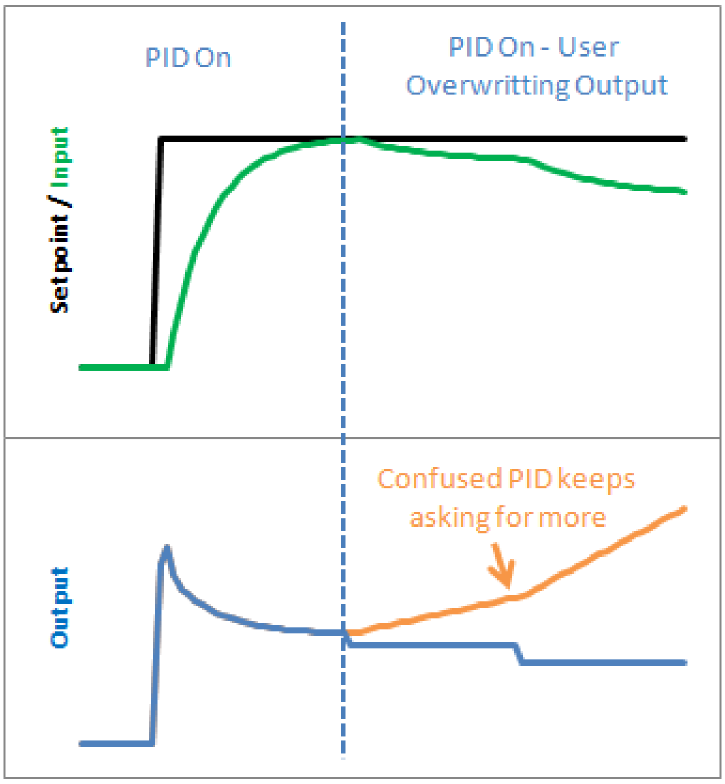
« Improving the Beginner’s PID: Reset Windup
Improving the Beginner’s PID: Initialization »

# Improving the Beginner’s PID: On/Off

(This is Modification #5 in a larger series on writing a solid PID algorithm)

## The Problem

As nice as it is to have a PID controller, sometimes you don’t care what it has to say.



Let’s say at some point in your program you want to force the output to a certain value (0 for example) you could certainly do this in the calling routine:

```
void loop()
{
Compute();
Output=0;
}
```

This way, no matter what the PID says, you just overwrite its value. This is a terrible idea in practice however. The PID will become very confused: “I keep moving the output, and nothing’s happening! What gives?! Let me move it some more.” As a result, when you stop over-writing the output and switch back to the PID, you will likely get a huge and immediate change in the output value.

## The Solution

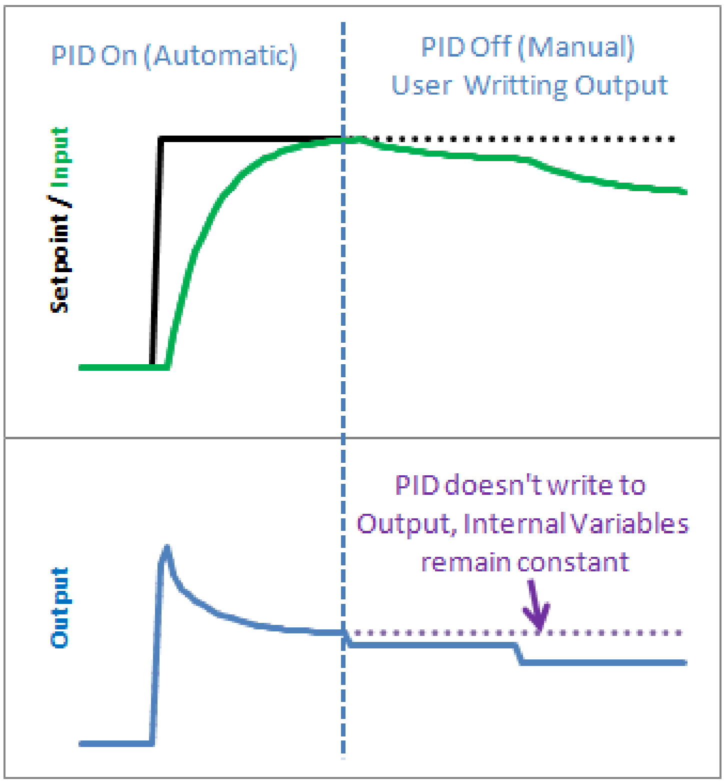
The solution to this problem is to have a means to turn the PID off and on. The common terms for these states are “Manual” (I will adjust the value by hand) and “Automatic” (the PID will automatically adjust the output). Let’s see how this is done in code:

## The Code

```
/*working variables*/
unsigned long lastTime;
double Input, Output, Setpoint;
double ITerm, lastInput;
double kp, ki, kd;
int SampleTime = 1000; //1 sec
double outMin, outMax;
bool inAuto = false;

#define MANUAL 0
#define AUTOMATIC 1

void Compute()
{
   if(!inAuto) return;
   unsigned long now = millis();
   int timeChange = (now - lastTime);
   if(timeChange>=SampleTime)
   {
      /*Compute all the working error variables*/
      double error = Setpoint - Input;
      ITerm+= (ki * error);
      if(ITerm> outMax) ITerm= outMax;
      else if(ITerm< outMin) ITerm= outMin;
      double dInput = (Input - lastInput);

      /*Compute PID Output*/
      Output = kp * error + ITerm- kd * dInput;
      if(Output > outMax) Output = outMax;
      else if(Output < outMin) Output = outMin;

      /*Remember some variables for next time*/
      lastInput = Input;
      lastTime = now;
   }
}

void SetTunings(double Kp, double Ki, double Kd)
{
  double SampleTimeInSec = ((double)SampleTime)/1000;
   kp = Kp;
   ki = Ki * SampleTimeInSec;
   kd = Kd / SampleTimeInSec;
}

void SetSampleTime(int NewSampleTime)
{
   if (NewSampleTime > 0)
   {
      double ratio  = (double)NewSampleTime
                      / (double)SampleTime;
      ki *= ratio;
      kd /= ratio;
      SampleTime = (unsigned long)NewSampleTime;
   }
}

void SetOutputLimits(double Min, double Max)
{
   if(Min > Max) return;
   outMin = Min;
   outMax = Max;

   if(Output > outMax) Output = outMax;
   else if(Output < outMin) Output = outMin;

   if(ITerm> outMax) ITerm= outMax;
   else if(ITerm< outMin) ITerm= outMin;
}

void SetMode(int Mode)
{
  inAuto = (Mode == AUTOMATIC);
}
```

A fairly simple solution. If you're not in automatic mode, immediately leave the Compute function without adjusting the Output or any internal variables.

## The Result



It's true that you could achieve a similar effect by just not calling Compute from the calling routine, but this solution keeps the workings of the PID contained, which is kind of what we need. By keeping things internal we can keep track of which mode were in, and more importantly it let's us know when we change modes. That leads us to the next issue…

Next >>

Tags: Arduino, Beginner's PID, PID

This entry was posted on Friday, April 15th, 2011 at 3:05 pm and is filed under Coding, PID. You can follow any responses to this entry through the RSS 2.0 feed. You can leave a response, or trackback from your own site.

## 10 Responses to "Improving the Beginner's PID: On/Off"

1. *Autopilot* says:
   July 26, 2011 at 6:15 pm

   Hello,

   Thanks for your tutorial but isn't it a good idea to reset integral sum before resuming automatic control after manual mode? Let's imagine controller had accumulated some output in integral term, then it's switched to manual mode for some time, then again back to automatic. In that case controller would try to use old accumulated value (based on history) under perhaps different conditions. What's your view?

2. *Brett* says:
   July 27, 2011 at 6:34 am

   Definitely agree on that. That's covered in the next section: Initialization.

3. *Adrian* says:
   August 19, 2011 at 8:35 am

   I believe there is an error in the definition of the timeChange variable, on line 17. I have found that it is possible that timeChange can become negative when switching between MANUAL and AUTOMATIC. This was stopping the subsequent if statement from return to correct value , resulting in the PID not updating.

   Changing the definition from int to unsigned long appears to have resolved this issue for me.

   Oh, and thanks a heap for both the Arduino PID library, and your excellent documentation.

   Best regards,
   Adrian

4. *Brett* says:
   August 19, 2011 at 8:50 am

   @Adreian
   yes that is indeed an error. there's a conversation on the sample time page that offers a solution.

5. *Adrian* says:
   August 19, 2011 at 9:20 am

   Thanks for the fast response confirming my hack solution might actually work. : )

   Cheers, Adrian

6. *Kyle* says:
   April 9, 2012 at 1:25 pm

   One 'complaint', line 15, if(!inAuto) return;, is bad programming. A more correct way would be while(inAuto) { … }. Also, I was trying to think of a way to fix your last change that was written the same way, if(Min > Max) return;. You could rewrite it the same way. Both changes would require you to make the test false at the end of the while loop, so it would drop out properly. Yes, it is more code, but, it is written more correctly.

7. *Kyle* says:
   April 9, 2012 at 11:32 pm

   Stupid me. I should have suggested you just change your logic instead of changing the language completely. So, rather than if(!inAuto) return; you should do if (inAuto) { … }. The same goes for the if (Min > Max) return;. It should be if (Min < Max) { … }. The preceding follows good programming practices.

8. *Anthony* says:
   April 30, 2012 at 1:32 pm

   Great article!

   I just wanted to respond to Kyle's comments about the "bad" programming style of having multiple returns in your function. There's nothing wrong with that. In fact, if multiple conditions decide whether to exit a function, it's sure as hell a whole lot nicer to have a return there instead of a large amount of nested ifs. In this case, the return at line 15 makes it very clear what will happen in manual mode, so I prefer it to the if clause around the whole rest of the function (as Kyle suggests). Just my 2 cents.

   Cheers, Anthony

9. *Terry G* says:
   April 14, 2016 at 8:25 am

   I think there is a error with the integral term of the PID controller. When used in conjunction with the kp term the resulting PI controller eliminates the error as desired. However when the integral term is set to zero the output remains constant, that is the output does not revert back to the error one would expect when using kp alone. Could be wrong but this is not usual PID action.

10. *Brett* says:
    April 14, 2016 at 8:36 am

    @Terry G, "usual PID action" varies on this point. depending on the manufacturer, you can set a bias (or baseline) value from which the pid can operate, or the p-only controller can always work from 0. I decided to choose the first option, since I have encountered various processes where p-only is fine, but you want the output to be centered around 70% say. also, if you want this algorithm to operate from a baseline of 0, it's simply a matter of setting output=0 before putting the pid in automatic. everybody's happy!

## Leave a Reply

Name (required)

Mail (will not be published) (required)

Website

Submit Comment

- Search for: Search

- ## Links
  -
  -
  -

- ## This Site
  - About
  - Project Index

- ## Categories
  - PID (23)
    - Coding (11)
    - Front End (2)
    - Showcase (4)
  - Projects (40)
    - Craft (6)
    - Electronic (12)
    - Mechanical (26)
  - Uncategorized (5)

- ## Archives

- November 2012
- September 2012
- July 2012
- June 2012
- April 2012
- March 2012
- January 2012
- December 2011
- October 2011
- September 2011
- August 2011
- July 2011
- June 2011
- May 2011
- April 2011
- September 2010
- August 2010
- July 2010
- March 2010
- November 2009
- October 2009
- September 2009
- August 2009
- July 2009
- June 2009
- May 2009

---