# LIÇÃO DE PROGRAMAÇÃO AVANÇADA EV3

# Controle Proporcional

Por Droids Robotics

# Por quê Controle Proporcional?

- Controle Proporcional é muito útil para FLL
- O robô move proporcionalmente – movendo mais ou menos baseado em como longe o robô está da distância do alvo
  - Por um seguidor de linha, o robô pode fazer uma curva acentuada se é mais longe da linha
- Controle Proporcional pode ser mais preciso e veloz para conseguir missões feitas!
- Cada controle proporcional consiste em dois estágios:
  1. **Computando um erro** → a que distância o robô está do trajeto
  2. **Marcando a correção** → faça o robô tomar uma ação que é proporcional ao erro (isso é por quê é chamado de controle proporcional)

# Aprendendo o quê é Proporcional

- Em nosso time, nós discutimos “proporcional” como um jogo.
- De olhos vendados. Ele ou ela tem que subir sobre a sala tão rapido quanto eles pararam exatamente na linha desenhada no chão (use fita adesiva para desenhar a linha no chão).
- O resto do time tem que dar os comandos.
- Quando seu companheiro está longe, a pessoa vendada deve mover rápido e dar grandes passos. Mas como ele se aproxima para a linha, se ele continua executando, ele ultrapassará. Então, você terá que chamar o seu amigo vendado para vir devagar e dar pequenos passos.
- Você deve programar o robô da mesma maneira!

# Aprender Como Codificar Controle Proporcional

- Aprender como usar controle proporcional, nós damos a você três diferentes exemplos:
- Seguidor Cachorro: use ultrassônico
  - Nós usamos movimentos ultrassônicos proporcionais na temporada Fúria da Natureza para fazer seguramente, nós acertamos o modelo de Base Isolado e o Sinal de Evacuação com a quantidade certa
- Seguidor de Linha: use sensor de cor
  - Nós usamos proporcional (ou inteiro PID) em todas linhas no tapete para fazer nossos movimentos mais eficientemente
- Gyro Turn: use sensor de giro
  - Nós usamos controle proporcional para fazer corretamente para conseguirmos virar a quantidade que queremos

# Aplicações para Controle Proporcional

| Aplicação | Objetivo | Erro | Correção |
|---|---|---|---|
| **Seguidor Cachorro** | Chegue ao alvo distante da parede | Como muitas medidas do da localização do alvo (curso_distância-alvo-distância) | Mova mais rápido baseado na distância |
| **Seguidor de Linha** | Fique na borda da linha | A que distância está nossas leituras de luz a partir daquela borda da linha (curso_luz-alvo_luz) | Curva mais acentuada baseada na distância da linha |
| **Gyro Turn** | Vire ao ângulo do alvo | Quantos graus nós estamos da volta alvo | Vire mais rápido baseado nos graus remanescentes |

# Ultrassônico: Seguidor Cachorro

Nós estamos tentando fazer um programa que fica 7 cm de um objeto em movimento. Esse programa usa controle proporcional.

Esse código foi escrito pelos Droids Robotics


Parte 1: Compute o erro.
Erro é a Distância Atual – Alvo
Nós optamos por 15 cm como um alvo

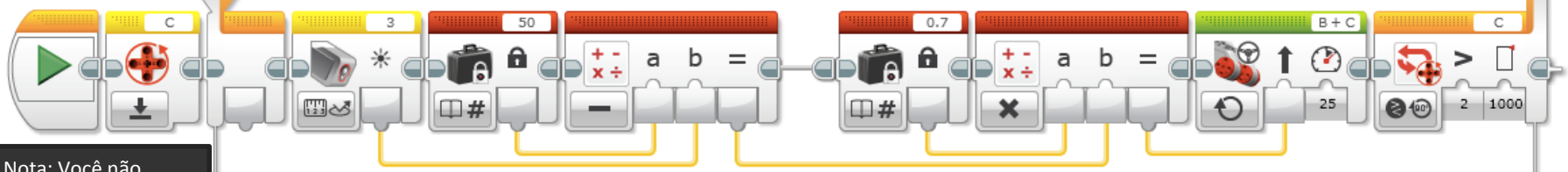
Parte 2: Compute e aplique a correção
- Nós multiplicamos o erro da Parte 1 por 10 para determinar a velocidade.
- Nós pegamos 5 para criar um intervalo razoável para nosso robô
- Exemplo:
- Leitura do sensor = 10 cm
- Erro = 3 cm
- Erro*5 = (3*5) = 15 de força
- 15 de força é uma boa velocidade para ser usado a 3 cm do alvo para nosso robô.

# Cor: seguidor de linha

Nota: Esse programa usa Sensores do Cor no Modo Luz. Isso significa que você terá que calibrar seus sensores. Por favor leia nossas lições de calibragem antes de continuar! :-)

Nós recomendamos que seu time use o seguidor de linha proporcional . Será mais razoável do que 4 seguidores de linha nesta lição. Não há seguidor de linha melhor, mas um seguidor de linha que usa “p” é um grande começo.
Um seguidor de linha proporcional muda o ângulo da curva baseada em quão longe da linha o robô está.

Cada programação proporcional deve ter 2 partes: Parte 1 computa o erro (no caso, o quão longe você está da linha) e Parte 2 computa uma correção que é proporcional ao erro (no caso o quanto vira). Você pode usar o controle proporcional com outros sensores tão bons quanto. Ele realmente trabalha bem!




Nota: Você não precisa usar um Bloco de Constante com um fio de dados. Nós só fizemos aquilo então poderia ser mais obvio do que multiplicar por uma constante de nossa escolha.

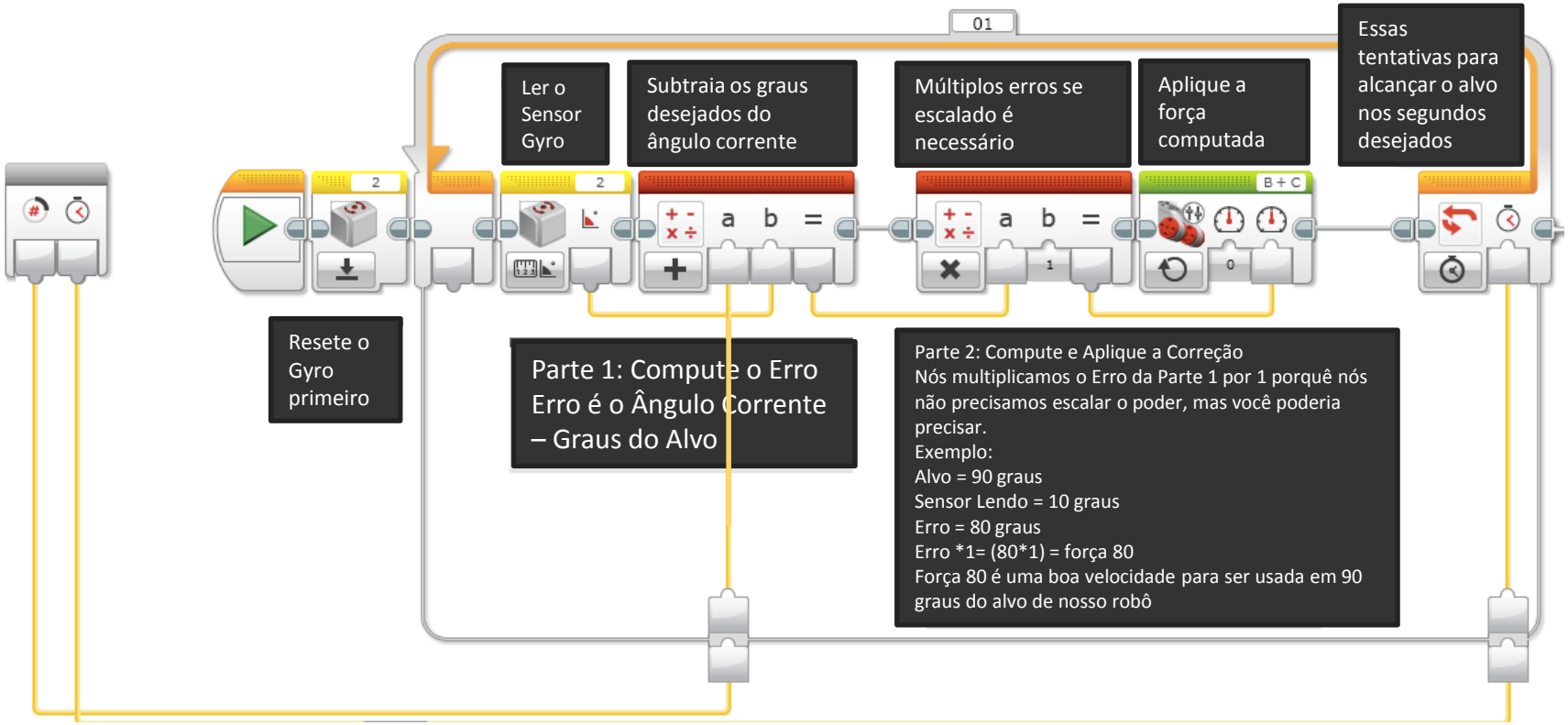
Parte 1: Compute o Erro
- Nosso objetivo é estar na borda da linha (sensor de luz = 50). O Bloco Matemático sobre computar o quão longe desligado do robô está de nosso trajeto de 50.
- O Bloco de Constante é sobre nosso alvo. Você pode mudá-lo para diferentes tipos de linha.
- Note que no pior caso, seu sensor de luz lerá 0 ou 100 (Caminho desligado da linha!). Isso dará um erro = 50 ou -50.

Parte 2: Compute e Aplique a Correção
- Nós multiplicamos o Erro da Parte 1 por 0,7 para determinar o valor de virada.
- - Nós pegamos 0,7 então aquilo quando nós temos o pior caso de erro de 50 ou -50, o Bloco de Mover em Linha Reta deve estar acima 35 ou -35 o qual é uma curva estreita.
- Você pode ajustar esse valor para fazer seu seguidor de linha se adequar ao que você precisa.

Esse seguidor de linha termina depois de 1000 degraus. Ajuste para sua necessidade.

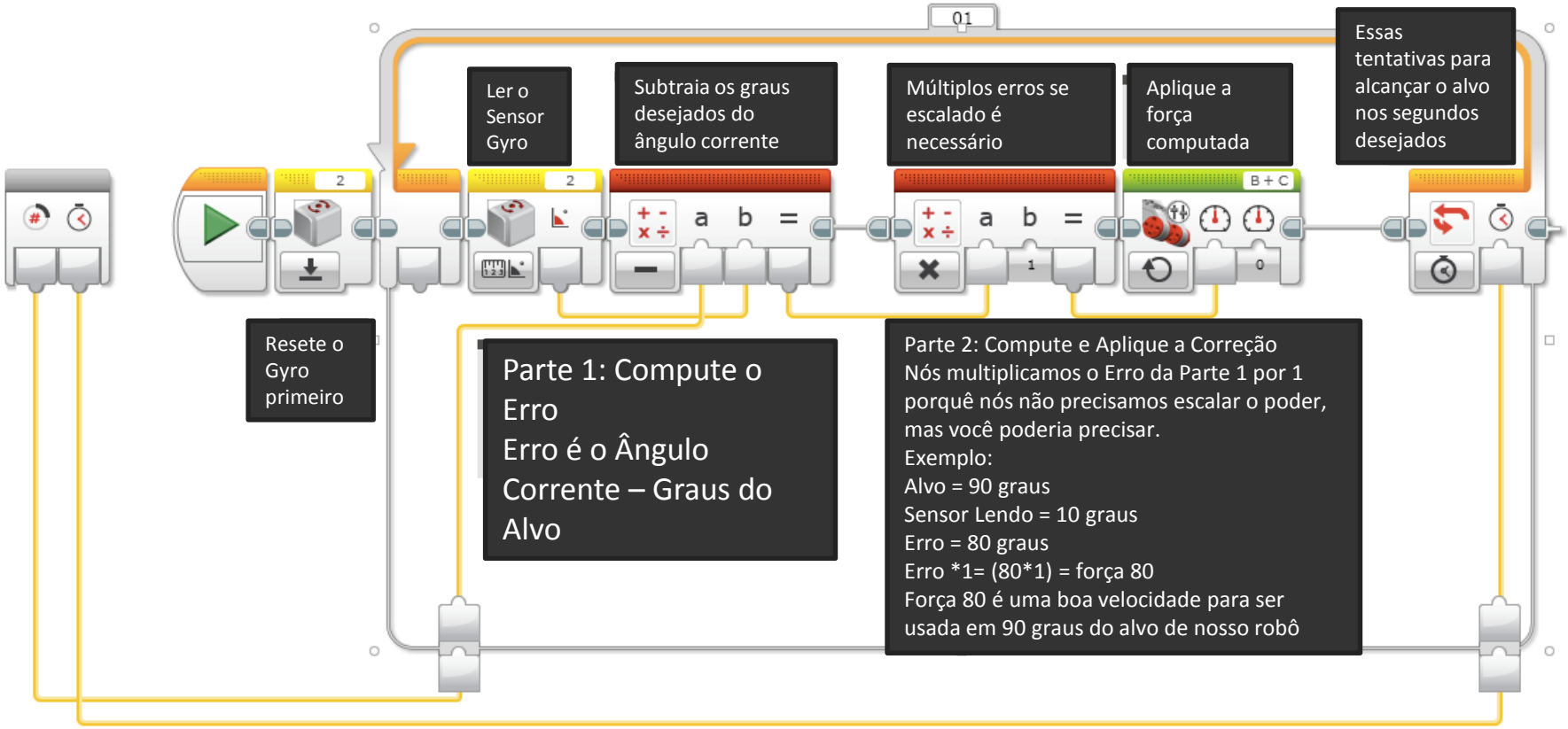
# Giro: Curva na esquerda

O objetivo desta programação é criar uma curva esquerda articulada proporcional que termina depois da soma dos segundos. Obrigado "Constructions Mavericks" pelo código original que nós modificamos para essa lição! :-)



# Gyro: Curva na direita

O objetivo desta programação é criar uma curva direita articulada proporcional que termina depois da soma dos segundo. Obrigado “Constructions Mavericks” pelo código original que nós modificamos para essa lição! :-)



# Créditos

- Esse tutorial foi criado por Sanjay Seshan e Arvind Seshan do Droids Robotics (team@droidsrobotics.org).
- Código Gyro Turn original foi provido pelos “Construction Mavericks” (frank.levine@gmail.com)
- Mais lições em www.ev3lessons.com
- Esse tutorial foi traduzido por João Victor Quintanilha, José Mateus e Bruno Leonardo.