ÁBACO ELETRÔNICO COM SINTETIZADOR DE VOZ

BELÉM

2023

IDENTIFICAÇÃO

Este projeto faz parte da avaliação da disciplina de Circuitos Elétricos, do curso de Engenharia da Computação, no Instituto de Tecnologia, sediado na Universidade Federal do Pará.

A proposição do trabalho é desenvolver um ábaco, do tipo ábaco aberto, com um circuito capaz de indicar a posição das “bolinhas” quando estas forem movimentadas, permitindo assim, realizar o processo de contagem decimal e a realização de operações matemáticas básicas. Em seguida, o circuito deverá processar os valores inseridos no ábaco a fim de reproduzir verbalmente o valor correspondente ao resultado numérico.

Tal produto será desenvolvido pela empresa Oracle Solutions (Tabela 1), com os seus membros sendo descritos logo a seguir (Tabela 2), e as suas respectivas tarefas sendo detalhadas na Seção 2.3.

Tabela 1 - Informações da empresa.

Oracle Solutions - Informações	
Endereço	Prédio Espaço Inovação – Parque de Ciência e Tecnologia, PCT Guamá – Avenida Perimetral da Ciência, Km 01. Guamá, Belém, PA. CEP: 66075-750.
Telefone	91 9 3489-8063
Email	vendas@oracle.br
Site	oraclesolutions.com.br

Fonte: De própria autoria.

Tabela 2 - Divisão de cargos e atividades.

Nome	Matrícula	Função
Danton Araujo Rodrigues da Cunha	202206840011	Projetista de Hardware
Jean Dean Monteiro Pereira	202006840021	Gerente de Projetos
Joel Tavares Miranda	202206840054	Projetista de Software
Kauan Miranda Tavares	202206840033	Projetista de Hardware
Marco Antonio do Espirito Santo Maues Junior	202206840038	Projetista de Software

Fonte: De autoria própria.

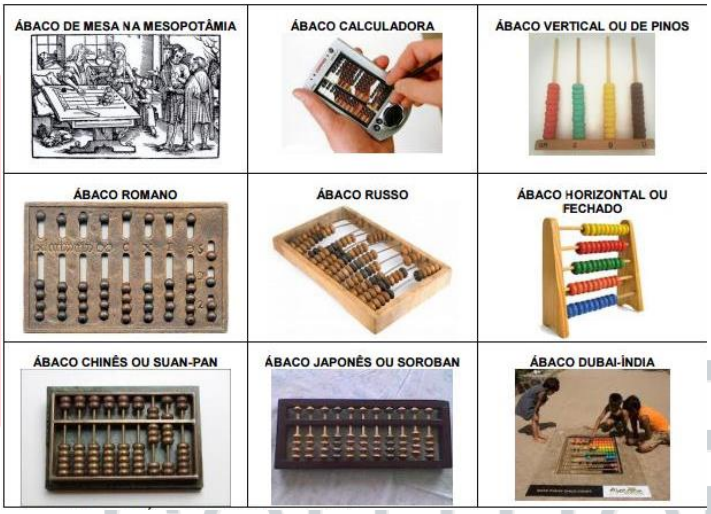
SUMÁRIO

1 INTRODUÇÃO

1.1 Compreensão do problema

Primeiramente, foi feito um estudo sobre a proposição do trabalho, começando pelo ábaco e o seu funcionamento. O ábaco é um antigo instrumento de cálculo que utiliza o Sistema Numérico Decimal, tendo a sua provável origem na Mesopotâmia há mais de 5500 anos a.C. Com o ábaco, conseguimos fazer cálculos e representar os números no sistema de numeração decimal, ou seja, valores de 0 a 9 em cada posição do ábaco, conforme a Imagem 1.

Imagem 1 - Diversos tipos de ábacos existentes.



Fonte: EscolaWeb.

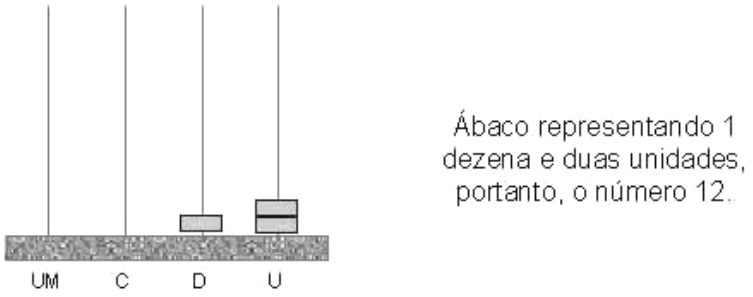
Neste caso, o presente projeto utilizaremos o ábaco aberto, que utiliza pinos verticais para a inserção de argolas, onde cada pino representa uma posição do Sistema Numérico Decimal (base 10) respectivamente, da esquerda para a direita: Unidade de Milhar, Centena, Dezena e Unidade, semelhante à Imagem 2.

Imagem 2 - Ábaco do tipo aberto.

Fonte: Amazon.

As operações matemáticas feitas neste ábaco restringem a adição e subtração, diferentemente de outros tipos de ábaco. Cada argola representa uma unidade no pino referente a posição inserida, onde cada pino pode receber até 10 argolas, conforme a base 10 do sistema indo-arábico. Por exemplo, o número 12 pode ser representado conforme a Imagem 3.

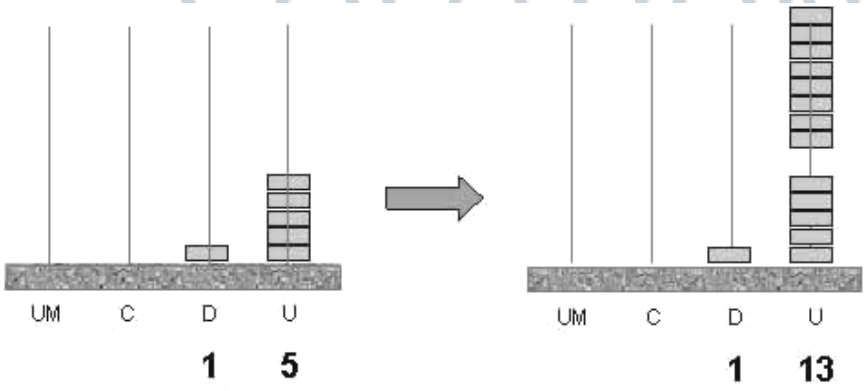
Imagem 3 - Exemplo de representação do valor 12 no ábaco aberto.



Fonte: Moura.

Quando um pino tiver 10 argolas, elas poderão ser substituídas por uma argola a ser inserida no pino de ordem superior, à esquerda, que equivalerá a uma unidade da ordem subsequente, semelhante ao exemplo da soma de 15 + 8, como mostra a Imagem 4.

Imagem 4 - Exemplo de representação da soma de 15 + 8 no ábaco aberto.



Fonte: Moura.

Tal instrumento é utilizado amplamente na educação básica no ensino de matemática e operações aritméticas. Todavia, atividades lúdicas utilizando o ábaco aberto, com peças coloridas e ornamentos para o público infantil não possui adaptações ou implementações tecnológicas para pessoas cegas, apesar de outros tipos de ábacos como o Soroban serem utilizados para o ensino de deficientes visuais (LARAMARA, 2020).

2 OBJETIVOS

A princípio, temos os seguintes objetivos, onde pretendemos atingir o primeiro objetivo na Fase 1.

- Simular um circuito que realize a contagem de elementos semelhante ao funcionamento de um ábaco.
- Testar uma proposta inicial do funcionamento do circuito com a sintetização de voz.
- Desenvolver um protótipo funcional do ábaco com sintetizador de voz.

3 METODOLOGIA

3.1 Desenvolvimento do ábaco

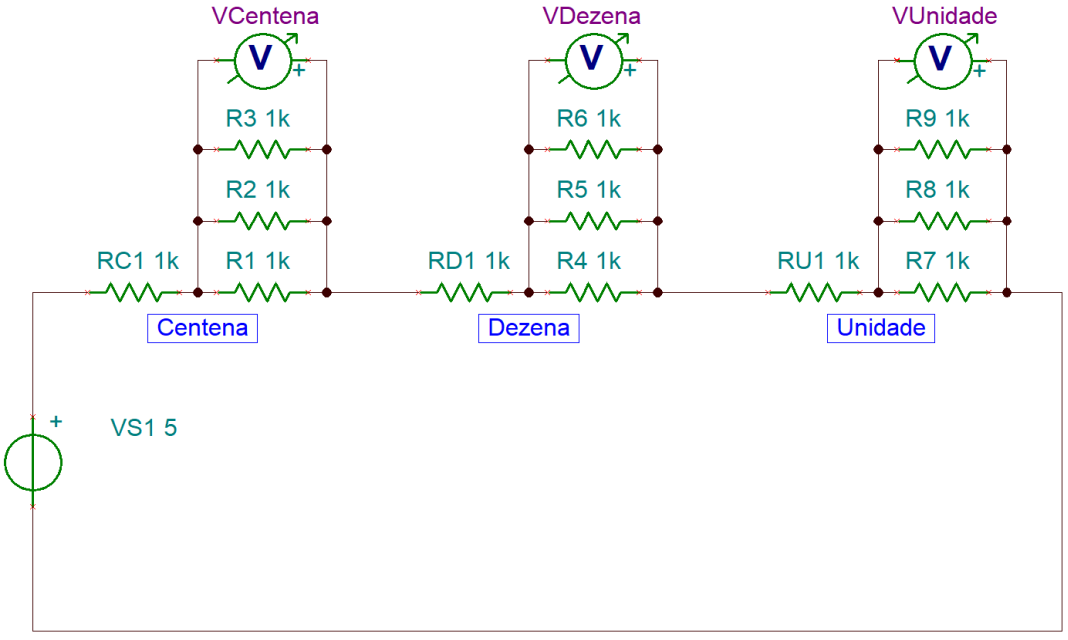
O desenvolvimento do nosso ábaco partiu de duas problemáticas: a contagem de elementos em cada pino do ábaco, a sintetização de voz para a verbalização dos números e resultados de operações matemáticas. As possíveis opções de componentes foram ponderadas em vantagens e desvantagens, além de terem as suas funcionalidades simuladas e testadas. A partir disso, a escolha do componente responsável por processar e reproduzir esses valores foi feita.

3.1.1 Contagem de elementos no ábaco

3.1.1.1 Elaboração do funcionamento do circuito

A contagem de argolas no pino de cada posição será feita a partir do cálculo para determinar o valor de uma resistência desconhecida em cada pino do circuito utilizando um resistor de valor conhecido em série com um conjunto de resistores de valores desconhecidos, com essa configuração desconhecida sendo uma associação de resistores em paralelo, que funcionariam como as argolas do ábaco, para assim, ser possível contabilizar o número de resistores, semelhante à representação do valor 333 conforme a Imagem 5. Esse valor poderia ser obtido através da Equação (1), referente a divisão de tensão.

Imagem 5 - Esquema de circuito representando o número 333.



Fonte: Tina-TI.

Para determinar a quantidade de resistores na associação em paralelo em cada haste, foi utilizado a propriedade de divisão de tensão em resistores de mesmo valor, conforme a Equação 1. Ajustando algebricamente, a partir da Equação 2, temos o valor da resistência caso esses resistores em paralelo estivessem em série. Como desejamos saber apenas a quantidade de resistores em um circuito com todos os resistores iguais, podemos dividir por R_2 e utilizar a Equação 3.

$$\frac{v_{out}}{v_{in}} = \frac{R_1}{R_1 + R_2} \tag{1}$$

$$\frac{v_{out}}{v_{in}} = \frac{R_1}{R_1 + R_2} \Rightarrow R_2 = R_1\left(\frac{v_{in}}{v_{out}} - 1\right) \tag{2}$$

$$R_2 = \frac{R_1\left(\frac{V_{in}}{V_{out}} - 1\right)}{R_1} \approx \left(\frac{V_{in}}{V_{out}} - 1\right) \tag{3}$$

Onde:

R_1 é o valor de resistência conhecido.

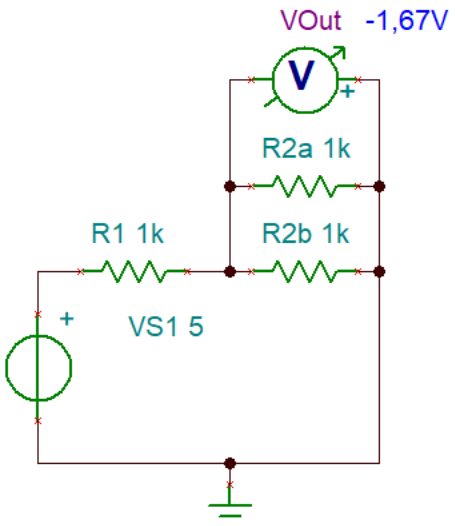
R_2 é a quantidade aproximada de resistores associados em paralelo.

R_{in} é o valor de tensão de entrada.

R_{out} é o valor de tensão de saída.

Vamos aplicar um exemplo: digamos que vamos inserir o valor 2 em uma posição qualquer no ábaco, conforme a Imagem 6.

Imagem 6 - Esquema de circuito representando o número 2.



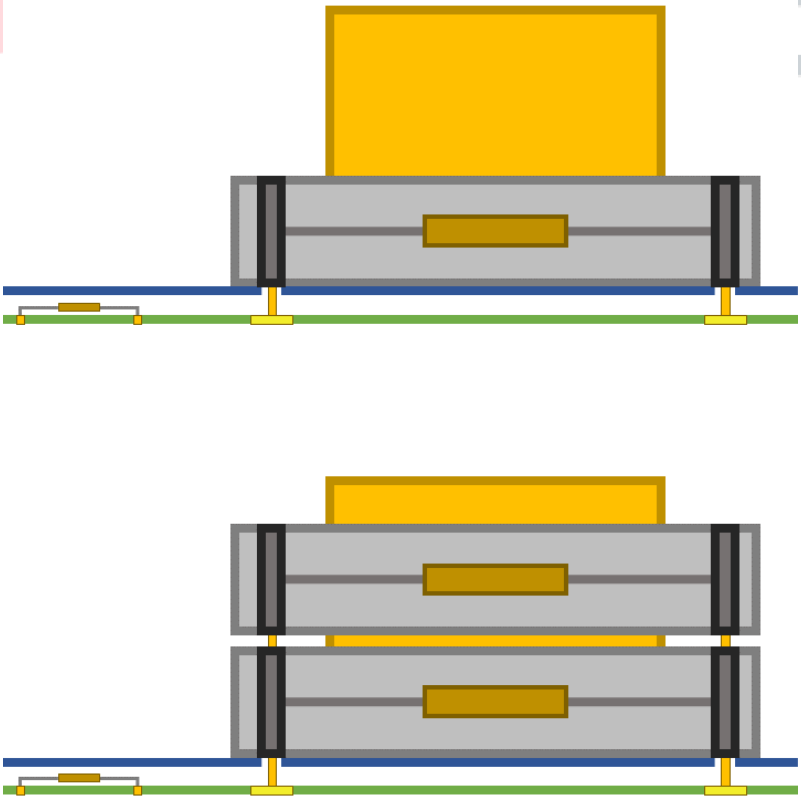
Fonte: Tina-TI.

$$R_2 = \left(\frac{V_{in}}{V_{out}} - 1\right) \Rightarrow \left(\frac{5V}{-1,67V * (-1)} - 1\right) \approx 2 \qquad (4)$$

Com isso, conforme a Equação 4, notamos que podemos obter, a quantidade de resistores associados em paralelo em um determinado trecho do circuito. Essa equação não possui sentido conceitual dentro do contexto de circuitos, porém, não serve para obtermos o valor de resistência equivalente no ponto. Mais adiante, na Seção 4.1, tal equação é implementada e foi atestado que funciona corretamente.

Já a implementação estrutural não fugiria da ideia proposta para o circuito, com a contagem de argolas no pino de cada posição podendo ser feita a partir do "empilhamento" de resistores, que ao se encaixarem, formariam uma associação de resistores em paralelo. Todas as argolas teriam resistores de mesmo valor, encapsulados na sua construção interna e conectores, que permitiriam encaixar no circuito ou em outras argolas, como mostra a Imagem 7.

Imagem 7 - Representação do funcionamento do circuito (vista frontal).



Fonte: De autoria própria.

3.1.2 Verbalização dos números e resultados

A sintetização de voz, para a verbalização dos valores inseridos e dos resultados da operação matemática, é um requisito que exige uma complexidade maior ao dispositivo. Para isso, a partir de pesquisas referente a projetos de sintetizadores de voz utilizando Arduino ou utilizando Raspberry Pi, chegamos nos seguintes resultados, conforme as Tabelas 3 e 4:

Tabela 3 - Soluções de sintetização de voz encontradas para o Arduino.

Soluções para a sintetização de voz - Arduino	
Solução 1: Usando a biblioteca **Talkie** para o Arduino	
Vantagens	- É suportado nativamente pelo Arduino
Desvantagens	- Não tem suporte para a língua portuguesa - Não seria possível falar valores grandes por extenso, por exemplo “trezentos e dezesseis”
É necessário componentes extras?	Sim, é necessário o módulo amplificador de áudio MAX98365A para que a voz seja enviada para um alto falante.

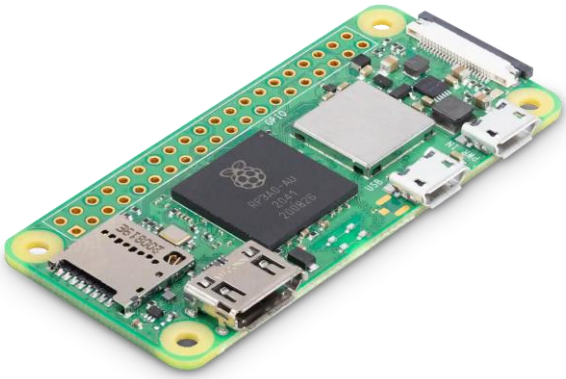
Fonte: De autoria própria.

Tabela 4 - Soluções de sintetização de voz encontradas para o Arduino.

Soluções para a sintetização de voz - Raspberry	
Solução 1: Usando a biblioteca **pyttsx3** em Python	
Vantagens	- Poucas linhas de código em Python executam a sintetização de voz. - Possui suporte para a língua portuguesa. - Reproduz o nome de valores por extenso. - Possui uma ótima qualidade de sintetização no Windows.
Desvantagens	- Não permite ser executado nativamente pelo Arduino. - A execução de sintetizador no Linux é de péssima qualidade, semelhante ao Talkie do Arduino.
É necessário componentes extras?	Sim, é necessário o módulo amplificador de áudio MAX98365A para que a voz seja enviada para um alto falante.
Solução 2: Usando a biblioteca **gTTS** em Python	
Vantagens	- Poucas linhas de código em Python executam a sintetização de voz. - Possui suporte para a língua portuguesa. - Reproduz o nome de valores por extenso. - Possui uma ótima qualidade de sintetização no Linux e no Windows.
Desvantagens	- Não permite ser executado nativamente pelo Arduino. - Necessita de conexão à Internet.
É necessário componentes extras?	Sim, é necessário o módulo amplificador de áudio MAX98365A para que a voz seja enviada para um alto falante.

Fonte: De autoria própria.

Imagem 8 - Raspberry Pi Zero 2 W.

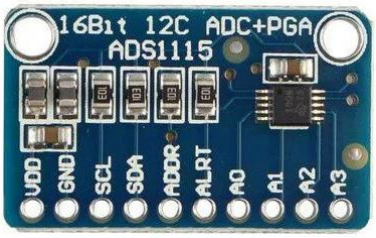


Fonte: RaspberryPi.

Portanto, a partir dos resultados pontuados, decidiu-se utilizar o Raspberry Pi 2 W (Imagem 8) como dispositivo que não apenas realizará a contagem de elementos, como também irá fazer o processamento para a verbalização dos valores inseridos e operados no ábaco por meio da biblioteca gTTS, que também consta entre as bibliotecas avaliadas por Junior (2020). Devido ao seu poder de processamento, o sistema Raspberry Pi OS baseado no Debian Linux, é capaz de executar scripts em Python, permitindo que o desenvolvimento do código seja mais fácil, além de também excluir a necessidade de se utilizar um computador para o funcionamento do dispositivo, ou seja, facilita a utilização do produto pelo público, uma vez que seria necessário apenas plugá-lo na tomada. Apesar da biblioteca gTTS exigir a conexão de internet, o Raspberry Pi é capaz de conectar-se à uma rede WiFi, fato que permitiria a execução do código sem a necessidade de comprar.

Todavia, diferentemente do Arduino, o Raspberry Pi Zero 2W não possui portas analógicas, inviabilizando a leitura direta dos valores de tensão, que por sua vez, impediria a contagem de elementos inseridos no ábaco. Para resolver isso, será utilizado o módulo conversor analógico-digital ADS1115, também utilizado em alguns projetos com Arduino, conforme a Imagem 9. Este módulo possui entradas para realizar a leitura de valores analógicos que são convertidos para valores digitais, para em seguida, serem enviados para o Raspberry Pi, permitindo assim, a utilização do Raspberry Pi.

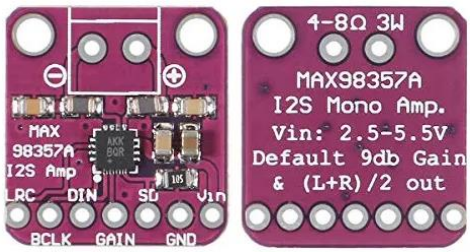
Imagem 9 - Módulo conversor analógico-digital ADS1115.



Fonte: Eletrogate.

Além disso, tanto o Arduino quanto o Raspberry Pi Zero 2W não possuem conexões diretas para a saída de áudio, com isso, será necessário utilizar o módulo amplificador de áudio MAX98357A, semelhante à Imagem 10, que recebe entrada de áudio digital, não apenas decodificando em analógico, mas também amplificando-o diretamente em um alto-falante ao conectá-lo em sua saída.

Imagem 10 - Módulo conversor analógico-digital ADS1115.



Fonte: Carrefour.

Por último, como saída, utilizaremos um auto falante de 1Ω/2W a ser conectado no módulo MAX98357A, permitindo assim, a reprodução de áudios pelo Raspberry Pi, semelhante à Imagem 11.

Imagem 11 - Alto falantes semelhantes ao que será utilizado no projeto.

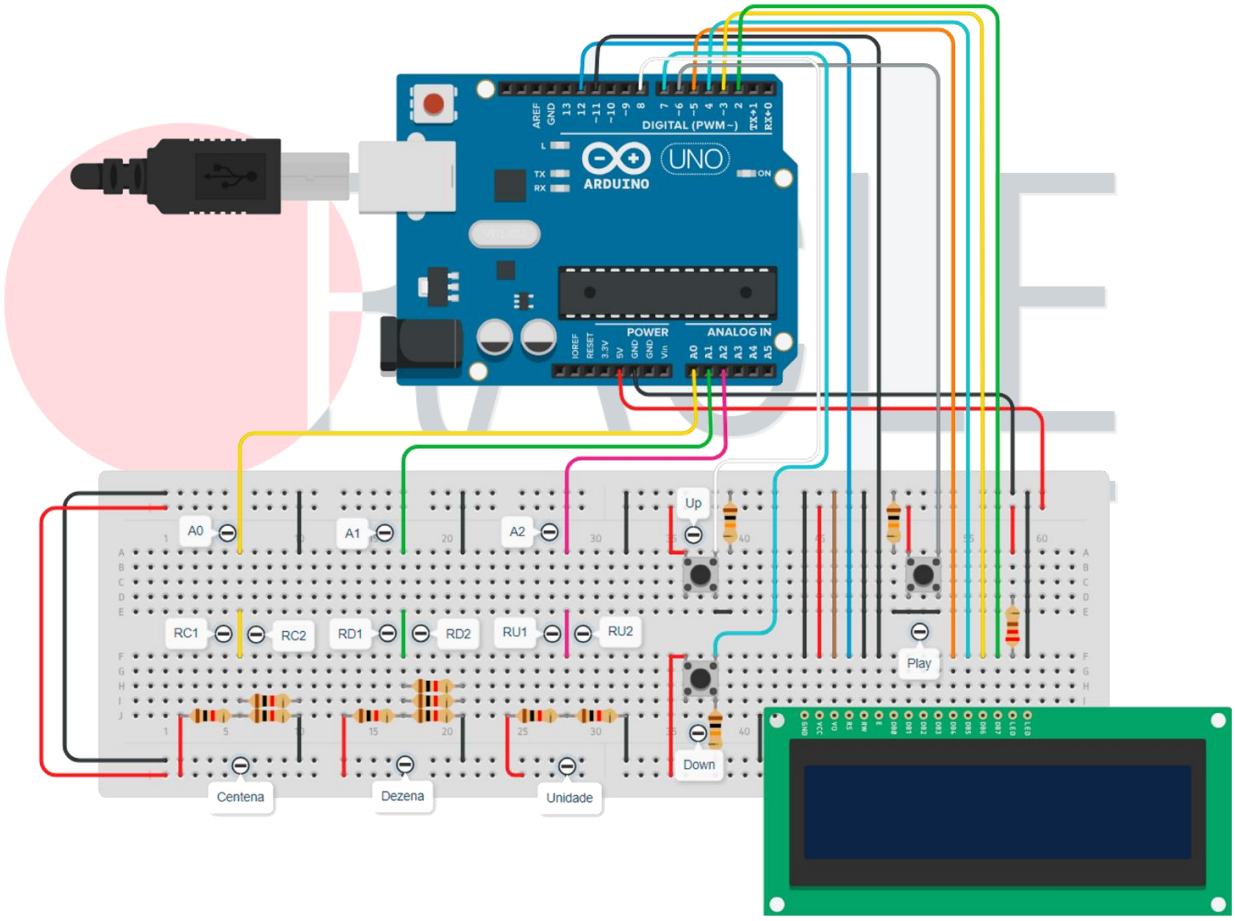
Fonte: Amazon.

4 RESULTADOS

4.1 Simulação no TinkerCAD

Para testarmos o funcionamento do circuito, foi feita uma simulação do circuito utilizando um display LCD 16x2, a fim de facilitar a visualização dos valores inseridos durante os testes. Tal simulação tinha como objetivo atestar se a contagem de elementos inseridos no circuito poderia ser, de fato, realizada. E para isso, foi utilizado a plataforma TinkerCAD da Autodesk.

Imagem 12 - Esquema do circuito implementado no TinkerCAD.



Fonte: TinkerCAD.

Seguindo o mesmo princípio de funcionamento do circuito apresentado na Imagem 12, neste circuito com Arduino, também foram utilizados resistores de valores conhecidos (RC1, RD1 e RU1) em série com um conjunto de resistores em paralelo que teriam um valor indeterminado. Entre esses dois resistores (de valor conhecido e de valor indeterminado), foram inseridos jumpers que são conectados nas portas analógicas A0,

A1 e A2. Essas portas recebem valores analógicos que são convertidos para valores de tensão que são utilizados para o cálculo da condutância.

Para determinar a quantidade de resistores em paralelo em cada haste, foi utilizado uma adaptação da propriedade de divisão de tensão em resistores conforme a Equação 3.

Tratando-se do Arduino, essa tarefa é exercida pela função `lerValor()`, a qual utiliza a função `analogRead()`, a qual lê o valor de um pino analógico especificado. A placa Arduino possui um conversor analógico-digital 10 bits de 6 canais. Isso significa que este irá mapear tensões entre 0 e a tensão operacional (5V ou 3.3V) para valores inteiros entre 0 e 1023. No Arduino UNO, por exemplo, isso permite uma resolução entre leituras de: 5 volts / 1024 unidades, ou .0049 volts (4.9 mV) por unidade.

Considerando o dado de leitura a_0 do pino A_0, o cálculo da quantidade de resistores em paralelo em uma haste é dado por:

$$r := a_o \rightarrow b := r \times v_{in} \tag{3}$$

$$v_{out} := \frac{b}{1024} \tag{4}$$

$$b := \frac{v_{in}}{v_{out}} - 1 \tag{5}$$

$$R_2[i] :\approx \frac{R_1 \left(\frac{V_{in}}{V_{out}} - 1 \right)}{R_1} \tag{6}$$

Onde $R_2[i]$ é a quantidade de resistores na haste i, calculada por meio da função `lerValor()`. Adotando o mesmo valor de resistência para todos os resistores utilizados, concluímos que o número de resistores, e, portanto, de argolas em uma haste, é dada por $R_2[i]/R_b$, onde R_b é a resistência-base dos resistores. Note que $R_1 = R_b$.

A partir da obtenção do valor da resistência desconhecida R_2, ao inserir resistores de mesmo valor em paralelo, os valores das resistências associadas em paralelo são somados. Como todos os resistores utilizados possuem o mesmo valor de 1K Ohm, o código divide o valor obtido por 1000, que resulta em valores unitários representando a quantidade de resistores em cada pino. Após fazer a contagem, os valores de cada pino

são salvos em variáveis do tipo float de um vetor, onde a primeira posição do vetor representa a posição da centena, enquanto que a terceira posição representa a unidade. Por fim, esses valores são unificados em uma única variável, também do tipo float, representando finalmente um número entre 0 e 999. Para aproximar do propósito do trabalho, foram implementadas duas operações matemáticas para testar cálculos: adição e subtração. As operações são escolhidas antes de inserir os valores no ábaco.

A fim de facilitar a interação do usuário e permitir a seleção da operação, foram inseridos botões de controle: Play, Up e Down. Os botões Up e Down permitem alternar entre as operações matemáticas disponíveis, enquanto que o botão Play serve para selecionar uma das operações e confirmar a inserção dos valores no ábaco. Os botões Play, Down e Up são ligados nas portas digitais 6, 7 e 8 respectivamente, além de serem conectados também em 5V e no GND em série com resistores de 10K ohms.

Tabela 5 - Componentes utilizados na simulação TinkerCAD.

Nome	Quantidade	Componente
U1	1	Arduino Uno R3
RLCD	1	Resistor 220 Ω
RC1 RD1 RU1 R1 R2 R3 R4 R5 R6	9	Resistor 1 kΩ
RDown RPlay RUp	3	Resistor 10k Ω
SBotão Play SBotão Down SBotão Up	3	Botão
ULCD	1	LCD 16x9

Fonte: TinkerCAD.

Com isso, vamos ao funcionamento do circuito do ponto de vista do usuário:

1. Quando o Arduino liga, é exibida uma mensagem "Sistema Ok".
2. O Arduino exibe uma mensagem na tela, onde aparece na primeira linha a mensagem "Operacao:" e na segunda linha, a seguinte mensagem "1.(+) 2.(-)".

3. Usando dois botões, chamados Up e Down, a opção poderia ser alternada, por exemplo, ao apertar o botão Down, a opção selecionada indicaria "1.(+) 2.(-)*". Ao aperta a opção Up, o display mostraria "1.(+)* 2.(-)".
4. Ao apertar um terceiro botão Play, seria selecionada a opção desejada, onde "1.(+)" é a operação de soma e "2.(-)" é a operação de subtração.
5. Depois, o Arduino exibiria a mensagem "Insira o primeiro valor". E o Arduino aguardaria o botão Play ser pressionado para que, em seguida, o arduino exibisse na segunda linha a mensagem "Valor1: {valorLido}".
6. Depois do primeiro valor, o Arduino exibiria a mensagem "Insira o segundo valor". E o arduino aguardaria o botão Play ser pressionado para que, em seguida, o arduino exibisse na segunda linha a mensagem "Valor2: {valorLido}". Os valores dos resistores associados são lidos pelos pinos A0, A1 e A2 e unificados como um único valor inteiro.
7. Por fim, o Arduino faria a operação matemática selecionada (soma ou subtração), exibindo no display, na primeira linha a mensagem "Resultado:" e na segunda linha, o resultado da operação matemática.

Agora, vamos ao funcionamento a partir do código, presente no Anexo I:

1. O código começa incluindo a biblioteca LiquidCrystal.h, que é usada para controlar o display LCD.
2. O código define os pinos usados para conectar o display LCD e os pinos analógicos usados para ler valores. Ele também define etiquetas para cada pino analógico.
3. O código declara várias variáveis e constantes que serão usadas posteriormente, incluindo variáveis para armazenar os valores lidos dos pinos analógicos, configurações do botão e informações sobre a operação selecionada.
4. A função setup() é executada uma vez quando o dispositivo é ligado ou reiniciado. Nesta função, o código realiza as seguintes tarefas:
5. Inicializa o display LCD com 16 colunas e 2 linhas.
6. Configura os pinos dos botões como entradas.
7. Exibe uma mensagem de inicialização no LCD.

8. Exibe as opções de operação (adição e subtração) no LCD e aguarda a seleção do usuário.
9. A função loop() é executada continuamente após a função setup().
10. Leitura do estado dos botões Up, Down e Play.
11. Se a operação ainda não foi selecionada (operacaoSelecionada == false), o código permite ao usuário escolher entre adição e subtração usando os botões Up e Down, exibindo a opção selecionada no LCD.
12. Quando o botão Play é pressionado (estadoBotaoPlayAnterior == HIGH), a operação é selecionada, e a mensagem correspondente é exibida no LCD.
13. Após a seleção da operação, o código verifica se o primeiro valor já foi inserido (primeiroValorInserido == false).
14. Se o primeiro valor ainda não foi inserido, o código instrui o usuário a inserir o valor 1 e aguarda até que o botão Play seja pressionado novamente.
15. Quando o usuário pressiona o botão Play, é chamada a função lerValor(), que é usada para ler valores analógicos dos pinos especificados e calcular um valor numérico com base nos valores lidos. Os passos realizados dentro dessa função incluem a leitura dos valores analógicos, o cálculo da resistência e a conversão para um valor numérico.
16. O valor lido é armazenado na variável valorLido e exibido no LCD.
17. Se o primeiro valor já foi inserido, o código instrui o usuário a inserir o valor 2 e aguarda até que o botão Play seja pressionado novamente.
18. O segundo valor é lido, e a operação selecionada (adição ou subtração) é realizada nos valores lidos.
19. O resultado é exibido no LCD.

Portanto, a partir da simulação feita, pôde-se atestar que é possível fazer um circuito, utilizando um Arduino para fazer a contagem de elementos resistivos inseridos em paralelo. Tal teste viabiliza a utilização de placas Arduino e Raspberry Pi para a implementação final do projeto, já que ambos os dispositivos possuem funcionalidades e utilidades semelhantes.

5 PERSPECTIVAS

5.1 Estrutura do ábaco

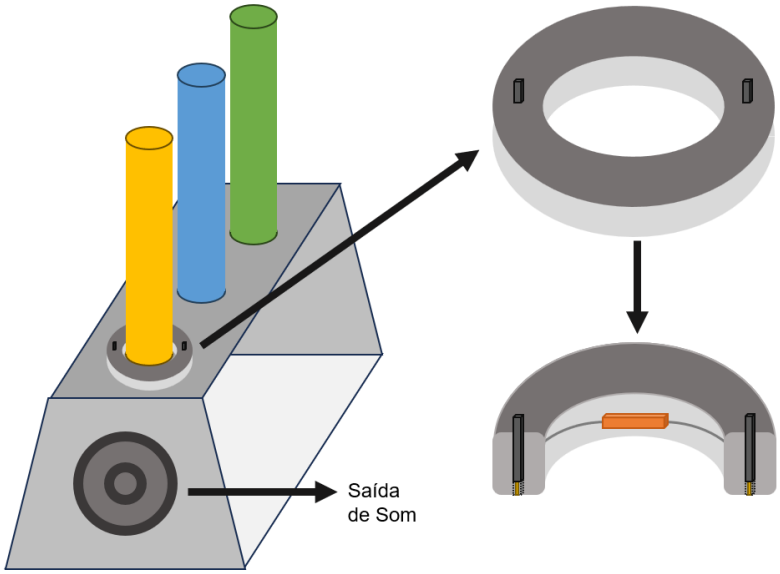
Após atestar a viabilidade do circuito e escolher os componentes, foi possível conceber a construção da estrutura completa do ábaco. Podemos dividir a estrutura em duas partes: a parte interna e a parte externa.

5.1.1 Parte externa do ábaco

Conforme as ilustrações da Imagem 13 e 14, a parte externa do ábaco consiste:

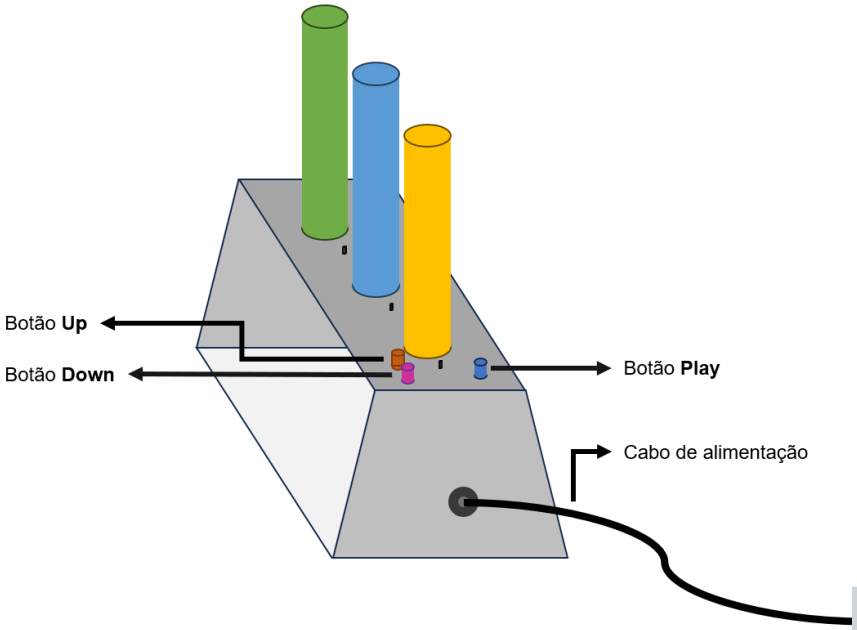
- Hastes verticais (pinos), onde serão inseridas as argolas com resistores de 1K ohm encapsulados.
- Pontos de conexão onde as argolas serão encaixadas.
- Botões de controle Play, Up e Down.
- Alto Falante para a saída de áudio de 1Ω e 2W.
- Cabo de alimentação para conectar o dispositivo na rede elétrica.

Imagem 13 - Ilustração mostrando a composição das argolas e da saída de som.



Fonte: De autoria própria.

Imagem 14 - Ilustração mostrando os botões de controle, cabo de alimentação e os pontos de conexão.



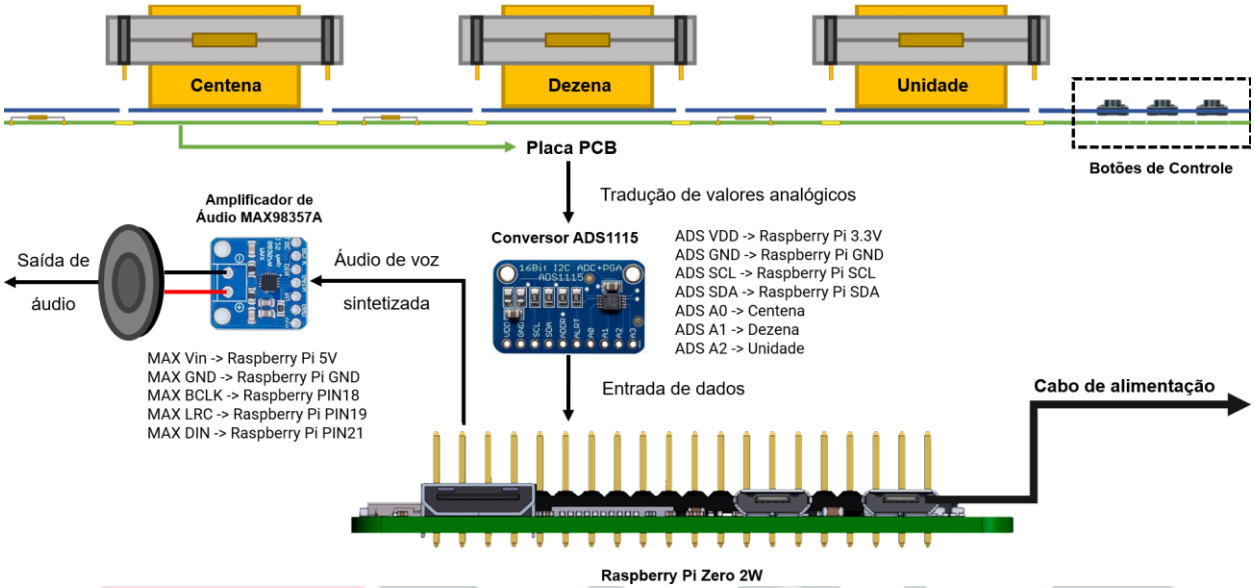
Fonte: De autoria própria.

5.1.2 Parte interna do ábaco

Conforme a ilustração da Imagem 15, a parte interna do ábaco consiste:

- Raspberry Pi Zero 2W, responsável pela contagem das argolas e pelo processamento.
- Conversor ADS1115 que converterá os valores analógicos para digitais, que por sua vez, serão enviadas ao Raspberry Pi Zero 2W.
- Placa PCB que conectará os componentes com conexões em comum, bem como receber os comandos dos botões.
- Botões de controle para permitir a seleção, inserção, correção e reinicialização do sistema.
- Amplificador de áudio MAX98357A que estará conectado na saída de som, controlando-a.

Imagem 15 - Parte interna do ábaco, demonstrando os componentes, suas conexões e funções.

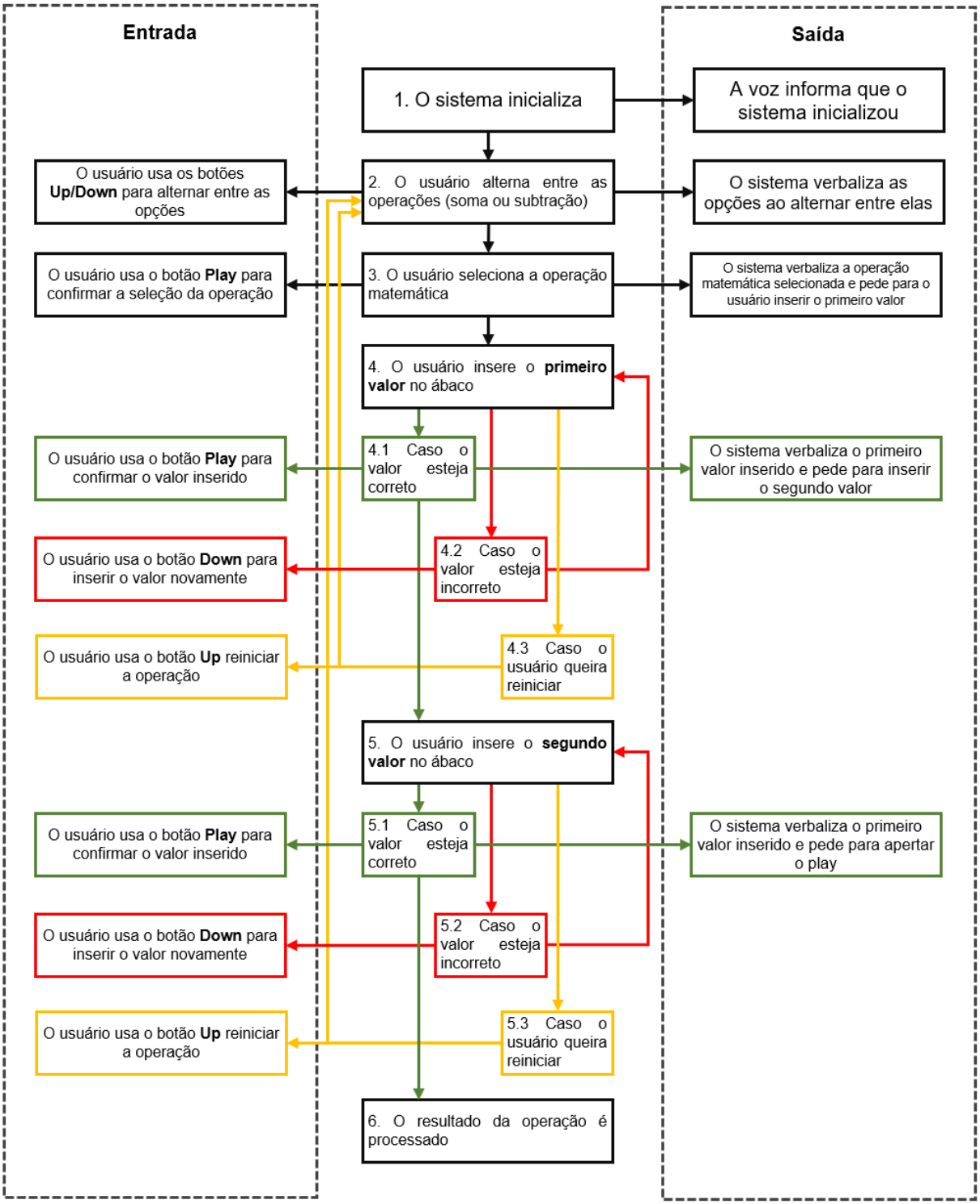


Fonte: De autoria própria.

5.1.3 Lógica do funcionamento do circuito

Foi feito um fluxograma para facilitar a compreensão e a elaboração do circuito, a partir das interações do usuário, bem como a entrada e a saída do circuito, conforme a Imagem 16.

Imagem 16 - Fluxograma do passo a passo do funcionamento do código a ser implementado.



Fonte: De autoria própria.

Tal lógica foi implementada parcialmente no Arduino, funcionando normalmente conforme a simulação descrita na seção 2.1.2.2. Após a escolha do Raspberry Pi Zero 2W, essa lógica terá que ser implementada novamente em Python utilizando esse novo componente. Isso poderá ser feito utilizando a biblioteca GPIO, que controla valores de entrada e saída do Raspberry Pi, semelhante ao funcionamento do Arduino.

5.1.4 Requisitos Mínimos para ser o produto mínimo viável

- Contabilizar corretamente o número de argolas inseridas no ábaco.
- Realizar as operações de soma e subtração.
- Reproduzir, em português, a voz sintetizada referente aos valores inseridos e resultado de operações realizadas.

5.1.5 Visão geral dos componentes do ábaco

Com o exposto acima, podemos então delimitar os componentes necessários para fazer o protótipo, conforme a Tabela 6.

Tabela 6 - Lista de componentes a serem utilizados.

Item	Seção do projeto	Valor	Quantidade	Valor total
Raspberry Pi Zero 2W	Controle	R$ 330,00	1	R$ 330,00
Botões de Controle	Controle	R$ 1,00	3	R$ 3,00
Placa de Fenolite	Controle	R$ 5,00	1	R$ 5,00
Estrutura de MDF	Estrutura	R$ 40,00	1	R$ 40,00
Conversor Analógico-Digital ADS1115	Controle	R$ 5,00	1	R$ 5,00
Amplificador de Áudio MAX98357A	Controle	R$ 30,00	1	R$ 30,00
Resistor 1kΩ - 1% de precisão	Controle	R$ 0,40	33	R$ 13,20
Resistor 10kΩ - 1% de precisão	Controle	R$ 0,60	3	R$ 1,80
Bornes - 3 conexões	Controle	R$ 1,00	2	R$ 2,00
Bornes - 2 conexões	Controle	R$ 1,00	1	R$ 1,00
Pino - 2 conexões	Controle	R$ 1,00	3	R$ 3,00
Alto falante 1Ω e 2W	Controle	R$ 0,60	3	R$ 1,80
			Valor total	**R$ 435,80**

Fonte: De autoria própria.

6 DIFICULDADES

- A ausência de projetos semelhantes disponíveis na internet.
- O custo e disponibilidade dos componentes.
- A falta de soluções para a sintetização de voz na língua portuguesa.

7 CONCLUSÃO

Conforme o exposto, nota-se que o objetivo inicial de se testar a contagem de elementos, semelhante ao funcionamento do ábaco, é possível de ser feito utilizando Arduino. Contudo, ainda será necessário implementar esse mesmo circuito utilizando os componentes escolhidos e testar a sintetização de voz com o Raspberry Pi. Tais implementações serão feitas em fases posteriores.

8 ORGANIZAÇÃO DA EMPRESA E CRONOGRAMA DE EXECUÇÃO

Tabela 7 - Divisão de cargos e atividades.

Fase	Etapas	Função				
		Danton	**Jean**	**Joel**	**Kauan**	**Marco**
Fase 1	F1.1	X	X	X	X	X
	F1.2	X	X	X	X	X
	F1.3	X	X	X	X	X
	F1.4	X	X	X	X	X
	F1.5	X	X	X	X	X
Fase 2	F2.1		X			
	F2.2	X			X	
	F2.3		X	X		X
	F2.4	X	X	X	X	X
	F2.5	X	X	X	X	X
Fase 3	F3.1	X	X	X	X	X
	F3.2	X	X	X		
	F3.3				X	X
	F3.4	X	X	X	X	X

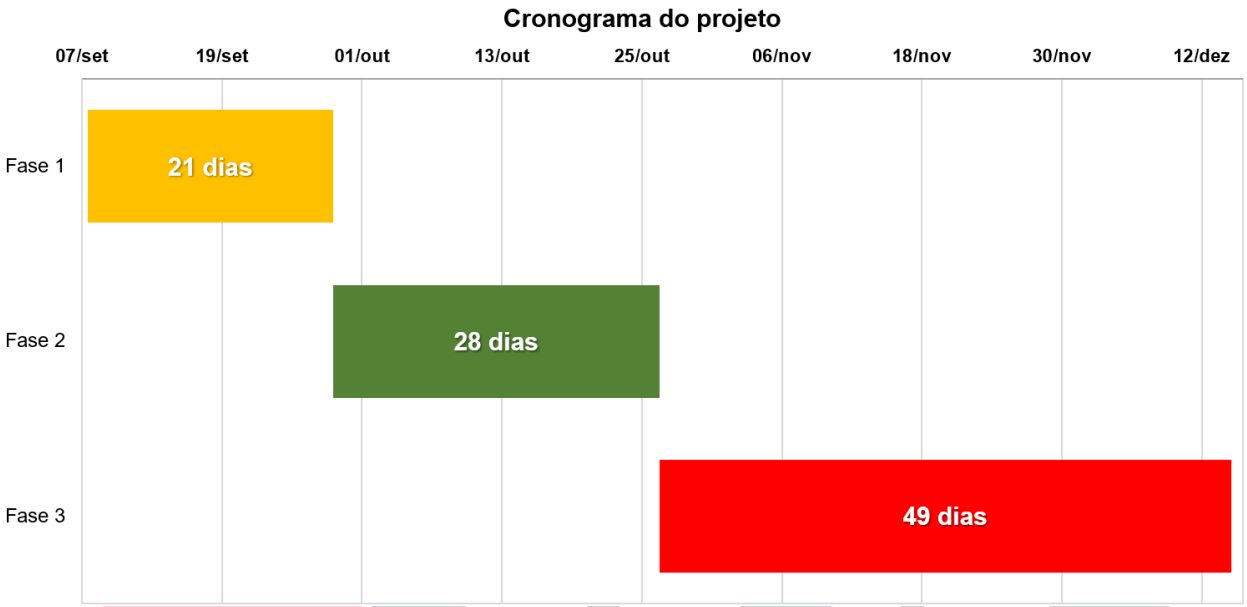
Fonte: De autoria própria.

Quadro 1 - Objetivos, atividades, prazos e responsáveis por Fases.

<table>
<tr><th colspan="3">Fase 1</th></tr>
<tr><td>Objetivo</td><td colspan="2">F1.1: Criar empresa e logomarca.
F1.2: Compreender o problema e soluções já existentes.
F1.3: Documentar as pesquisas realizadas.
F1.4: Estabelecer uma ideia de protótipo.
F1.5: Dividir as tarefas entre os membros da empresa.</td></tr>
<tr><td>Atividades</td><td colspan="2">- Pesquisas bibliográficas
- Simulações, se necessário.
- Testes de componentes, se necessário.
- Definir um nome e criar uma logomarca.
- Elaborar o relatório sobre a Fase 1.</td></tr>
<tr><td rowspan="2">Prazo</td><td>Início</td><td>Fim</td></tr>
<tr><td>08/09/2023</td><td>28/09/2023</td></tr>
<tr><td>Responsáveis</td><td colspan="2">● F1.1: Todos
● F1.2: Todos
● F1.3: Todos
● F1.4: Todos
● F1.5: Todos</td></tr>
<tr><th colspan="3">Fase 2</th></tr>
<tr><td>Objetivo</td><td colspan="2">F2.1: Pesquisa e compra de componentes.
F2.2: Elaborar e testar a placa PCB e os componentes de hardware.
F2.3: Elaborar e testar o código no Raspberry Pi.
F2.4: Testar o protótipo.
F2.5: Verificar problemas e limitações.</td></tr>
<tr><td>Atividades</td><td colspan="2">- Criar uma lista de materiais.
- Simulações.
- Testes de componentes.
- Elaboração do código.
- Elaborar o relatório sobre a Fase 2.</td></tr>
<tr><td rowspan="2">Prazo</td><td>Início</td><td>Fim</td></tr>
<tr><td>29/09/2023</td><td>26/10/2023</td></tr>
<tr><td>Responsáveis</td><td colspan="2">● F2.1: Jean
● F2.2: Danton e Kauan
● F2.3: Jean, Joel e Marco.
● F2.4: Todos
● F2.5: Todos</td></tr>
<tr><th colspan="3">Fase 3</th></tr>
<tr><td>Objetivo</td><td colspan="2">F3.1: Corrigir possíveis problemas e limitações anteriores.
F3.2: Fazer uma pesquisa de qualidade do protótipo.
F3.3: Implementar sugestões de melhorias.
F3.4: Elaborar uma apresentação e um relatório final.</td></tr>
<tr><td>Atividades</td><td colspan="2">- Criar uma lista de materiais.
- Simulações.
- Testes de componentes.
- Elaboração do código.</td></tr>
<tr><td rowspan="2">Prazo</td><td>Início</td><td>Fim</td></tr>
<tr><td>27/10/2023</td><td>14/12/2023</td></tr>
<tr><td>Responsáveis</td><td colspan="2">● F3.1: Todos.
● F3.2: Jean, Danton e Joel
● F3.3: Marco e Kauan
● F3.4: Todos.</td></tr>
</table>

Fonte: De autoria própria.

Imagem 17 - Gráfico de Gantt referente às fases do projeto.



Fonte: De autoria própria.

BIBLIOGRAFIA

Amazon: Aexit 2 PCS 2W 8 Ohm Retangular Multimídia Magneto Interno Alto-falante 35mm x 20mm. Disponível em: https://www.amazon.com.br/Aexit-Retangular-Multimídia-Magneto-Alto-falante/dp/B079WGN59S. Acesso em: 28/09/2023.

Amazon: Ábaco Aberto Com 5 Hastes 50 Argolas Em E.V.A Shrink. Disponível em: https://www.amazon.com.br/Ábaco-Aberto-Hastes-Argolas-Shrink/dp/B07CYK637Z. Acesso em: 28/09/2023.

Arduino e Cia: Usando portas analógicas na Raspberry Pi. 2021. Disponível em: https://www.arduinoecia.com.br/usando-portas-analogicas-na-raspberry-pi-com-ads1115/. Acesso em: 28/09/2023.

Cap Sistema: Como fazer um medidor de Ohm Arduino (Ohmímetro). Disponível em: https://capsistema.com.br/index.php/2021/01/03/como-fazer-um-medidor-de-ohm-arduino-ohmimetro/. Acesso em: 28/09/2023.

Carrefour: Max98357 Módulo Amplificador De Potência De Áudio I2s Classe D. Disponível em: https://www.carrefour.com.br/aitrip-3-pcs-max98357-modulo-amplificador-de-potencia-de-audio-i2s-classe-d-amplificadores-de-audio-mp926063693/p. Acesso em: 28/09/2023.

EletroGate: Módulo Conversor Analógico Digital ADC ADS1115. Disponível em: https://www.eletrogate.com/modulo-conversor-analogico-digital-adc-ads1115. Acesso em: 28/09/2023.

JUNIOR, Ivaldo Faria Gomes Brandão. **Medium**: Reconhecimento de fala e TTS offline em português. Disponível em: https://medium.com/@ivaldobrandao/reconhecimento-de-fala-e-tts-offline-em-português-8f2c67a71dfc. Acesso em: 28/09/2023.

DENTELLA, Luca. 2020. **Lucadentella**: Raspberry Pi Zero, audio output via I2S. Disponível em: http://www.lucadentella.it/en/2017/04/26/raspberry-pi-zero-audio-output-via-i2s/. Acesso em: 28/09/2023.

Laramara: O que é Soroban para cegos. 2020. Disponível em: https://laramara.org.br/o-que-e-soroban-para-cegos/. Acesso em: 28/09/2023.

MOURA, Manoel Oriosvaldo de. Materiais Pedagógicos para o Ensino de Matemática: **ÁBACO**. Disponível em: http://paje.fe.usp.br/~labmat/edm321/1999/material/_private/abaco.htm. Acesso em: 28/09/2023.

Raspberry Pi: Raspberry Pi Zero 2 W. Disponível em:

https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/. Acesso em: 28/09/2023.
SANTOS, Tayna Elias dos. **EscolaWeb**: Você já ouviu falar sobre o Ábaco? Disponível em: https://escolaweb.educacao.al.gov.br/roteiro-de-estudo/titulo-voce-ja-ouviu-falar-sobre-o-abaco-56344. Acesso em: 28/09/2023.
Texas Instruments. 2016. **Tina-TI** (Versão V9) [Software de simulação].

ANEXO I – Código utilizado na simulação com Arduino

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // Configuração dos pinos do LCD

const int numPins = 3; // Número de pinos analógicos
int analogPins[numPins] = {0, 1, 2}; // Define os pinos analógicos a serem lidos
char pinLabels[numPins][3] = {"C:", "D:", "U:"}; // Etiquetas para os pinos analógicos

int raw[numPins];
int Vin = 5;
float Vout[numPins];
float R1 = 1000;
float R2[numPins];
float buffer[numPins];

int valorLido = 0; // Variável para armazenar o valor lido de C, D e U

const int botaoUpPin = 8; // Pino do botão Up
const int botaoDownPin = 7; // Pino do botão Down
const int botaoPlayPin = 6; // Pino do botão Play

int estadoBotaoUpAnterior = HIGH; // Estado anterior do botão Up
int estadoBotaoDownAnterior = HIGH; // Estado anterior do botão Down
int estadoBotaoPlayAnterior = HIGH; // Estado anterior do botão Play

int opcaoSelecionada = 1; // Inicialmente, a opção 1 é selecionada (soma)

bool operacaoSelecionada = false; // Indica se a operação foi selecionada
bool primeiroValorInserido = false; // Indica se o primeiro valor foi inserido

void setup() {
  lcd.begin(16, 2); // Inicializa o display LCD com 16 colunas e 2 linhas
  Serial.begin(9600);

  pinMode(botaoUpPin, INPUT);
  pinMode(botaoDownPin, INPUT);
  pinMode(botaoPlayPin, INPUT);

  // Mensagem de inicialização
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Sistema Ok");
  Serial.println("Sistema Ok");

  delay(2000);

  // Exibe a opção selecionada
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Operacao:");
  Serial.println("Operacao:");
  lcd.setCursor(0, 1);
  Serial.println("1.(+)    2.(-)");
  lcd.print("1.(+)    2.(-)");

}

void loop() {
```

```
  estadoBotaoUpAnterior = digitalRead(botaoUpPin);
  estadoBotaoDownAnterior = digitalRead(botaoDownPin);
  estadoBotaoPlayAnterior = digitalRead(botaoPlayPin);

  // Seleciona a opção com os botões Up e Down
  if (!operacaoSelecionada) {
    if (digitalRead(botaoUpPin) == HIGH) {
      lcd.setCursor(0, 1);
      lcd.print("1.(+)*   2.(-) ");
      Serial.println("1.(+)*   2.(-) ");
      opcaoSelecionada = 1;
    } else if (digitalRead(botaoDownPin) == HIGH) {
      lcd.setCursor(0, 1);
      Serial.println("1.(+)    2.(-)*");
      opcaoSelecionada = 2;
    }
  }

  // Aguarda o botão Play para prosseguir
  if (!operacaoSelecionada && digitalRead(botaoPlayPin) == HIGH && estadoBotaoPlayAnterior ==
HIGH) {
    operacaoSelecionada = true;
    lcd.clear();
    lcd.setCursor(0, 0);
    if (opcaoSelecionada == 1) {
      lcd.print("Operacao: +");
      Serial.println("Operacao: +");
    } else {
      lcd.print("Operacao: -");
      Serial.println("Operacao: -");
    }
    delay(2000);
  }
  if (operacaoSelecionada) {
    if (!primeiroValorInserido) {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Insira o valor 1");
            Serial.println("Insira o valor 1");

      while (digitalRead(botaoPlayPin) == LOW) {
        // Aguarda o botão Play ser pressionado
      }

      valorLido = lerValor();

      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Valor1: ");
      Serial.println("Valor1: ");
      lcd.print(valorLido);
      Serial.println(valorLido);
      lcd.setCursor(0, 1);
      lcd.print("Aguarde...");
      Serial.println("Aguarde...");
      delay(4000); // Aguarda 2 segundos

      primeiroValorInserido = true;
    } else {
      lcd.clear();
```

```
      lcd.setCursor(0, 0);
      lcd.print("Insira o valor 2");
      Serial.println("Insira o valor 2");

      while (digitalRead(botaoPlayPin) == LOW) {
        // Aguarda o botão Play ser pressionado
      }

      int segundoValor = lerValor();
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Valor2: ");
      Serial.println("Valor2: ");
      lcd.print(segundoValor);
      Serial.println(segundoValor);
      lcd.setCursor(0, 1);
      lcd.print("Aguarde...");
      Serial.println("Aguarde...");
      delay(4000); // Aguarda 2 segundos
      valorLido += (opcaoSelecionada == 1) ? segundoValor : -segundoValor;
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Resultado:");
      Serial.println("Resultado:");
      lcd.setCursor(0, 1);
      lcd.print(valorLido);
      Serial.println(valorLido);
      delay(5000);
      operacaoSelecionada = false; // Reinicia o processo
      primeiroValorInserido = false;
    }
  }

  delay(100);
}
int lerValor() {
  int valor = 0;
  for (int i = 0; i < numPins; i++) {
    raw[i] = analogRead(analogPins[i]);

    if (raw[i]) {
      buffer[i] = raw[i] * Vin;
      Vout[i] = (buffer[i]) / 1024.0;
      buffer[i] = (Vin / Vout[i]) - 1;
      R2[i] = R1 * buffer[i];
      valor = valor * 10 + (int)(round(R2[i] / 1000));
/      lcd.setCursor(i * 4, 1);
      lcd.print("       "); // Limpa o espaço anterior
      Serial.println("       "); // Limpa o espaço anterior
      lcd.setCursor(i * 4, 1);
      lcd.print((int)(round(R2[i] / 1000)));
      Serial.println((int)(round(R2[i] / 1000)));
    }
  }
  return valor;
}
```