Guia de Programação para a FTC

Introdução: O Cérebro do Robô

A Programação é o componente que transforma um conjunto de peças em um robô funcional e inteligente. No FTC, a programação é realizada principalmente usando **Java** ou **Blocks**, e o sistema de controle é centrado no **REV Control Hub**. Este guia explora as melhores práticas e conceitos do básico ao avançado para maximizar o desempenho do seu robô nas fases Autônoma e Teleoperada.

1. Fundamentos do Sistema de Controle (Básico)

O sistema de controle do FTC é baseado no Android. O código é executado no **Robot Controller (Control Hub)** e o controle é feito via **Driver Station**.

1.1. Linguagens de Programação

Blocks: Uma interface de programação visual baseada em blocos, ideal para iniciantes. Permite que a equipe se concentre na lógica do robô sem se preocupar com a sintaxe complexa do Java.

Java: A linguagem padrão para o FTC. Oferece controle total sobre o hardware, acesso a bibliotecas avançadas e é a escolha para equipes que buscam otimização de desempenho e algoritmos complexos. O desenvolvimento é feito no **Android Studio**.

1.2. Estrutura Básica do Código (OpMode)

Todo o código do robô é escrito em classes chamadas **OpModes** (Operation Modes).

LinearOpMode: O código é executado sequencialmente, de cima para baixo. É o tipo mais comum para a fase Autônoma.

TeleOp: O código é executado em um loop contínuo (`while (opModeIsActive())`) para responder continuamente às entradas do joystick.

Dica de Programação (Básico)	Descrição
Inicialização Segura	Sempre use o método `waitForStart()` para garantir que o robô só comece a se mover após o início oficial da partida.
Nomenclatura Clara	Nomeie motores, servos e sensores de forma clara no arquivo de configuração e no código (ex: `motorGarra`, `sensorCor`).

2. Programação Teleoperada (TeleOp) (Intermediário)

A fase TeleOp é onde os motoristas controlam o robô. O objetivo é ter um controle suave, responsivo e intuitivo.

2.1. Controle de Drivetrain

Mapeamento de Joystick: O mapeamento mais comum usa um joystick para o movimento (frente/trás, strafe) e o outro para a rotação.

Controle Suave (Ramping): Evite que o robô vá de 0% a 100% de potência instantaneamente. Implemente um **ramping** (aceleração gradual) para evitar derrapagens e desgaste excessivo dos componentes.

Funções de Potência: Use funções não-lineares (ex: potência = `power * power`) para dar aos motoristas mais controle em baixas velocidades, mantendo a capacidade de velocidade máxima.

2.2. Mecanismos e Automação Parcial

Controle de Posição (Servos): Use servos para mecanismos de garra ou posicionamento. Defina posições de forma constante (ex: `POSICAO_ABERTA = 0.5`, `POSICAO_FECHADA = 0.9`) em vez de usar números mágicos.

Automação de Um Botão: Para mecanismos complexos (ex: um elevador que precisa subir, mover um braço e soltar uma peça), use um único botão do joystick para iniciar uma sequência de ações. Isso reduz a carga cognitiva do motorista.

3. Programação Autônoma (Autonomous)

(Avançado) A fase Autônoma é a mais desafiadora e onde os algoritmos avançados brilham.

3.1. Navegação Precisa (Odometria e Sensores)

Odometria (Road Runner): O uso de rodas de odometria (encoder wheels) para rastrear a posição do robô no campo é o padrão ouro para navegação precisa. A biblioteca **Road Runner** é a mais utilizada para planejar trajetórias suaves e otimizadas.

Visão Computacional (OpenCV e EasyOpenCV): Use a câmera do Control Hub ou uma webcam para identificar peças do jogo, cores ou padrões no campo. Isso permite que o robô tome decisões dinâmicas no início da partida.

IMU (Unidade de Medição Inercial): Use o IMU interno do Control Hub para manter o robô em linha reta e para rotações precisas.

3.2. Estruturas de Código Avançadas

Máquina de Estados (State Machine): Em vez de usar longos `sleep()` s, organize o código autônomo em uma **Máquina de Estados**. Cada estado representa uma ação (ex: `IR_PARA_PONTO_A` , `COLETAR_PECA` , `DEPOSITAR_PECA`). Isso torna o código mais robusto e fácil de depurar.

Programação Orientada a Objetos (POO): Crie classes separadas para o Drivetrain, o Elevador e a Garra. Isso permite que o código seja reutilizável e mais limpo.

Dica de Programação (Avançado)	Descrição
Calibração Constante	Calibre todos os sensores (cor, distância, IMU) antes de cada execução autônoma para garantir a precisão.

Depuração Remota	Use o logcat do Android Studio ou a interface web do Control Hub para depurar o código remotamente e visualizar o estado das variáveis em tempo real.
Controle PID	Implemente um controlador **PID (Proporcional-Integral-Derivativo)** para motores e servos que precisam de controle de posição ou velocidade muito preciso (ex: braços de elevação).

4. Otimização e Boas Práticas

4.1. Configuração de Hardware

Arquivo de Configuração: Garanta que o arquivo de configuração de hardware no Control Hub corresponda exatamente aos nomes e portas usados no código. Um erro aqui é a causa mais comum de falhas.

Atualizações de Firmware: Mantenha o firmware do Control Hub e dos módulos de expansão sempre atualizado para garantir a estabilidade e acesso aos recursos mais recentes.

4.2. Colaboração e Versionamento

Git e GitHub: Use o **Git** para controle de versão. Isso permite que vários programadores trabalhem no código simultaneamente e garante que a equipe possa reverter para uma versão anterior estável em caso de erros.

Comentários e Documentação: Comente o código de forma clara, explicando a lógica por trás de funções complexas. Use o **Javadoc** para documentar classes e métodos.

Conclusão

A Programação no FTC é uma jornada contínua de aprendizado e otimização. Dominar o Java, entender a estrutura do OpMode e aplicar técnicas avançadas como Odometria e Máquinas de Estado transformará seu robô de um simples controle remoto em um sistema autônomo de alta performance.