Aula 39– Loop gravitacional automatizado [parte III]

Módulo 4

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

Revisão Textual

Kellen Pricila dos Santos Cochinski

Projeto Gráfico e Diagramação

Edna do Rocio Becker

Apoio Técnico

Equipe UFMS

2026

Introdução

Chegamos à última etapa do nosso projeto do loop gravitacional automatizado!

Peguem o protótipo finalizado nas aulas anteriores e façam uma revisão completa: verifiquem se as pistas estão firmes, se o servomotor está bem fixado, se o sensor LDR está no lugar certo e se a bolinha percorre todo o trajeto sem travamentos. Se algo ficou pendente ou solto, este é o momento de finalizar. Com a estrutura 100% pronta, vamos dar o passo mais importante: programar o cérebro do robô para que ele funcione sozinho, detectando a bolinha e acionando o braço mecânico no momento certo.

Curiosidade maker: a alma da máquina

Nas aulas anteriores, construímos o corpo do robô — a estrutura física, os sensores e os motores. Mas um corpo sem instruções não faz nada. É aí que entra a programação: o conjunto de comandos que diz ao Arduino o que fazer, quando fazer e como reagir a cada situação. Na indústria 4.0, máquinas inteiras são controladas por código: sensores enviam dados, processadores tomam decisões e atuadores executam ações. O que vamos fazer hoje é exatamente isso — dar inteligência ao nosso loop gravitacional.

Objetivos desta aula

- Programar o microcontrolador Arduino Uno utilizando a plataforma mBlock;
- Configurar a leitura do sensor LDR para detectar a chegada da bolinha;
- Programar o microservo SG90 para elevar a bolinha até a pista superior;
- Ajustar os valores de sensibilidade e os ângulos do servo para um ciclo confiável;
- Testar e validar o funcionamento contínuo do sistema.

Lista de materiais

- Protótipo completo com os componentes eletrônicos (montado nas aulas anteriores);
- Arduino Uno (já conectado ao circuito);
- Cabo USB para conectar o Arduino ao notebook;
- Notebook com o software mBlock;
- Bolinha de pinguepongue ou similar.

Roteiro da aula

Programação passo a passo (mBlock)

Com o protótipo montado e verificado, vamos programar o Arduino para o loop gravitacional funcionar sozinho. Sigam a sequência abaixo no mBlock, conectando cada bloco na ordem correta.

Abram o mBlock e vamos preparar o ambiente de desenvolvimento. No painel esquerdo (dispositivos), cliquem em (botão azul +), busquem por `"arduino uno"` e selecionem a placa com o rostinho do robô Bino (Robótica Paraná, desenvolvida por Cleiton Rosa). Assim que a placa for carregada no painel esquerdo, cliquem no campo de texto no topo da tela, definam o nome do projeto, por exemplo: "Loop Gravitacional" e cliquem no ícone do disquete (guardar). Com a placa adicionada e o projeto salvo, o ambiente está configurado. Agora é só arrastar os blocos de programação para a área de trabalho e dar vida ao loop gravitacional!

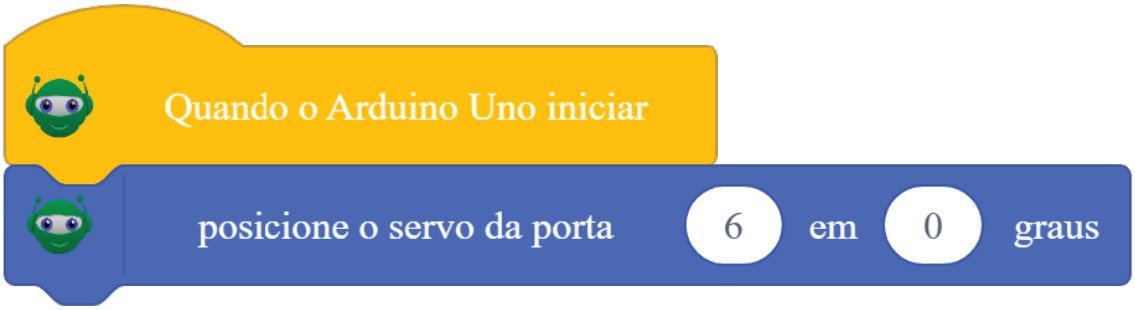
Selecione o bloco **<Quando o Arduino Uno iniciar?>**, Figura 1. Este é o ponto de partida — tudo que conectarmos abaixo dele será executado assim que o Arduino ligar ou for reiniciado.

Figura 1 - Primeiro bloco da programação

Fonte: mBlock, 2026.

Conecte abaixo o bloco (azul, bloco de comando), Figura 2. No primeiro campo, selecione a porta 6 (o pino onde conectamos o fio amarelo do servo). No segundo campo, digite 0. Isso garante que o braço comece abaixado, pronto para receber a bolinha.

Figura 2 - Calibrando o servomotor



Fonte: mBlock, 2026.

Vamos fazer o ajuste fino do braço mecânico, Figura 3. Antes de continuar, desconectem o braço (pazinha) do eixo do servomotor. Com o braço solto, conecte a placa e enviem o código para o Arduino clicando em **carregar**. O servo vai se posicionar internamente em 0°. Agora, encaixem o braço de volta no eixo do servo de forma que ele fique exatamente na posição inicial — ou seja, com o "cestinho" (o buraco onde a bolinha cai) alinhado sobre o sensor LDR. Esse ajuste manual garante que, quando o código mandar o servo para 0°, o braço esteja na posição exata que precisamos, sem necessidade de ficar recalibrando valores no software.

Figura 3 - Ajuste fino do braço mecânico

Fonte: Roberto Rodrigues, 2026.

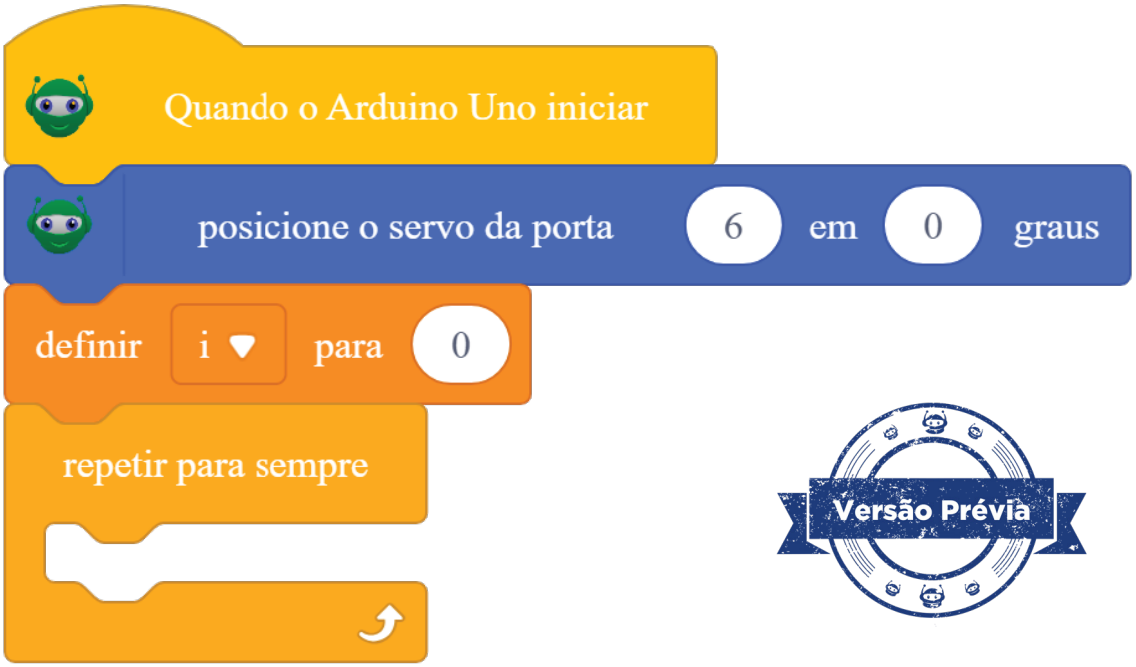
Pronto — agora sim, vamos continuar com o resto da programação!

Criar a variável de ajuste: para que o mBlock reconheça e processe o valor de calibração do sensor, precisamos primeiro declarar essa informação no sistema. No painel de blocos ao centro da tela, acessem a categoria Variáveis (cor laranja) e cliquem no botão **<Criar uma Variável>**. Na janela que se abrirá, digitem apenas a letra "i" (minúscula) no campo do nome e confirmem clicando em OK.

Após criar a variável, araste o bloco **<definir__ para__>** (laranja, bloco de variáveis), Figura 4. No primeiro campo, selecione a variável "**i**". No segundo, mantenha o valor **0**. Essa variável será usada como margem de tolerância do sensor.

Para o movimento do loop Infinito, conecte o bloco (amarelo, bloco de controle em formato de "C") **<repetir_para_sempre>**. Esse bloco mantém o programa rodando em ciclo contínuo — tudo dentro dele será executado repetidamente sem parar.

Figura 4 - Programação II



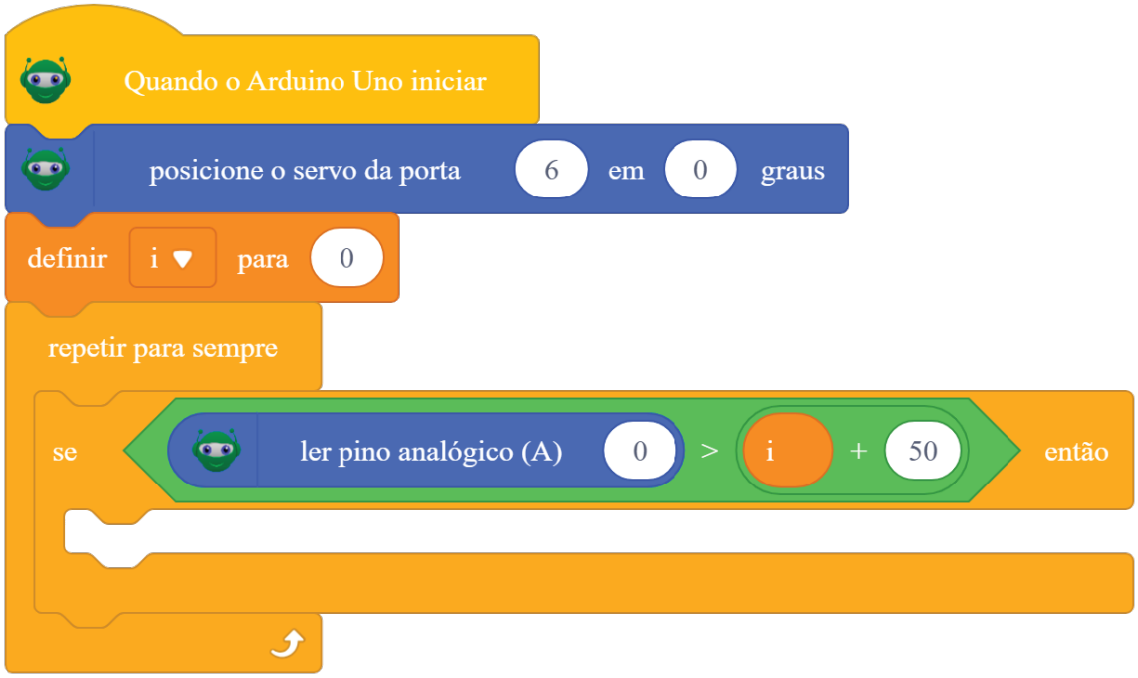
Fonte: mBlock, 2026.

Primeira condição: bolinha detectada? Dentro do bloco, repetir para sempre, conecte o bloco **<se... então>** (amarelo, bloco de controle condicional), Figura 5. Na lacuna hexagonal da condição, monte a seguinte expressão usando blocos verdes (operadores):

- Arraste o bloco **<ler pino analógico (A)__>** (verde, bloco de sensor) e selecione a porta **A0**.
- Conecteo ao lado esquerdo do bloco **<>>** (verde, operador de comparação "maior que").
- Arraste o bloco **<__ + __>** (verde, operador de soma) para o lado direito do operador.
- No primeiro campo da soma, coloque o bloco **<i>** (laranja, bloco de variável).
- No segundo campo da soma, digite **50**.

Traduzindo a condição: "se o valor de luz lido no pino **A0** for maior que **50** (com margem **i**), significa que a bolinha chegou e bloqueou a luz ambiente."

Figura 5 - Programação III

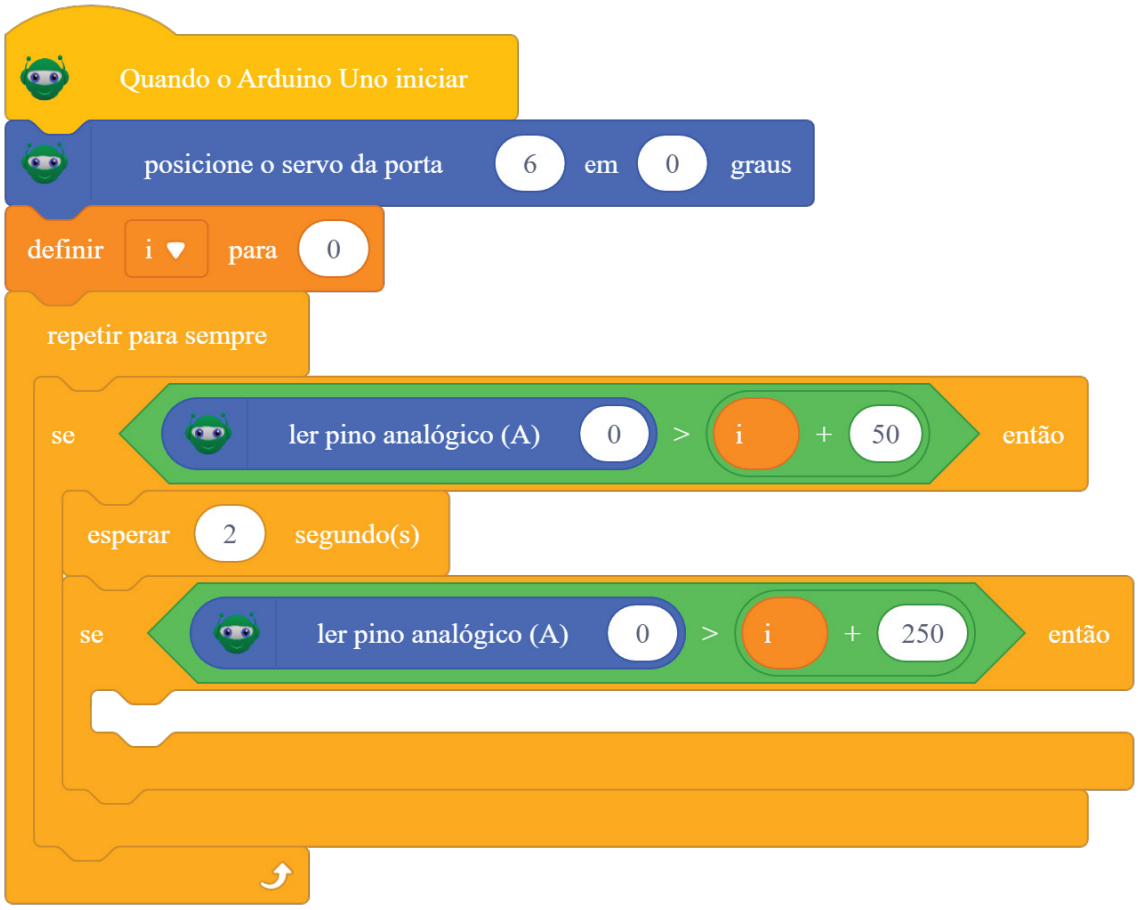


Fonte: mBlock, 2026.

Acionar o servo, Figura 6:

- Arraste o bloco **<esperar_segundo(s)>**, tempo para a bolinha percorrer todo o trajeto até o cesto novamente, coloque **2** segundos. No teste, você verá se é necessário mudar esse tempo.
- Arraste e encaixe o segundo **<se... então>**.
- Arraste o bloco **<ler pino analógico (A)__>** (verde, bloco de sensor) e selecione a porta **A0**.
- Conecteo ao lado esquerdo do bloco **<>>** (verde, operador de comparação "maior que").
- Arraste o bloco **<__ + __>** (verde, operador de soma) para o lado direito do operador.
- No primeiro campo da soma, coloque o bloco (laranja, bloco de variável).
- No segundo campo da soma, digite **250**.Dentro do segundo **<se... então>**, conecte na sequência:

Figura 6 - Programação IIII



Fonte: mBlock, 2026.

Programação final, Figura 7:

- Bloco **<posicione o servo da porta 6 em 170 graus>** — o braço sobe, empurrando a bolinha de volta ao topo da pista.
- Bloco **<posicione o servo da porta 6 em 0 graus>** — o braço volta à posição inicial, aguardando a próxima bolinha.

A sequência completa de blocos deve ficar assim:

Figura 7 - Programação completa

Fonte: mBlock, 2026.

Confira a programação completa desta aula no [mBlock Online](): Versão Prévia

Desafios

- Que tal cronometrar quantos ciclos completos o loop gravitacional consegue fazer em 1 minuto sem intervenção? Testem diferentes inclinações nas pistas e vejam se a velocidade da bolinha aumenta ou diminui.
- Que tal substituir a bolinha de pinguepongue por uma de isopor (mais leve) e depois por uma de gude (mais pesada)? Anotem as diferenças no tempo de resposta do sensor e na força necessária do servo.

E se...

O projeto não funcionar?
Não se preocupem — isso faz parte do processo de engenharia. Todo protótipo pode apresentar falhas na primeira tentativa. O importante é saber diagnosticar e ajustar. Abaixo, algumas situações comuns e o que verificar:

Verifique

Se o problema for...	**Verifique se...**
O servo não se movimenta.	Os fios estão nos pinos corretos (amarelo no pino 6, vermelho no 5V, preto no GND)? A placa Arduino está recebendo energia pelo cabo USB?
O servo se movimenta na hora errada.	O sensor LDR está posicionado exatamente no fundo do cesto? A luz ambiente está interferindo — tente cobrir o sensor com a mão para simular a sombra da bolinha.
A bolinha não chega ao cesto.	As pistas estão inclinadas o suficiente? Existe algum "degrau" ou obstáculo no trajeto? Aplique cola quente para alinhar as emendas.
O braço do servo não empurra a bolinha.	O braço foi encaixado após o upload do código (com o servo em 0°)? O ângulo de 170° é suficiente para elevar a bolinha até a próxima pista?
A bolinha quica e sai da pista.	As laterais das pistas são altas o suficiente? Reforce com tiras de papelão nas bordas.

Feedback e finalização

Chegamos ao final da construção do l**oop gravitacional automatizado**!
Cada grupo deve agora apresentar seu protótipo funcionando para a turma. Durante a apresentação, respondam:

1. **Quantos ciclos consecutivos** o robô conseguiu realizar sem intervenção manual?
2. **Qual foi o maior desafio** — a montagem mecânica, as conexões elétricas ou a lógica de programação?
3. **O que vocês mudariam** se pudessem refazer o projeto do zero?

Se ficou alguma dúvida em qualquer etapa, assistam novamente ao vídeo do protótipo funcionando — ver o **loop gravitacional** em ação ajuda a conectar cada peça, cada fio e cada bloco de código no lugar certo.

Lembremse: um projeto como este é a prova de que a Física, a Robótica e a Programação trabalham juntas para criar soluções incríveis. Vocês acabaram de construir, do zero, um sistema automatizado que funciona com energia potencial gravitacional e lógica de sensores — isso é **engenharia de verdade**!

REFERÊNCIAS

ARDUINO. **Documentação de Referência da Linguagem Arduino.** Disponível em: https://www.arduino.cc/reference/pt/. Acesso em: 20, abr. 2026.

HEWITT, Paul G. **Física conceitual.** 12. ed. Porto Alegre: Bookman, 2015.

MORAES, Paulo S. **Robótica educacional com Arduino.** São Paulo: Érica, 2019.

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

Kellen Pricila dos Santos Cochinski

Marcelo Gasparin

Michele Serpe Fernandes

Michelle dos Santos

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 didáticopedagógico exclusivo em sala de aula.