Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Rupyly: O uso de programação em blocos para modelos robóticos educacionais

Ana Caroline da Rocha Braz

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof.a Dr.a Carla Maria Chagas e Cavalcante Koike

Coorientador
Prof.a Dr.a Dianne Magalhães Viana

Brasília
2025

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Rupyly: O uso de programação em blocos para modelos robóticos educacionais

Ana Caroline da Rocha Braz

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof.a Dr.a Carla Maria Chagas e Cavalcante Koike (Orientador)
CIC/UnB

Prof.a Dr.a Genaína Nunes Rodrigues
Universidade de Brasília

Prof.a Dr.a Leticia Lopes Leite
Universidade de Brasília

Prof. Dr. Marcelo Grandi Mandelli
Coordenador do Bacharelado em Ciência da Computação

Brasília, 18 de julho de 2025

Dedicatória

Dedico este trabalho aos meus familiares, minha mãe, meu pai e meu irmão, por todo amor e apoio incondicional. E também aos amigos que cultivei ao longo dos últimos anos, cuja presença e incentivo foram essenciais para que eu chegasse até aqui.

Agradecimentos

Agradeço, primeiramente, a Deus, pela sabedoria e força, que me sustentaram em cada etapa desta jornada.

Aos meus familiares, em especial à minha mãe, Silvia, e ao meu pai, João, por estarem sempre ao meu lado, oferecendo todo o apoio e incentivo de que precisei, mesmo quando escolhi seguir um caminho diferente do que imaginavam. Ao meu irmão, João Víctor, pelos momentos de partilha de raivas e alegrias com o nosso grandioso Flamengo, e pelas incontáveis vezes em que trouxe leveza e descontração à minha caminhada.

À minha orientadora, Carla Koike, pela orientação, pela paciência e, principalmente, pelos valiosos "puxões de orelha" que, mais do que advertências, foram lições que me ajudaram a ser mais responsável com meus compromissos e mais organizada ao longo desta jornada acadêmica.

À professora Maristela Holanda, uma das idealizadoras do projeto Meninas.comp, do qual faço parte desde o início da minha trajetória na Universidade de Brasília. Sou profundamente grata pelas inúmeras oportunidades que esse projeto me proporcionou, desde participar de eventos e ministrar oficinas até conhecer amigas incríveis, que hoje estão ao meu lado nos melhores e nos mais desafiadores momentos da faculdade e da vida.

Ao grupo de pesquisa Ereko, pelo voto de confiança no meu trabalho e pelo incentivo constante para que este projeto se concretizasse da melhor forma possível.

E, por fim, mas não menos importante, agradeço também à minha banda favorita, Twenty One Pilots, cujas músicas me deram forças nos momentos mais difíceis. Através de suas letras, aprendi a seguir em frente, a acreditar nos meus sonhos, a priorizar a saúde mental e a *"Stay alive"* para poder viver todas essas experiências que hoje guardo com tanto carinho.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

Este trabalho descreve o desenvolvimento do software Rupyly, um ambiente de programação em blocos que visa facilitar o ensino de robótica educacional através da integração entre *software* (baseado em Blockly, com bibliotecas modulares e geração de código em C) e *hardware* (plataforma Arduino via Arduino CLI). Criado para reduzir as barreiras de entrada na programação e no pensamento computacional, o Rupyly oferece uma interface intuitiva que permite a iniciantes criar e executar lógicas de programação para serem usados em modelos educacionais robóticos. Testes de usabilidade e implementações realizados pelos membros do grupo de pesquisa Ereko, e pessoas que nunca tiveram contato com programação em blocos e/ou arduino, mostram a sua eficácia em reduzir barreiras cognitivas e sintáticas, aumentar a motivação e o engajamento dos aprendizes, e consolidar a compreensão de conceitos de programação e robótica. O projeto demonstra ser uma ferramenta educacional valiosa, com potencial para futuras expansões e aprimoramentos.

Palavras-chave: Programação em Blocos, Blockly, Modelos educacionais, Robótica Educacional, Educação

Abstract

This paper describes the development of Rupyly, a block-based programming environment that aims to facilitate the teaching of educational robotics through the integration of software (based on Blockly, with modular libraries and code generation in C) and hardware (Arduino platform via Arduino CLI). Created to reduce the barriers to entry in programming and computational thinking, Rupyly offers an intuitive interface that allows beginners to create and execute programming logic for use in educational robotics models. Usability tests and implementations conducted by members of the Ereko research group, as well as by individuals who have never had contact with block-based programming and/or Arduino, demonstrate its effectiveness in reducing cognitive and syntactical barriers, increasing learner motivation and engagement, and consolidating understanding of programming and robotics concepts. The project proves to be a valuable educational tool, with potential for future expansion and improvements.

Keywords: Block Programming, Blockly, Educational Models, Educational Robotics, Education

Sumário

Lista de Figuras

Lista de Abreviaturas e Siglas

API Interface de Programação de Aplicações.

BBP Linguagem de programação baseadas em blocos.

CC Corente Contínua.

CDF *Cognitive Dimensions Framework*.

DSL Linguagens de domínio específico.

IDE *Integrated Development Environment*.

IoT Internet das Coisas.

IPC *Inter-Process Communication*.

LED *Light Emitting Diode*.

MIT *Massachusetts Institute of Technology*.

PWM modulação por largura de pulso.

RISC *Reduced Instruction Set Computer*.

TBP Linguagem de programação textual.

UI Interface do Usuário.

UX experiência do usuário.

VPL Linguagem de programação visual.

ZDP Zona de Desenvolvimento Proximal.

Capítulo 1

Introdução

A programação tem se mostrado um desafio significativo para estudantes universitários recém-ingressos, demandando uma elevada capacidade cognitiva. Isso ocorre tanto para a compreensão dos problemas propostos quanto para a escrita dos códigos necessários à sua resolução. Diversos estudos (1; 2; 3) apontam para essa complexidade inicial, que é demonstrada pela necessidade de adaptação e introdução gradual à lógica de programação (4; 5; 6). Essa complexidade pode gerar dificuldades significativas no processo de aprendizagem, afetando o engajamento e o desempenho dos estudantes.

Ao longo dos anos, diversos métodos vêm sendo desenvolvidos com o intuito de tornar o ensino de programação mais acessível e eficiente. Entre eles, destacam-se abordagens que utilizam a robótica como ferramentas de aprendizagem, conforme exemplificado por trabalhos em (7) e (8). Nessas metodologias, a robótica funciona como um contexto prático para a formulação e validação de hipóteses, promovendo a aplicação dos conceitos de programação em situações reais e motivadoras. Projetos nessa linha buscam integrar tecnologia e educação, estimulando o interesse dos alunos e facilitando a construção do conhecimento.

Nesse cenário, a programação visual surge como uma ferramenta poderosa para a democratização do ensino da computação. Linguagens visuais permitem a criação de programas por meio de elementos gráficos, como blocos ou fluxogramas, que representam comandos e podem ser conectados para formar códigos funcionais. Essa abordagem tem se mostrado particularmente eficaz no ensino de crianças e adolescentes, graças à sua interface intuitiva e acessível, que facilita o desenvolvimento, o entendimento e a assimilação dos conceitos básicos de programação (6; 9; 10; 11).

O Blockly (12), um conjunto de bibliotecas criado pela Google, exemplifica esse avanço, permitindo a construção de blocos customizados para o ensino de programação. Além de possibilitar a montagem visual de códigos, o Blockly realiza a tradução dos blocos para linguagens como JavaScript, Python, PHP, Lua e Dart, abrangendo desde declarações

de variáveis até operações matemáticas, o que amplia sua aplicabilidade em diversos contextos educacionais.

O grupo de pesquisa Ereko da Universidade de Brasília atua no desenvolvimento e utilização de modelos robóticos educacionais, incluindo robôs modulares e móveis com rodas, que são amplamente utilizados em oficinas e atividades de extensão. Durante essas experiências, identificou-se uma dificuldade recorrente dos participantes em relação à programação dos robôs, que limita o potencial pedagógico dessas iniciativas.

Diante desse desafio, o presente trabalho tem como objetivo o desenvolvimento de uma biblioteca de blocos exclusivos para cada modelo robótico educacional criado pelo grupo Ereko, visando facilitar a construção e a compreensão dos códigos tanto pelos integrantes do grupo quanto pelos participantes das oficinas. A biblioteca integra a tradução automática dos blocos para a linguagem C, utilizada na programação das placas Arduino que controlam os robôs, promovendo uma transição didática entre a programação visual e a textual.

O desenvolvimento seguiu uma metodologia sistemática, contemplando análise de requisitos, design da biblioteca, implementação, testes e documentação, garantindo a eficiência, flexibilidade e integração dos blocos.

Parte desta pesquisa foi apresentada no *International Symposium on Project Approaches in Engineering Education* (2024) (13) sendo premiada como melhor artigo de estudante e no XII Congresso Nacional de Engenharia Mecânica (2024) (14), o que reforça sua contribuição para a área de robótica educacional.

Espera-se que a solução final da proposta contribua ainda mais para o ensino de programação e robótica, tornando-o mais acessível, interativo e eficaz, além de fomentar a autonomia dos usuários no desenvolvimento de seus projetos educacionais.

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolvimento de uma biblioteca de blocos exclusivos para cada modelo educacional criado pelo grupo Ereko, afim de facilitar tanto a construção e programação dos modelos criados pelos integrantes, como facilitar o desenvolvimento, entendimento e compreensão dos participantes das oficinas realizados pelo grupo.

1.1.2 Objetivos específicos

- Construção de blocos específicos para programação dos modelos robóticos educacionais do grupo Ereko;

- Adicionar os blocos criados a uma biblioteca exclusiva para cada modelo educacional;
- Criação de uma interface web baseada na biblioteca Blockly;
- Integração da aplicação com a placa Arduino Uno;
- Criação de um E-book explicativo sobre a ferramenta.

1.2 Estrutura do Documento

Esta monografia está organizada em seis capítulos. O Capítulo 2 corresponde ao Referencial Teórico, dividido em duas seções: a Seção 2.1 aborda a linguagem de programação visual, enquanto a Seção 2.2 trata da robótica no contexto educacional. O Capítulo 3 apresenta os trabalhos correlatos, destacando projetos e tecnologias que serviram de base ou inspiração para o desenvolvimento deste estudo. O Capítulo 4 refere-se à metodologia adotada, contemplando a análise de requisitos, o design da biblioteca desenvolvida, os processos de implementação, bem como os testes realizados e a documentação elaborada. O Capítulo 5 expõe os principais resultados obtidos a partir da aplicação prática do projeto. Por fim, o Capítulo 6 apresenta as considerações finais, acompanhadas de sugestões para trabalhos futuros.

Capítulo 2

Referencial Teórico

Sendo o pilar principal para o desenvolvimento do projeto, a fundamentação teórica fornece conceitos que serão utilizadas no decorrrer da pesquisa. Linguagem de programação visual e Robótica educacional são os princípais tópicos tratados nesse capítulo.

2.1 Linguagens de programação visual

A programação, em sua essência, envolve a criação de instruções lógicas para que um computador execute tarefas específicas. Tradicionalmente, isso é feito por meio de Linguagem de programação textual (TBP), que exigem a escrita de código em uma sintaxe rigorosa. No entanto, o campo da computação testemunhou o surgimento e a crescente proeminência das Linguagem de programação visual (VPL), que oferecem uma alternativa mais intuitiva e acessível.

As VPLs representam uma categoria de linguagens que se distinguem das TBPs por permitirem a criação de programas por meio de elementos gráficos. Em vez de complexas linhas de código textual, os usuários manipulam blocos, ícones ou fluxogramas que simbolizam comandos, estruturas de controle e dados. Esses elementos visuais podem ser conectados e organizados de forma intuitiva, formando a lógica subjacente do programa. Essa abordagem "arrastar e soltar" minimiza a carga cognitiva associada à memorização de sintaxe e à depuração de erros de digitação, permitindo que os aprendizes se concentrem na lógica algorítmica e na resolução de problemas.

2.1.1 O Papel das VPLs no Ensino de Programação

O destaque das VPLs tem sido particularmente notável no ensino de programação para crianças e adolescentes. A interface amigável e acessível dessas linguagens facilita significativamente o desenvolvimento, o entendimento e a compreensão dos conceitos fundamentais

da programação. Ao remover as barreiras sintáticas, as VPLs tornam a experiência de aprendizado mais envolvente e menos frustrante para iniciantes. Exemplos proeminentes como Scratch (15) e LEGO Mindstorms EV3 (16) são amplamente utilizados em ambientes educacionais em todo o mundo, servindo como portas de entrada para o pensamento computacional.

2.1.2 Desafios e Eficácia das VPLs: Uma Análise Crítica

Apesar da crescente adoção das VPLs, a discussão sobre sua eficácia e as condições ideais para seu uso é contínua na literatura. Whitley (1997) (17), em seu artigo, já realizava uma revisão crítica sobre a programação visual, questionando a validade das supostas vantagens cognitivas atribuídas às VPLs e destacando a carência de estudos experimentais robustos para sustentar suas premissas pedagógicas e de usabilidade. Embora algumas pesquisas iniciais tenham demonstrado benefícios específicos como maior acurácia em tarefas com notações visuais (18), ou melhor desempenho em manipulação de matrizes (19), Whitley aponta que outras evidências revelaram limitações importantes, como menor eficiência em tarefas de lógica condicional (20) ou problemas de visibilidade em ambientes de planilha eletrônica. O autor argumenta que o sucesso das VPLs é condicionado ao contexto da tarefa, enfatizando o princípio do *match-mismatch*, que preconiza o alinhamento entre a natureza da notação e o tipo de problema a ser resolvido. Essa perspectiva ressalta a necessidade de mais estudos empíricos, fundamentados em teorias cognitivas, para guiar o desenvolvimento e a aplicação efetiva dessas linguagens, especialmente em ambientes de programação por usuários finais.

Em linha com essa discussão sobre a complexidade e a usabilidade, o artigo de Green e Petre (1996) (20) introduz uma estrutura de avaliação crucial para VPLs sob a ótica da interação humano-computador. O *Cognitive Dimensions Framework* (CDF) descreve aspectos estruturais de artefatos cognitivos, como linguagens de programação, por meio de dimensões como níveis de abstração, proximidade com o domínio do problema e resistência a mudanças. Ao aplicar o CDF a VPLs comerciais, os autores identificaram pontos fortes, como a proximidade com o domínio do problema e a redução de erros comuns, mas também desafios significativos. Entre as limitações, destaca-se a alta dificuldade em fazer mudanças, a dificuldade em usar layout para comunicar significado e problemas de visibilidade, particularmente em estruturas de controle. O CDF oferece uma linguagem comum para designers e avaliadores, permitindo identificar trade-offs entre as dimensões e reforçando a importância de melhorias em áreas como notação secundária e visibilidade.

2.1.3 A Transição das VPLs para Linguagens Textuais

Apesar dos benefícios iniciais no engajamento e na introdução aos conceitos de programação, a transição de VPLs (especialmente as baseadas em blocos) para linguagens de programação textuais como Python ou Java apresenta desafios consideráveis. O estudo de Strong, Bresnihan e Tangney (2025) (21) realiza uma revisão sistemática da literatura para mapear estratégias que facilitem essa transição. A pesquisa destaca que a dificuldade nessa passagem pode levar à desmotivação e ao abandono dos estudos em computação.

Os autores identificaram temas cruciais que permeiam as abordagens de transição: suporte técnico (incluindo tradução explícita entre blocos e código textual, auxílio na sintaxe e ferramentas de depuração) e motivação e engajamento (com pedagogias que utilizam *scaffolding*, tarefas relevantes para os interesses dos alunos e feedback visual rápido). A revisão aponta que ambientes de dual-modalidade, que exibem correspondências entre blocos e código textual (como BlockPy), são promissores. Contudo, a pesquisa ressalta a carência de estudos empíricos robustos e longitudinais que avaliem a eficácia dessas abordagens. A maioria dos estudos avaliou grupos pequenos e intervenções de curta duração, com pouca representação de contextos informais ou da diversidade de alunos (idade, gênero ou experiência prévia).

2.1.4 Abordagens Comparativas e Híbridas: VPLs e TBPs em Sinergia

A necessidade de otimizar a transição e aprimorar as metodologias de ensino de programação levou a estudos comparativos e ao desenvolvimento de abordagens híbridas. Recentemente, o estudo de Soomro (2023) (22) aprofundou a compreensão sobre a eficácia das VPLs em comparação com as TBPs no ensino de programação. Conduzido no Paquistão com 150 estudantes, incluindo alunos do ensino fundamental, primário e universitário, divididos em três grupos (VPL, TBP e híbrido), a pesquisa utilizou Scratch e LEGO Mindstorms EV3 para a abordagem visual, e Python e C++ para a textual. Os resultados são bastante reveladores: uma significativa maioria dos estudantes (68,8%) expressou preferência pelas VPLs, destacando a redução de erros de sintaxe (devido ao arrastar e soltar blocos), o aumento da motivação e interação, e a compreensão mais rápida de conceitos abstratos. O desempenho foi superior no grupo de VPL, com média de 90% nos testes conceituais, contrastando com a alta taxa de desistência e maior tempo de aprendizagem observados no grupo de TBP. As conclusões do estudo reforçam que as VPLs são mais eficazes para iniciantes, reduzindo a curva de aprendizagem e aumentando a confiança e o interesse. Contudo, as TBPs, embora mais desafiadoras, são consideradas necessárias para a transição para o ensino superior. A pesquisa sugere, portanto, que

uma abordagem híbrida, começando com VPLs e progressivamente migrando para TBPs, pode gerar os melhores resultados.

Essa perspectiva de transição é amplamente explorada no artigo de Yuhan e David (2021) (23). Eles fornecem um panorama abrangente do ecossistema das Linguagem de programação baseadas em blocos (BBP), analisando suas características estruturais e as estratégias para facilitar a transição para linguagens textuais. A partir de uma revisão sistemática de 101 ambientes BBP, os autores categorizam as abordagens de transição em quatro tipos principais: ambientes exclusivamente baseados em blocos (*blocks-only*), transição unidirecional, dual-modalidade e híbridos. A análise revela que, embora a maioria dos ambientes (52%) adote o modelo *blocks-only*, útil para a introdução de conceitos fundamentais, abordagens mais sofisticadas, como a dual-modalidade e os híbridos, que promovem um aprendizado contínuo e fluido, ainda são menos comuns. O estudo também aponta desafios como a limitada acessibilidade para pessoas com deficiência visual e a dificuldade de acesso público a alguns ambientes, fornecendo valiosos critérios para a seleção de ferramentas pedagógicas e apontando lacunas para futuras pesquisas em acessibilidade e sustentabilidade.

2.1.5 Inovações na Concepção e Aplicação de VPLs

A complexidade e a usabilidade das VPLs continuam a ser áreas de pesquisa ativa. Um trabalho relevante na área de ambientes de programação visual é a proposta de Kurihara et al. (2015) (24), que desenvolveram uma interface baseada em blocos para Linguagens de domínio específico (DSL). O objetivo foi tornar o ensino de programação mais acessível para iniciantes, onde os blocos são representados por frases em linguagem natural e automaticamente traduzidos em código textual válido. Esse design elimina erros de sintaxe e reduz as barreiras comuns enfrentadas por aprendizes. O sistema inclui funcionalidades como "dicas" (blocos reutilizáveis que encapsulam trechos de código) e "macros" (procedimentos definidos pelo usuário), que contribuem para a organização e abstração dos programas, facilitando a compreensão de estruturas mais complexas. Aplicado ao contexto da linguagem *Processing*, amplamente utilizada para arte digital e design interativo, o ambiente permite a visualização em tempo real dos efeitos das alterações nos blocos através do *live coding*. Entre os benefícios identificados, destacam-se a usabilidade intuitiva, o suporte à transição para linguagens textuais e a flexibilidade para adaptação a outras linguagens. Contudo, os autores também apontam limitações, como a dificuldade de lidar com programas extensos e a necessidade de melhorias na organização visual da interface. Essa proposta se insere no panorama mais amplo das iniciativas que buscam combinar acessibilidade e rigor computacional, dialogando com abordagens como Scratch e Blockly.

Em suma, as linguagens de programação visual desempenham um papel crucial na democratização do acesso à computação, especialmente para iniciantes. A literatura recente valida sua eficácia pedagógica, ao mesmo tempo, em que aponta para a necessidade de abordagens híbridas e o aprimoramento contínuo de seus ambientes. A interação entre a preferência dos alunos, o desempenho, as dificuldades enfrentadas e as características dos próprios ambientes visuais é essencial para o avanço das metodologias de ensino de programação e para a preparação de futuras gerações de programadores.

2.2 Robótica educacional

A Robótica Educacional desponta como uma metodologia pedagógica inovadora, cujas raízes podem ser rastreadas até o trabalho seminal de Seymour Papert. Em 1971, Papert, em colaboração com Cynthia Solomon, publicou o artigo (25), no qual propunham uma visão revolucionária: crianças deveriam utilizar computadores não apenas como dispositivos para consumir informações, mas como ferramentas ativas para expandir sua criatividade e potencial de aprendizado. Para os autores, o engajamento com a tecnologia deveria ser proativo, envolvendo a criação, a exploração e a aprendizagem por meio da programação de robôs e do desenvolvimento de uma ampla variedade de atividades educativas (25; 26). Essa perspectiva alinha-se diretamente com o Construcionismo, teoria pedagógica de Papert, que postula que o aprendizado é mais eficaz quando o aluno constrói um produto tangível e compartilhável, utilizando materiais concretos, como os robôs.

Nesse contexto, a Robótica Educacional é amplamente reconhecida como uma ferramenta pedagógica que fomenta a criação de ambientes de aprendizagem colaborativos e interdisciplinares. Zapata, Voles e Gusmán (2004) (27) defendem que essa abordagem promove a transversalidade curricular, integrando diferentes áreas do conhecimento na resolução de problemas propostos. Isso permite que os estudantes estabeleçam relações significativas entre conteúdos diversos, formulando representações próprias para compreender e solucionar os desafios em que estão envolvidos.

A relevância da Robótica Educacional para o desenvolvimento de competências multifacetadas é consistentemente destacada na literatura. Almeida (2015) (28) enfatiza que a Robótica Educacional contribui diretamente para o aprimoramento de habilidades essenciais, como a capacidade de pesquisa, o pensamento crítico, a resiliência para contornar dificuldades durante a resolução de problemas e o fortalecimento do raciocínio lógico.

Uma das características mais marcantes dessa abordagem é sua adequação a metodologias baseadas na resolução de problemas concretos. Ao propor desafios práticos e tangíveis, a robótica não apenas desperta o interesse e a motivação dos alunos, mas também promove o pensamento crítico e o raciocínio lógico de forma ativa e significativa.

Esse engajamento prático é particularmente relevante em contextos educacionais onde conteúdos escolares são percebidos como abstratos ou desmotivadores (29).

A aplicação da Robótica Educacional em diferentes níveis de ensino tem sido objeto de diversas investigações. Zilli (2004) (30), em sua dissertação de mestrado, investigou a aplicação da Robótica Educacional no Ensino Fundamental em escolas de Curitiba. A autora argumenta que a robótica contribui significativamente para o ensino ao promover a interdisciplinaridade e o estímulo ao raciocínio lógico e à autonomia dos estudantes. Fundamentando-se nas teorias construtivistas e construcionistas, Zilli propôs um modelo de implantação que contempla tanto a formação docente quanto a integração curricular e a adequação de infraestrutura, reconhecendo as barreiras ainda existentes para a ampliação do uso da robótica em escolas públicas.

Corroborando essa perspectiva, Silva (2009) (31), em sua tese de doutorado, propôs uma metodologia para o ensino de robótica pautada na teoria sócio-histórica de Vygotsky. A autora conceitua a robótica como um artefato cultural que atua como mediador da aprendizagem, impulsionando o desenvolvimento cognitivo por meio da interação social e da colaboração entre pares. Essa abordagem alinha-se a conceitos vygotksyanos como a Zona de Desenvolvimento Proximal (ZDP) e as funções psicológicas superiores, onde a aprendizagem ocorre de forma otimizada com o suporte de um mediador e a interação com o ambiente social. Utilizando o kit LEGO Mindstorms e o software educacional RoboEduc em oficinas com crianças de 8 a 10 anos, a pesquisa demonstrou que a robótica estimula a criatividade, o raciocínio lógico e habilidades sociais cruciais, como o planejamento e a resolução colaborativa de problemas.

Por sua natureza intrinsecamente multidisciplinar, a Robótica Educacional integra a construção de protótipos – geralmente por meio de kits de montagem específicos – com a aplicação de softwares de programação. Essa combinação única permite que os alunos desenvolvam projetos que colocam em prática, de forma concreta e contextualizada, os conceitos aprendidos em diversas disciplinas curriculares, consolidando o conhecimento de maneira significativa (26; 28).

Nesse sentido, a Robótica Educacional se configura como uma ferramenta pedagógica de valor inestimável para o processo de ensino-aprendizagem. Ela estimula o aluno a questionar, refletir criticamente, testar hipóteses e buscar soluções autônomas para desafios complexos. O contato com essa tecnologia não apenas aprimora o raciocínio computacional, mas também promove a interação do estudante com o mundo ao seu redor, possibilitando o desenvolvimento de competências essenciais para a formação de cidadãos críticos, criativos e preparados para os desafios e as demandas de uma sociedade contemporânea cada vez mais tecnológica.

Este capítulo abordou a evolução e o impacto das Linguagens de Programação Visual

(VPLs) no ensino, destacando sua capacidade de tornar a programação mais intuitiva e acessível, especialmente para iniciantes. Embora reconheça as vantagens pedagógicas das VPLs, a discussão também explora desafios como a transição para linguagens textuais e a necessidade de estudos mais robustos para avaliar sua eficácia em diferentes contextos. A análise de plataformas como *Scratch*, App Inventor, e projetos *open source* para Arduino, juntamente com o papel fundamental da Robótica Educacional como metodologia de ensino que promove o pensamento crítico e a interdisciplinaridade, estabelece a base para os Trabalhos Correlatos e as Tecnologias abordadas na próxima seção.

Capítulo 3

Trabalhos Correlatos

Este capítulo apresenta uma revisão dos principais trabalhos e tecnologias que serviram de base ou inspiração para o desenvolvimento deste projeto. São abordados modelos robóticos educacionais, como o Erekopapa e o Arabeko, que utilizam a robótica e a programação como ferramentas pedagógicas. Em seguida, são exploradas oficinas práticas que articulam teoria e prática no ensino de tecnologias, com ênfase na abordagem com Arduino. Por fim, discute-se o uso do Blockly como linguagem de programação visual e analisa-se uma seleção de softwares baseados em programação em blocos, contextualizando suas aplicações na educação.

3.1 Grupo Ereko

O grupo de pesquisa Ereko da Universidade de Brasília, fundado em 2009 pelas professoras Carla Koike e Dianne Viana, destaca-se por sua atuação pioneira na América Latina no campo da robótica modular reconfigurável. As linhas de pesquisa do grupo englobam robótica, educação e prototipagem, com o objetivo primordial de avançar no estudo e desenvolvimento dessa área inovadora (32).

3.2 Modelos Robóticos Educacionais

Essa seção apresenta os modelos robóticos educacionais utilizados como base para o desenvolvimento das bibliotecas. Inicialmente, é apresentado o Erekopapa (3.2.1) e a seguir o Arabeko (3.2.2).

3.2.1 Erekopapa

O Erekopapa (33) é um robô minimalista (Figura 3.1) que foi projetado por alunos de graduação do grupo de pesquisa Ereko para abordar conceitos básicos no âmbito do ensino da robótica. Ele é utilizado nas oficinas do grupo para abordar temas introdutórios, como programação básica com Arduino, montagem de circuitos com *protoboard*, além de oferecer uma introdução à modelagem e impressão 3D, método utilizado na fabricação de suas peças.

O robô consiste essencialmente de duas partes: o corpo e a cabeça (Figura 3.2). O corpo desempenha o papel de suporte para um Servomotor[1], ao qual a cabeça está conectada diretamente em seu eixo. A cabeça, por sua vez, abriga dois *Light Emitting Diode* (LED)[2], que simbolizam os olhos, e um buzzer[3] no lugar da boca. Assim, durante a oficina, os participantes podem controlar o ângulo de rotação da cabeça por meio do Servomotor, acionar os LEDs para piscar e controlar os sons do buzzer através de um botão ou de forma automática, dependendo da escolha de programação.

Figura 3.1: Projeto Final do Erekopapa completo e montado.

Figura 3.2: Projeto do Erekopapa

Este robô é amplamente utilizado nas oficinas do grupo de pesquisa para turmas iniciantes do ensino básico em robótica. Apesar da sua simplicidade, é comum que os

[1]É um tipo de atuador eletromecânico que permite um controle preciso da posição, velocidade e torque de um eixo.

[2]É um componente eletrônico que emite luz quando uma corrente elétrica o atravessa

[3]É um componente eletrônico que produz sons através da vibração de uma membrana quando uma corrente elétrica é aplicada

participantes ainda enfrentem dificuldades tanto na montagem dos circuitos quanto na programação do dispositivo.

3.2.2 Arabeko

O Arabeko (34; 35; 36; 37) é um robô móvel de acionamento diferencial, concebido para explorar conceitos de média e alta complexidade para iniciantes na área de robótica, em contraste com o Erekopapa apresentado anteriormente na seção 3.2.1. Importante ressaltar que o Arabeko encontra-se em processo de registro, o que impede a inclusão de quaisquer fotos ou desenhos do robô neste trabalho.

O corpo principal do robô é constituído por um chassi que oferece suporte aos componentes essenciais para seu funcionamento, incluindo Arduino, *protoboard*[4], ponte H[5], baterias, motores Corente Contínua (CC)[6] e rodas. Todas as peças são produzidas utilizando impressão 3D, e os participantes das oficinas são instruídos sobre o uso das ferramentas necessárias para reproduzir, modificar e aprimorar as peças existentes do robô. Além disso, eles são capazes de modelar peças complementares conforme as necessidades específicas do projeto. O robô tem um módulo *bluetooth* que se comunica com um *smartphone* e transfere as informações para o Arduino via porta serial.

Devido à sua natureza versátil, o robô permite a adição de outros componentes para explorar conceitos mais avançados. É possível integrar facilmente sensores ultrassônicos em posições estratégicas na parte frontal e traseira do robô. Isso permite que os participantes explorem a aquisição de dados e planejem, na programação, a resposta do robô ao detectar um obstáculo à sua frente. Uma adição bastante interessante é um braço robótico com garra, o que pode representar um desafio considerável para os participantes das oficinas. Eles precisam lidar não apenas com a programação do braço, mas também com o controle da movimentação do robô.

3.3 Oficinas

As oficinas conduzidas pelo grupo de pesquisa Ereko, nas quais são empregadas as ferramentas descritas neste trabalho, têm como público-alvo alunos e professores do ensino básico de escolas da rede pública, envolvendo os robôs mencionados na seção 3.2 (34; 35; 37; 36; 38).

[4]É uma placa com furos e conexões internas que permite a montagem temporária e a prototipagem de circuitos eletrônicos sem a necessidade de soldagem

[5]É um circuito eletrônico utilizado para controlar o sentido de rotação de motores DC, permitindo que girem tanto no sentido horário quanto no anti-horário

[6]É um tipo de máquina elétrica que converte energia elétrica em energia mecânica, utilizando corrente contínua (CC) como fonte de alimentação

Inicialmente, são oferecidas formações para os professores, abordando tanto as ferramentas quanto as metodologias pedagógicas necessárias para sua aplicação. Este treinamento visa capacitar os professores a replicar as oficinas com seus alunos e a adaptar os conteúdos para a melhor atender às necessidades específicas das instituições em que lecionam.

As oficinas são estruturadas em dois módulos, cada um composto por quatro sessões. O primeiro módulo, baseado no robô Erekopapa (Seção 3.2.1), abrange conceitos fundamentais de modelagem e impressão 3D, lógica de programação, e montagem de circuitos básicos em *protoboard* e Arduino. O segundo módulo, baseado no robô Arabeko (Seção 3.2.2), concentra-se em conceitos mais avançados, como o uso de sensores, motores DC, ponte H, e comunicação *Bluetooth*, além da criação de aplicativos na plataforma *AppInventor*.

3.4 Arduino

O Arduino foi desenvolvido em 2005 no *Interaction Design Institute*, localizado na cidade de Ivrea, Itália, por Massimo Banzi. O objetivo do professor Banzi era encontrar uma abordagem mais acessível para ensinar eletrônica e programação de computadores aos seus alunos de design de interação, uma vez que, na época, os sistemas de prototipagem existentes eram bastante custosos.

Com esse objetivo, e com a colaboração de seu aluno David Mellis, responsável pelo desenvolvimento da linguagem de programação do Arduino, decidiram criar uma placa própria. Esta nova placa, denominada Arduino, era disponibilizada em forma de kit, permitindo que os alunos realizassem seus próprios projetos. A popularidade do Arduino cresceu rapidamente à medida que o público reconheceu a facilidade de uso do sistema e seu baixo custo, além da possibilidade de utilizar componentes próprios, tornando-o uma excelente introdução à programação de microcontroladores (39; 40).

3.4.1 Estrutura do Arduino

O Arduino é uma plataforma tanto física quanto eletrônica baseada em um microcontrolador simples, que integra um microprocessador, memória e periféricos de entrada, como sensores, e de saída, como LEDs, motores, displays, entre outros. Tanto o hardware quanto o software do Arduino são de código aberto, o que significa que todo o código e os componentes necessários são disponibilizados pela equipe do Arduino e podem ser utilizados por qualquer pessoa e para qualquer finalidade. Assim, a placa pode ser montada manualmente ou adquirida já montada, enquanto o software de programação pode ser baixado gratuitamente (40; 41).

A plataforma Arduino, por meio dos sensores conectados aos seus terminais de entrada, é capaz de receber e interpretar variáveis e convertê-las em sinais elétricos. Esses sinais podem, então, controlar uma variedade de dispositivos de saída, como luzes, motores, buzzers ou atuadores diversos, conectados aos seus terminais de saída. Dessa forma, é possível implementar desde sistemas simples, como acender um LED com o apertar de um botão, até aplicações mais complexas, como robôs autônomos ou sistemas de monitoramento ambiental.

Um dos principais diferenciais do Arduino é sua facilidade de uso, que o torna especialmente popular em contextos educacionais e projetos de prototipagem rápida. A programação da placa é feita por meio da linguagem Arduino, baseada em C/C++, e realizada por meio da *Integrated Development Environment* (IDE) oficial, que permite escrever, compilar e carregar o código diretamente na placa via cabo USB. Também há suporte a outras formas de desenvolvimento, como via Arduino CLI (42), que permite compilar e enviar código pela linha de comando, facilitando a integração com outros softwares e automações.

Além disso, a comunidade ativa ao redor do Arduino contribui constantemente com bibliotecas, tutoriais e exemplos de código, o que fortalece ainda mais seu uso em diferentes áreas, como robótica, automação residencial, Internet das Coisas (IoT), arte interativa e projetos maker. Essa combinação de acessibilidade, versatilidade e suporte comunitário faz do Arduino uma das plataformas mais utilizadas em projetos de computação física e prototipagem eletrônica atualmente.

3.4.2 Hardware Arduino

Atualmente, existem inúmeros modelos de placas Arduino e, com o passar dos anos, as placas foram evoluindo e ficando cada vez mais acessíveis e sofisticadas. No entanto, todas as versões são baseadas em um microprocessador de 8 bits *Atmel AVR Reduced Instruction Set Computer* (RISC) (39).

Neste trabalho, mais especificamente, foi utilizado o Arduino UNO (Figura 3.3) que utiliza o ATmega328 com memória flash de 32KB e pode comutar automaticamente entre USB e corrente contínua (DC). Criada em 2010, esta placa é uma das mais recentes no mercado, ela possui 14 entradas/saídas digitais, dos quais 6 podem ser usados como saídas de modulação por largura de pulso (PWM), 6 entradas analógicas, um cristal oscilador de 16MHz, um cabeçalho ICSP e um botão reset (39; 40; 43).

O maior diferencial das placas antigas é a inclusão de um microcontrolador programado ATmega8U2 como um conversor USB-para-serial, substituindo o chipset FTDI obsoleto usado nas versões anteriores. O ATmega8U2 pode ser reprogramado para fazer o Arduino se parecer com outro dispositivo USB, tal como mouse, teclado ou *joystick*. Além disso,

ele possui uma tensão integrada de 3,3V mais confiável, o que ajuda na estabilidade de algumas proteções que causavam problemas no passado (39).

Figura 3.3: Arduino Uno com ATmega328

3.4.3 Ambiente de Desenvolvimento Integrado do Arduino

O IDE é uma aplicação de software que ajuda os programadores a desenvolver código de software de maneira eficiente, aumentando a produtividade do desenvolvedor, combinando recursos como edição, compilação, teste e empacotamento de software em uma aplicação fácil de usar.

O IDE do Arduino possui um editor de texto para escrita de códigos, uma área de mensagens, um console de texto, uma barra de ferramentas com botões para funções comuns e um conjunto de menus. Além disso, permite a conexão com o hardware Arduino para realizar o upload dos programas e possibilitar a comunicação entre o computador e a placa (Figura 3.4).

Os programas desenvolvidos no IDE do Arduino são chamados de *sketches* e salvos em extensão .ino. Essas *sketches* podem ser escritas no editor de texto da interface, que conta com funcionalidades como recortar, colocar, localizar e substituir trechos do código. A área de mensagens fornece um feedback ao salvar e exportar os projetos, além de exibir mensagens em caso de erros. Já o console mostra as saídas geradas pela IDE, incluindo mensagens de erro detalhadas e outras informações relevantes. Enquanto, a barra de ferramentas disponibiliza botões para verificar e enviar os programas à placa, bem como criar, abrir, salvar *sketches* e acessar o monitor serial (40; 44).

3.4.4 Arduino CLI

O Arduino CLI é uma ferramenta de código aberto, desenvolvida em Go (Golang), que permite a utilização de placas Arduino por meio da linha de comando. Com ela, é possível compilar, verificar e enviar (*sketches*) diretamente para as placas, além de gerenciar

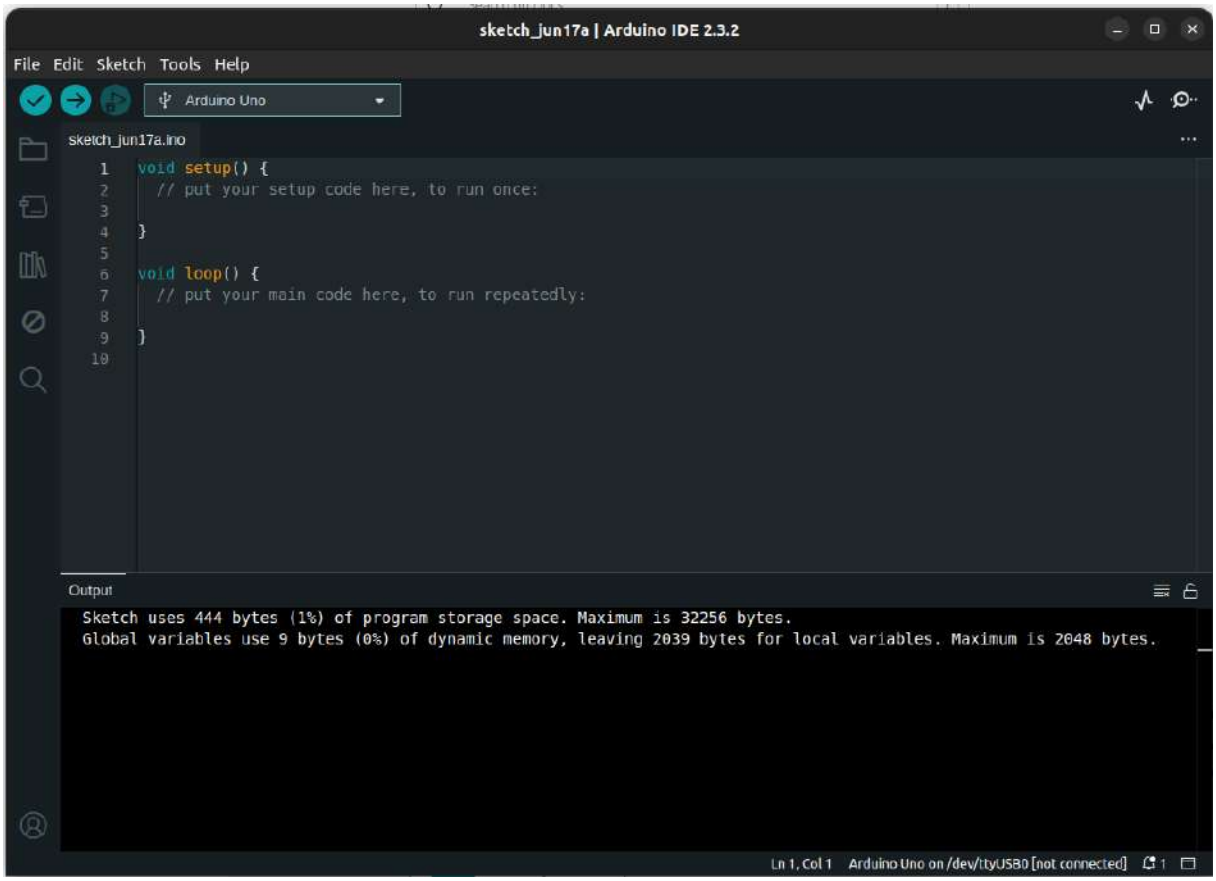


Figura 3.4: Ambiente de Desenvolvimento Integrado (IDE) do Arduino

bibliotecas, placas e outras ferramentas necessárias ao desenvolvimento de projetos com Arduino, sem a necessidade de uma interface gráfica. Neste contexto, destacam-se os três pilares fundamentais do Arduino CLI, sendo eles: interface de linha de comando, interface gRPC e incorporação (45).

O primeiro pilar, Interface de linha de comando, pode ser utilizado tanto por usuário humano como robô. Para humanos, a experiência do usuário (UX) é aspecto essencial, para isso é adotada uma estrutura baseada em subcomandos, que organiza as diversas operações disponíveis de maneira lógica. Essa abordagem permite ao usuário navegar pela interface de forma eficiente, com acesso à ajuda contextual específica. Para robôs, a ferramenta foi pensada na forma programática, tornando-a útil em *pipelines* de automação e sistemas de integração/entrega contínua (CI/CD). Uma das características importantes nesse contexto é a capacidade de operar sem a necessidade de um arquivo de configuração, isso porque todas as opções definidas no arquivo *arduino-cli.yaml* (46) podem ser fornecidas diretamente por meio de flags na linha de comando ou por meio de variáveis de ambiente (45).

O segundo pilar, interface gRPC, a ferramenta pode operar como um servidor gRPC

expondo um conjunto de procedimentos remotos (RPCs) que refletem os mesmos recursos disponíveis por meio da interface de linha de comando, permitindo que aplicações clientes se conectem e os utilizem remotamente. O gRPC é um framework de chamadas de procedimento remoto (RPC) de alto desempenho, que permite a comunicação eficiente entre aplicações cliente e servidor. Uma de suas principais vantagens é ser independente de linguagem: embora os exemplos sejam frequentemente escritos em Go (Golang), clientes podem ser implementados em diversas linguagens, como Python, JavaScript, entre outras, viabilizando cenários amplamente heterogêneos (45).

O terceiro pilar, incorporação, permite que desenvolvedores integrem diretamente os módulos do Arduino CLI em outras aplicações escritas na linguagem Go durante o processo de compilação. Por ser desenvolvido em Golang, o Arduino CLI possui uma arquitetura modular e bem estruturada, que facilita sua reutilização como biblioteca de software. A Interface de Programação de Aplicações (API) interna é exposta principalmente por meio do pacote "commands" (46), o qual pode ser importado por outros programas em Go, possibilitando a incorporação de uma instância completa e funcional da Arduino CLI dentro dessas aplicações (45).

3.5 Blockly

O *Blockly* (12) é um conjunto de bibliotecas desenvolvido pelo Google com o propósito de criar editores que representem conceitos de programação por meio de blocos visuais. Inspirado no *Scratch*, o *Blockly* foi projetado inicialmente com foco educacional, especialmente para o ensino de programação a crianças, por meio da construção de jogos e animações interativas. Como biblioteca totalmente *open-source*, tem sido amplamente adotada em diferentes contextos, estendendo sua aplicação para além do ambiente educacional.

A biblioteca oferece uma ampla variedade de blocos que representam instruções típicas de linguagens de programação, abrangendo desde declarações de variáveis até operações matemáticas, lógicas e relacionais. Também disponibiliza estruturas de controle como condicionais (*if/else*) e laços de repetição (*loops*), possibilitando a construção de programas completos de maneira visual e intuitiva. Esses blocos, ao serem conectados, formam sequências lógicas que podem ser executadas no ambiente do *Blockly* ou convertidas automaticamente para linguagens como *JavaScript*, *Python*, *PHP*, *Lua* e *Dart*.

Além de suas funcionalidades básicas, o *Blockly* disponibiliza ferramentas auxiliares que facilitam a personalização do ambiente. Dentre elas, destaca-se a *Blockly Factory*, uma ferramenta online criada pelo próprio Google que permite a criação e configuração de blocos personalizados. Por meio dessa interface, desenvolvedores podem definir a sintaxe, a aparência e o comportamento dos blocos, adaptando-os às necessidades específicas

de seus projetos (Figura 3.5). Essa flexibilidade torna o *Blockly* especialmente útil em iniciativas voltadas ao ensino de programação e ao desenvolvimento de soluções educacionais interativas, promovendo uma abordagem mais acessível e visual para a resolução de problemas computacionais (12).

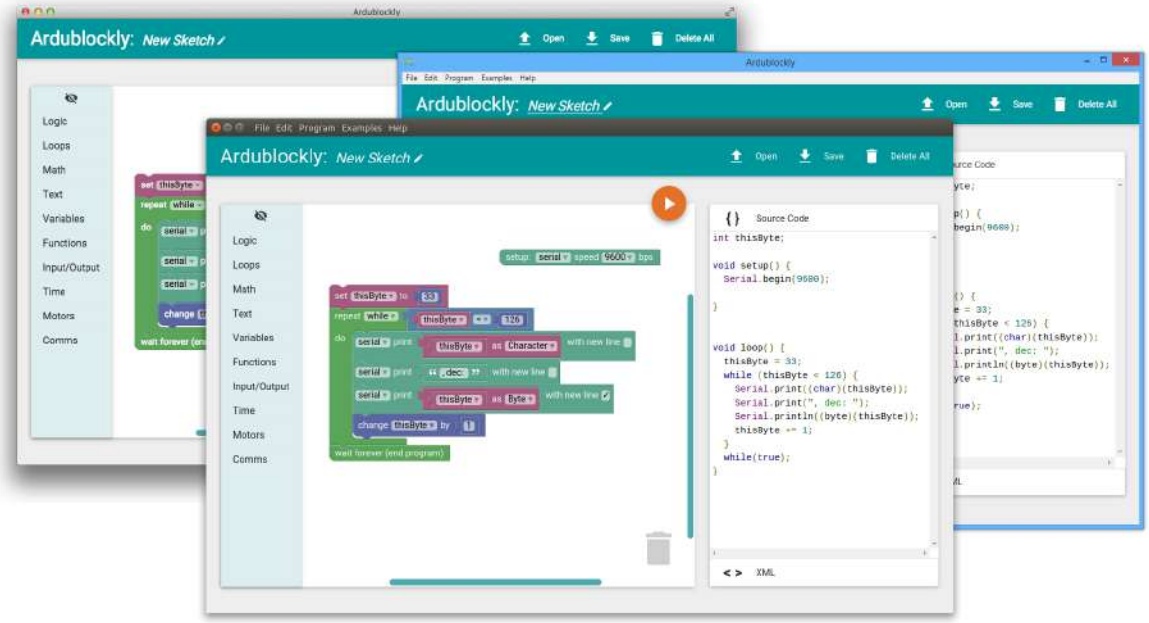


Figura 3.5: Exemplo de aplicação do Blockly no software Ardublockly.

3.6 Softwares de programação em blocos

Diversas plataformas de programação em blocos têm sido utilizadas como ferramentas educacionais para facilitar o ensino de programação, especialmente para iniciantes e público jovem. Entre os softwares mais reconhecidos e citados pelos participantes deste estudo, destacam-se:

Scratch (15) (Figura 3.6) é uma linguagem de programação visual desenvolvida pelo *Massachusetts Institute of Technology* (MIT) Media Lab, amplamente utilizada para o ensino de conceitos básicos de programação por meio da construção de histórias interativas, jogos e animações. Sua interface intuitiva e colorida permite a montagem de scripts através de blocos encaixáveis, favorecendo a aprendizagem lúdica e criativa.

Blockly Games (47) é uma série de jogos educacionais que utilizam a biblioteca Blockly para introduzir os fundamentos da programação a crianças e iniciantes. Através de desafios progressivos, os usuários aprendem conceitos como comandos, *loops* e condicionais, enquanto interagem com uma interface gráfica amigável.

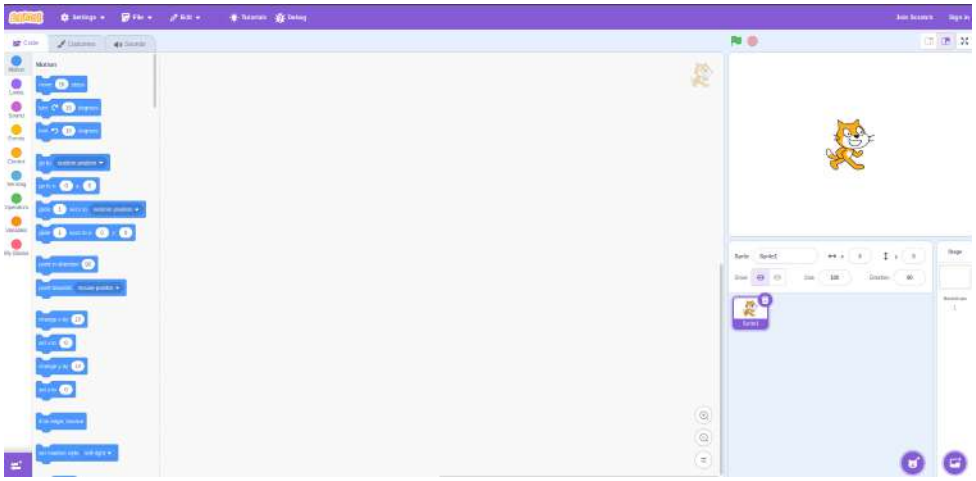


Figura 3.6: Página Incial do Scratch

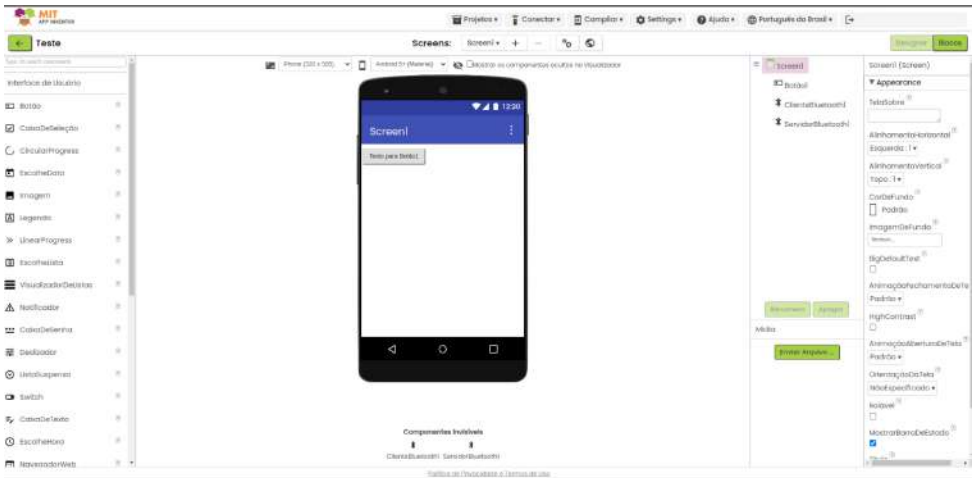


Figura 3.7: Página Inicial do App inventor

Tinkercad (48) é uma ferramenta online da Autodesk que, além de modelagem 3D, inclui um ambiente de simulação de circuitos eletrônicos com programação em blocos para microcontroladores, como o Arduino. Essa funcionalidade possibilita que os usuários aprendam programação e eletrônica de forma integrada e prática.

App Inventor (49) (Figura 3.7) é uma plataforma desenvolvida pelo MIT que permite a criação de aplicativos para dispositivos móveis utilizando uma interface de programação visual baseada em blocos. Destinada a iniciantes, oferece recursos que simplificam o desenvolvimento de aplicativos Android, tornando o processo acessível sem necessidade de conhecimentos avançados em linguagens textuais.

mBlock (50) é uma plataforma educacional baseada em Scratch que suporta a programação de robôs, dispositivos *IoT* e placas Arduino. O ambiente combina programação

em blocos e código textual, permitindo uma transição gradual para linguagens mais avançadas, e é amplamente utilizado em contextos educacionais para ensino de robótica.

Além das plataformas comerciais e educacionais já mencionadas, existem também projetos de código aberto que buscam facilitar a programação em blocos para o Arduino, ampliando o acesso e possibilitando maior personalização e integração.

O **ArduBlockly** (51) é um projeto que oferece uma interface gráfica baseada em blocos para programação de placas Arduino. Ele integra a geração automática de código C++ a partir dos blocos visuais, permitindo que iniciantes construam programas de maneira intuitiva.

O **BlocklyDuino** (52) é outra iniciativa open source que utiliza a biblioteca Blockly para criar um ambiente de programação visual para Arduino. Com uma interface amigável, o BlocklyDuino permite que usuários, principalmente iniciantes, montem programas com blocos que são automaticamente convertidos em código Arduino padrão.

Por fim, o **Blockly at rduino** (53) é um projeto focado em oferecer uma solução modular para programação visual em Arduino, com ênfase na customização dos blocos e integração com hardware específico.

Essas plataformas de programação em blocos podem ser categorizadas com base em suas funcionalidades e público-alvo. Enquanto *Scratch* e *Blockly* oferecem uma introdução lúdica à lógica de programação para iniciantes, sem foco em hardware, o App Inventor se especializa na criação de aplicativos móveis. Já o *Tinkercad* e o mBlock integram programação com simulação ou controle de hardware de robótica e eletrônica, permitindo uma transição gradual para linguagens textuais mais avançadas.

Paralelamente, projetos de código aberto como ArduBlockly, BlocklyDuino e Blockly at rduino se dedicam especificamente à programação de placas Arduino via blocos. Essas iniciativas são cruciais por sua natureza *open source* e pela capacidade de converter diretamente blocos em código Arduino, facilitando a interação com *hardware* real para iniciantes. Além disso, exemplificam o crescente interesse e esforço da comunidade em criar soluções acessíveis para o ensino de programação e robótica, alinhando-se com os objetivos deste trabalho, que busca desenvolver uma biblioteca personalizada para modelos educacionais do grupo Ereko.

Este capítulo detalhou os modelos educacionais Erekopapa e Arabeko, robôs desenvolvidos para o ensino de robótica, desde conceitos básicos, como programação e eletrônica com Arduino e impressão 3D (Erekopapa), até tópicos mais avançados, como sensores e comunicação Bluetooth (Arabeko). Aborda também a estrutura das oficinas do grupo Ereko e a relevância do Arduino como plataforma de prototipagem de hardware e seu ambiente de desenvolvimento, incluindo a IDE e o Arduino CLI. Ao discutir sobre VPLs, como *Scratch* e *Blockly*, e projetos focados em Arduino, este capítulo estabelece o contexto

para a necessidade e o desenvolvimento da biblioteca personalizada, que será detalhada no próximo capítulo sobre o processo de desenvolvimento, incluindo análise de requisitos e etapas de implementação.

Capítulo 4

Metodologia

Neste capítulo são descritos os métodos e etapas adotados para o desenvolvimento do software proposto, desde a criação do e-book de primeiros passos, levantamento de requisitos até a fase de testes e documentação. O processo, estruturado de forma incremental, traz foco na clareza do design, modularidade do código e na validação prática da funcionalidade em ambiente real.

O nome Rupyly é uma junção que reflete o design central do software. “Rupy” deriva do dialeto Guarani, especificamente do Guarani Paraguaio (*Avañe*), onde funciona como uma posposição significando “através de”, “por meio de”,“de acordo com”, ou “para”. Isso transmite a ideia do software como uma ferramenta ou meio para alcançar algo. O componente “ly” vem de Blockly, o editor de programação visual fundamental sobre o qual o software é construído. Assim, Rupyly encapsula sua essência: uma ferramenta que capacita os usuários por meio da programação intuitiva baseada em blocos facilitada pelo Blockly (54).

4.1 E-book - Rupyly: Programando em Blocos

O E-book Rupyly: Programando em Blocos (Figura 4.1) foi desenvolvido com o objetivo de servir como material introdutório para novos usuários da plataforma Rupyly. Trata-se de um guia essencial para aqueles que estão tendo o primeiro contato com o ambiente de programação em blocos voltado ao uso com placas Arduino.

O conteúdo do E-book está organizado de forma didática e progressiva, permitindo ao leitor compreender desde os elementos básicos da interface até a criação de programas completos utilizando os blocos personalizados disponíveis. A seguir, descrevem-se os principais tópicos abordados:

- Sobre a autora: Apresenta informações sobre a idealizadora da plataforma Rupyly, contextualizando sua trajetória e motivações para o desenvolvimento do projeto;

Figura 4.1: Capa do E-book - Rupyly: Programando em Blocos

- Objetivo do material: Explica o propósito do E-book, que é fornecer uma introdução clara e acessível à utilização da plataforma, especialmente para iniciantes em programação e robótica educacional;
- Interface da plataforma: Descreve os principais elementos visuais da interface do Rupyly, incluindo a área de montagem de blocos, os menus superiores, os botões de controle e as opções de exportação;
- Bibliotecas implementadas: Detalha as bibliotecas específicas que foram criadas para suportar o funcionamento dos blocos personalizados da plataforma;
- Blocos personalizados: Apresenta cada um dos blocos disponíveis na plataforma, explicando suas funcionalidades e parâmetros;
- Erekopapa: Explica o funcionamento do Erekopapa, incluindo um exemplo prático de como construir um programa utilizando essa biblioteca na plataforma. O exemplo é acompanhado de imagens e orientações passo a passo;
- Arabeko: Fornece uma explicação semelhante à anterior, focada no componente Arabeko. No caso, são apresentadas as duas formas construídas: autônomo, onde os movimentos já são pré-determinados, e original, onde o usuário pode programar da forma que quiser. Também inclui um exemplo completo de montagem de programa utilizando blocos específicos;

- Download e envio para o Arduino: Ensina como realizar o download do código gerado a partir da programação em blocos, além de demonstrar o procedimento para envio direto do código para a placa Arduino conectada ao computador, utilizando a funcionalidade integrada da plataforma.

Esse E-book atua como uma ponte entre o usuário e o software, promovendo a autonomia na construção de projetos educacionais com base em programação visual e robótica.

4.2 Análise dos requisitos

A análise de requisitos tem como objetivo identificar e descrever as funcionalidades e características esperadas da biblioteca. Esta etapa foi essencial para orientar o design e a implementação do projeto, garantindo que a solução atenda às necessidades identificadas. Ela é separada em dois tipos funcionais e não funcionais.

4.2.1 Requisitos Funcionais

Os requisitos funcionais definem as funcionalidades que a biblioteca deve oferecer ao usuário. A seguir, são apresentadas as principais funções identificadas durante o desenvolvimento do software:

- O software deve permitir que o usuário faça o download do E-book sobre os primeiros passos;
- O software deve possibilitar que o usuário feche uma ou ambas as áreas de trabalho e possa restaurá-las posteriormente;
- O software deve fornecer acesso às bibliotecas criadas e, consequentemente, aos blocos pertencentes a cada biblioteca;
- O software deve permitir que o usuário arraste blocos para a área de trabalho designada e ofereça opções para duplicar, adicionar comentários, colapsar, deletar ou obter explicações sobre cada bloco;
- O software deve possibilitar o *zoom in* e *zoom out* na área destinada aos blocos;
- O software deve traduzir os blocos para a linguagem C na área respectiva, facilitando o entendimento do usuário;
- O software deve permitir o download do código gerado pela combinação dos blocos, no formato de arquivo com extensão .ino;

- O software deve possibilitar a conexão direta do computador com o Arduino, permitindo o envio e a compilação do código gerado a partir da junção dos blocos.

4.2.2 Requisitos Não Funcionais

Os requisitos não funcionais definem as restrições e características de qualidade esperadas para o funcionamento da biblioteca, tais como desempenho, compatibilidade e usabilidade:

- O software deve ser compatível com os sistemas operacionais Windows, Linux e macOS;
- O código-fonte deve ser organizado em módulos, facilitando a manutenção e a reutilização, e deve estar devidamente documentado;
- A interface deve ser intuitiva e de fácil utilização, mesmo para usuários iniciantes em computação e robótica;
- O software deve permitir futuras integrações com novos tipos de blocos, garantindo escalabilidade;
- O código-fonte deve estar disponibilizado sob licença aberta, permitindo seu uso, modificação e redistribuição;
- O software deve fornecer feedback visual claro ao usuário, indicando se o envio dos comandos foi concluído com sucesso ou se ocorreram erros, possibilitando a correção;
- O tempo de execução de comandos simples não deve exceder 2 segundos, garantindo boa responsividade;
- O software deve ser compatível com múltiplas versões de placas Arduino e diferentes ambientes de desenvolvimento (IDEs);
- O software deve permitir a expansão para inclusão de novos robôs.

4.3 Design da biblioteca

O projeto do software foi criado com o objetivo de desenvolver uma ferramenta acessível, modular e extensível, visando otimizar o processo de ensino de programação e robótica com a plataforma Arduino, especialmente para usuários iniciantes. Em comparação com os trabalhos apresentados na Seção 3.6, o Rupyly propõe a criação de uma biblioteca dedicada, que reúne todos os blocos necessários para um determinado modelo robótico

educacional. Essa abordagem visa oferecer uma biblioteca exclusiva, estruturada e otimizada para o uso em oficinas de ensino, promovendo maior organização e facilidade no processo de aprendizagem.

Para alcançar essa finalidade, a solução foi estruturada com base em uma arquitetura modular, composta por diversos componentes independentes que interagem de forma sinérgica. Essa interconexão garante uma experiência de desenvolvimento integrada, possibilitando aos usuários a criação de programas por meio de uma interface intuitiva de programação por blocos, a geração automática do código correspondente e o envio direto para as placas Arduino. Tal abordagem minimiza as barreiras iniciais e facilita a compreensão dos conceitos fundamentais, promovendo um ambiente de aprendizado coeso e eficiente.

4.3.1 Visão Geral da Arquitetura

A Figura 4.2 apresenta uma visão geral da arquitetura do software, dividida entre os módulos *frontend* e *backend*. O *frontend*, representado no lado esquerdo da figura, compreende os elementos com os quais o usuário interage diretamente. Entre eles estão a área para montagem dos blocos com blocos personalizados de acordo com cada biblioteca, a interface gráfica construída com HTML, CSS e JavaScript, além da lógica de geração automática do código Arduino a partir da montagem dos blocos.

O *backend*, por sua vez, é o responsável pelo processamento das funcionalidades do sistema. Ele é implementado em Node.js [1] e utiliza o Electron [2] para empacotar a aplicação como um software desktop. A principal funcionalidade do backend é compilar e enviar o código gerado para a placa Arduino, utilizando a ferramenta Arduino CLI. A comunicação entre o *frontend* e o *backend* ocorre por meio de comandos locais e integração direta com o sistema operacional do usuário, dispensando o uso de servidores externos.

Essa arquitetura modular favorece a manutenção, extensibilidade e autonomia da aplicação, permitindo que o usuário execute todo o fluxo, desde a criação do programa até sua execução na placa, de forma local e independente de conexão com a internet.

4.3.2 Componentes do Sistema

- Interface Rupyly (Frontend): Permite que o usuário crie programas arrastando blocos visuais. Essa interface foi construída com base na biblioteca Blockly do Google e customizada de forma a adicionar uma nova área onde é possível visualizar a geração do código em C, resultando em um código compatível com a IDE do Arduino.

[1] https://nodejs.org/pt
[2] https://www.electronjs.org/pt/

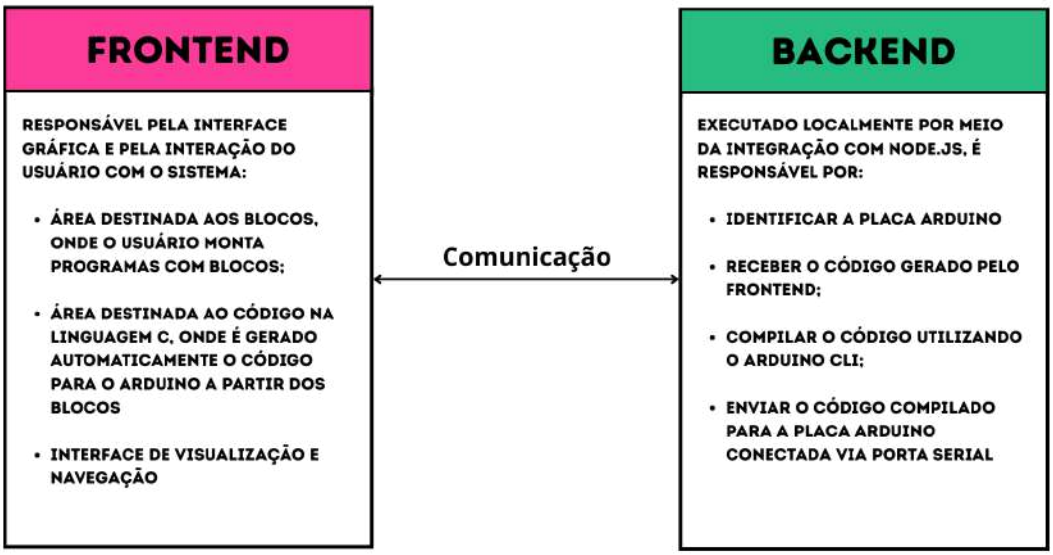


Figura 4.2: Arquitetura Geral do Software

- Servidor Node.js (Backend): Controla o fluxo entre a interface e a compilação do código. Recebe o código gerado, salva-o temporariamente em um arquivo ".ino", e executa comandos do "arduino-cli" para compilar e enviar o programa à placa.
- Arduino CLI: Ferramenta oficial da Arduino que permite compilar e enviar códigos para placas Arduino diretamente via linha de comando, sem depender da IDE tradicional. O Node.js executa essa ferramenta automaticamente.
- Feedback ao Usuário: A interface exibe mensagens sobre erros de compilação, sucesso no envio do código e outras notificações úteis, permitindo que o usuário corrija eventuais problemas ou valide o funcionamento do sistema.

4.3.3 Fluxo de Funcionamento

O processo de uso segue os seguintes passos (Figura 4.3):

1. O usuário realiza o download do e-book contendo os primeiros passos para a utilização do software e a introdução à programação;
2. A partir das instruções contidas no e-book, o usuário constrói um programa utilizando os blocos visuais na área de trabalho designada;
3. O programa desenvolvido visualmente é automaticamente convertido para código-fonte na linguagem C;

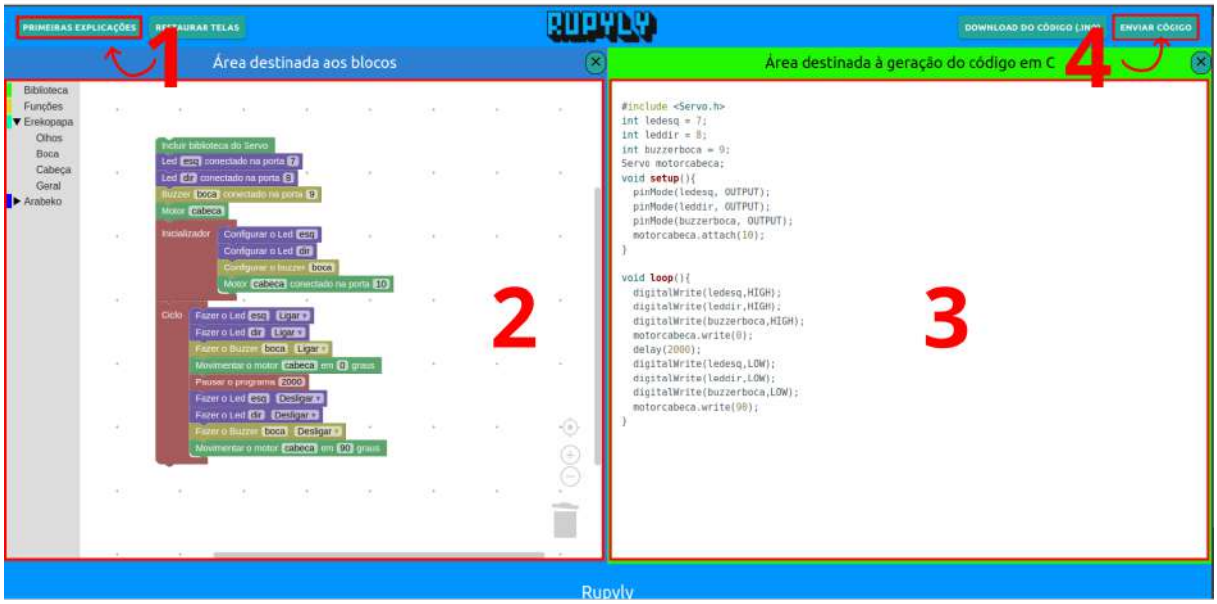


Figura 4.3: Fluxo de Funcionamento

4. O código C gerado é preparado para o Arduino e salvo como um arquivo *".ino"* em um diretório temporário;

5. O ambiente Node.js do software executa a ferramenta *"arduino-cli"* para compilar o código e realizar o upload para a placa Arduino conectada;

6. A saída gerada pelo processo de compilação e upload é capturada e exibida diretamente na interface do software, fornecendo feedback imediato ao usuário.

O código da Figura 4.3 é um exemplo da biblioteca do Erekopapa. Primeiramente, são declarados e inicializados os leds (olhos) e o buzzer (boca) como saídas, e o servo motor (cabeça). No ciclo (*loop*), os olhos e a boca ligam enquanto a cabeça está em 0º graus por 2 segundos. Em seguida, eles desligam e a cabeça se move para 90º graus, repetindo o processo.

4.3.4 Justificativas de Design

A escolha por uma arquitetura modular permite maior flexibilidade e manutenção do sistema. A divisão entre *frontend* e *backend* possibilita que futuras melhorias, como suporte a múltiplos usuários ou novos tipos de hardware, sejam facilmente implementadas. Já o uso de Node.js como backend facilita a execução de comandos do sistema operacional e o gerenciamento de arquivos, enquanto o "arduino-cli" fornece uma maneira robusta e automatizável de compilar e enviar código Arduino.

A interface em blocos, por sua vez, torna o aprendizado mais intuitivo, reduzindo a complexidade sintática da programação textual e incentivando a exploração e criatividade dos usuários.

Além disso, a motivação para o desenvolvimento de bibliotecas separadas para o Erekopapa e Arabeko surge a partir da análise das diferenças estruturais, funcionais e nível de dificuldade entre os dois robôs utilizados nas oficinas. Enquanto, o Erekopapa apresenta uma estrutura mais simples e lúdica, o Arabeko possui uma arquitetura mais complexa. Essas diferenças representam desafios distintos tanto na montagem quanto na programação dos robôs, exigindo abordagens específicas para cada modelo.

4.4 Implementação

A implementação da biblioteca seguiu o design proposto, utilizando ferramentas e linguagens adequadas ao contexto. Esta etapa envolveu a codificação dos módulos, integração entre componentes e tratamento de possíveis limitações técnicas. A seguir, detalham-se os principais aspectos do processo de desenvolvimento.

4.4.1 Ambiente de Desenvolvimento

O desenvolvimento foi realizado em ambiente Linux, utilizando o editor Visual Studio Code [3] como principal IDE. As linguagens utilizadas foram JavaScript e HTML/CSS no *frontend*, enquanto o *backend* foi construído com Node.js. A integração com placas Arduino foi viabilizada por meio do Arduino CLI, uma interface de linha de comando que permite compilar e enviar códigos diretamente para a placa. O framework Electron foi utilizado para empacotar a aplicação como um software desktop, tornando-a multiplataforma e independente de navegador.

4.4.2 Implementação do Frontend

O *frontend* foi estruturado utilizando a biblioteca Blockly, que permite a construção de código por meio de blocos visuais. Foram desenvolvidos blocos personalizados para representar componentes eletrônicos como LEDs, buzzer, motor, entre outros. Esses blocos foram associados a comandos específicos como “ligar” e “desligar”. Por exemplo, o componente LED pode ser ligado ou desligado conforme a lógica criada pelo usuário (Figura 4.4), assim como o buzzer pode emitir som ou não. Cada bloco contém uma definição visual (descrição e encaixe) e uma função geradora de código em linguagem C/C++ compatível

[3] https://code.visualstudio.com/

com a plataforma Arduino, permitindo que o programa completo seja montado dinamicamente conforme o usuário organiza os blocos na interface.

Figura 4.4: Exemplo ilustrativo de programação em blocos para iniciantes, demonstrando o controle de um LED (pisca-pisca) e os conceitos de inicialização e ciclo de execução.

A interface gráfica (Figura 4.5 e Figura 4.6) foi construída com HTML e estilizada com CSS e *Tailwind*[4], priorizando clareza e acessibilidade. Também foram adicionados botões funcionais para gerar, visualizar e enviar o código, além de mensagens de status que informam o progresso da compilação e upload.

A primeira versão da interface (Figura 4.5) foi desenvolvida com um layout simplificado e direto, priorizando sua utilização em fases de teste iniciais. O objetivo principal foi criar uma base funcional que permitisse a validação de conceitos e a coleta de *feedback* para futuras iterações.

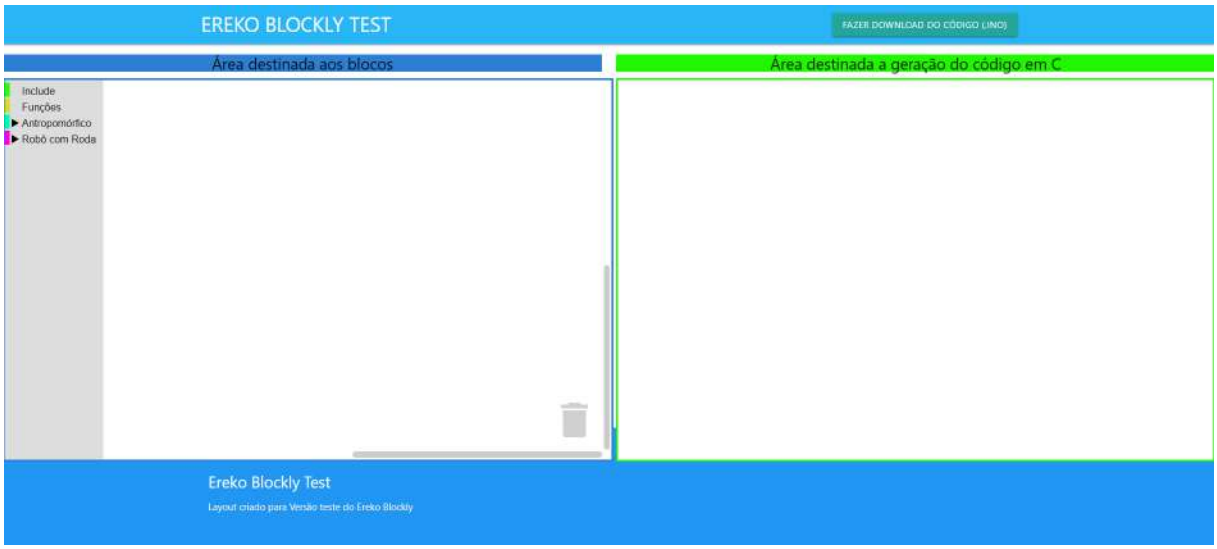


Figura 4.5: 1ª Versão - Interface do Rupyly

A segunda versão da interface (Figura 4.6) foi projetada para otimizar a experiência do usuário (UX). Esta atualização introduz novos botões para acesso rápido ao e-book,

[4]https://tailwindcss.com/

restauração de telas e envio direto para o Arduino. As áreas de blocos de programação e código em C foram aprimoradas com funcionalidades de *zoom in/out* e a opção de serem fechadas, oferecendo maior flexibilidade e organização.

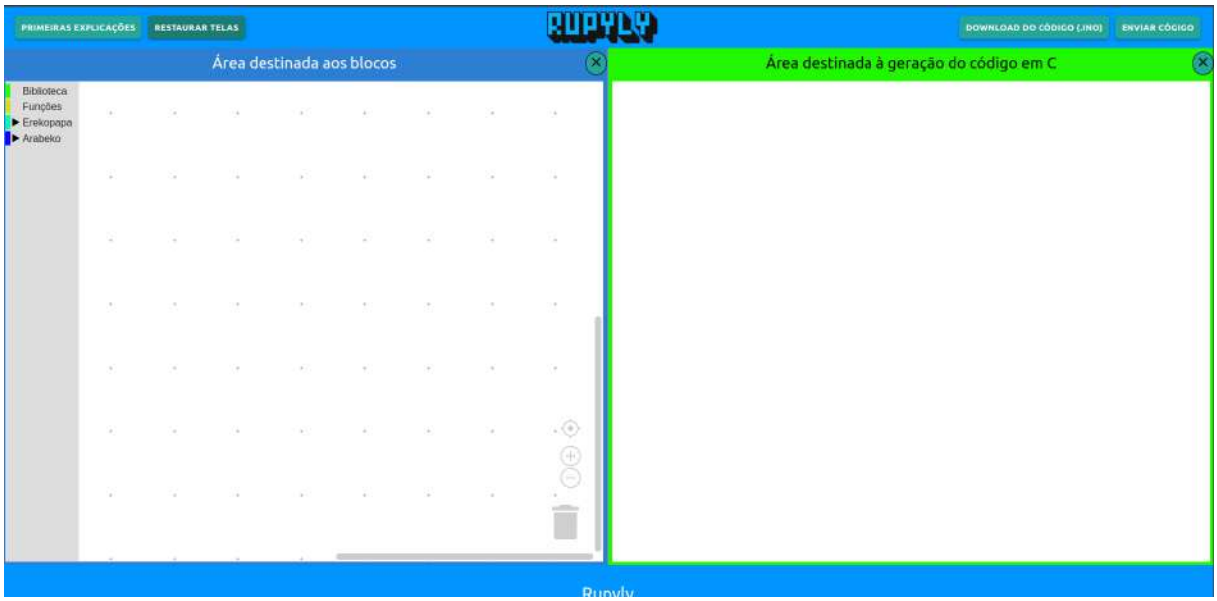


Figura 4.6: 2ª Versão - Interface do Rupyly

4.4.3 Implementação do Backend

O *backend* foi implementado com Node.js e é responsável pela comunicação com o sistema operacional e a execução de comandos da Arduino CLI. Após a geração do código Arduino no *frontend*, o *backend* recebe esse código e executa os seguintes passos:

- Identificação da Placa Arduino (Listing 4.1): O software detecta e identifica automaticamente a placa Arduino conectada ao computador;
- Criação Dinâmica do Arquivo ".ino": Um arquivo ".ino" é gerado dinamicamente, contendo o código-fonte C traduzido a partir dos blocos visuais;
- Compilação do Código: O comando "arduino-cli compile" é executado para compilar o código, transformando-o em um formato executável pela placa Arduino;
- Upload para a Placa: Após a compilação, o comando "arduino-cli upload" é utilizado para transferir o código compilado para a memória da placa Arduino.

```
1 function detectarPorta(callback) {
2   exec("arduino-cli board list", (err, stdout, stderr) => {
3     if (err) {
4       console.error("Erro ao listar placas:", stderr);
```

```
      return;
    }
    const linhas = stdout.split("\n").slice(1);
    for (const linha of linhas) {
      const colunas = linha.trim().split(/\s+/);
      if (colunas.length > 0 && colunas[0].startsWith("/dev") || colunas
    [0].startsWith("COM")) {
        porta = colunas[0];
        console.log(`Porta detectada automaticamente: ${porta}`);
        if (typeof callback === "function") {
          callback();
        }
        return;
      }
    }
    console.error("Nenhuma placa encontrada. Conecte uma placa e tente
    novamente.");
    setTimeout(() => detectarPorta(callback), 5000);
  });
}
```

Listing 4.1: Função JavaScript para detecção automática da porta serial, utilizando o comando "arduino-cli" board list para identificar o dispositivo conectado.

O Electron integra a interface gráfica com o *backend*, possibilitando a criação de uma aplicação *desktop* completa. A comunicação entre os dois módulos é feita através de *Inter-Process Communication* (IPC), garantindo que as ações do usuário na interface reflitam diretamente no processo de envio do código para o Arduino.

4.4.4 Implementação dos blocos

Para permitir que o usuário programe utilizando blocos específicos voltados para o uso com Arduino, foram criados blocos personalizados com funcionalidades compatíveis com sensores, atuadores e elementos de entrada e saída. Esses blocos foram desenvolvidos com o auxílio da ferramenta **Blockly Factory**, disponibilizada pela própria equipe do Google Blockly (55).

Essa ferramenta permite a criação visual de blocos, sem a necessidade de escrever manualmente o código em JavaScript. Através da interface do Blockly Factory, é possível configurar:

- O formato visual do bloco;
- As entradas e saídas (parâmetros, menus, valores);

- O comportamento semântico (ou seja, o código que será gerado, geralmente em C/C++ para uso com Arduino);
- A forma como o bloco se conecta a outros blocos na estrutura da linguagem visual.

Uma vez definido o bloco na interface do Blockly Factory, a ferramenta gera automaticamente o código XML do bloco e as funções correspondentes para a geração de código, que posteriormente foram integradas à interface principal utilizando o sistema do Blockly.

A Figura 4.7 apresenta um exemplo da interface da Blockly Factory durante a criação de um bloco personalizado para acionar um LED.

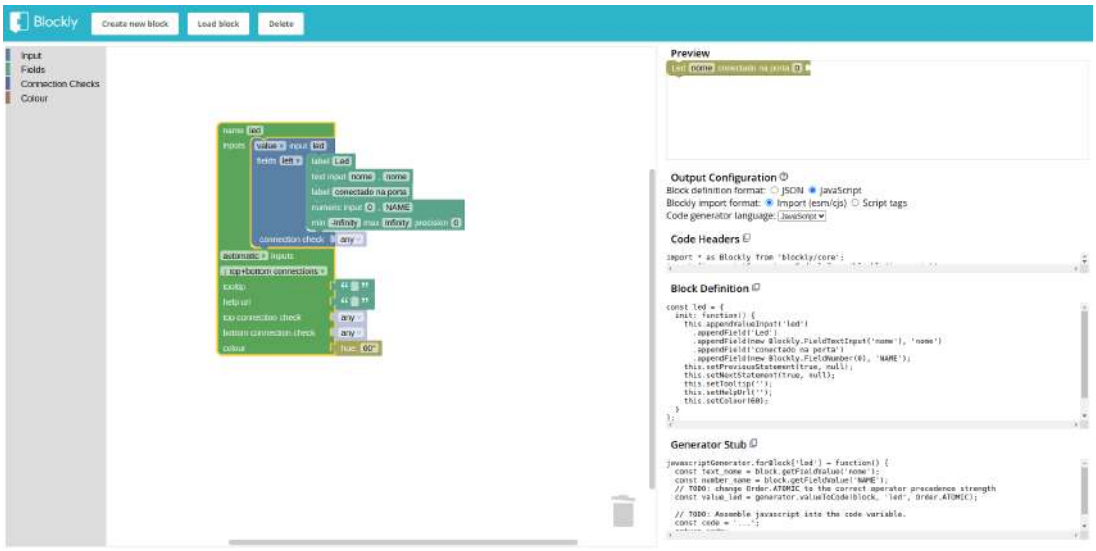


Figura 4.7: Interface do Blockly Factory durante a criação de um bloco personalizado.

Sendo assim, utilizando essa ferramenta, os blocos foram criados com foco na clareza e funcionalidade, de modo a facilitar o entendimento por parte dos usuários, especialmente iniciantes. Cada bloco foi desenhado com nomenclaturas diretas e intuitivas, além de cores e categorias bem definidas.

Na Figura 4.8, pode-se observar um exemplo de uma das categorias desenvolvidas, intitulada *Funções*. Nela, estão agrupados blocos que tratam da criação e chamada de funções, estruturas condicionais e ciclos de execução. A organização facilita o uso pedagógico e a reutilização de trechos de código.

4.4.5 Desafios Técnicos e Soluções

Durante a implementação, surgiram diversos desafios técnicos que exigiram adaptações e pesquisas aprofundadas.

Um dos principais desafios foi encontrar uma forma estável e funcional de realizar a comunicação entre o *software* e a placa Arduino. Como o objetivo era permitir que o código

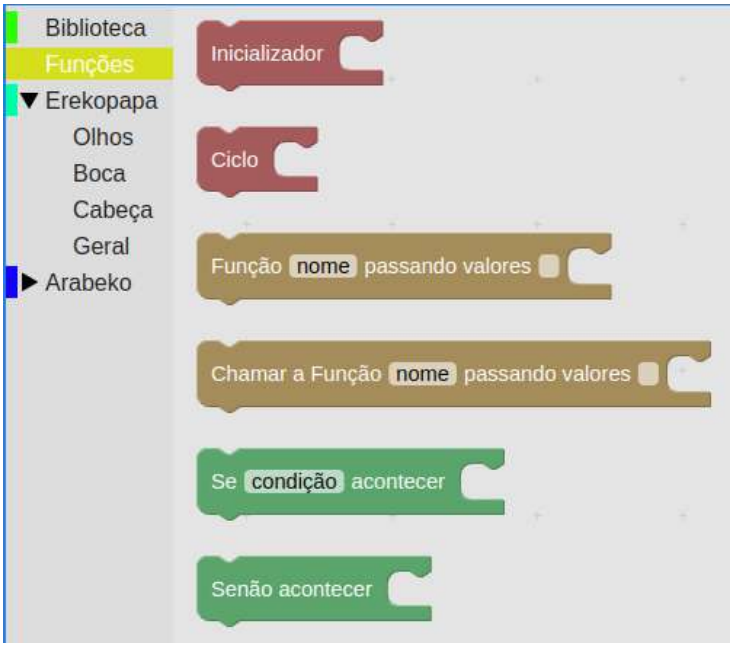


Figura 4.8: Exemplo da aba “Funções” com blocos personalizados criados com Blockly Factory.

gerado pelos blocos fosse enviado diretamente à placa, foi necessário estudar diferentes abordagens de conexão serial e utilização do Arduino CLI. Ainda assim, a integração completa entre o *backend* e o *frontend* não foi possível de ser executada diretamente via navegador por limitações de segurança e acesso ao sistema de arquivos e portas. Dessa forma, optou-se por manter o *backend* rodando localmente na aplicação empacotada com Electron.

Outro obstáculo foi a conversão do código visual construído com blocos para a linguagem C/C++, utilizada pelo Arduino. Para isso, foram utilizadas bibliotecas já existentes no GitHub, geralmente adaptadas de outros projetos educacionais. Alguns exemplos analisados e utilizados como base foram os repositórios dos projetos *BlocklyDuino* (52) e *ArduBlockly* (51), que auxiliaram na estruturação da geração de código a partir dos blocos personalizados.

Além disso, diversas iterações de testes foram realizadas com o objetivo de melhorar a experiência do usuário, tanto no navegador (em ambiente local) quanto por meio da aplicação empacotada no Electron. Os testes permitiram ajustes na organização visual dos blocos, na clareza das instruções, no feedback fornecido durante a geração e envio do código e na responsividade da interface.

Essas dificuldades, embora desafiadoras, foram fundamentais para o amadurecimento do projeto e permitiram construir uma solução funcional e acessível para iniciantes em programação com Arduino.

4.5 Testes e Validação

Após a implementação do sistema, foram realizados testes com o objetivo de verificar a conformidade da biblioteca com os requisitos estabelecidos, bem como avaliar a usabilidade da plataforma. Esta seção descreve os testes aplicados e os cenários avaliados.

4.5.1 Primeiro Teste de Sistema e Usabilidade com Usuários

Com o intuito de avaliar a integridade do sistema e a usabilidade da plataforma, foram realizados encontros individuais com integrantes do grupo de pesquisa e professores da área de computação. O objetivo principal dessa etapa foi identificar oportunidades de melhoria com base na experiência prática dos usuários, além de coletar *feedbacks* relevantes para o processo de validação do projeto.

Durante o teste, os usuários foram convidados a realizar as seguintes tarefas:

- Leitura de uma cartilha contendo orientações sobre o uso da plataforma Rupyly, sendo que este material serviu como base para o e-book posteriormente. O material explicava as funcionalidades da interface, a lógica de funcionamento dos blocos, a estrutura das bibliotecas disponíveis, o processo de criação de programas e o envio do código gerado para a placa Arduino;
- Utilizar os blocos de programação para criar um programa simples para acionar LEDs, buzzer, Servomotor do Erekopapa;
- Gerar e fazer o Download do código-fonte a partir da montagem dos blocos;
- Preenchimento de um formulário digital com perguntas objetivas e discursivas sobre a experiência de uso da plataforma.

O formulário foi estruturado em três eixos principais de avaliação:

1. Nível de familiaridade dos participantes com Arduino e a com programação em blocos;
2. Avaliação da interface gráfica, dos blocos, das bibliotecas e do layout da plataforma;
3. Percepção sobre o potencial do sistema como ferramenta educacional para o ensino de Arduino e conceitos de programação.

Os encontros ocorreram de forma presencial, com duração média de 30 minutos a uma hora. Participaram da atividade nove pessoas, entre docentes e discentes do grupo

de pesquisa Ereko, selecionadas de acordo com o nível de conhecimento dos modelos robóticos, e os dados obtidos foram analisados tanto quantitativa quanto qualitativamente.

Com relação ao perfil dos participantes (Figura 4.9), observou-se que sete dos nove já haviam tido contato prévio com a plataforma Arduino, enquanto dois relataram nunca ter utilizado o microcontrolador anteriormente. Quanto à familiaridade com programação em blocos, oito participantes declararam já ter tido experiências anteriores com esse tipo de linguagem visual, o que facilitou a navegação e a compreensão da proposta do sistema.

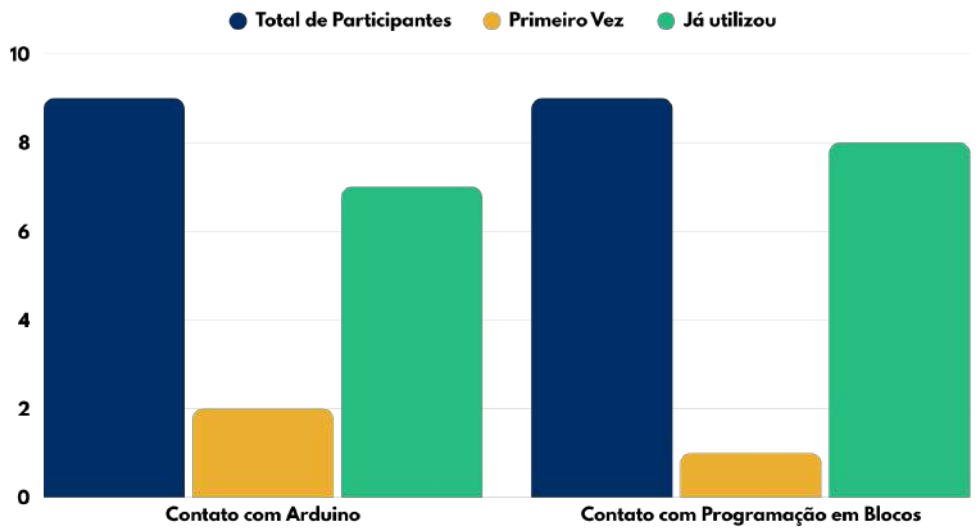


Figura 4.9: Perfil dos participantes da primeira versão

4.5.2 Segundo Teste de Sistema e Usabilidade com Usuários

Após a implementação da versão funcional da plataforma Rupyly e a criação do E-book *Rupyly: Programando em Blocos*, foi conduzida uma nova fase de testes com usuários, com o intuito de validar as melhorias implementadas a partir do primeiro teste e aprofundar a análise da usabilidade do sistema.

Esta etapa teve como objetivos: verificar a estabilidade do sistema, identificar possíveis falhas remanescentes, compreender a experiência de uso dos participantes e avaliar se as funcionalidades desenvolvidas atendiam aos requisitos definidos previamente.

A metodologia adotada consistiu na aplicação de um teste de sistema e usabilidade com usuários representando o público-alvo do projeto, principalmente estudantes já experientes e iniciantes em robótica. O software Rupyly foi disponibilizado em sua versão atualizada, juntamente com o E-book de apoio, permitindo que os participantes explorassem a plataforma de maneira autônoma.

Durante o teste, os usuários foram convidados a realizar as seguintes tarefas:

- Acessar a interface do Rupyly e reconhecer os elementos disponíveis na tela;
- Leitura total ou parcial do E-book;
- Utilizar os blocos de programação para criar um programa simples para acionar LEDs, buzzer, Servomotor do Erekopapa; ou
- Utilizar os blocos de programação para criar um programa simples para acionar os motores do Arabeko, para que pudesse andar em formatos geométricos diferentes;
- Gerar o código-fonte a partir da montagem dos blocos;
- Enviar o código gerado diretamente para uma placa Arduino conectada ao computador.

Ao final da atividade, os participantes responderam a um formulário *on-line* contendo perguntas objetivas e abertas sobre a usabilidade da plataforma, clareza da interface, compreensão dos blocos personalizados e facilidade na execução das tarefas propostas.

O formulário foi estruturado em sete seções:

1. Experiência geral do software;
2. Experiência do usuário com o E-book – *Rupyly: Programando em Blocos*;
3. Avaliação da interface e da navegação;
4. Avaliação da Biblioteca do Erekopapa;
5. Avaliação da Biblioteca do Arabeko;
6. Avaliação da clareza e funcionalidade dos blocos;
7. Avaliação sobre a aplicabilidade do software para o ensino de programação e Arduino.

A aplicação do teste ocorreu durante o mês de junho de 2025, de forma presencial, com duração de 30 minutos a uma hora. Foram selecionados para a atividade 13 usuários (Figura 4.10), sendo discentes da Universidade de Brasília com diferentes níveis de familiaridade com programação e Arduino. Desses, oito não haviam participado da primeira rodada de testes, enquanto cinco já haviam contribuído anteriormente.

Em relação à formação acadêmica, sete participantes eram estudantes do curso de Ciência da Computação, quatro de Engenharia Mecatrônica, um mestrando em Sistemas Mecatrônicos e um formado em Aviação.

A maioria dos participantes já possuía experiência prévia com tecnologias correlatas: dez afirmaram que não era a primeira vez utilizando Arduino, e outros dez também já haviam tido contato com programação em blocos. Entre os softwares mencionados anteriormente utilizados destacam-se: *Scratch*, *App Inventor*, *Tinkercad*, *Blueprint (Unreal Engine 5)*, *Blockly Games* e mBlock.

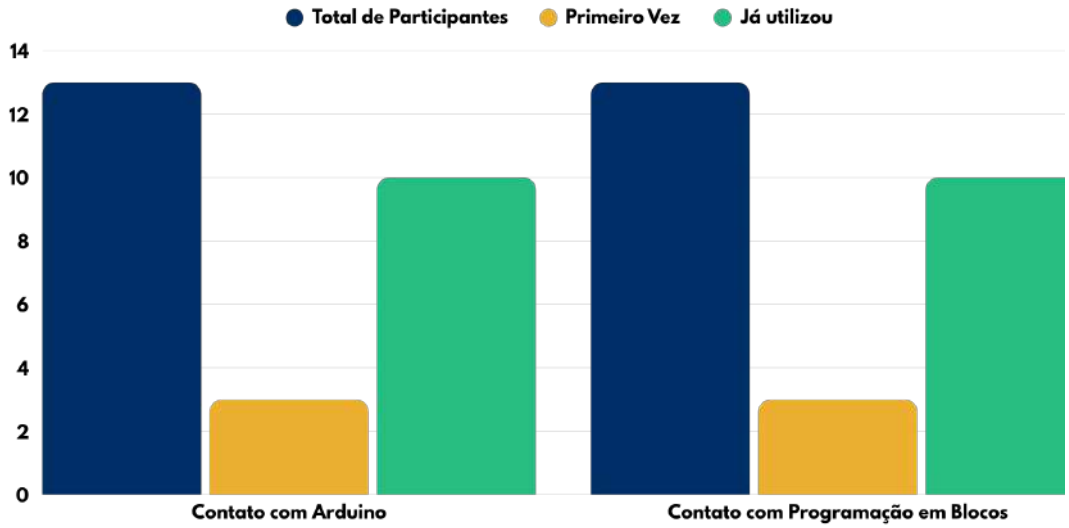


Figura 4.10: Perfil dos participantes da segunda versão

As respostas coletadas foram analisadas sob uma abordagem tanto quantitativa quanto qualitativa, fornecendo contribuições valiosas para a identificação de pontos fortes da plataforma, bem como para o planejamento de futuras melhorias.

4.6 Documentação

A documentação de software pode ser definida como um artefato cuja principal finalidade é comunicar informações relevantes sobre o sistema ao qual está associada (56). Ela abrange um conjunto de manuais, tanto técnicos quanto gerais, organizados por meio de diferentes recursos como textos explicativos, comentários, dicionários de dados, diagramas, fluxogramas, gráficos e ilustrações (57). Essa documentação é essencial não apenas para descrever a estrutura e o comportamento do sistema, mas também para oferecer suporte à visualização e ao controle da arquitetura, facilitar a identificação de oportunidades de simplificação e reaproveitamento de componentes e contribuir para a mitigação de riscos no desenvolvimento de software (58).

No contexto deste trabalho, a documentação e o controle de versão foram elaborados e mantidos por meio da plataforma GitHub, organizando-se em um repositório dedicado.

Essa abordagem permitiu o registro detalhado, transparente e cronológico de todas as etapas do desenvolvimento, favorecendo a rastreabilidade das modificações e o alinhamento com as boas práticas de engenharia de software. Contudo, é importante destacar que, devido ao processo de registro em andamento do projeto, o código-fonte não está, neste momento, publicamente acessível ou conforme os princípios do movimento de software livre e open-source, embora a estrutura e o método de documentação sigam essas diretrizes para futura replicabilidade, manutenção e evolução.

Este capítulo detalhou os métodos e etapas de desenvolvimento do software Rupyly, um projeto que visa simplificar a programação visual para robôs educacionais como o Erekopapa e o Arabeko. Inicia-se com a descrição do E-book Rupyly: Programando em Blocos, um guia introdutório essencial para a plataforma. Em seguida, são apresentadas a análise de requisitos funcionais e não funcionais, que guiaram o design e a implementação da biblioteca. A seção de design da biblioteca explica a arquitetura modular do software, com uma visão geral do *frontend* e *backend* e a justificativa das escolhas técnicas, incluindo o uso de Blockly e Arduino CLI. O capítulo também aborda a implementação detalhada dos componentes e blocos personalizados, os desafios técnicos superados e as duas fases de testes e validação com usuários, que permitiram aprimorar a usabilidade da interface e a eficácia pedagógica. Finalmente, a seção de documentação agora se volta para a gestão do projeto via GitHub, estabelecendo a base para a apresentação dos resultados das fases de teste, que serão detalhados no próximo capítulo.

Capítulo 5

Resultados

Neste capítulo serão apresentados os principais resultados obtidos ao longo do desenvolvimento do projeto, com foco nas experiências práticas, materiais produzidos e evolução das ferramentas educacionais propostas. A análise é fundamentada em observações realizadas durante oficinas, testes com usuários e reflexões a partir da aplicação dos recursos desenvolvidos.

5.1 Experiência do Usuário

5.1.1 Primeira Versão

A análise da experiência do usuário na primeira versão da plataforma (Figura 5.1) considerou aspectos relacionados à disposição visual da interface, à localização e organização das bibliotecas, bem como à usabilidade das áreas destinadas à construção dos blocos e à visualização do código gerado. Os participantes da avaliação atribuíram notas em uma escala de 1 a 5, onde 1 representava uma experiência "muito ruim" e 5 indicava uma experiência "excelente".

Com relação ao layout geral, 7 dos 9 participantes atribuíram nota 4, enquanto os 2 restantes atribuíram nota 5, indicando uma avaliação predominantemente positiva, mas com margem para ajustes visuais e de organização.

No que refere à localização das bibliotecas, os resultados indicaram uma recepção majoritariamente positiva: 5 participantes (55,6%) atribuíram a nota máxima (5), enquanto 2 (22,2%) concederam nota 4 e os 2 restantes (22,2%) nota 3. Esta distribuição sugere uma boa aceitação da estrutura de navegação e acessibilidade das bibliotecas na interface.

Por outro lado, a separação e categorização entre as bibliotecas exibiu uma percepção mais variada entre os participantes. 4 (44,4%) atribuíram nota 5, três (33,3%) concederam nota 4, um (11,1%) marcou nota 3, e outro (11,1%) atribuiu nota 2. Essa variabilidade

nos resultados aponta que, embora a categorização dos blocos por biblioteca seja funcional para alguns usuários, ela ainda demanda aprimoramentos em termos de clareza e padronização para alcançar maior eficácia e unanimidade na percepção dos usuários.

Em relação às áreas destinadas à manipulação dos blocos e do código em C, ambas foram bem avaliadas. Seis participantes deram nota 5 e quatro marcaram nota 4 em cada uma dessas seções, reforçando a percepção de que os espaços são adequados e funcionais para a construção dos programas.

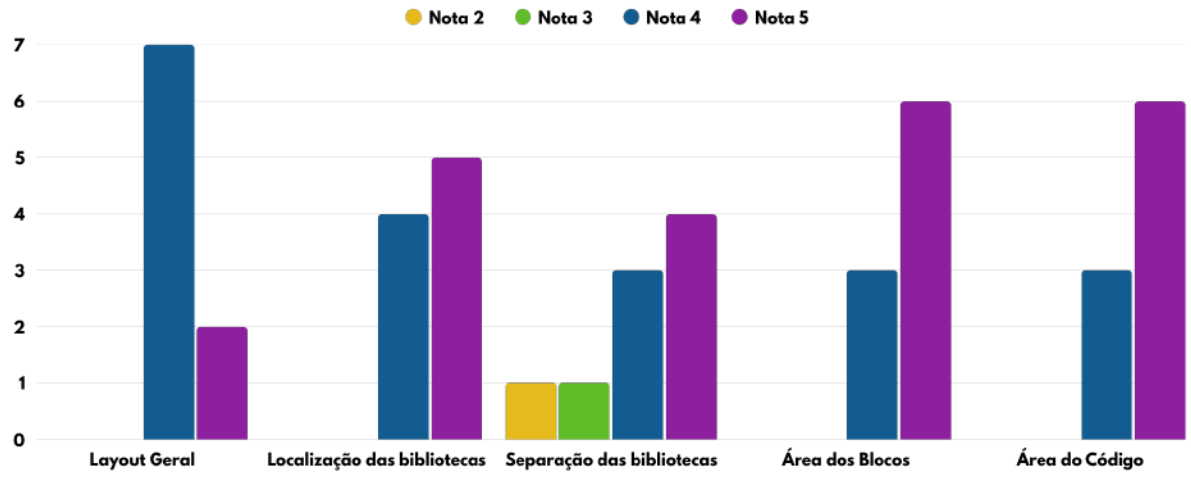


Figura 5.1: Resultado do Feedback do Layout - 1ª Versão

Além das avaliações quantitativas, os participantes forneceram sugestões qualitativas valiosas, que podem contribuir significativamente para a evolução da interface. As principais recomendações foram:

- Inserir uma opção de blocos mais genéricos, voltados à criação de funções personalizadas pelo usuário, ampliando as possibilidades de uso do sistema para fins mais flexíveis e avançados;
- Reorganizar algumas funções alocadas em módulos educacionais específicos para categorias mais gerais, visando facilitar a reutilização desses blocos em diferentes contextos, considerando especialmente a coexistência de múltiplos projetos. Tal reorganização deve ser guiada por um mapeamento cuidadoso das funcionalidades comuns entre os projetos atuais e possíveis expansões futuras;

- Adotar uma paleta de cores mais uniforme e harmoniosa. Foi destacado que determinadas cores, como o verde-fluorescente, destoam excessivamente do restante da interface e podem comprometer a legibilidade e o conforto visual;
- Possibilitar a edição de determinados dados diretamente na área de código em C, permitindo que essas alterações reflitam automaticamente nos blocos correspondentes, promovendo maior integração entre a programação textual e visual.

As contribuições apresentadas nesta etapa evidenciam a importância de continuar refinando a interface gráfica da plataforma com base nas necessidades reais dos usuários. Para isso, levou-se em consideração, a busca por um equilíbrio entre acessibilidade, flexibilidade e consistência visual essencial para o amadurecimento da segunda versão.

5.1.2 Segunda Versão

A segunda versão do software apresentou avanços significativos em relação à experiência do usuário, conforme evidenciado pelas respostas obtidas no formulário de avaliação aplicado a 13 participantes. Assim como na primeira versão, os participantes da avaliação atribuíram notas em uma escala de 1 a 5, onde 1 representava uma experiência "muito ruim" e 5 indicava uma experiência "excelente".

Para esta nova versão, foi desenvolvida levando em consideração, principalmente, as sugestões dos participantes. Com isso em mente, foram adicionados blocos mais genéricos às bibliotecas, permitindo a criação de funções personalizadas e ampliando as possibilidades de programação. A biblioteca também foi reorganizada para facilitar a localização dos blocos necessários, tornando a construção do código mais intuitiva. Além disso, foi adotada uma nova paleta de cores para melhorar a acessibilidade e a experiência visual de todos os usuários.

Sendo assim, no que diz respeito à experiência geral, 10 participantes atribuíram nota 5 e os 3 restantes deram nota 4, indicando um alto nível de satisfação com a nova versão. Ademais, todos os respondentes afirmaram que o software atendeu plenamente às suas expectativas, o que reforça a efetividade das melhorias implementadas.

Entre os aspectos mais destacados positivamente, os usuários mencionaram:

- Facilidade na montagem dos blocos;
- Simplicidade geral da interface;
- Paleta de cores variadas e agradáveis;
- Navegação intuitiva e fácil compreensão das funcionalidades.

Apesar da recepção positiva, algumas dificuldades pontuais foram identificadas por alguns participantes. Entre elas, destacam-se:

- Dificuldade na edição ou escrita dos nomes nos blocos já pré-configurados;
- Necessidade de utilizar funções adicionais dentro de blocos específicos, o que exigiu um entendimento mais aprofundado da lógica de programação;
- Dificuldades iniciais na compreensão de alguns conceitos de programação, sobretudo por parte de usuários iniciantes.

Ainda assim, todos os participantes relataram que o sistema se mostrou intuitivo e que não houve dificuldades significativas na compreensão das funcionalidades básicas do software. Tais resultados indicam que a segunda versão apresenta um avanço substancial em termos de usabilidade, promovendo uma experiência mais fluida, acessível e condizente com o público-alvo do projeto.

Em comparação à primeira versão, a avaliação da usabilidade da interface da plataforma na segunda versão revelou uma alta satisfação geral dos usuários (Figura 5.2). O layout geral foi amplamente aprovado, com 8 participantes atribuindo a nota máxima (5) e os 5 restantes optando pela nota 4. A localização das bibliotecas obteve consenso absoluto, recebendo nota 5 de todos os 13 participantes, indicando sua perfeita adequação. Similarmente, a separação das bibliotecas por categorias foi muito bem recebida, com 12 avaliações de nota 5 e apenas 1 de nota 4. A área destinada à construção dos blocos também demonstrou excelente usabilidade, com 11 participantes atribuindo nota 5 e 2 optando por nota 4. A única variação mais notável ocorreu na área de visualização do código gerado, onde 11 usuários deram nota 5, 1 deu nota 4 e 1 deu nota 3, sugerindo que, embora altamente funcional, há um pequeno espaço para refinamento nesse aspecto. No geral, os resultados sublinham a eficácia do design da interface em proporcionar uma experiência intuitiva e positiva.

As observações levantadas nesta etapa servirão como incentivo para ajustes pontuais e para o contínuo refinamento da interface, especialmente no que diz respeito à flexibilidade dos blocos pré-configurados e ao apoio conceitual para iniciantes.

5.2 E-book - Rupyly: Programando em Blocos

O e-book desenvolvido atua como uma ponte crucial entre o usuário e o software, fomentando a autonomia na criação de projetos educacionais que combinam programação visual e robótica.

Segundo os usuários (Figura 5.3), o material cumpriu bem esse papel. Em um formulário aplicado com 13 participantes das oficinas, 3 afirmaram ter lido o e-book por

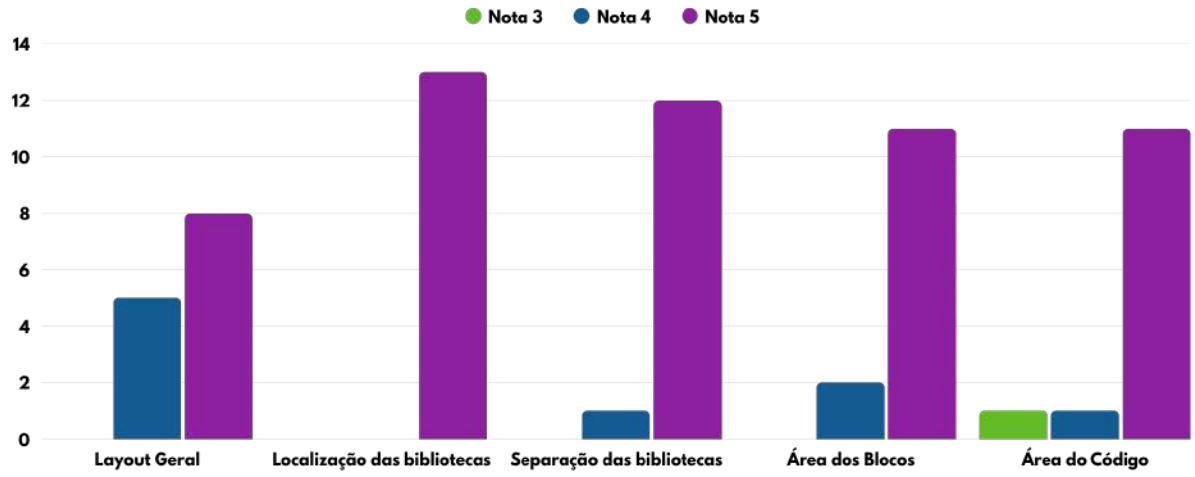


Figura 5.2: Resultado do Feedback do Layout - 2ª Versão

completo, enquanto 8 realizaram a leitura parcial e 2 não realizaram a leitura. Quando questionados se o material atendeu às suas expectativas, 9 participantes responderam de forma positiva.

Além disso, 100% dos respondentes afirmaram que a leitura foi fácil de entender, e aproximadamente 90% relataram que o conteúdo os ajudou a compreender o funcionamento do software e a realizar a montagem dos blocos para programar os robôs.

Apesar da avaliação majoritariamente positiva, os participantes também apresentaram sugestões construtivas que apontam possíveis melhorias no material. Entre os principais pontos destacados estão:

- A extensão do e-book foi considerada excessiva por alguns usuários, o que pode comprometer a atenção e o engajamento durante a leitura. Foi sugerida a divisão do conteúdo em materiais menores, organizados por casos de uso específicos;
- A visualização completa dos trechos de código em linguagem C, especialmente na seção referente ao Arabeko, foi mencionada como um aspecto a ser aprimorado;
- Alguns participantes indicaram que o conteúdo poderia ser mais conciso, com foco em informações mais diretas e objetivas;
- Houve apontamentos sobre a densidade de informações, que pode representar um desafio adicional para iniciantes;

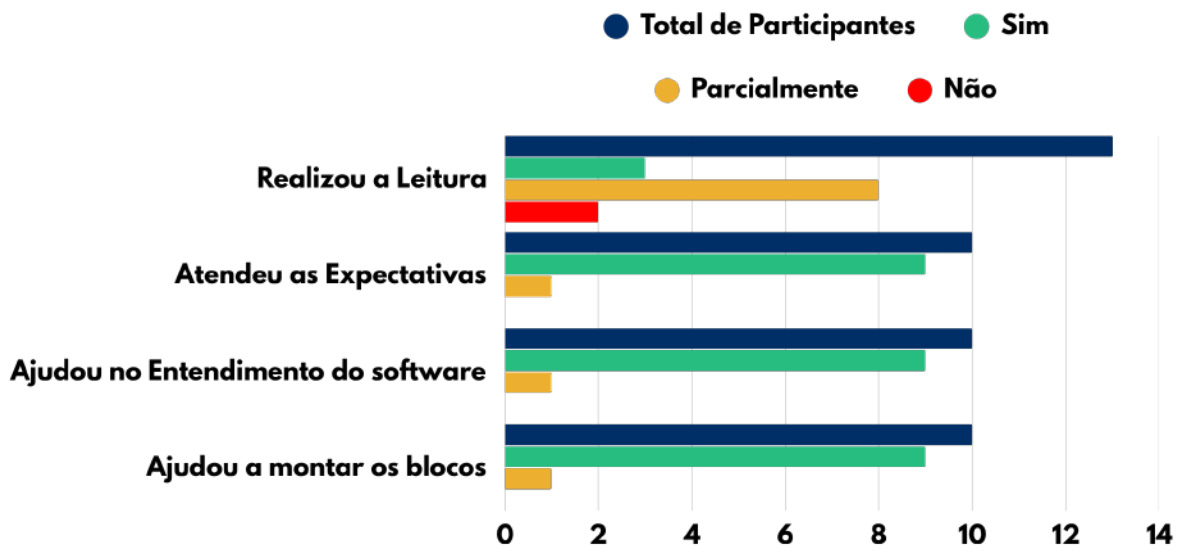


Figura 5.3: Resultados da Leitura do E-book

- Foi sugerida a padronização do tamanho das páginas, a fim de proporcionar uma experiência de leitura mais uniforme e agradável.

As observações reunidas evidenciam que o e-book apresenta um forte potencial como recurso introdutório e de apoio ao processo de aprendizagem. No entanto, ressaltam também a importância de aprimoramentos voltados à concisão do conteúdo, à estruturação modular e à adaptação às necessidades de diferentes perfis de usuários.

5.3 Blocos em Geral

5.3.1 Primeiro Versão

A avaliação dos blocos da primeira versão (Figura 5.4) do software contou com a participação de 9 usuários. O objetivo desta etapa foi compreender a percepção geral sobre a aparência, usabilidade e estrutura lógica dos blocos oferecidos na interface de programação.

Em relação à aparência visual, 7 participantes atribuíram nota 5, indicando alto grau de satisfação, enquanto os outros 2 atribuíram nota 4, demonstrando uma avaliação positiva de forma geral.

Quanto à intuitividade e facilidade de uso, 5 participantes classificaram os blocos com nota 5, 3 atribuíram nota 4, 1 deram nota 3 e 1 nota 2. Esses dados revelam que, em-

bora a maioria tenha considerado os blocos intuitivos, ainda há espaço para melhorias na usabilidade, especialmente para usuários com menos familiaridade com lógica de programação. Entretanto, a busca por uma maior intuitividade deve ser equilibrada para não comprometer a futura transição dos usuários para a programação baseada em código textual, evitando que a excessiva simplificação crie lacunas no entendimento fundamental da sintaxe e da lógica.

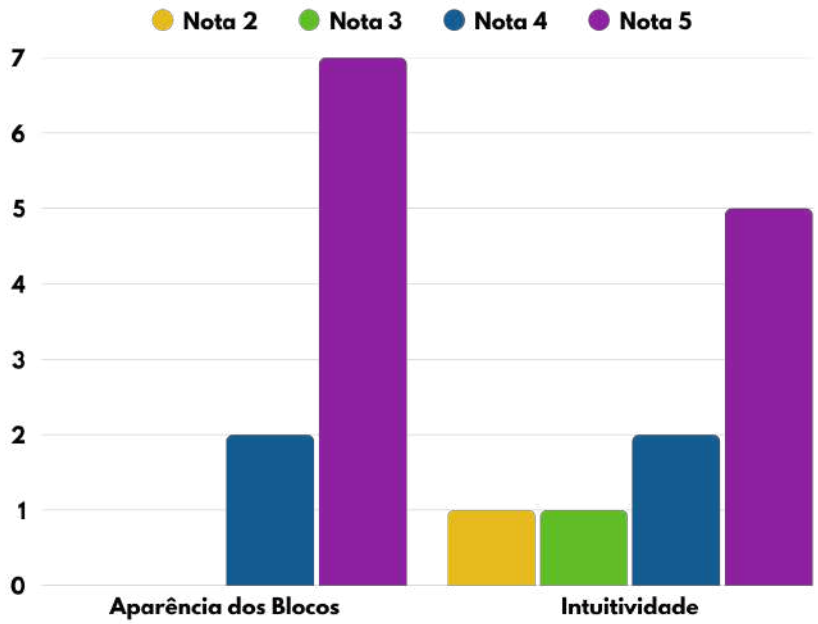


Figura 5.4: Resultados em relação aos Blocos - 1ª Versão

Além das notas atribuídas, os participantes também contribuíram com sugestões qualitativas importantes para o aprimoramento da ferramenta. As principais recomendações foram:

- Estabelecer uma separação mais clara entre blocos iniciais (que devem ser inseridos no início da programação) e os demais, com o objetivo de auxiliar usuários iniciantes que ainda não compreendem a estrutura básica de um programa;
- Implementar um sistema de preenchimento automático para nomes de variáveis já declaradas, de modo que, ao iniciar a digitação, o ambiente de desenvolvimento ofereça sugestões contextuais, facilitando a reutilização correta das variáveis;
- Permitir a seleção múltipla de blocos, possibilitando que o usuário copie e cole vários elementos simultaneamente, o que otimiza o tempo de construção do código;

- Substituir os campos de texto utilizados em estruturas condicionais (como os blocos *if*) por blocos específicos para condições, oferecendo uma abordagem mais estruturada e menos propensa a erros.

Essas recomendações reforçam a importância de uma interface ainda mais orientada à experiência do usuário, contemplando tanto aspectos de acessibilidade para iniciantes quanto recursos de eficiência para usuários mais avançados. Tais apontamentos serão considerados no processo de evolução da ferramenta, com vistas a tornar a construção dos programas mais clara, prática e intuitiva.

5.3.2 Segunda Versão

A segunda versão dos blocos (Figura 5.5) foi avaliada por 13 participantes, com o objetivo de verificar a efetividade das melhorias implementadas em relação à aparência, linguagem, organização e usabilidade da interface.

Em relação à aparência visual, 12 participantes atribuíram nota 5, enquanto apenas 1 participante marcou nota 4, demonstrando uma aceitação ainda mais positiva em comparação com a versão anterior.

Quanto à linguagem utilizada nos blocos, todos os 13 participantes atribuíram nota 5, evidenciando que o vocabulário adotado se mostrou claro, acessível e adequado ao público-alvo.

No quesito intuitividade da interface, 12 participantes marcaram nota 5 e 1 participante marcou nota 4, o que reforça a percepção de que os blocos tornaram-se mais fáceis de compreender e utilizar na nova versão.

Outro ponto avaliado foi a organização dos blocos em bibliotecas separadas, especialmente a distinção entre blocos de uso comum e blocos específicos. Diversos comentários destacaram essa separação como uma melhoria significativa, tanto por tornar a navegação mais intuitiva quanto por facilitar a localização dos blocos desejados. Os participantes elogiaram a clareza dessa divisão, considerando-a uma solução funcional e bem-vinda, especialmente para iniciantes.

Os resultados indicam que as atualizações realizadas na segunda versão contribuíram positivamente para a experiência do usuário, promovendo maior clareza, organização e acessibilidade no ambiente de programação. A adoção de uma linguagem simples, aliada à estrutura modular e intuitiva dos blocos, demonstrou-se eficaz para o público-alvo, conforme evidenciado pelas avaliações e comentários recebidos.

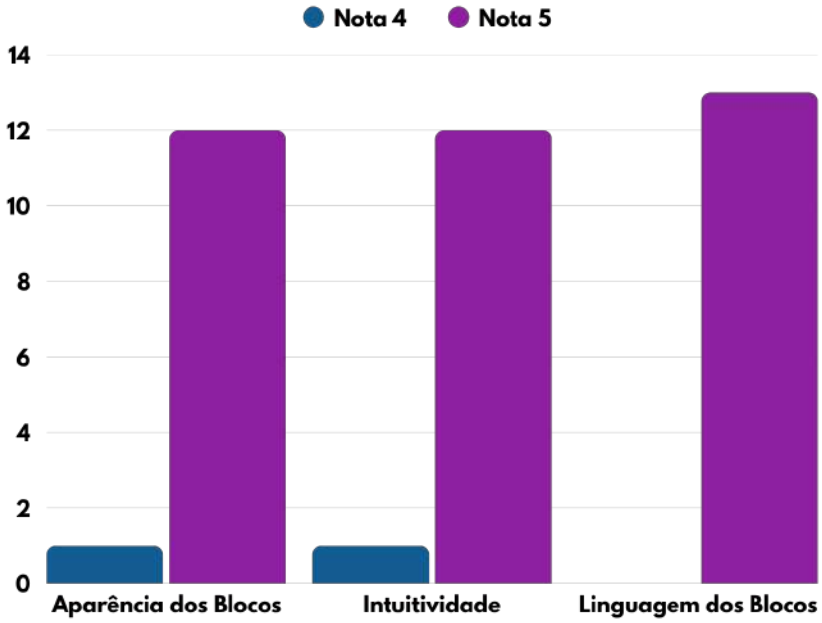


Figura 5.5: Resultados em relação aos Blocos - 2ª Versão

5.4 Biblioteca do Erekopapa

Foi criada uma biblioteca específica, chamada Erekopapa, para a plataforma Rupyly, dividida em partes relacionadas à montagem dos modelos durante as oficinas realizadas. Essa organização teve como objetivo facilitar o entendimento e a utilização dos blocos por parte dos alunos participantes. A biblioteca foi estruturada da seguinte forma:

- Olhos: contém os blocos relacionados ao controle dos LEDs;
- Boca: reúne os blocos destinados ao controle do buzzer;
- Cabeça: disponibiliza os blocos para acionamento de motores;
- Geral: agrega blocos de funções genéricas que podem ser utilizadas tanto na função *void setup* quanto na *void loop*.

A interface do sistema foi projetada para permitir que, à medida que os blocos são organizados na área de montagem, o código em linguagem C correspondente seja gerado automaticamente na lateral da tela. Ao concluir a construção do programa, o usuário pode utilizar o botão *Download* para exportar o arquivo no formato ".ino", compatível com a IDE do Arduino, possibilitando a compilação e execução direta do código.

5.4.1 Primeira Versão

Durante os testes da primeira versão do software, nove participantes avaliaram a biblioteca personalizada (Figura 5.6). Em relação à aparência geral da biblioteca, quatro participantes atribuíram nota 5 (máxima) e cinco deram nota 4. Quanto à organização da biblioteca em seções (Olhos, Boca, Cabeça e Geral), quatro participantes deram nota 5, três atribuíram nota 4, um atribuiu nota 3 e um atribuiu nota 2. Esses resultados indicam que, apesar de bem recebida em termos visuais, a estruturação poderia ser ainda mais clara para alguns usuários.

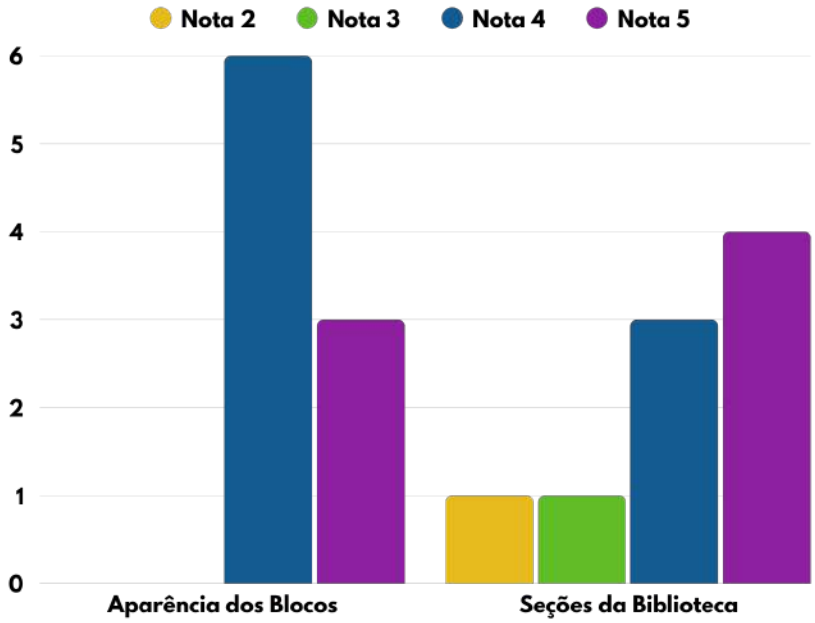


Figura 5.6: Resultados em relação à Biblioteca do Erekopapa - 1ª Versão

Em relação ao potencial do uso da biblioteca para fins educacionais, oito participantes afirmaram que a plataforma seria eficaz para ensinar e aprender programação com Arduino, enquanto um respondeu “talvez”.

Além das avaliações quantitativas, foram registradas sugestões qualitativas importantes, que contribuíram para melhorias nas versões subsequentes:

- Revisar o código C gerado por alguns blocos, pois apresentavam erros como ausência de letras ou símbolos;
- Incluir explicações ou descrições breves sobre a funcionalidade de cada bloco;
- Adicionar pequenas imagens ao lado do texto dos blocos para facilitar o reconhecimento visual;

- Ajustar a paleta de cores para melhorar a diferenciação entre os blocos e tornar a interface mais agradável.

Tais apontamentos revelam uma participação crítica e construtiva por parte dos usuários, o que foi essencial para a evolução da plataforma. As observações foram incorporadas no desenvolvimento da segunda versão da biblioteca, visando proporcionar uma experiência mais fluida e intuitiva.

5.4.2 Segunda Versão

Na segunda versão, a biblioteca de blocos personalizados manteve a mesma estrutura lógica, com a divisão em quatro categorias: olhos, boca, cabeça e geral. No entanto, foram incorporadas diversas melhorias sugeridas pelos participantes na primeira rodada de testes. Entre as principais mudanças, destaca-se a remoção do texto em formato de código nos blocos, que foi substituído por uma linguagem mais acessível e uma revisão minuciosa do código C gerado por alguns blocos, corrigindo possíveis erros. Além disso, foi incluída uma breve descrição funcional em cada bloco, com o objetivo de facilitar o entendimento do seu propósito e aplicação.

Durante a nova fase de teste, 11 dos 13 participantes utilizaram a biblioteca personalizada aplicada ao modelo Erekopapa (Figura 5.7). Desse grupo de 11, cinco participantes já haviam testado a primeira versão da biblioteca. E após a nova avaliação, 10 atribuíram nota máxima (5) à aparência visual da biblioteca, enquanto um avaliou com nota 4. Em relação à organização dos blocos baseada nas partes físicas do robô (cabeça, olhos, boca e geral), a avaliação foi unânime: todos os participantes atribuíram nota 5, indicando uma excelente recepção quanto à clareza e estrutura.

Outro aspecto avaliado foi a facilidade de uso da biblioteca por novos usuários. Dos 11 participantes, 10 afirmaram que a biblioteca está suficientemente intuitiva para quem está iniciando na plataforma, evidenciando avanços significativos em termos de usabilidade.

Além disso, todos os participantes concordaram que a forma de ensino proposta por meio da interface e da biblioteca, integrando programação em blocos com conceitos de robótica, é adequada para o aprendizado de alunos iniciantes, reforçando a viabilidade pedagógica do projeto.

Esses resultados demonstram que as alterações feitas com base nos testes iniciais tiveram impacto positivo na experiência dos usuários e contribuíram para tornar a biblioteca mais acessível, compreensível e eficaz em seu propósito educativo.

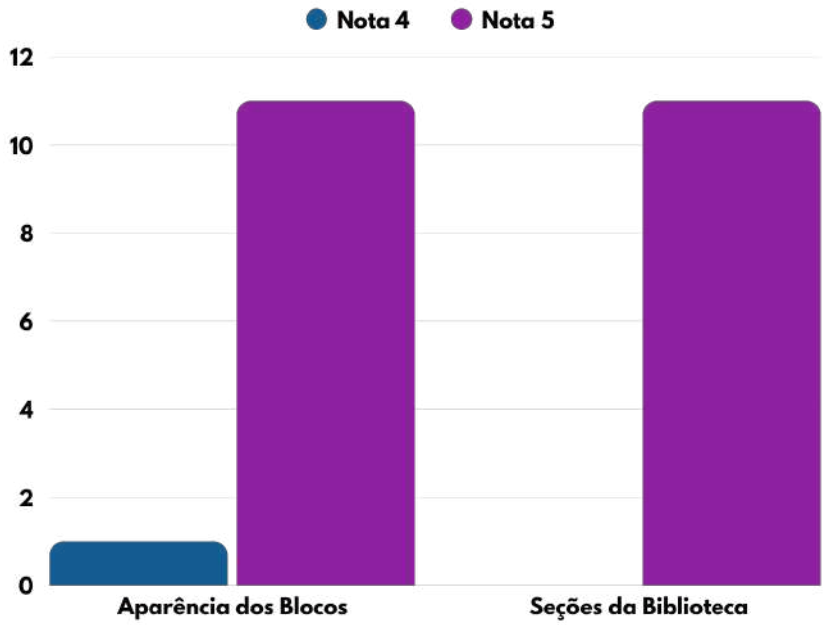


Figura 5.7: Resultados em relação à Biblioteca do Erekopapa - 2ª Versão

5.5 Biblioteca do Arabeko

Inicialmente, o Arabeko foi concebido como uma única biblioteca geral, contendo todos os blocos organizados por categorias específicas de componentes. No entanto, durante a fase inicial de testes, observou-se que essa abordagem não atendia adequadamente usuários iniciantes ou com pouco conhecimento sobre Arduino. Diante dessa limitação, tornou-se evidente a necessidade de segmentar a biblioteca em diferentes níveis de complexidade, a fim de tornar a experiência mais acessível e didática.

Na primeira versão adaptada, a biblioteca foi dividida em três níveis distintos:

- Fácil: composta por blocos com código completo para executar movimentos simples, sem necessidade de modificações por parte do usuário;
- Médio: incluía blocos que também possuíam o código completo, mas com as direções dos movimentos separadas, exigindo um pequeno grau de montagem e lógica por parte do usuário;
- Difícil: voltado a usuários com maior familiaridade com programação, oferecendo blocos mais abertos, permitindo ao usuário construir seu próprio programa a partir de funções básicas.

Apesar da proposta inicial de níveis progressivos, notou-se, ao longo das oficinas e validações práticas, que a divisão em três níveis causava certa confusão entre os parti-

cipantes. Assim, na segunda versão da biblioteca, optou-se por uma organização mais simples e funcional, estruturada em apenas dois módulos:

- Autônomo: voltado para iniciantes, este módulo oferece blocos com código completo que executam ações individuais e específicas (por exemplo, movimentar-se para frente, movimentar-se em forma de quadrado ou circular, etc.);
- Original: voltado a usuários mais experientes ou em fase de transição, este módulo disponibiliza todos os blocos básicos necessários para a construção completa do programa, além de oferecer também blocos de direções pré-montados, que servem como apoio para facilitar a montagem do código final.

Essa reformulação teve como objetivo tornar a ferramenta mais intuitiva, promovendo um aprendizado mais fluido e adaptável às diferentes realidades dos participantes.

5.5.1 Primeira Versão

A biblioteca Arabeko na primeira versão foi estruturada em três níveis de complexidade para atender a diferentes perfis de usuários, desde iniciantes até os mais experientes, facilitando a aprendizagem progressiva do robô e da programação em blocos.

No nível fácil, foram criados blocos que já contêm todo o código necessário para o funcionamento do Arabeko. Por exemplo, o bloco "Fazer um quadrado de 10 cm" gera automaticamente o código em linguagem C para a IDE do Arduino, que faz o Arabeko se deslocar formando um quadrado de lado 10 cm. De forma semelhante.

No nível médio, o usuário pode modificar as direções que o Arabeko irá percorrer, contando com categorias específicas de blocos para essa finalidade. A categoria "Direções" disponibiliza blocos que indicam as direções a serem seguidas pelo robô, enquanto a categoria "Funções Gerais" inclui blocos relacionados à configuração de pinos, valores digitais, controle da intensidade de componentes, conversão de valores entre intervalos e estruturas condicionais.

No nível difícil, destinado a usuários mais avançados, a biblioteca disponibiliza blocos livres para edição e criação, permitindo maior flexibilidade e personalização dos programas. As principais categorias são: "Serial", para comunicação assíncrona entre a placa Arduino e computadores ou dispositivos externos; "Funções", que abrangem configuração de pinos, controle digital e analógico, conversão de valores e estruturas condicionais; e "Variáveis", com blocos para declaração de variáveis dos tipos *byte* e inteiro (int), além de blocos de atribuição para manipulação dessas variáveis em operações posteriores.

Quanto à avaliação da primeira versão da biblioteca Arabeko (Figura 5.8), os participantes expressaram as seguintes opiniões:

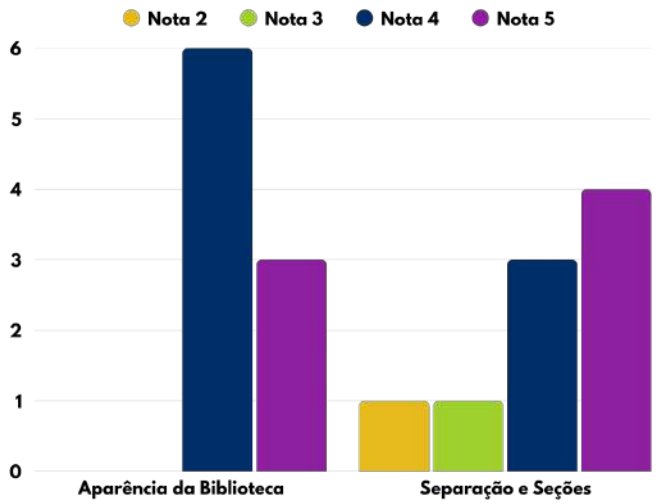


Figura 5.8: Resultados em relação à Biblioteca do Arabeko - 1ª Versão

- Aparência: 6 dos 9 participantes atribuíram nota 4, enquanto 3 deram nota 5;
- Separação das categorias: 4 participantes deram nota 5, 4 deram nota 4, 1 deu nota 3 e 1 deu nota 2;
- Aprendizagem na forma proposta: 8 dos 9 acreditam que a proposta facilita o aprendizado, enquanto 1 marcou "talvez", justificando a necessidade de melhorias no layout.

Foram ainda feitas sugestões para aprimorar a biblioteca, entre as quais se destacam:

- Melhorar a paleta de cores para maior harmonia visual;
- Criar um bloco de movimentação mais geral para simplificar comandos básicos;
- Corrigir códigos relacionados a alguns blocos, nos quais estavam faltando palavras ou símbolos;
- Reavaliar a necessidade de determinados blocos, considerando sua relevância pedagógica e funcional.

5.5.2 Segunda Versão

A avaliação da segunda versão da biblioteca Arabeko na plataforma contou com 4 dos 13 usuários envolvidos nos testes, sendo que um desses participantes já havia testado a versão anterior. Essa biblioteca é considerada mais avançada e, devido à complexidade e ao conhecimento prévio necessário sobre o robô, nem todos os participantes se sentiram familiarizados ou tiveram oportunidade de testá-la durante a atividade.

Apesar do número reduzido de usuários que utilizaram a biblioteca Arabeko nesta fase, os dados coletados oferecem percepções importantes sobre a aplicabilidade e usabilidade dos blocos específicos para esse modelo. A limitada familiaridade dos participantes com o robô reflete a necessidade de um maior suporte pedagógico e materiais de apoio que auxiliem na compreensão e uso da biblioteca em contextos educacionais.

Entre os quatro participantes que testaram a biblioteca Arabeko na segunda versão (Figura 5.9), três atribuíram nota máxima (5) à aparência da biblioteca na modalidade Autônomo, enquanto um avaliou com nota 4. Já na modalidade Original, três deram nota 5 e um atribuiu nota 3.

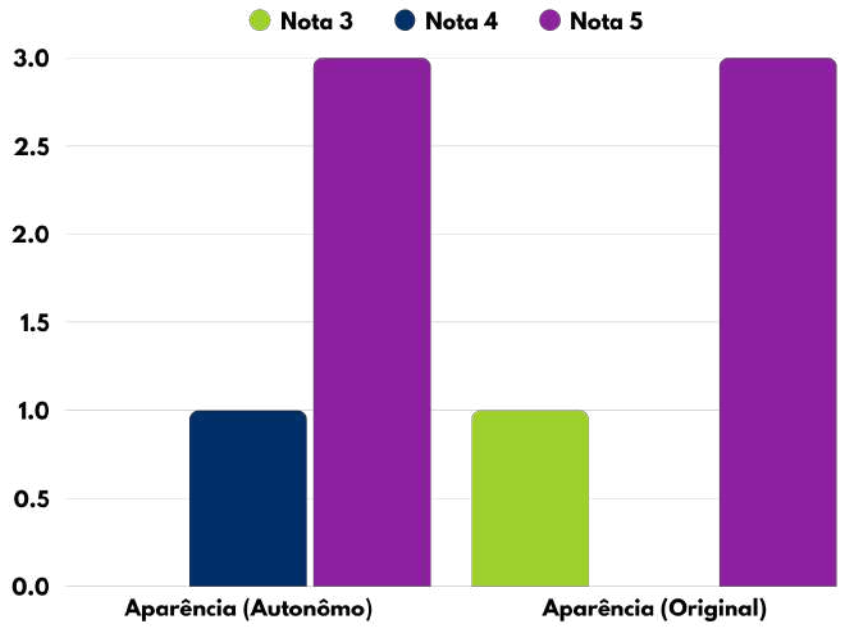


Figura 5.9: Resultados em relação à Biblioteca do Arabeko - 2ª Versão

Os usuários consideraram interessante a separação entre as versões Autônomo e Original, reconhecendo que essa divisão pode ser útil para distintos níveis de aprendizado. No entanto, foi observado que não ficou completamente claro que a versão Autônomo contém o código 100% pronto, o que pode gerar dúvidas durante o uso.

Quanto à intuitividade da biblioteca, três participantes afirmaram que a ferramenta é intuitiva, enquanto um marcou “talvez”, justificando que, caso utilizada com crianças, podem surgir dificuldades na compreensão e manuseio dos blocos.

Todos os participantes concordaram que o método de aprendizagem proposto pela biblioteca é adequado e eficaz para o ensino dos conceitos relacionados.

Foram sugeridas algumas melhorias, entre as quais se destacam:

- Substituir o termo “Autônomo” por expressões como “Palestra de Exemplos” ou “Exemplos Prontos”, para deixar mais claro o propósito dos blocos que vêm com código completo;
- Renomear a biblioteca atualmente intitulada “Motor” para “Braço”, considerando que os blocos referem-se especificamente a esse componente do robô Arabeko.

5.6 Rupyly no ensino de programação e arduino

5.6.1 Primeira Versão

Na avaliação da primeira versão da plataforma, os participantes foram convidados a opinar sobre o potencial do Rupyly como ferramenta de apoio ao ensino de programação e Arduino. Os resultados demonstraram uma recepção promissora, ainda que acompanhada de observações críticas que indicaram pontos de melhoria.

Dos 9 participantes, 6 afirmaram que utilizariam a aplicação com certeza em contextos educacionais, enquanto 3 marcaram a opção “talvez”. Entre os que demonstraram hesitação, destacaram-se comentários sobre a necessidade de testar o sistema com alunos em sala de aula e a importância de ajustes adicionais para ampliar a aplicabilidade da ferramenta.

Quanto à capacidade do Rupyly em facilitar o ensino de Arduino, 7 dos 9 participantes responderam de forma afirmativa. As respostas indicam que a proposta da plataforma contribui para tornar os conceitos relacionados ao microcontrolador mais acessíveis a iniciantes.

No que se refere ao ensino de programação, 8 dos 9 participantes consideraram que a plataforma é eficaz para esse fim. Um dos usuários, ao marcar a opção “talvez”, justificou que a ferramenta seria mais indicada para o treino e a fixação de conceitos previamente aprendidos, ao invés de um ensino introdutório completo.

Alguns comentários também apontaram dificuldades específicas enfrentadas durante o uso, como a confusão causada pelo fato de alguns blocos já possuírem nomes ou instruções parcialmente definidas. Essa característica, segundo os participantes, poderia dificultar a compreensão do funcionamento dos blocos por parte de usuários iniciantes.

Essas observações contribuíram diretamente para o planejamento das melhorias que seriam implementadas na versão seguinte da plataforma, com foco na clareza dos blocos, reforço pedagógico e aprimoramento da experiência do usuário.

5.6.2 Segunda Versão

Na segunda rodada de testes, com a versão atualizada da plataforma, observou-se uma evolução positiva nas percepções dos participantes quanto ao uso do Rupyly em contextos educacionais. Dos 13 participantes, 11 afirmaram que utilizariam a ferramenta em atividades de ensino de programação e robótica, enquanto os 2 restantes marcaram a opção "talvez".

Quando questionados sobre a capacidade da plataforma de facilitar o ensino de Arduino, 12 participantes responderam afirmativamente. Apenas um marcou a opção "talvez", justificando que, por o código ser gerado automaticamente, a compreensão do funcionamento interno do Arduino poderia não ser plenamente explorada pelos alunos, o que reduziria o aprendizado em termos mais técnicos.

Quanto à eficácia da plataforma no ensino de conceitos de programação, 10 dos 13 participantes consideraram a ferramenta eficiente, enquanto 3 optaram por "talvez". As justificativas apresentadas indicam que, para o público infantil ou iniciante, seria necessário um maior nível de simplificação e acompanhamento. Alguns destacaram que o uso da ferramenta seria mais efetivo se houvesse um tutor ou mediador explicando o funcionamento dos blocos e orientando os alunos durante a experiência de uso.

Ainda assim, os participantes reconhecem que o Rupyly é eficaz para a introdução e o reforço de conceitos básicos de programação, funcionando como uma ponte entre o pensamento computacional e a prática com Arduino.

Essas observações refletem avanços em relação à primeira versão e reforçam o potencial do sistema como recurso didático, sobretudo quando utilizado em ambientes mediados por educadores ou em oficinas orientadas.

Este capítulo detalhou a evolução do projeto Rupyly através de uma análise aprofundada dos resultados obtidos em suas fases de desenvolvimento e teste. Observou-se uma melhora significativa na experiência do usuário da primeira para a segunda versão da plataforma, especialmente na intuitividade da interface e na organização dos blocos. O e-book foi validado como um recurso de apoio eficaz, enquanto as bibliotecas para os robôs Erekopapa e Arabeko demonstraram grande potencial pedagógico, apesar de necessitarem de ajustes pontuais para otimizar a clareza e a acessibilidade. As avaliações reforçam o potencial do Rupyly como ferramenta didática para o ensino de programação e Arduino, especialmente em ambientes mediados, apontando o caminho para refinamentos futuros.

Capítulo 6

Considerações finais

6.1 Conclusões

Este trabalho abordou o desafio de tornar a programação e a robótica acessíveis a iniciantes, propondo o desenvolvimento do Rupyly, um ambiente de programação em blocos que integra software e hardware através da plataforma Arduino. A iniciativa buscou combinar a facilidade inerente às linguagens de programação visual (VPLs) com a experiência prática e motivadora do controle de dispositivos eletrônicos. O resultado é uma ferramenta educacional robusta, ideal para contextos de ensino e formação inicial.

A concepção do Rupyly foi fundamentada em ferramentas consolidadas como Blockly para a interface de programação em blocos e Arduino CLI para a interação com o hardware, permitindo um fluxo contínuo desde a montagem lógica do código até sua compilação e upload direto para a placa Arduino. Essa arquitetura integra eficientemente a abstração visual com a aplicação prática, um ponto crucial para a aprendizagem de conceitos computacionais.

Ao longo de seu desenvolvimento e aplicação, o Rupyly demonstrou contribuições significativas para o ensino de lógica de programação e para o estímulo à aprendizagem ativa por meio da robótica. A abordagem baseada em blocos revelou-se altamente eficaz na redução de barreiras sintáticas e cognitivas frequentemente encontradas por iniciantes em linguagens textuais, promovendo um ambiente de aprendizado mais inclusivo e motivador. A modularidade das bibliotecas internas (Erekopapa e Arabeko), com níveis de complexidade adaptáveis, facilitou a progressão do usuário, enquanto funcionalidades como o *live coding* e a geração automática de código em C permitiram não só a visualização imediata da programação textual, mas também atuaram como uma ponte essencial para a compreensão e futura transição a esse tipo de escrita de código.

Os testes de usabilidade com usuários validaram a eficácia da plataforma, evidenciando sua interface intuitiva, a organização lógica dos blocos e a clareza do material de apoio,

como o e-book “Rupyly: Programando em Blocos”. As melhorias implementadas entre as versões iniciais e a final do projeto, incluindo a simplificação da linguagem nos blocos, a adição de descrições funcionais detalhadas e a reestruturação das bibliotecas, foram cruciais para aprimorar a experiência do usuário e reforçar a aceitação da ferramenta em ambientes educacionais. A integração fluida com o Arduino CLI consolidou o Rupyly como uma solução prática e completa para oficinas de robótica e aulas de programação, permitindo que os alunos se concentrem na lógica e na criatividade.

Em síntese, o Rupyly representa uma inovação pedagógica que aproveita o potencial das VPLs para tornar a robótica educacional mais acessível e impactante. Ele não apenas facilita os primeiros passos na programação, mas também fomenta o pensamento computacional, a resolução de problemas e o desenvolvimento de competências essenciais para a formação de indivíduos ativos e criativos na era digital.

6.2 Trabalhos Futuros

O desenvolvimento do Rupyly representa um avanço significativo na criação de ambientes acessíveis para o ensino de programação e robótica educacional. Contudo, o campo de atuação e o potencial de aprimoramento da plataforma são vastos. Para consolidar e expandir a relevância do Rupyly, sugerem-se as seguintes direções para trabalhos futuros:

- Aprimoramento e Manutenção Contínua do Software:
  - Otimização de Desempenho: Realizar análises de desempenho para otimizar o tempo de compilação e upload para a placa Arduino, especialmente para projetos mais complexos e em ambientes com recursos computacionais limitados.
  - Refatoração de Código: Investir na refatoração do código-fonte das bibliotecas Erekopapa e Arabeko, buscando maior modularidade, legibilidade e manutenibilidade, o que facilitaria a contribuição de outros desenvolvedores e a adição de novas funcionalidades.
  - Melhorias na Interface do Usuário (UI): Implementar funcionalidades avançadas na interface, como ferramentas de depuração visual (visualização do estado das variáveis, passo a passo da execução do código), histórico de projetos e opções personalizáveis para o ambiente de trabalho.
- Expansão da Biblioteca de Blocos e Funções:
  - Novos Componentes e Sensores: Desenvolver e integrar novos conjuntos de blocos que suportem uma gama mais ampla de sensores (ultrassom, temperatura, luz, cor) e atuadores (motores de passo, servos mais complexos, displays

LCD), permitindo a criação de robôs com funcionalidades mais diversificadas e complexas.

  - Funções de Comunicação: Adicionar blocos para comunicação sem fio (Wi-Fi), possibilitando a interação entre múltiplos robôs ou com outros dispositivos inteligentes, e a criação de projetos de Internet das Coisas (IoT).
  - Blocos para Lógica Avançada: Introduzir blocos para estruturas de dados mais complexas (arrays, listas, structs) e algoritmos avançados, facilitando a transição para conceitos de programação de nível superior.

- Realização de Testes Abrangentes e Oficinas Pedagógicas:

  - Estudos de Longo Prazo: Conduzir estudos de caso e pesquisas longitudinais em ambientes educacionais diversos (escolas públicas e privadas, oficinas), avaliando o impacto do Rupyly no desenvolvimento do pensamento computacional, na motivação para a aprendizagem e na transição para linguagens textuais ao longo de um período prolongado.
  - Testes com Diferentes Faixas Etárias: Expandir os testes para incluir outras faixas etárias, como adolescentes mais velhos ou adultos iniciantes, a fim de avaliar a adaptabilidade da ferramenta a diferentes níveis de maturidade cognitiva e experiência prévia.
  - Workshops e Formação de Educadores: Organizar e documentar oficinas de formação para professores, capacitando-os a utilizar o Rupyly de forma eficaz em suas práticas pedagógicas e a criar materiais didáticos complementares. Coletar feedback desses educadores será crucial para aprimorar a plataforma.

- Integração e Ecossistema

  - Portabilidade para Outras Plataformas: Explorar a possibilidade de adaptar o Rupyly para outras plataformas de hardware além do Arduino (ESP32, micro:bit, Raspberry Pi), ampliando seu alcance e sua aplicabilidade em diferentes projetos de robótica e eletrônica.
  - Material Didático Complementar: Desenvolver uma coleção abrangente de tutoriais, exemplos de projetos e desafios gradualmente mais complexos, que possam ser utilizados por professores e alunos para explorar todas as funcionalidades do Rupyly.

Essas direções futuras não apenas aprimorarão o Rupyly como ferramenta educacional, mas também contribuirão para a pesquisa sobre a eficácia das linguagens de programação

visual e da robótica educacional na formação de cidadãos críticos, criativos e tecnologicamente alfabetizados.

Referências

[1] Pereira, Filipe D, Elaine Oliveira, Alexandra Cristea, David Fernandes, Luciano Silva, Gene Aguiar, Ahmed Alamri e Mohammad Alshehri: *Early dropout prediction for programming courses supported by online judges*. Em *International Conference on Artificial Intelligence in Education*, páginas 67–72. Springer, 2019. 1

[2] Lima, Marcos, Leandro Silva Galvão Carvalho, Elaine Harada Teixeira de Oliveira, David Braga Fernandes Oliveira e Filipe Dwan Pereira: *Classificação de dificuldade de questões de programação com base em métricas de código*. Em *Anais do XXXI Simpósio Brasileiro de Informática na Educação*, páginas 1323–1332. SBC, 2020. 1

[3] Araujo, Ada, Daniel Lopes Zordan Filho, Elaine Harada Teixeira Oliveira, Leandro Silva Galvão Carvalho, Filipe Dwan Pereira e David Braga Fernandes Oliveira: *Mapeamento e análise empírica de misconceptions comuns em avaliações de introdução à programação*. Em *Anais do Simpósio Brasileiro de Educação em Computação*, páginas 123–131. SBC, 2021. 1

[4] Fonseca, Samuel, Elaine Oliveira, Filipe Pereira, David Fernandes e Leandro Silva Galvão Carvalho: *Adaptação de um método preditivo para inferir o desempenho de alunos de programação*. Em *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 30, página 1651, 2019. 1

[5] Luxton-Reilly, Andrew, Ibrahim Albluwi, Brett A Becker, Michail Giannakos, Amruth N Kumar, Linda Ott, James Paterson, Michael James Scott, Judy Sheard e Claudia Szabo: *Introductory programming: a systematic literature review*. Em *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, páginas 55–106, 2018. 1

[6] Souza, Raul Porto De: *Uso da biblioteca de programação em blocos blockly como forma de auxílio ao aprendizado da disciplina de algoritmos e programação utilizando a linguagem c*. 2018. Disponível em`http://repositorio.unesc.net/handle/1/8169`(Março 2024). 1

[7] Saleiro, Mário, Bruna Carmo, Joao MF Rodrigues e JM Hans du Buf: *A low-cost classroom-oriented educational robotics system*. Em *Social Robotics: 5th International Conference, ICSR 2013, Bristol, UK, October 27-29, 2013, Proceedings 5*, páginas 74–83. Springer, 2013. 1

[8] Iturrate, Iñigo, Gustavo Martín, Javier García-Zubia, Ignacio Angulo, Olga Dziabenko, Pablo Orduña, Gustavo Alves e André Fidalgo: *A mobile robot platform for*

open learning based on serious games and remote laboratories. Em *2013 1st International Conference of the Portuguese Society for Engineering Education (CISPEE)*, páginas 1–7. IEEE, 2013. 1

[9] Heinen, Eduarth: *Raspiblocos: ambiente de programação didático baseado em raspberry pi e blockly*. B.S. thesis, Universidade Tecnológica Federal do Paraná, 2015. 1

[10] Pasternak, Erik: *Visual programming pedagogies and integrating current visual programming language features*. Tese de Mestrado, Carnegie Mellon University, Pittsburgh, PA, August 2009. 1

[11] Ribas, Elisângela, Guilherme Dal Bianco e Regis Alexandre Lahm: *Programação visual para introdução ao ensino de programação na educação superior: uma análise prática.* RENOTE. Revista Novas Tecnologias na Educação, 2016. 1

[12] Google Developers: *Blockly — A visual programming library*, 2024. `https://developers.google.com/blockly`, Acesso em: Março 2024. 1, 18, 19

[13] Braz, Ana Caroline, Marcus Oliveira, Batista, Carla Koike, Dianne Viana e Jones Silva: *Enhancing learning with block programming in educational robots*. 14:490–497, 2024, ISSN 2183-1378. International Symposium on Project Approaches in Engineering Education. 2

[14] Braz, Ana Caroline, Marcus Oliveira, Batista, Carla Koike, Dianne Viana e Jones Silva: *O uso da programação em blocos para modelos educacionais*. janeiro 2024. XII Congresso Nacional de Engenharia Mecânica. 2

[15] Resnick, Mitchel, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jeffrey Silver, Brian Silverman *et al.*: *Scratch: programming for all*. Communications of the ACM, 52(11):60–67, 2009. 5, 19

[16] *Lego mindstomrs-ev3*. `https://education.lego.com/en-us/downloads/mindstorms-ev3/software/`. Acesso em: Junho 2025. 5

[17] WHITLEY, K.N.: *Visual programming languages and the empirical evidence for and against*. Journal of Visual Languages Computing, 8(1):109–142, 1997, ISSN 1045-926X. `https://www.sciencedirect.com/science/article/pii/S1045926X96900300`. 5

[18] Day, Ruth S.: *Alternative representations*. Volume 22 de *Psychology of Learning and Motivation*, páginas 261–305. Academic Press, 1988. `https://www.sciencedirect.com/science/article/pii/S0079742108600432`. 5

[19] Pandey, Rajeev K. e Margaret M. Burnett: *Is it easier to write matrix manipulation programs visually or textually? an empirical study*. páginas 344–351, 1993. Proceedings of the 2015 International Conference on Soft Computing and Software Engineering (SCSE'15). 5

[20] Green, T.R.G. e M. Petre: *Usability analysis of visual programming environments: A 'cognitive dimensions' framework*. Journal of Visual Languages Computing, 7(2):131–174, 1996, ISSN 1045-926X. https://www.sciencedirect.com/science/article/pii/S1045926X96900099. 5

[21] Strong, Glenn, Nina Bresnihan e Brendan Tangney: *Supporting learners in the transition from block-based to text-based programming, a systematic review*. Journal of Computer Languages, página 101342, 2025, ISSN 2590-1184. https://www.sciencedirect.com/science/article/pii/S2590118425000280. 6

[22] Soomro, Abdul Hameed: *An exploration of visual and textual based programming languages: A comparative analysis*. International Journal of Electrical Engineering amp;amp; Emerging Technology, 6(1):20–23, Jun. 2023. https://ijeeet.com/index.php/ijeeet/article/view/140. 6

[23] Lin, Yuhan e David Weintrop: *The landscape of block-based programming: Characteristics of block-based environments and how they support the transition to text-based programming*. Journal of Computer Languages, 67:101075, 2021, ISSN 2590-1184. https://www.sciencedirect.com/science/article/pii/S259011842100054X. 7

[24] Kurihara, Azusa, Akira Sasaki, Ken Wakita e Hiroshi Hosobe: *A programming environment for visual block-based domain-specific languages*. Procedia Computer Science, 62:287–296, 2015, ISSN 1877-0509. https://www.sciencedirect.com/science/article/pii/S1877050915025879, Proceedings of the 2015 International Conference on Soft Computing and Software Engineering (SCSE'15). 7

[25] Seymout, A e Solomon Cynthia: *Twenty thing to do with a computer*, 1971. Disponível em https://dspace.mit.edu/handle/1721.1/5836 (acessado em 13 de setembro). 8

[26] Braz, Raíza de Souza: *Robótica educacional e o desenvolvimento do pensamento computacional na educação básica: mapeamento sistemático da literatura*. 2021. 8, 9

[27] Gatica Zapata, Nibaldo, Miguel Ripoll Novales e J. Valdivia Guzmán: *La robótica educativa como herramienta de apoyo pedagógico.*, 2004. Disponível em https://pt.slideshare.net/CarlosCarvajalSegovi/roboticaeducativa-54279925 (acessado em 13 de setembro). 8

[28] Almeida, Carlos Manuel dos Santos: *A importância da aprendizagem da robótica no desenvolvimento do pensamento computacional: um estudo com alunos do 4º ano*. Tese de Doutoramento, 2015. 8, 9

[29] Ribeiro, Célia Rosa, Clara Pereira Coutinho e Manuel FM Costa: *A robótica educativa como ferramenta pedagógica na resolução de problemas de matemática no ensino básico*. 2011. 9

[30] Zilli, Silvana do Rocio: *A robótica educacional no ensino fundamental: Perspectivas e prática*. Dissertação (mestrado em engenharia de produção), Universidade Federal de Santa Catarina, 2004. 9

[31] Silva, Alzira Ferreira da: *RoboEduc: Uma Metodologia de Aprendizado com Robótica Educacional*. Tese de Doutoramento, Universidade Federal do Rio Grande do Norte, Natal, RN, 2009. Tese de Doutorado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica. 9

[32] *Grupo ereko*, 2025. `dgp.cnpq.br/dgp/espelhogrupo/2451965780752898`, Acesso em: Agosto 2025. 11

[33] Koike, Carla M. C. C., Dianne M. Viana, Jones Y. M. A. Silva, Aleteia Patricia F. A. V. Paumgartten, Maristela T. Holanda, Marcus J. A. de Oliveira, Filipe A. Batista, Ana Caroline R. Braz, Maria Luiza R. Sousa, Ana Cecilia C. Olinda, Andressa A. Pereira e Danielly R. dos Santos: *Erekopapa*, 2024. Depositante: Fundação Universidade de Brasília. n. BR3020240068492. Depósito: 04 out. 2024. Concessão: 12 dez. 2024. 12

[34] Oliveira, Marcus, Filipe Batista, Isabelle Aragão, Carla Koike, Dianne Viana e Jones Silva: *EducaÇÃo stem interativa no desenvolvimento de um kit de robÓtica mÓvel*. 62:287–296, 2024. XII Congresso Nacional de engenharia mecânica. 13

[35] Oliveira, Marcus, Filipe Batista, Isabelle Aragão, Carla Koike, Dianne Viana e Jones Silva: *Introducing stem projects in multidisciplinary teacher training*. 13:468–476, 2023, ISSN 2183-1378. International Symposium on Project Approaches in Engineering Education. 13

[36] Reichert Costa, Arthur, FILIPE Batista, Larissa Santos, Marcus Jessé Oliveira, Dianne Viana, JONES Silva, EMILLY Cornelio, Fernanda Silva e Carla Koike: *Tutoria e mÚltiplas abordagens em oficina de modelagem e impressÃo 3d*. dezembro 2020. XLVIII Congresso Brasileiro de Educação em engenharia e III Simpósio Internacional de Educação em Engenharia da ABENGE. 13

[37] Oliveira, Marcus, Filipe Batista, Isabelle Aragão, Carla Koike, Dianne Viana e Jones Silva: *Active learning workshops production: Impacts and benefits for engineering students*. 10:206–213, 2020, ISSN 2183-1378. International Symposium on Project Approaches in Engineering Education. 13

[38] Reichert Costa, Arthur, FILIPE Batista, Larissa Santos, Marcus Jessé Oliveira, Dianne Viana, JONES Silva, EMILLY Cornelio, Fernanda Silva e Carla Koike: *Capacitação de tutores para estratégias de aprendizagem ativa nas áreas de stem*. dezembro 2020. XLVIII Congresso Brasileiro de Educação em engenharia e III Simpósio Internacional de Educação em Engenharia da ABENGE. 13

[39] Evans, Martin, Joshua Noble e Jordan Hochenbaum: *Arduino em ação*. Novatec Editora, 2013. 14, 15, 16

[40] BARBOSA, José Wilian: *Sistema de irrigação automatizado utilizando a plataforma arduino*. Trabalho de conclusão de curso, Fundação Educacional do Município de Assis–FEMA-Assis, 2013. 14, 15, 16

[41] Moreira, Michele Paulino Carneiro, Mairton Cavalcante Romeu, Francisco Regis Vieira Alves e Francisco Roberto Oliveira da Silva: *Contribuições do arduino no ensino de física: uma revisão sistemática de publicações na área do ensino*. Caderno Brasileiro de Ensino de Física, 35(3):721–745, 2018. 14

[42] Developers, Google: *Arduino-cli*, 2025. Disponível em `https://docs.arduino.cc/arduino-cli/` (Junho 2025). 15

[43] Developers, Google: *Arduino uno*, 2025. Disponível em `https://docs.arduino.cc/hardware/uno-rev3/` (Junho 2025). 15

[44] Developers, Google: *Ide do arduino*, 2025. Disponível em `https://docs.arduino.cc/software/ide-v1/tutorials/Environment/` (Junho 2025). 16

[45] Developers, Google: *Integrações do arduino-cli*, 2025. Disponível em `https://docs.arduino.cc/arduino-cli/integration-options/` (Junho 2025). 17, 18

[46] Developers, Google: *Arduino-cli github*, 2020. Disponível em `https://github.com/arduino/arduino-cli` (Junho 2025). 17, 18

[47] *Blockly games*. `https://blockly.games/`. Acesso em: Junho 2025. 19

[48] *Tinkercad*. `https://www.tinkercad.com/`. Acesso em:Junho 2025. 20

[49] *Mit app inventor*. `https://appinventor.mit.edu/`. Acesso em: Junho 2025. 20

[50] *mblock*. `https://www.mblock.cc/`. Acesso em: Junho 2025. 20

[51] Atencio, Carlos Pereira: *ardublockly*, 2016. Disponível em `https://github.com/carlosperate/ardublockly/tree/master` (Abril 2025). 21, 35

[52] Canet, Sébastien e Fred Lin: *Blocklyduino*, 2020. Disponível em `https://github.com/BlocklyDuino/BlocklyDuino-v2?tab=readme-ov-file` (Abril 2025). 21, 35

[53] College, Technologies: *Blockly at rduino*, 2020. Disponível em `https://github.com/technologiescollege/Blockly-at-rduino` (Junho 2025). 21

[54] *Rupy - dialeto guarani*, 2025. `https://www.perplexity.ai/search/verificar-se-rupy-pode-ser-uma-sJsfSvnmQgeSs7MI9WB4gA`, Acesso em: Agosto 2025. 23

[55] Developers, Google: *Blockly factory*, 2024. Disponível em `https://developers.google.com/blockly?hl=pt-br` (Março 2024). 33

[56] Forward, Andrew: *Software documentation: Building and maintaining artefacts of communication.* University of Ottawa (Canada), 2002. 39

[57] Coelho, Hilda Simone: *Documentação de software: uma necessidade.* Texto Livre: linguagem e tecnologia, 2(1):17–21, 2009. 39

[58] Booch, Grady: *UML: guia do usuário.* Elsevier Brasil, 2006. 39

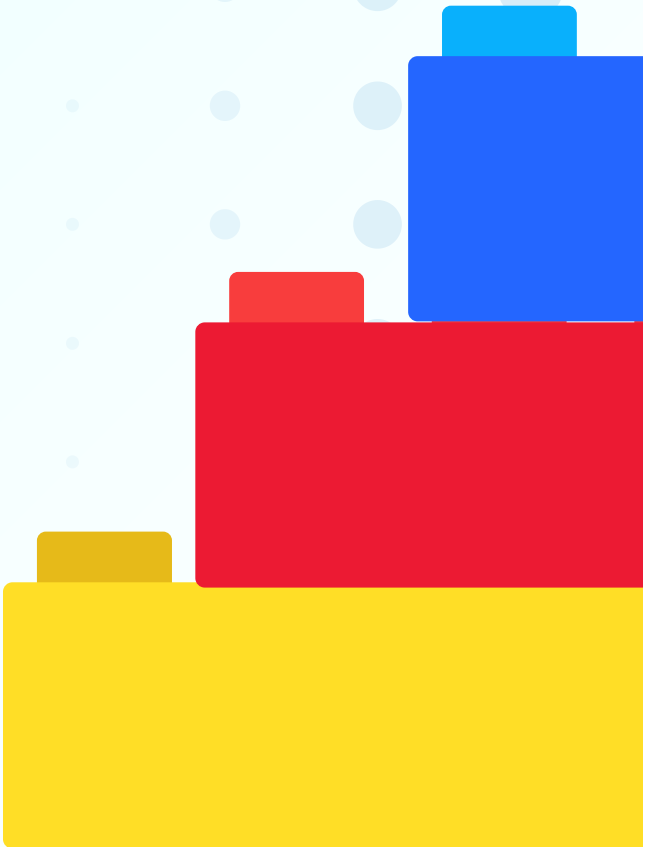
Apêndice A

E-book: Rupyly Programando em Blocos

PUPYLY

Programando em Blocos

Ana Caroline da Rocha Braz
Carla Koike



UnB

EREKO
ROBÓTICA NA EDUCAÇÃO

.RUPYLY

Programando em Blocos

Universidade de Brasília

Instituto de Ciência Exatas

Rupyly - Programando em Blocos

Publicado

Brasília - Distrito Federal

Autoras

Ana Caroline da Rocha Braz

Graduanda em Ciência da Computação na Universidade de Brasília

Carla Cavalcante Koike

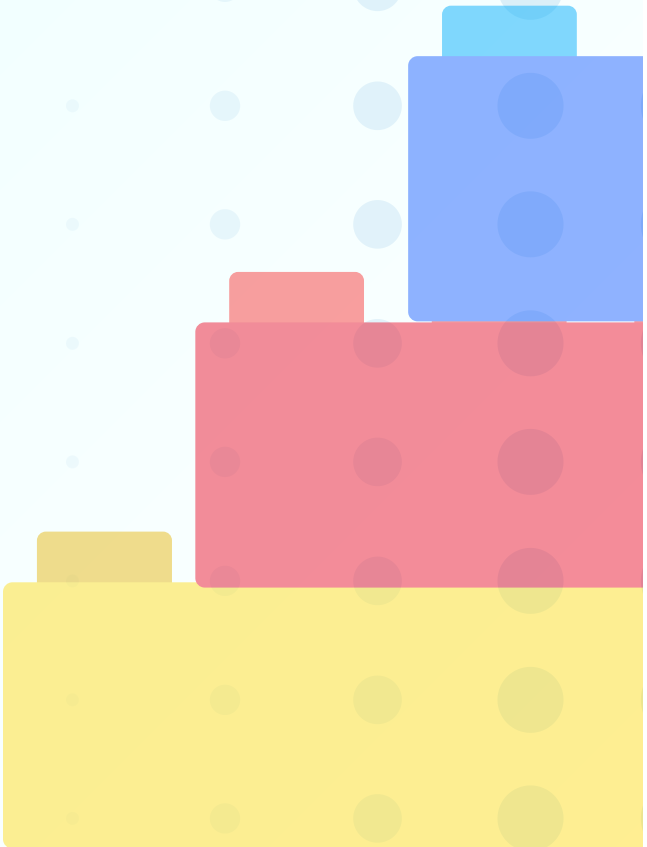
Professora no Departamento de Ciência da Computação da Universidade de Brasília

Arte e Ilustrações

Elementos visuais fornecidos por Canva e outras ilustrações são de autoria própria.

Como citar este livro

BRAZ, Ana C. R., KOIKE, Carla C., .Rupyly Programando em blocos. UnB, Brasília, DF, 2025.



SOBRE A AUTORA

Graduanda em Ciência da Computação pela Universidade de Brasília, onde participa ativamente do projeto de extensão Meninas.comp. Entusiasta pela robótica e pela educação que impulsionam a explorar novas fronteiras e desafios.

SOBRE O LIVRO

Este livro foi desenvolvido para apresentar o software Rupyly a novos usuários e guiá-los na construção de blocos e códigos de forma prática e intuitiva.

Aqui, você encontrará:

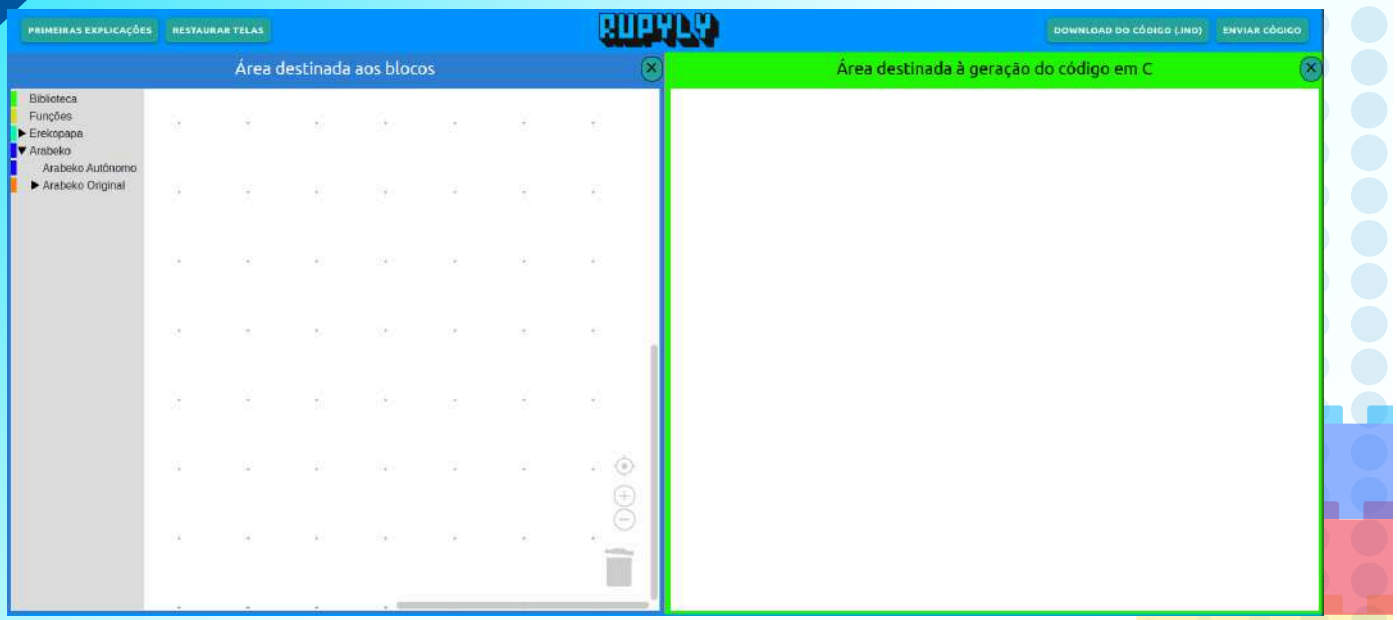
- Uma introdução detalhada à interface do Rupyly
- Explicações completas sobre cada bloco disponível
- Exemplos de código para facilitar o aprendizado
- Os modelos educacionais que serviram de base para o desenvolvimento das bibliotecas do software

Seja bem-vindo a essa jornada de aprendizado e exploração com o Rupyly. Espero que você aproveite ao máximo essa experiência.

Boa leitura e divirta-se!

SUMÁRIO

INTERFACE


PRIMEIRAS EXPLICAÇÕES
RESTAURAR TELAS
RUPYLY
DOWNLOAD DO CÓDIGO (.INO)
ENVIAR CÓGIGO
Área destinada aos blocos
Área destinada à geração do código em C
Biblioteca
Funções
Erekopapa
Arabeko
Arabeko Autônomo
Arabeko Original

INTERFACE

A interface do Rupyly foi pensada para ser simples e intuitiva.
Na barra superior, há quatro botões, dois de cada lado, com o nome do software centralizado.
Eles facilitam o acesso rápido às principais funções.

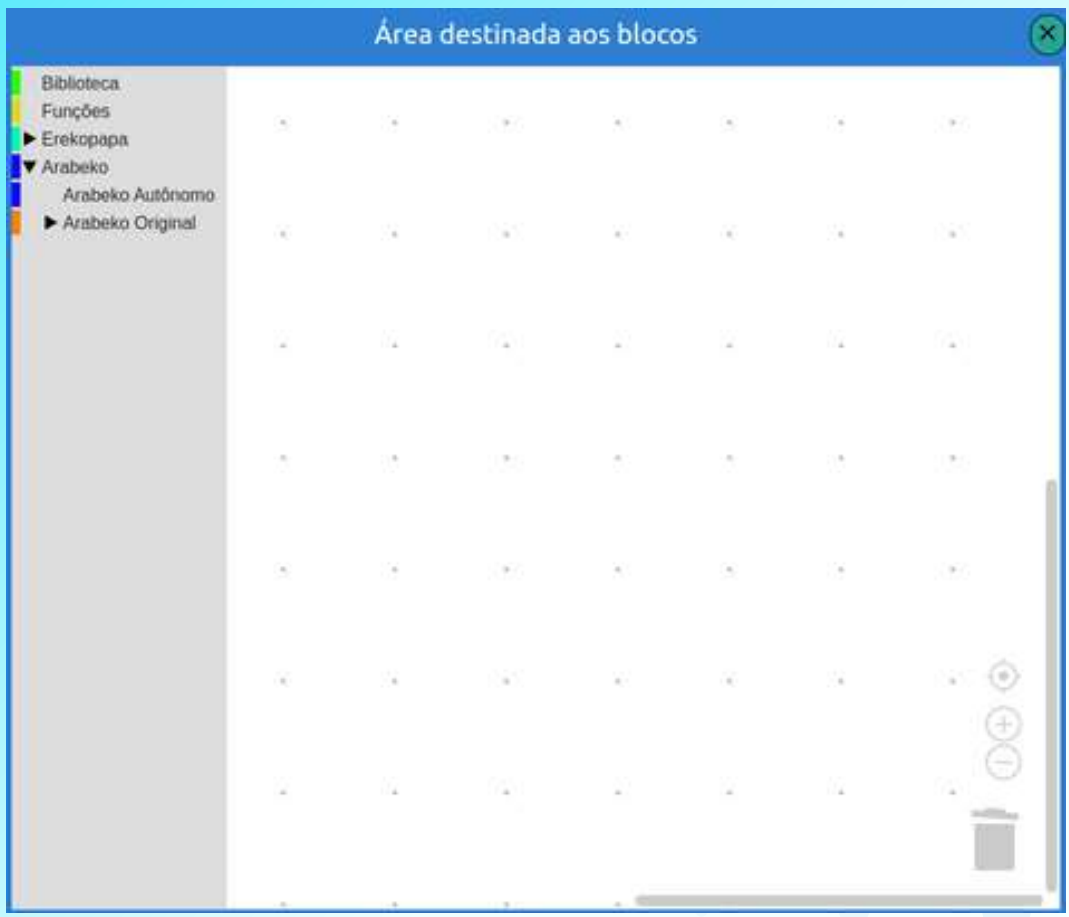
À esquerda da barra superior, estão os botões:

- Primeiras Explicações: permite baixar este e-book com as instruções iniciais sobre o software;
- Restaurar Telas: usado para reabrir as áreas de blocos ou código caso tenham sido fechadas.

À direita da barra superior, estão os botões:

- Download do Código (.ino): baixa o código gerado pelos blocos no formato .ino, pronto para uso na IDE do Arduino;
- Enviar Código: envia o código diretamente para a placa Arduino conectada.

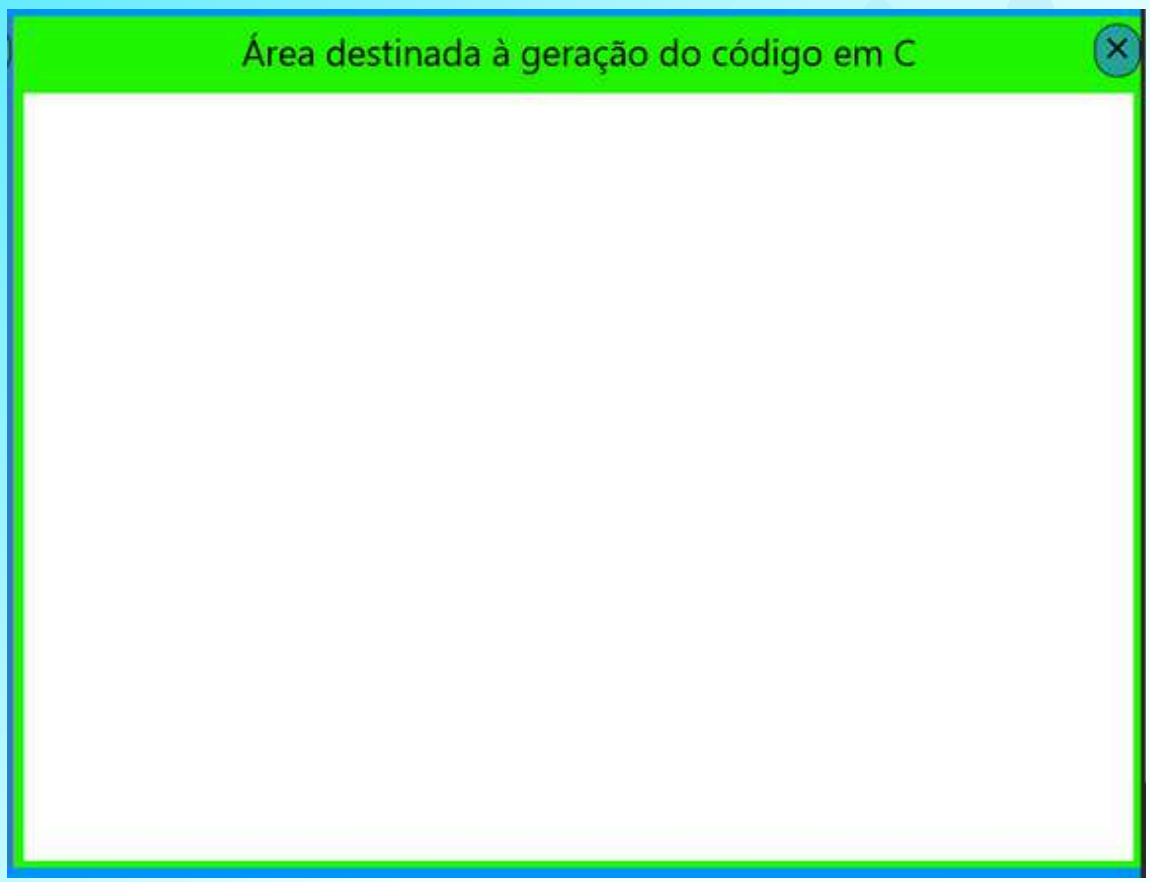
INTERFACE



A área esquerda é o espaço de trabalho principal, onde você cria sua programação em blocos. Nela, você encontra:

- Botão de Fechar: Oculta temporariamente a área para otimizar o espaço da tela;
- Bibliotecas de Blocos: Blocos organizados por modelos educacionais para construção do código;
- Área de Trabalho: Espaço em branco para arrastar, soltar e conectar blocos visualmente;
- Botões de Zoom: Ampliam ou reduzem a visualização da área para melhor organização;
- Lixeira: Permite excluir blocos indesejados, mantendo o espaço de trabalho limpo.

INTERFACE



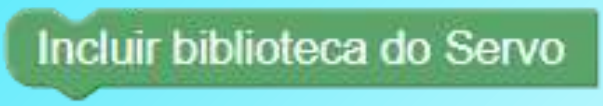
A segunda área, à direita, é onde o código em C é gerado automaticamente. Enquanto você monta os blocos, o código correspondente surge instantaneamente, facilitando a compreensão da lógica sem a preocupação com a sintaxe. Há também um botão para ocultar essa seção, caso prefira focar apenas nos blocos.

BIBLIOTECAS

As bibliotecas estão localizadas na área de blocos, exibindo os elementos relacionados ao tema de cada seção ao serem clicadas. As principais são:

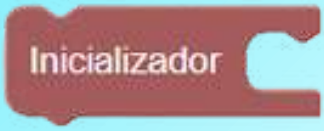
- Biblioteca: Blocos para inclusão de bibliotecas essenciais (Motor, Serial);
- Funções: Blocos para criação de funções e as principais funções do Arduino;
- Erekopapa: Blocos exclusivos para o funcionamento do robô Erekopapa;
- Arabeko Autônomo: Blocos para o Arabeko operar autonomamente (movimentos pré-definidos);
- Arabeko Original: Todos os blocos para o usuário programar livremente os movimentos do Arabeko.

BLOCOS



- Carrega a biblioteca que fornece os comandos para operar o servo motor.

- Carrega a biblioteca que simula uma porta serial em pinos digitais do Arduino.

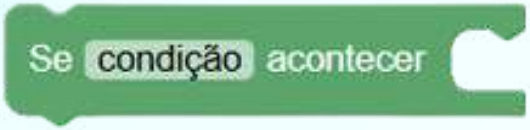


- Configura pinos, comunicação serial e outros parâmetros iniciais.

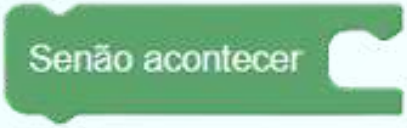
- Função principal que é executada repetidamente enquanto o Arduino estiver ligado.

- Declara uma função personalizada com parâmetros para reutilizar trechos de código.

- Usa a função criada, enviando os valores que ela precisa para funcionar.



- Verifica se a condição é verdadeira e executa o bloco de código.



- Se a condição do bloco anterior não for verdadeira, esse outro bloco é executado.

BLOCOS

- Define o pino ao qual o LED está conectado.

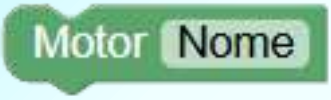
- Diz ao Arduino se o pino vai receber ou enviar sinais.

- Faz o pino ficar ligado (HIGH) ou desligado (LOW).

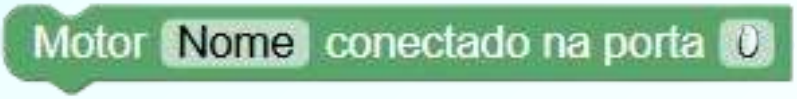
- Define o pino onde o buzzer será controlado.

- Diz ao Arduino se o pino vai receber ou enviar sinais.

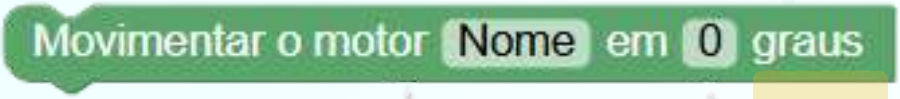
- Faz o pino ficar ligado (HIGH) ou desligado (LOW).



- Declara o nome do motor.



- Define o pino usado para controlar o motor.

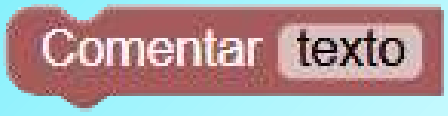


- Define a posição do servo em graus (geralmente de 0 a 180).

BLOCOS

- Inicia a comunicação serial entre o Arduino e o computador.

- Interrompe a execução do programa pelo tempo especificado (1s = 1000ms).



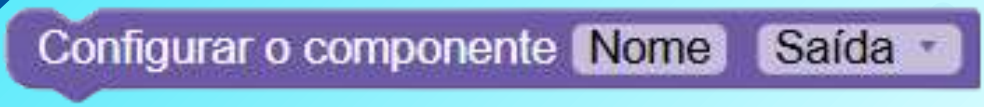
- Usado para fazer comentários no código.

- Bloco exclusivo do arabeko fazendo andar em quadrado.

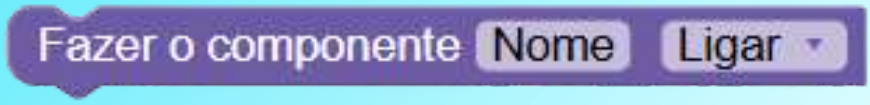
- Bloco exclusivo do arabeki fazendo andar em círculo.

- Bloco exclusivo do arabeko fazendo andar apenas para frente.

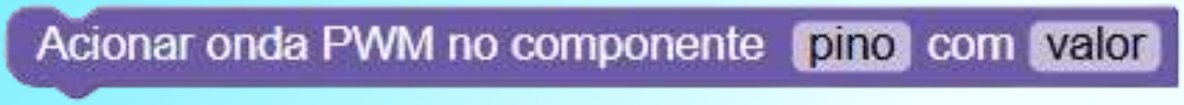
BLOCOS



- Diz ao Arduino se o pino vai receber ou enviar sinais.

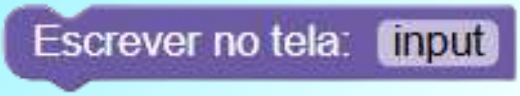


- Faz o pino ficar ligado (HIGH) ou desligado (LOW).



- Envia um sinal analógico (PWM) a um pino para controlar intensidade ou velocidade.

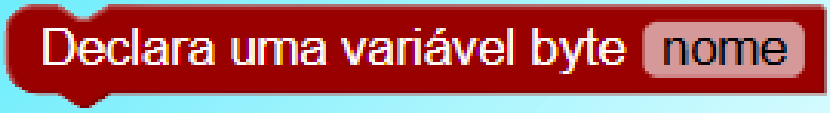
- Cria uma nova porta serial usando dois pinos digitais (RX e TX).



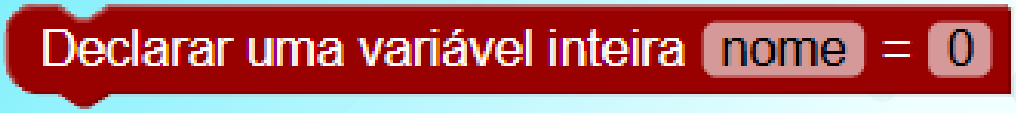
- Escreve algo na tela do computador, mas tudo junto na mesma linha.

- Escreve algo na tela do computador e pula para a próxima linha.

BLOCOS



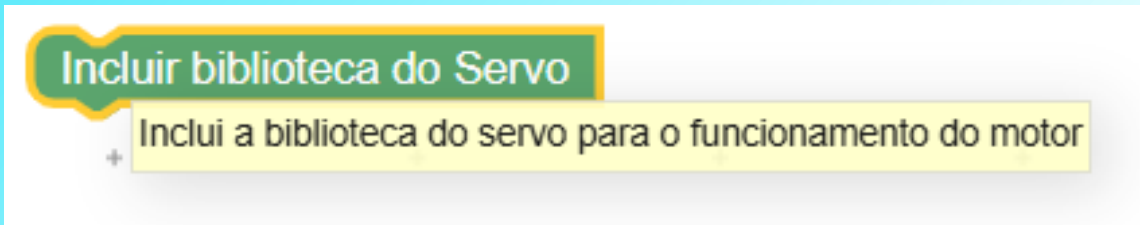
- Cria uma variável que pode guardar um número entre 0 e 255.



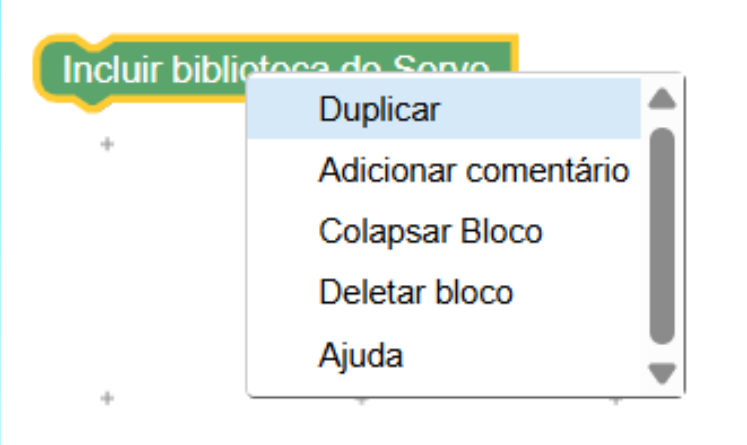
- Cria uma variável inteira e define seu valor inicial.

- Atribui o conteúdo de texto à variável.

BLOCOS



- Ao passar o mouse sobre um bloco, uma breve descrição será exibida, explicando sua função.



Ao clicar com o botão direito do mouse sobre o bloco, cinco opções ficam visíveis. São elas:

- Duplicar: cria uma cópia idêntica do bloco selecionado;
- Adicionar comentário: permite inserir uma anotação explicativa no bloco;
- Colapsar bloco: reduz visualmente o bloco, ocupando menos espaço na área de trabalho;
- Deletar bloco: remove o bloco selecionado da área de montagem;
- Ajuda: direciona para o site oficial do Arduino com explicações mais detalhadas sobre a função correspondente ao bloco.

COMO MONTAR O EREKOPAPA

- O Erekopapa é um robô simples usado para ensinar conceitos básicos de robótica, como programação com Arduino, montagem de circuitos em protoboard e introdução à modelagem e impressão 3D para fabricação das peças.

- Ele é composto por 2 partes:
  - Cabeça - onde contém 2 LEDs que representam os olhos e um buzzer que faz o papel da boca
  - Corpo - que suporta um servomotor e a cabeça, conectada ao eixo do servomotor

Erekopapa. INPI nº BR302024006849 2

- Durante as oficinas, os participantes podem controlar a rotação da cabeça, fazer os LEDs piscarem e ativar sons no buzzer, seja por meio de um botão ou automaticamente, conforme a programação escolhida.

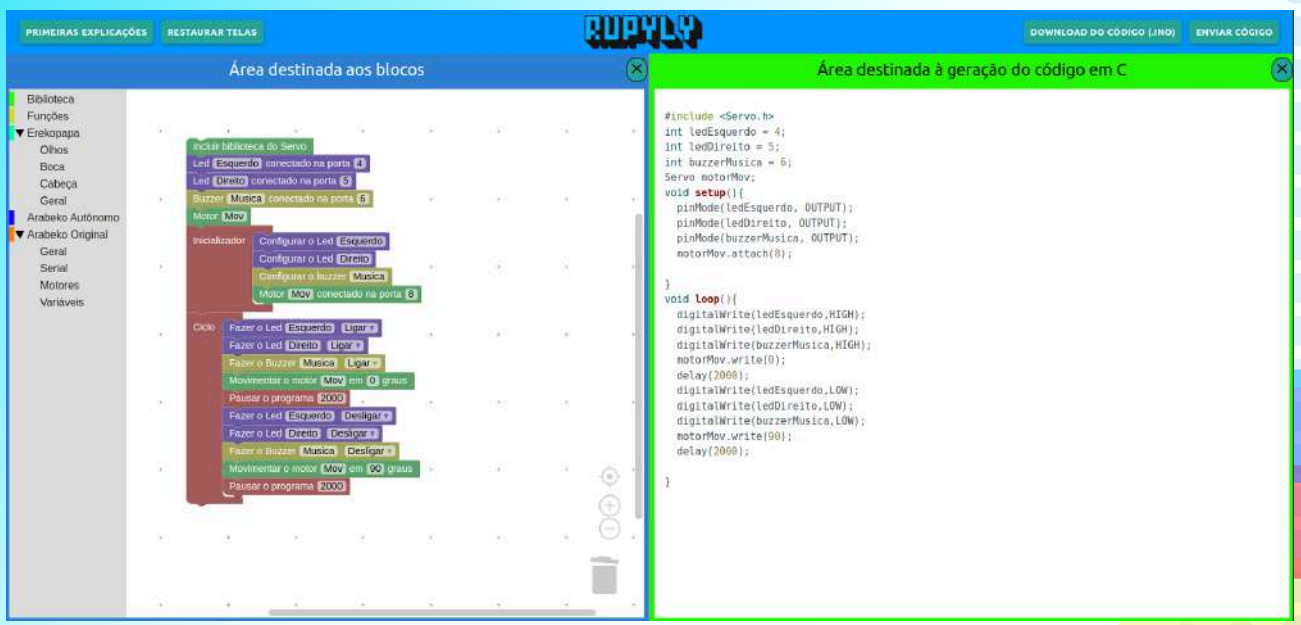
COMO MONTAR O EREKOPAPA

- Para programar o Erekopapa, vamos usar algumas bibliotecas, sendo elas:
  - Biblioteca do Servomotor:
    - Essa biblioteca traz todos os comandos para controlar o servomotor, que é o motor responsável por mover a cabeça do robô.
  - Funções Básicas:
    - Inclui as funções essenciais para estruturar o código, como o início do programa e repetições de ações.
  - Biblioteca Erekopapa:
    - Contém funções pré-prontas para controlar os LEDs (olhos), o buzzer (boca) e a movimentação da cabeça do robô.

Erekopapa. INPI nº BR30202400684 92

COMO MONTAR O EREKOPAPA

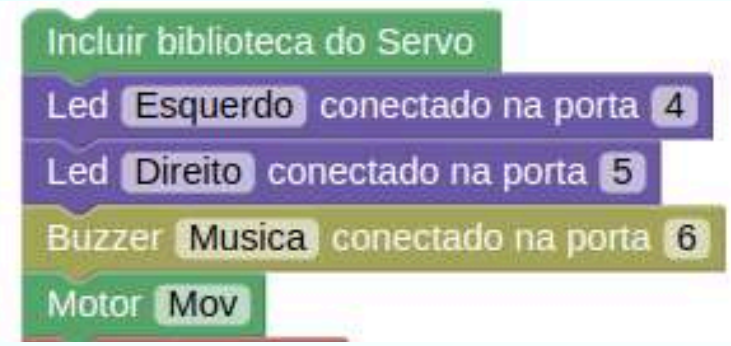
Exemplo de código para o Erekopapa:



COMO MONTAR O EREKOPAPA

1ª Parte:

- #include <Servo.h>
  - Indica ao programa para utilizar a biblioteca de controle do servomotor da cabeça do robô.
- int ledEsquerdo = 4;
  - Declara uma variável ledEsquerdo que armazena o número do pino 4, conectado ao LED do olho esquerdo.
- int ledDireito = 5;
  - Declara a variável ledDireito para o pino 5, conectado ao LED do olho direito.
- int buzzerMusica = 6;
  - Declara a variável buzzerMusica para o pino 6, conectado ao buzzer (componente de som).
- Servo motorMov;
  - Cria um objeto motorMov para gerenciar o servomotor.




```
#include <Servo.h>
int ledEsquerdo = 4;
int ledDireito = 5;
int buzzerMusica = 6;
Servo motorMov;
```

COMO MONTAR O EREKOPAPA

2ª Parte:

- void setup()
  - Função que executa uma única vez ao iniciar o Arduino, utilizada para configurações iniciais.
- pinMode(ledEsquerdo, OUTPUT);
  - Configura o pino do LED esquerdo como saída de sinal.
- pinMode(ledDireito, OUTPUT);
  - Configura o pino do LED esquerdo como saída de sinal.
- pinMode(buzzerMusica, OUTPUT);
  - Configura o pino do buzzer esquerdo como saída de sinal.
- motorMov.attach(8);
  - Associa o servomotor da cabeça ao pino digital 8 do Arduino.


```
void setup(){
  pinMode(ledEsquerdo, OUTPUT);
  pinMode(ledDireito, OUTPUT);
  pinMode(buzzerMusica, OUTPUT);
  motorMov.attach(8);

}
```

COMO MONTAR O EREKOPAPA

3ª Parte:

- void loop()
  - Função que executa continuamente enquanto o Arduino estiver ligado.
- digitalWrite(ledEsquerdo, HIGH);
  - Acende o LED esquerdo.
- digitalWrite(ledDireito, HIGH);
  - Acende o LED direito.
- digitalWrite(buzzerMusica, HIGH);
  - Ativa o som do buzzer.
- motorMov.write(0);
  - Gira o servomotor da cabeça para a posição de 0 graus.
- delay(2000);
  - Pausa a execução por 2000 milessegundos ou 2 segundos.
- digitalWrite(ledEsquerdo, LOW);
  - Apaga o LED esquerdo.
- digitalWrite(ledDireito, LOW);
  - Apaga o LED direito.
- digitalWrite(buzzerMusica, LOW);
  - Desativa o som do buzzer.
- motorMov.write(90);
  - Move o servomotor da cabeça para a posição de 90 graus.
- delay(2000);
  - Pausa a execução por 2000 milessegundos ou 2 segundos.


```
void loop(){
  digitalWrite(ledEsquerdo,HIGH);
  digitalWrite(ledDireito,HIGH);
  digitalWrite(buzzerMusica,HIGH);
  motorMov.write(0);
  delay(2000);
  digitalWrite(ledEsquerdo,LOW);
  digitalWrite(ledDireito,LOW);
  digitalWrite(buzzerMusica,LOW);
  motorMov.write(90);
  delay(2000);

}
```

COMO MONTAR O ARABEKO

- O Arabeko é um robô móvel com tração diferencial, desenvolvido para ensinar conceitos intermediários e avançados de robótica a iniciantes. Ele possui um chassi impresso em 3D que abriga componentes como Arduino, protoboard, ponte H, motores, baterias e rodas. Os participantes aprendem a montar, personalizar e criar novas peças utilizando modelagem 3D.

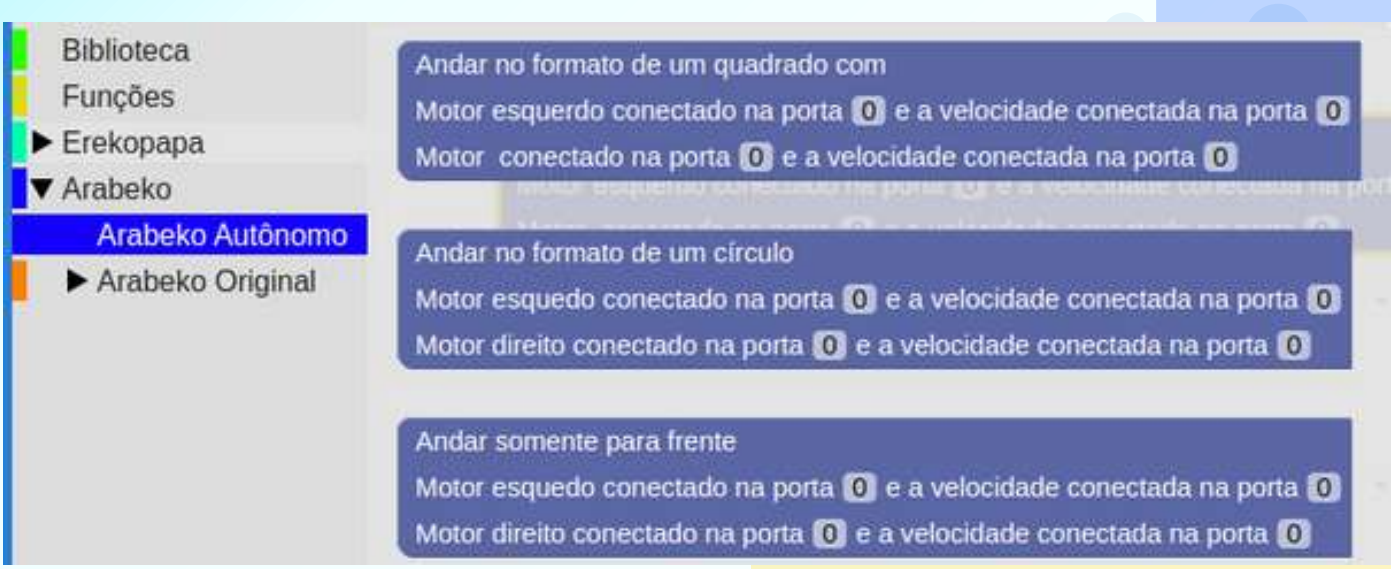
- O robô conta com comunicação via Bluetooth com um smartphone, e permite a expansão com sensores ultrassônicos para detecção de obstáculos. Também é possível acoplar um reboque com braço robótico e garra, desafiando os participantes a integrar programação, controle de movimento e manipulação simultaneamente.

COMO MONTAR O ARABEKO

Formas de usar o Arabeko no software

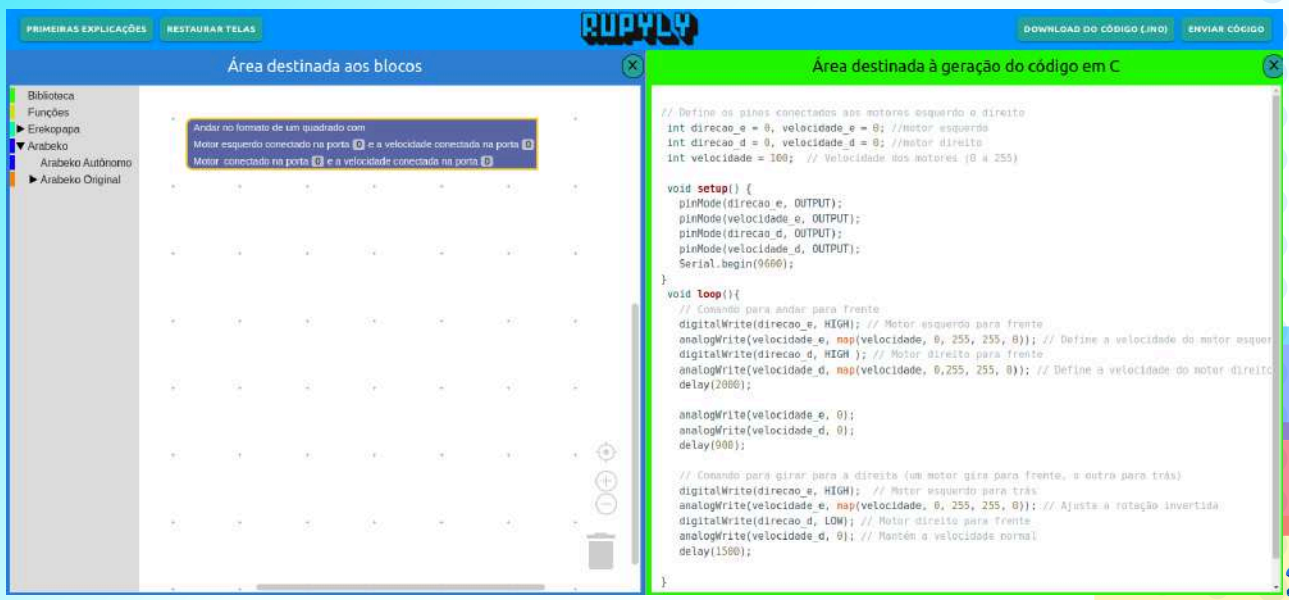
O Arabeko pode ser programado de duas maneiras diferentes, dependendo da oficina ou do nível do participante:

1. Modo automático - Arabeko Autonômo
   - Neste modo, o Arabeko possui movimentos pré-programados, selecionáveis na biblioteca. As opções incluem andar em quadrado, em círculo ou em linha reta.
   - Ideal para iniciantes, permite o uso rápido do robô sem necessidade de conhecimento prévio.



COMO MONTAR O ARABEKO

Para o Arabeko autônomo, basta arrastar um bloco na tela, indicar as portas conectadas no arduino e o código completo já será gerado automaticamente, com comentários explicando cada parte.



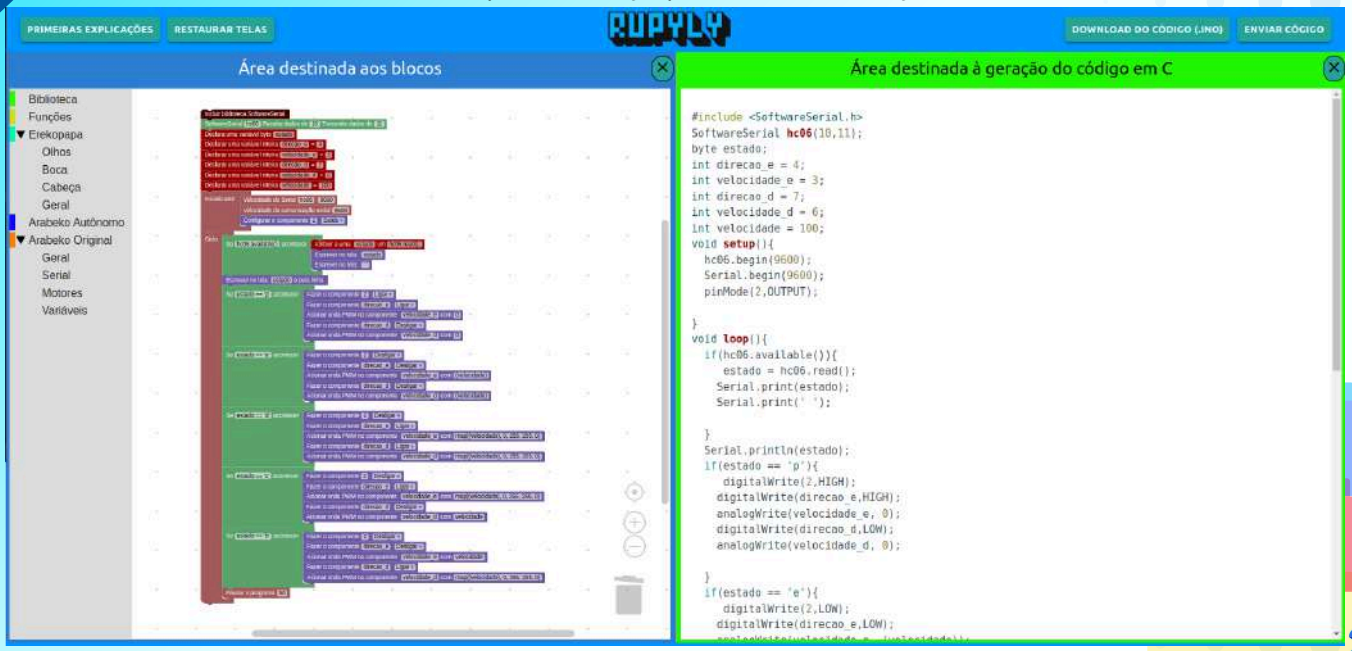
COMO MONTAR O ARABEKO

Formas de usar o Arabeko no software

2. Modo livre - Arabeko Original
   - Neste modo, o usuário tem total liberdade para criar o próprio código de movimento do robô.
   - Há também blocos para conectividade Bluetooth, possibilitando o controle remoto, por exemplo, via celular.
   - Ideal para quem já tem conhecimento em programação e busca explorar mais funcionalidades.

COMO MONTAR O ARABEKO

Exemplo de código para Arabeko Original:



COMO ENVIAR PARA O ARDUINO

Para enviar o código ao Arduino, existem duas formas:

1. Download e uso da IDE do Arduino
   - Baixe o código gerado e abra-o na IDE oficial. Isso permite edição manual, visualização e correção de erros diretamente no terminal da IDE.

DOWNLOAD DO CÓDIGO (.INO)

2. Envio direto pelo botão "Enviar Código"
   - Conectando o Arduino via USB, o software detecta a porta serial e, ao clicar no botão, compila e envia o código diretamente ao Arduino, se não houver erros.

ENVIAR CÓGIGO

ACOMPANHE MAIS

GitHub

Anabraz26

Linked in

Ana Caroline da Rocha Braz

Instagram

@AnaBraz26

UnB

EREKO

ROBÓTICA NA EDUCAÇÃO

Apêndice B

Formulário de Feedbacks da 1ª Versão

Feedback do Ereko-Blockly

Este formulário tem por objetivo coletar feedbacks sobre o aplicativo Ereko-blockly para o uso no ensino de arduíno durante as oficinas realizadas com os modelos educacionais criado pelo grupo de pesquisa Ereko da UnB.

* Indica uma pergunta obrigatória

1. É a primeira vez que você mexe com Arduíno? *

   Marcar apenas uma oval.

   - ◯ Sim
   - ◯ Não

2. É a primeira vez que você tem contato com programação em blocos? *

   Marcar apenas uma oval.

   - ◯ Sim
   - ◯ Não

Layout do aplicativo

A seguir, classifique cada pergunta em uma escala de 1 a 5, onde 1 representa 'Ruim' e 5 representa 'Muito Bom'.

3. O que você achou do layout do site? *

   Marcar apenas uma oval.

	1	2	3	4	5	
Ruim	◯	◯	◯	◯	◯	Muito boa

4. O que você achou do local das bibliotecas? *

Marcar apenas uma oval.

	1	2	3	4	5	
Ruin	◯	◯	◯	◯	◯	Muito boa

5. O que você achou da separação das bibliotecas? (Em modelos educacionais específicos) *

Marcar apenas uma oval.

	1	2	3	4	5	
Ruin	◯	◯	◯	◯	◯	Muito Boa

6. O que você achou da área destinada aos blocos? *

Marcar apenas uma oval.

	1	2	3	4	5	
Ruin	◯	◯	◯	◯	◯	Muito Boa

7. O que você achou da área destinada a geração do código em C? *

Marcar apenas uma oval.

	1	2	3	4	5	
Ruin	◯	◯	◯	◯	◯	Muito Boa

8. Sugestão de melhorias para o Layout *

Biblioteca Robô Antropomórfico

A seguir, classifique cada pergunta em uma escala de 1 a 5, onde 1 representa 'Ruim' e 5 representa 'Muito Bom'.

9. O que você achou da aparência da biblioteca? *

Marcar apenas uma oval.

	1	2	3	4	5	
Ruim	◯	◯	◯	◯	◯	Muito boa

10. O que você achou da separação em partes da biblioteca? *

Marcar apenas uma oval.

	1	2	3	4	5	
Ruim	◯	◯	◯	◯	◯	Muito boa

11. Você acredita que os alunos vão aprender da forma proposta? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não
- ◯ Talvez

12. Se você marcou não ou talvez, explique o por que

13. Sugestão de melhorias para a biblioteca do robô antropomórfico *

Biblioteca Robô com roda

Por favor, classifique cada pergunta em uma escala de 1 a 5, onde 1 representa 'Ruim' e 5 representa 'Muito Bom'.

14. O que você achou da aparência da biblioteca? *

Marcar apenas uma oval.

	1	2	3	4	5	
Ruim	◯	◯	◯	◯	◯	Muito boa

15. O que você achou da separação em partes da biblioteca? *

Marcar apenas uma oval.

	1	2	3	4	5	
Ruim	◯	◯	◯	◯	◯	Muito boa

16. Você acredita que os alunos vão aprender da forma proposta? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não
- ◯ Talvez

17. Se você marcou não ou talvez, explique o por que

_____

_____

_____

_____

_____

18. Sugestões de melhorias para a biblioteca do robô com roda *

_____

_____

_____

_____

_____

Sobre os blocos em geral

Por favor, classifique cada pergunta em uma escala de 1 a 5, onde 1 representa 'Ruim' e 5 representa 'Muito Bom'.

19. O que você achou da aparência dos blocos? *

Marcar apenas uma oval.

	1	2	3	4	5	
Ruin	◯	◯	◯	◯	◯	Muito boa

20. O quão intuitivo você achou os blocos? *

Marcar apenas uma oval.

	1	2	3	4	5	
Pou	◯	◯	◯	◯	◯	Muito intuitivo

21. Sugestões de melhorias para os blocos em geral *

Sobre utilizar o aplicativo para o ensino de programação e Arduíno

22. Você utilizaria esse aplicativo? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não
- ◯ Talvez

23. Você acredita que esse aplicativo facilitaria o ensino de Arduíno? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não
- ◯ Talvez

24. Se você marcou não ou talvez, explique o por que

25. Você achou que o aplicativo é eficaz em ensinar conceitos de programação? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não
- ◯ Talvez

26. Se você marcou não ou talvez, explique o por que

27. Você teve alguma dificuldade específica ao usar o aplicativo? *

28. Espaço para qualquer outra sugestão!

Apêndice C

Formulário de Feedbacks da 2ª Versão

Feedback do Rupyly

Este formulário tem por objetivo coletar feedbacks sobre o aplicativo Rupyly para o uso no ensino de arduino durante as oficinas realizadas com os modelos educacionais criado pelo grupo de pesquisa Ereko da UnB.

* Indica uma pergunta obrigatória

1. Você é de qual curso? *

   Marcar apenas uma oval.

   - ◯ Engenharia Mecatrônica
   - ◯ Engenharia da Computação
   - ◯ Ciência da computação
   - ◯ Outro

2. Se você respondeu outro, qual seria?

   _____

3. Você realizou teste do software na sua primeira versão? *

   Marcar apenas uma oval.

   - ◯ Sim
   - ◯ Não

4. É a primeira vez que você mexe com Arduíno? *

   Marcar apenas uma oval.

   - ◯ Sim
   - ◯ Não

5. É a primeira vez que você tem contato com programação em blocos? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

6. Se não, qual outros software de programação em bloco você já teve contato?

_____

Experiência Geral do Software

A seguir, classifique cada pergunta em uma escala de 1 a 5, onde 1 representa 'Muito Ruim' e 5 representa 'Excelente'.

7. Como você avalia sua experiência geral com do software? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Excelente

8. O software atendeu suas expectativas? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Parcialmente
- ◯ Não

9. O que você mais gostou no software? *

_____

10. O que você menos gostou ou teve mais dificuldade? *

11. O software foi intuitivo de se usar? *

Marcar apenas uma oval.

- Sim
- Parcialmente
- Não

12. Você teve alguma dificuldade em entender alguma funcionalidade? Se sim, qual? *

13. Você sentiu falta de alguma funcionalidade? Se sim, qual? *

14. Caso queira dizer mais sobre sua experiência em geral pode utilizar esse espaço!

Sobre o Livro - Rupyly Primeiros passos

15. Você leu o E-book? *

Marcar apenas uma oval.

- Sim *Pular para a pergunta 16*
- Parcialmente *Pular para a pergunta 16*
- Não *Pular para a pergunta 21*

Sobre o Livro - Rupyly Primeiros passos

16. O E-book atendeu as suas expectativas? *

Marcar apenas uma oval.

- Sim
- Parcialmente
- Não

17. A linguagem usada no E-book foi fácil de entender? *

Marcar apenas uma oval.

- Sim
- Parcialmente
- Não

18. O E-book te ajudou no entendimento de como funciona o software? *

Marcar apenas uma oval.

- Sim
- Parcialmente
- Não

19. O E-book te ajudou a montar os blocos para programar os robôs desejados? *

Marcar apenas uma oval.

- Sim
- Parcialmente
- Não

20. Caso queira deixar alguma sugestão ou crítica ao E-book pode utilizar esse espaço!

Sobre o Layout do Software

A seguir, classifique cada pergunta em uma escala de 1 a 5, onde 1 representa 'Muito Ruim' e 5 representa 'Excelente'.

21. O que você achou do layout do site? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Excelente

22. O quão intuitivo você achou o layout? *

Marcar apenas uma oval.

	1	2	3	4	5	
Nad	◯	◯	◯	◯	◯	Muito intuitivo

23. O que você achou da localização das bibliotecas? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Excelente

24. O que você achou da separação das bibliotecas? (Em modelos educacionais específicos) *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Excelente

25. O que você achou da área destinada aos blocos? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Excelente

26. O que você achou da área destinada a geração do código em C? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Excelente

27. Caso queira deixar alguma sugestão ou crítica ao Layout pode utilizar esse espaço!

Biblioteca Erekopapa

28. Você programou o Erekopapa? *

Marcar apenas uma oval.

- ◯ Sim *Pular para a pergunta 29*
- ◯ Não *Pular para a pergunta 36*

Biblioteca Erekopapa

Por favor, classifique cada pergunta em uma escala de 1 a 5, onde 1 representa 'Muito Ruim' e 5 representa 'Excelente'.

29. O que você achou da aparência da biblioteca do erekopapa? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Excelente

30. O que você achou da separação da biblioteca em partes de construção do erekopapa? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Excelente

31. Você acredita que a biblioteca está intuitiva a novos usuários? (Leve em consideração a linguagem que está nos blocos) *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não
- ◯ Talvez

32. Se você marcou não ou talvez, explique o por quê

_____

_____

_____

_____

_____

33. Você acredita que os alunos vão aprender da forma proposta? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não
- ◯ Talvez

34. Se você marcou não ou talvez, explique o por quê

35. Caso queira deixar alguma sugestão ou crítica a Biblioteca do Erekopapa pode utilizar esse espaço!

Biblioteca Arabeko

36. Você programou o Arabeko? *

Marcar apenas uma oval.

- Sim *Pular para a pergunta 37*
- Não *Pular para a pergunta 45*

Biblioteca Arabeko

Por favor, classifique cada pergunta em uma escala de 1 a 5, onde 1 representa 'Ruim' e 5 representa 'Muito Bom'.

37. O que você achou da aparência da biblioteca do Arabeko autônomo? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Excelente

38. O que você achou da aparência da biblioteca do Aarabeko original? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Excelente

39. O que você achou em separação o Arabeko em forma "autônomo" e "original"? *

40. Você acredita que a biblioteca está intuitiva a novos usuários? (Leve em consideração a linguagem que está nos blocos) *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não
- ◯ Talvez

41. Se você marcou não ou talvez, explique o por que

42. Você acredita que os alunos vão aprender da forma proposta? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não
- ◯ Talvez

43. Se você marcou não ou talvez, explique o por que

_____

_____

_____

_____

_____

44. Caso queira deixar alguma sugestão ou crítica a Biblioteca do Arabeko pode utilizar esse espaço!

_____

_____

_____

_____

_____

Sobre os blocos em geral

Por favor, classifique cada pergunta em uma escala de 1 a 5, onde 1 representa 'Ruim' e 5 representa 'Muito Bom'.

45. O que você achou da aparência dos blocos? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Excelente

46. O que você achou da linguagem dos blocos? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Excelente

47. O quão intuitivo você achou os blocos? *

Marcar apenas uma oval.

	1	2	3	4	5	
Pou	◯	◯	◯	◯	◯	Muito intuitivo

48. O que você achou de separar Blocos que seriam em comum das bibliotecas em lugar diferente? ("Biblioteca", para os includes, e "Funções", para funções gerais) *

49. Caso queira deixar alguma sugestão ou crítica aos Blocos pode utilizar esse espaço!

Sobre utilizar o aplicativo para o ensino de programação e Arduíno

50. Você utilizaria esse aplicativo para o ensino de programação e Robótica? *

Marcar apenas uma oval.

- Sim
- Não
- Talvez

51. Você acredita que esse aplicativo facilitaria o ensino de Arduíno? *

Marcar apenas uma oval.

- Sim
- Não
- Talvez

52. Se você marcou não ou talvez, explique o por que

53. Você achou que o aplicativo é eficaz em ensinar conceitos de programação? *

Marcar apenas uma oval.

- Sim
- Não
- Talvez

54. Se você marcou não ou talvez, explique o por que

55. Caso queira deixar alguma sugestão ou crítica relacionada ao ensino pode utilizar esse espaço!