Proportional–Integral–Derivative Controller (PID)

What is PID?

PID is a type of control loop that takes in sensor data to control a robotic device. In this case I used gyroscopic data, for PID turning. Example: You tell the robot to turn 90° (π/2 Radians) the robot might not be fully acuate while turning for a number of reasons: For example, if there is a movable object in the trajectory of the robot and they collide, the robots code is unaware of this and moves it but the collision caused the robot to only turn 85° instead of the desired 90°. This is a problem because now the entire robots course is off. With PID controls the robot would know the course if off and it would keep turning until it reached 90° from the original angle of rotation. In code you can specify the amount it's accurate to. For example, if you set the target value to 0.5°, the robot will correct itself until it's 0.5° within the desired rotation.

Example of PID.

Let's say you have an oven that has a heating element and temperature sensor. If the temperature of the heating element is 5°F and you want to heat it to 10°F your oven should keep heating it. What happens when it gets to 10°F? Your oven should keep heating until it reaches 11°F and then it should shut off. The oven should repeat this in an infinite cycle until the oven is turned off by the user.

Example:

Temperature 5°F Keep heating.

Temperature 11°F Shut off.

Temperature 9°F Keep heating.

Temperature 11°F Shut off.

How did I learn PID?

In the "Sources" folder there is an introduction to PID PDF document. That is the main source of information that helped me learn about PID controllers. In addition, I learned from the Vex Robotics Competition Discord server that helped with trouble shooting errors.

Code Examples

Written using VEX API and C++. Disclaimer: this was the last code contributed to the GitHub repository and not the last verified working copy there might be unknown errors/mistakes.

```
 #include <vex.h>
#include <AdvancedMovement.h>
#include <utils.h>



bool isInstalled; // Is the gyroscope connected?
```

```
motor_group motors(LeftDriveSmart, RightDriveSmart);
// Motors (In group for better control)

void AdvancedMovement::setup() // YOU NEED TO CALL THIS BEFORE USING!!!!!!
{
  Inertial.calibrate(); // Calibrates the gyroscope.
  isInstalled = Inertial.installed(); // Checks if the gyroscope is installed.
  if(!isInstalled) // If it isn't installed do line 42.
    printlnColored((char *) "Inertial not installed", red);
}

void AdvancedMovement::turnPID(int angle)
{
  double kP = 0;
  double kI = 0;
  double kD = 0;
  int integral = 0;
  int error = 0;
  int prevError = 0;
  int power = 0;
  int derivative = 0;

  // Loop while the absolute value of the error is greater than 0.5
  while (abs(error) > 0.5)
  {
    error = angle - Inertial.rotation(deg);
    integral = integral + error;
    if (error == 0 || error > angle)
      integral = 0;
    if (error > 360)
      integral = 0;

    derivative = error - prevError;

    prevError = error;

    power = error * kP + integral * kI + derivative * kD;

    LeftDriveSmart.spin(fwd, power, volt); // Moves motor fwd.

    RightDriveSmart.spin(fwd, -1 * power, volt);
    // Moves motor opposite of other motor.

    wait(15, msec); // Waits 15 milliseconds.
  }
}
```

GitHub

GitHub was made by me for the robotics team (Not required by VEX or the robotics club)
(GitHub is a place to share code)