BotForge: Uma linguagem de programação para o desenvolvimento de protótipos de seguidores de linha

Eduardo M. de A. Rosa[1], Luan Estevam[1], Isabel O. Trindade[1], Anna Dállety de Almeida[2], Leandro R. Mattioli[1]

[1]Departamento de Eletromecânica
Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG)
Araxá, MG – Brazil

[2]Departamento de Minas e Construção Civil
Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG)
Araxá, MG – Brazil

{rosa.eduardo,estevam.luan,trindade.isabel}@aluno.cefetmg.br,
anna.almeida@aluno.cefetmg.br, leandromattioli@cefetmg.br

Abstract. *This work presents the BotForge programming language, aimed at Brazilian elementary and high school students with little or no prior programming experience, for the purpose of designing line-following robot prototypes. The language was designed to operate on specific hardware, considering the available sensors and actuators. An integrated development environment capable of generating Arduino projects from BotForge code has already been designed and publicly deployed on a web page. The project is currently undergoing user testing in order to guide the direction and priorities of its development.*

Resumo. *Este trabalho apresenta a linguagem de programação BotForge, voltada para alunos do ensino fundamental e ensino médio do Brasil, com pouca ou nenhuma experiência prévia em programação, para o propósito de conceber protótipos de robôs seguidores de linha. A linguagem foi projetada para operar em um hardware específico, em termos dos sensores e atuadores disponíveis. Um ambiente de desenvolvimento integrado para gerar projetos Arduino a partir de códigos em BotForge já foi concebido e implantado publicamente em uma página Web. O projeto encontra-se em fase de testes com os usuários, a fim de nortear os caminhos e prioridades do desenvolvimento.*

1. Introdução

A robótica, com sua natureza multidisciplinar, é comumente usada para o aprendizado de programação no ensino fundamental [Viana et al. 2025]. Tradicionalmente, ferramentas de programação visual baseadas em blocos são adotadas para simplificar o processo de desenvolvimento de pensamento computacional [Sáez-López et al. 2019]. Essa abordagem evita distrações provocadas pela elevada carga sintática das linguagens textuais convencionais, e além disso confere maior consistência sintática por não permitir conectar blocos incompatíveis.

No entanto, ainda existe uma lacuna entre essa experiência e o uso das linguagens de programação textual convencionais, mesmo naquelas com pouca pontuação sintática. A linguagem Python, por exemplo, pode ser usada no ambiente Arduino, por meio de interpretadores de comandos [Kasotaki e Dimokas 2025; Steiner 2009]. Contudo, nesta linguagem a indentação do código não é apenas estilo, mas sim parte da sintaxe para delimitação dos blocos: um espaço em branco a mais ou a menos já torna o código inválido, o que pode ser um desafio para iniciantes desatentos na

digitação. Além disso, não há distinção entre maiúsculas e minúsculas, e as palavras reservadas são todas em inglês.

Supõe-se, então, que seria benéfico ter uma etapa intermediária entre o primeiro contato com a área da programação – baseado nas ferramentas visuais, e o uso das linguagens comerciais. Para o treinamento em algoritmos de forma geral, existem projetos como o PSeInt [Castillo e Arsenio 2024].

Assim, neste trabalho, propõe-se uma nova linguagem, para esse conjunto de requisitos, denominada BotForge. Além das facilidades sintáticas e semânticas, a linguagem também simplifica a integração com sensores e atuadores, omitindo os detalhes de funcionamento interno e possuindo uma placa alvo com uma pinagem pré-definida.

2. Especificação da Linguagem

Cada aspecto da linguagem foi definido priorizando simplicidade sintática e semântica para para iniciantes em programação e robótica, sem perder de vista, por outro lado, as funcionalidades desejadas e a viabilidade de implementação durante a codificação do compilador fonte-a-fonte. Nesse sentido, a pontuação foi significativamente reduzida, usando quebras de linha para delimitar as instruções e delimitação de escopo mediante palavras reservadas. Esta seção apresenta algumas características da linguagem.

Identificadores: os nomes de variáveis e funções seguem praticamente as mesmas regras convencionais, mas sem diferenciar caracteres acentuados e não acentuados e nem aplicar distinção entre letras maiúsculas e minúsculas, eliminando erros comuns de iniciantes.

Tipos de dados: um outro desafio para iniciantes em linguagens de baixo nível é na quantidade de diferentes tipos de dados, principalmente quando se trata de tipos numéricos. A linguagem BotForge trabalha apenas com os tipos primitivos `número`, `texto` e `lógico` (`verdadeiro` ou `falso`), além do suporte nativo à listas, que são implementadas como listas dinâmicas encadeadas. Todos os números são do tipo ponto flutuante, com conversões de tipo (casting) e truncamento aplicados automaticamente em alguns contextos, como indexação de coleções ordenadas.

Funções e Tarefas: a linguagem provê uma série de funções *built-in*, que expõem uma versão simplificada da API do Arduino. A definição de funções pelo programador foi substituida por um mecanismo para a definição de **tarefas** periódicas ou do tipo *one-shot*, aproveitando recursos do sistema operacional FreeRTOS.

2.1. Nível de abstração da linguagem

Linguagens de baixo nível permitem controle fino do hardware, otimização de uso de memória e de tempo de processamento. Em sistemas embarcados comerciais, essas características são desejáveis, justificando, muitas vezes, o maior tempo de desenvolvimento e depuração envolvido no uso destas tecnologias. Por outro lado, as linguagens de alto nível conseguem abstrair boa parte das operações repetitivas e detalhadas para um ferramental mais próximo do pensamento humano, configurando uma experiência de codificação mais fluida mas, ao mesmo tempo, sacrificando a versatilidade e a capacidade de adequação das instruções em baixo nível.

Para iniciantes, deixar o programador mais distante de certos detalhes internos de implementação é uma estratégia interessante. No caso da linguagem BotForge, essa simplificação já foi iniciada no projeto da própria sintaxe da linguagem. Ainda assim, outros recursos foram embutidos para facilitar o uso dos sensores e atuadores do sistema:

- aplicação de filtro de média móvel nos sensores, totalmente encapsulado na linguagem;

- valores adimensionais, como velocidade de motor ou nível de reflexão, normalizados para faixas mais comuns, por exemplo entre 0 e 100;

3. Especificação do Hardware

A seleção dos módulos que compõem o protótipo do seguidor de linha visou uma adequação aos requisitos exigidos em competições da área. Especificamente, além da função seguidor de linha, foram considerados os desafios de reconhecer marcações coloridas em uma pista que indicam a direção a ser seguida, subir e descer rampas, se localizar em uma área fechada e mover objetos até um destino fixo. O kit tomado como alvo principal da programação em BotForge é apresentado na A Figura 1.

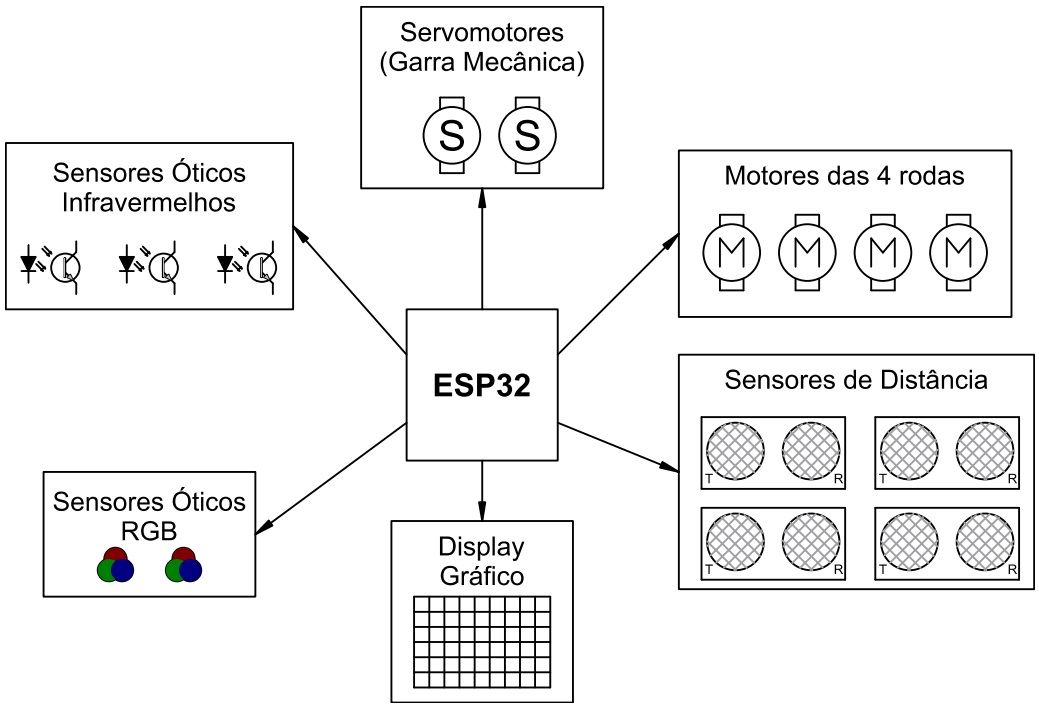

Figura 1: Visão geral dos componentes de hardware

O hardware desenvolvido permite expansão para acomodar mais sensores e atuadores, baseados na interface I^2C. Contudo, a linguagem BotForge, até o presente momento, ainda não dispõe de funções para realizar esse tipo de integração.

4. Ambiente de desenvolvimento e processo de compilação e gravação

O ambiente de desenvolvimento consiste em um navegador Web, para a etapa de codificação, e no Arduino IDE para a etapa final de gravação. O editor usa a biblioteca Monaco, o mesmo componente presente no popular Visual Studio Code, com a adição das regras para destaque de sintaxe da linguagem BotForge.

A compilação segue o processo tradicional de (i) obter uma lista de *tokens* a partir do código de entrada, por meio de um analisador léxico, (ii) submeter o fluxo de *tokens* a um analisador sintático, que constrói uma árvore de sintaxe abstrata baseada em um conjunto de regras de transformação e (iii) visitar a árvore e gerar o código em Arduino, por meio de um analisador semântico. O código em Arduino é então complementado com arquivos de cabeçalho e a solução é enviada em um pacote ZIP, contendo o *sketch* em Arduino a ser usado no processo de gravação. O sistema não exige nenhum ajuste pelo usuário no código gerado. O fluxo completo, do código BotForge à gravação no hardware, é apresentado na Figura 2.

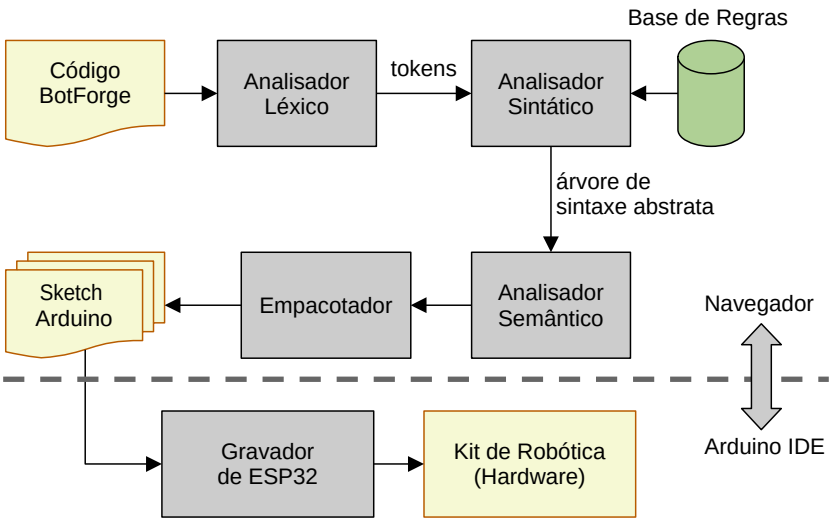

Figura 2: Etapas do processo completo da compilação ao upload para o hardware

Uma solução inteiramente baseada na Web, sem a necessidade de nenhum programa adicional instalado no lado do cliente, está em fase de desenvolvimento.

5. Resultados

A ferramenta foi hospedada por meio de um serviço de implantação contínua (Continuous Deployment), podendo ser acessada gratuitamente no endereço https://cefetmg-araxa.gitlab.io/pet-eai/botforge/.

Um trecho de código em BotForge e uma amostra do código Arduino gerado pode ser visto na Figura 3.

Baixar projeto Arduino | Salvar | Abrir

```
tarefa configurar faz
  mostrar('Sistema Iniciado.')
fimtarefa

tarefa loop faz
  mostrar('Distância 1: ')
  mostrar(sensordistancia1)
  se sensorlinha1 < 40 e sensorlinha2 > 60 então
    motor1 = 60
    motor2 = -60
  fimse
  se sensorlinha2 < 40 e sensorlinha1 > 60 então
    motor1 = -60
    motor2 = 60
  fimse
fimtarefa
```

```
void loop() {
  Serial.println("Distância 1: ");
  Serial.println(bfLerDistanciaFiltrada(1));
  if (bfLerLinha(1) < 40 && bfLerLinha(2) > 60)
  {
    bfAtualizarMotor(1, 60);
    bfAtualizarMotor(2,  - 60);
  }
  if (bfLerLinha(2) < 40 && bfLerLinha(1) > 60)
  {
    bfAtualizarMotor(1,  - 60);
    bfAtualizarMotor(2, 60);
  }
}
```

Figura 3: Código em BotForge (à esquerda) e parte do código Arduino gerado (à direita)

Uma importante consequência da forma como a solução foi desenvolvida é a capacidade de apresentar os erros de compilação enquanto o código é digitado, sem a

necessidade de disparar a ação de tentar baixar o projeto.

6. Conclusões

O compilador já está operacional e foi usado para especificar algumas rotinas básicas de seguidores de linha. A linguagem, mais próxima ao português e com carga sintática significativa menor quando comparado com as linguagens textuais tradicionais, indica o potencial da ferramenta no ensino de robótica para iniciantes.

Contudo, o kit físico ainda está em fase de construção. Uma vez concluído, pretende-se dar continuidade ao projeto elaborando um manual de referência e alguns tutoriais e então aplicando as ferramentas desenvolvidas em projetos de extensão com escolas do município de Araxá – MG. O resultado da atividade irá nortear as melhorias tanto do kit como da linguagem de programação.

7. Agradecimentos

Os autores Eduardo Marques de Amorim Rosa e Anna Dállety de Almeida, integrantes do PET-CEFET-MG, agradecem ao programa pelo apoio ao desenvolvimento deste trabalho.

Referências

Castillo, T. e Arsenio, L. (2024). El uso del PSeInt como herramienta educativa en el desarrollo del pensamiento lógico de los estudiantes del IESTP Recuay–Ancash 2023.

Kasotaki, S. e Dimokas, N. (2025). Emotional Engagement in STEM Programming: Micro:Bit and MicroPython vs. Traditional Python in Upper Secondary Education. In *2025 10th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*.

Sáez-López, J.-M., Sevillano-García, M.-L. e Vazquez-Cano, E. (1 dec 2019). The effect of programming on primary school students' mathematical and scientific understanding: educational use of mBot. *Educational Technology Research and Development*, v. 67, n. 6, p. 1405–1425.

Steiner, H.-C. (jun 2009). Firmata : Towards Making Microcontrollers Act Like Extensions of the Computer. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. Zenodo. https://doi.org/10.5281/zenodo.1177689.

Viana, R. P., Marques, R. C. F. V., Fonseca, G. M. R. e Almeida, P. L. C. (20 jul 2025). Pensamento Computacional e Cultura Maker na educação básica: programação em bloco e Robótica Educacional no Projeto de Extensão MyCode. *Anais do Workshop de Inovação, Desenvolvimento, Educação e Inclusão com Ações Maker (IDEIA); 2025: Anais do I Workshop de Inovação, Desenvolvimento, Educação e Inclusão com Ações MakerDO*. https://doi.org/10.5753/ideia.2025.8643