REMOTERIDE: CARRO ROBÔ CONTROLADO POR APLICATIVO MOBILE DESENVOLVIDO EM FLUTTER

Maria Luiza Paes Martins[1], Carlos Danilo Gaioli Euzebio[1], Anna Patricia Zakem China[1]

[1]Faculdade de Tecnologia de FATEC Ribeirão Preto (FATEC)

Ribeirão Preto, SP – Brasil

marialuizapaes22@gmail.com, carlos.euzebio@fatec.sp.gov.br, anna.china@fatec.sp.gov.br

Resumo. *Este trabalho visa desenvolver um aplicativo mobile de controle remoto e um carro robô, com o objetivo de aprimorar conhecimentos em lógica e pensamento computacional. O projeto envolve a construção do carro robô utilizando um Raspberry Pi Pico como microcontrolador e a criação de um aplicativo de controle remoto desenvolvido com o framework Flutter. Foram utilizadas diversas ferramentas, como o Fritzing para a criação de diagramas de conexão, a IDE Thonny para programação em MicroPython e o Flutter para o desenvolvimento do aplicativo. O projeto permitiu a aquisição de novos conhecimentos, embora tenha enfrentado desafios na integração completa do sistema. Futuramente, busca-se concluir essa integração, aprimorando a comunicação entre os componentes. Este projeto contribuiu significativamente para o desenvolvimento das habilidades dos participantes em pensamento computacional e na integração de hardware e software.*

Abstract. *This project aims to develop a mobile remote-control application and a robot car, with the objective of enhancing knowledge in logic and computational thinking. The project involves building the robot car using a Raspberry Pi Pico as the microcontroller and creating a remote-control application developed with the Flutter framework. Various tools were used, such as Fritzing for creating connection diagrams, the Thonny IDE for programming in MicroPython, and Flutter for developing the application. The project contributed to aquiring new knowledge, although several challenges were faced to integrate the system. In the future, the goal is to complete this integration, improving communication among the components. This project significantly contributed to the participants' skills development in computational thinking and integrating hardware and software.*

1. Introdução

O pensamento computacional, é uma competência essencial e abrangente que transcende a programação de computadores, englobando a resolução de problemas, a concepção de sistemas e a compreensão do comportamento humano através de métodos e modelos da ciência da computação. Este conceito deve ser considerado uma habilidade fundamental, comparável à leitura, escrita e aritmética, e incorporado à educação básica.

O pensamento computacional utiliza ferramentas mentais da ciência da computação, como abstração, decomposição, reconhecimento de padrões, pensamento recursivo e processamento paralelo. Ele permite a reformulação de problemas complexos em problemas solucionáveis, considerando a dificuldade e os recursos computacionais disponíveis. Além disso, o pensamento computacional inclui a capacidade de encontrar soluções aproximadas, utilizar aleatoriedade e lidar com falsos positivos e negativos.

Nesse contexto, o presente trabalho tem como objetivo desenvolver um carro robô e um aplicativo mobile para controle remoto, visando contribuir para a aquisição de novos conhecimentos em pensamento computacional. Assim, será criado um aplicativo que permita o controle eficiente do carro utilizando o *framework* Flutter.

O projeto envolverá a montagem do *hardware*, a programação do carro robô em Python e o desenvolvimento do aplicativo utilizando o *framework* Flutter e a linguagem Dart. A análise do projeto permitirá aprimorar o pensamento computacional dos estudantes envolvidos

Este artigo está estruturado da seguinte maneira: O texto inicia com a Introdução, que aborda o conceito de pensamento computacional, sua relevância e como se relaciona com a robótica e o desenvolvimento de aplicativos móveis. Na Seção 2, são discutidos o papel da robótica como ferramenta educacional, a interconexão entre a robótica e o pensamento computacional, bem como o crescimento e a importância do desenvolvimento de aplicativos móveis na atualidade. A Seção 3 detalha as ferramentas empregadas no projeto, incluindo o framework Flutter, o microcontrolador Raspberry Pi Pico, a IDE Thonny, e o uso do MicroPython para a programação. Na Seção 4, são descritos os processos envolvidos na criação do aplicativo móvel RemoteRide e na montagem do carro robô, desde a aquisição dos materiais até a programação dos componentes. Por fim, na seção 5, são apresentadas as considerações finais, que avaliam os resultados alcançados, destacam os desafios enfrentados e delineiam as perspectivas futuras para o projeto.

2. Desenvolvimento de aplicativos e robótica: Um caminho para o pensamento computacional e criatividade

A robótica moderna está passando por mudanças significativas devido à integração cada vez maior de *hardware* e *software*. Este fenômeno é particularmente evidente no crescente mercado brasileiro de robótica. De acordo com o Mapa do Ecossistema Robótico Brasileiro (MOBILE TIME, 2020), o número de robôs no país aumentou 68% em apenas um ano, relata a Forbes (2021) de 60 mil em 2019 para 101 mil em 2020. Além disso, a robótica desempenha um papel fundamental em diversas indústrias, incluindo manufatura, a medicina, a exploração espacial e agricultura, melhorando a eficiência, a precisão e a segurança do trabalho.

Esta é uma área em crescimento, não apenas como um campo de pesquisa e desenvolvimento tecnológico, mas também como uma ferramenta eficaz de aprendizagem. No contexto educacional, a robótica desempenha um papel vital na promoção do pensamento computacional, principalmente ao processo de pensamento que envolve expressar soluções para problemas computacionais através de algoritmos. Esta área ensina conceitos de programação, e também estimula a resolução criativa de problemas, a lógica de programação e pensamento analítico (WING, 2006).

Kurshan (2016 p. 2) propõe a seguinte definição para o Pensamento Computacional: “O Pensamento Computacional é uma distinta capacidade criativa, crítica e estratégica humana de saber utilizar os fundamentos da computação, nas mais diversas áreas do conhecimento, com a finalidade de identificar e resolver problemas, de maneira individual ou colaborativa, através de passos claros, de tal forma que uma pessoa ou uma máquina possam executá-los eficazmente”. Ademais, uma forma de estimular o pensamento computacional se dá através do desenvolvimento de aplicativos para dispositivos móveis, pois aumenta a criatividade e estimula o desenvolvimento de aplicações para resoluções de problemas do cotidiano com o uso celular. Em 2022, 160,4 milhões de pessoas no país possuíam telefone celular para uso pessoal, o que representa 86,5% da população com 10 anos ou mais, segundo

dados da Pesquisa Nacional de Domicílios Contínua de Tecnologia da Informação e Comunicação, feita pelo IBGE (BRASIL, 2022). Esse número significativo destaca não apenas a presença massiva dos celulares na sociedade brasileira, mas também as possibilidades de os utilizar como ferramentas educacionais.

Além disso, neste contexto, o conceito de "prosumidor", introduzido por Alvin Toffler (1980) recebe uma nova perspectiva, pois o aluno não apenas consome informação, mas também se torna um produtor ativo. Não se trata apenas de utilizar, mas de criar soluções próprias, e expandir suas habilidades técnicas no processo. Essa abordagem não apenas desafia a visão tradicional do uso da tecnologia na educação, mas também capacita o aluno como um criador, permitindo-lhe transformar a teoria em prática em projetos significativos.

3. Materiais e Métodos

Nesta seção, explora-se mais cada uma das ferramentas essenciais para o desenvolvimento do projeto.

3.1 Modelos de contexto

Na fase inicial de definição do sistema, é interessante decidir os escopos e limites do sistema. Isso significa decidir quais recursos devem ser incluídos no sistema e o que o ambiente do sistema oferece. Segundo Ian Sommerville (2011), todo processo precisa ser, de alguma maneira, documentado.

Com isso, optou-se por utilizar diagrama de classe, diagrama de caso de uso e levantamento de requisitos funcionais e não funcionais incorporados à UML (SOMMERVILLE e SAWYER, 1997; SOMMERVILLE, 2011). A ferramenta usada para a criação destes documentos foi o Astah UML, pois é uma ferramenta fácil de aprender e utilizar (ASTAH, 2024).

3.2 Flutter: Framework para Aplicativos Eficientes e a linguagem por trás Dart

O Flutter foi escolhido para o processo de desenvolvimento do aplicativo. Ele permite a criação de aplicativos nativos para Android e iOS usando a linguagem de programação Dart. O diferencial do Flutter reside na sua capacidade de compilar código nativo para a arquitetura *Advanced RISC Machine* (ARM) em ambas as plataformas, garantindo um desempenho excepcional (FLUTTER, 2021).

O Flutter se baseia no motor de renderização Skia 2D, que é reconhecido por sua versatilidade e é utilizado não apenas nos produtos da Google, mas também em outras aplicações. Além disso, o Flutter oferece a funcionalidade de compilação em tempo real (*hot reload*), acelerando o processo de desenvolvimento de aplicativos (FLUTTER, 2021).

Uma das características distintivas do Flutter são os *widgets*, que representam componentes da interface do usuário, como botões, listas e animações. Esses widgets podem ser categorizados em dois tipos principais: *Stateless* e *Stateful*. Os *Stateless Widgets* são adequados para elementos estáticos, enquanto os *Stateful Widgets* podem armazenar e manipular informações, permitindo uma reutilização eficiente de código e uma abordagem modular para o desenvolvimento de interfaces de usuário complexas. (FLUTTER, 2021)

O Dart é a linguagem de programação que dá suporte ao Flutter, e foi criada pela Google com o objetivo de simplificar o desenvolvimento de aplicações *web* complexas. Ele oferece recursos avançados de programação assíncrona, codificação e decodificação de dados, manipulação de datas e protocolos de comunicação *web*. Sua criação visou aprimorar o desempenho e a experiência do usuário em aplicações *web* (FLUTTER, 2021).

3.3 Raspberry Pi Pico

O Raspberry Pi Pico é uma placa de desenvolvimento baseada no microcontrolador RP2040, que representa a inovação da Raspberry Pi Foundation no campo de microcontroladores. O RP2040 é um processador ARM Cortex-M0+ dual-core com 264 KB de SRAM interna e suporte para até 16 MB de memória *flash* externa para armazenamento de programas. Além disso, opera a uma frequência de 133 MHz, oferecendo uma rica variedade de pinos GPIO, recursos internos e, o que é notável, um preço altamente acessível. A escolha deste microcontrolador para o projeto foi baseada não apenas em sua capacidade técnica, mas também em sua acessibilidade financeira e dimensões compactas (RASPBERRY, 2024).

3.4 Thonny Python IDE

Foi escolhida a IDE Thonny para programar o carro robô, pois simplifica a abordagem inicial à programação. Integrado com Python 3.10, exige apenas a instalação de um único pacote para iniciar, proporcionando uma experiência de aprendizado descomplicada. Sua interface inicial é desprovida de recursos que possam distrair os iniciantes. Após o primeiro contato com os programas, é possível acessar uma tabela de variáveis para visualizar o impacto dos comandos Python. Além disso, Thonny oferece um depurador simplificado, permitindo a execução passo a passo dos programas com facilidade (THONNY,2024).

3.5 A Escolha do MicroPython para Programar o Raspberry Pi Pico

MicroPython é uma implementação completa da linguagem de programação Python, desenvolvida para executar diretamente em *hardware* embarcado, como o Raspberry Pi Pico, com o objetivo de ser o mais compatível possível com o Python padrão, permitindo a fácil transferência de código do desktop para um microcontrolador ou sistema embarcado. Ele oferece um prompt interativo (o REPL) para executar comandos imediatamente via USB Serial, além de um sistema de arquivos integrado. A versão do MicroPython para o Pico inclui módulos para acessar *hardware* específico do chip em um nível mais baixo (RASPBERRY, 2024).

4. Desenvolvimento da Aplicação

Os documentos desenvolvidos durante o processo, como diagrama de caso de uso, documento de requisitos e diagrama de caso de uso, estão disponíveis no GitHub (distribuídos no README.md e no diretório documentos_tcc), através do link: https://github.com/MaaLuu21/CONTROLE-CARRO-ROBO.

4.1 Desenvolvimento do aplicativo mobile RemoteRide

Ao iniciar o desenvolvimento da aplicação RemoteRide, e a primeira tela desenvolvida foi a *Splash Screen* (Figura 1). Segundo Pontes (2018), o termo "*Splash Screen*" apareceu em aplicações desktop, muitas vezes referidas como tela de apresentação. Esses programas foram usados para carregar funções de software para exibir uma pequena tela do logotipo do aplicativo durante a inicialização. Os códigos de todas as estão disponíveis no GitHub, através do link: https://github.com/MaaLuu21/CONTROLE-CARRO-ROBO.

Figura 1. Tela de *Splash Screen*
Fonte: (Autoria própria, 2023)

A próxima tela do aplicativo é a tela principal (figura 2A), responsável por movimentar o carro robô. Possui quatro botões que o usuário pode usar para mover o carro para frente e para trás, virar à direita e à esquerda. Continuando na mesma tela, ao clicar no *pop-up* menu, localizado no canto superior direito (figura 2B), o usuário pode ver o nome do dispositivo *bluetooth* ao qual está conectado.

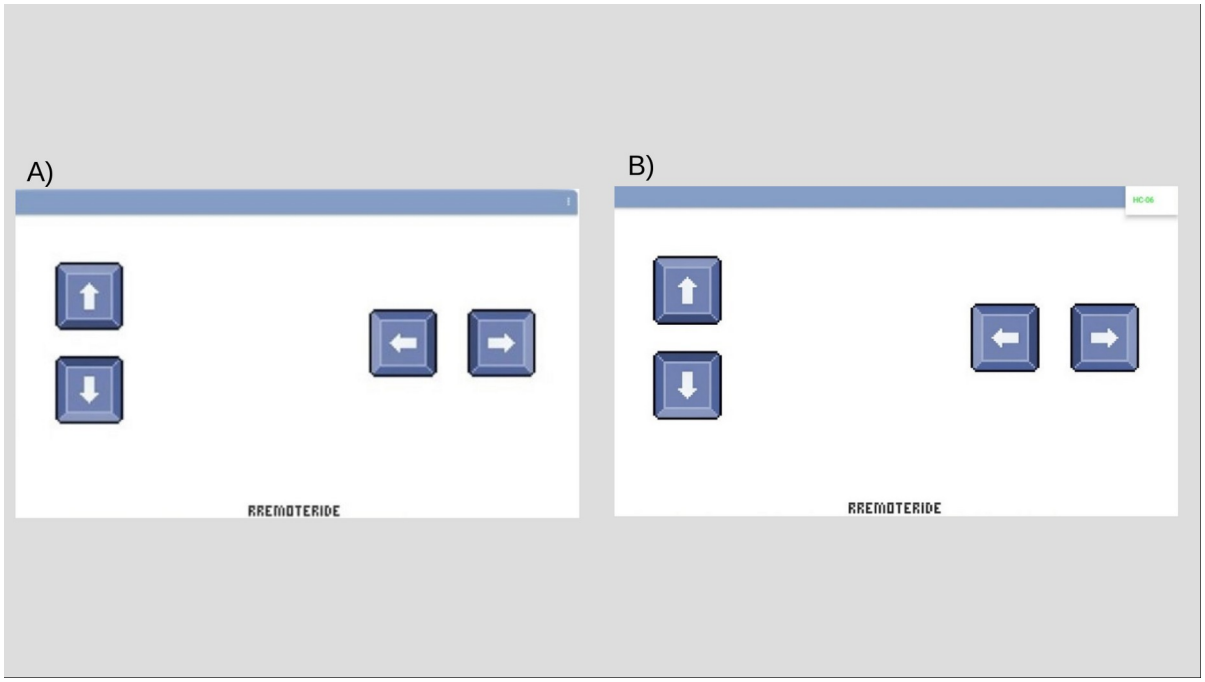


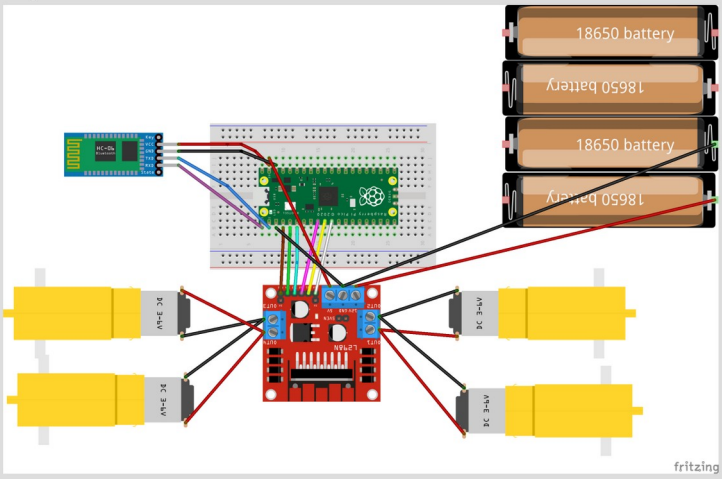
Figura 2. A) Tela principal; B) Pop-up menu
Fonte: (Autoria própria, 2023)

4.2 Desenvolvimento do carro robô

Nesta etapa, inicia-se o desenvolvimento do carro robô. Para tal foram adquiridos os seguintes materiais: (i) a estrutura do carro, (ii) o microcontrolador Raspberry Pi Pico, (iii) uma ponte H L298N, (iv) um módulo *bluetooth* HC-05, (v) quatro motores TT DC, (vi) quatro rodas, (vii) quatro pilhas de 1,5 V, (viii) uma protoboard e (ix) fios jumpers macho/fêmea. As figuras

abaixo (figura 3A e 3B) mostra o diagrama de conexões, desenvolvido com a ferramenta Fritzing (FRITZING, 2024).

A)



B)

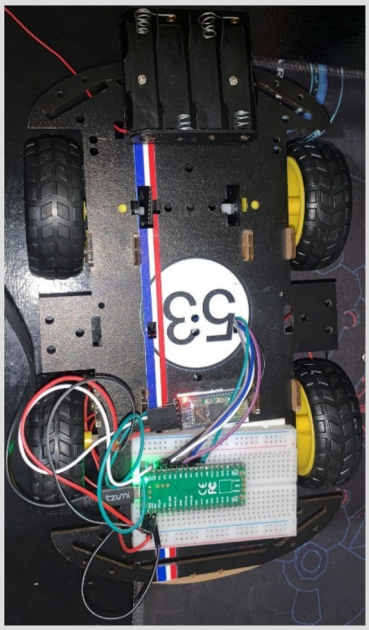
Raspberry Pi Pico	HC-05
VSYS (PIN 389)	VCC
GND	GND
GPIO 0 (PIN 1)	TXD
GPIO 1 (PIN 2)	RXD

Raspberry Pi Pico	Ponte H L298N
GPIO 2 (PIN 4)	ENA
GPIO 3 (PIN 5)	IN1
GPIO 4 (PIN 6)	IN2
GPIO 5 (PIN 7)	IN3
GPIO 6 (PIN 9)	IN4
GPIO 7 (PIN 10)	ENB
VSYS (PIN 39)	5V

Figura 3. A) Diagrama de conexões B) Tabela das ligações
Fonte: (Autoria própria, 2023)

A montagem física do carro teve início após a aquisição dos materiais e a elaboração do diagrama. A primeira imagem (figura 4A), mostra a parte superior do carro, onde é possível localizar o microcontrolador, responsável pela programação do carro. Já a segunda imagem (figura 4B), mostra a parte interna do carro, onde fica a ponte H, responsável pela alimentação dos motores.

A)



B)

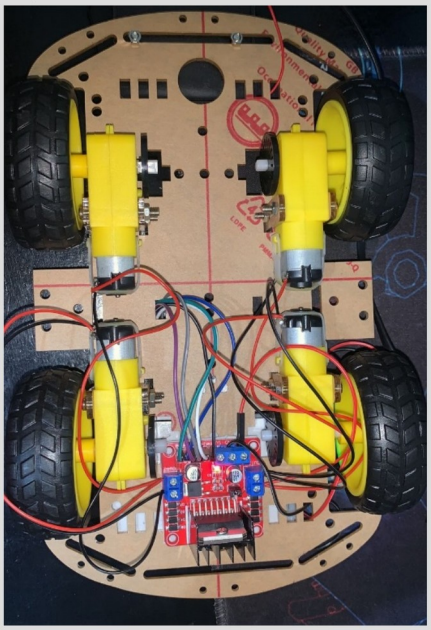


Figura 4. A) Parte superior do carro B) Parte inferior do carro
Fonte: (Autoria própria, 2024)

Após a montagem do carro, o próximo passo foi o desenvolvimento do código responsável pela movimentação do carro. O código (Figura 4A e 4B) começa importando as bibliotecas necessárias para o controle dos componentes do carro, como motores e comunicação serial.

Em seguida, define os pinos GPIO (Entrada/Saída de Propósito Geral) conectados aos motores do carro e à interface de comunicação serial (UART). IN1, IN2, IN3 e IN4 são os pinos que controlam a direção dos motores, enquanto ENA e ENB são os pinos para controlar a velocidade dos motores através de PWM (Modulação por Largura de Pulso). A velocidade do carro é configurada usando PWM, com uma velocidade máxima definida. Isso permite controlar a potência entregue aos motores e, portanto, a velocidade do carro. O código define quatro funções para os movimentos básicos do carro: para frente, para trás, virar à esquerda e virar à direita. Cada função configura os pinos GPIO de acordo com a direção desejada para os motores do carro.

O programa entra em um loop infinito, onde verifica continuamente se há dados disponíveis na interface de comunicação serial (UART). Se houver dados disponíveis, o programa lê o próximo caractere recebido. Com base no caractere recebido, o programa decide qual função de movimento chamar. Por exemplo, se receber "U", chama a função para mover o carro para frente; se receber "D", chama a função para mover o carro para trás; “L” chama a função para virar à esquerda; “R” chama a função para virar à direita.

A)

```
#inclui as blibliotecas
from machine import UART,Pin, PWM
from time import sleep

# Define os pinos de OUTPUT
ENA = PWM(Pin(2))
IN1 = Pin(3,Pin.OUT)
IN2 = Pin(4,Pin.OUT)
IN3 = Pin(5,Pin.OUT)
IN4 = Pin(6,Pin.OUT)
ENB = PWM(Pin(7))

uart = UART(0, 9600)

# Velocidade do carro
speed = 65025 # 0 - 65025
ENA.duty_u16(speed)
ENB.duty_u16(speed)

#Função responsavel de mover o carro para a frente
def forward():
    IN1.on()
    IN2.off()
    IN3.on()
    IN4.off()
#Função responsavel de mover o carro para a tras
def backward():
    IN1.off()
    IN2.on()
```

B)

```
    IN3.off()
    IN4.on()
#Função responsavel de mover o carro para a esquerda
def left():
    IN1.on()
    IN2.off()
    IN3.off()
    IN4.on()
#Função responsavel de mover o carro para a direita
def right():
    IN1.off()
    IN2.on()
    IN3.on()
    IN4.off()
#loop
while True:
    if uart.any():
        value = uart.readline()
        print(value)

        if value == b'U':
            forward()
        elif value == b'D':
            backward()
        elif value == b'L':
            left()
        elif value == b'R':
            right()
```

Figura 5. A) Primeira parte do código B) Segunda parte do código
Fonte: (Autoria própria, 2024)

5. Considerações Finais

O desenvolvimento da aplicação RemoteRide e do carro robô teve como objetivo principal a aquisição e a aplicação de novos conhecimentos em lógica computacional e programação, além de estimular habilidades em pensamento computacional.

Os objetivos do projeto foram em grande parte alcançados. Novos conhecimentos foram adquiridos, como o uso de ferramentas específicas para o desenvolvimento do projeto. Entre essas ferramentas, destacam-se o Fritzing, utilizado para a criação dos diagramas de conexão, facilitando a visualização e a montagem do circuito; a IDE Thonny, escolhida pela sua simplicidade e facilidade de uso, foi fundamental para a programação do carro robô

utilizando MicroPython; e o Flutter, através do desenvolvimento do aplicativo de controle remoto, que permitiu explorar mais o framework, aprendendo a implementar novos widgets e a forçar a orientação inicial do dispositivo.

A programação do carro robô com MicroPython representou um aprendizado significativo, permitindo uma abordagem prática da linguagem Python. O projeto também proporcionou a experiência de lidar com uma nova biblioteca de controle de hardware.

Infelizmente, devido a algumas dificuldades, como a lógica por trás do aplicativo não foi possível concluir a integração total do carro robô com a aplicação. Como encaminhamento futuro, pretende-se concluir esta integração, resolvendo as pendências lógicas e aperfeiçoando o sistema de comunicação entre os componentes.

Em resumo, o projeto RemoteRide alcançou seus objetivos educacionais e técnicos em grande parte, contribuindo para o desenvolvimento das competências dos participantes em pensamento computacional. A experiência prática adquirida será valiosa para projetos futuros, e a base construída permitirá a exploração de novas áreas. A conclusão da integração total dos sistemas permanece como uma meta a ser atingida em projetos subsequentes.

6. Referências bibliográficas

ASTAH. Powerful and Fast UML Diagramming Software, 2024. Disponível em: <https://astah.net/products/astah-uml/>. Acesso em: 3 mar 2024.

BRASIL. Celular Segue Como Aparelho Mais Utilizado Para Acesso à Internet No Brasil, 2022. Disponível em: <www.gov.br/mcom/pt-br/noticias/2022/setembro/celular-segue-como-aparelho-mais-utilizado-para-acesso-a-internet-no-brasil>. Acesso em: 05 mai 2024.

FLUTTER. Flutter documentation, 2023. Disponível em: <https://docs.flutter.dev/>. Acesso em: 25 ago 2023.

FORBES. 7 tendências do mercado de robótica para a próxima década. Disponível em: <https://forbes.com.br/forbes-tech/2021/09/7-tendencias-do-mercado-de-robotica-para-a-proxima-decada/>. Acesso em: 31 ago 2023.

FRITZING. Eletronics made easy Software, 2020. Disponível em: < https://fritzing.org/>. Acesso em: 1 abr 2024.

KURSHAN, B. Thawing from a Long Winter in Computer Science Education. Forbes, p. 2, fev. 2016.

MOBILE TIME. Mapa do Ecossistema Brasileiro de Bots – 2020. Disponível em: <https://www.mobiletime.com.br/pesquisas/mapa-do-ecossistema-brasileiro-de-bots-2020/>. Acesso em: 25 ago 2023.

PONTES, Guilherme. Progressive Web Apps: Construa aplicações progressivas com React. Vila Maria São Paulo: Casa do Código, 2018. 458 p.

RASPBERRY PI. RP2040. Disponível em: <https://www.raspberrypi.com/products/raspberry-pi-pico/>. Acesso em: 2 set 2023.

SOMMERVILLE, I; SAWYER, P. Requirements engineering: a good practice guide. John Wiley & Sons, Inc., 1997.

SPRINGER LINK. Introducing the Raspberry Pi Pico. Disponível em: <https://link.springer.com/chapter/10.1007/978-1-4842-8135-2_1>. Acesso em: 29 ago 2023.

THONNY. Thonny Python IDE for beginners. Disponível em: <https://thonny.org/>. Acesso em: 15 mar 2024.

TOFFLER, Alvin. A Terceira Onda. 2. ed. [S.l.]: Grupo Editorial Record, 1980.

WING, J. M. Computational Thinking. Communications of the ACM, v. 49, n. 3, p. 33-35, 2006.