**Improving the Beginner’s PID – Introduction**

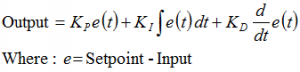
In conjunction with the release of the new [Arduino PID Library](http://www.arduino.cc/playground/Code/PIDLibrary) I’ve decided to release this series of posts. The last library, while solid, didn’t really come with any code explanation. This time around the plan is to explain in great detail why the code is the way it is. I’m hoping this will be of use to two groups of people:

* People directly interested in what’s going on inside the Arduino PID library will get a detailed explanation.
* Anyone writing their own PID algorithm can take a look at how I did things and borrow whatever they like.

It’s going to be a tough slog, but I think I found a not-too-painful way to explain my code.  I’m going to start with what I call “The Beginner’s PID.”  I’ll then improve it step-by-step until we’re left with an efficient, robust pid algorithm.

### The Beginner’s PID

Here’s the PID equation as everyone first learns it:

[](http://brettbeauregard.com/blog/wp-content/uploads/2011/03/pidalgorithm.png)

This leads pretty much everyone to write the following PID controller:

|  |  |
| --- | --- |
|  | pid.jpg |

Compute() is called either regularly or irregularly, and it works pretty well. This series isn’t about “works pretty well” though. If we’re going to turn this code into something on par with industrial PID controllers, we’ll have to address a few things:

1. [**Sample Time -**](http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/improving-the-beginner%E2%80%99s-pid-sample-time) The PID algorithm functions best if it is evaluated at a regular interval. If the algorithm is aware of this interval, we can also simplify some of the internal math.
2. [**Derivative Kick -**](http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/improving-the-beginner%E2%80%99s-pid-derivative-kick) Not the biggest deal, but easy to get rid of, so we’re going to do just that.
3. [**On-The-Fly Tuning Changes -**](http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/improving-the-beginner%E2%80%99s-pid-tuning-changes) A good PID algorithm is one where tuning parameters can be changed without jolting the internal workings.
4. [**Reset Windup Mitigation -**](http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/improving-the-beginner%E2%80%99s-pid-reset-windup)We’ll go into what Reset Windup is, and implement a solution with side benefits
5. [**On/Off (Auto/Manual) -**](http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/improving-the-beginner%E2%80%99s-pid-onoff) In most applications, there is a desire to sometimes turn off the PID controller and adjust the output by hand, without the controller interfering
6. [**Initialization -**](http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/improving-the-beginner%E2%80%99s-pid-initialization) When the controller first turns on, we want a “bumpless transfer.” That is, we don’t want the output to suddenly jerk to some new value
7. [**Controller Direction -**](http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/improving-the-beginners-pid-direction) This last one isn’t a change in the name of robustness per se. it’s designed to ensure that the user enters tuning parameters with the correct sign.

Once we’ve addressed all these issues, we’ll have a solid PID algorithm. We’ll also, not coincidentally, have the code that’s being used in the lastest version of the Arduino PID Library. So whether you’re trying to write your own algorithm, or trying to understand what’s going on inside the PID library, I hope this helps you out. Let’s[GET STARTED[http://cdncache-a.akamaihd.net/items/it/img/arrow-10x10.png](http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/#76828695)](http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/#76828695).  
[Next >>](http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/improving-the-beginner%E2%80%99s-pid-sample-time)

UPDATE: In all the code examples I’m using doubles. On the Arduino, a double is the same as a float (single precision.) True double precision is WAY overkill for PID. If the language you’re using does true double precision, I’d recommend changing all doubles to floats.