

ROBÓTICA

AULA 36

Primeiros Passos Módulo 4



Robô
Equilibrista - III

Diretoria de Tecnologia e Inovação

GOVERNADOR DO ESTADO DO PARANÁ

Carlos Massa Ratinho Júnior

SECRETÁRIO DE ESTADO DA EDUCAÇÃO

Roni Miranda Vieira

DIRETOR DE TECNOLOGIA E INOVAÇÃO

Claudio Aparecido de Oliveira

COORDENADOR DE TECNOLOGIAS EDUCACIONAIS

Marcelo Gasparin

Produção de Conteúdo

Viviane Dziubate

Validação de Conteúdo

Darice Alessandra Deckmann Zanardini

Roberto Carlos Rodrigues

Viviane Dziubate

Revisão Textual

Kellen Pricila dos Santos Cochinski

Projeto Gráfico e Diagramação

Edna do Rocio Becker

Apoio Técnico

Equipe UFMS

2026

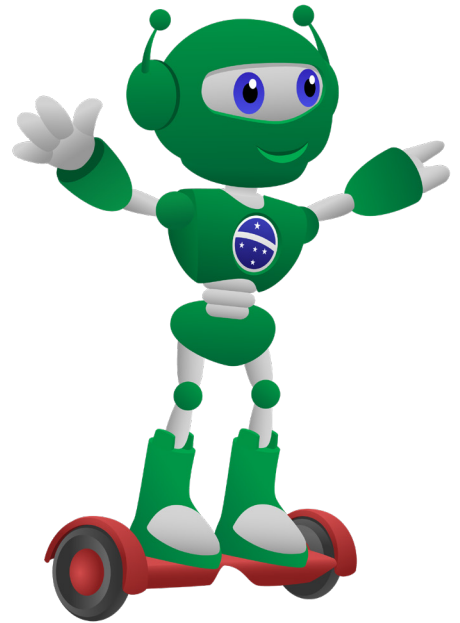
SUMÁRIO

Introdução	2
Objetivos	4
Roteiro da aula	5
1. Contextualização	5
2. Feedback e finalização	13
Referências bibliográficas	13

Introdução

Bem-vindos à aula 36, nesta aula, vamos mergulhar no coração pulsante do nosso robô: o código!

Chegamos ao momento mais empolgante e desafiador do nosso projeto: dar vida ao robô equilibrista através da programação. Equilibrar um robô de duas rodas não é tarefa simples – é um desafio que testa paciência, observação e criatividade. Mas aqui vai uma verdade fundamental: não é impossível, e vocês vão conseguir. Antes de vocês, outras pessoas enfrentaram o mesmo obstáculo e saíram com um robô que desafia a gravidade. O segredo está na iteração: testar, ajustar, observar e repetir. Vamos juntos nesta jornada, passo a passo, com a confiança de que a física e a programação estão do nosso lado.



Imagine o cenário: um robô que, ao ser inclinado, “sente” a mudança através do MPU-6050, calcula a inclinação em graus e comanda os motores para corrigir o equilíbrio. Isso não é ficção científica – é física aplicada! O acelerômetro e giroscópio do MPU-6050 medem aceleração e rotação, traduzindo forças gravitacionais em dados numéricos. Os motores, por sua vez, geram torque para contrabalançar o centro de massa deslocado. Em termos de programação, usamos loops infinitos e condições “se” para criar um controle proporcional simples: se o ângulo de pitch (inclinação frente/trás) ultrapassar um limiar (threshold¹), os motores aceleram na direção oposta para recuperar o equilíbrio. É como um pêndulo invertido controlado por código – complexo, mas compreensível quando dividido em partes.

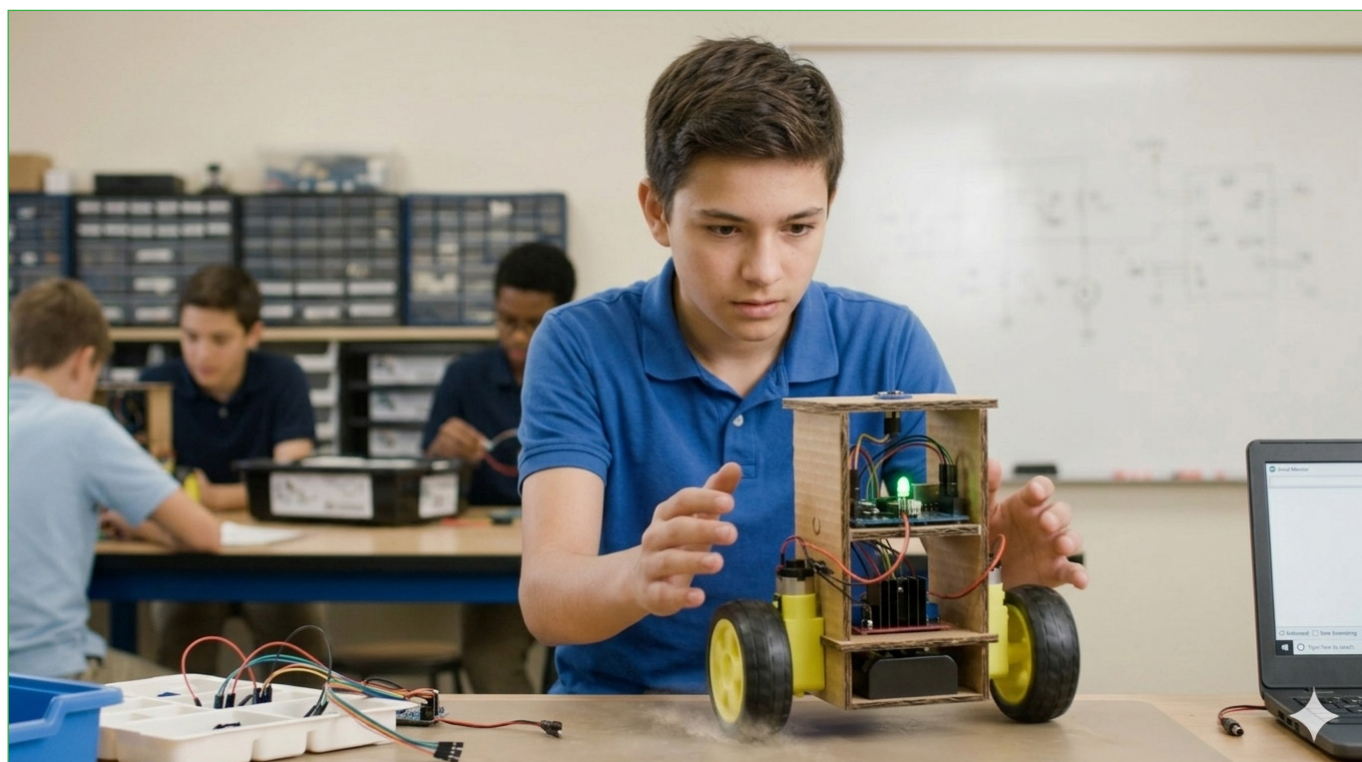
Por que é difícil? Porque cada robô é único: variações na montagem (como o centro de massa ou atrito das rodas) afetam diretamente o comportamento. Um threshold de 3 graus pode funcionar para um grupo, mas outro pode precisar de 5 graus. Vibrações dos motores podem falsear leituras do sensor, e uma bateria fraca pode causar quedas de tensão. Mas é exatamente essa variabilidade que torna o projeto educativo: vocês

¹ Thresholds (pronuncia-se “trésholds”) são valores-limite ou pontos de corte que definem quando o nosso robô deve reagir. É como uma linha imaginária que, quando ultrapassada, aciona uma ação específica.

aprendem a depurar, ajustar e otimizar, habilidades essenciais em engenharia e ciência. A Física (Lei de Newton: $\text{força} = \text{massa} \times \text{aceleração}$) conecta-se diretamente com a prática (código que responde a dados reais) e o mBlock torna isso acessível – blocos visuais substituem linhas de código complexas, permitindo foco na lógica.

Lembrem-se: o sucesso vem da observação atenta. Nos primeiros testes, o robô pode cair rapidamente ou oscilar sem parar. Não desistam! Anotem o que acontece (ângulos no monitor serial, direção dos motores) e ajustem: diminuam thresholds para reações mais rápidas, aumentem para reduzir tremores. Testem por 5-7 minutos inicialmente, sem pressa. Pequenas mudanças – como inverter um fio na ponte H ou adicionar um offset (deslocamento) ao ângulo – podem transformar uma queda em equilíbrio estável. Vocês têm o conhecimento das aulas anteriores (montagem, sensores, motores) e agora o código das figuras anexas como guia. Vamos focar no funcionamento: o loop lê o MPU-6050 continuamente, verifica condições e comanda motores via PWM. É simples no conceito, mas poderoso na execução.

Preparem-se para uma aula intensa, mas gratificante. Vocês não estão apenas programando um robô – estão criando um sistema que une Física, Matemática e Engenharia. No final, ver o robô ficar em pé será a prova de que perseverança vence desafios. Vamos nessa, com calma e confiança: vocês vão conseguir equilibrar o impossível!



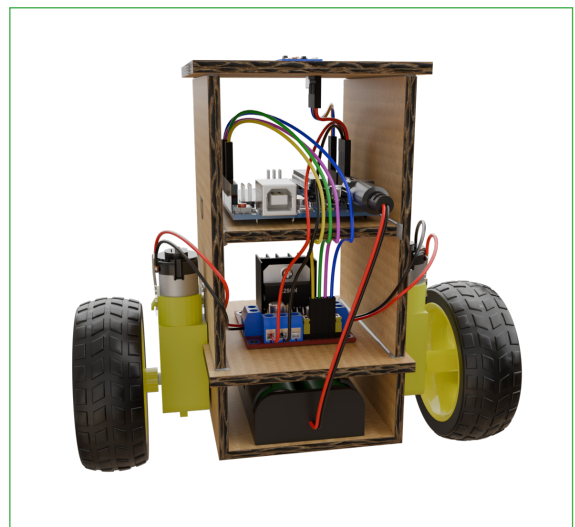
Objetivos desta aula

- Aplicar conceitos de forças, torque, equilíbrio dinâmico, centro de massa, movimento (velocidade, aceleração) e rotação em sistemas reais de controle;
- Dominar o funcionamento do MPU-6050 (acelerômetro e giroscópio 3 eixos), leitura de sensores, controle de motores DC via ponte H e integração com Arduino;
- Programar e implementar controle proporcional simples (PID básico), loops infinitos, estruturas condicionais e ajustes de thresholds para equilíbrio estável;
- Depurar código e hardware, identificando causas de instabilidade (ex.: ruído de vibração, desalinhamento de massa);
- Valorizar a experimentação iterativa e a paciência em ajustes finos;
- Reconhecer a aplicação prática da física e programação em tecnologias cotidianas, como hoverboards e robôs autônomos.

Lista de materiais

- Protótipo do robô equilibrista (montado nas aulas anteriores, com Arduino Uno, MPU-6050, ponte H, motores DC e bateria 18650);
- Notebook com mBlock (versão 5 ou superior, extensão MPU-6050 instalada);
- Cabo USB para conectar Arduino ao computador;
- Papel e caneta para anotações de testes e ajustes.

Figura 1 – Lista de materiais



Fonte: Roberto Rodrigues, 2025.

Nota para o professor(a): certifique-se de que o mBlock tenha a extensão MPU-6050 instalada. Esta aula é adaptável; valores de thresholds variam conforme a montagem física de cada grupo.

Roteiro da aula

1. Contextualização

Vocês já imaginam um hoverboard equilibrando-se sozinho? É exatamente isso que vamos programar hoje! Nosso código lê o MPU-6050, calcula inclinações e comanda motores para manter o equilíbrio. É simples, mas mágico: física virando ação digital. Vamos focar no código das figuras – cada bloco é uma decisão que salva o robô de uma queda!

Explicaremos o código passo a passo, baseado nas figuras anexas, para vocês entenderem como cada bloco funciona no robô. Começaremos com a inicialização do sensor, que será o “olho” do robô, e avançaremos para as condições que controlam os motores, garantindo que o equilíbrio seja mantido.

Antes de começar a programar, prepare o ambiente no mBlock para garantir que tudo funcione perfeitamente. Nesta aula, você deve adicionar a extensão MPU-6050 ao seu código no mBlock e usar o Arduino Uno padrão do mBlock. Siga estes passos:

1. Abra o mBlock: inicie o software no seu notebook. Certifique-se de que está na versão 5 ou superior para compatibilidade com as extensões.

2. Selecione o dispositivo: no canto superior esquerdo da tela, clique no ícone de dispositivo (parece o robô “B1N0” ou

placa). Escolha “Arduino Uno” da lista de dispositivos disponíveis. Isso configura o mBlock para programar especificamente o Arduino Uno, garantindo que os blocos sejam traduzidos corretamente para o hardware.

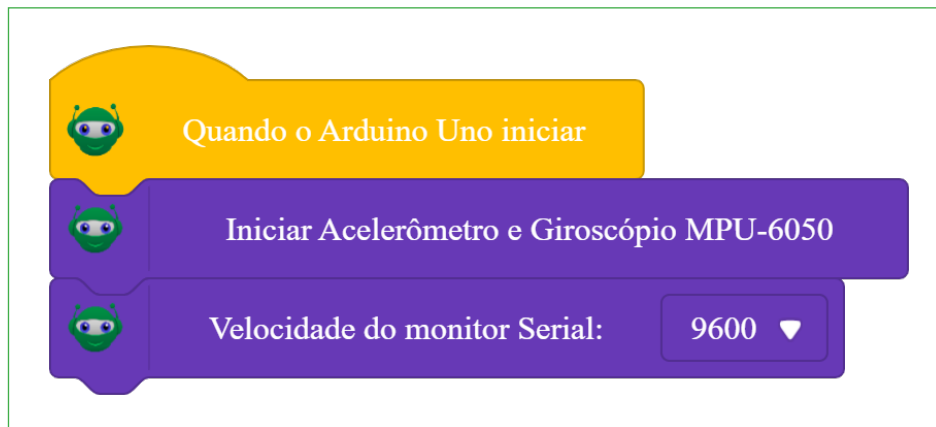
3. Adicione a extensão MPU-6050: vá para a aba “extensões” (ou “extensions” em inglês) no menu lateral esquerdo. Procure por “MPU-6050” na lista de extensões disponíveis. Clique em “adicionar” ou “install” para baixar e ativar a extensão. Isso adiciona os blocos específicos para o sensor de aceleração e giroscópio ao painel de blocos. Sem essa extensão, você não poderá usar os comandos de leitura do MPU-6050.

4. Verifique a conexão: conecte o Arduino Uno ao computador via cabo USB. No mBlock, clique em “conectar” (ou “connect”) para confirmar que o dispositivo foi reconhecido. Agora você está pronto para arrastar os blocos e criar o código.

Essa configuração é essencial porque o mBlock precisa “saber” que estamos trabalhando com o Arduino Uno e o sensor MPU-6050. Se a extensão não estiver instalada, os blocos não aparecerão no painel.

Vamos programar agora! Observe a Figura 2: ela mostra como iniciar o MPU-6050. Selecione o bloco **"Quando o Arduino Uno iniciar"** é o ponto de partida – ele configura o sensor para começar a medir aceleração e rotação. Depois, adicionamos **"Iniciar Acelerômetro e Giroscópio MPU-6050"** e definimos a velocidade de comunicação para **"Velocidade do monitor Serial: 9600"**, o que permite leituras rápidas sem sobrecarregar o Arduino.

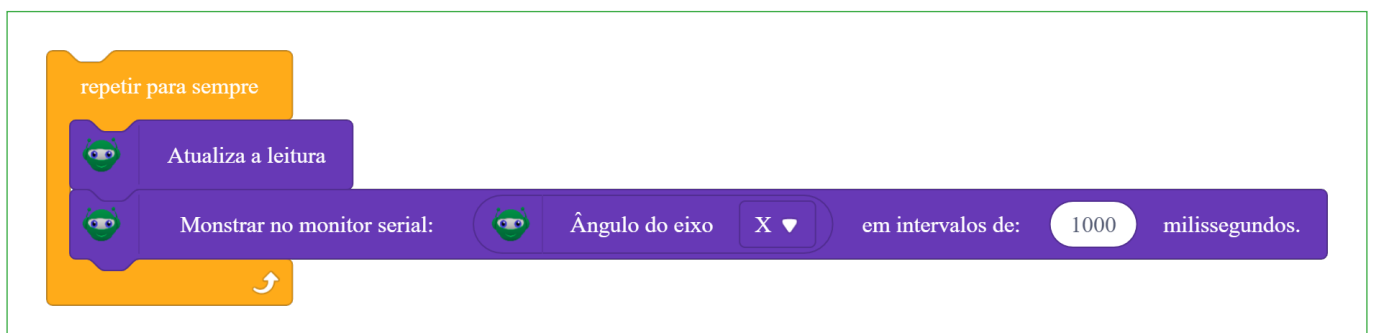
Figura 2 – Iniciando a leitura do sensor



Fonte: mBlock, 2026.

No loop **"Repetir para sempre"**, atualizamos a leitura do sensor e definimos o ângulo do eixo X a cada 1 segundo. Isso significa que o robô está constantemente "ouvindo" sua própria inclinação, traduzindo forças gravitacionais em números que o código pode usar. Observe a Figura 3.

Figura 3 – Leitura do sensor



Fonte: mBlock, 2026.

Agora, vamos para a Figura 4, que introduz a lógica de equilíbrio. Aqui, usamos condições “se” para verificar thresholds – valores-limite que determinam quando agir.

Figura 4 – Introduz a lógica de equilíbrio



Fonte: mBlock, 2026.

Se o ângulo X for maior que 3 graus (indicando queda para a direita), o código acelera os motores para a direita (Pino 11 com PWM 250, pino 10 como 0, pino 6 como 250 e pino 5 com 0), criando torque para corrigir.



No bloco “Senão” insira mais um bloco “Se Senão” e dentro do “se” conforme Figura 5, complete: se o ângulo do eixo x for menor que -8 graus (queda para a esquerda), acelera para a esquerda (Pino 11 com PWM 0, pino 10 como 250, pino 6 como 0 e pino 5 com 250). Caso contrário, os motores param. Esses thresholds assimétricos (3 e 8) compensam desbalanceamentos físicos, como um centro de massa deslocado, garantindo que o robô reaja de forma proporcional à inclinação.

Figura 5 – Lógica de equilíbrio

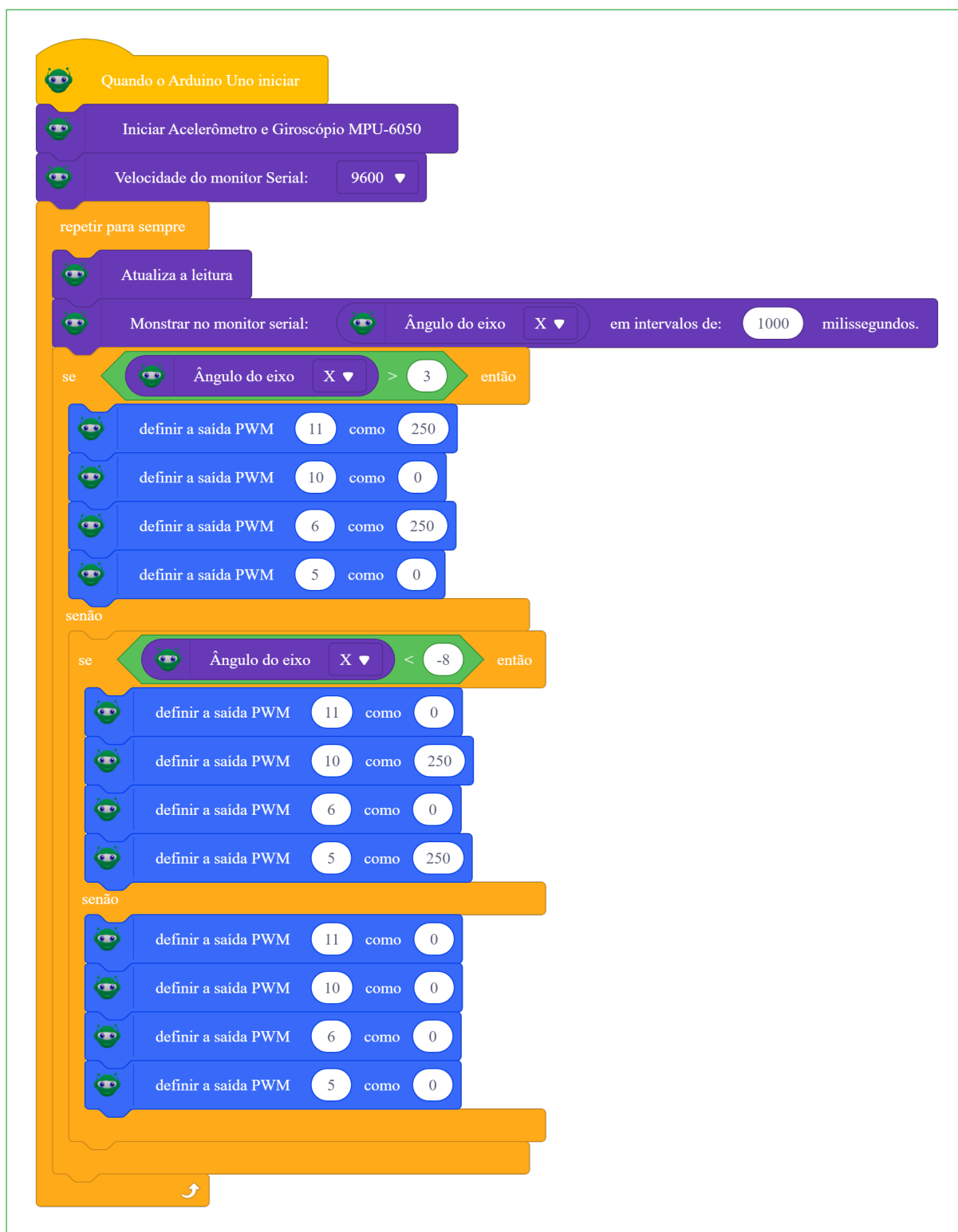


Fonte: mBlock, 2026.

Por fim, a Figura 6 combina tudo em um código completo. Esse loop infinito garante que o robô monitore e ajuste continuamente, transformando dados do sensor em comandos motores.

O controle é proporcional: quanto maior for a inclinação, maior a correção, mas ajustes finos evitam oscilações. Testem por 5-7 minutos inicialmente – se oscilar, aumente thresholds; se cair rápido, diminua-os.

Figura 6 – Código completo



Fonte: mBlock, 2026.

Com o código carregado e testado, vocês acabaram de transformar um conjunto de componentes em um sistema inteligente que desafia a gravidade. Agora, o robô não é mais apenas hardware – ele tem uma mente digital!

Confira o código completo do robô equilibrista no mBlock online pelo link:



<https://planet.mblock.cc/project/7037865> .

Desafio final: o mestre do equilíbrio

Grupos, o desafio agora é otimizar o robô para que ele se mantenha equilibrado por pelo menos 1 minuto sem cair (depois do tempo de espera de 5-7 minutos para estabilização do acelerômetro), ajustando apenas os thresholds e PWM. Quem conseguir isso ganha o título de “Engenheiro de Equilíbrio”! Anotem seus valores iniciais e finais para compartilhar com a turma.



E se... (cenários para refletir)

- E se o robô oscilar demais? Aumente os thresholds para 5/10, reduzindo reações excessivas.
- E se ele cair antes de reagir? Diminua os thresholds para 2/5, tornando-o mais sensível.
- E se o centro de massa estiver deslocado? Ajuste thresholds assimétricos, como 4/6, para compensar.
- E se as vibrações falsearem o sensor? Use média de leituras ou adicione um filtro no código.

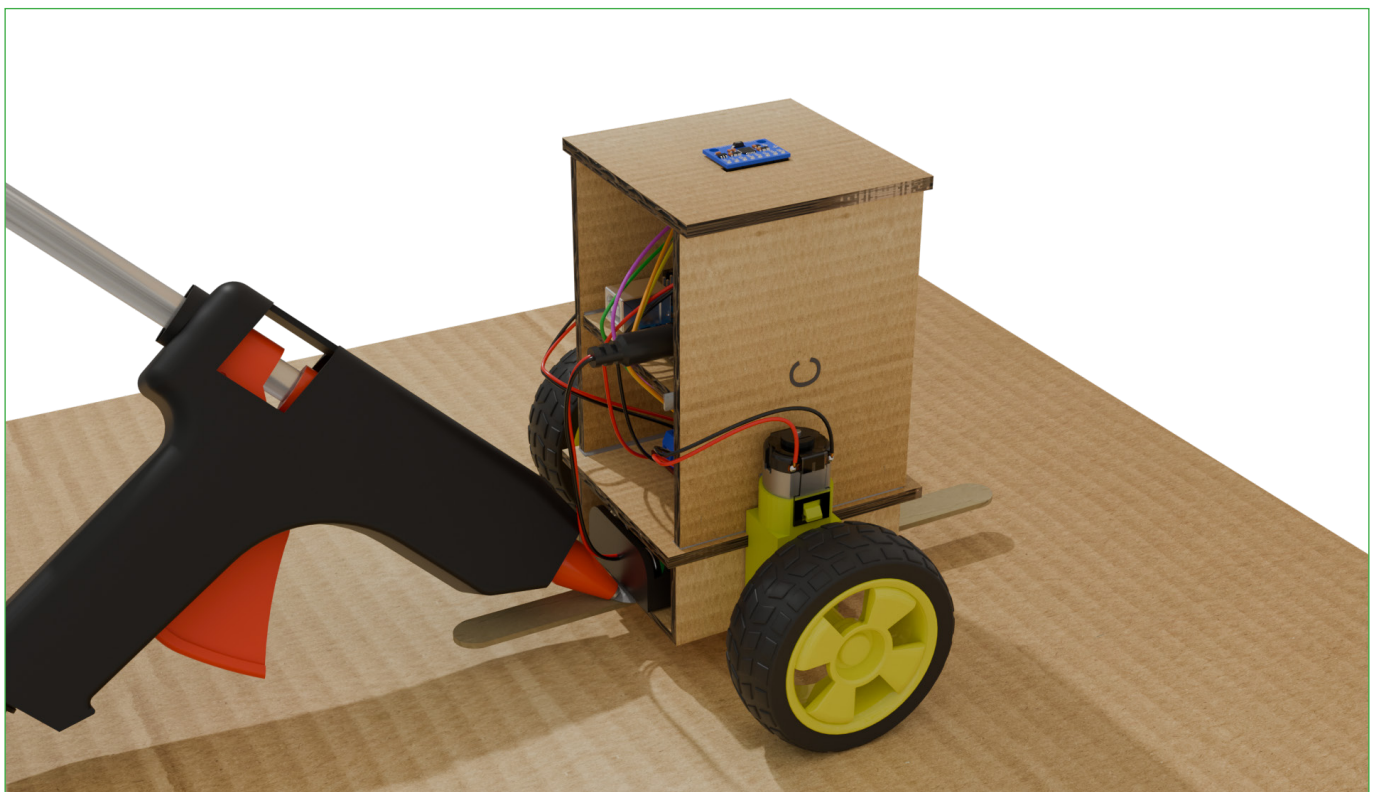
Dicas de mestre para o sucesso

- Teste em etapas: comece com USB (sem bateria) para verificar leituras do MPU-6050.
- Ajuste iterativo: mude um parâmetro por vez (ex.: só thresholds) e observe o efeito.
- Documente tudo: anote ângulos, PWM e comportamento para identificar padrões.
- Paciência é chave: o equilíbrio perfeito pode precisar de vários testes – cada erro é uma lição.

Adaptação quando não funciona como previsto

Se, mesmo após ajustes, o robô não conseguir equilibrar-se sozinho, vamos adaptar um suporte embaixo, conforme a Figura 7, pode ser com palitos de sorvete ou algo similar.

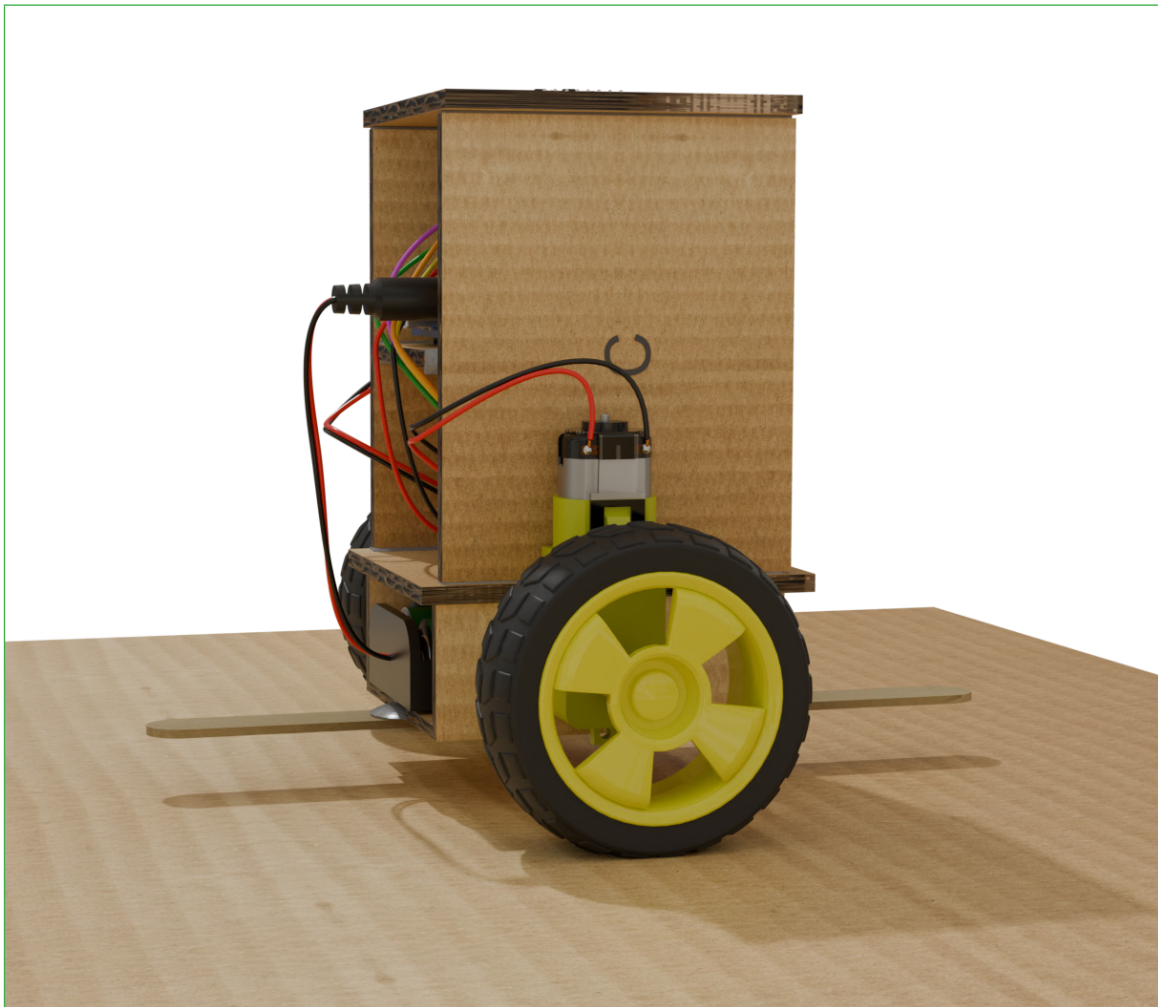
Figura 7 – Suporte auxiliar



Fonte: Roberto Rodrigues, 2026.

Esse suporte auxiliar (Figura 8) funciona como uma base estabilizadora, ajuda a reduzir a inclinação crítica, permitindo que o código funcione melhor enquanto vocês refinam o projeto. É uma solução prática para continuar aprendendo sem frustração.

Figura 8 – Suporte auxiliar de equilíbrio adaptável



Fonte: Roberto Rodrigues, 2026.

Essa adaptação nos lembra que, na vida, às vezes precisamos de apoios temporários para superar desafios. Assim como um robô pode precisar de uma base extra para equilibrar-se, nós, humanos, também nos adaptamos a situações difíceis – seja com ajuda de amigos, ferramentas ou novas perspectivas. Reflitam: como vocês já se adaptaram a um problema inesperado na escola ou em casa? Essa flexibilidade é o que nos torna resilientes, assim como o robô que aprende a equilibrar-se com ajustes e apoios.

2. Feedback e finalização

Equilibrar um robô que contribua com o comportamento de um sistema é um desafio real da engenharia moderna, e é essencial para criar máquinas mais seguras, rápidas e precisas. Vocês não apenas programaram um robô – desenvolveram habilidades de resolução de problemas, trabalho em equipe e pensamento crítico que serão valiosas em qualquer carreira.

Por fim, organizem os materiais de volta ao kit de Robótica para garantir que tudo esteja pronto para as próximas aulas. Continuem explorando e inovando, pois o futuro da tecnologia está nas mãos de mentes curiosas e resilientes como as de vocês.

REFERÊNCIAS

ARDUINO. Documentação de Referência da Linguagem Arduino. Disponível em: <https://www.arduino.cc/reference/pt/>. Acesso em: 10, nov. 2025.

mBlock. mBlock 5 - Programming Tool for STEAM Education. Disponível em: <https://www.mblock.cc/>. Acesso em: 27 jan. 2026.



UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL (UFMS)
FACULDADE DE COMPUTAÇÃO (FACOM)

PROFESSORES

- Amaury Antônio de Castro Junior
- Anderson Corrêa de Lima
- Glauder Guimarães Ghinozzi
- Graziela Santos de Araújo
- Said Sadique Adi

ESTUDANTES

- Bruno Pereira Wesner da Silva - Engenharia de Computação
- Caetano de Medeiros Santana - Sistemas de Informação
- Fernanda das Neves Merqueades Santos - Ciência da Computação
- Filipe de Andrade Machado - Ciência da Computação
- Gabriel Pereira Falcão - Ciência da Computação
- Guilherme Siqueira Fiani - Engenharia de Software
- Jenniffer Oliveira Checchia - Ciência da Computação
- Maria Paula do Nascimento Santos - Engenharia de Computação
- Pedro Paulo de Oliveira Andrade - Ciência da Computação
- Vinicius Wagner da Silva - Engenharia de Software

DIRETORIA DE TECNOLOGIAS E INOVAÇÃO (DTI)
COORDENAÇÃO DE TECNOLOGIAS EDUCACIONAIS (CTE)

EQUIPE ROBÓTICA PARANÁ

- Adilson Carlos Batista
- Ailton Lopes
- Andrea da Silva Castagini Padilha
- Cleiton Rosa
- Darice Alessandra Deckmann Zanardini
- Edna do Rocio Becker
- Enzo Enrico Giacomini Piolla
- Kellen Pricila dos Santos Cochinski
- Marcelo Gasparin
- Michele Serpe Fernandes
- Michelle dos Santos
- Regeane Vaz Guedes
- Roberto Carlos Rodrigues
- Sandra Aguera Alcova Silva
- Viviane Dziubate

Os materiais, aulas e projetos da “Robótica Paraná” foram produzidos pela Coordenação de Tecnologias Educacionais (CTE), da Diretoria de Tecnologia e Inovação (DTI), da Secretaria de Estado da Educação Paraná (SEED), com o objetivo de subsidiar as práticas docentes com os estudantes por meio da Robótica.

Este material foi produzido para uso didático-pedagógico exclusivo em sala de aula.



Este trabalho está licenciado com uma Licença
Creative Commons – CC BY-NC-SA
[Atribuição - NãoComercial - Compartilha Igual 4.0](https://creativecommons.org/licenses/by-nc-sa/4.0/)

