

# Banco de testes

Um dos passos mais importantes no processo de design de engenharia e no ciclo de vida do desenvolvimento de software é o teste. Ao trabalhar com código, garantir que ele funcione sem erros e atenda ao padrão decidido na fase de planejamento do processo é crucial. Para garantir que o código está funcionando conforme o previsto, os testes precisam ser realizados. Antes de entrar nas seções de introdução à programação, [Test Bed - Blocks](#) ou [Test Bed - OnBot Java](#); é importante entender o teste, os benefícios de criar um banco de testes, os componentes necessários para as próximas seções e como usar os gamepads. Siga adiante nesta seção para aprender mais sobre o teste!

Seção	Objetivos da seção
Noções básicas de testes	Saiba por que é um dos aspectos mais importantes do desenvolvimento de software e como ele difere da solução de problemas.
Banco de testes	Por que a criação de uma plataforma de teste de atuadores e sensores pode ajudar na programação. Este banco de testes, ou algo equivalente, será usado nas seções seguintes.
Utilizando controles	Compreender as convenções de nomenclatura para programar um gamepad.

Tenha em mente que esta é a introdução ao guia básico de programação. Test Bed - Blocks e Test Bed - OnBot Java irão guiá-lo pelos fundamentos da programação com o Sistema de Controle REV.

## Noções básicas de testes

O objetivo dos testes é identificar, isolar e corrigir possíveis problemas em um projeto antes que o design seja colocado em uso. Os testes assumem diferentes formas ou fornecem diferentes métricas para diversos propósitos no design. Um mecanismo, como um atirador, por exemplo, pode ser testado para confirmar que está funcionando de forma confiável. Durante a fase de planejamento do processo de design, você deve criar várias métricas de desempenho, qualidade e confiabilidade. Quando o design é construído, ou o programa é escrito, essas métricas ajudarão a identificar se o mecanismo atende aos padrões esperados. Se os padrões de operação não forem atendidos, o problema precisa ser isolado.

Para corrigir um problema no processo de design, é necessário isolar a origem do problema. Para entender como isso funciona, considere o seguinte exemplo:

A equipe adquiriu recentemente um Control Hub e um Motor Core Hex. Eles conectam o Motor Core Hex ao Control Hub usando o cabeamento correto, mas quando tentam executar o código, o motor não se move. Qual é a razão mais provável para essa falha: O problema está no programa; o problema está no motor; o problema está no fio que conecta o motor ao Hub; o problema está no Hub.

Sem mais informações, não há uma maneira precisa de descobrir por que o motor não está funcionando. Para estreitar as possibilidades, os diferentes componentes precisam ser testados até que a causa raiz do problema seja encontrada. A prática comum é começar com um código que se sabe funcionar, como um dos códigos de exemplo no SDK. Se o motor ainda não funcionar, a próxima coisa que a equipe deve verificar é se os fios estão funcionando conforme o esperado. A equipe deve passar por cada componente, testando ou solucionando problemas, para identificar o que está funcionando e o que não está.

Uma vez que a origem do problema foi isolada, é necessário corrigi-lo. A duração da correção depende das fontes do problema e de sua profundidade. Por exemplo, se um modo operacional não estiver funcionando conforme o esperado, a correção pode ser uma mudança simples, como no arquivo de configuração ou no hardwareMap. Um problema maior que requer um redesenho, como um mecanismo que não atende às métricas de desempenho, aciona um reinício do processo de design de engenharia.

# Testes x Solução de problemas

Anteriormente, o teste foi definido como o processo de identificar, isolar e corrigir possíveis problemas durante o processo de design. Isso difere da solução de problemas, que é o processo de identificar, isolar e corrigir problemas em um mecanismo que passou pelo processo de teste e funcionou conforme o esperado.

Na seção de solução de problemas, foram usados exemplos da luz indicadora do motor de um carro. Nesse exemplo, o indicador conhecido de uma falha foi a luz do motor do carro. A luz do motor informa ao motorista que há algo errado com o carro, mas para encontrar a causa do problema, etapas de solução de problemas e diagnóstico devem ser realizadas. Para manter essa comparação, o teste é o que os engenheiros do carro usam para estabelecer as métricas do desempenho esperado do motor. Se esses padrões não forem atendidos, a luz do motor acende para alertar o motorista sobre o problema.

## Banco de testes

Um dos inconvenientes ao testar código em um sistema de componentes, como o REV Control System, é que não há garantia de que todos os componentes estão funcionando como deveriam. Por exemplo, se um motor no robô não estiver funcionando, existem várias razões potenciais para a falha. O motor, a porta do motor no Control Hub, o fio que conecta o motor à porta e o código são todas causas potenciais de falha do motor.

Se uma falha ocorrer após a montagem do robô, pode ser difícil retroceder e fazer alterações ou solucionar problemas sem ter que desmontar o robô. Uma das maneiras de se preparar para essa circunstância é criar um banco de testes antes de criar um robô.

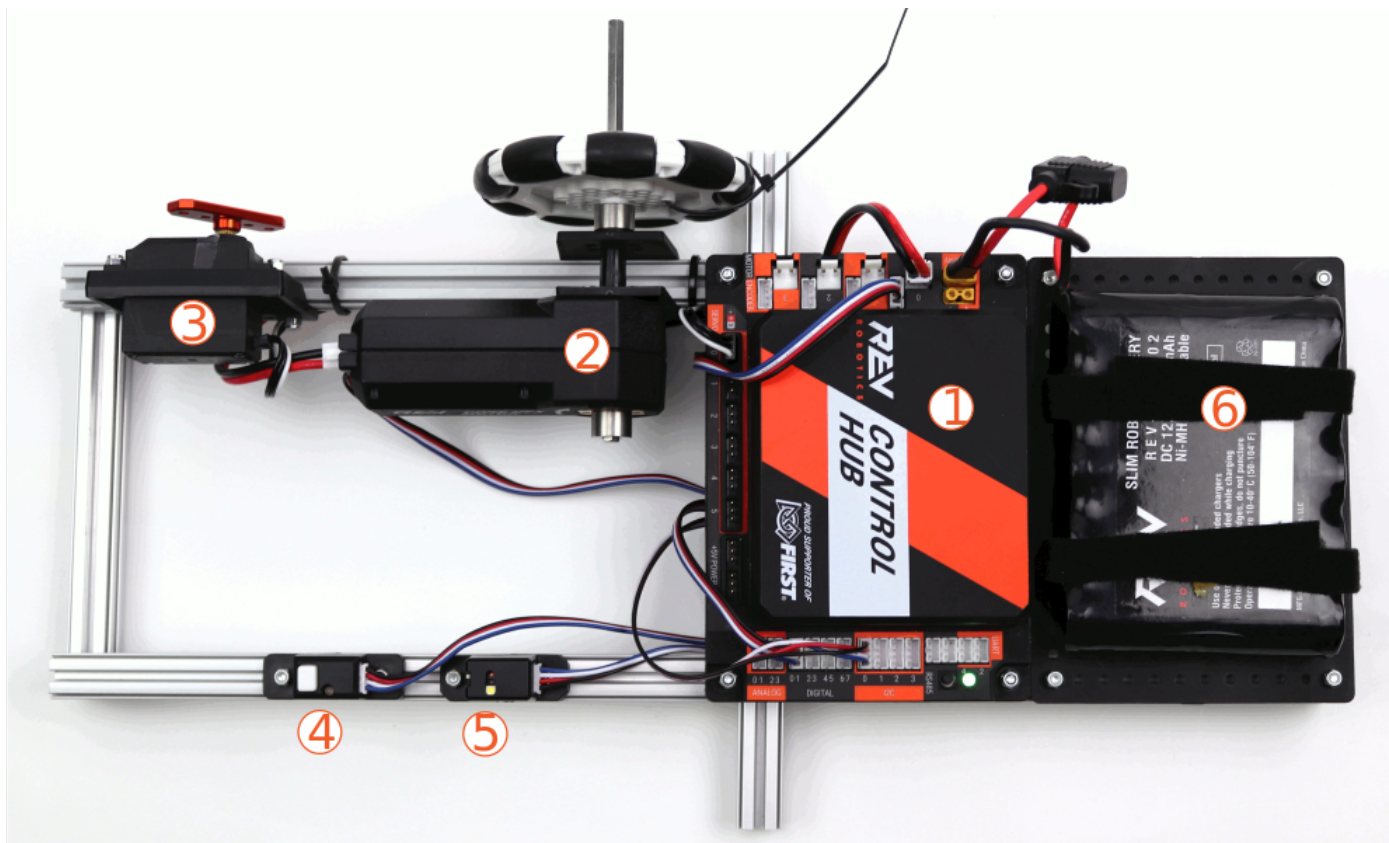
Ao testar código, não assuma que uma falha ocorre devido ao mecanismo em vez do código. Testar e solucionar problemas, embora sejam conceitos semelhantes, são fundamentalmente diferentes. Verificar o código ou usar um código conhecido que funcione sempre deve ocorrer antes de solucionar problemas com componentes como atuadores e sensores.

Um banco de testes é um ambiente de teste para componentes de hardware e software, comumente usado no mundo da engenharia. As aplicações de bancos de testes incluem uma ampla variedade de equipamentos e testes de medição. Em alguns casos, um banco de testes é um equipamento para testar um produto específico; em outros casos, é um sistema de componentes que cria um ambiente de teste. Independentemente disso, o objetivo final de um banco de testes é garantir que um componente esteja funcionando antes de ser usado para seu propósito pretendido.

Criar um banco de testes facilita o processo de solução de problemas se houver uma falha durante o teste de código. O objetivo desta seção é criar um banco de testes para testar código básico nas seções de Test Bed - Blocks e Test Bed - OnBot Java.

# Criando um banco de testes

O design de um banco de testes depende do caso de uso e dos recursos disponíveis. Por exemplo, um dos requisitos de design para o banco de testes apresentado abaixo foi a acessibilidade. Observe que a disposição dos componentes de hardware na extrusão permite que os atuadores, sensores e Control Hub sejam removidos ou trocados com facilidade.



Outra consideração importante de design para este banco de testes foi incluir os componentes comuns necessários para ensinar aos usuários o básico da programação com o REV Control System. Neste caso, os componentes foram escolhidos a partir do REV FTC Starter Kit.

1. Control Hub
2. REV Core Hex Motor
3. Smart Robot Servo
4. Touch Sensor
5. Color Sensor V3
6. Bateria

Qualquer um desses componentes do banco de testes pode ser trocado por um componente equivalente. Por exemplo, se você tiver um Expansion Hub em vez de um Control Hub. No entanto, com um Expansion Hub, pode ser necessário considerar o local para o telefone do controlador do robô.

Existem outras considerações de design menores, mas importantes, a serem feitas para um banco de testes. Por exemplo, ao adicionar um atuador a um banco de testes, considere as seguintes perguntas:

#### **Qual nível de restrição o atuador precisa?**

- Um dos benefícios de criar um banco de testes para motores ou outros atuadores é que os motores podem ser devidamente restritos durante o processo de teste. Nesse caso, fornecer suporte e restrição básicos de movimento é valioso.

## Como você poderá observar o comportamento do atuador?

- O banco de testes de exemplo usa uma roda com um enforcador de cabo de nylon (zip tie) para ajudar os usuários a visualizar o comportamento do motor. Fitas adesivas ou outros marcadores também podem ser usados. Para o propósito deste guia, um banco de testes semelhante ao exemplo pode ser construído.

# Utilizando controles

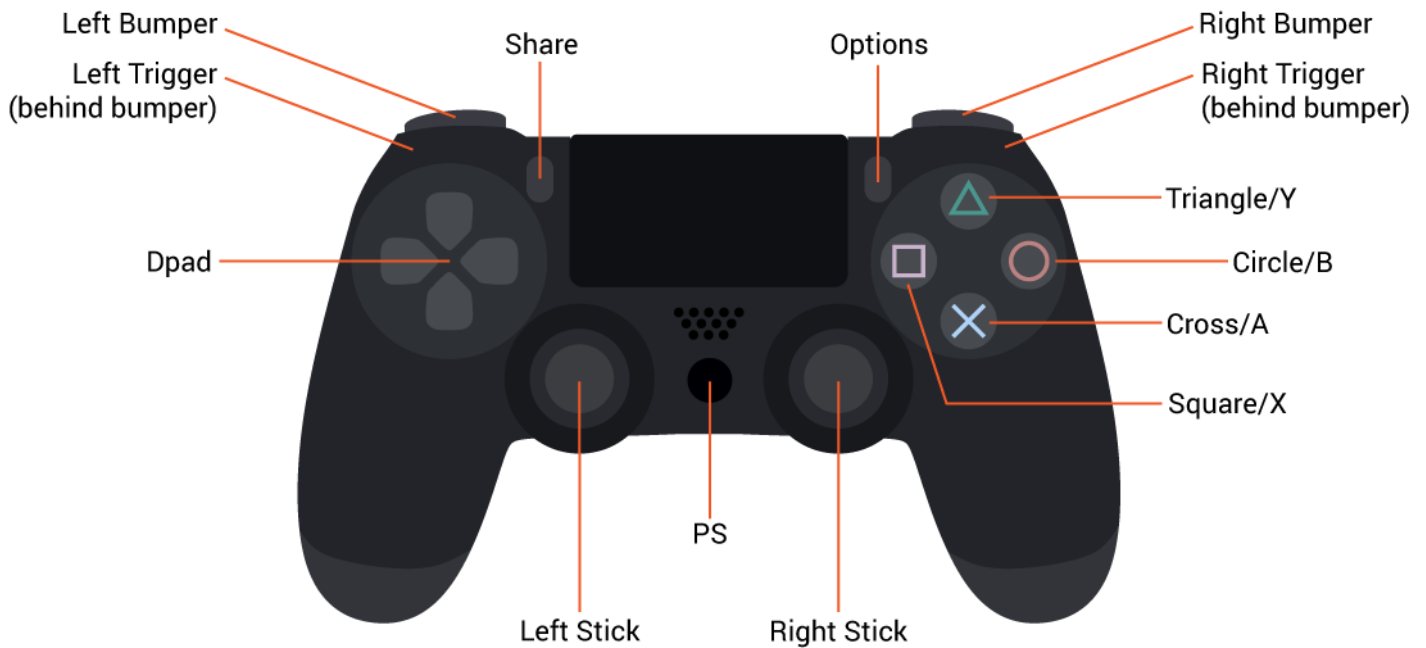
As seções do Banco de Testes destacam os componentes de robô necessários para aprender os conceitos básicos de programação utilizados nas seções Test Bed - Blocks e Test Bed - OnBot Java. No entanto, há dois componentes adicionais necessários para ter sucesso nos testes de seu código: um Driver Hub (ou dispositivo Android equivalente para Estação do Condutor) e um gamepad.

Para obter informações sobre a configuração de um Driver Hub e gamepad, por favor, visite o guia [Começando com o Driver Hub](#).



Todos os botões em um gamepad podem ser programados para uma tarefa ou comportamento específico. Ao longo do Guia Olá Robô, você encontrará várias situações onde o gamepad é utilizado. Conhecer a convenção de nomenclatura geral para os gamepads ajudará você a programá-los corretamente. O guia pressupõe que você está usando um gamepad da Logitech ou um gamepad PS4, como o Etpark Wired Controller for PS4 (REV-39-1865). Para entender como programar um gamepad, especialmente com diferenças na forma como certos botões são nomeados, consulte o gráfico e a tabela a seguir, que mostram a correspondência entre as linhas

de código e cada botão.



# Tipos de Dados

## Booleano

Dados booleanos têm dois valores possíveis: Verdadeiro e Falso. Esses dois valores também podem ser representados por Ligado e Desligado, ou 1 e 0. Botões, bumpers e gatilhos no gamepad fornecem dados booleanos ao seu robô. Por exemplo, um botão que não está pressionado retornará o valor Falso e um botão que está pressionado retornará o valor Verdadeiro.

## Float

Dados de ponto flutuante são números que podem incluir casas decimais e valores positivos ou negativos. No gamepad, os dados de ponto flutuante retornados estarão entre 1 e -1 para a posição do joystick em cada eixo. Alguns exemplos de valores possíveis são 0.44, 0, -0.29 ou -1.

---

Revisão #2

Criado 18 dezembro 2023 11:27:16 por Enzo Coutinho

Atualizado 18 dezembro 2023 11:57:53 por Enzo Coutinho