FTC Programming Guide

For Beginners

What is FTC Programming?

Programming is the **"brain"** of your robot, giving it life and purpose on the competition field. It's how you tell your robot what to do, when to do it, and how to react to its environment.

Two main modes:

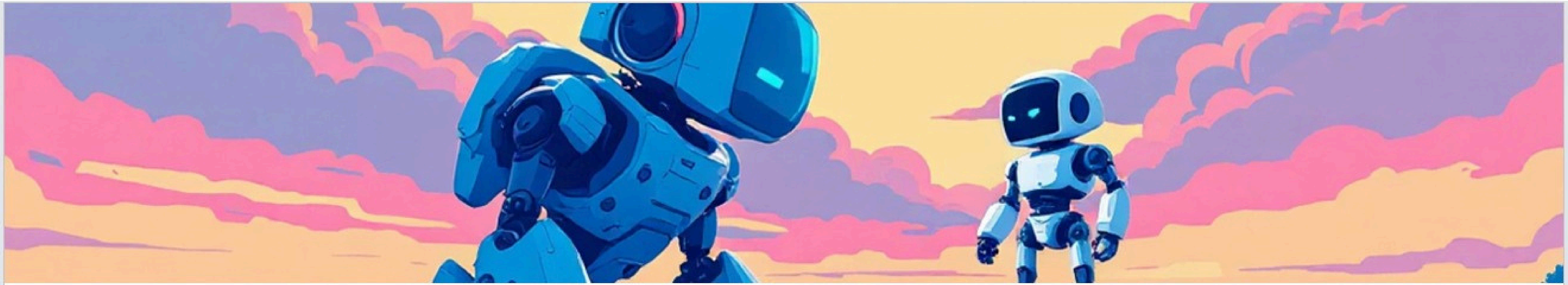
TeleOp and Autonomous

Controls movement and mechanisms

Your code controls every aspect, from precise movements to operating complex mechanisms like arms and grippers. Without programming, your robot is just a collection of parts.

```
            }
            else if (gamepad1.b) {
                intake.setPower(INTAKE_DEPOSIT);
            }



            /* Here we implement a set of if else statements to set our
            We check to see if a specific button is pressed, and then m
            intake and wrist) to match. For example, if we click the
            to start collecting. So it moves the armPosition to the
            it folds out the wrist to make sure it is in the corre
            turns the intake on to the COLLECT mode.*/

            if(gamepad1.right_bumper){
                /* This is the intaking/collecting arm posit
                armPosition = ARM_COLLECT;
                wrist.setPosition(WRIST_FOLDED_OUT);
                intake.setPower(INTAKE_COLLECT);
                }

                else if (gamepad1.left_bumper){
                    /* This is about 20° up from t
                    Note here that we don't set
                    select this "mode", this m
                    they were doing before w
                    armPosition = ARM_CLEAR_
                }

                else if (gamepad1.y){
                    /* This is the co
                    armPosition = AR
                }

                else if (gamepad
```

STRIKE



TeleOp vs. Autonomous

FTC matches are divided into two distinct periods, each requiring a different programming approach.

TeleOp Mode

This is the driver-controlled period where you use a gamepad to operate the robot. Your code translates joystick and button inputs into robot actions, allowing for real-time strategic play.

Autonomous Mode

In this period, the robot acts completely independently. You pre-program a sequence of actions for the robot to perform, such as moving to specific locations, picking up objects, or scoring points, all without human input.

OpMode Structure

An OpMode is the core of your robot's program. It defines the specific set of instructions your robot will follow. Every OpMode has a clear structure to ensure organised and predictable behaviour.

01

The OpMode

Your main program file, containing all instructions for a specific robot behaviour (e.g., "Red_Autonomous" or "Driver_Controlled").

02

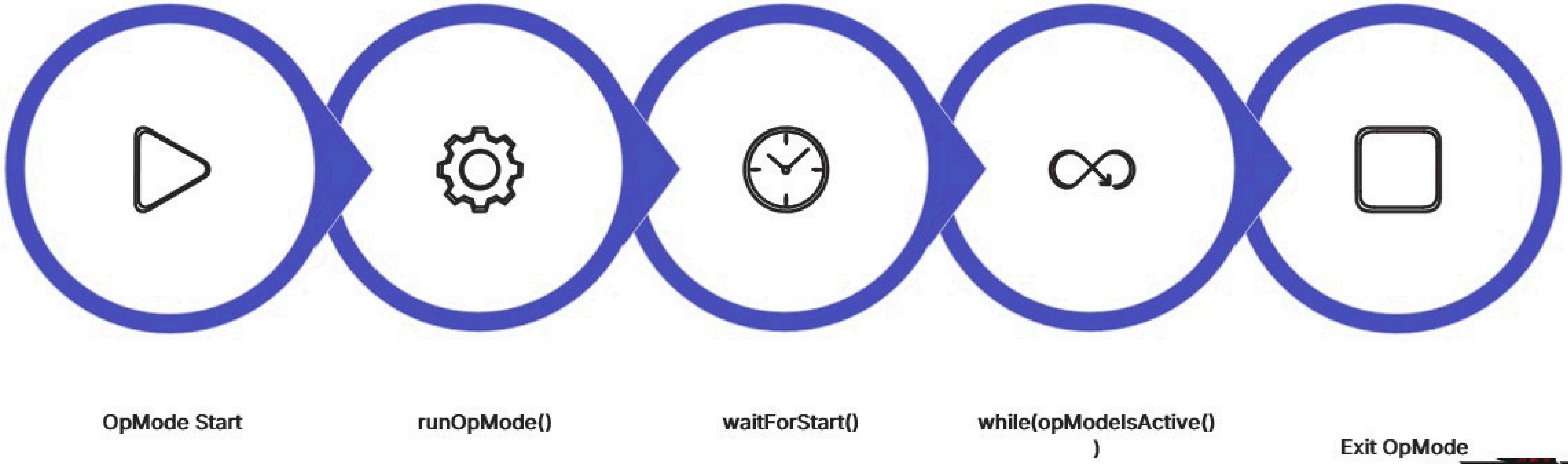
runOpMode() Method

This is where your robot initialises and waits for the match to start. All setup code runs here, but the robot won't move until `waitForStart()` is called.

03

while(opModeIsActive()) Loop

Once the match begins, the code inside this loop continuously executes. This is where you put your robot's primary actions and decision-making logic for both TeleOp and Autonomous.



HardwareMap

The `HardwareMap` is your code's bridge to the physical components of your robot. It allows you to name and reference motors, servos, and sensors from within your program.

Think of it as a dictionary that connects the descriptive names you use in your code (e.g., "left_motor") to the actual hardware ports on your Control Hub or Expansion Hub.

- Connects code to real hardware
- Used for motors, servos, and sensors
- Ensures your code knows which physical component to control

Mecanum Drive Basics

Mecanum wheels offer incredible maneuverability, allowing your robot to move in any direction without turning its body. This can be a huge advantage in dynamic game scenarios.

1 **Forward & Backward**

All wheels spin in the same direction.

2 **Strafe (Side-to-Side)**

Wheels spin in opposing diagonal pairs, allowing lateral movement.

3 **Rotation**

Wheels on one side spin opposite to the other, making the robot pivot.

4 **Combined Movements**

Mix and match inputs for diagonal or curved paths.

STRIKE

Speed Control

Controlling your robot's speed is crucial for precision and strategy. Sometimes you need raw power; other times, delicate movements are key.

Turbo Mode

Full power for quick travel across the field or pushing opponents. Use when speed is paramount.

Slow Mode

Reduced power for fine adjustments, aligning with game elements, or precise scoring. Essential for accuracy.

Implementing speed control with gamepad buttons allows drivers to switch between modes instantly, adapting to different game situations.

Servos and Mechanisms

Beyond driving, your robot needs to interact with game elements. Servos are the workhorses for these mechanisms, providing controlled angular movement.

Claw Mechanism

Used to grab and release game objects. A servo can precisely open and close the claw.

Lift System

Raises and lowers objects or robot components to different heights for scoring or navigating obstacles.

Turret

Allows mechanisms to rotate horizontally, expanding the robot's reach and aiming capabilities.

Programming servos involves setting specific positions (angles) for them to move to. This allows for repeatable and accurate mechanism operation.

STRIKE

Autonomous Basics

Crafting an effective autonomous period requires precise, pre-programmed movements. This is achieved using encoders on your motors.

Encoder-Based Movement

Encoders are sensors on motors that count rotations, allowing the robot to know exactly how far its wheels have turned. This enables accurate distance-based movement.

RUN_TO_POSITION

This motor mode instructs a motor to run until it reaches a specific encoder target position. It's fundamental for precise autonomous driving.

Initialize

Reset Encoders

Drive Forward

Turn & Act

Summary & Beyond

You've taken the first step into the exciting world of FTC programming!

Key Concepts

We've covered the robot's brain, different control modes, OpMode structure, hardware communication, and basic movement control.

Structure is Key

Understanding the foundational elements makes complex programming tasks much more manageable and less daunting.

Keep Learning!

Programming is an iterative process. Experiment, troubleshoot, and continually refine your code to achieve robot excellence.

Programming becomes easy with structure. Now go forth and create!

STRIKE