

Writing the code for a PIDSubsystem in C++

PIDSubsystems use feedback to control the actuator and drive it to a particular position. In this example we use an elevator with a 10-turn potentiometer connected to it to give feedback on the height. The skeleton of the PIDSubsystem is generated by the RobotBuilder and we have to fill in the rest of the code to provide the potentiometer value and drive the motor with the output of the imbedded PIDController.

Setting the PID constants

Setting the PID constants

Make sure the Elevator PID subsystem has been created in the RobotBuilder. In the case of our elevator we use a proportional constant of 6.0 and 0 for the I and D terms. Once it's all set, generate C++ code for the project using the Export menu or the C++ toolbar menu.

Add constants for the Elevator preset positions

Add constants for the Elevator preset positions

Elevator constants define potentiometer voltages that correspond to fixed positions on the elevator. These values can be determined using the print statements, the LiveWindow or SmartDashboard.

Initialize the elevator position in the Elevator constructor

Initialize the elevator position in the Elevator constructor

Set the elevator initial position so when the robot starts up it will move to that position. This will get the robot to a known starting point. Then enable the PIDController that is part of the PIDSubsystem. The elevator won't actually move until the robot itself is enabled because the motor outputs are initially off, but when the robot is enabled, the PID controller will already be running and the elevator will move to the "STOW" starting position.

Writing the code for a PIDSubsystem in C++

Set the ReturnPIDInput method to return voltage

Set the ReturnPIDInput method to return voltage

By default the RobotBuilder generated ReturnPIDInput() method returns the potentiometer value in raw units (a value between 0-1023). Since the setpoints are all in voltages (0.0 - 5.0V) the ReturnPIDInput() method must be changed to return volts to match.

That's all that is required to create the Elevator PIDSubsystem in C++. To operate it with commands to actually control the motion see: [Operating a PIDSubsystem from a command in C++](#).