sistema de acionamento da abertura do teto do corredor frio abrir indevidamente, por uma falha qualquer, e isso não é normalmente monitorado, e neste caso tem-se uma perda grande de eficiência energética, uma vez que até ser detectado por algum funcionário, haverá perda de ar frio diretamente para o corredor quente.

2.2 OBJETIVOS DO PROJETO

O objetivo principal deste projeto é monitorar a abertura indevida do teto do corredor frio em um data center. Além disso, também é necessário o monitoramento da temperatura e umidade dentro do corredor frio.

2.3 REQUISITOS DO PROJETO

Os requisitos do projeto foram organizados em funcionais e não funcionais, conforme detalhado na sequência.

2.3.1 Requisitos funcionais

[RFU-01]: O sistema deve emitir alarme caso ocorra abertura indevida do teto do corredor frio;

[RFU-02]: O sistema deve emitir alarme caso a temperatura do corredor frio fique menor que 15°C ou ultrapasse 32°C;

[RFU-03]: O sistema deve emitir alarme caso a umidade do corredor frio fique seja menor que 20% ou maior que 80%;

2.3.2 Requisitos não funcionais

[RNF-01]: Os alarmes devem ser externados via protocolo MQTT;

[RNF-02]: Todos os sinais devem possuir estampa de tempo (dia/mês/ano – hora:min:seg);

[RNF-03]: O sistema deve enviar regularmente, a cada 5min, via protocolo MQTT, informações de que todo o sistema (*hardware/software*) está funcionando perfeitamente (*watchdog*);

[RNF-04]: Em caso de perde da comunicação sem fio, o sistema deve armazenar todos os alarmes e enviá-los assim que a comunicação for restabelecida. O sistema deve possuir capacidade de armazenamento por uma semana.

2.4 DESCRIÇÃO DO FUNCIONAMENTO DO PROJETO

O projeto em questão utiliza a plataforma Raspberry Pi Pico W, ilustrada na Figura 4, que é um sistema embarcado baseado no microcontrolador RP2040¹, com comunicação sem fio de 2,4 GHz (802.11 n) e Bluetooth 5.2 embutidas, possibilitando a utilização em projetos IoT (Internet das Coisas).

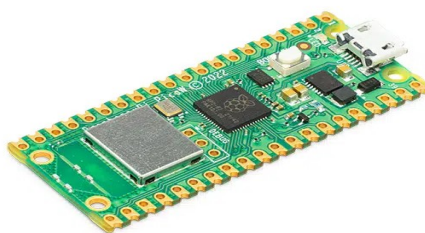


Figura 4: Raspberry Pi Pico W.

Fonte: [10]

Existem no mercado inúmeras alternativas de controladores programáveis para endereçar este caso de uso, contudo, o microcontrolador RP2040, especialmente quando embarcado na plataforma Raspberry Pi Pico W é uma alternativa quase imbatível no quesito

1 <https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf>

preço, uma vez que custa algo em torno de \$6.00², sendo que toda a plataforma de programação (*software*) é gratuita [2], [3].

Neste projeto, além da plataforma Raspberry Pi Pico W, serão utilizados os seguintes sensores:

- (a) Ultrassom: para medir distâncias, uma vez que quando o teto do corredor frio bascula, estes sensores podem captar a diminuição da distância interna no corredor, inferindo assim que o teto basculou;
- (b) Temperatura: estes sensores podem ser alocados tanto a frente dos ativos computacionais – medindo a temperatura antes do ar frio passar pelos servidores – como atrás, medindo assim a temperatura após o ar frio passar por dentro dos ativos;
- (c) Umidade: alocado no interior do corredor frio de forma a medir a umidade do sistema.

Os sinais dos sensores serão monitorados constantemente pelo Raspberry Pi Pico W, e caso ocorra desvios, essas informações serão enviadas por meio do protocolo MQTT ao sistema de monitoramento.

Para prototipação, será utilizado a placa BitDogLab, ilustrada na Figura 5, que segundo [4] é um *hardware* de código aberto, de baixo custo e fácil de usar, projetado especificamente para ensinar conceitos da STEAM³ de maneira interativa e envolvente.

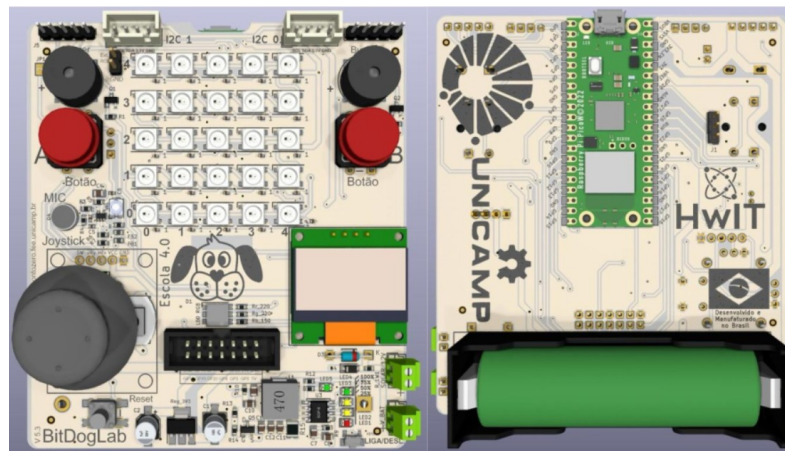


Figura 5: Placa BitDogLab.

Fonte: [4]

A conexão dos sensores de ultrassom, temperatura e umidade será realizado por meio do conector de expansão de GPIOs (Conector IDC), conforme a pinagem apresentada na Figura 6.

2 <https://www.raspberrypi.com/products/raspberry-pi-pico/>

3 *STEAM* é uma abordagem pedagógica que integra as disciplinas de Ciência, Tecnologia, Engenharia, Artes e Matemática.

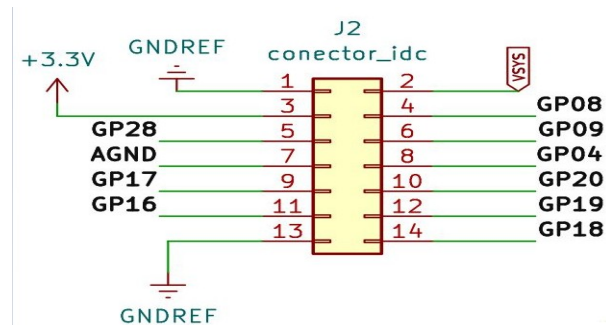


Figura 6: Conexões de expansão do BitDogLab.
Fonte: [11]

2.5 JUSTIFICATIVA

Conforme apresentado em [5], os data centers normalmente contêm grandes quantidades de equipamentos de TI, como computadores, *storages*, *racks*, elementos de rede, entre outros. Neste sentido, os autores afirmam que [5, p. 788]:

O consumo de energia do data center pode ser decomposto em sistema de equipamentos de TI (50%, incluindo servidores, equipamentos de armazenamento e equipamentos de rede), sistemas de ar condicionado e refrigeração (37%, sendo sistema de refrigeração a ar em torno de 25% e sistema de alimentação e retorno de ar cerca de 12%), sistema de distribuição (10%) e sistema de iluminação auxiliar (3%).

Desta forma, o projeto em questão vai ao encontro da meta 7.3 dos Objetivos do Desenvolvimento Sustentável (ODS) da ONU “até 2030, dobrar a taxa global de melhoria da eficiência energética” [6], bem como está em linha no sentido da busca pela excelência operacional, endereçando assim total alinhamento estratégico.

2.6 ORIGINALIDADE (PESQUISA)

Em pesquisa no Google Acadêmico⁴ não foram encontrados trabalhos específicos de monitoramento do teto de corredores frio em Data Centers, contudo foram encontrados diversos trabalhos de monitoramento de temperatura e umidades, tais como os listados abaixo:

A. M. Santiago et al., “**Adaptive Model IoT for Monitoring in Data Centers**”, IEEE Access, vol. PP, p. 1–1, dez. 2019, doi: 10.1109/ACCESS.2019.2963061.

G. da S. Donizetti, L. F. da C. Rodrigues, E. S. de Almeida, e M. A. A. Santana, “**Uso da IoT para Monitoramento de Temperatura, Umidade e Presença em Data Centers**”, Rev. Eletrônica Iniciaç. Científica Em Comput., vol. 21, no 1, Art. no 1, maio 2023, Acesso em: 16 de fevereiro de 2025. [Online]. Disponível em: <https://journals-sol.sbc.org.br/index.php/reic/article/view/2144>

4 <https://scholar.google.com.br/?hl=pt>

3 COMPONENTES DE HARDWARE

3.1 DIAGRAMA DE BLOCOS

A Figura 7 apresenta o diagrama de blocos do *hardware* envolvido no sistema.

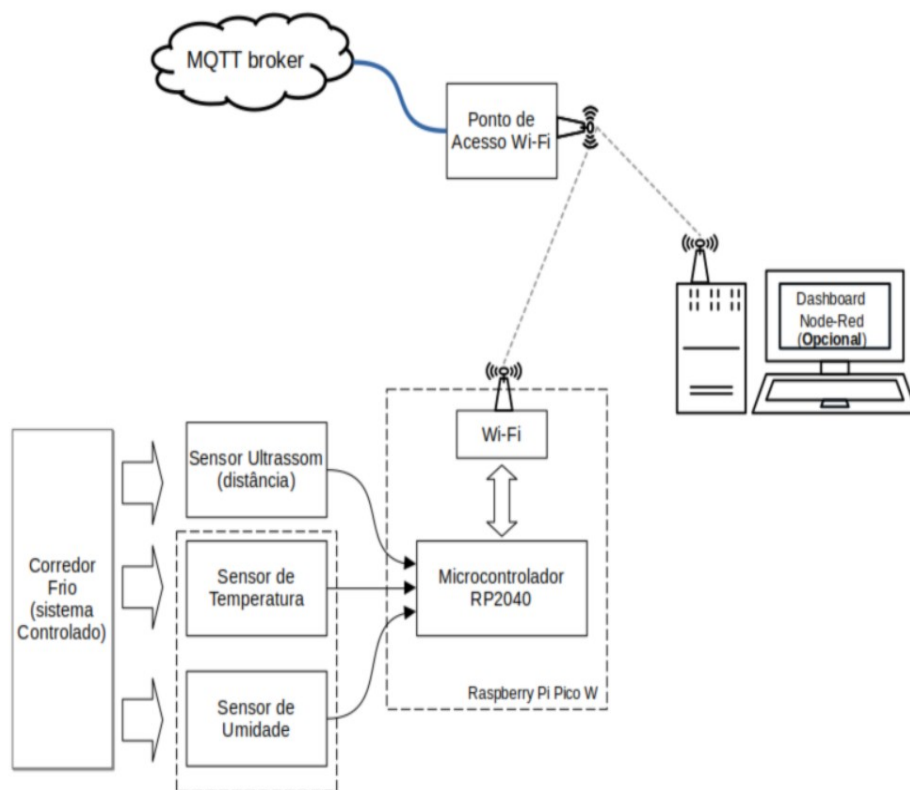


Figura 7: Diagrama de blocos de *hardware*.

Fonte: Produzido pelo autor.

3.2 FUNÇÃO DOS BLOCOS

(a) Placa Raspberry Pi Pico W.

Constituída pelo microcontrolador RP2040 e módulo Wi-Fi (chip CYW43439) padrão 802.11n. Para o protótipo foi utilizado o kit *BitDogLab*.

O microcontrolador coleta todos os dados dos sensores, realizando para isso todas as configurações e processamentos necessários, depois, empacota todas essas informações no protocolo MQTT, enviando-as para o chip Wi-Fi do módulo Pi Pico W que as publica (cliente *publisher*), sendo-as recebidas pelo MQTT Broker, que neste projeto rodará em nuvem.

(b) Sensor de ultrassom HC-SR04.

Utiliza o princípio da emissão de uma onda sonora e na medição do tempo que leva para a recepção do eco, com isso, é possível calcular a distância envolvida. Em condições normais do corredor frio, isto é, o teto fechado, tem-se a distância padrão do corredor, caso

qualquer parte do teto bascule, esse sensor captará a diminuição da distância, emitindo um alarme.

(c) Sensor de temperatura e umidade DHT-22.

Este sensor, em um único envólucro, é capaz de medir simultaneamente temperatura e umidade, simplificando assim o projeto em questão. Ele utiliza um protocolo serial para comunicação.

(d) Ponto de acesso Wi-Fi.

Dispositivo de rede que permite a conexão de dispositivos sem fio a uma rede com fio e a nuvem.

(e) Broker MQTT (*Message Queuing Telemetry Transport*)

Aplicação que recebe as informações de um cliente *publisher* (RP2040), realiza filtros e as envia aos clientes *subscribers* (computador/*dashboard*) inscritos no tópico referente as informações necessárias, neste caso: distância, temperatura e umidade.

(f) Computador com aplicação *dashboard*.

Será o cliente *subscriber* dos dados dos sensores instalados no corredor frio, sendo que essas informações alimentarão um *dashboard* dinâmico. Um desejo (opcional) é ainda a integração com a ferramenta *Node-Red*⁵.

3.3 CONFIGURAÇÃO DOS BLOCOS

(a) Sensor de ultrassom HC-SR04

Conforme apresentado no *datasheet*⁶ do HC-SR04 e em [7]:

- (i) O pino Vcc deve ser conectado em 5V;
- (ii) O pino Trig é um pino de saída e deve enviar um pulso para disparar o processo de medição;
- (iii) O pino Echo é um pino de entrada, este deve ser monitorado pelo microcontrolador de forma a medir o tempo de reflexão da onda e consequentemente a distância. **IMPORTANTE:** necessário utilizar divisor resistivo para ajustar a tensão em 3,3V no RP2040, conforme [7, p. 35], caso contrário, há risco de queima do microcontrolador.

A medição segue os seguintes passos:

- (i) O RP2040 deve setar o pino Trig do sensor em nível alto por pelo menos 10µs;
- (ii) O sensor emite 8 pulsos ultrassônicos e coloca o pino Echo em nível alto;
- (iii) Quando o sensor detecta o eco dos pulsos, ele retorna o pino Echo para o nível baixo;
- (iv) Se o eco não for detectado em aproximadamente 38ms, o pino Echo retorna ao nível baixo.

⁵ Ferramenta *low-code* de programação que permite criar aplicações para a Internet das Coisas (IoT).

⁶ <https://www.handsontec.com/dataspecs/HC-SR04-Ultrasonic.pdf>

Medindo o tempo em que o pino Echo fica em nível alto, pode-se calcular a distância, a Figura 8 ilustra o processo.

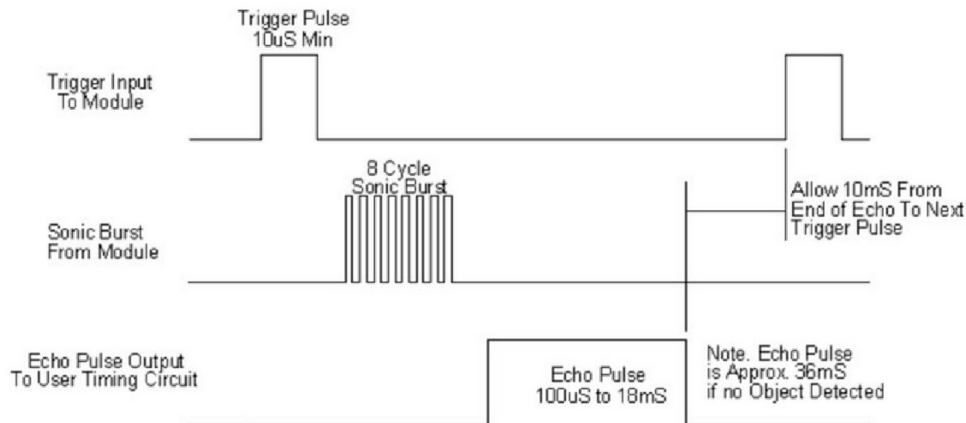


Figura 8: Datasheet do HC-SR04.

(b) Sensor de temperatura e umidade DHT-22

Conforme apresentado no *datasheet*⁷ do DHT-22 e em [7]:

- (i) O pino Vcc deve ser conectado em 3V;
- (ii) O pino de Dados (SDA) necessita de um resistor de pull-up.

A medição segue os seguintes passos:

- (i) O RP2040 força a linha de dados para terra por 1ms, solicitando ao sensor a leitura;
- (ii) Após o RP2040 soltar a linha de dados, de 20μs a 40μs, o sensor: i) confirma a solicitação, forçando a linha de dados para terra por 80μs; ii) aguarda mais 80μs; e iii) envia a resposta com 40 bits, conforme Figura 9.

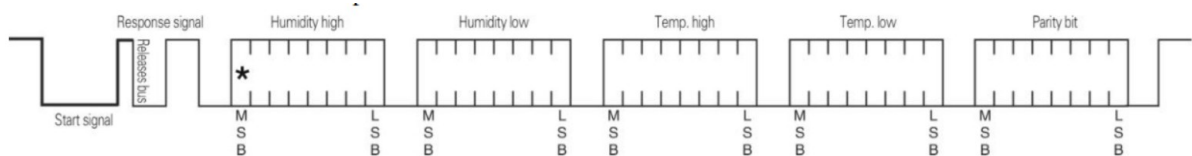


Figura 9: Datasheet do DHT22.

Sendo que, para cada bit: i) a linha de dados é forçada para terra por 50μs; ii) para representar “0”, a linha de dados é solta por 26μs a 28μs; e iii) para representar “1”, a linha de dados é solta por 70μs.

3.4 ESPECIFICAÇÕES

A Tabela 1 apresenta a análise das especificações em função dos requisitos, sendo a coluna “Análise crítica” a avaliação dos cliente quanto a proposta. Considerando a abordagem

7 <https://cdn.awsli.com.br/945/945993/arquivos/AM2302.pdf>

evolucionária proposta neste trabalho, alguns requisitos só serão atendidos após a redefinição do problema, isto é, não estará presente na Solução Inicial.

Tabela 1: Análise crítica do atendimento aos requisitos.

Requisito	Especificação	Análise crítica
[RFU-01]: O sistema deve emitir alarme caso ocorra abertura indevida do teto do corredor frio.	O sensor HC-SR04 permitirá monitoramento de corredores frios de até 4m de comprimento, caso o tamanho seja maior, será necessária utilização de mais de um sensor no corredor. Para verificar se a abertura é realmente indevida, será necessário também o monitoramento do contato seco da central de alarme de incêndio	Cliente de acordo, contudo solicita que o alarme seja “Teto do corredor frio ABERTO”
[RFU-02]: O sistema deve emitir alarme caso a temperatura do corredor frio fique menor que 15°C ou ultrapasse 32°C	O sensor DHT22 permite um range de funcionamento de -40°C até 80°C, possuindo ainda uma resolução de 0,1°C, atendendo assim plenamente ao requisito em questão	Cliente de acordo
[RFU-03]: O sistema deve emitir alarme caso a umidade do corredor frio fique seja menor que 20% ou maior que 80%	O sensor DHT22 permite um range de funcionamento de 0 até 99,9%, possuindo ainda uma resolução de 0,1°C, atendendo assim plenamente ao requisito em questão	Cliente de acordo
[RNF-01]: Os alarmes devem ser externados via protocolo MQTT	Na abordagem evolucionária, esta funcionalidade será desenvolvida apenas na 1ª redefinição do problema, isto é, não estará presente na Solução Inicial	Cliente de acordo
[RNF-02]: Todos os sinais devem possuir estampa de tempo (dia/mês/ano – hora:min:seg)	Na abordagem evolucionária, esta funcionalidade será desenvolvida apenas na 1ª redefinição do problema, isto é, não estará presente na Solução Inicial	Cliente de acordo
[RNF-03]: O sistema deve enviar regularmente, a cada 5min, via protocolo MQTT, informações de que todo o sistema (hardware/software) está funcionando perfeitamente (watchdog)	Na abordagem evolucionária, esta funcionalidade será desenvolvida apenas na 1ª redefinição do problema, isto é, não estará presente na Solução Inicial	Cliente de acordo

[RNF-04]: Em caso de perde da comunicação sem fio, o sistema deve armazenar todos os alarmes e enviá-los assim que a comunicação for restabelecida. O sistema deve possuir capacidade de armazenamento por uma semana	Na abordagem evolucionária, esta funcionalidade será desenvolvida apenas na 1ª redefinição do problema, isto é, não estará presente na Solução Inicial	Cliente de acordo
---	--	-------------------

3.5 LISTA DE MATERIAIS

A Tabela 2 apresenta a lista de materiais apenas com os componentes do sistema embarcado, o ponto de acesso, o computador e o acesso ao serviço de nuvem para executar o MQTT Broker não foram considerados.

Tabela 2: Lista de Materiais

Item	Descrição	Quantidade
01	Pi Pico W (Kit BitDogLab)	01
02	Sensor de Ultrassom HC-SR04	01
03	Sensor de temperatura e umidade DHT22	01

3.6 DESCRIÇÃO DA PINAGEM

A Tabela 3 apresenta a descrição da pinagem utilizada nos periféricos, no Pi Pico W e na BitDogLab.

Tabela 3: Pinagem dos componentes.

Periféricos		Pi Pico W (RP2040)	Conector IDC BitDogLab
HC-SR04	VCC (5V)	VSYS (39)	2
	Trigger	GP08 (11)	4
	Echo ⁸	GP09 (12)	6
	GND	GNDREF ⁹	1
DHT22	Vcc (3V)	36 (3V3_OUT)	3
	GND	GNDREF	13
	Dados (SDA)	GP04 (6)	8

⁸ Necessário utilizar divisor resistivo para ajustar a tensão em 3,3V no RP2040, conforme [7, p. 35]

⁹ GNDREF: Na BitDogLab os pínos 3, 8, 13, 18, 23, 33 e 38 foram interligados.

3.7 CIRCUITO COMPLETO DO HARDWARE

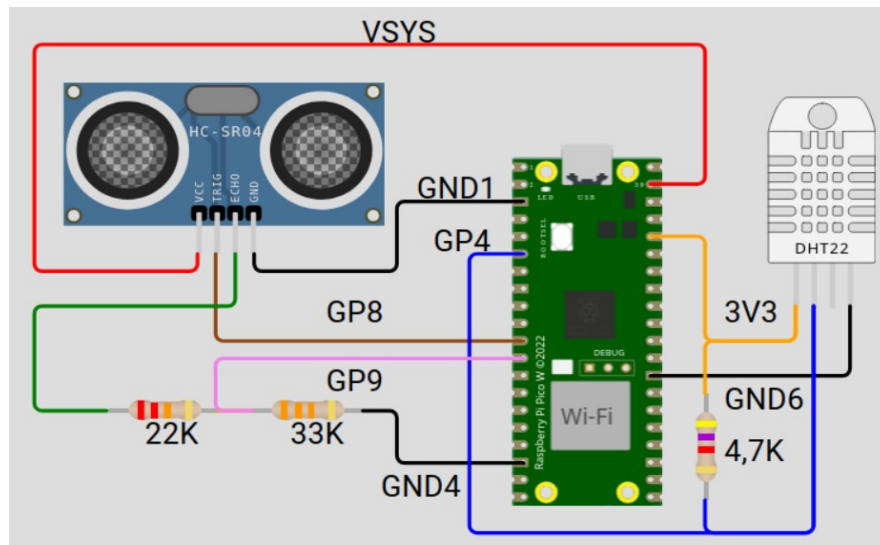


Figura 10: Circuito de hardware.

Fonte: Produzido pelo autor na plataforma [Wokwi](https://wokwi.com/projects/422958975184963585)¹⁰.

4 COMPONENTES DE SOFTWARE

4.1 BLOCOS FUNCIONAIS

A Figura 11 apresenta o diagrama de blocos das camadas de *software*.

4.2 DESCRIÇÃO DAS FUNCIONALIDADES

Para o sensor de ultrassom HC-SR04, o função de *callback* **alarm_callback** realiza a temporização de 10 μ s necessária para deixar o pino Trig em nível alto. Já a função de *callback* **gpio_callback** realiza a contagem do tempo em que o pino Echo fica em nível alto e o tempo em que fica em nível baixo. Por fim, a função **mede_distancia**, realiza o cálculo da distância em função dos tempos medidos.

Já para o sensor de temperatura e umidade DHT22, a função **solicita_leitura_dht** configura o GP04 como saída digital e aplica um nível baixo de 1 ms no pino SDA do sensor. Já a função **leitura_dht**, reconfigura o GP04 para entrada digital, e lê os 40 bits de resposta do sensor, sendo 2 bytes para umidade, 2 bytes para temperatura e 1 byte de *paridade*.

¹⁰ <https://wokwi.com/projects/422958975184963585>

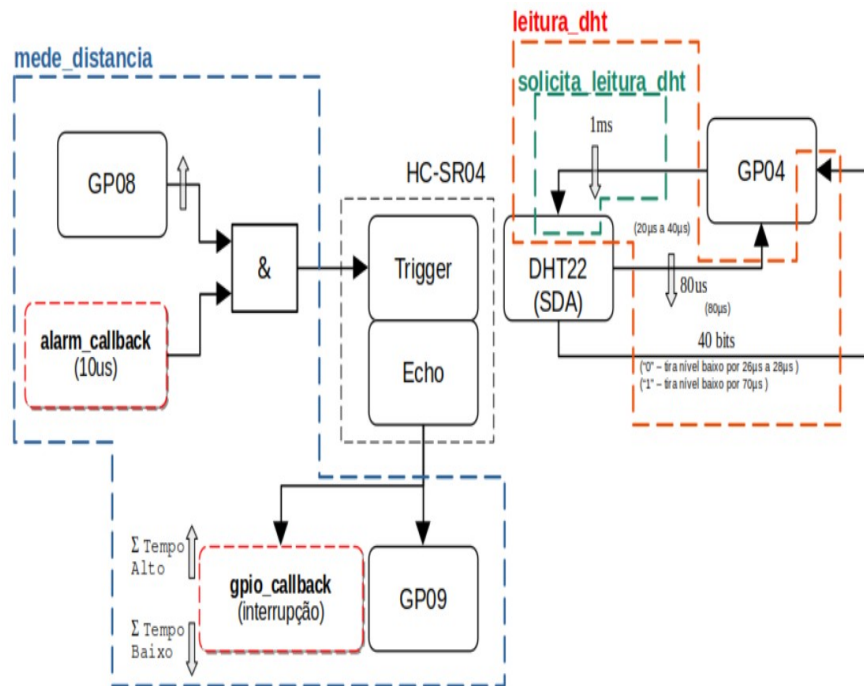


Figura 11: Diagrama de blocos do *software*.

Fonte: Produzido pelo autor.

4.3 DEFINIÇÃO DAS VARIÁVEIS

A Tabela 4 apresenta as principais variáveis utilizadas no firmware.

Tabela 4: Principais variáveis utilizadas.

Variável	Descrição
volatile int start_ticks	Variável global do tipo inteiro, que pode ser alterada a qualquer momento (<i>volatile</i>), e que mede – via interrupção – o tempo em que o pino Echo do sensor de ultrassom fica em nível alto.
volatile int end_ticks	Variável global do tipo inteiro, que pode ser alterada a qualquer momento (<i>volatile</i>), e que mede – via interrupção – o tempo em que o pino Echo do sensor de ultrassom fica em nível baixo.
volatile bool timer_fired	Variável global do tipo booleana, que pode ser alterada a qualquer momento (<i>volatile</i>), que indica (flag) que o <i>alarm</i> de 10us foi disparado, podendo assim ser retirado o nível alto do pino trigger do sensor de ultrassom.

float distance_cm	Variável local do tipo <i>float</i> que calcula a distância medida pelo sensor de ultrassom em função da diferença entre os tempos ($\text{end_ticks} - \text{start_ticks}$), multiplicado por 0,0343 e dividido por 2.
int data[5]	Variável local do tipo vetor de inteiros com cinco posições, sendo utilizada para armazenar as saídas do sensor DHT22, sendo: data[0] e data[1] a umidade; data[2] e data[3] a temperatura; e data[4] a paridade.
lendo_dht	Agrupamento (<i>struct</i>) dos dados de umidade e temperatura só sensor DHT22, sendo ambas as variáveis do tipo <i>float</i> .

4.4 FLUXOGRAMA

A Figura 12 apresenta o fluxograma completo do *software*.

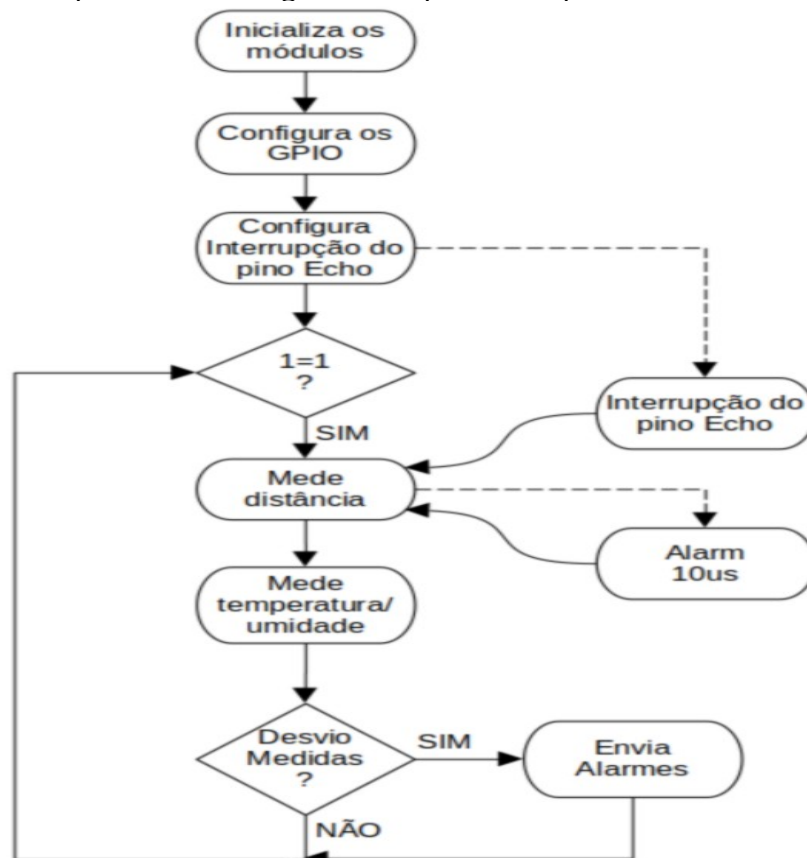


Figura 12: fluxograma do software.

Fonte: Produzido pelo autor.

4.5 INICIALIZAÇÃO

O processo de inicialização do *software* (*firmware*), conforme [8], envolveu:

i) inicializar o stdio {stdio_init_all()};

- ii) aguardar 2 seg para estabilizar o sensor {sleep_ms(2000)};
- iii) inicializar o pino GP04, associado ao pino de dados do sensor DHT {gpio_init(DHT_PIN)};
- iv) inicializar o pino GP09, associado ao pino Echo do sensor HC-SR04 {gpio_init(ECHO_PIN)};
- v) ajustar o pino GP09 como entrada {gpio_set_dir(ECHO_PIN, GPIO_IN)};
- vi) inicializar o pino Trigger do HC-SR04 {gpio_init(TRIG_PIN)};
- vii) ajustar o pino GP08 como saída {gpio_set_dir(TRIG_PIN, GPIO_OUT)}; e
- viii) habilitar interrupção no pino Echo {gpio_set_irq_enabled_with_callback(ECHO_PIN, GPIO_IRQ_EDGE_RISE | GPIO_IRQ_EDGE_FALL, true, &gpio_callback)}.

4.6 PROTOCOLOS DE COMUNICAÇÃO

O protocolo MQTT será utilizado na próxima versão versão “Solução Redefinida”, com QOS 1 – conformação de entrega pelo MQTT Broker.

5 EXECUÇÃO DO PROJETO

5.1 METODOLOGIA

Conforme apresentado no capítulo 1 Introdução, a metodologia de projeto adotada neste trabalho envolveu as etapas iterativas propostas em [1], em uma abordagem evolucionária, conforme Figura 1. Sendo assim, essa primeira entrega é nossa “Solução Inicial”, acordada com o cliente conforme a Tabela 1, sendo que após a “Redefinição do Problema” trabalharemos em uma “Solução Redefinida” contendo todas as funcionalidades acordadas.

5.1.1 Pesquisas realizadas

Além dos artigos apresentados no item 2.6 Originalidade (pesquisa), foi consultado vários portais com assuntos correlatos, com destaque para:

<https://github.com/carolinedunn/pico-weather-station/blob/main/weatherstation.c>
<https://datasheets.raspberrypi.com/pico/raspberry-pi-pico-c-sdk.pdf>
https://github.com/raspberrypi/pico-examples/blob/master/gpio/dht_sensor/dht.c
<https://www.exasub.com/development-kit/raspberry-pi-pico/how-to-use-c-sdk-to-create-uf2-file-which-interface-ultrasonic-sensor-with-raspberry-pi-pico/>

5.1.2 Escolha do *hardware*

Conforme apresentado na coluna “Especificação” da Tabela 1- Análise crítica do atendimento aos requisitos, o *hardware* adotado atende aos requisitos requisitos do projeto. A Figura 13 apresenta o hardware real montado com o *firmware* carregado.

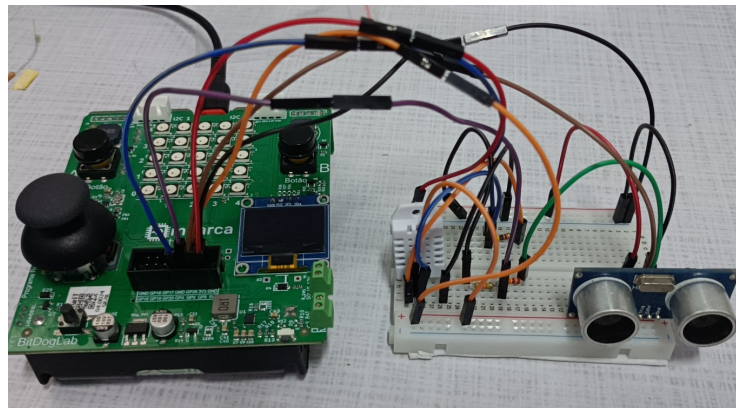


Figura 13: hardware real montado.

5.1.3 Definição das funcionalidades do *software*

O *firmware* foi desenvolvido conforme a coluna “Especificação” da Tabela 1- Análise crítica do atendimento aos requisitos. Sendo que a “Solução Inicial”, acordada com o cliente, não possui ainda todas as funcionalidades.

5.1.4 Inicialização da IDE

Para a compilação do código foi utilizado a IDE *Visual Studio Code - VSCode*¹¹ no Ubuntu 24.10; para configurar o ambiente de desenvolvimento foram utilizados as seguintes referências: Set Up Raspberry Pi Pico SDK on Ubuntu 22.04¹²; Instalação e Configuração do VSCode para as linguagens C/C++¹³; e Raspberry Pi Pico and RP2040 – C/C++¹⁴. Também foi instalado um Plug-in do Raspberry Pi Pico no VSCode¹⁵.

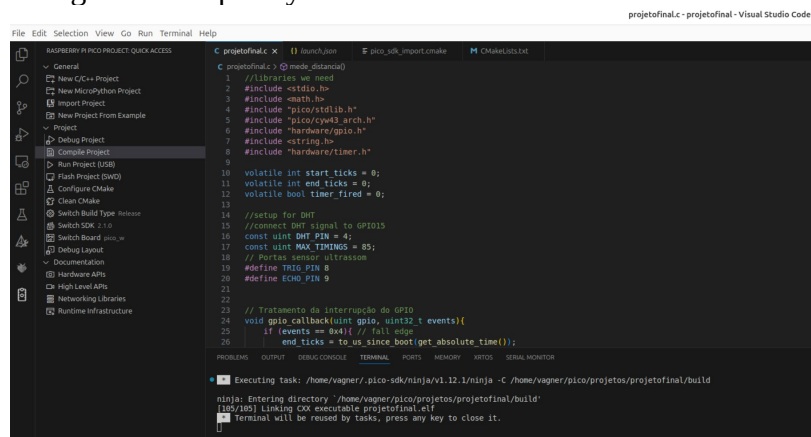


Figura 14: Final da compilação.

11 <https://code.visualstudio.com/>

12 <https://lindevs.com/set-up-raspberry-pi-pico-sdk-on-ubuntu>

13 <https://youtu.be/OOUcnVhJWGo?si=IywwYloRTZ67FEJ>

14 https://youtu.be/B5rQSoOmR5w?si=x20Q8KQh_9FgQVG7

15 <https://www.raspberrypi.com/news/get-started-with-raspberry-pi-pico-series-and-vs-code/>

Na Figura 14 é possível observar a saída na aba Terminal: “[105/105] Linking CXX executable **projetoFinal.elf**”, sendo o “**projetoFinal.elf**”¹⁶ o arquivo no Formato Executável e de Ligação (*Executable and Linkable Format*).

5.1.5 Programação na IDE

Todo o código fonte foi desenvolvido em linguagem C, utilizando o SDK da Raspberry Pi Pico¹⁷, estando disponível no Github do autor¹⁸.

A Figura 15 apresenta os arquivos do diretório raiz do projeto já com o resultado final da compilação, sendo que o arquivo “**projetoFinal.uf2**” deve ser carregado na BitDogLab.

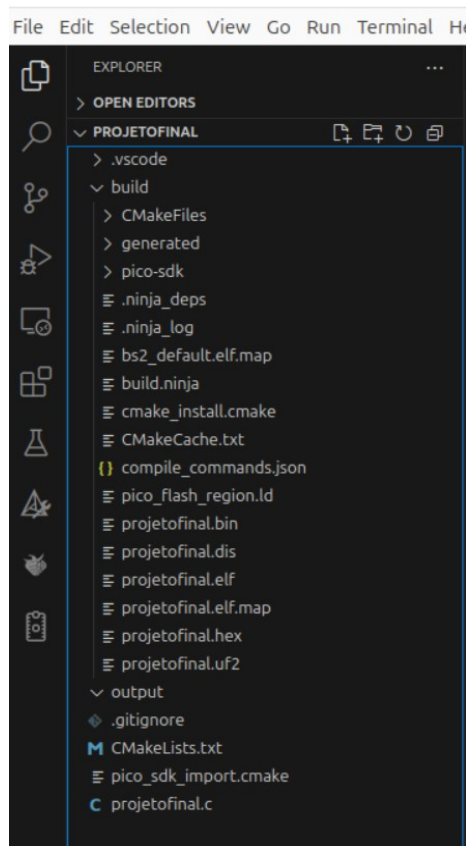


Figura 15: Diretório raiz do projeto.

5.1.6 Depuração

O desenvolvimento do projeto teve início na plataforma de simulação Wokwi¹⁹, na qual foi possível depurar o sistema com base na console de saída, conforme destacado na Figura 16.

16 https://pt.wikipedia.org/wiki/Executable_and_Linkable_Format

17 https://www.raspberrypi.com/documentation/microcontrollers/c_sdk.html

18 <https://github.com/vsvasconcelos/embarcotech>

19 <https://wokwi.com/projects/422958975184963585>

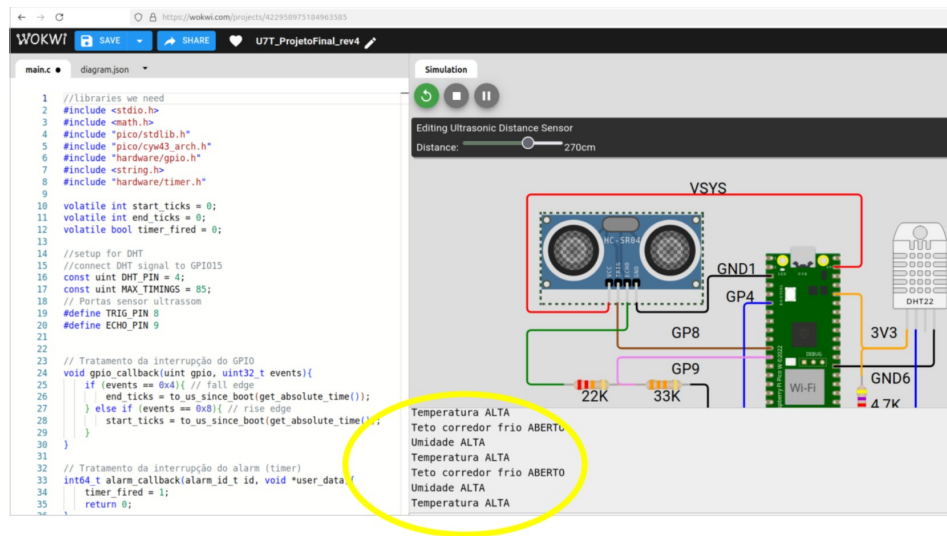


Figura 16: Desenvolvimento no simulador Wokwi.

Com o código validado no Wokwi, realizamos a compilação VSCode e posterior depuração via SERIAL MONITOR, conforme destacado na Figura 17.

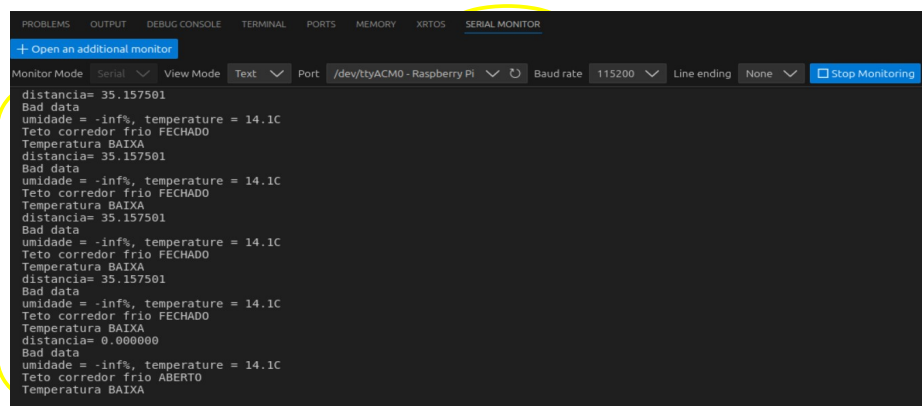


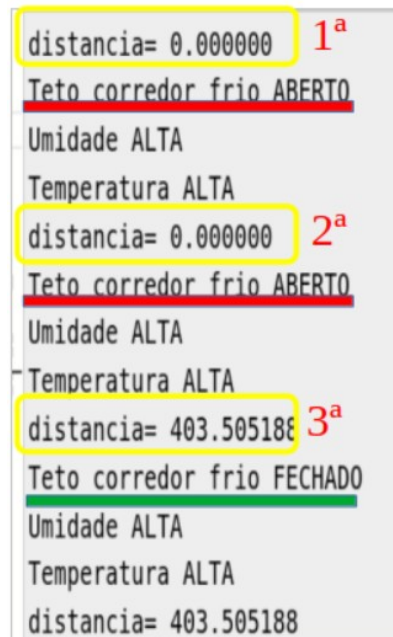
Figura 17: Depuração via Serial Monitor no VSCode.

5.2 TESTES DE VALIDAÇÃO

Inicialmente foi testado o sensor de ultrassom, obtendo resultados satisfatórios na plataforma Wokwi, contudo notou-se que somente após a 3ª leitura do sensor o resultado está em conformidade com a valor ajustado, conforme apresentado na Figura 18, onde a distância do sensor foi ajustada para 400 cm.

Após a 3ª leitura, todos os ajustes na distância foram medidos corretamente, considerando a incerteza do sensor, sendo que o alarme “Teto do corredor frio ABERTO” só é apresentado para valores **menos** que 300 cm, conforme requisito.

Com relação ao sensor de temperatura e umidade, todas as combinações de ajustes foram realizadas e o sistema alarmou conforme os requisitos apontados na Tabela 1.



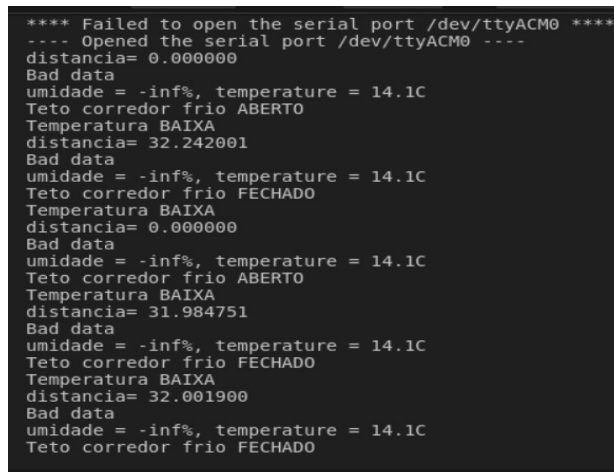
```

distancia= 0.000000 1ª
Teto corredor frio ABERTO
Umidade ALTA
Temperatura ALTA
distancia= 0.000000 2ª
Teto corredor frio ABERTO
Umidade ALTA
Temperatura ALTA
distancia= 403.505188 3ª
Teto corredor frio FECHADO
Umidade ALTA
Temperatura ALTA
distancia= 403.505188

```

Figura 18: Validação Sensor ultrassom

Na sequência, o circuito real foi testado verificando a saída do monitor serial do VSCode, conforme Figura 19, onde se observa que realmente as primeiras medidas do distância do sensor de ultrassom apresentam desvios. Além disso, o sensor DHT22 está enviando a informação inf% para umidade e de 14,1°C para temperatura, contudo a temperatura ambiente está muito mais alta, algo por volta de 25°C.



```

**** Failed to open the serial port /dev/ttyACM0 ****
---- Opened the serial port /dev/ttyACM0 ----
distancia= 0.000000
Bad data
umidade = -inf%, temperature = 14.1C
Teto corredor frio ABERTO
Temperatura BAIXA
distancia= 32.242001
Bad data
umidade = -inf%, temperature = 14.1C
Teto corredor frio FECHADO
Temperatura BAIXA
distancia= 0.000000
Bad data
umidade = -inf%, temperature = 14.1C
Teto corredor frio ABERTO
Temperatura BAIXA
distancia= 31.984751
Bad data
umidade = -inf%, temperature = 14.1C
Teto corredor frio FECHADO
Temperatura BAIXA
distancia= 32.001900
Bad data
umidade = -inf%, temperature = 14.1C
Teto corredor frio FECHADO

```

Figura 19: saída do monitor serial do VSCode.

5.3 DISCUSSÃO DOS RESULTADOS

Conforme resultados obtidos, os próximos passos envolve: i) conseguir um outro sensor DHT22 para verificar se os problemas encontrados estão no sensor ou não; e ii) pesquisar a razão das medidas iniciais do sensor de ultrassom estarem incorretas.

Além disto, a ideia é avançar no desenvolvimento, entregando todas as funcionalidades descritas na Tabela 1.

O projeto em questão, proporcionou uma experiência de aprendizagem impar, onde conseguimos vivenciar todo o ciclo de desenvolvimento de um sistema embarcado incluído sua documentação.

5.4 VÍDEO DE FUNCIONAMENTO

Link do YouTube: <https://youtu.be/ZwQJTpjLW7Q>

REFERÊNCIAS

- [1] C. E. Cugnasca, “Projetos de sistemas embarcados”. 2018. Acesso em: 16 de fevereiro de 2025. [Online]. Disponível em: <https://integra.univesp.br/courses/2710/pages/texto-base-projetos-de-sistemas-embarcados-%7C-carlos-eduardo-cugnasca>
- [2] S. S. Mumpuni e R. P. Astutik, “Implementation of Raspberry Pi PICO as PLC for Monitoring and Control in Swiftlet Cultivation Based on Weintek HMI”, *G-Tech: Jurnal Teknologi Terapan*, vol. 9, nº 1, p. 266–274, 2025, Acesso em: 9 de fevereiro de 2025. [Online]. Disponível em: <https://ejournal.uniramalang.ac.id/index.php/g-tech/article/view/6054>
- [3] “Raspberry Pi Documentation - Pico C SDK”. Acesso em: 9 de fevereiro de 2025. [Online]. Disponível em: <https://www.raspberrypi.com/documentation/pico-sdk/>
- [4] F. Fruett, F. Pereira Barbosa, S. Cardoso Zampolli Fraga, e P. Ivo Aragão Guimarães, “Empowering STEAM Activities With Artificial Intelligence and Open Hardware: The BitDogLab”, *IEEE Transactions on Education*, vol. 67, nº 3, p. 462–471, jun. 2024, doi: 10.1109/TE.2024.3377555.
- [5] J. L. R. de Brito, P. H. Iara dos Santos Matai, e M. R. dos Santos, “Data Center e Eficiência Energética: Data Center and Energy Efficiency”, *Brazilian Journal of Business*, vol. 5, nº 2, p. 786–795, 2023, Acesso em: 9 de fevereiro de 2025. [Online]. Disponível em: <https://ojs.brazilianjournals.com.br/ojs/index.php/BJB/article/view/58952>
- [6] “ODS 7 - Energia Acessível e Limpa - Ipea - Objetivos do Desenvolvimento Sustentável”. Acesso em: 9 de fevereiro de 2025. [Online]. Disponível em: <https://www.ipea.gov.br/ods/ods7.html>
- [7] D. Quadros, *Usando Sensores com a Raspberry Pi Pico*. Leanpub, 2024. Acesso em: 11 de fevereiro de 2025. [Online]. Disponível em: <https://leanpub.com/sensorespico>
- [8] R. Almeida, “Arquitetura de desenvolvimento de software - parte I”, Embarcados - Sua fonte de informações sobre Sistemas Embarcados. Acesso em: 19 de fevereiro de 2025. [Online]. Disponível em: <https://embarcados.com.br/arquitetura-de-desenvolvimento-de-software-i/>
- [9] “Sistema de Túnel para Data Center”, Ellan. Acesso em: 9 de fevereiro de 2025. [Online]. Disponível em: <https://configure.ellan.com.br/rack-para-servidor/tunel-frio>
- [10] R. P. Ltd, “Buy a Raspberry Pi Pico”, Raspberry Pi. Acesso em: 9 de fevereiro de 2025. [Online]. Disponível em: <https://www.raspberrypi.com/products/raspberry-pi-pico/>
- [11] “Esquemático do BitDogLab”. Acesso em: 11 de fevereiro de 2025. [Online]. Disponível em: https://github.com/BitDogLab/BitDogLab/blob/main/kicad/bitdoglabsmd/bitdoglab_main/bitdoglab_smd_schematics.pdf