UNIVERSIDADE ESTADUAL DO RIO GRANDE DO SUL
UNIDADE EM GUAÍBA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

ULISSES MAFFAZIOLI

Arquitetura de condicionador digital para sensores de natureza resistiva

Trabalho de Conclusão apresentado como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Prof. Dra. Letícia Vieira Guimarães
Orientadora

Porto Alegre, 15 de dezembro de 2023

UNIVERSIDADE ESTADUAL DO RIO GRANDE DO SUL
Reitor:
Vice-Reitor:
Pró-Reitor de Graduação:
Coordenador do curso de Engenharia de Computação:

AGRADECIMENTOS

Cheguei até aqui pelo esforço, não me considero uma pessoa brilhante, não sou dotado de uma inteligência eficaz para resolver problemas complicados de exatas, mas me considero extremamente dedicado e resiliente, por isso me orgulho do mérito de chegar até aqui, pois muitos que têm facilidade e condições, não chegam.

Eu agradeço a todos que estiveram envolvidos nesta minha jornada de alguma forma. Em especial, gostaria de citar alguns nomes e dizer o quão importantes foram para meu desenvolvimento intelectual e pessoal.

Diferente de mim, Lucas Karr Osielski é uma pessoa dotada de uma inteligência superior para as exatas, sem sua ajuda, não teria chegado tão longe. Minha esposa Ândria Coelho Machado, que era apenas minha namorada quando ingressei na engenharia, ela sempre esteve ao meu lado, suportando os piores momentos das intensas jornadas de estudos, grato, meu amor! Ao meu amigo e colega de trabalho Victor de Souza, no qual eu admiro e gosto muito, você segurou a barra nos momentos que eu precisei. Não podendo esquecer de mencionar o icônico Vitor Wolff, que compunha nosso trio de trabalhos acadêmicos , além de aprender muito com nossos trabalhos em grupo, eu me divertia intensamente! A todos os professores, a qual devo meu eterno reconhecimento pela realização deste nobre trabalho. Quero citar alguns nomes em especial. Primeiro para a minha talentosa, engenhosa e inteligentíssima orientadora Letícia Vieira Guimarães, foi exponencialmente bom poder dividir o laboratório contigo nas cadeiras de eletrônica e na construção do protótipo para o TCC. Ao sábio professor Roberto Ribeiro Baldino, que me ensinou a pensar e resolver problemas de forma interativa, onde eu aprendia falando e ele ensinava ouvindo - Amígdalas! Reforço aqui que não vi a barra na prova de cálculo 2, eu sabia resolver aquela questão do campo elétrico. Ao professor João Leonardo Fragoso, que muito cedo me deu a maior lição que poderia receber na universidade. Desde então, fui outra pessoa, me sinto satisfeito em ter conseguido cumprir minha promessa a tempo, de enviar o *K&S* em *System Verilog* (ver no meu *github*:ulissesmaffa), mesmo com 4 anos de atraso. A professora *outlier* Adriane Parraga, uma pessoa encantadora, dotada de uma inteligência invejável e de uma capacidade incrível de ensinar, mas principalmente, inspirar! O grande professor Celso Maciel da Costa, que tive a honra de almoçar semanalmente no Kikanto durante um semestre inteiro, gostava muito de ouvir suas histórias sobre Paris ou conversar sobre um mundo mais justo e igualitário, onde o ser humano seja valorizado e possa ter uma vida digna! Sua paciência, racionalidade e inteligência inspira a todos que o cercam. Aos professores e funcionários que não foram citados, não se sintam desprestigiados, guardo todos vocês em minhas mais carinhosas memórias.

As demais pessoas que não citei, mas que de alguma forma participaram ativamente deste processo, a todos, meu mais sincero agradecimento. Foi intenso, foi difícil, foi exaustivo, foi divertido, foi engraçado, foi desafiador, foi estressante, foi prazeroso, foi mágico!

Enfim, sem mais, muito obrigado!

“Creiam-me, o menos mau é recordar, ninguém se fie da felicidade presente; há nela uma gota de baba de Caim. Corrido o tempo e cessado o espasmo, então sim, então talvez se pode gozar deveras, porque entre uma e outra dessas ilusões, melhor é a que se gosta sem doer.”

Machado de Assis em Memórias Póstumas de Brás Cubas

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

CI	Circuito Integrado
CPU	Unidade Central de Processamento
FF-SR	*Flip-Flop Set-Reset*
GPIO	*General-Purpose Input/Output*
I2C	Inter-Integrated Circuit
I2S	*Inter-IC Sound*
IoT	*Internet of Things*
LwIP	*Lightweight IP*
MCU	*Microcontroller Unit*
NTC	*Negative Temperature Coefficient*
OMM	Organização Mundial de Meteorologia
PCB	*Printed Circuit Board*
R-T	Resistência Elétrica e Temperatura
RoHS	*Restriction of Hazardous Substances*
SD	*Secure Digital*
SoC	*System on Chip*
TH	*Time High*
TL	*Time Low*
TLS	*Transport Layer Security*
UART	*Universal Asynchronous Receiver/Transmitter*
UERGS	Universidade Estadual do Rio Grande do Sul
V-I	Tensão e Corrente
VCC	Tensão de alimentação de um circuito integrado

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

O projeto desenvolvido tem por objetivo o desenvolvimento de um sistema de medição de temperatura do solo, um parâmetro importante no manejo agrícola que influencia em grande proporção o crescimento das plantas. Esse sistema também é aplicável em outras áreas que necessitam de monitoramento de temperatura. Utilizam-se, como sensor, os coeficientes de temperatura negativo (NTC) TTC-203 integrados a um circuito temporizador CI 555, componentes comuns em lojas de eletrônica no Brasil. A caracterização dos sensores é apresentada detalhadamente no decorrer deste trabalho, ilustrando com tabelas, imagens e equações as respostas dos sensores às variações de temperatura. O tratamento dos sinais de onda quadrada gerados pelo CI 555 em modo astável também é descrito com detalhes, desde as implicações da escolha do capacitor de 100nF até a utilização de dois sensores para carga e descarga com Fator de Trabalho com baixa amplitude de variação para diferentes temperaturas. Esta configuração permite a conversão da variação de temperatura, que altera a resistência elétrica do NTC, em uma variação de frequência de uma onda quadrada, de 100Hz a 1.000Hz que será utilizada como sinal digital no microcontrolador. O desenvolvimento de software embarcado em hardware, utilizando o microcontrolador ESP32, inclui a implementação de filas, listas, interrupções de hardware, funções e *threads* para uma amostragem eficiente do sinal, facilitando futuras melhorias e aplicações, o código fonte em C é disponibilizado para análise e evolução. O design em placa de circuito impresso (PCB) incorpora o CI 555 e um sistema de alimentação autônomo baseado em baterias, desenvolvido de forma personalizada para aplicação prática em campo. Os resultados, obtidos em testes de laboratório e em campo, demonstram a eficiência e precisão do sistema na leitura da resistência dos sensores e na transformação destes dados em sinal de onda quadrada. Embora as medições do solo não forneçam resultados conclusivos, devido ao pequeno volume de dados coletados, a arquitetura do sistema se mostra robusta na medição da resistência. Melhorias potenciais e futuras aplicações do projeto são contempladas. Incluem-se aprimoramentos no *firmware* para reduzir o consumo de energia, utilizando o modo *DeepSleep* do microcontrolador ESP32 para minimizar o consumo energético durante períodos de inatividade. Considera-se também a implementação de um arquivo de configurações e parâmetros, permitindo o uso de diferentes tipos de sensores resistivos e mudança da forma de coleta de sinal, através de mudança de borda do sinal e não *pooling* como foi implementado. Além disso, sugere-se melhorias no hardware, como a utilização do RST do CI555 durante o *DeepSleep*, para aumentar a eficiência energética.

Palavras-Chave: Termistor NTC, Temperatura do Solo, Monitoramento, Microcontrolador.

ABSTRACT

The developed project aims to develop a soil temperature measurement system, an important parameter in agricultural management that greatly influences plant growth. This system is also applicable in other areas that require temperature monitoring. Negative Temperature Coefficient (NTC) sensors, specifically TTC-203, are used and integrated into a CI 555 timer circuit, components commonly found in electronics stores in Brazil. The characterization of the sensors is presented in detail throughout this work, illustrating with tables, images, and equations the sensors' responses to temperature variations. The processing of square wave signals generated by the CI 555 in astable mode is also described in detail, from the implications of choosing a 100nF capacitor to the use of two sensors for charge and discharge with a Duty Cycle with low amplitude variation for different temperatures. This configuration allows the conversion of temperature variation, which changes the electrical resistance of the NTC, into a frequency variation of a square wave, from 100Hz to 1,000Hz, which will be used as a digital signal in the microcontroller. The development of embedded software in hardware, using the ESP32 microcontroller, includes the implementation of queues, lists, hardware interrupts, functions, and threads for efficient signal sampling, facilitating future improvements and applications, and the source code in C is made available for analysis and evolution. The design on a printed circuit board (PCB) incorporates the CI 555 and an autonomous battery-based power system, custom-developed for practical field application. The results, obtained from laboratory and field tests, demonstrate the system's efficiency and precision in reading sensor resistance and transforming this data into a square wave signal. Although soil measurements do not provide conclusive results, due to the small volume of data collected, the system's architecture proves robust in measuring resistance. Potential improvements and future applications of the project are contemplated. These include enhancements in the firmware to reduce energy consumption, using the ESP32 microcontroller's DeepSleep mode to minimize energy consumption during periods of inactivity. The implementation of a configuration and parameter file is also considered, allowing the use of different types of resistive sensors and changing the way of signal collection, through edge change rather than pooling as implemented. Additionally, improvements in the hardware are suggested, such as the use of the RST of the CI555 during DeepSleep, to increase energy efficiency.

Keywords: NTC Thermistor, Soil Temperature, Monitoring, Microcontroller.

1 INTRODUÇÃO

A temperatura do solo é uma grandeza física de extrema importância para agricultura e ecologia pelo papel que desempenha nas interações solo-planta. Segundo RICIERI (2005) Essa importância está relacionada às influências em inúmeros processos, destacando-se a germinação das sementes, o desenvolvimento e a atividade das raízes em absorver água e nutrientes do solo, a atividade de microrganismos, a difusão de solutos e gases, o desenvolvimento de moléstias, a velocidade das reações químicas do solo. Muitos autores têm demonstrado relações entre o desenvolvimento e a produção das culturas agrícolas e a temperatura do solo, inclusive a possibilidade de poder-se estimar a duração do subperíodo semeadura-emergência a partir do conhecimento do regime térmico do solo.

A influência da insolação local, orientação geográfica e outros fatores semelhantes torna a temperatura do solo consideravelmente diferente da temperatura do ar RICIERI (2005). MOTA (1983) apontou que, para a vida vegetal, a temperatura do solo é mais crítica do que a do ar. Por exemplo, em regiões tropicais, temperaturas elevadas do solo podem ser prejudiciais para certas culturas, que têm um desenvolvimento ótimo em faixas de temperatura específicas e podem não crescer adequadamente se essas condições ideais não forem atendidas. A diferença entre a temperatura ideal e as temperaturas extremas pode ser crítica para o sucesso de muitos tipos de plantações. Essa dinâmica da temperatura do solo é fundamental para a agricultura, pois temperaturas desfavoráveis durante a estação de crescimento podem atrasar ou prejudicar as colheitas. Horticultores, em particular, valorizam solos que aquecem rapidamente na primavera. Agricultores dedicam esforços significativos para modificar a temperatura do solo, e muitas vezes, o sucesso ou fracasso de suas atividades agrícolas está diretamente ligado às variações de temperatura do solo.

O processo de aquecimento e resfriamento do solo, influenciado pela intensidade da radiação solar, varia ao longo do dia e do ano, afetando as camadas de solo abaixo RICIERI (2005). BERGAMASCHI e GUADAGNIN (1993) notaram que as variações de temperatura do solo são marcadamente menores nos primeiros centímetros de profundidade, com um atraso observado na ocorrência dos picos de temperatura máxima e mínima. Este fenômeno é explicado pela intensidade e pela lentidão do fluxo de calor no solo. Complementarmente, a Organização Mundial de Meteorologia - OMM (1971) estabelece que as profundidades padrão para medição da temperatura do solo são 10, 20, 50 e 100 cm. VAREJÃO-SILVA (2006) acrescenta que as medições abaixo da superfície do solo são geralmente mais precisas do que as acima da superfície, devido à capacidade do solo de reter calor e suavizar rápidas variações de temperatura, que são influenciadas pela radiação solar e tendem a apresentar um atraso temporal de aproximadamente uma hora.

KLAR (1974) destaca que as variações na temperatura do solo, especialmente em suas camadas mais superficiais, são em grande parte influenciadas pelas trocas de calor com o ar. Essas oscilações são continuamente moldadas pelo ciclo de radiação, provocando flutuações diárias notáveis nos primeiros trinta centímetros do solo descoberto. O aquecimento e resfriamento do solo ocorrem devido ao balanço energético entre a superfície do solo e a atmosfera. Esse processo resulta na dispersão de uma onda de calor para camadas mais profundas do solo por condução e na transferência de calor para a atmosfera, tanto por condução quanto, principalmente, por convecção, como explicam AZEVEDO e GALVANI (2003).

Dada a relevância da temperatura do solo, este trabalho visa desenvolver um sistema integrado de hardware e software para medir a temperatura do solo. A fundamentação teórica abrange dois temas principais: temperatura e sinais. Na seção sobre temperatura, discute-se desde o conceito básico de temperatura até o funcionamento dos termistores NTC, que alteram sua resistência de acordo com a variação da temperatura, conforme especificado em THINKING ELECTRONIC INDUSTRIAL CO., LTD. (2020). Na parte de geradores e condicionadores de sinais, a transformação de um sinal analógico em digital é explicada de maneira prática, utilizando o circuito temporizador 555 em modo multivibrador astável, além de uma introdução ao microcontrolador ESP32. Embora não seja possível detalhar todos os aspectos que envolvem o trabalho, os elementos principais são discutidos, comentados e referenciados de forma coesa.

O desenvolvimento do projeto é dividido em seis partes detalhadas: 1) Proposta de Solução Técnica, descrevendo os principais componentes do sistema, incluindo a fonte de alimentação, o microcontrolador ESP32 como núcleo lógico, os sensores NTC para detecção de variações de temperatura e o Circuito Oscilador com CI 555 para medição das alterações na resistência dos sensores; 2) Caracterização dos Sensores NTC, com testes em laboratório e análise de dados; 3) Implementação do Circuito Oscilador com CI 555, para converter sinais analógicos em digitais; 4) Desenvolvimento do Firmware no Microcontrolador ESP32; 5) Prototipagem e Ensaios, integrando o hardware e o software com uma PCB personalizada e testes com a ESP32; 6) Versão 2 de Protótipo para Testes em Campo, incluindo a adição de três sensores e novas funcionalidades no hardware e no firmware, com testes funcionais em condições reais de campo.

2 FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica se concentra em explorar dois temas que são fundamentais para o projeto: temperatura e sinais elétricos. A temperatura é abordada de uma forma ampla e geral, gradualmente se concentrando nos termistores NTC e na maneira como eles permitem medir variações de temperatura através de mudanças na resistência elétrica. Posteriormente, o trabalho aprofundou o estudo de geradores e condicionadores de sinais elétricos objetivando transformar um sinal analógico, como o gerado pela variação da resistência nos termistores NTC, em um sinal digital mensurável em frequência. Esta conversão é usada para aferir com exatidão as diferenças de resistência e, por consequência, as variações de temperatura.

2.1 Temperatura

"A temperatura influencia todos os fenômenos físicos naturais, bem como processos fisiológicos, tecnológicos e térmicos. É um dos parâmetros mais importantes em qualquer tipo de pesquisa." MICHALSKI (2001)

James Clerk Maxwell conceitua a temperatura como a quantificação do estado térmico de um corpo e sua capacidade de transferir calor. A reflexão detalhada de Maxwell fornece o arcabouço para o entendimento contemporâneo de temperatura, destacando que a sensação térmica é influenciada por múltiplos fatores. A necessidade de métodos de medição objetivos e precisos foi ressaltada, impulsionando o desenvolvimento de escalas de temperatura numéricas MAXWELL (1904).

Medir a temperatura é importante em pesquisa e desenvolvimento nas áreas de ciência e tecnologia, uma vez que uma vasta gama de propriedades físicas e químicas exibem variação dependente da temperatura. Conforme evidenciado no trabalho de GUADAGNINI et al. (2005), este conceito se estende também à importância da medição de temperatura no controle de processos, onde ela atua como uma variável significativa. Ao longo do tempo, diversos instrumentos para medir a temperatura têm sido criados, empregando sensores elétricos e não elétricos, visando obter leituras mais precisas para os propósitos aos quais se aplicam GUADAGNINI et al. (2005).

2.1.1 Escala Termodinâmica de Temperatura

A Escala Termodinâmica de Temperatura busca representar numericamente a intensidade do calor e as relações térmicas entre diferentes objetos e eventos MICHALSKI (2001). Dada a inexistência de métodos diretos e as limitações de certos termômetros, surge a necessidade de uma escala de temperatura universal. A Escala Kelvin Termodinâmica, fundamentada no ciclo reversível ideal de Carnot, atende a essa necessidade, atribuindo valores específicos de temperatura termodinâmica a estados térmicos conhecidos e sendo independente da substância termométrica utilizada MAXWELL (1904).

2.1.2 Sensores de Temperatura

Um sensor de temperatura é essencial numa cadeia de medição e instrumentação térmica. Segundo MICHALSKI (2001), eles podem ser classificados em duas categorias: o *'Self-Sustaining Cross-Converter'*, termo introduzido por MC-GHEE et al. (1999), que é um componente capaz de transformar a informação de entrada em informação de saída sem a necessidade de energia externa; e o *'Modulator',* como os termistores — resistores que variam com a temperatura e que necessitam de uma energia de suporte (Es) para operar. A Figura 2.1 ilustra essas duas categorias, mostrando que, enquanto o *'Modulator'* requer energia de suporte, o *'Self-Sustaining Cross-Converter'* opera independentemente. No entanto, ambos são suscetíveis a interferências de energia ou informação de ruído (Ec/Ic). A energia ou informação de saída de cada bloco é denotada como Eo ou Io, respectivamente.

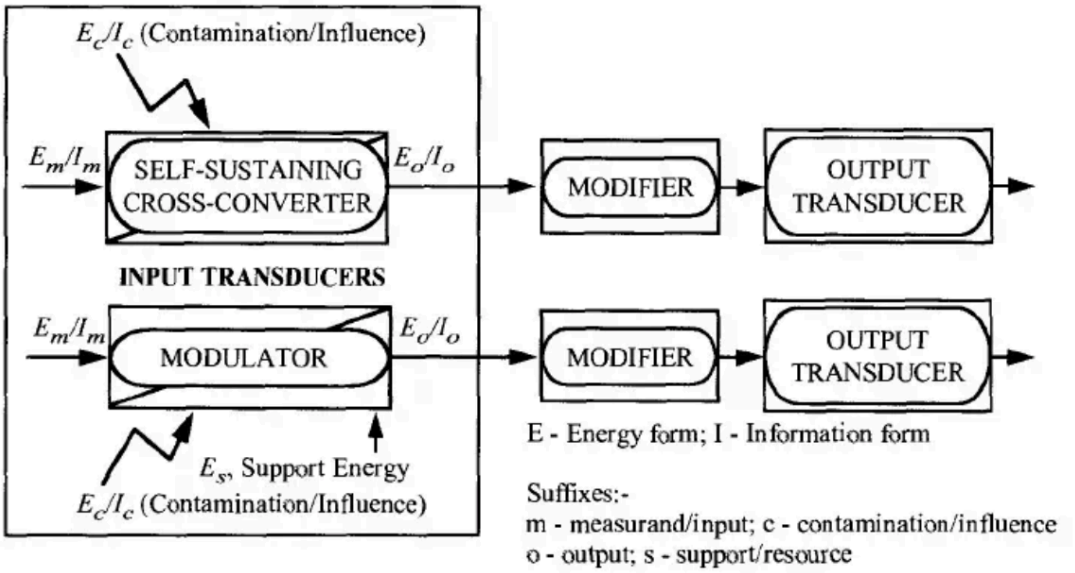


Figura 2.1: Um diagrama de blocos de medição de temperatura. MICHALSKI (2001, p. 12)

2.1.2.1 Função e Classificação dos Sensores

Os sensores de temperatura, como descritos por MC-GHEE et al. (1999), ampliam nossa capacidade de perceber relações térmicas entre entidades, transformando informações de temperatura em outras grandezas físicas. Esses sinais de medição, após serem obtidos, são modificados por conversores de dados, amplificadores, filtros, ou outros condicionadores para produzir o sinal de saída desejado MICHALSKI (2001).

Na diversificada área da termometria, a classificação dos sensores pode ser realizada de diversas maneiras, conforme evidenciado por diferentes pesquisadores. Este estudo alinha-se principalmente às metodologias de MC-GHEE et al. (1993) e MC-GHEE et al. (1996, 1999), que estruturam os sensores segundo critérios como função, estrutura, forma de energia e circuitos de condicionamento.

2.1.2.2 Mecanismo de Transferência de Calor

Inspirada na forma como percebemos o calor, essa classificação agrupa os sensores pela forma de transferência de calor, distinguindo entre métodos de contato e não contato. Tal distinção é vital, abarcando métodos diretos, que medem variáveis

associadas ao fluxo de energia térmica, e métodos inferenciais, que avaliam propriedades térmicas utilizando energia externa MC-GHEE et al. (1999).

2.1.2.3 Diagramas e Foco do Estudo

Para elucidar a classificação, são frequentemente empregados diagramas-chave e dendrogramas MICHALSKI (2001), que evidenciam a hierarquia e categorização dos sensores, demonstrando a diversidade e complexidade dos dispositivos, ver na Figura 2.2.

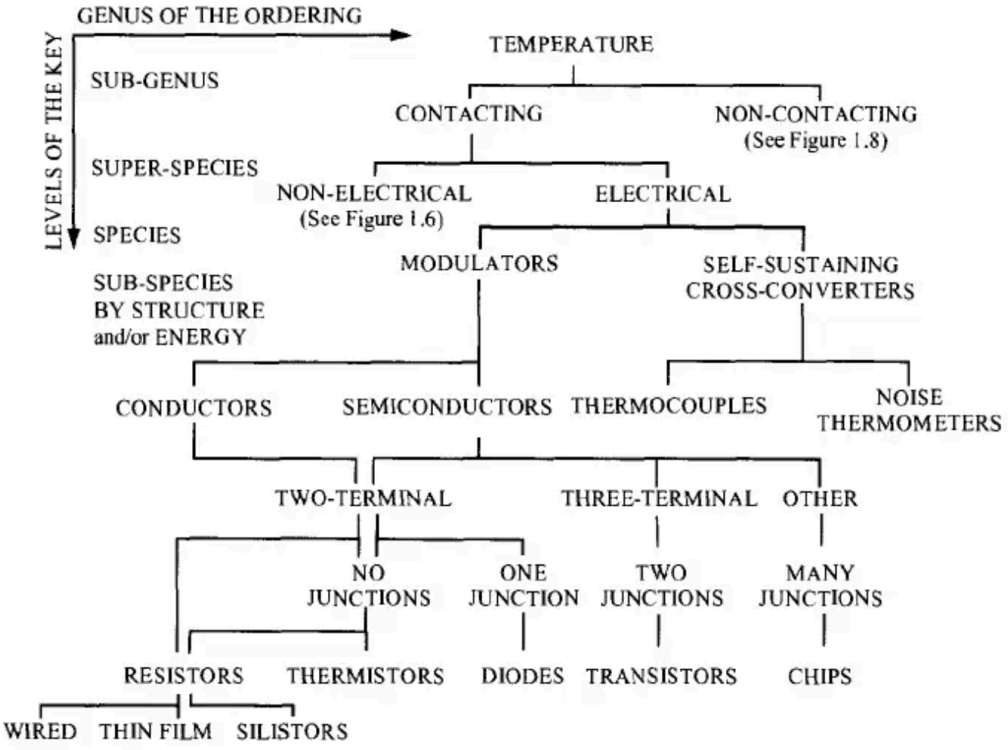


Figura 2.2: Classificação de sensores elétricos de temperatura de contato. MICHALSKI (2001, p.16)

O foco deste estudo é sobre sensores de temperatura de contato, especificamente termistores, que são de natureza resistiva. A escolha por esses sensores é devido à versatilidade e aplicabilidade em meio à vasta gama de dispositivos de medição, conforme ilustrado pela classificação na literatura.

2.1.3 Termistores NTC

Originado do descritor em inglês THERMally Sensitive ResISTOR (Resistor Sensível ao Calor), o termistor apresenta dois tipos fundamentais: Coeficiente Negativo de Temperatura (NTC) e Coeficiente Positivo de Temperatura (PTC). De acordo com BAKER (1999), o termistor NTC é ideal para medições de temperatura com precisão, ao passo que o PTC é mais adequado para aplicações de comutação. BAKER (1999) também destaca que o termistor NTC pode ser empregado em três diferentes modos de operação, cada um servindo a uma variedade de aplicações. Um desses modos explora as características de resistência do termistor em relação à temperatura, enquanto os demais modos se beneficiam das características de tensão versus corrente e de corrente ao longo do tempo do dispositivo.

A história dos termistores NTC remonta ao trabalho de Faraday em 1833, onde ele observou uma particularidade no sulfeto de prata, Ag2S, cuja resistência diminuía com o aumento das temperaturas. Essa observação levou ao reconhecimento do potencial comercial desse fenômeno, principalmente a partir de 1930, com as propostas inovadoras de Samuel Ruben, fundador da Duracell. Ao longo do século 20, os termistores NTC ganharam destaque em diversas aplicações industriais, chegando a um volume de mercado impressionante de 400 milhões de euros no último ano registrado FEITEIRA et. al (2010).

Conforme descrito em INSTRUMENTAÇÃO E TÉCNICAS DE MEDIDAS (2017), os NTC são os termistores mais comuns para medidas de temperatura, mas também podem ser empregados com base no seu auto aquecimento. Quando funcionam como um medidor de temperatura, a resistência de um termistor NTC pode ser descrita aproximadamente por uma exponencial, conforme a Equação 2.1, onde: R(T) é a resistência em função da temperatura (T); R0 é a resistência de referência, geralmente a resistência medida em kΩ a 25ºC; constante de Euler (e) elevada na potência de β multiplicando a diferença do inverso da temperatura variável (T) e temperatura de referência (T0):

$$R(T) = R_0 \times e^{\beta\left(\frac{1}{T} - \frac{1}{T_0}\right)} \quad (2.1)$$

A relevância do termistor NTC em comparação com outras tecnologias, como termopares metálicos, é evidenciada por sua maior sensibilidade e precisão, além de ser financeiramente mais viável. Esse termistor oferece uma sensibilidade superior, com coeficientes de temperatura negativos de resistividade de até -6% / °C. FEITEIRA et al. (2010).

Em termos matemáticos, o valor de β, uma constante importante que representa a temperatura característica do termistor, pode ser calculado pela Equação 2.2, onde: (R) é a resistência variável e (R0) é a resistência referência, usualmente é a resistência na temperatura ambiente, que é de 25ºC.

$$\beta = \frac{ln\left(\frac{R}{R_0}\right)}{\left(\frac{1}{T} - \frac{1}{T_0}\right)} \quad (2.2)$$

Este valor é essencial, pois determina a resposta do termistor NTC a variações de temperatura e é determinado pelas características intrínsecas do material do termistor em THE ORGANIC CHEMISTRY TUTOR (2020) demonstra o cálculo de maneira prática.

2.1.3.1 Especificações TTC-203

O termistor NTC de modelo TTC203, segundo o datasheet THINKING ELECTRONIC INDUSTRIAL CO., LTD. (2020), apresenta diversas características que o tornam atrativo para uma ampla gama de aplicações. Este componente está em conformidade com as diretrizes RoHS, significando que sua fabricação não envolve substâncias perigosas. Adicionalmente, versões que são livres de halogênio (HF) estão disponíveis, refletindo um compromisso com a compatibilidade ambiental. Com um

corpo de tamanho Φ5mm revestido em resina, o TTC203 é projetado para ser durável e confiável. Seus terminais são do tipo radial, ideais para integração em diversas interfaces eletrônicas. Quanto às especificações operacionais e elétricas do TTC203, destacam-se:

- **Resistência à Temperatura de 25°C (R25):** 20 kΩ. Esta resistência possui variações de tolerância de 5%, 10% e 15%, dependendo da especificação;
- **Coeficiente de Temperatura (Beta)**, avaliado no intervalo de 25 / 50°C: 4250. Este valor é fundamental para entender como a resistência do termistor varia com a temperatura;
- **Dissipação Máxima de Potência a 25°C:** 450mW, indicando a potência máxima que o termistor pode dissipar em condições padrão sem sofrer danos;
- **Constante de Tempo Térmico:** 20 segundos, referente ao tempo em que o termistor responde a 63,2% de uma mudança de temperatura;
- **Faixa de Temperatura Operacional:** de -30°C a +125°C, garantindo confiabilidade em um vasto range de ambientes.

Curvas características "R-T" e "V-I", representativas do comportamento do termistor em variadas condições, são providas para uma compreensão detalhada de sua operação e para auxiliar em sua correta implementação THINKING ELECTRONIC INDUSTRIAL CO., LTD. (2020).

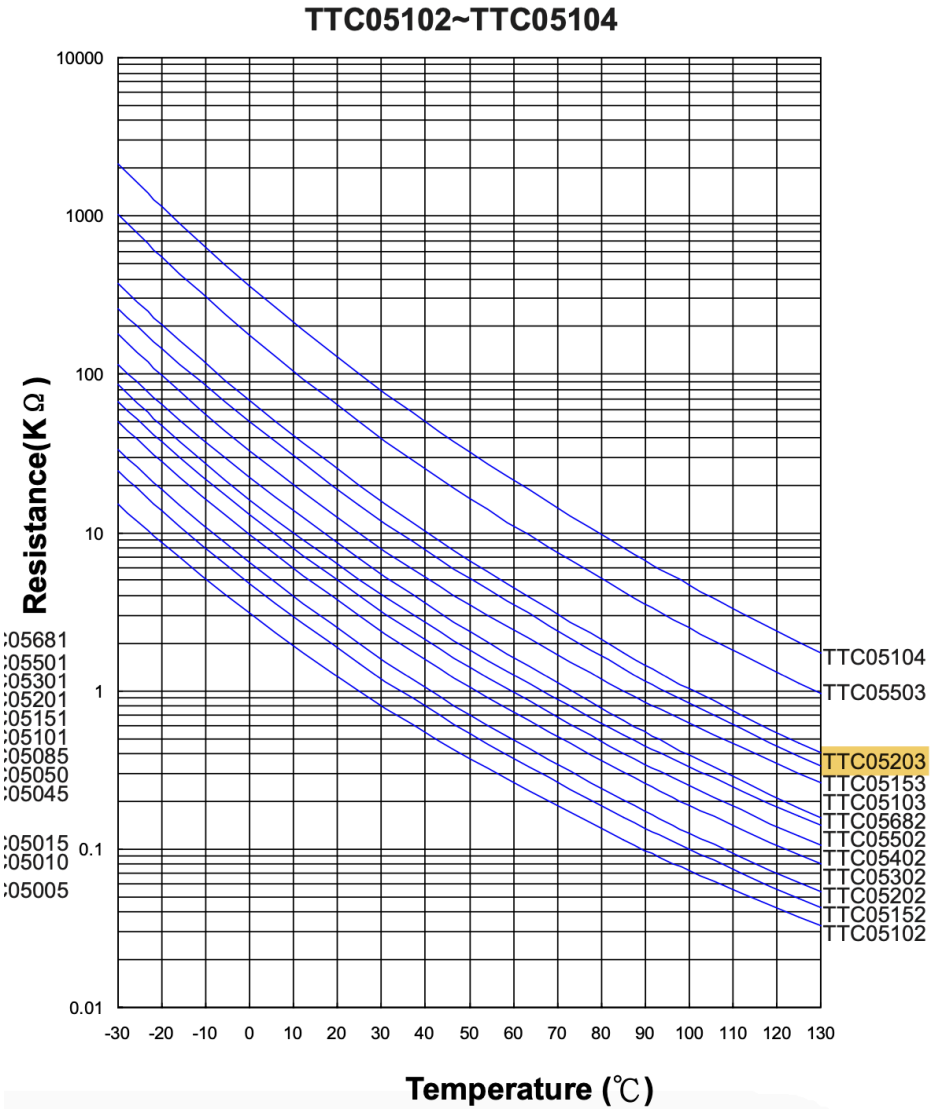


Figura 2.3: Curvas características R-T. THINKING ELECTRONIC INDUSTRIAL CO., LTD. (2020, p. 33)

A Figura 2.4 demonstra a curva característica da tensão (V) em função da corrente (I) da família de termistores TTC segundo o datasheet do fabricante.

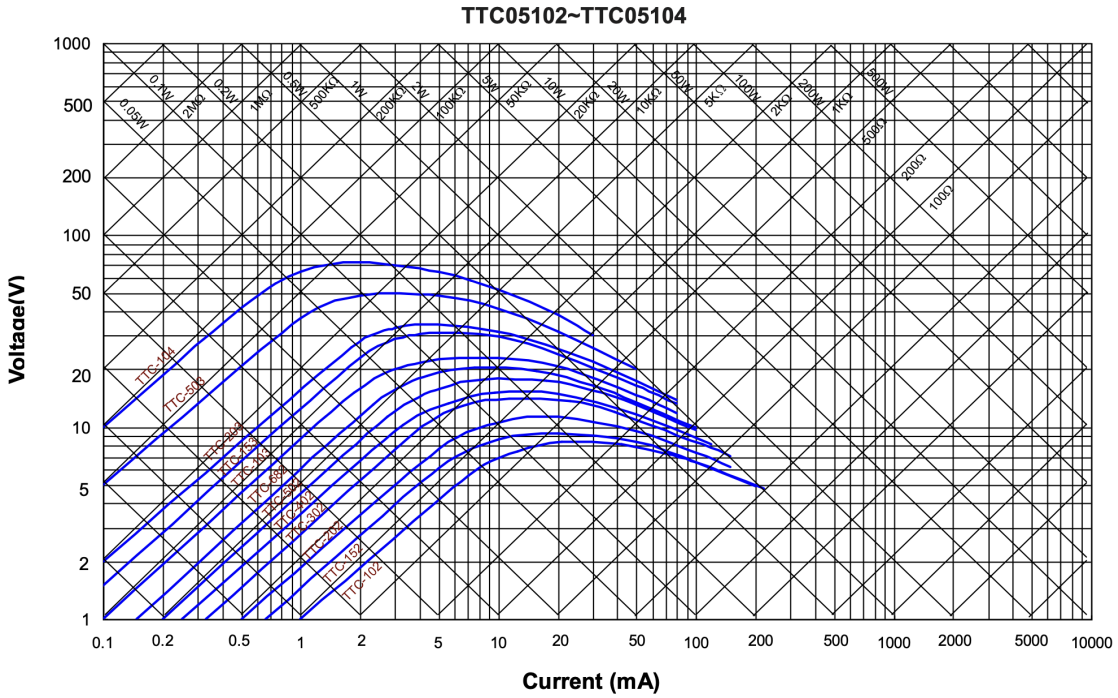


Figura 2.4: Curvas características V-I. THINKING ELECTRONIC INDUSTRIAL CO., LTD. (2020, p. 34)

2.1.3.2 Linearização de Termistores

A resposta característica de um termistor em relação à temperatura é intrinsecamente não linear. Embora essa característica possa ser desejável em algumas aplicações, em muitos casos, essa não linearidade pode complicar o processo de leitura ou introduzir erros inaceitáveis, especialmente em aplicações que demandam alta precisão. Para enfrentar essa questão, várias técnicas de linearização têm sido propostas e empregadas ao longo dos anos.

Conforme descrito no livro de Instrumentação e Técnicas de Medida da INSTRUMENTAÇÃO E TÉCNICAS DE MEDIDAS (2017), uma das abordagens mais comuns envolve a associação de um termistor, cuja resistência varia com a temperatura R(T), em paralelo com um resistor de valor fixo (Rp). A resistência resultante desta combinação é dada pela Equação 2.3, onde (Rp) é a resistência colocada em paralelo com o NTC que tem uma resistência representada por (Rt).

$$R(T) = \frac{R_P \cdot R_T}{R_P + R_T} \qquad (2.3)$$

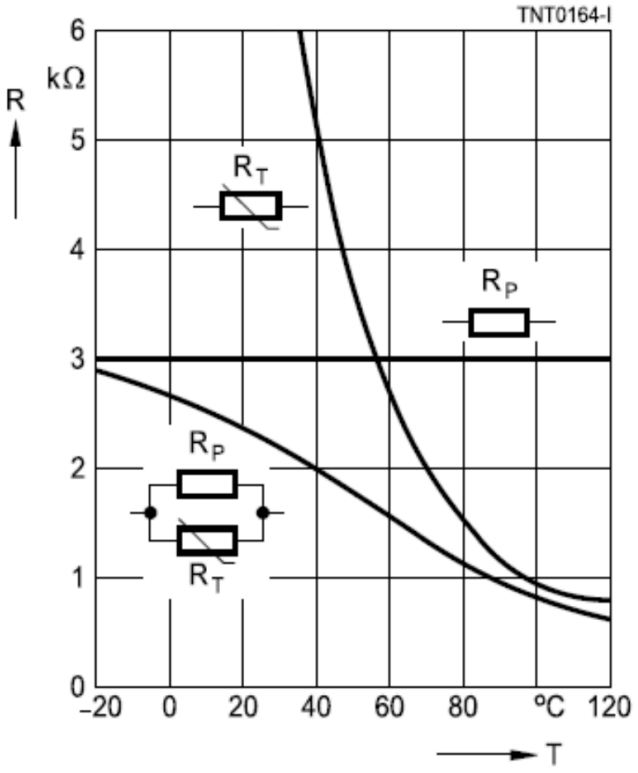


Figura 2.5: Linearização de NTC com resistência em paralelo. INSTRUMENTAÇÃO E TÉCNICAS DE MEDIDAS (2017 p.49)

Para realizar uma linearização simples em torno de uma única temperatura, busca-se um ponto de inflexão na curva de resistência (R), que ocorre quando a primeira derivada da resistência em relação à temperatura é mínima. A Equação (2.4) define essa derivada, levando em conta a resistência de calibração (Rp). A condição de ponto de inflexão é satisfeita quando a segunda derivada da resistência com relação à temperatura, dada pela Equação (2.5), é igual a zero na temperatura de calibração (Tc). Para determinar o valor de (Rp) apropriado para essa temperatura, utiliza-se a Equação (2.6), que relaciona (Rp) com (Tc), a constante β e a temperatura de referência (T0). Esta abordagem simplifica a relação resistência-temperatura próxima a (Tc) , permitindo uma aproximação linear adequada para medições de temperatura.

$$\frac{dR}{dT} = \frac{R_P^2}{\left(R_T + R_P\right)^2} \cdot \frac{dR_t}{dT} \qquad (2.4)$$

$$\frac{d^2R}{dT^2}\Big|_{T=T_c} = 0 \qquad (2.5)$$

$$R_p = R_{TC} \cdot \frac{\beta - 2 \cdot T_c}{\beta + 2 \cdot T_c} \qquad (2.6)$$

Uma forma mais abrangente de linearização que engloba uma faixa de operação pode ser alcançada para qualquer função não linear, garantindo que variações iguais de

temperatura resultem em variações equivalentes na resistência. Assim, para temperaturas extremas T1 (a mais alta) e T3 (a mais baixa), é possível determinar relações entre as variações de resistência e temperatura que maximizem a linearidade da resposta, conforme descrito nas Equações 2.7, 2.8, 2.9 e 2.10.

$$T_1 - T_2 = T_2 - T_3 \quad (2.7)$$

$$R_{T1} - R_{T2} = R_{T2} - R_{T3} \quad (2.8)$$

$$\frac{R_p \cdot R_{T1}}{\left(R_p + R_{T1}\right)} - \frac{R_p \cdot R_{T2}}{\left(R_p + R_{T2}\right)} = \frac{R_p \cdot R_{T2}}{\left(R_p + R_{T2}\right)} - \frac{R_p \cdot R_{T3}}{\left(R_p + R_{T3}\right)} \quad (2.9)$$

$$R_p = \frac{R_{T2} \cdot \left(R_{T3} + R_{T1}\right) - 2 \cdot R_{T3} \cdot R_{T1}}{R_{T3} + R_{T1} - 2 \cdot R_{T2}} \quad (2.10)$$

2.2 Geradores de Sinais e Condicionamento de Sinais

No projeto de sistemas eletrônicos, segundo SEDRA (et.al 2007) há um aumento crescente na necessidade de sinais com formas de onda padronizadas, por exemplo, senoidal, triangular ou pulso. Dentre os sistemas que exigem sinais padronizados, podemos citar: sistemas de teste e de medição, nos quais os sinais, novamente com uma variedade de formas de ondas, são empregados para teste e caracterização dos dispositivos e circuitos eletrônicos. Há dois métodos bastante diferentes para a geração de senóides, talvez os mais comumente usados das formas de ondas padronizadas.

1. O **Método de Oscilação Ressonante** é uma técnica que usa realimentação positiva e circuitos ressonantes para gerar ondas senoidais, com a amplitude controlada por mecanismos não-lineares.
2. O **Método de Modelagem da Onda Triangular** começa com uma onda triangular, que é então transformada em uma onda senoidal através de circuitos específicos.

Além das senoides, os sistemas eletrônicos muitas vezes requerem outras formas de onda, como as quadradas, triangulares ou em pulso. Para atender a essa demanda, uma série de circuitos especializados foi desenvolvida, chamados de "circuitos formadores de ondas" ou "geradores de função" SEDRA (et.al 2007).

Os Multivibradores são um exemplo-chave desses circuitos especializados. Essencialmente, um multivibrador é um circuito eletrônico que oscila entre diferentes estados, e, com base em sua configuração, pode gerar diversas formas de onda padrão. Existem vários tipos de multivibradores, cada um com sua própria característica e aplicação SEDRA (et.al 2007).

Multivibradores Astáveis, por exemplo, não têm um estado estável, mas oscilam constantemente entre dois estados, produzindo uma forma de onda quadrada. Multivibradores Monostáveis têm um único estado estável e, quando perturbados, mudam temporariamente para um segundo estado antes de retornar ao estado estável. Isso é útil para gerar pulsos de duração específica. Por fim, Multivibradores Bistáveis

têm dois estados estáveis e podem ser alternados entre eles, servindo, por exemplo, como memória ou latches SEDRA (et.al 2007).

Para criar essas diferentes formas de onda, os multivibradores podem ser combinados com outros componentes e circuitos. Por exemplo, um multivibrador acoplado a um integrador pode transformar uma onda quadrada em triangular. Em muitos casos, os designs podem se beneficiar do uso de componentes integrados, como o Temporizador 555. Este é um chip versátil que, com a configuração correta, pode servir como base para uma variedade de geradores de função, desde osciladores até temporizadores. No final, seja através de design discreto ou componentes integrados, o objetivo é o mesmo: criar sinais com precisão e estabilidade, adequados para as necessidades do sistema eletrônico em questão SEDRA (et.al 2007).

2.2.1 Geração de Ondas Quadradas Usando Multivibrador Astável

Antes de serem explorados o multivibrador astável e o CI 555 para geração de ondas quadradas, é fundamental compreender o comportamento do multivibrador biestável. A biestabilidade é alcançada utilizando um amplificador DC que possui realimentação positiva e um ganho de malha que supera a unidade. Ao analisar o circuito apresentado na Figura 2.8 correspondente, observa-se que o amplificador, inicialmente, tende a saturar em direção a uma das tensões de alimentação (seja ela superior ou inferior). Esse comportamento depende tanto do ruído presente na entrada do amplificador operacional quanto do desvio de tensão inerente ao próprio AmpOp SEDRA (et.al 2007).

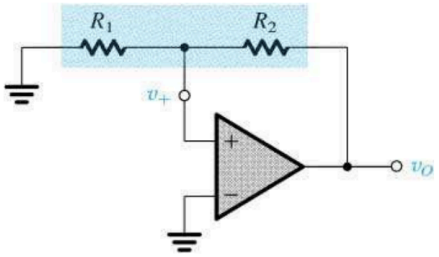


Figura 2.8: Uma realimentação positiva capaz de operação biestável. SEDRA (et.al 2007 p.743)

Um ponto crucial no circuito é o divisor de tensão composto por R_1 e R_2. Ele fornece uma fração da tensão de saída ao terminal positivo de entrada, determinada pela relação SEDRA (et.al 2007) na Equação 2.11:

$$\beta = \frac{R_1}{R_1+R_2} \qquad (2.11)$$

Se o produto $(A \cdot \beta)$ for maior que um, o que é comum nesses circuitos, desencadeia-se um processo regenerativo. Esse processo continua até que o amplificador operacional atinja sua saturação em um nível L_+. Assim, a tensão no terminal positivo de entrada, v_+, é dada por SEDRA (et.al 2007) na Equação 2.12:

$$v_+ = L_+ \frac{R_1}{R_1+R_2} \qquad (2.12)$$

Isso resulta em uma tensão positiva, mantendo o amplificador operacional na saturação positiva. Vale ressaltar que o amplificador também pode saturar em direção à tensão negativa. Nesse caso, as tensões seriam representadas por SEDRA (et.al 2007) na Equação 2.13 e 2.14:

$$v_0 = L_- \qquad (2.13)$$

$$v_+ = L_- \frac{R_1}{R_1+R_2} \qquad (2.14)$$

Conclui-se que o circuito apresenta dois estados ativos, e ambos podem ser mantidos indefinidamente. Contudo, existe um terceiro estado de equilíbrio instável, conforme mostrado na Figura 2.9, onde $v_+ = 0$ e $v_0 = 0$. Na presença de qualquer perturbação, como um desvio de tensão ou ruído, o circuito comuta para um dos estados ativos SEDRA (et.al 2007).

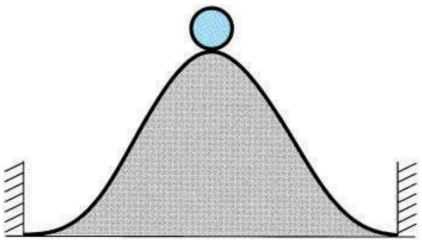


Figura 2.9: Uma analogia física para a operação do circuito biestável. A bola não consegue permanecer no topo do monte por certo período (um estado de equilíbrio instável ou metaestabilidade); um distúrbio inexoravelmente presente causará a queda da bola para um dos lados do monete, onde ela poderá permanecer indefinidamente (os dois estados estáveis). SEDRA (et.al 2007 p.743).

Para alterar o estado de um circuito biestável, pode-se utilizar qualquer um dos terminais conectados à terra como terminal de entrada. Dependendo de qual terminal for utilizado como entrada, pode resultar em uma configuração inversora ou não inversora. A configuração inversora é ilustrada na respectiva Figura 2.10.

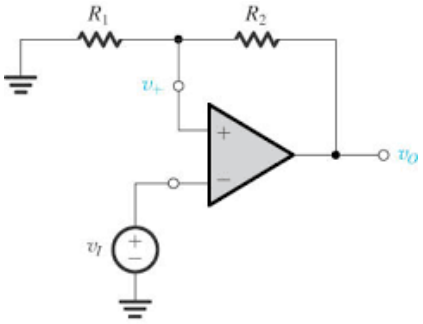


Figura 2.10: Um circuito biestável concebido a partir da malha de realimentação positiva da figura 2.8 aplicando-se v1 através de R1. SEDRA (et.al 2007 p.745).

Suponhamos que a saída esteja em L_+ e $v_+ = (\beta \cdot L_+)$. Se a tensão de entrada v_I aumentar ligeiramente acima de v_+, conforme indicado na Figura 2.11, v_0 passará para L_- devido ao ganho infinito do amplificador operacional. Consequentemente, $v_+ = (\beta \cdot L_-)$ e mantém-se assim para quaisquer valores de $v_+ > (\beta \cdot L_-)$.

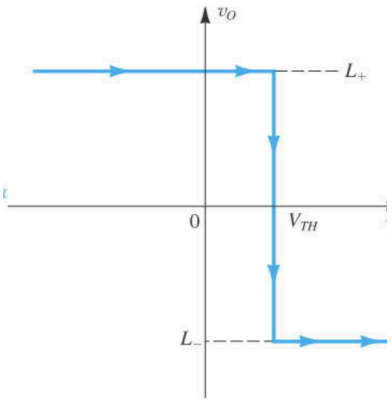


Figura 2.11: A característica de transferência do circuito em figura 2.10 para v_I crescente. SEDRA (et.al 2007 p.744).

Quando a tensão começa a diminuir, o multivibrador só retornará ao seu estado original quando $v_I > (\beta \cdot L_-)$, conforme mostrado na Figura 2.12.

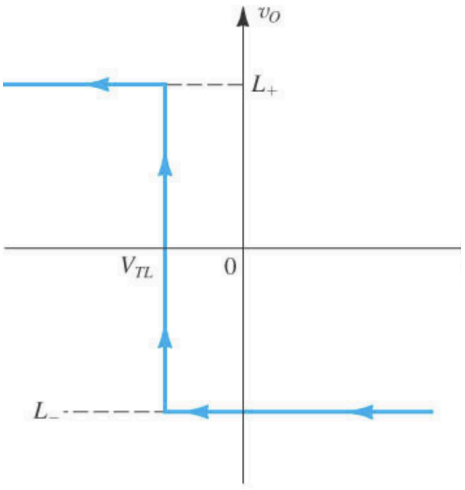


Figura 2.12: A característica de transferência para v_I decrescente. SEDRA (et.al 2007 p.744).

Este fenômeno, em que os níveis de v_I necessários para mudar o estado do multivibrador são diferentes dependendo de v_I estar aumentando ou diminuindo, é chamado de histerese. A relação entre v_0 e v_I para esse circuito é apresentada na Figura 2.13 da referência anterior.

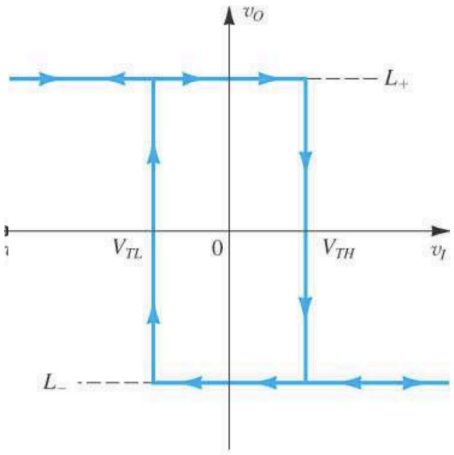


Figura 2.13: As características de transferência completas. SEDRA (et.al 2007 p.744).

Finalmente, é possível alterar o estado do multivibrador aplicando um impulso de curta duração. A amplitude deve ser maior que $\left(\beta \cdot L_+\right)$ (quando a saída está no nível alto) ou menor que $\left(\beta \cdot L_-\right)$ (quando a saída está no nível baixo).

Para gerar uma onda quadrada, é possível forçar o Multivibrador biestável a mudar de estado periodicamente. Uma das maneiras de conseguir isso é incluir um circuito RC na malha de realimentação. Vale ressaltar que este circuito, ao contrário do biestável, não possui estados estáveis, sendo, portanto, denominado circuito astável. Funcionamento:

- **Estado Inicial:** Considerando L_+ como o estado inicial na saída do amplificador operacional, o capacitor C carregará até L_+ via resistor R. A tensão v_- através do capacitor C aumentará exponencialmente, com uma constante de tempo τ=CR. Quando $v_- = v_+ = \left(\beta \cdot L_+\right)$, o Multivibrador biestável (configuração inversora) transiciona para o estado $v_0 = L_-$ e $v_+ = \left(\beta \cdot L_-\right)$. Neste ponto, o condensador começa a descarregar exponencialmente até L_-. Esse estado persiste até $v_- = v_+ = \left(\beta \cdot L_-\right)$, momento em que o multivibrador biestável volta para o estado positivo $v_0 = L_+$. Esse transicionamento é ilustrado na figura 2.14.

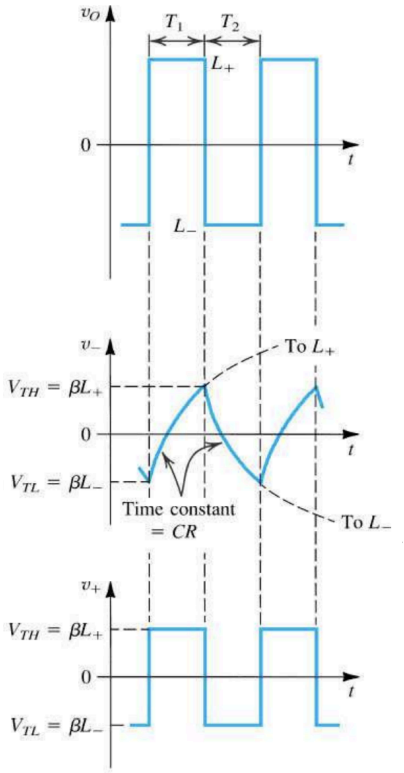


Figura 2.14: Formas de ondas em vários nós do circuito biestável. SEDRA (et.al 2007 p.747).

- **Durante a Carga do Condensador:**

$$v_- = L_+ - \left(L_+ - \beta \cdot L_-\right) \cdot e^{-\frac{t}{\tau}} \quad (2.15)$$

onde:

$$\tau = R \cdot C \quad (2.16)$$

Quando o multivibrador muda de estado em $t = T_1$, temos:

$$T_1 = \tau \cdot ln\left(\frac{1-\beta \cdot \frac{L_-}{L_+}}{1-\beta}\right) \quad (2.17)$$

- **Durante a Descarga do Condensador:**

$$v_- = L_- - \left(L_- - \beta \cdot L_+\right) \cdot e^{-\frac{t}{\tau}} \quad (2.18)$$

Substituindo $v_- = \left(\beta \cdot L_-\right)$ em $t = T_2$, obtemos:

$$T_2 = \tau \cdot ln\left(\frac{1-\beta \cdot \frac{L_+}{L_-}}{1-\beta}\right) \quad (2.19)$$

Se for considerado que $L_+ = -L_-$, a duração total do ciclo será $T = T_1 + T_2$:

$$T = 2 \cdot \tau \cdot ln\left(\frac{1-\beta}{1+\beta}\right) \quad (2.20)$$

Este gerador de onda quadrada permite a variação de frequência ao alternar diferentes capacitores (geralmente em décadas) e ajustando o resistor R de forma contínua, o que possibilita um controle contínuo da frequência. O circuito base que representa este gerador pode ser visualizado na Figura 2.15.

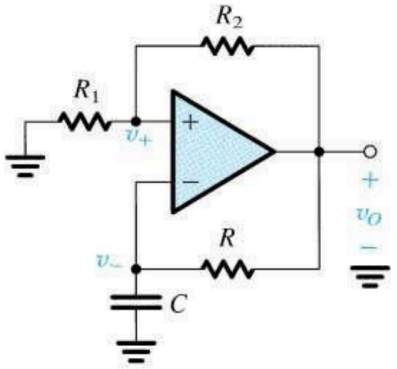


Figura 2.15: Esquema do circuito obtido quando o multivibrador biestável é implementado com o circuito da Figura 2.10. SEDRA (et.al 2007 p.747).

2.2.2 O Circuito do Temporizador 555

No âmbito da microeletrônica, diversos circuitos integrados temporizadores, encapsulados, são disponíveis, atuando como pilares na construção de multivibradores, tanto monoestáveis quanto astáveis. Entre estes, o temporizador 555 destaca-se devido à sua popularidade e versatilidade. Segundo SEDRA (et.al 2007), o 555 foi lançado em 1972 pela *Signetics Corporation* como um circuito bipolar. Atualmente, não só mantém sua presença em sua forma original bipolar, mas também na versão em tecnologia CMOS, sendo distribuído por diversos fabricantes.

A estrutura interna do 555 é ilustrada na Figura 2.16, onde pode-se observar sua organização em blocos. Este circuito é composto majoritariamente por dois comparadores, um *flip-flop SR*, e um transistor Q1 que atua como chave. Para sua operação adequada, é essencial uma fonte de alimentação, geralmente de 5V. Associado a essa alimentação, um divisor resistivo, formado por três resistores idênticos, R1, define as tensões limiares para os comparadores: (⅔)Vcc para o comparador 1 e (⅓)Vcc para o segundo.

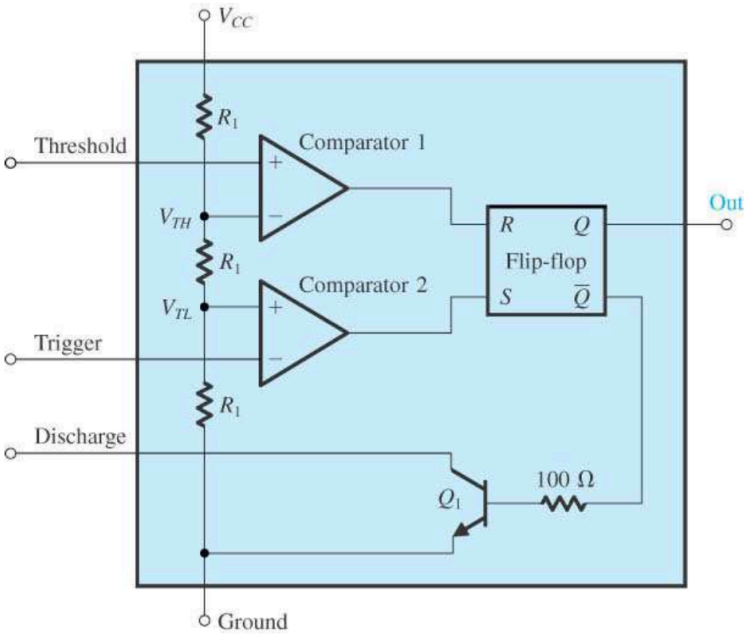


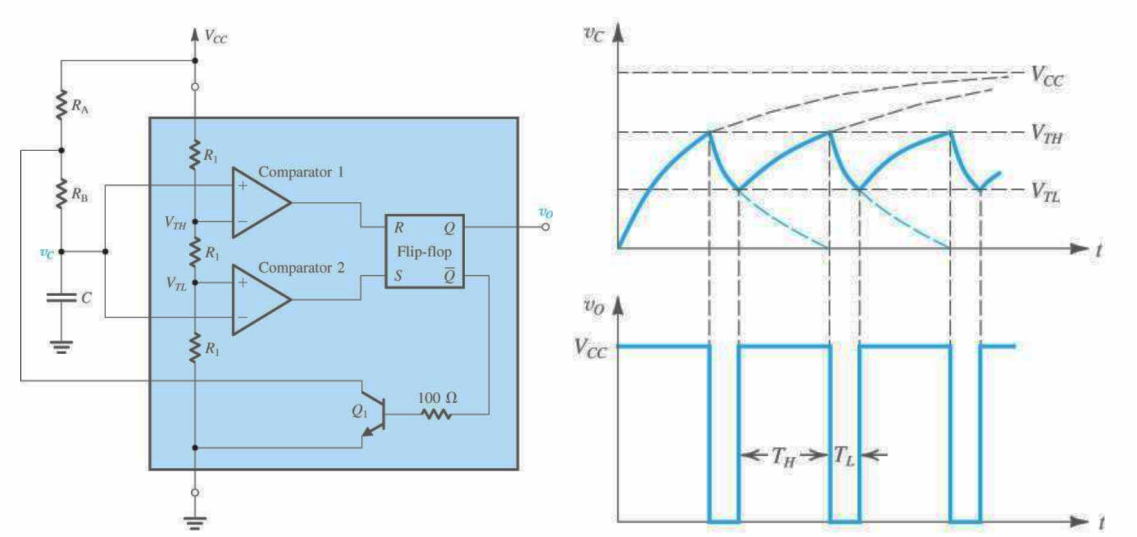
Figura 2.16: Diagrama de blocos representando o circuito interno de um circuito integrado temporizador do tipo 555. SEDRA (et.al 2007 p.751).

Um aspecto essencial do 555 é o *flip-flop SR*, também reconhecido como latch. Este é um circuito biestável com saídas distintas e complementares: Q e Q/. Em seu estado "*set*", Q torna-se "alta" (quase Vcc), enquanto Q/ é "baixa" (próxima a 0V). Em contrapartida, no estado "reset", a dinâmica se inverte. A transição entre esses estados é governada pelos terminais *set* (S) e *reset* (R). No 555, a interconexão destes terminais com os comparadores é direta: o terminal reset se liga ao comparador 1, e o set ao comparador 2.

A operacionalidade do 555 é aprimorada por seus terminais externos. Por exemplo, o terminal "Limiar" é conectado ao comparador 1, enquanto o "Disparo" está vinculado ao comparador 2. Além destes, o transistor Q1 tem seu coletor associado ao terminal "Descarga", e, consolidando a lógica do dispositivo, a saída Q do *flip-flop* dirige-se ao terminal "Saída" do 555.

2.2.2.1 Um Multivibrador Astável usando CI 555

A Figura 2.17 (a) ilustra a configuração de um multivibrador astável utilizando o circuito integrado 555, juntamente com dois resistores externos, Ra e Rb, e um capacitor externo C. Para compreender o funcionamento do circuito, consulte as formas de onda retratadas na Figura 2.17 (b).



(a) (b)

Figura 2.17: (a) O temporizador 555 conectado de forma a implementar um multivibrador. (b) Formas de onda do circuito em (a). SEDRA (et.al 2007 p.753).

Considerando que o capacitor C esteja inicialmente descarregado e que o *flip-flop* esteja no estado "*set*", a tensão de saída v_0 estará em nível alto, e o transistor Q1 estará em corte. Nesse cenário, o capacitor C começa a carregar através dos resistores R_a e R_b, fazendo com que a tensão V_c cresça exponencialmente em direção a Vcc. Quando V_c atinge o nível V_{TL}, a saída do comparador 2 torna-se baixa. No entanto, isso não afeta o funcionamento geral do circuito, e o *flip-flop* mantém-se no estado "set". Esse estado persiste até que V_c ultrapasse o limiar V_{TH} do comparador 1. Quando isso ocorre, a saída do comparador 1 torna-se alta, e o *flip-flop* é colocado no estado "reset", levando v_0 ao nível baixo, e Q1 passa a conduzir SEDRA (et.al 2007).

A ativação do transistor Q1 faz com que apareça uma tensão próxima a zero volts no ponto entre R_a e R_b, permitindo que C comece a descarregar através de R_b e do coletor de Q1. A tensão V_c então diminui exponencialmente com uma constante de tempo $C \cdot R_b$ até atingir o limiar V_{TL}. Ao cruzar este limiar, a saída do comparador 2 torna-se alta, e o *flip-flop* volta ao estado "*set*", iniciando um novo ciclo SEDRA (et.al 2007).

Segundo SEDRA (et.al 2007), o comportamento descrito faz com que o circuito oscile, produzindo uma forma de onda quadrada na saída. A frequência de oscilação é determinada observando a Figura 2.16 (b). Durante o intervalo Th, V_c aumenta de V_{TL} para V_{TH}. Esse aumento exponencial em V_c é descrito pela Equação 2.21:

$$V_c = Vcc - \left(Vcc - V_{TL}\right) \cdot e^{-\frac{t}{C \cdot \left(R_a + R_b\right)}} \quad (2.21)$$

Onde t=0 marca o início de Th. Substituindo os valores dados para V_c, obtemos a equação 2.22:

$$Th = 0.69 \cdot C \cdot \left(R_a + R_b\right) \quad (2.22)$$

Durante o intervalo Tl, V_c decresce de V_{TH} para V_{TL}, e sua queda exponencial é descrita pela equação 2.23:

$$V_c = V_{TH} \cdot e^{-\frac{t}{C \cdot R_b}} \quad (2.23)$$

Com t=0 no início de Tl. Daí, temos a equação 2.24:

$$Tl = 0.69 \cdot C \cdot R_b \quad (2.24)$$

Combinando as equações 2.22 e 2.24 para Th e Tl, obtemos o período dado pela equação 2.25:

$$T = Th + Tl = 0.69 \cdot C \cdot \left(R_a + 2 \cdot R_b\right) \quad (2.25)$$

Por fim, o fator de trabalho (Fator de Trabalho) é dado pela equação 2.26:

$$Fator\ de\ Trabalho = \frac{Th}{Th+Tl} = \frac{R_a+R_b}{R_a+2 \cdot R_b} \quad (2.26)$$

Notando que este fator será sempre superior a 50%. Tal condição é alcançada quando R_a é significativamente menor que R_b.

Segundo BRAGA (2023) Para modificar o Fator de Trabalho, tornando-o menor que 0,5, por exemplo, é possível inserir um diodo em paralelo com R_b. Sem o diodo, Th (tempo em que a saída é alta) está relacionado tanto a R_a quanto a R_b, visto que a carga do capacitor ocorre através da combinação em série de ambos os resistores. No entanto, com a adição do diodo, a carga do capacitor será feita apenas através de R_a devido à direção de polarização do diodo, enquanto a descarga será realizada apenas por R_b. Deste modo, as equações são simplificadas para as equações 2.27 e 2.28:

$$Th = 0.69 \cdot C \cdot R_a \quad (2.27)$$

$$Tl = 0.69 \cdot C \cdot R_b \quad (2.28)$$

A partir dessas relações, o Fator de Trabalho, que representa a proporção do período em que a saída é alta em relação ao período total, é dado pela equação 2.29:

$$Fator\ de\ Trabalho = \frac{Th}{Th+Tl} = \frac{R_a}{R_a+R_b} \quad (2.29)$$

Agora, alterando os valores de R_a e R_b de forma independente, é possível ajustar o Fator de Trabalho para qualquer valor entre 0 e 1. Isso oferece uma flexibilidade muito maior na definição do Fator de Trabalho do circuito oscilador.

Para validar o comportamento teórico do multivibrador astável, realizou-se uma simulação computacional utilizando o software *LTspice*, reconhecido pela precisão na simulação de circuitos integrados como o CI 555. A Figura 2.18 ilustra o esquemático do circuito, onde o CI 555 está configurado em modo astável, com os resistores R_a e R_b de 10kΩ e 15kΩ, respectivamente, e um capacitor C1 de 100nF.

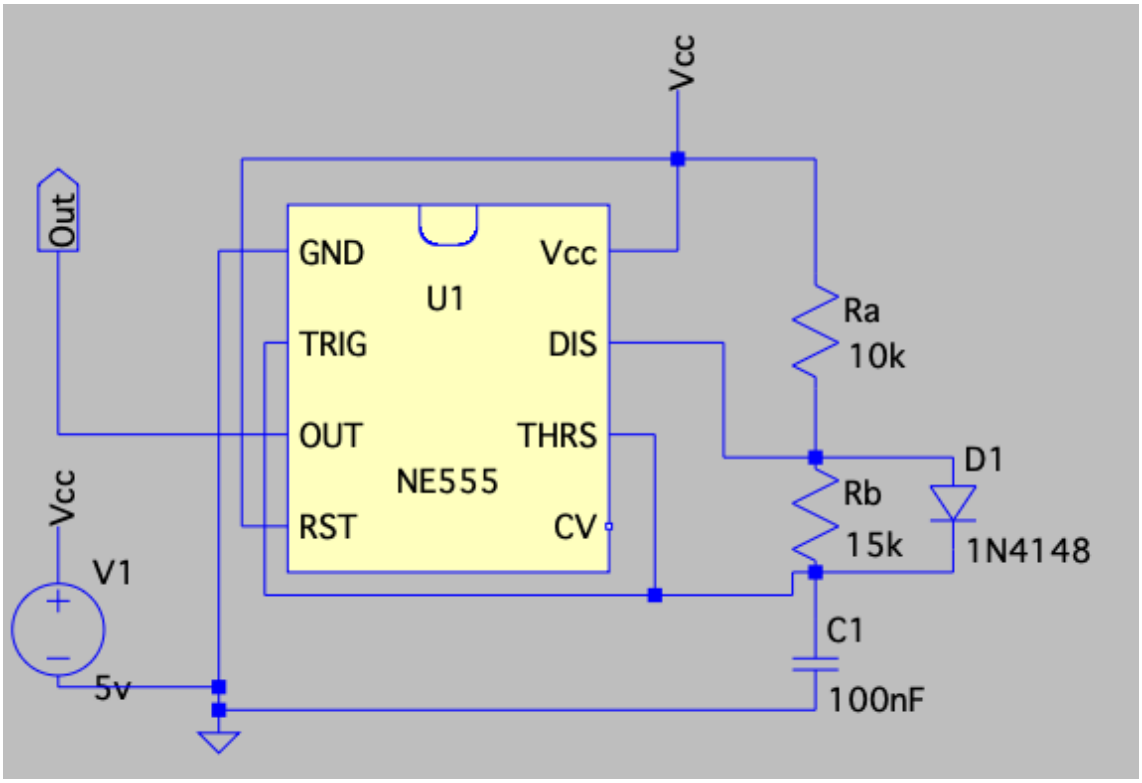


Figura 2.18: Esquemático de circuito configurado para funcionar como multivibrador astável em simulador computacional *LTspice*. Autor (2023).

Esta configuração destina-se a produzir uma onda quadrada oscilante na saída do CI, com a frequência e o ciclo de trabalho ajustáveis através dos valores de R_a, R_b e C1. Executou-se uma simulação transitória por 10ms, como mostrado na Figura 2.19, para analisar a saída oscilatória do circuito.

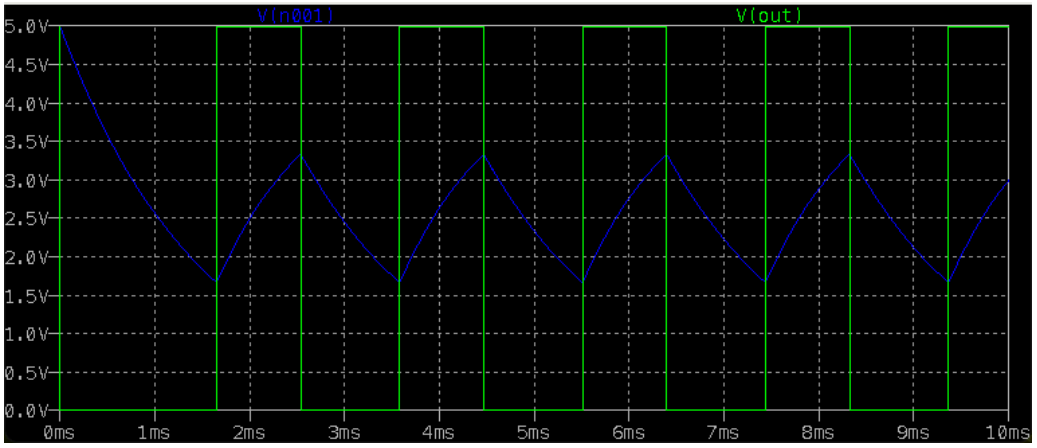


Figura 2.19: *Output* de circuito demonstrado na Figura 2.18. Autor (2023).

As simulações de circuitos eletrônicos podem divergir dos cálculos teóricos devido às tolerâncias dos componentes, comportamentos não ideais e efeitos parasitários. Com isso em mente, observou-se que:

Análise de TH:

$$Th_{LTSpice} = 0.000885836\,s \quad (2.30)$$

$$Th_{Teórico} = 0.693 \cdot R_a \cdot C1 = 0.693 \cdot 10^4 \cdot 10^{-7} = 0.000693\,s \quad (2.31)$$

A diferença entre $Th_{LTSpice}$ e o $Th_{Teórico}$ é:

$$\Delta Th = Th_{LTSpice} - Th_{Teórico} \quad (2.32)$$

$$\Delta Th = 0.000192836\,s \quad (2.33)$$

A resistência apresentada pelo diodo R_{Diodo}, assumindo que seja o único fator adicional afetando o Th, é:

$$R_{Diodo} = \left(\frac{Th_{LTSpice}}{0.693 \cdot C1}\right) - R_a \quad (2.34)$$

$$R_{Diodo} = \left(\frac{0.000885836}{0.693 \cdot 100 \cdot 10^{-9}}\right) - 10^4 \quad (2.35)$$

$$R_{Diodo} = 2782.63\Omega \quad (2.36)$$

Calculando novamente o $Th_{Teórico}$ levando em consideração o R_{Diodo} temos que:

$$Th_{Teórico\ com\ R_{Diodo}} = 0.693 \cdot \left(R_a + R_{Diodo}\right) \cdot C1 \quad (2.37)$$

$$Th_{Teórico\ com\ R_{Diodo}} = 0.693 \cdot \left(10^4 + 2782.63\right) \cdot 10^{-7} = 0.000885836\,s \quad (2.37)$$

Análise de TL:

$$Tl_{LTSpice} = 0.00104295\,s \quad (2.38)$$

$$Tl_{Teórico} = 0.693 \cdot R_b \cdot C1 \quad (2.39)$$

$$Tl_{Teórico} = 0.693 \cdot 15 \cdot 10^{3} \cdot 10^{-7} = 0.00103950s \quad (2.40)$$

A diferença entre $Tl_{LTSpice}$ (2.38) e o $Tl_{Teórico}$ (2.40) é pequena, e pode ser atribuída a arredondamentos ou a pequenas imprecisões nos modelos de simulação.

$$\Delta Tl = Tl_{LTSpice} - Tl_{Teórico} \quad (2.41)$$

$$\Delta Tl = 0.00104295s - 0.00103950s = 0.00000345\,s \quad (2.42)$$

Esta pequena diferença em TL pode ser considerada dentro da margem de erro esperada para simulações e componentes eletrônicos.

O fato de TL estar tão próximo do valor teórico indica que o modelo de simulação para o resistor Rb e o capacitor C1 é bastante preciso. No entanto, a maior discrepância em TH sugere que há fatores adicionais influenciando o carregamento do capacitor através de R_a que não são capturados pela fórmula teórica simples. A presença do "diodo 1N4148" no circuito, conforme indicado no arquivo de simulação do *LTSpice*, introduz uma resistência efetiva adicional que altera o tempo de carregamento do capacitor e, consequentemente, o tempo em que a saída permanece alta.

Considerando que o diodo está em polarização reversa durante o carregamento do capacitor, a resistência de fuga ou a resistência dinâmica no estado reverso do diodo poderiam ser responsáveis pela discrepância observada. A resistência efetiva calculada, aproximadamente 2782,63Ω, reflete a combinação da resistência intrínseca do diodo com possíveis efeitos parasitas e outros aspectos do modelo de simulação que não são contemplados na formulação teórica. Para uma compreensão detalhada da metodologia e das configurações utilizadas na simulação, o *script* do *LTSpice* está disponível no Apêndice A.

2.2.3 Microcontrolador ESP32

Conforme BABIUCH et al. (2019), o avanço acelerado da *IoT* e dos sistemas embarcados é impulsionado pela disponibilidade de módulos de hardware e software avançados. As placas de desenvolvimento, integrando interfaces de comunicação e periféricos ao processador principal, têm ganhado destaque. O chip ESP32, em particular, vem ganhando popularidade, evoluindo em variantes de hardware e desenvolvimento de software. Este chip, reconhecido pela comunidade de desenvolvedores e acadêmicos como sucessor do *ESP8266*, é amplamente empregado em diversas aplicações, conforme evidenciado em pesquisas científicas recentes.

Segundo o datasheet ESP32 WROOM (Espressif Systems, s.d.) o módulo ESP32-WROOM-32 é uma solução MCU versátil, que combina *Wi-Fi*, *Bluetooth*® e *Bluetooth LE*, direcionado para uma ampla gama de aplicações. Desde redes de sensores de baixo consumo até tarefas exigentes como codificação de voz, streaming de música e decodificação de MP3, sua flexibilidade é ampla. No núcleo do módulo está o chip ESP32-D0WDQ6, que é projetado para ser escalável e adaptável. O chip possui dois

núcleos de CPU que podem ser controlados individualmente, com a frequência do relógio da CPU ajustável de 80 MHz a 240 MHz. Além disso, o chip conta com um coprocessador de baixo consumo que pode ser utilizado para economizar energia, executando tarefas que requerem menos capacidade de processamento, como o monitoramento de periféricos.

O ESP32 integra um conjunto rico de periféricos, incluindo sensores de toque capacitivos, interface para cartão SD, *Ethernet*, SPI de alta velocidade, UART, I2S e I2C. Com a integração de *Bluetooth*, *Bluetooth LE* e *Wi-Fi*, o módulo é capaz de atender a uma ampla gama de aplicações, com a vantagem de usar o *Wi-Fi* para grande alcance físico e conexão direta à internet através de um roteador, enquanto o *Bluetooth* permite conexões convenientes a *smartphones* ou transmissão de beacons de baixa energia ESP32 WROOM (Espressif Systems, s.d.).

O consumo de energia em modo sleep do chip ESP32 é inferior a 5μA, o que o torna ideal para aplicações alimentadas por bateria e dispositivos vestíveis. O módulo suporta uma taxa de dados de até 150 Mbps e potência de saída de 20 dBm na antena, garantindo uma ampla gama física. Portanto, o ESP32-WROOM-32 oferece especificações líderes de indústria e o melhor desempenho em termos de integração eletrônica, alcance, consumo de energia e conectividade. O sistema operacional escolhido para o ESP32 é o *freeRTOS* com LwIP, e também conta com aceleração de hardware para TLS 1.2 ESP32 WROOM (Espressif Systems, s.d.).

Tabela 2.1: Especificações do ESP32-WROOM-32

Categories	Items	Specifications
Certification	RF certification	See certificates for ESP32-WROOM-32
	Wi-Fi certification	Wi-Fi Alliance
	Bluetooth certification	BQB
	Green certification	RoHS/REACH
Test	Reliablity	HTOL/HTSL/uHAST/TCT/ESD
Wi-Fi	Protocols	802.11 b/g/n (802.11n up to 150 Mbps)
		A-MPDU and A-MSDU aggregation and 0.4 μs guard interval support
	Center frequency range of operating channel	2412 ~ 2484 MHz
Bluetooth	Protocols	Bluetooth v4.2 BR/EDR and Bluetooth LE specification
	Radio	NZIF receiver with –97 dBm sensitivity
		Class-1, class-2 and class-3 transmitter
	Audio	AFH CVSD and SBC
Hardware	Module interfaces	SD card, UART, SPI, SDIO, I2C, LED PWM, Motor PWM, I2S, IR, pulse counter, GPIO, capacitive touch sensor, ADC, DAC, Two-Wire Automotive Interface (TWAI®), compatible with ISO11898-1 (CAN Specification 2.0)
	Integrated crystal	40 MHz crystal
	Integrated SPI flash	4 MB
	Operating voltage/Power supply	3.0 V ~ 3.6 V
	Operating current	Average: 80 mA
	Minimum current delivered by power supply	500 mA
	Recommended operating ambient temperature range	–40 °C ~ +85 °C
	Package size	18 mm × 25.5 mm × 3.10 mm
	Moisture sensitivity level (MSL)	Level 3

Fonte: ESP32 WROOM (Espressif Systems, s.d.)

3 DESENVOLVIMENTO DO TRABALHO

Os experimentos e procedimentos técnicos descritos nesta seção, importantes para a fase de prototipação do projeto, foram realizados no laboratório de eletrônica da Universidade Estadual do Rio Grande do Sul, na unidade de Guaíba, ao longo de 2023. Este laboratório, equipado com osciloscópio, estação de solda, placa PCB padrão, componentes eletrônicos entre outros equipamentos e instrumentos, proporcionou as condições ideais para a execução dos experimentos. No decorrer desta seção, serão detalhados os equipamentos e as condições ambientais específicas sob as quais os experimentos foram conduzidos, elucidando como cada elemento contribuiu para os resultados alcançados.

3.1 Proposta de Solução Técnica

A proposta técnica deste projeto é a criação de um sistema de medição de temperatura do solo sensível e precisamente calibrado. A solução visa integrar sensores termistores NTC em um circuito eletrônico otimizado, cuja resposta em frequência será cuidadosamente analisada e convertida em leituras de temperatura através de um microcontrolador ESP32 programado para realizar o tratamento de sinal. Esta seção explora a viabilidade, os desafios e as inovações que constituem o cerne deste dispositivo.

3.1.1 Objetivos do Dispositivo de Medição de Temperatura do Solo

Consequentemente, a solução mais promissora é o desenvolvimento de um sistema capaz de realizar leituras de temperatura em um ou múltiplos pontos e em profundidades variadas do solo. Os dados recolhidos podem influenciar as decisões críticas que contribuem para a manutenção do ambiente ideal para o cultivo.

Um esquema representativo desta solução é descrito e pode ser visualizado em um diagrama subsequente. Este esquema é composto por vários blocos que interagem entre si para formar o sistema completo, que inclui:

- **Alimentação:** Provisão de energia estável para todos os componentes do sistema;
- **Microcontrolador (ESP32):** O núcleo lógico, responsável por gerenciar e processar os dados dos sensores. Traduz o sinal de frequência em dados digitais que podem ser analisados pelo microcontrolador;
- **Sensores NTC:** Elementos sensíveis que detectam mudanças de temperatura no solo, convertendo-as em sinais elétricos;
- **Circuito Oscilador com CI 555:** Mede as alterações na resistência dos sensores, convertendo-as em um sinal de frequência interpretável;
- Interface: Software que transforma a frequência do CI 555 em temperatura na ESP32.

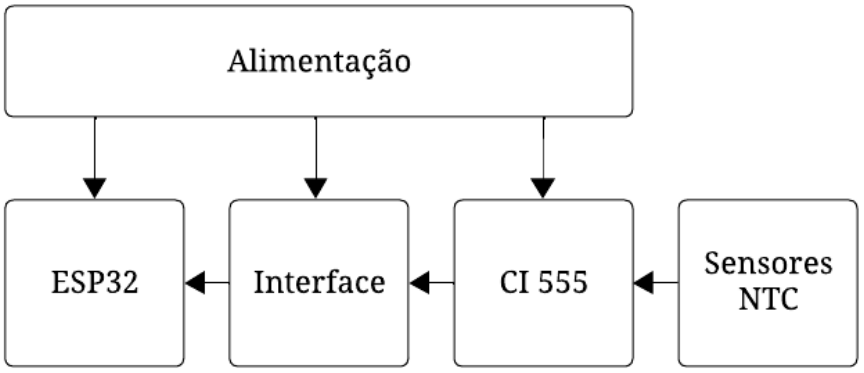


Figura 3.1: Diagrama em blocos dos componentes de hardware do sistema. Autor (2023).

O processo inicia-se com os sensores NTC, que, estrategicamente posicionados no solo, captam variações térmicas e as convertem em resistência elétrica variável. O CI 555, ao ser acoplado a esses sensores, gera um sinal de frequência proporcional, que é posteriormente processado pelo microcontrolador ESP32. Este converte o sinal em dados digitais prontos para análise. Embora a comunicação dos dados para análise remota seja uma extensão natural e valiosa deste projeto, tal funcionalidade será o objeto central do trabalho desenvolvido por LIMA (2023) que está em andamento, que estabelecerá o fundamento para uma futura implementação de monitoramento à distância.

3.2 Caracterização dos Sensores NTC

O sensor NTC varia sua resistência conforme a mudança de temperatura, ver capítulo 2.1.3, foram realizados alguns experimentos para verificar tal comportamento e aferir a constante Beta.

3.2.1 Experimentação para o Coeficiente Beta

Foram realizadas 34 medidas, variando a temperatura de 1,1 até 25 graus celsius. O termômetro padrão utilizado para aferir a temperatura é do tipo culinário, da marca: *INCOTERM,* modelo: 6132, facilmente encontrado em lojas de varejo online no Brasil. Acompanhando a variação da temperatura e da resistência, obteve-se os seguintes dados ilustrados na tabela 3.1:

Tabela 3.1: Medidas de temperatura em laboratório do termômetro padrão e resistência do sensor NTC.

Medida	Temperatura (ºC)	Temperatura (K)	Resistência (kΩ)
1	1,1	274,25	68,39
2	5,2	278,35	56,8
3	5,4	278,55	56,4
4	5,8	278,95	56,2

5	6	279,15	55,95
6	6,2	279,35	55,2
7	6,3	279,45	54,8
8	6,5	279,65	54,65
9	6,7	279,85	54,4
10	7,5	280,65	42
11	8	281,15	41
12	8,6	281,75	40,7
13	9	282,15	4,1
14	9,5	282,65	39,7
15	10,5	283,65	38,8
16	11	284,15	38,2
17	11,5	284,65	37,7
18	14	287,15	33
19	15	288,15	32,4
20	18	291,15	28,5
21	18,8	291,95	27,2
22	19	292,15	26
23	19,7	292,85	25
24	21	294,15	24,35
25	21,5	294,65	24,07
26	22	295,15	23,76
27	22,5	295,65	23,43
28	23	296,15	23,07
29	23,1	296,25	22,97
30	23,5	296,65	22,62
31	24	297,15	22,06
32	24	297,15	21,61
33	24,5	297,65	20,17
34	25	298,15	20,05

Fonte: Autor (2023)

O cálculo do Beta é realizado utilizando duas medidas específicas, conforme detalhado nas Equações 2.42, 2.43 e 2.44. As medidas escolhidas são 274.25K para temperatura à ser calculada e 298.15K para temperatura referência (25ºC).

$$\beta = \frac{ln\left(\frac{R}{R_0}\right)}{\frac{1}{T} - \frac{1}{T_0}} \quad (2.42)$$

$$\beta = \frac{ln\left(\frac{68390}{20050}\right)}{\frac{1}{274.25} - \frac{1}{298.15}} \quad (2.43)$$

$$\beta = 4197.853 \quad (2.44)$$

Com o coeficiente Beta calculado como 4197,853 e sabendo que a resistência do termistor a 25°C é de 20kΩ, foi possível modelar a seguinte equação para obter a resistência em função da temperatura conforme Equação 2.45:

$$R(T) = 2000 \cdot e^{4197.853 \cdot \left(\frac{1}{T} - \frac{1}{298.15}\right)} \quad (2.45)$$

Onde R(T) é a resistência do termistor à temperatura T em Kelvin. A Figura 3.1 ilustra a curva de resistência em função da temperatura para o termistor NTC, demonstrando como a resistência decresce exponencialmente com o aumento da temperatura.

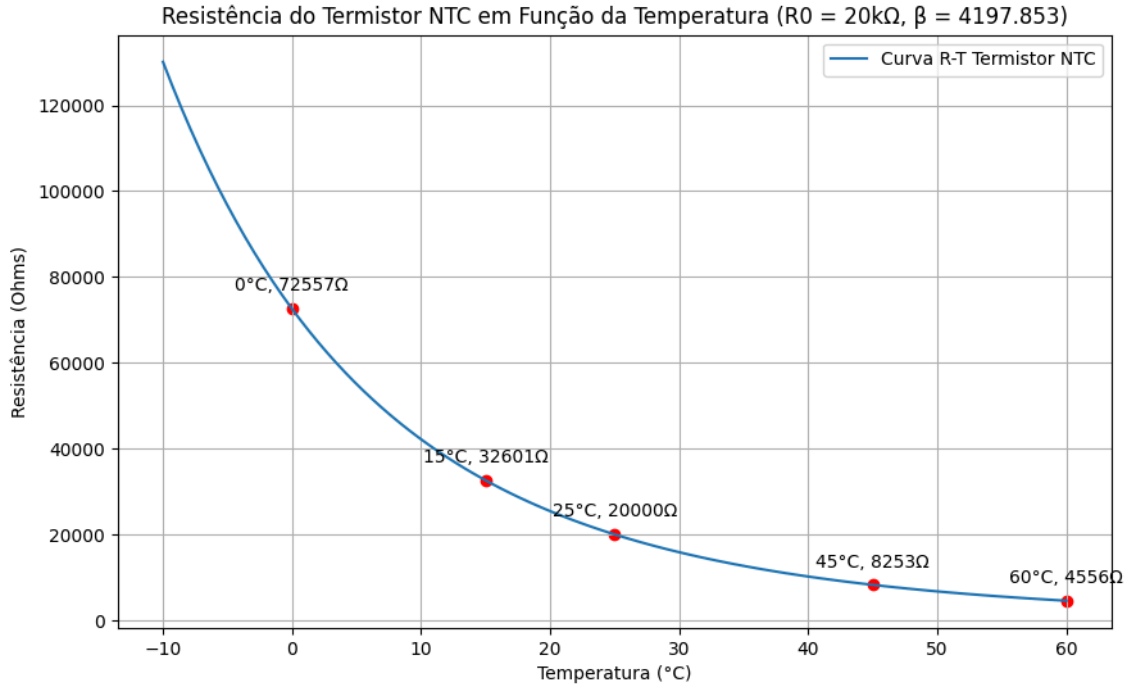


Figura 3.2: Resistência do termistor NTC em função da temperatura. Autor (2023)

Esta modelagem é essencial para a compreensão e aplicação prática dos sensores NTC em medições de temperatura, permitindo a conversão direta das leituras de resistência em valores de temperatura.

3.2.2 Desenvolvimento da Tabela de Resposta do Sensor

A partir dos experimentos realizados e dos cálculos efetuados, foi desenvolvida uma tabela comparativa que relaciona as temperaturas com as resistências calculadas do

sensor NTC. Essa comparação foi feita utilizando tanto o coeficiente Beta fornecido pelo fabricante (4260), ver em THINKING ELECTRONIC INDUSTRIAL CO., LTD. (2020), quanto o Beta calculado experimentalmente (4197,853). A tabela abaixo apresenta os valores de resistência para temperaturas variando de -10°C a 60°C, em intervalos de 10°C:

Tabela 3.2: Resposta ao sensor.

Número	Temperatura (°C)	Resistência (Fabricante)	Resistência (Calculada)
1	-10	133768.1284	130110.5262
2	-5	98914.2071	96634.2144
3	0	73954.4032	72556.9022
4	5	55874.0205	55042.8393
5	10	42633.9719	42165.7810
6	15	32838.0842	32601.3971
7	20	25519.2206	25428.6554
8	25	20000.0000	20000.0000
9	30	15800.9734	15855.3896
10	35	12579.3731	12664.7522
11	40	10087.7978	10189.0234
12	45	8146.0462	8253.4894
13	50	6621.7172	6729.3633
14	55	5416.7192	5520.9305
15	60	4457.7989	4556.4946

Fonte: Autor (2023).

Esses dados ilustram como a resistência do sensor varia com a temperatura e destacam as diferenças nas estimativas quando se utiliza o valor de Beta fornecido pelo fabricante em comparação com o valor de Beta obtido através de cálculos experimentais. Esta tabela é uma ferramenta importante para entender a resposta do

sensor NTC em diferentes condições de temperatura, oferecendo um guia prático para a interpretação dos dados do sensor em aplicações reais.

3.3 Implementação do Circuito Oscilador com CI 555

Esta seção detalha a montagem de um circuito oscilador empregando o temporizador 555 e a escolha de um capacitor de 100nF. O objetivo é verificar a correlação entre as variações de temperatura e as correspondentes alterações na frequência do sinal.

3.3.1 Funcionamento do Circuito Astável Baseado no Temporizador 555

No desenvolvimento do circuito astável baseado no CI 555, considerou-se o comportamento teórico previamente estabelecido para este componente. O circuito projetado aproveita a característica de alternância entre níveis lógicos proporcionada pelo carregamento e descarregamento de um capacitor. Quando o capacitor está carregando, observa-se um sinal de nível lógico alto na saída do CI. Em contraste, o descarregamento do capacitor resulta em um sinal de nível lógico baixo.

A inovação do projeto consiste na utilização de dois sensores de temperatura do tipo NTC em vez de apenas um. Esta configuração permite equilibrar o Fator de Trabalho do sinal de saída para uma ampla gama de temperaturas. O Fator de Trabalho é determinado pela relação entre o tempo de carga (TH), dado por TH = 0,693 * (NTC_1 + Rdiodo) * C1, e o tempo de descarga (TL), definido por TL = 0,693 * NTC_2 * C1. Assim, é possível manter a estabilidade do sinal em condições variadas de temperatura.

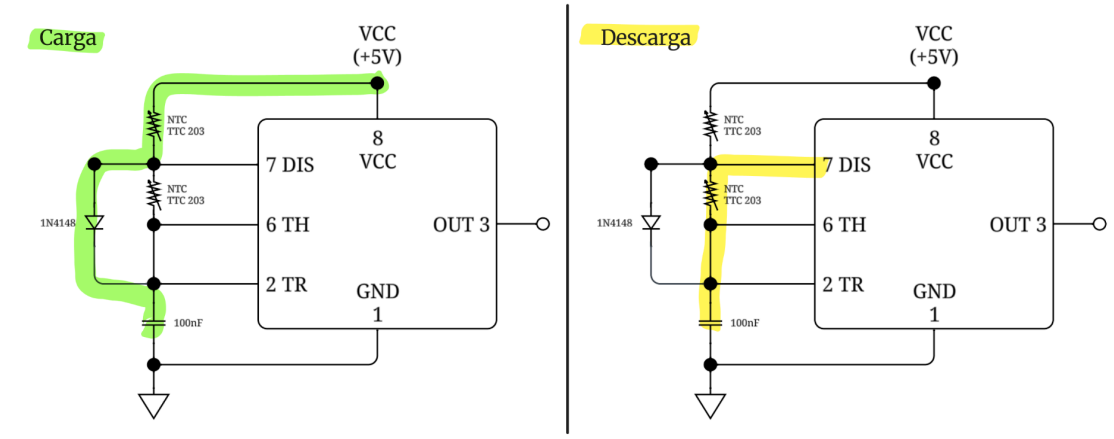


Figura 3.3: Funcionamento de carga e descarga de um circuito astável baseado no temporizador 555. Autor (2023).

Experimentos foram conduzidos para ilustrar o efeito da resistência do diodo e dos sensores NTC sobre o Fator de Trabalho em três cenários de temperatura distintos:

- Baixa Temperatura (70kΩ):
  - Tempo de carga (TH): 0.005044s
  - Tempo de descarga (TL): 0.004851s
  - Fator de Trabalho: 50.97%
  - Frequência: 101.06 Hz
- Temperatura Ambiente (20kΩ):
  - Tempo de carga (TH): 0.001579s
  - Tempo de descarga (TL): 0.001386s

    - Fator de Trabalho: 53.25%
    - Frequência: 337.29 Hz
- Alta Temperatura (6kΩ):
    - Tempo de carga (TH): 0.000609s
    - Tempo de descarga (TL): 0.000416s
    - Fator de Trabalho: 59.41%
    - Frequência: 976.15 Hz

Caso a resistência do diodo não fosse considerada, o Fator de Trabalho resultante seria idêntico em todas as temperaturas, uma vez que tanto o carregamento quanto o descarregamento do capacitor seriam influenciados unicamente pela resistência dos sensores NTC, identificados como Ra e Rb respectivamente. O estudo das variações de Fator de Trabalho demonstra a importância da inclusão da resistência do diodo para a estabilização do ciclo de trabalho do circuito em diferentes condições térmicas.

A frequência do circuito também desempenha papel crucial, com o valor do capacitor determinando diretamente a resolução do sinal de saída. Devido à relação inversa entre a resistência do NTC e a temperatura, a frequência na saída do CI 555 varia proporcionalmente com a temperatura: maior temperatura corresponde a maior frequência. Para assegurar uma resolução de sinal apropriada para amostragem subsequente, foi estabelecida uma resolução média situada entre 10Hz e 20Hz. Assim, a escolha de um capacitor de 100nF permite que a saída do circuito diferencie cada grau Celsius dentro deste intervalo de frequência, garantindo uma resolução média de 15,25Hz para temperaturas variando de 1,2°C a 60,3°C, correspondendo a frequências de 70Hz a 970Hz, respectivamente.

3.3.2 Interface do Circuito com Microcontrolador ESP32

Para a interface do circuito astável baseado no CI 555 com o microcontrolador ESP32, foi necessária a implementação de um mecanismo eficaz de regulação de tensão. A saída de 3.3V do CI 555 é crucial para a compatibilidade com os níveis de tensão lógica do ESP32, segundo datasheet ESP32 WROOM (Espressif Systems, s.d.). Para atingir esta regulação, um diodo Zener de 3.3V foi integrado ao circuito, conectado em paralelo com a entrada do microcontrolador, atuando como um regulador de tensão para manter um nível estável de 3.3V, independentemente das flutuações na corrente SEDRA (et.al 2007).

Uma resistência de 21,25Ω foi estrategicamente posicionada em série entre a saída do CI 555 e o diodo Zener. Essa resistência é essencial para limitar a corrente através do diodo Zener, garantindo que a corrente direcionada ao ESP32 não exceda os 40mA, o limite seguro para o dispositivo. O valor da resistência foi calculado levando-se em conta a tensão de alimentação de 5V do CI 555 e a tensão de zenerização de 3.3V, resultando em uma queda de tensão de 1.7V na resistência sob uma corrente de 80mA.

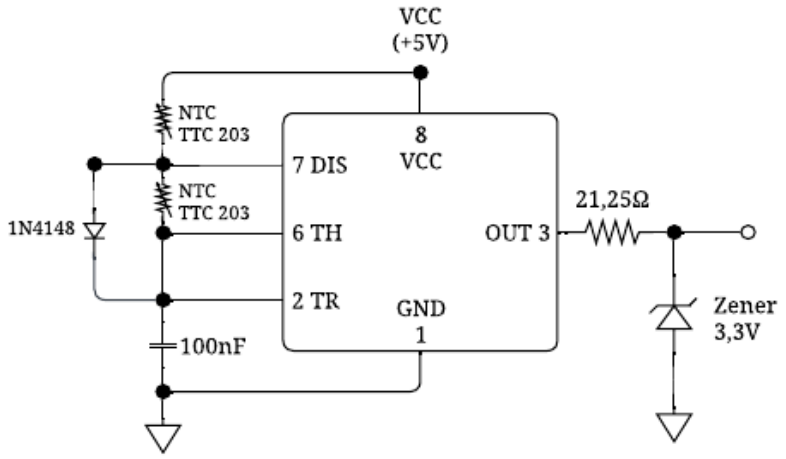


Figura 3.4: Configuração de um circuito astável baseado no temporizador 555 com componentes do sistema para medir a temperatura e responder em frequência. Autor (2023).

A configuração série-paralelo cuidadosamente selecionada oferece uma tensão de saída estável e segura para o microcontrolador, protegendo-o contra potenciais riscos de sobretensão ou sobrecorrente. A escolha meticulosa da resistência e a inclusão do diodo Zener ilustram a aplicação prática dos princípios fundamentais da eletrônica, refletindo a atenção ao detalhe no design de um circuito que deve operar dentro de parâmetros de tensão e corrente bem definidos.

3.3.3 Design do Circuito

Com base nos conceitos e análises discutidos anteriormente, o circuito foi cuidadosamente projetado e sua estrutura pode ser visualizada na Figura 3.5 (a). Este esquema gráfico não apenas sintetiza as escolhas de design e os componentes essenciais, mas também antecipa a organização e o layout para a implementação em uma placa de circuito impresso (PCB), detalhando as conexões e trilhas que serão utilizadas para a montagem final, que pode ser visualizada nas Figuras 3.5 (b) e 3.5 (c).

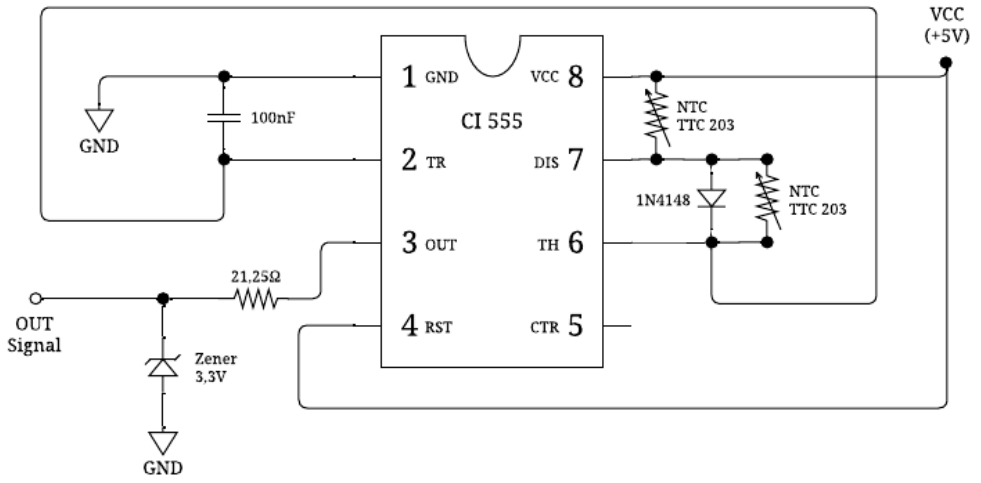


Figura 3.5 (a): Esquemático de um circuito de medição de temperatura utilizando o CI 555 e sensores NTC. Autor (2023).

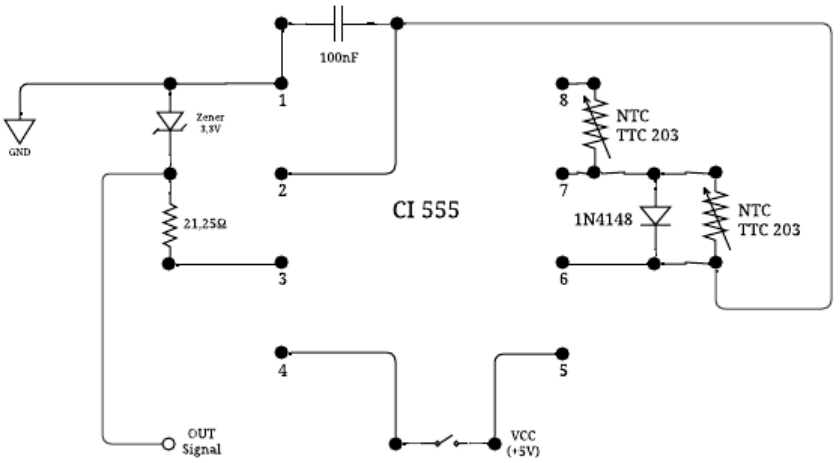


Figura 3.5 (b): Esquemático de um circuito de medição de temperatura utilizando o CI 555 e sensores NTC, preparando para implementação em placa PCB. Autor (2023).

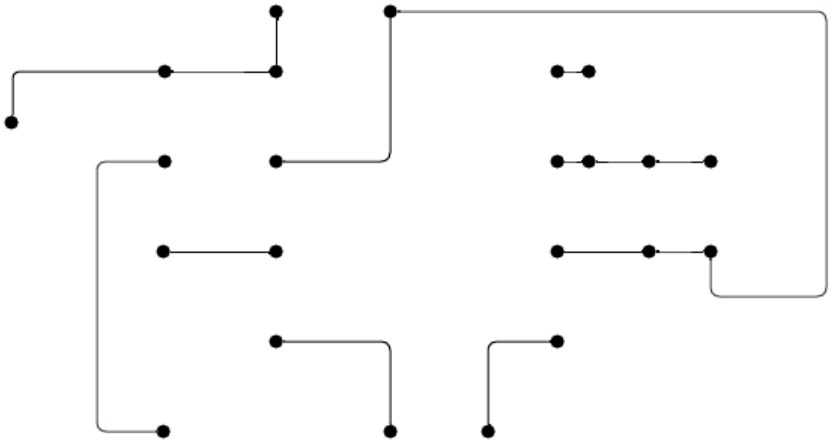


Figura 3.5 (c): Esquemático de um circuito de medição de temperatura utilizando o CI 555 e sensores NTC, preparado para implementação em placa PCB com suas respectivas trilhas. Autor (2023).

3.3.4 Avaliação da Resposta em Frequência do Sinal de Saída

Para avaliar a resposta em frequência do sinal de saída do CI 555 em relação às variações de temperatura, foi conduzido uma série de testes controlados. As temperaturas foram ajustadas gradualmente de 1,2°C a 60,3°C, e um total de 57 amostras distintas foram coletadas. Esse procedimento cuidadoso permitiu uma análise aprofundada da relação entre a temperatura e a frequência gerada pelo CI 555. Os dados obtidos estão detalhados na Tabela 3.3, ilustrando a correspondência entre as temperaturas medidas e as respectivas frequências de saída.

Tabela 3.3: Avaliação da resposta em frequência do sinal de saída.

Medida	Temperatura (ºC)	Frequência (Hz)
1	1.2	70

2	2.3	76
3	11.8	99
…	…	…
55	55.5	855
56	56	875
57	60.3	970

Fonte: Autor (2023).
(Note: Os dados completos estão disponíveis Apêndice B.)

Foi observado que a relação entre temperatura e frequência não é linear, o que levou à divisão do conjunto de dados em seis subconjuntos menores. Cada subconjunto representa uma faixa de temperatura específica, permitindo uma modelagem mais precisa através de regressões lineares.

O Gráfico da Figura 3.6 ilustra todas as regressões lineares dos diferentes conjuntos de dados em um único gráfico. Cada linha de regressão linear representa um conjunto de dados específico, mostrando como a temperatura varia em função da frequência para diferentes intervalos de temperatura. As cores distintas para cada conjunto de dados facilitam a comparação visual entre eles.

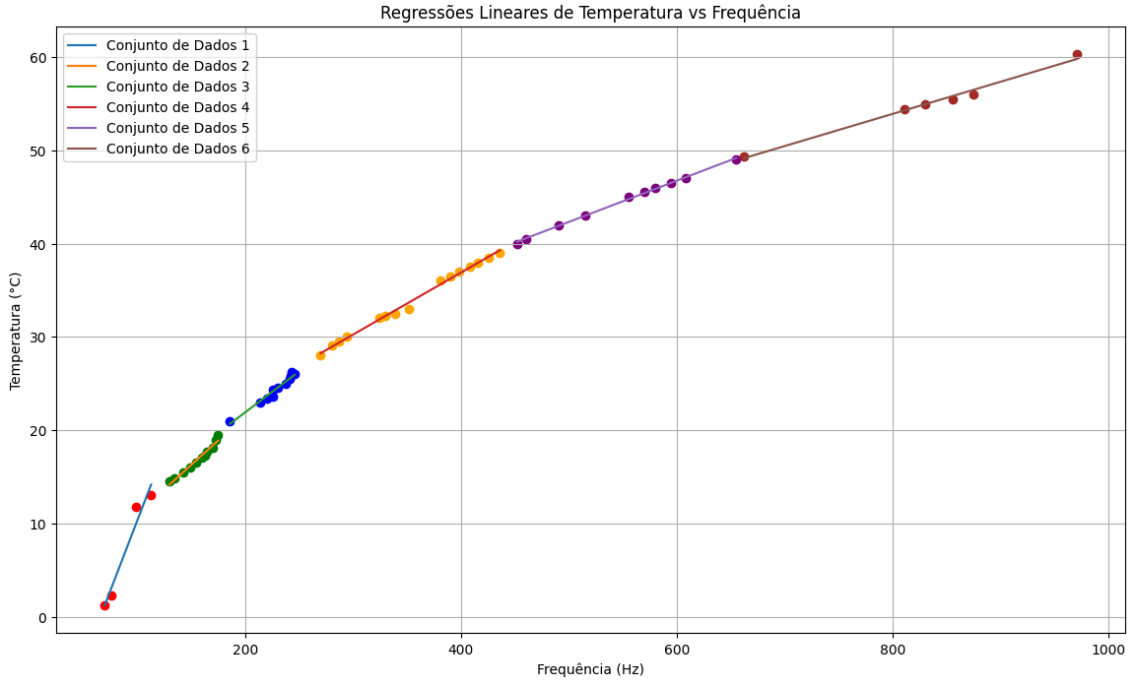


Figura 3.6: Regressões lineares de temperatura vs frequência. Autor (2023).

Para cada um desses conjuntos, foram realizadas regressões lineares, resultando nas seguintes Equações: 2.47, 2.48, 2.49, 2.50, 2.51 e 2.52, , que permitem estimar a temperatura a partir da frequência observada, essas Equações serão utilizadas no *Firmware* para aferir a temperatura com base na frequência:

- Conjunto de Dados 1 (1.2°C a 13°C):

$$Temperatura = 0.30 \cdot Frequência - 19.90 \quad (2.47)$$

- Conjunto de Dados 2 (14.5°C a 19.5°C):

$$Temperatura = 0.10 \cdot Frequência + 0.55 \quad (2.48)$$

- Conjunto de Dados 3 (21°C a 26°C):

$$Temperatura = 0.09 \cdot Frequência + 4.31 \quad (2.49)$$

- Conjunto de Dados 4 (28°C a 39°C):

$$Temperatura = 0.07 \cdot Frequência + 10.28 \quad (2.50)$$

- Conjunto de Dados 5 (40°C a 49°C):

$$Temperatura = 0.04 \cdot Frequência + 20.21 \quad (2.51)$$

- Conjunto de Dados 6 (49.4°C a 60.3°C):

$$Temperatura = 0.03 \cdot Frequência + 26.33 \quad (2.52)$$

3.4 Desenvolvimento do Firmware no Microcontrolador ESP32

O *firmware* desenvolvido tem como meta principal realizar a leitura da frequência do sinal proveniente do CI 555. Para alcançar este objetivo, foram implementadas técnicas de programação avançadas em C, bem como funções, listas, filas e *THReads*, aproveitando as funcionalidades do FreeRTOS e a arquitetura do ESP32 para criar um sistema de amostragem, processamento e cálculo de frequência eficiente.

Um dos principais desafios encontrados no desenvolvimento do *firmware* foi a necessidade de coletar e processar um grande volume de dados em tempo real, mantendo a precisão e a eficiência. Para superar isso, foi empregado o uso de interrupções de hardware, filas de mensagem e *threads*, características essenciais do *FreeRTOS*. Será detalhado cada componente do sistema, desde a configuração dos pinos GPIO para leitura do sinal até a lógica de cálculo da frequência e estimativa da temperatura, enfatizando como essas escolhas contribuem para a eficácia do projeto.

Este capítulo, portanto, proporcionará uma visão abrangente do processo de desenvolvimento do firmware, destacando as escolhas técnicas, os métodos implementados e as soluções adotadas para enfrentar os desafios inerentes a este tipo de projeto de sistema embarcado.

3.4.1 Configuração Inicial

A escolha do GPIO34 para a leitura de sinais do CI555 foi uma decisão estratégica, influenciada por suas características específicas. Este pino, configurado como entrada, é ideal para capturar os sinais de alta precisão. A ausência de resistências de pull-up ou pull-down internas é uma vantagem, pois minimiza interferências externas, garantindo a fidelidade do sinal recebido.

```C/C++
	//Configuração do pino que recebe o sinal do sensor
	esp_rom_gpio_pad_select_gpio(TEMP_PIN);
	gpio_set_direction(TEMP_PIN, GPIO_MODE_INPUT);
```

Fonte: Autor (2023).

Para a amostragem do sinal é utilizado o *High Resolution Timer (ESP Timer)* da ESP32, uma interrupção baseada em hardware que garante alta precisão na medição do tempo. Este timer, configurado via software, permite especificar a função de callback que é chamada em intervalos definidos, essencial para a amostragem precisa do sinal. Detalhes adicionais sobre o funcionamento do ESP Timer podem ser encontrados na documentação do fabricante ESPRESSIF (conforme acessado em 4 dez 2023). A configuração do timer é realizada conforme mostrado a seguir:

```C/C++
	// Configuração do timer
	const esp_timer_create_args_t timer1Conf = {
		.callback = timerTrigger1,
		.name = "Timer1",
	};
	esp_timer_create(&timer1Conf, &timer1Handle);
```

Fonte: Autor (2023).

timerTrigger1 é a função de callback, ativada pelo timer para realizar a amostragem do sinal em intervalos precisos.

A coleta do sinal em do projeto é efetuada pela função *timerTrigger1*. Para compreender seus componentes, é importante primeiro entender a configuração da amostragem. É estabelecido a constante AMOSTRAS para cobrir uma ampla gama de frequências, considerando uma máxima de até 1.000 Hz. Com 5.000 amostras e uma taxa de amostragem de 50 microssegundos, é possível capturar eficientemente o espectro de frequências desejado. Para um sinal de 1.000 Hz, temos 20 amostras por ciclo, o que permite analisar aproximadamente 250 ciclos.

Por exemplo, para um sinal com um Fator de Trabalho de 60%, obtém-se os seguintes resultados:

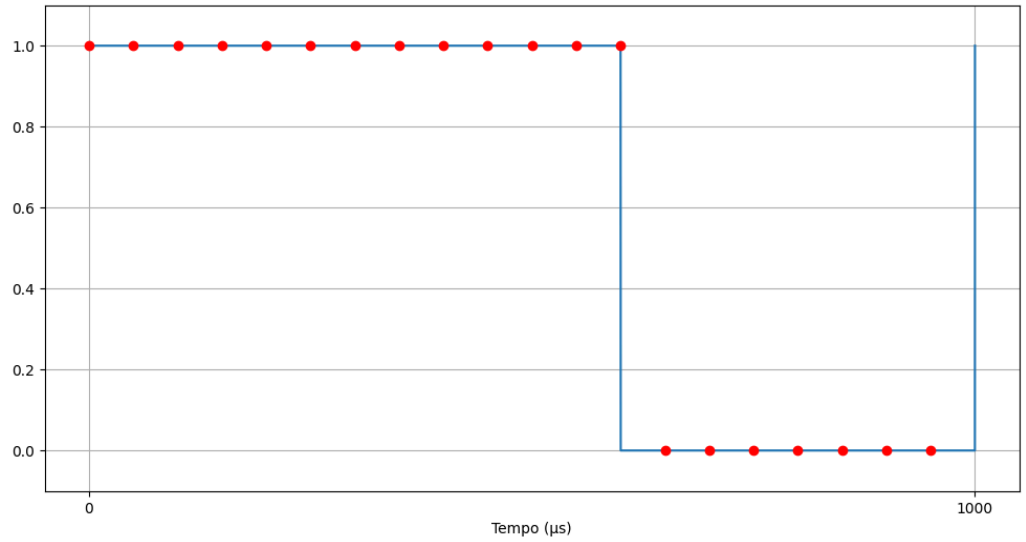


Figura 3.7: Exemplo de sinal de frequência de 1.000Hz com 5.000 amostras com uma taxa de amostragem de 50µs. Autor (2023).

- Quantidade de pontos em 1: 13
- Quantidade de pontos em 0: 7
- Tempo total: 1.000µs
- Frequência calculada: 1.000Hz

Em contraste, para um sinal de 100Hz, cada ciclo dura 10 milissegundos, e a mesma taxa de amostragem resulta em 200 amostras por ciclo, permitindo a análise de 25 ciclos completos. Com um Fator de Trabalho de 45%, tem-se:

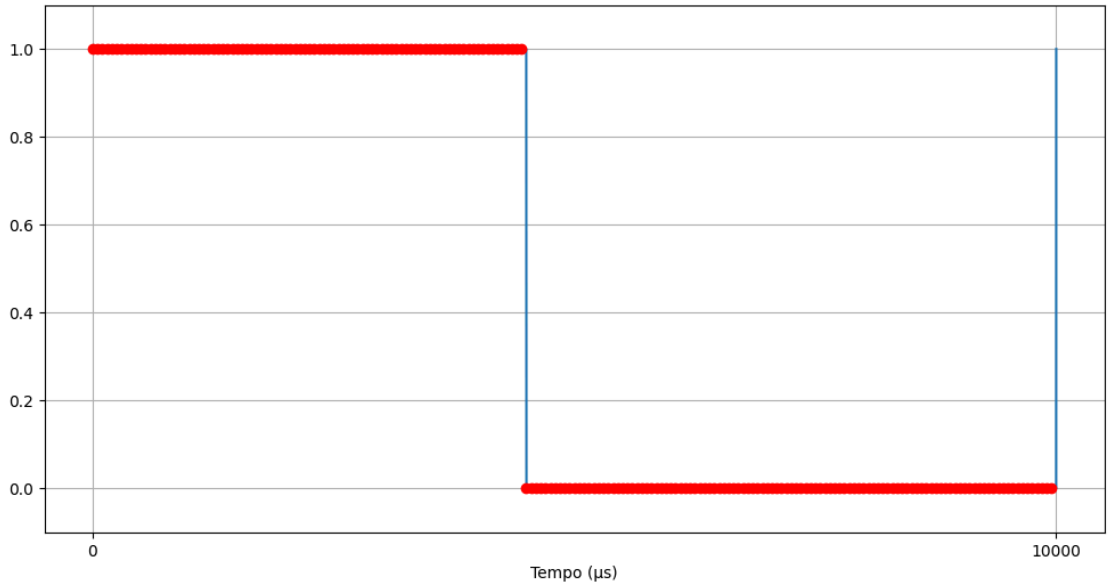


Figura 3.8: Exemplo de sinal de frequência de 100 Hz com 200 amostras com uma taxa de amostragem de 10ms. Autor (2023).

- Quantidade de pontos em 1: 90
- Quantidade de pontos em 0: 110

- Tempo total: 10µs
- Frequência calculada: 100 Hz

Esse comparativo destaca como a taxa de amostragem e o total de amostras permitem adaptar a análise para frequências diferentes, mantendo a precisão e a confiabilidade na avaliação da frequência média.

Para demonstrar o comportamento real do sinal, foram realizadas algumas medidas com o sistema em funcionamento, essas medidas não serviram para a calibração do sistema, apenas para demonstração do sinal de onda quadrada com diferentes frequências, as Figuras 3.9 (a), (b), (c) e (d) ilustram esses comportamentos.

Figura 3.9 (a): Exemplo do sinal em osciloscópio a 15ºC com frequência de 126.26Hz e Fator de Trabalho de 59.09%. Autor (2023).

Figura 3.9 (b): Exemplo do sinal em osciloscópio a 20ºC com frequência de 196.08Hz e Fator de Trabalho de 57.3%. Autor (2023).

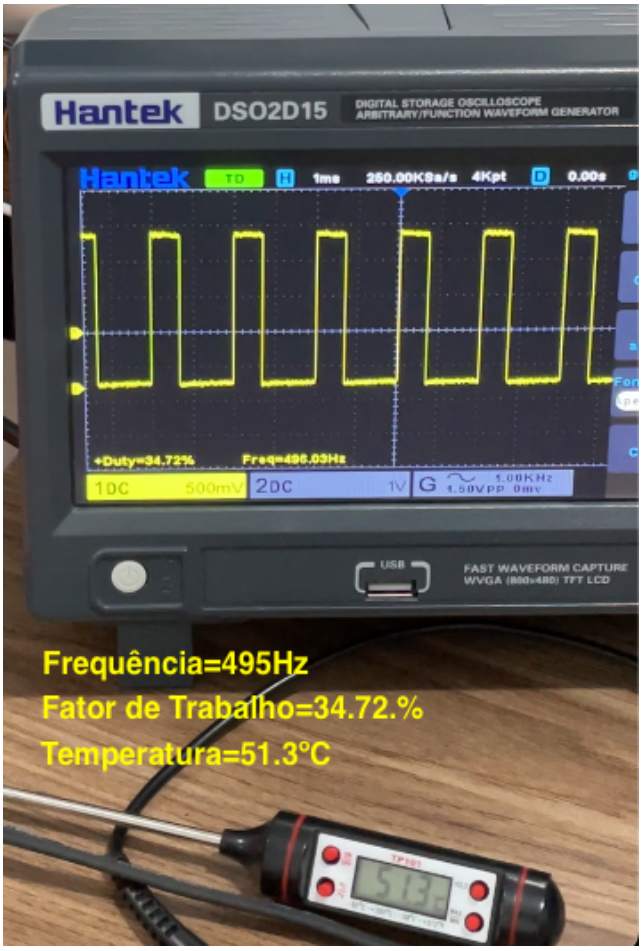


Figura 3.9 (c): Exemplo do sinal em osciloscópio a 51.3ºC com frequência de 495Hz e Fator de Trabalho de 34.72%. Autor (2023).

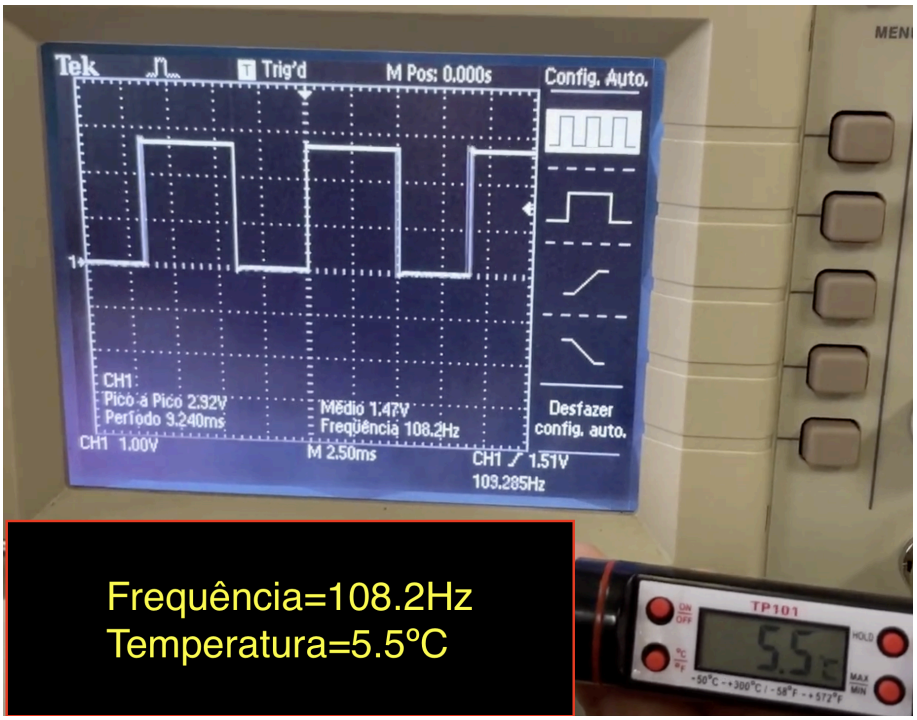


Figura 3.9 (d): Exemplo do sinal em osciloscópio a 5.5ºC com frequência de 108.2Hz sem medida de Fator de Trabalho. Autor (2023).

Prosseguindo para a função "*timerTrigger1"*, ela inicia a leitura do estado do pino "TEMP_PIN" através de *"gpio_get_level"*, que captura o estado atual do sinal. A função *"xQueueSendFromISR"* é usada para enviar este sinal à *"signalQueue"*. Essa função é ideal para o contexto de uma interrupção, pois evita bloqueios e permite o envio eficiente do sinal para processamento posterior. Após a coleta de todas as 5.000 amostras, a função conclui o ciclo de amostragem, parando o *timer* e reinicializando as variáveis para o próximo ciclo. Ver em diagrama de bloco ilustrado na figura 3.10 e trecho de código parcial.

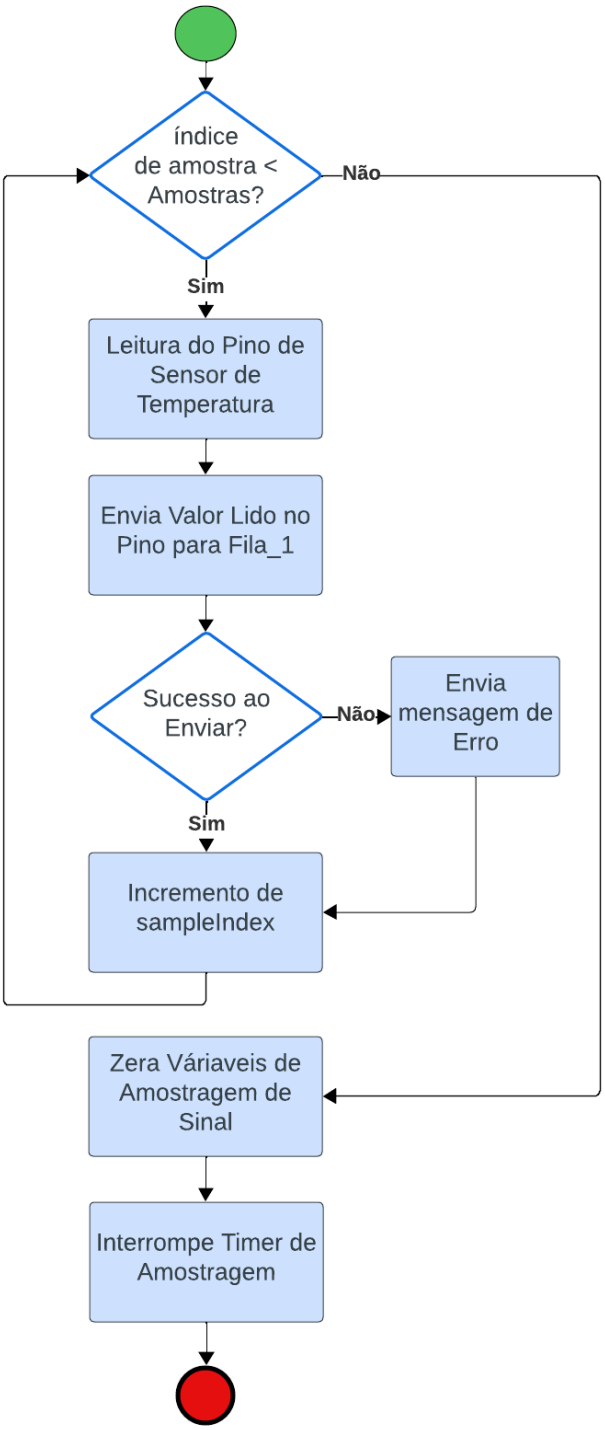


Figura 3.10: Diagrama de blocos de funcionamento da função de amostragem do sinal "*timerTrigger1*". Autor (2023).

```
C/C++
//Função de coleta do sinal
void timerTrigger1(void *arg) {
  if (sampleIndex<AMOSTRAS) {
    signal=gpio_get_level(TEMP_PIN);
    BaseType_t xHigherPriorityTaskWoken = pdFALSE;
            if (xQueueSendFromISR(signalQueue, (void *)&signal,
&xHigherPriorityTaskWoken) != pdPASS) {
          ESP_LOGW("timerTrigger1", "Falha ao adicionar à fila!
Overflow?");
    }
    sampleIndex++;
  } else {
    // ESP_LOGI("timerTrigger1", "Terminei de amostrar o sinal! -
c0=%i - c1=%i",c0,c1);
    sampleIndex=0;
    dc=((float)c1/(c1+c0))*100;
    c0=0;
    c1=0;
    start = true;
    gpio_set_level(LED_PIN, 0);
    esp_timer_stop(timer1Handle);
  }
}
```

Fonte: Autor (2023).

3.4.2 Processamento do Sinal

A *thread "signalProcessingTask"*, criada no início, é dedicada ao processamento paralelo do sinal coletado.

```
C/C++
// Cria a tarefa de processamento de sinal
xTaskCreate(signalProcessingTask, "signalProcessingTask",
2048, NULL, 5, NULL);
```

Fonte: Autor (2023).

Ela monitora continuamente a fila *"signalQueue"*, onde os dados do "*timerTrigger1"* são acumulados. No início da função *"signalProcessingTask"*, a

variável *"currentSignal"* é inicializada com um valor inválido (-1). Isso é feito para lidar com a primeira iteração do loop de processamento. Durante a primeira iteração, *"currentSignal"* é configurado para ser igual ao *"receivedSignal"* (o sinal recebido da fila *"signalQueue"*).

Essa abordagem é usada para garantir que o processamento do sinal comece corretamente, independentemente do estado inicial do sinal. Isso é importante porque a coleta do sinal pode começar em qualquer ponto do ciclo do sinal, e não necessariamente no início. Ao esperar pela primeira mudança no sinal (de 0 para 1 ou de 1 para 0), a função assegura que está processando um ciclo completo do sinal, eliminando possíveis distorções causadas por iniciar a coleta no meio de um ciclo.

Ao detectar essa mudança, a função empacota o valor atual do sinal e a contagem acumulada em uma estrutura *"SignalData_t"*, que tem o seguinte formato:

```
C/C++
typedef struct {
  int value;  // 0 or 1
  int count;  // Count of 0s or 1s
} SignalData_t;
```

Fonte: Autor (2023).

Os dados processados são enviados para *"sequenceQueue"*, a segunda fila do processo, que será utilizada somente após a finalização da amostragem do sinal. A escolha de um buffer com tamanho de 500 para esta fila foi baseada em uma avaliação cuidadosa da frequência do sinal: este tamanho é suficiente para a maioria dos casos, mas dependendo da frequência do sinal, pode ser considerado um limite para o funcionamento adequado do sistema.

O processo de empacotamento dos dados (contagem de zeros e uns em uma estrutura de dados personalizada) na *"sequenceQueue"* é eficiente e ocupa menos espaço de armazenamento, como demonstrado no trecho de código da função *"signalProcessingTask"*. Neste código, a função aguarda a recepção de sinais da *"signalQueue"*, detecta mudanças no sinal recebido e, ao identificar uma mudança, empacota o valor atual do sinal junto com a contagem acumulada, enviando para a *"sequenceQueue"*. O diagrama ilustrado na figura 3.11 e o trecho parcial do código está detalhado no texto.

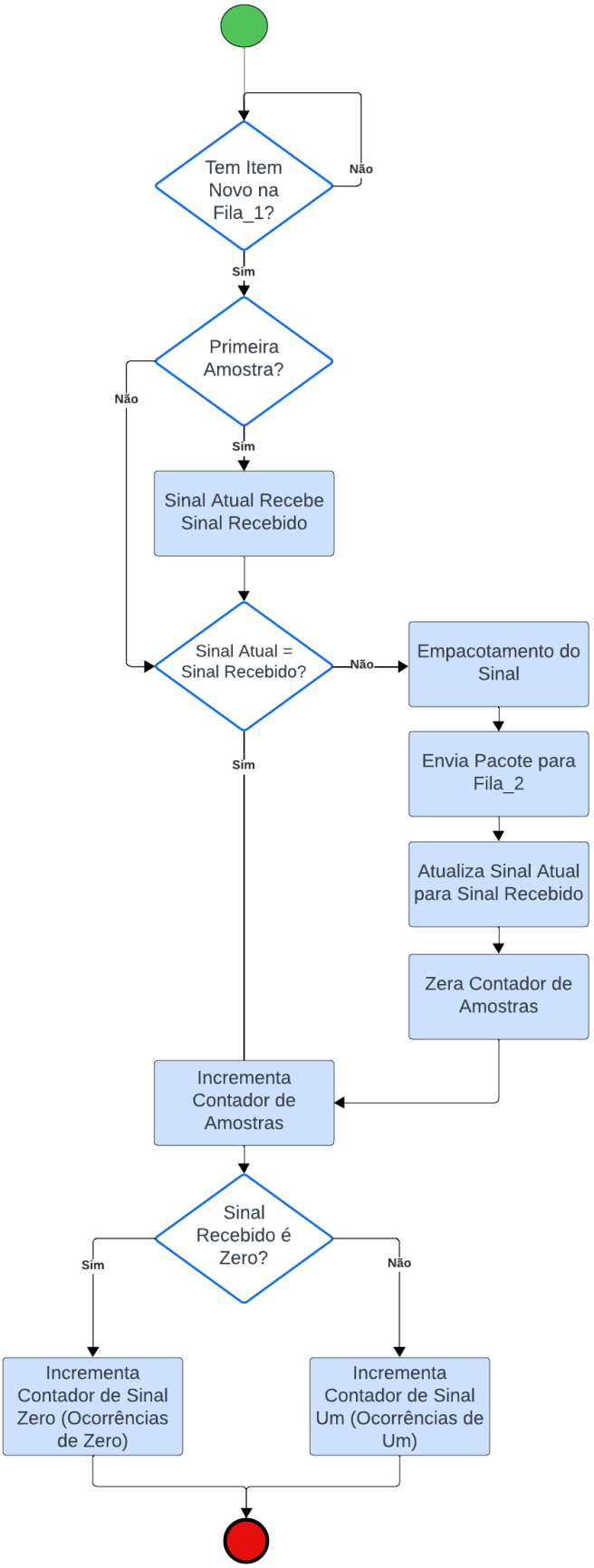


Figura 3.11: Função de empacotamento do sinal *"signalProcessingTask"*. Autor (2023).

```C/C++
//Processamento do sinal
void signalProcessingTask(void *pvParameters) {
  int receivedSignal;
  int currentSignal = -1; //Inicia com valor inválido
  int currentCount = 0;

  while (1) {
    if (xQueueReceive(signalQueue, &receivedSignal,
portMAX_DELAY)) {
      if (currentSignal == -1) { //Primeira interação
        currentSignal = receivedSignal;
      }
      //Verifica se o sinal mudou
      if (currentSignal != receivedSignal) {
        SignalData_t data = {currentSignal, currentCount};
        xQueueSend(sequenceQueue, &data, portMAX_DELAY);
//Envia estrutura de dados para a segunda fila
        // Reset
        currentSignal = receivedSignal;
        currentCount = 0;
      }
      //Incrementa contadores
      currentCount++;
      if(receivedSignal == 0) {
        c0++;
      } else {
        c1++;
      }
    }
  }}
```

Fonte: Autor (2023).

3.4.3 Conversão do Sinal em Frequência para Temperatura

A função "*CalcFrequency",* responsável pela conversão, é executada após a fase de amostragem do sinal e empacotamento do sinal, garantindo que o hardware e as interrupções programadas funcionem adequadamente. Os dados de empacotamento do sinal estão armazenados na fila "*sequenceQueue"*. Esta fila contém uma estrutura de dados que registra a contagem de 'uns' e 'zeros' observados no sinal. O primeiro e o último elemento da fila serão eliminados para garantir que não haverá dados faltantes, a Figura 3.12 ilustra a estrutura de dados esperado da fila:

Figura 3.12: Abstração do empacotamento do sinal. Autor (2023).

Um detalhe importante é que a função não processa os dados se a fila contiver menos de três itens, indicando potencialmente uma falha na amostragem do sinal. A interpretação dos dados da fila é realizada convertendo as contagens em período do sinal. Sabendo a quantidade de 'uns' e 'zeros' contados, é possível determinar o período do sinal. Este cálculo é feito da seguinte maneira:

```
C/C++
  while(uxQueueMessagesWaiting(sequenceQueue) > 1) {
    xQueueReceive(sequenceQueue, &nextData, portMAX_DELAY);
    uint64_t periodUs = (currentData.count + nextData.count) *
    AMOSTRAGEM_US;
    totalPeriodUs += periodUs;
    numPeriods++;
    currentData = nextData;
  }
```

Fonte: Autor (2023).

Após calcular o período do sinal, a frequência é obtida da seguinte forma:

```C/C++
uint64_t avgPeriodUs = totalPeriodUs / numPeriods;
float frequencyHz = 1000000.0 / avgPeriodUs;
```

Fonte: Autor (2023).

Com a frequência calculada, seleciona-se o conjunto de dados apropriado para aplicar a equação linear correspondente:

```C/C++
   if(frequencyHz < 115){//Dados 1 - RMSE=1.15
     temp = frequencyHz*0.30-19.9;
     dados=1;
    }else if(frequencyHz >= 115 && frequencyHz <= 180){//Dados 2 - RMSE=0.30
     temp = frequencyHz*0.10+0.55;
     dados=2;
    }else if(frequencyHz >180 && frequencyHz <=260){//Dados 3 - RMSE=0.29
     temp = frequencyHz*0.09+4.231;
     dados=3;
    }else if(frequencyHz >260 && frequencyHz <=440){//Dados 4 - RMSE=0.28
     temp = frequencyHz*0.07+10.28;
     dados=4;
    }else if(frequencyHz >440 && frequencyHz <=655){//Dados 5 - RMSE=0.14
     temp = frequencyHz*0.04+20.21;
     dados=5;
    }else{//Dados 6 - RMSE=0.34
     temp = frequencyHz*0.03+26.33;
     dados=6;
    }
```

Fonte: Autor (2023).

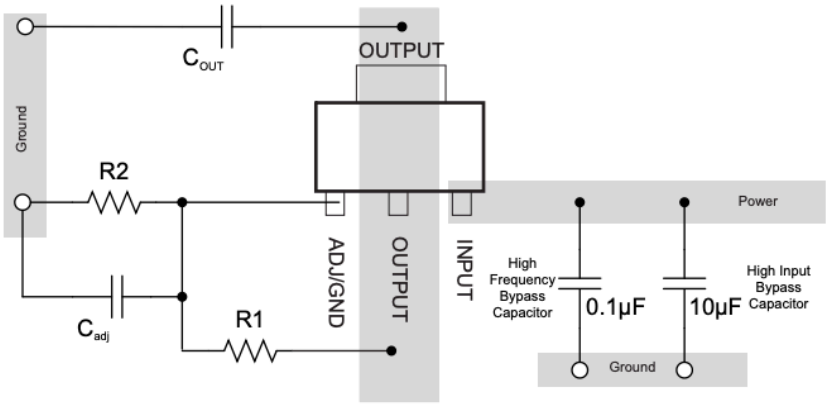
3.5 Prototipagem e Ensaios

Implementação do circuito com todos os seus componentes. Preparação do protótipo para teste em campo.

3.5.1 Elaboração do Módulo de Alimentação com LM317

Visando a autonomia do protótipo, fora da bancada de eletrônica do laboratório, onde usualmente se dispõe de uma fonte de alimentação, foi imperativo projetar um circuito dedicado para alimentar tanto o microcontrolador ESP32 quanto o circuito do sensor (descrito no capítulo 3.3 Circuito Oscilador com CI 555 O LM317), conforme

especificado no datasheet do fabricante TEXAS INSTRUMENTS (2020), é um regulador de tensão positivo ajustável de três terminais. Este regulador tem a capacidade de fornecer até 1,5A, com uma gama de tensões de saída ajustáveis de 1,2V a 37V. Notável pela sua facilidade de uso, o LM317 requer somente a adição de dois resistores externos para a configuração da tensão de saída desejada. Incorpora, ainda, funcionalidades como limitação de corrente interna, desligamento por excesso de temperatura e mecanismos de proteção contra sobrecarga, conferindo-lhe alta segurança operacional. Em alinhamento com estas características, optou-se por adotar a configuração recomendada pelo fabricante no datasheet, proporcionando, assim, uma alimentação estável e segura tanto para o ESP32 quanto para o circuito sensorial.



$$V_O = V_{REF} (1 + R2 / R1) + (I_{ADJ} \times R2)$$

Figura 3.13: Exemplo de *layout* para implementação padrão. TEXAS INSTRUMENTS (2020).

O sistema será alimentado por duas baterias Samsung ICR18650-22P em série. Cada bateria tem uma voltagem nominal de 3,7V e atinge 4,2V quando totalmente carregada, conforme o datasheet do fornecedor. Com esta configuração, o sistema recebe uma tensão média de 8V, garantindo um uso prolongado e a facilidade de recarregar as baterias. A implementação do LM317, conforme ilustrado na Figura 3.14, assegura uma tensão de saída de 5V, importante para o funcionamento adequado tanto do circuito sensor quanto do microcontrolador ESP32. Os valores e a configuração escolhidos foram baseados em testes conduzidos no laboratório de eletrônica da UERGS, seguindo as diretrizes e melhores práticas indicadas no datasheet do fabricante, garantindo assim o desempenho otimizado e a confiabilidade do sistema.

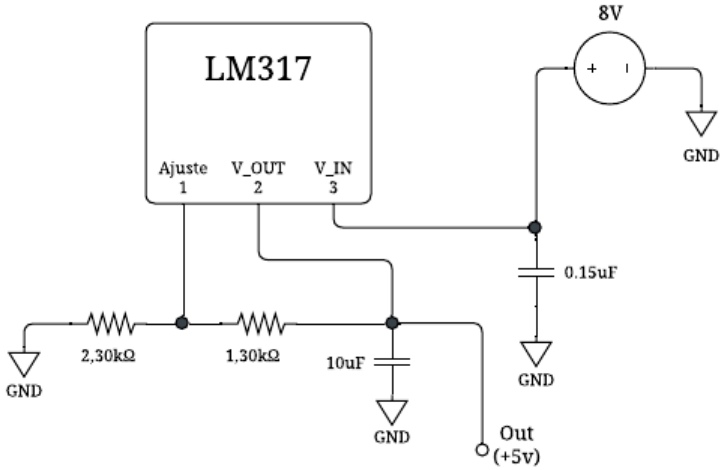


Figura 3.14: Esquemático do circuito de alimentação usando LM317. Autor (2023).

3.5.2 Criação e Produção da Placa de Circuito Impresso

Depois de testar e validar todos os componentes do sistema na protoboard, o próximo passo foi reunir esses elementos em uma placa de circuito impresso. A PCB foi projetada para incluir o circuito do sensor e o circuito de alimentação. Enquanto a bateria e o microcontrolador ESP32 são acoplados externamente, a placa possui espaços específicos para facilitar essas conexões.

A figura 3.15 mostra o layout da PCB, detalhando onde cada componente será colocado. Este desenho foi essencial para garantir que tudo coubesse de maneira organizada e funcional.

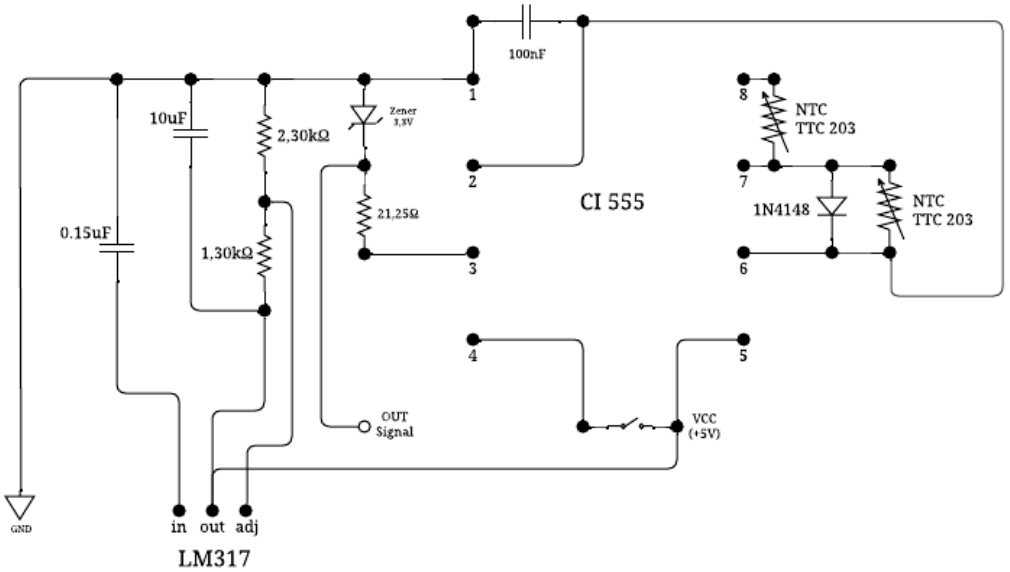


Figura 3.15: Esquemático do circuito de medição de temperatura com todos os seus componentes. Autor (2023).

A Figura 3.16 apresenta a PCB finalizada. As trilhas foram criadas manualmente, e o processo de corrosão do cobre foi realizado utilizando percloreto de ferro, visto em

Corrosão em Circuito Impresso (acesso em 12 dez 2023). As dimensões da placa foram calculadas para se ajustarem a um gabinete padrão (Patola).

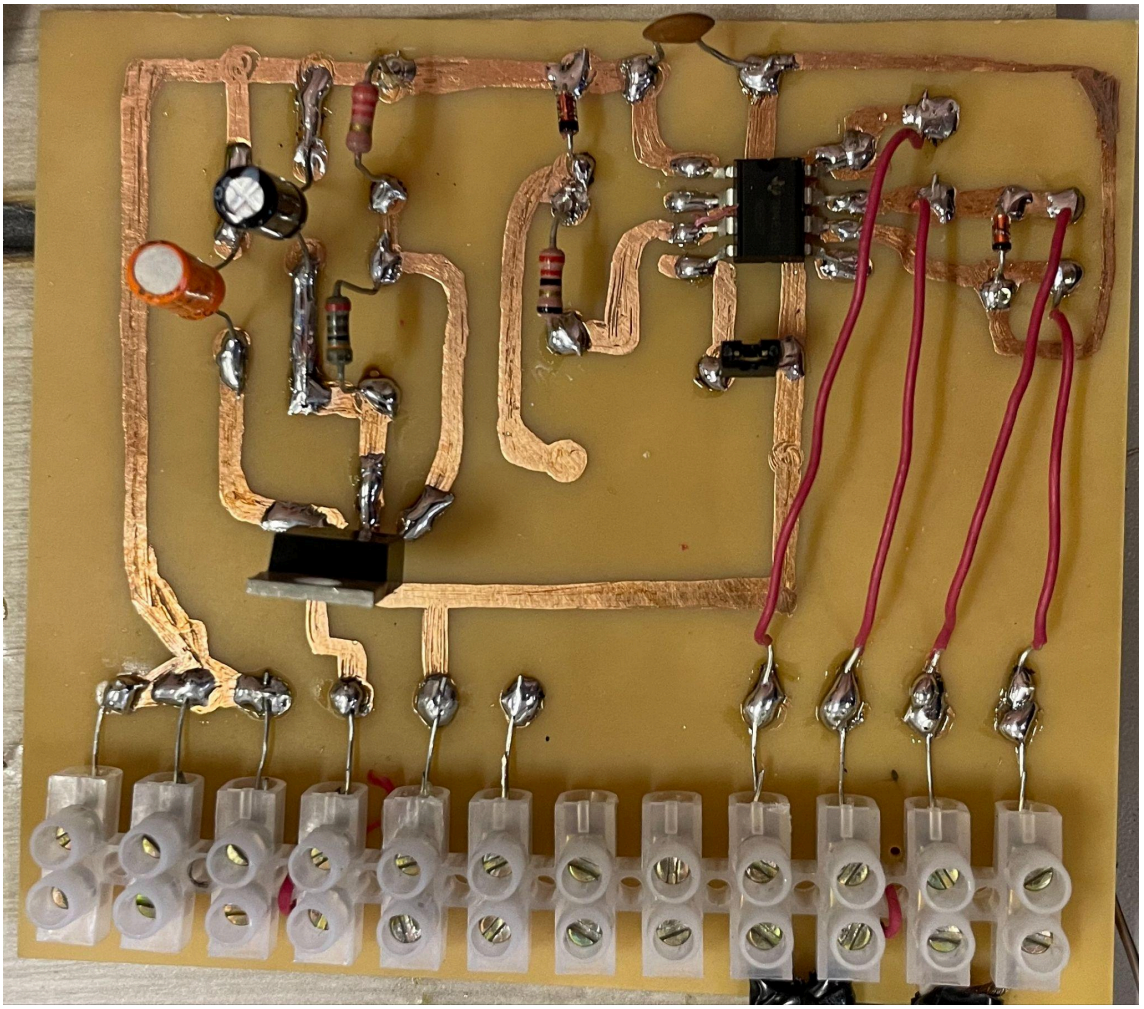


Figura 3.16: PCB implementada. Autor (2023).

Por fim, a Figura 3.17 exibe a placa completamente montada com as baterias e o ESP32. Além disso, foi incluído o módulo LoRa, que faz parte do projeto de TCC do Matheus, ver em LIMA (2023), mencionado anteriormente neste trabalho.

Figura 3.17: Patola com circuito de medição de temperatura, ESP32, LoRa e baterias. Autor (2023).

As Figuras 3.18 (a) e 3.18 (b) trazem imagens do protótipo final, agora completamente montado e pronto para ser testado em campo.

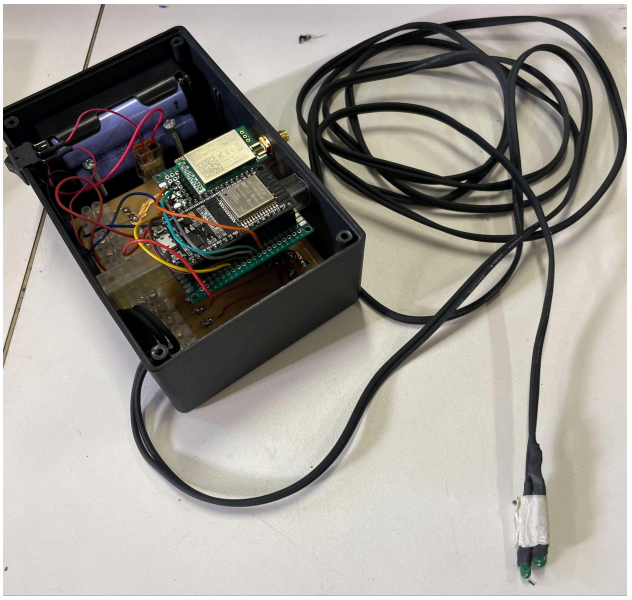


Figura 3.18 (a): Protótipo implementado com tampa aberta. Autor (2023).

Figura 3.18 (b): Protótipo implementado com tampa fechada. Autor (2023).

3.5.3 Testes Funcionais e Avaliação de Desempenho

Nos testes funcionais realizados com o sistema de medição de temperatura, foram analisadas 57 amostras, com temperaturas variando de 1,2°C a 60,3°C. O foco era verificar a precisão do sistema ao calcular a temperatura a partir dos dados coletados.

Na tabela 3.4, é apresentado as temperaturas reais (Temp), utilizando o mesmo equipamento utilizado para descobrir o coeficiente β no capítulo 3.2.1 - termômetro de marca: *INCOTERM* e modelo: 6132, as temperaturas calculadas pelo sistema (Temp Sistema), e o erro entre esses valores (Erro) utilizando a Equação (2.53).

$$Erro = \sqrt{Temp^2 - Temp\ Sistema^2} \quad (2.53)$$

Tabela 3.4: Testes funcionais utilizando protótipo implementado.

Amostra	Temp (°C)	Temp Sistema (°C)	Erro (°C)
1	1,20	1,10	0,1
2	2,30	2,90	0,6
3	11,80	9,80	2
4	13,00	14,00	1
5	14,50	13,55	0,95
6	14,80	14,05	0,75
7	15,50	14,85	0,65
8	16,00	15,45	0,55
9	16,50	16,05	0,45
10	17,00	16,55	0,45
11	17,30	16,85	0,45
12	17,70	17,05	0,65
13	18,10	17,55	0,55
14	19,00	17,85	1,15
15	19,50	18,05	1,45
16	21,00	21,05	0,05
17	23,00	23,57	0,57
18	23,40	24,11	0,71
19	23,60	24,65	1,05
20	24,30	24,65	0,35
21	24,50	25,01	0,51

22	25,00	25,73	0,73
23	25,50	26,00	0,5
24	25,80	26,09	0,29
25	26,20	26,18	0,02
26	26,00	26,45	0,45
27	28,00	29,18	1,18
28	29,10	29,95	0,85
29	29,50	30,37	0,87
30	30,00	30,86	0,86
31	32,00	32,96	0,96
32	32,30	33,38	1,08
33	32,50	34,01	1,51
34	33,00	34,92	1,92
35	36,10	36,95	0,85
36	36,50	37,58	1,08
37	37,00	38,14	1,14
38	37,50	38,84	1,34
39	38,00	39,40	1,4
40	36,50	40,10	3,6
41	39,00	40,80	1,8
42	40,00	39,89	0,11
43	40,50	40,13	0,37
44	42,00	41,03	0,97
45	43,00	41,78	1,22
46	45,00	42,98	2,02
47	45,50	43,43	2,07
48	46,00	43,73	2,27
49	46,50	44,15	2,35
50	47,00	44,57	2,43
51	49,00	45,98	3,02
52	49,40	46,19	3,21
53	54,40	50,66	3,74

54	54,90	51,23	3,67
55	55,50	51,98	3,52
56	56,00	52,58	3,42
57	60,30	55,43	4,87

Fonte: Autor (2023).

Os resultados indicam um desempenho razoavelmente bom, mas com algumas imprecisões. O erro médio de aproximadamente 1.3453 graus indica uma ligeira tendência à subestimação, e o desvio padrão do erro de cerca de 1.712 graus sugere que as temperaturas estimadas podem variar em torno de ±1.712 graus em relação às temperaturas reais. Essas informações são importantes para entender as limitações do sistema e podem ser utilizadas para futuras melhorias, como o ajuste de parâmetros do modelo ou a inclusão de mais dados para treinamento, com o objetivo de aumentar a precisão e reduzir o erro.

O gráfico da Figura 3.19 ilustra visualmente a comparação entre as temperaturas reais e as calculadas pelo sistema para cada uma das amostras. É possível observar que, em geral, as temperaturas calculadas seguem a tendência das temperaturas reais, embora com algumas variações.

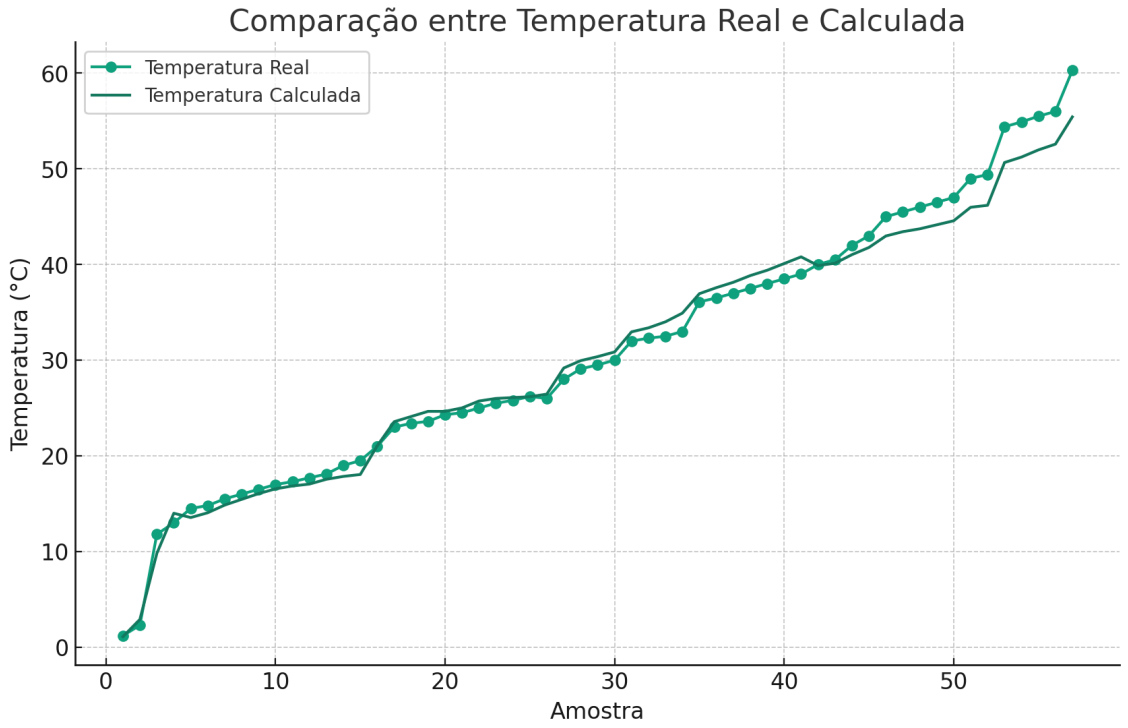


Figura 3.19: Gráfico comparativo da temperatura real e a calculada pelo sistema. Autor (2023).

3.6 Versão 2 de Protótipo para Teste em Campo

A segunda versão do protótipo desenvolvido para testes em campo é projetada com o objetivo específico de implementar três circuitos independentes de coleta de temperatura. Esta configuração permite a aquisição de dados térmicos em três distintos

pontos de profundidade no solo. O primeiro sensor é posicionado na superfície do solo, o segundo é instalado a uma profundidade de 20 centímetros e o terceiro circuito, localizado a 40 centímetros de profundidade.

3.6.1 Criação e Produção do Protótipo Versão 2 com 3 Sensores

Para a montagem desta versão do protótipo, optou-se por reaproveitar o design esquemático ilustrado na Figura 3.4 (a), que demonstra um circuito de medição de temperatura baseado no CI 555, acompanhado de um par de sensores NTC. Para atender à necessidade de realizar três medições de temperatura distintas, o protótipo foi adaptado para incluir seis sensores NTC, operando em três pares. Cada par de sensores está dedicado a uma das profundidades especificadas: superfície, 20 centímetros e 40 centímetros. A Figura 3.20 apresenta a organização dos três circuitos na protoboard, dispostos em camadas sobrepostas, ilustrando a integração eficiente dos componentes para a coleta de dados térmicos.

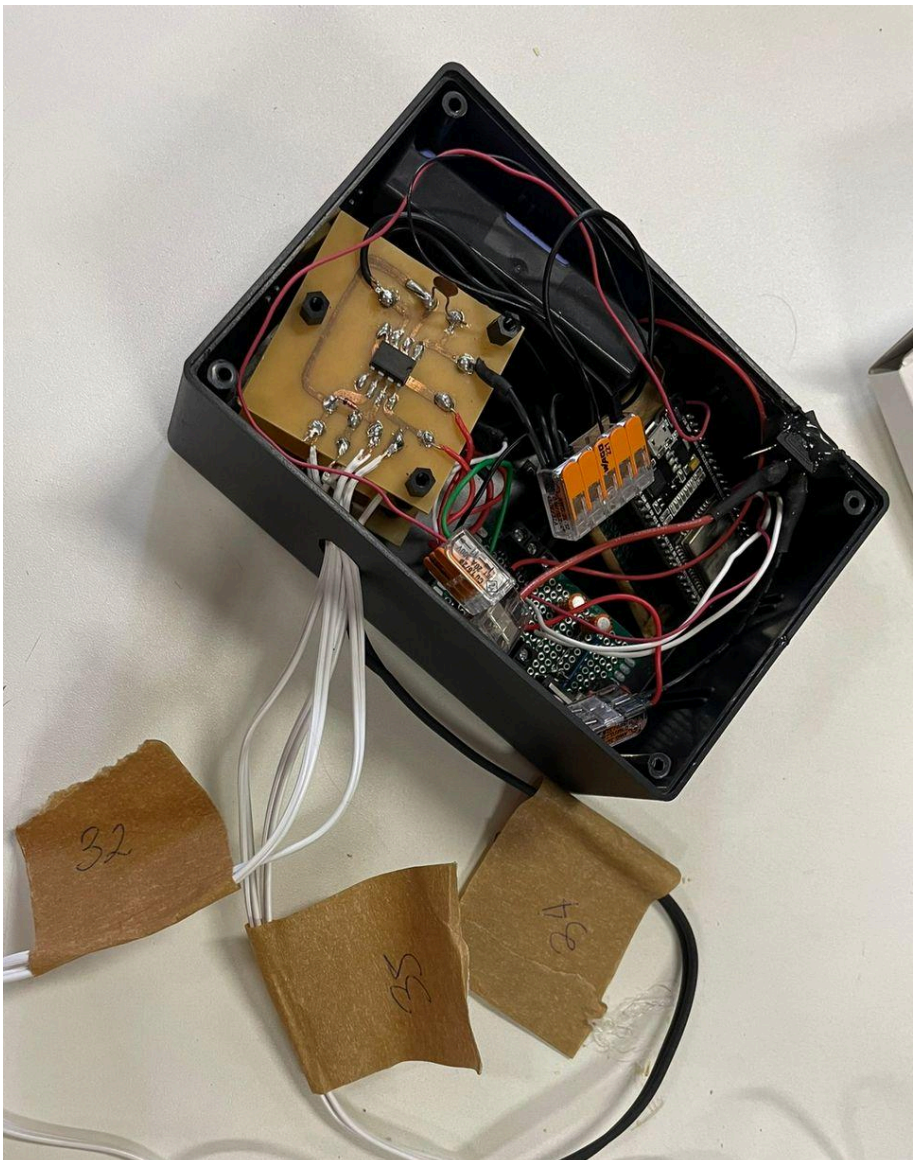


Figura 3.20: Protótipo com circuito adaptado para coletar 3 medidas de temperatura. Autor (2023).

3.6.2 Adaptações do Firmware para Versão 2

As modificações no *firmware* da ESP32, destinadas à segunda versão do protótipo, foram focadas na integração e no gerenciamento dos dados provenientes dos novos sensores. Importante ressaltar que o *kernel* do *firmware* original permaneceu inalterado. As alterações principais incluíram a adição de duas novas portas para a recepção de dados dos sensores adicionais, uma revisão no laço principal do programa e a implementação de um sistema de arquivos local, permitindo a gravação dos dados diretamente na memória interna da ESP32.

Um trecho crítico do código adaptado é a função *"getPinTemp()"*, que gerencia a alternância entre os pinos dos sensores. Esta função opera com um contador "*ctrTempPin"* para iterar entre os três pinos de temperatura (TEMP_PIN_1, TEMP_PIN_2, TEMP_PIN_3), garantindo a coleta sequencial de dados de cada sensor.

No *loop* principal do programa, implementado no trecho *"while(1)"*, a ESP32 está programada para coletar a temperatura de hora em hora. Dentro deste *loop*, existe um *subloop* que executa nove vezes para coletar dados de cada um dos três sensores três vezes. A cada iteração, o sistema verifica se a variável "*start"* está definida como verdadeira para iniciar o *timer,* realiza a leitura do pino de temperatura apropriado através da função "*getPinTemp():* , e ativa um *LED* durante a coleta de dados. Após a coleta, há um intervalo de 10 segundos antes da próxima iteração. Concluído o ciclo de coleta, o sistema registra uma mensagem de *log* e entra em um período de espera de uma hora antes de iniciar o próximo ciclo.

Essa abordagem permite uma coleta de dados eficiente e sistemática, otimizando o uso dos recursos da ESP32 e garantindo a precisão e a confiabilidade das medições de temperatura. O código completo pode ser acessado no github conforme o link acesso em 12 dez 2023 (https://github.com/ulissesmaffa/tcc.git).

3.6.3 Implementação de Protótipo Versão 2 Funcional

A implementação do protótipo ocorreu em 5 de dezembro de 2023, às 17h00 horário de Brasília, UTC-3, na unidade da Universidade Estadual do Rio Grande do Sul (UERGS), localizada em Guaíba, Rio Grande do Sul. A instalação foi realizada em um local específico, cujas coordenadas geográficas são 30,11961° S e 51,36896° W. A Figura 3.21 apresenta uma imagem ilustrativa dos sensores já posicionados no solo, evidenciando a disposição prática e a profundidade em que foram enterrados para a coleta de dados.

Figura 3.21: Implementação do protótipo no solo de Guaíba. Autor (2023).

3.6.4 Testes Funcionais e Avaliação de Desempenho do Protótipo Versão 2

Os dados obtidos a partir dos testes realizados entre os dias 5 e 6 de dezembro de 2023 fornecem *insights* significativos sobre as variações de temperatura do solo em diferentes profundidades. Os testes, conduzidos na unidade da UERGS em Guaíba, Rio Grande do Sul, monitoraram as temperaturas na superfície, a 20 cm e a 40 cm de profundidade, coletando dados de hora em hora, conforme mostra a tabela 3.5.

Tabela 3.5: Temperatura do solo em diferentes profundidades em Guaíba.

Data e Hora (Brasília, UTC-3)	Simplificação do Formato Data e Hora	Temperatura Superfície (ºC)	Temperatura 20cm abaixo do solo (ºC)	Temperatura 40cm abaixo do solo (ºC)
05/12/2023 17:00:00	D5-17h	31,08	28,68	28,73
05/12/2023 18:00:00	D5-18h	31,21	27,05	27,17
05/12/2023 19:00:00	D5-19h	31,06	26,52	26,42
05/12/2023 20:00:00	D5-20h	30,93	26,35	26,06
05/12/2023 21:00:00	D5-21h	30,83	26,13	25,69
05/12/2023 22:00:00	D5-22h	30,72	25,95	25,65
05/12/2023 23:00:00	D5-23h	30,78	25,78	25,56

06/12/2023 00:00:00	D6-00h	30,83	25,92	25,59
06/12/2023 01:00:00	D6-01h	30,79	25,77	25,65
06/12/2023 02:00:00	D6-02h	30,80	25,71	25,63
06/12/2023 03:00:00	D6-03h	30,78	25,69	25,60
06/12/2023 04:00:00	D6-04h	30,85	25,59	25,46
06/12/2023 05:00:00	D6-05h	30,88	25,68	25,56
06/12/2023 06:00:00	D6-06h	31,05	26,15	28,68
06/12/2023 07:00:00	D6-07h	32,08	29,10	31,25
06/12/2023 08:00:00	D6-08h	32,94	30,26	32,73
06/12/2023 09:00:00	D6-09h	33,44	31,24	34,66
06/12/2023 10:00:00	D6-10h	33,42	30,91	33,90
06/12/2023 11:00:00	D6-11h	32,99	30,31	32,81
06/12/2023 12:00:00	D6-12h	32,52	29,90	32,14
06/12/2023 13:00:00	D6-13h	32,31	29,61	31,53
06/12/2023 14:00:00	D6-14h	32,16	29,41	31,12
06/12/2023 15:00:00	D6-15h	31,82	28,92	30,20
06/12/2023 16:00:00	D6-16h	31,73	28,92	29,92
06/12/2023 17:00:00	D6-17h	31,86	28,79	28,84
06/12/2023 18:00:00	D6-18h	31,13	26,19	26,74

Fonte: Autor (2023).

Observou-se que a temperatura na superfície do solo apresentou uma tendência de aumento gradual durante o período da manhã do dia 6, alcançando um pico de 33,44°C às 09:00. Posteriormente, houve uma leve redução nas horas subsequentes. Interessantemente, as temperaturas a 20 cm e 40 cm de profundidade seguiram um padrão similar, embora com valores menores, indicando uma influência direta da temperatura da superfície nas camadas mais profundas.

No período noturno do dia 5, as temperaturas registradas em todas as profundidades mantiveram-se relativamente estáveis, com uma variação mínima. Isso sugere que as camadas do solo mantêm um equilíbrio térmico mais consistente durante a noite.

Um ponto notável foi a significativa elevação das temperaturas em todas as profundidades registradas na manhã do dia 6, possivelmente devido à intensificação da radiação solar. As temperaturas a 40 cm de profundidade mostraram um aumento mais pronunciado em comparação com as outras profundidades, atingindo 34,66°C às 09:00. Isso pode indicar uma maior retenção de calor nesta profundidade durante as horas de maior incidência solar.

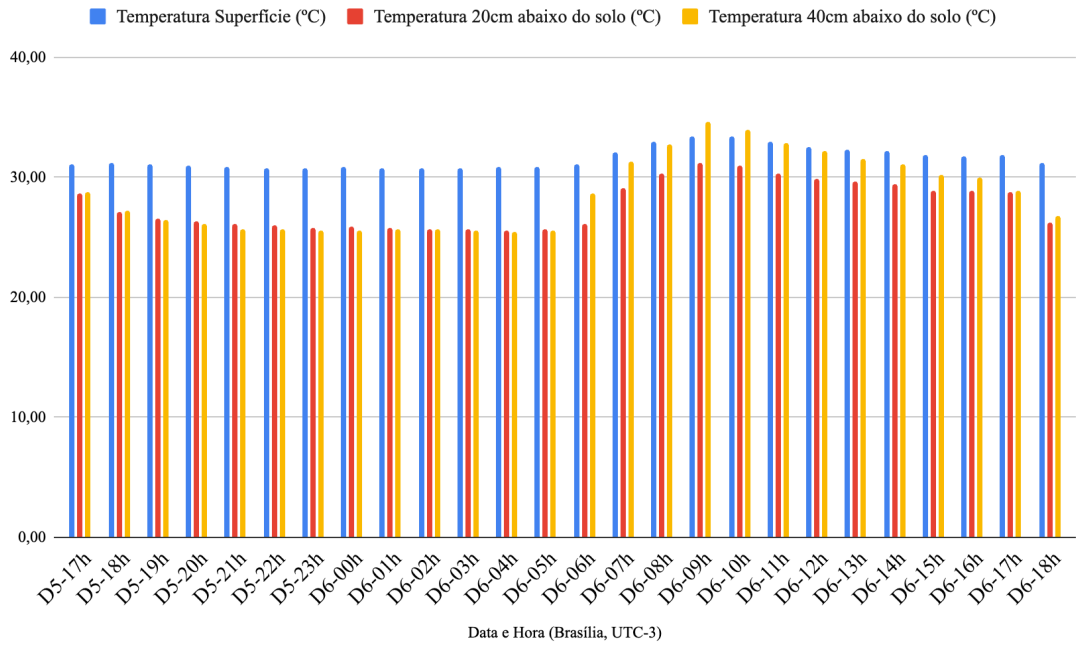


Figura 3.22: Temperatura do solo em diferentes profundidades em Guaíba. Autor (2023).

4 CONCLUSÃO

Este trabalho de conclusão de curso abordou o desenvolvimento de um sistema de medição de temperatura do solo, utilizando sensores NTC, um circuito oscilador baseado no CI 555 e um microcontrolador ESP32, que são facilmente encontrados em lojas especializadas em eletrônica no Brasil. A abordagem técnica envolveu o design de um circuito eletrônico personalizado e a criação de software embarcado em microcontrolador, resultando em um método eficaz para a obtenção de medições exatas da temperatura do solo através da resposta em frequência do CI 555 à variação da resistência dos sensores NTC.

Os dados coletados em laboratório e em campo não são conclusivos para avaliar a capacidade de coletar variações de temperatura do solo, carecem de calibração para diminuir o erro e realizar mais amostras em campo. No laboratório foi possível observar que a saída em frequência do CI 555 foi eficiente na conversão das mudanças de resistência dos sensores NTC em leituras de temperatura compreensíveis, enquanto o software embarcado no microcontrolador processou com eficiência as principais faixas de temperatura. Isso destacou o potencial de utilidade do sistema não apenas para medições de temperatura do solo, mas também para outras aplicações que requerem monitoramento de temperatura.

O funcionamento independente do design e a validação em testes de campo indicam a aplicabilidade do sistema para além de contextos acadêmicos, potencialmente beneficiando práticas agrícolas e outras aplicações, permitindo agregar múltiplos sensores e diferentes tipos de sensores, desde que sejam resistivos (variam a resistência).

Os resultados obtidos apontam para a contribuição significativa do dispositivo nas práticas de manejo agrícola, especialmente na gestão eficiente de recursos hídricos e na escolha de coberturas de solo baseadas em dados históricos de temperatura. Isso pode resultar em práticas agrícolas mais sustentáveis.

Para futuras melhorias, recomenda-se expandir a funcionalidade do dispositivo, incorporando sensores adicionais para monitorar outras grandezas físicas relevantes à produção agropecuária. A integração com sistemas de agricultura inteligente, utilizando tecnologias como a Internet das Coisas *(IoT)*, poderia enriquecer a coleta de dados e facilitar decisões agrícolas baseadas em informações detalhadas. Além disso, é necessário otimizar a durabilidade e a eficiência energética do sistema. O ajuste do CI 555 para limitar a coleta de dados desnecessária, com uso do RST e a implementação de um modo de economia de energia no microcontrolador, por meio de funções como *DeepSleep*, são essenciais para reduzir o consumo de energia e prolongar a vida útil do dispositivo. A miniaturização do dispositivo e a redução de custos são também áreas críticas de desenvolvimento, potencializando a adoção em larga escala do sistema em diversos contextos de aplicação.

REFERÊNCIAS

AZEVEDO, T. R.; GALVANI, E. Ajuste do ciclo médio mensal horário da temperatura do solo em função da temperatura do ar. Revista Brasileira de Agrometeorologia, v. 11, n. 2, p. 123-130, 2003.

BABIUCH, M., FOLTYNEK, P., & SMUTNY, P. (2019). *Using the ESP32 Microcontroller for Data Processing*. 2019 *20th International Carpathian Control Conference (ICCC)*. doi:10.1109/carpathiancc.2019.8765944

BAKER B. C., *Thermistors in Single Supply Temperature Sensing Circuits* (Microchip Technology Inc., 1999)

BERGAMASCHI, H.; GUADAGNIN,M.R. Modelos de ajuste para médias de temperatura do solo, em diferentes profundidades. Rev. Bras. Agrometeorol., Santa Maria, v.1, n 1, p. 95-99, 1993.

BRAGA, Newton C. O Circuito Integrado 555. Disponível em: https://www.newtoncbraga.com.br/como-funciona/592-o-circuito-integrado-555-art011.html. Acesso em: 14 out. 2023.

Corrosão em Circuito Impresso. Tec-Ci. Disponível em: https://tec-ci.com.br/blog/circuito-impresso/corrosao-circuito-impresso/ Acesso em: 12 dez. 2023.

ESPRESSIF SYSTEMS. Esp32 Wroom32 Disponível em: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf. Acesso em: 4 dez. 2023.

ESPRESSIF. *High Resolution Timer (ESP Timer).* Disponível em: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/esp_timer.html Acesso em: 4 dez 2023.

FETEIRA, A., & REICHMANN, K. (2010). NTC *Ceramics: Past, Present and Future. Advances in Science and Technology,* 67, 124–133.

GUADAGNINI, Paulo H.; BARLETTE, Vania E. Um termômetro eletrônico de leitura direta com termistor. Revista Brasileira de Ensino de Física, v. 27, n. 3, p. 369-375, set. 2005. Disponível em: https://doi.org/10.1590/S1806-11172005000300011 . Acesso em: 1 nov. 2023.

KLAR, A. E. A influência do solo e do clima nas necessidades hídricas da cultura da cebola. Botucatu, 1974. 171 f. Tese (Livre-docência) - Faculdade de Ciências Agrárias Médicas e Biológicas de Botucatu, Universidade Estadual Paulista, 1974.

LIMA, Matheus Araujo de. RTOS Embarcado com LoRaWAN para monitoramento de dados de qualidade do solo na agricultura. 2023. Trabalho de Conclusão de Curso Engenharia de Computação - Universidade Estadual do Rio Grande do Sul. Em andamento.

MAXWELL, J.C. (1904) *Theory of Heat,* p. 1-348.

MC-GHEE, J., e HENDERSON, L.A., SYDENHAM, P.H., *(1999), Sensor science - Essentials for Instrumentation and Measurement Technology, Measurement,* 25, 89-113.

MC-GHEE, J., e HENDERSON, L.A. (1993) *Current trends in the theory and application of classification to instrumentation and measurement science, in L. Finkelstein and K. T. V. Grattan (Eds.) State and Advances of Measurement and Instrumentation Science,* Proc IMEKO TCI/TC7 Colloquium, City University, London, 32-37.

MICHALSKI, L., Eckersdorf, K., Kucharski, J. and McGhee, J., *Temperature Measurement*, 2nd edition, xiii, p. 1-501.

MOTA, F. S. Meteorologia agrícola. 6. ed. São Paulo: Nobel, 1983.

RICIERI, Reinaldo Prandini et al. Temperatura no perfil do solo utilizando duas densidades de cobertura e solo nu. Acta Scientiarum. Agronomy, v. 27, n. 1, 11 abr. 2005. Disponível em: https://doi.org/10.4025/actasciagron.v27i1.2127. Acesso em: 4 dez. 2023.

TEXAS INSTRUMENTS. LM317 3-Terminal Adjustable Output Positive Voltage Regulators. SLVS044Y – Setembro 1997 – Revisado em abril de 2020. Disponível em: https://www.ti.com/lit/ds/symlink/lm317.pdf Acesso em: 12 dez 2023.

THE ORGANIC CHEMISTRY TUTOR. Thermistors - NTC & PTC - Thermal Resistors - Temperature Sensors & Resettable Fuses. Publicado em 20 fev 2020. Disponível em: https://www.youtube.com/watch?v=nXmkkyw8v5A Acesso em: 10 dez 2023.

THINKING ELECTRONIC INDUSTRIAL CO., LTD. (2020). N*TC Thermistor: TTC05 Series*. [Datasheet]. Disponível em: https://www.thinking.com.tw/upload/productdesigntools2/files/en-NTC%20Thermistor-TTC05%20Series.pdf. Acesso em: 4 dez 2023

VAREJÃO-SILVA, Mário Adelmo. Meteorologia e Climatologia. Recife: [s. n.], 2006. E-book (449 p.). Disponível em: https://icat.ufal.br/laboratorio/clima/data/uploads/pdf/METEOROLOGIA_E_CLIMATOLOGIA_VD2_Mar_2006.pdf. Acesso em: 1 dez. 2023.

SEM AUTOR. Instrumentação e Técnicas de Medidas. 2ª ed. Rio de Janeiro: [s.n.], 2017. Disponível em: https://docplayer.com.br/68313117-Instrumentacao-e-tecnicas-de-medida-ufrj-2017-2.html. Acesso em: 10 dez 2023.

APÊNDICE A <LTSpice CI 555 Astável>

```
XU1 0 N001 Out Vcc NC_01 N001 N002 Vcc NE555
Ra Vcc N002 10k
Rb N002 N001 15k
D1 N002 N001 1N4148
C1 N001 0 100nF
V1 Vcc 0 5v
.model D D
.lib <Seu Endereço Local>/LTspice/lib/cmp/standard.dio
.tran 10m
.meas TRAN T_high TRIG V(Out)=2.5 RISE=3 TARG V(Out)=2.5 FALL=3 TD=0.1m
.meas TRAN T_low TRIG V(Out)=2.5 FALL=3 TARG V(Out)=2.5 RISE=3 TD=0.1m
.lib NE555.sub
.backanno
.end
```

APÊNDICE B <Tabela 3.3 completa>

Tabela 3.3 completa com todas as informações:

Medida	Temperatura (ºC)	Frequência (Hz)
1	1.2	70
2	2.3	76
3	11.8	99
4	13	113
5	14.5	130
6	14.8	135
7	15.5	143
8	16	149
9	16.5	155
10	17	160
11	17.3	163
12	17.7	165
13	18.1	170
14	19	173
15	19.5	175
16	21	186
17	23	214
18	23.4	220
19	23.6	226
20	24.3	226
21	24.5	230
22	25	238
23	25.5	241
24	25.8	242
25	26.2	243
26	26	246
27	28	270

28	29.1	281
29	29.5	287
30	30	294
31	32	324
32	32.3	330
33	32.5	339
34	33	352
35	36.1	381
36	36.5	390
37	37	398
38	37.5	408
39	38	416
40	38.5	426
41	39	436
42	40	452
43	40.5	460
44	42	490
45	43	515
46	45	555
47	45.5	570
48	46	580
49	46.5	594
50	47	608
51	49	655
52	49.4	662
53	54.4	811
54	54.9	830
55	55.5	855
56	56	875
57	60.3	970

Fonte: Autor (2023).

APÊNDICE C <CÓDIGO FONTE *FIRMWARE* 1 (UM) SENSOR>

C/C++

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/timers.h"
#include "esp_timer.h"
#include "freertos/task.h"
#include "driver/gpio.h"
#include "esp_log.h"
#include "freertos/queue.h"

#define LED_PIN 2
#define TEMP_PIN 34
#define BUFFER_SIZE 200
#define BUFFER_SIZE2 500
#define AMOSTRAGEM_US 50
#define AMOSTRAS 5000

esp_timer_handle_t timer1Handle;
QueueHandle_t signalQueue; //amostragem parte 1: fila para contagem de 1 e 0
QueueHandle_t sequenceQueue; //amostragem parte 2: fila para agrupamento de 1 e 0

//Variáveis
volatile int sampleIndex=0;
int c0 = 0, c1 = 0, signal;
float dc=0.0;
volatile bool start = true;

typedef struct {
  int value;  // 0 or 1
  int count;  // Count of 0s or 1s
} SignalData_t;

void timerTrigger1(void *arg) {
  if (sampleIndex<AMOSTRAS) {
    signal=gpio_get_level(TEMP_PIN);
    BaseType_t xHigherPriorityTaskWoken = pdFALSE;
                if (xQueueSendFromISR(signalQueue, (void *)&signal, &xHigherPriorityTaskWoken) != pdPASS) {
                ESP_LOGW("timerTrigger1", "Falha ao adicionar à fila! Overflow?");
    }
    sampleIndex++;
  } else {
       // ESP_LOGI("timerTrigger1", "Terminei de amostrar o sinal! - c0=%i - c1=%i",c0,c1);
    sampleIndex=0;
    dc=((float)c1/(c1+c0))*100;
```

```
      c0=0;
      c1=0;
      start = true;
      gpio_set_level(LED_PIN, 0);
      esp_timer_stop(timer1Handle);
    }
  }

  //Processamento do sinal
  void signalProcessingTask(void *pvParameters) {
    int receivedSignal;
    int currentSignal = -1; //Inicia com valor inválido
    int currentCount = 0;

    while (1) {
      if (xQueueReceive(signalQueue, &receivedSignal, portMAX_DELAY)) {
        if (currentSignal == -1) { //Primeira interação
          currentSignal = receivedSignal;
        }
        //Verifica se o sinal mudou
        if (currentSignal != receivedSignal) {
          SignalData_t data = {currentSignal, currentCount};
                xQueueSend(sequenceQueue, &data, portMAX_DELAY); //Envia
estrutura de dados para a segunda fila
          // Reset
          currentSignal = receivedSignal;
          currentCount = 0;
        }
        //Incrementa contadores
        currentCount++;
        if(receivedSignal == 0) {
          c0++;
        } else {
          c1++;
        }
      }
    }
  }

  void printQueueContents() {
    SignalData_t data;
    ESP_LOGI("Main", "Início da fila:");
     while (xQueueReceive(sequenceQueue, &data, pdMS_TO_TICKS(20))) { // 0
timeout, para não bloquear
          //  ESP_LOGI("Main",  "[Value:  %d,  Count:  %d]",  data.value,
data.count);
    }
    ESP_LOGI("Main", "Fim da fila");
  }

  void clearQueue(QueueHandle_t queue) {
    void *tempData;
    while (xQueueReceive(queue, &tempData, 0) == pdTRUE) {
      // Faz nada, apenas retira o item da fila
```

```
    }
  }

  void calcFrequency(){
    SignalData_t currentData, nextData;
    uint64_t totalPeriodUs = 0; // total de microssegundos
    int numPeriods = 0;     // número de períodos completos
    float temp=0.0; //Temperatura

    // Se não houver dados na fila, retorne
     if(uxQueueMessagesWaiting(sequenceQueue) < 3) {  // Precisa de pelo
menos 3 itens para ter "itens do meio"
      ESP_LOGE("calcFrequency", "Fila com menos de 3 itens.");
      return;
    }

    // Ignore o primeiro elemento
    xQueueReceive(sequenceQueue, &currentData, portMAX_DELAY);

    while(uxQueueMessagesWaiting(sequenceQueue) > 1) {
      xQueueReceive(sequenceQueue, &nextData, portMAX_DELAY);
           // Calcular período usando a contagem de 'currentData' e
'nextData'
      // Isso assume que sua sequência é alternada entre 0s e 1s.
            uint64_t periodUs = (currentData.count + nextData.count) *
AMOSTRAGEM_US;
      totalPeriodUs += periodUs;
      numPeriods++;
      // Prossiga para o próximo par
      currentData = nextData;
    }
      // Ignore o último item (já que ele foi parcialmente usado no
último período calculado)
    xQueueReceive(sequenceQueue, &nextData, portMAX_DELAY);
    if(numPeriods > 0) {
      uint64_t avgPeriodUs = totalPeriodUs / numPeriods;
       float frequencyHz = 1000000.0 / avgPeriodUs; // convertendo período
em frequência
      int dados=0;
      if(frequencyHz < 115){//Dados 1 - RMSE=1.15
        temp = frequencyHz*0.30-19.9;
        dados=1;

          }else if(frequencyHz >= 115 && frequencyHz <= 180){//Dados 2 -
RMSE=0.30
        temp = frequencyHz*0.10+0.55;
        dados=2;
           }else if(frequencyHz >180 && frequencyHz <=260){//Dados 3 -
RMSE=0.29
        temp = frequencyHz*0.09+4.231;
        dados=3;
           }else if(frequencyHz >260 && frequencyHz <=440){//Dados 4 -
RMSE=0.28
        temp = frequencyHz*0.07+10.28;
        dados=4;
```

```
            }else if(frequencyHz >440 && frequencyHz <=655){//Dados  5  -
RMSE=0.14
        temp = frequencyHz*0.04+20.21;
        dados=5;
       }else{//Dados 6 - RMSE=0.34
        temp = frequencyHz*0.03+26.33;
        dados=6;
       }

         ESP_LOGI("calcFrequency", "Frequência calculada: %.2f Hz - DC:
%.2fp.p - Dados: %i - Temp: %.2f", frequencyHz,dc,dados,temp);
     } else {
              ESP_LOGE("calcFrequency", "Não foi possível calcular a
frequência.");
     }
   }


   void app_main(void) {
     // Configuração do led interno
     esp_rom_gpio_pad_select_gpio(LED_PIN);
     gpio_set_direction(LED_PIN, GPIO_MODE_OUTPUT);

     // Configuração do pino que recebe o sinal do sensor
     esp_rom_gpio_pad_select_gpio(TEMP_PIN);
     gpio_set_direction(TEMP_PIN, GPIO_MODE_INPUT);

     // Configuração do timer
     const esp_timer_create_args_t timer1Conf = {
       .callback = timerTrigger1,
       .name = "Timer1",
     };

     esp_timer_create(&timer1Conf, &timer1Handle);

     // Cria a fila amostragem 1
     signalQueue = xQueueCreate(BUFFER_SIZE, sizeof(int));
     if (signalQueue == NULL) {
       ESP_LOGE("app_main", "Falha ao criar fila.");
       return;
     }

     // Cria a tarefa de processamento de sinal
         xTaskCreate(signalProcessingTask, "signalProcessingTask", 2048,
NULL, 5, NULL);

     // Cria a fila amostragem 2
     sequenceQueue = xQueueCreate(BUFFER_SIZE2, sizeof(SignalData_t));
     if (sequenceQueue == NULL) {
       ESP_LOGE("app_main", "Falha ao criar fila.");
       return;
     }

     while (1) {
       if (start) {//Iniciar o timer somente se start for true
```

```
        // printQueueContents();
        // printQueueSize();
        calcFrequency();
        clearQueue(sequenceQueue);

        start = false;
        gpio_set_level(LED_PIN, 1);
        // ESP_LOGI("Main", "Vou iniciar uma amostra de temperatura!");
        esp_timer_start_periodic(timer1Handle, AMOSTRAGEM_US);
      }
      vTaskDelay(4000 / portTICK_PERIOD_MS);
    }
  }
```

APÊNDICE D <CÓDIGO FONTE *FIRMWARE* 3 (TRÊS) SENSORES>

C/C++

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/timers.h"
#include "esp_timer.h"
#include "freertos/task.h"
#include "driver/gpio.h"
#include "esp_log.h"
#include "freertos/queue.h"
//=====================================
//Para trabalhar com sistema de arquivos
#include "esp_spiffs.h"
#include "esp_err.h"
#include <string.h>
//=====================================

#define LED_PIN 2
// #define BOTAO_PIN 0

#define TEMP_PIN_1 34
#define TEMP_PIN_2 35
#define TEMP_PIN_3 32

#define BUFFER_SIZE 200
#define BUFFER_SIZE2 500
#define AMOSTRAGEM_US 50
#define AMOSTRAS 5000

esp_timer_handle_t timer1Handle;
QueueHandle_t signalQueue; //amostragem parte 1: fila para contagem de
1 e 0
QueueHandle_t  sequenceQueue;  //amostragem  parte  2:  fila  para
agrupamento de 1 e 0

//Variáveis
volatile int sampleIndex=0;
int c0 = 0, c1 = 0, signal, tempPin=TEMP_PIN_1, ctrTempPin=0;
float dc=0.0;
volatile bool start = true;

typedef struct {
  int value;  // 0 or 1
  int count;  // Count of 0s or 1s
} SignalData_t;

//=====================================================================
==============================
esp_err_t init_spiffs() {
  ESP_LOGI("SPIFFS", "Inicializando o sistema de arquivos SPIFFS");

  esp_vfs_spiffs_conf_t conf = {
```

```
        .base_path = "/spiffs",
        .partition_label = NULL,
        .max_files = 5,
        .format_if_mount_failed = true
      };

      // Registra o sistema de arquivos no VFS
      esp_err_t ret = esp_vfs_spiffs_register(&conf);
      if (ret != ESP_OK) {
        if (ret == ESP_FAIL) {
            ESP_LOGE("SPIFFS", "Falha ao montar ou formatar o sistema de
arquivos");
        } else if (ret == ESP_ERR_NOT_FOUND) {
          ESP_LOGE("SPIFFS", "Partição SPIFFS não encontrada");
        } else {
            ESP_LOGE("SPIFFS", "Erro desconhecido ao inicializar SPIFFS:
%s", esp_err_to_name(ret));
        }
      } else {
        size_t total = 0, used = 0;
        if (esp_spiffs_info(NULL, &total, &used) == ESP_OK) {
            ESP_LOGI("SPIFFS", "Sistema de arquivos montado, total: %d,
usado: %d", total, used);
        }
      }
      return ret;
    }

    void write_to_file(const char *filename, const char *data) {
      FILE* f = fopen(filename, "a");
      if (f == NULL) {
          ESP_LOGE("write_to_file", "Falha ao abrir o arquivo %s para
escrita", filename);
        return;
      }

       fprintf(f, "%s\n", data); // Escreve a linha no arquivo e adiciona
uma quebra de linha
      fclose(f);
      ESP_LOGI("write_to_file", "Dados gravados no arquivo: %s", data);
    }

    void read_from_file(const char *filename) {
      ESP_LOGI("read_from_file", "[%lld] Inicio",esp_timer_get_time());
      FILE* f = fopen(filename, "r");
      if (f == NULL) {
          ESP_LOGE("read_from_file", "Falha ao abrir o arquivo %s para
leitura", filename);
        return;
      }

      int i = 0;
       char line[128]; // Ajuste o tamanho conforme necessário para seus
dados
      while (fgets(line, sizeof(line), f) != NULL) {
```

```
      // Exibe cada linha lida
      ESP_LOGI("read_from_file", "Linha [%i] lida: %s", i, line);
      i++; // Incrementa o contador para a próxima linha
    }

    fclose(f);
    ESP_LOGI("read_from_file", "[%lld] Fim",esp_timer_get_time());
  }

  void get_file_info(const char *filename) {
    FILE *f = fopen(filename, "r");
    if (f == NULL) {
      // O arquivo não existe, então cria um novo arquivo
      f = fopen(filename, "w");
      if (f == NULL) {
        ESP_LOGE("get_file_info", "Não foi possível criar o arquivo %s",
filename);
        return;
      }
      ESP_LOGI("get_file_info", "Arquivo %s criado", filename);
    } else {
      // O arquivo existe, lê as informações do arquivo
      int line_count = 0;
      char buffer[128]; // Buffer para ler cada linha

      while (fgets(buffer, sizeof(buffer), f) != NULL) {
        line_count++;
      }
          ESP_LOGI("get_file_info", "O arquivo %s contém %d linhas",
filename, line_count);
    }
    fclose(f);
  }

  void delete_file(const char *filename) {
    if (remove(filename) == 0) {
         ESP_LOGI("delete_file", "Arquivo '%s' excluído com sucesso.",
filename);
    } else {
          ESP_LOGE("delete_file", "Falha ao excluir o arquivo '%s'.",
filename);
    }
  }


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

  void timerTrigger1(void *arg) {
    if (sampleIndex<AMOSTRAS) {
      // signal=gpio_get_level(TEMP_PIN_1);
      signal=gpio_get_level(tempPin);
      BaseType_t xHigherPriorityTaskWoken = pdFALSE;
                  if (xQueueSendFromISR(signalQueue, (void *)&signal,
&xHigherPriorityTaskWoken) != pdPASS) {
```

```
                ESP_LOGW("timerTrigger1", "Falha ao adicionar à fila!
Overflow?");
       }
       sampleIndex++;
     } else {
         // ESP_LOGI("timerTrigger1", "Terminei de amostrar o sinal! -
c0=%i - c1=%i",c0,c1);
       sampleIndex=0;
       dc=((float)c1/(c1+c0))*100;
       c0=0;
       c1=0;
       start = true;
       gpio_set_level(LED_PIN, 0);
       esp_timer_stop(timer1Handle);
     }
   }

   //Processamento do sinal
   void signalProcessingTask(void *pvParameters) {
     int receivedSignal;
     int currentSignal = -1; //Inicia com valor inválido
     int currentCount = 0;

     while (1) {
       if (xQueueReceive(signalQueue, &receivedSignal, portMAX_DELAY)) {
         if (currentSignal == -1) { //Primeira interação
           currentSignal = receivedSignal;
         }
         //Verifica se o sinal mudou
         if (currentSignal != receivedSignal) {
           SignalData_t data = {currentSignal, currentCount};
                 xQueueSend(sequenceQueue, &data, portMAX_DELAY); //Envia
estrutura de dados para a segunda fila
           // Reset
           currentSignal = receivedSignal;
           currentCount = 0;
         }
         //Incrementa contadores
         currentCount++;
         if(receivedSignal == 0) {
           c0++;
         } else {
           c1++;
         }
       }
     }
   }

   void printQueueContents() {
     SignalData_t data;
     ESP_LOGI("Main", "Início da fila:");
      while (xQueueReceive(sequenceQueue, &data, pdMS_TO_TICKS(20))) { // 0
timeout, para não bloquear
```

```
            // ESP_LOGI("Main", "[Value: %d, Count: %d]", data.value,
data.count);
      }
      ESP_LOGI("Main", "Fim da fila");
    }

    void clearQueue(QueueHandle_t queue) {
      void *tempData;
      while (xQueueReceive(queue, &tempData, 0) == pdTRUE) {
        // Faz nada, apenas retira o item da fila
      }
    }

    void calcFrequency(){
      SignalData_t currentData, nextData;
      uint64_t totalPeriodUs = 0; // total de microssegundos
      int numPeriods = 0;     // número de períodos completos
      float temp=0.0; //Temperatura
      char log_buffer[256]; //gravar em arquivo

      // Se não houver dados na fila, retorne
       if(uxQueueMessagesWaiting(sequenceQueue) < 3) { // Precisa de pelo
menos 3 itens para ter "itens do meio"
        // ESP_LOGE("calcFrequency", "Fila com menos de 3 itens.");
                  sprintf(log_buffer, "[%lld] Fila com menos de 3
itens.",esp_timer_get_time());
        ESP_LOGE("calcFrequency", "%s",log_buffer);
        write_to_file("/spiffs/temp.txt", log_buffer);
        return;
      }

      // Ignore o primeiro elemento
      xQueueReceive(sequenceQueue, &currentData, portMAX_DELAY);

      while(uxQueueMessagesWaiting(sequenceQueue) > 1) {
        xQueueReceive(sequenceQueue, &nextData, portMAX_DELAY);
             // Calcular período usando a contagem de 'currentData' e
'nextData'
        // Isso assume que sua sequência é alternada entre 0s e 1s.
              uint64_t periodUs = (currentData.count + nextData.count) *
AMOSTRAGEM_US;
        totalPeriodUs += periodUs;
        numPeriods++;
        // Prossiga para o próximo par
        currentData = nextData;
      }
        // Ignore o último item (já que ele foi parcialmente usado no
último período calculado)
      xQueueReceive(sequenceQueue, &nextData, portMAX_DELAY);
      if(numPeriods > 0) {
        uint64_t avgPeriodUs = totalPeriodUs / numPeriods;
         float frequencyHz = 1000000.0 / avgPeriodUs; // convertendo período
em frequência
        int dados=0;
        if(frequencyHz < 115){//Dados 1 - RMSE=1.15
```

```
        temp = frequencyHz*0.30-19.9;
        dados=1;

          }else if(frequencyHz >= 115 && frequencyHz <= 180){//Dados 2 -
RMSE=0.30
        temp = frequencyHz*0.10+0.55;
        dados=2;
            }else if(frequencyHz >180 && frequencyHz <=260){//Dados 3 -
RMSE=0.29
        temp = frequencyHz*0.09+4.231;
        dados=3;
            }else if(frequencyHz >260 && frequencyHz <=440){//Dados 4 -
RMSE=0.28
        temp = frequencyHz*0.07+10.28;
        dados=4;
            }else if(frequencyHz >440 && frequencyHz <=655){//Dados 5 -
RMSE=0.14
        temp = frequencyHz*0.04+20.21;
        dados=5;
       }else{//Dados 6 - RMSE=0.34
        temp = frequencyHz*0.03+26.33;
        dados=6;
       }

       // ESP_LOGI("calcFrequency", "Frequência calculada[%i]: %.2f Hz -
DC:     %.2fp.p     -     Dados:     %i     -     Temp:     %.2f",
tempPin,frequencyHz,dc,dados,temp);
         sprintf(log_buffer, "[%lld] Frequência calculada[%i]: %.2f Hz -
DC: %.2fp.p - Dados: %i - Temp: %.2f",esp_timer_get_time(),tempPin,
frequencyHz, dc, dados, temp);
       ESP_LOGI("calcFrequency", "%s",log_buffer);
       write_to_file("/spiffs/temp.txt", log_buffer);
     } else {
           // ESP_LOGE("calcFrequency", "Não foi possível calcular a
frequência.");
            sprintf(log_buffer, "[%lld] Não foi possível calcular a
frequência[%i].",esp_timer_get_time(),tempPin);
       ESP_LOGE("calcFrequency", "%s", log_buffer);
       write_to_file("/spiffs/temp.txt", log_buffer);
     }
   }

   int getPinTemp(){
     if(ctrTempPin == 0){
       ctrTempPin++;
       return TEMP_PIN_1;
     } else if(ctrTempPin == 1){
       ctrTempPin++;
       return TEMP_PIN_2;
     } else {
       ctrTempPin=0;
       return TEMP_PIN_3;
     }
   }
```

```
    // //INTERRUPÇÃO PARA DELETAR E REINICIAR ESP
    // static volatile bool deleteAndRestartRequested = false;

    // static void IRAM_ATTR button_isr_handler(void* arg) {
    //     deleteAndRestartRequested = true;
    // }

    void app_main(void) {
      // Configuração do led interno
      esp_rom_gpio_pad_select_gpio(LED_PIN);
      gpio_set_direction(LED_PIN, GPIO_MODE_OUTPUT);

      // Configuração do pino que recebe o sinal do sensor
      esp_rom_gpio_pad_select_gpio(TEMP_PIN_1);
      gpio_set_direction(TEMP_PIN_1, GPIO_MODE_INPUT);

      esp_rom_gpio_pad_select_gpio(TEMP_PIN_2);
      gpio_set_direction(TEMP_PIN_2, GPIO_MODE_INPUT);

      esp_rom_gpio_pad_select_gpio(TEMP_PIN_3);
      gpio_set_direction(TEMP_PIN_3, GPIO_MODE_INPUT);

      // Configuração do timer
      const esp_timer_create_args_t timer1Conf = {
        .callback = timerTrigger1,
        .name = "Timer1",
      };

      esp_timer_create(&timer1Conf, &timer1Handle);

      // Cria a fila amostragem 1
      signalQueue = xQueueCreate(BUFFER_SIZE, sizeof(int));
      if (signalQueue == NULL) {
        ESP_LOGE("app_main", "Falha ao criar fila.");
        return;
      }

      // Cria a tarefa de processamento de sinal
          xTaskCreate(signalProcessingTask, "signalProcessingTask", 2048,
NULL, 5, NULL);

      // Cria a fila amostragem 2
      sequenceQueue = xQueueCreate(BUFFER_SIZE2, sizeof(SignalData_t));
      if (sequenceQueue == NULL) {
        ESP_LOGE("app_main", "Falha ao criar fila.");
        return;
      }

      // Inicializa o sistema de arquivos
      if (init_spiffs() != ESP_OK) {
          ESP_LOGE("app_main", "Erro ao inicializar o sistema de arquivos
SPIFFS");
        return; // Não continue se falhar
      }
```

```
    //Verifica arquivo, caso não tenha, crie
    get_file_info("/spiffs/temp.txt");
    // delete_file("/spiffs/temp.txt");

    while(1){//loop que irá coletar a temperatura de hora em hora.
      vTaskDelay(10000/ portTICK_PERIOD_MS); // Espera por 10
      int loop=0;
        while (loop<=8) {//Coleta a temperatura dos 3 sensores 3 vezes,
por isso o loop roda 9 vezes.
       if (start) {//Iniciar o timer somente se start for true
         calcFrequency();
         clearQueue(sequenceQueue);
         start = false;
          tempPin = getPinTemp();//Modificação para coletar informação de
3 sensores
         gpio_set_level(LED_PIN, 1);
         esp_timer_start_periodic(timer1Handle, AMOSTRAGEM_US);
       }
       vTaskDelay(10000 / portTICK_PERIOD_MS); // Espera por 10 segundos
       loop++;
      }

      ESP_LOGI("main", "Aguardando para próxima amostra...");
           write_to_file("/spiffs/temp.txt", "Aguardando  para  próxima
amostra...");
      // get_file_info("/spiffs/temp.txt");
      vTaskDelay(3600000 / portTICK_PERIOD_MS); // Espera por 1h
    }


    // Para pegar arquivo
    // while(1){
    //     read_from_file("/spiffs/temp.txt");
    //     vTaskDelay(600000 / portTICK_PERIOD_MS);
    // }
  }
```