The Application of PID Control in Student Projects

Dr. Alireza Kavianpour, DeVry University, Pomona

Dr. Alireza Kavianpour received his PH.D. Degree from University of Southern California (USC). He is currently Senior Professor at DeVry University, Pomona, CA. Dr. Kavianpour is the author and co-author of over forty technical papers all published in IEEE Journals or referred conferences. Before joining DeVry University he was a researcher at the University of California, Irvine and consultant at Qualcom Inc. His main interests are in the areas of embedded systems and computer architecture.

Miss Sogand Kavianpour

Miss Sogand Kavianpour received her B.S. degree in Electrical & Computer Engineering from the University of California, Irvine. She is currently working as a software developer.

Dr. Javad Shakib, DeVry University, Pomona

Student Projects and the Application of PID Control

This paper considers the different application of PID Controls. Specifically, we will take a look at how a group of students utilize this method in two different applications. Many of the Electrical Engineering students had a background in Mechatronics and Control Theory, which was beneficial in understanding the applications at hand. The first application included a robotic car. The students needed to direct the car to follow a predefined path. The second application included a robotic arm. The arm needed to reach a preset target within a specific time.

PID control is a feedback control in which it calculates an error as the difference between the desired value and a measured value. It then applies a correction based on proportional, integral, and derivative terms. The PID controller objective is to reduce the error by the adjusting a variable, such as the position of a robot arm or Robot car.

Some applications may require the use of only one variable: P, I, or D. This is achieved by setting the other parameters to zero. PID controller [6,7,8,9] can use two combinations of P, I, and D. It also can use the three terms to provide the appropriate system control. PI controllers are fairly common. A derivative action is sensitive to measurement noise, whereas the absence of an integral term may prevent the system from reaching its target value.

In this paper, the result of the PID controllers in the student projects such as robotic car and robotic arm, will be analyzed. The results are based on the application of proportional, integral, and derivative control. In addition, a MATLAB and C++ programs will be used to calculate PID values.

This paper suggests one possible method to implement the concepts that students have learned in the other courses, and use them in the real world applications. Robotic car and Robotic arm are two examples of the projects implemented in one of the courses in the **Electronic Engineering Technology (EET)** program called "Embedded Microprocessors System".

1- Introduction

There are a number of strategies for correcting output errors. In order to measure the error, you need to know both the current measurement and the correct measurement. The correct measurement is known as the set point. Obtaining the current measurement involves a sensor. Typically, a sensor's job is to receive an input. A motion sensor on an alarm system receives an input when the infrared beam is broken. This causes the controller to sound the alarm. Most systems correct errors using feedback: the output of the system is fed back into the sensor as an input. The sensor measures the current state of its own system. The current state is also known as the current position, current reading, and current output of the system. An example of this would be a thermostat attached to an engine. If the thermostat measures the temperature of the engine to be above the operating limit, it shuts down the engine[1].

Once the current state of the system is known, the controller calculates the difference between the current state and the target state. Based on this calculation, the controller must implement a correction strategy. The controller may have several strategies available to it. Below we will define several important terms and concepts.

Sensor

A sensor is a device that converts physical/chemical parameters to a measurable signal.

Actuators

An actuator is a component of machines that is responsible for moving or controlling a system.

Steady-state

Steady-state involves continually resetting the set point.

Open-loop control

Open-loop Control does not use feedback to determine if its output has achieved the desired goal every day. Open-loop System is easy to build but it is not reliable. It does not perform accurately unless calibrated and it

cannot be optimized. An example of an Open-loop System is a sprinkler system of a house programmed to turn on at given time.

Closed-loop Control

Closed-loop Control uses feedback to determine if its output has achieved the desired goal. Closed-loop System is difficult to build but is very reliable and performs accurately due to feedback. It is feasible to optimize. An example of a Closed-loop System is a sprinkler system that is programmed to turn on if the humidity of the soil is below a certain threshold.

Proportional Control

The correction is proportional to the amount of error. For example, an airplane is approaching an airport located at sea level. At different altitudes, the plane descends at various speeds.

Proportional Control Equations are as follows:

- Set Position – Current Position $= e(t)$
- Motor Speed $= K_p * e(t)$
- K_p= *proportional* gain and is the problem dependent.
- $e(t)$ = Error
- K_p determines how fast the object reaches the set point.

Derivative Control

Derivative control takes into account how fast the error is changing. The faster the error is increasing, the more correction that is applied. The slower the error is increasing; the less correction is applied. If a car stops suddenly in front of you on the freeway, you would apply a lot of brakes. If a car is stopping gradually, you would apply a little brake.

The Derivative Control Equations are as follows:

- Set Position – Current Position $= e(t)$
- $Derivative\ of\ e(t) = \frac{d}{dt} e(t)$

- $K_d * \frac{d}{dt} e(t) =$ Motor Speed

Integral Control

Consider an altimeter that takes a reading every *t* seconds. If the sum of all the errors adds up to zero, then integral control would not correct it. Now, let us look at a second situation where every time a reading is taken, the altitude is two feet. Then the integral control would integrate over the magnitude of the readings times the numbers of time it was read and apply a correction to two feet. Integral takes into account the time of the error before applying a correction.

The integral Control Equations are as follows:

- Set Position – Current Position $= e(t)$
- Integral of $e(t) = \int e(t)\,dt \bullet K_i * \int e(t)\,dt =$ Motor Speed
- K_i is integral gain.

The Integral can be calculated by approximation: divide the region under the curve into rectangles. In conclusion, the integral is the sum of the areas of these rectangles.

2- PID Control

PID control consists of:

- Proportional Control
- Integral Control
- Derivative Control

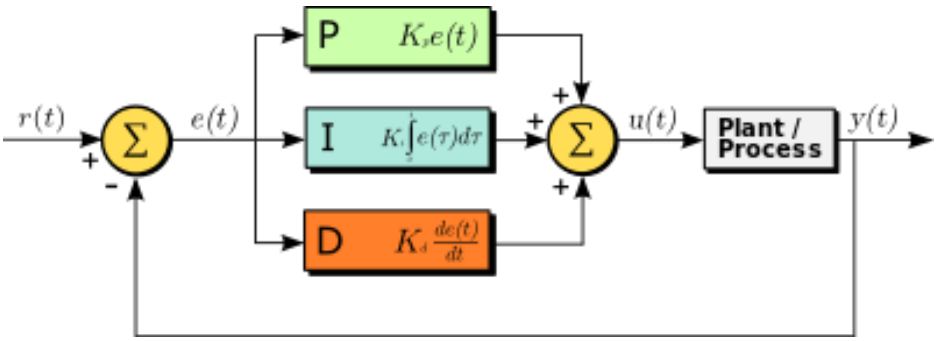

Figure 1: PID Control System The

PID Control Equations are as follows:

- $PID = K_p * e(t) + K_i * \int e(t)\,dt + K_d * \frac{d}{dt} e(t)$

Proportional control deals with present behavior.

Integral control deals with past behavior.

Derivative control deals with future behavior.

3- Student Projects

Robotic car and Robotic arm are two examples of the projects implemented in one of the **Electronic Engineering Technology (EET)** courses called "Embedded Microprocessors System". The objective in this course is to use embedded programming in the real life application such as the PID concepts that they already studied in the previous EET courses such as Control Theory and Mechatronics.

3.1 PID Control in a Robot Arm

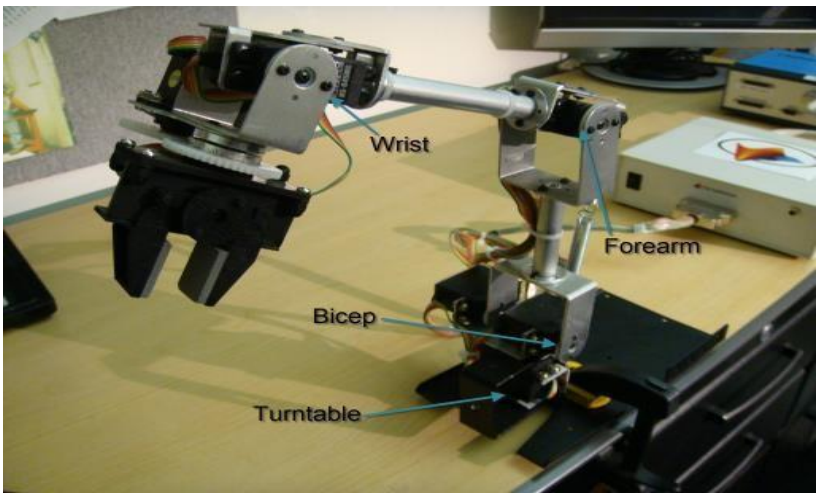

Figure 2: Robotic Arm

Consider a robotic arm (Figure 2) with four joints:

1: Turntable, 2: Bicep, 3: Forearm, 4: Wrist

Joints are moved by a DC/Servo motor. The Robotic Arm controller consists of four PIDs (one per joint). K_p, K_i, K_d for each PID loop is calculated separately. This process can be time-consuming. Alternatively, you can tune all four PID loops subject to system-level requirements. MATLAB has the tools to calculate PID values. The transfer function of a PID controller is found by taking the Laplace transform.

$$K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s}$$

We can also define a PID controller in MATLAB directly using the transfer function, for example:

```
Kp = 1;
Ki = 1;
Kd = 1;
C = Kp + Ki/s + Kd*s
C = (s^2 + s + 1)/s
```

Alternatively, we may use MATLAB's *pid* controller object to generate an equivalent continuous-time controller as follows:

```
C = Pid (5,1,2)
```

Kp = 5, Ki =1, Kd =2 C = Kp + Ki /s + Kd*s

3.2 PID control in a Robotic Car

In this project, students are required to design an autonomous robot car (Figure 3a) to navigate a track. The objective is to keep the car to follow the track (Fig 4) by PID control method. The processor board [2,3,4,10] for this project is the Axiom TWR-S12G128. It contains the MC9S12G128 Central Processor Unit (CPU). The board has many input/output ports, analog-to-digital (A/D) inputs, and pulse width modulation (PWM) outputs.

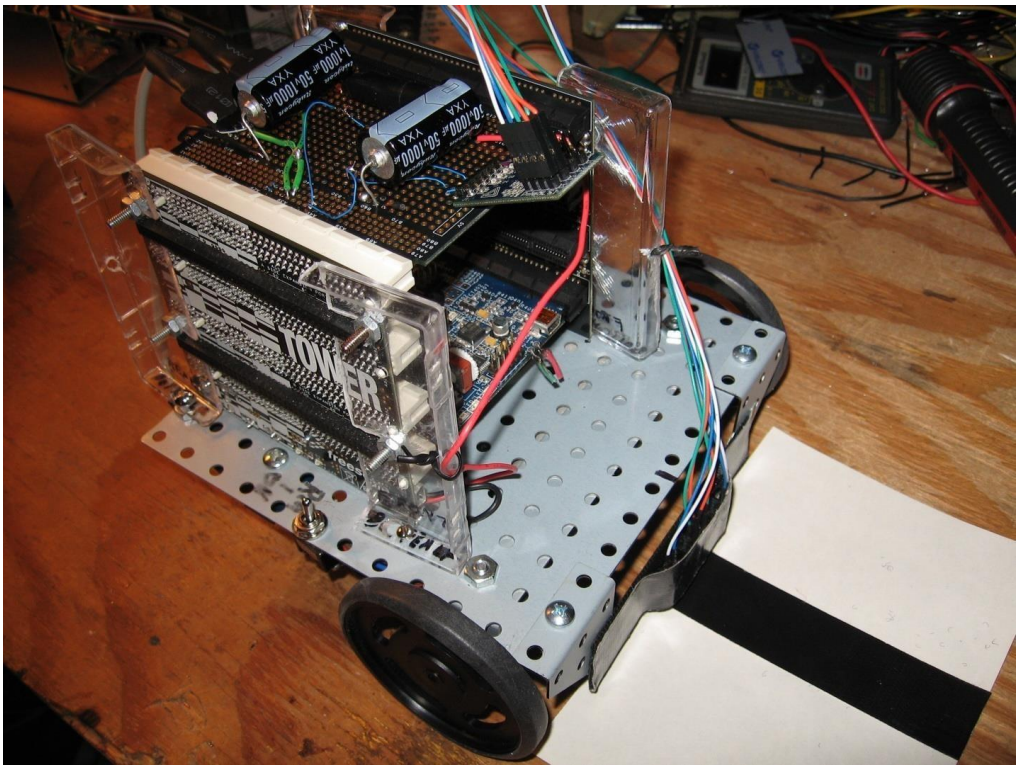

Figure 3a: Robotic Car

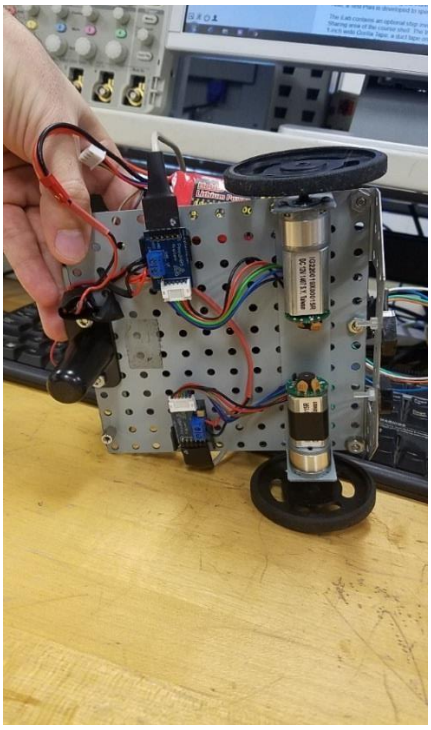

Figure 3b: Robotic Car

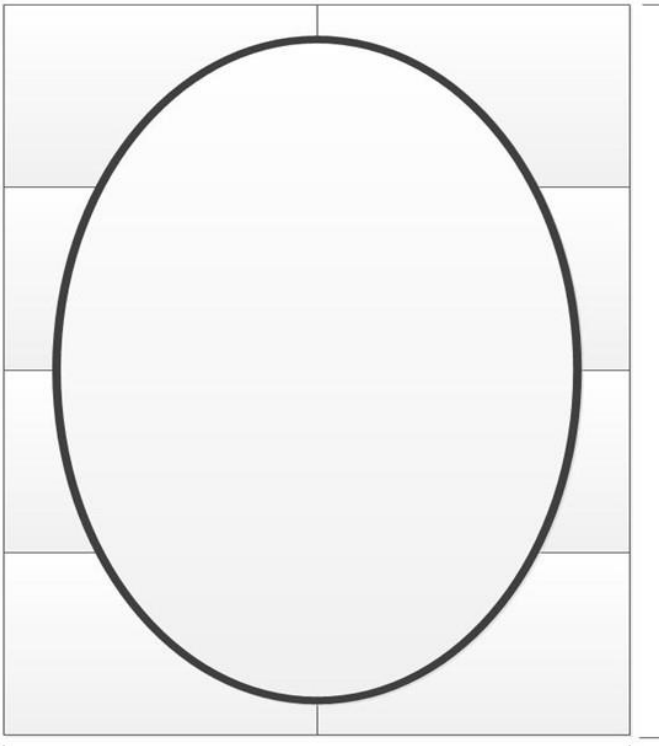

Figure 4: Car Track

Above we see the car track that the car must follow. This car also has two optical sensors. Figure 5 is an interface module for connecting optical sensors to one of the input port of the controller.

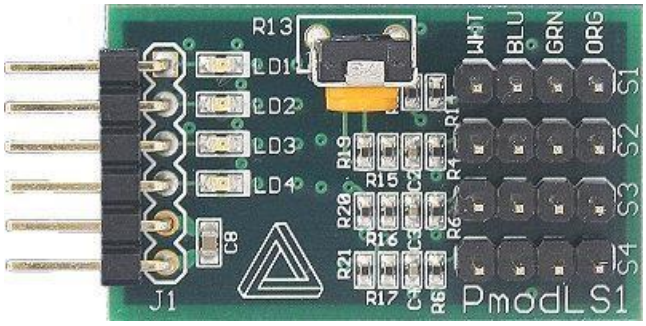

Figure 5: Sensor Driver

A robot car has two servo motors powered by an H-bridge. The H-brides receive input from the timer of the controller, as shown in Figure 6. The EN pins in the two H-Bridge are connected to two timers. The two timers control the speed of the motors. The program must stop the right wheel if the left sensor sees the white background. Similarly, the left wheel is stopped if the right sensor sees the white background. In either case, the wheel which is still turning moves the sensor back onto the black stripe. If both sensors light up, it means that the car has gone off the black stripe completely, and both motors stop.

Figure 6: Motor Driver

3.3 PID Program

The robot car is controlled by two different programs. One program using proportional control and the other program uses a combination of proportional and integral control. These two cases can be defined in MATLAB using the transfer function directly.

Case 1: Proportional control

```
Kp = 1;
Ki = 0;
Kd = 0;
C = Kp + Ki/s + Kd*s
C = (s^2 + s + 1)/s
```

Case 2: Proportional and integral control

```
Kp = 1;
Ki = 1;
Kd = 0;
C = Kp + Ki/s + Kd*s
C = (s^2 + s + 1)/s
```

In the following, the PID program written in C++ language will be discussed.

The microcontroller has eight timers. Each timer of the controller has two registers. They are called duty cycle register (PWMDTY) and period register, (PWMPER). These two registers control the speed of the motors. The function of PWME register is to enable the timer. The value of the registers for timer 0 and timer 1 considering the main clock of 24MHZ are as follows:

```
PWMPER0 = 200;
PWMDTY0 = 150;
PWME0 = 0x02;
PWMPER1 = 200;
PWMDTY1 = 150;
PWME1 = 0x02;
```

3.3.1 Proportional Control with C++ Program

This program reads output of the sensors and adjusts the speed of the motors accordingly. The program [5] is as follows:

```
sensor = PORTA & 0x03;
 switch(sensor)
{    case 0:        // Both sensors are
off.
   PWME = 0x03; // Turn both motors
on.     break;     case 1:       // Right sensor
is on.
   PWME = 0x01;  // Turn right motor on, left motor off.
   break;    case 2:       // Left
sensor is on.
   PWME = 0x02;   //Turn left motor on, right motor
off.     break;          case 3:       //Both sensors are on.
   PWME = 0x00; // Turn both motors off.
   break;
}
```

3.3.2 Proportional and Integral Control with C++ Program

This program reads the output of the sensors and adjusts the speed of motors. It continues to read the output of the sensors until errors are minimized. The program is as follows:

```
sensor = PORTA & 0x03;
LL: switch(sensor)
{    case 0:        // Both sensors are off.
  PWME = 0x03; // Turn both motors on.
  sensor = PORTA & 0x03;
   break;    case 1:       // Right sensor
is on.
   PWME = 0x01;  // Turn right motor on, left motor off.
   sensor = PORTA & 0x03;    break;
case 2:       // Left sensor is on.
   PWME = 0x02;   //Turn left motor on, right motor off.
   sensor = PORTA & 0x03;
   break;          case 3:       //Both
sensors are on.
   PWME = 0x00; // Turn both motors off.
sensor = PORTA & 0x03;
break; }
Go to LL;
```

By comparing the movement of the robot under these two programs, it is clear that the second program making the correction very fast and the car follows the track very smoothly. This program reads the output of the sensors until errors are minimized (proportional and integral control).

4- Conclusion

This paper suggests one possible method to implement the concepts that students have learned in the other courses, and use them in the real world applications. Robotic car and Robotic arm are two examples of the projects implemented in one of the courses in the **Electronic Engineering Technology (EET)** program called

"Embedded Microprocessors System". The objective in this course is to use embedded programming in the real world application such as the PID concepts that they already studied in the previous EET courses such as Mechatronics. These type of projects help students to understand the difficult concepts in their courses and relate theory to the real world applications.

5- References

1- Christopher T. Killian. (2006) Modern Control Technology, Third Edition, Thompson Delmar Learning.

2- Valvano, J. W. (2000). Embedded microcomputer systems: Real-time interfacing. Pacific Grove, CA: Brooks-Cole.

3- Mazidi, M. A., & Causey, D. (2009). HCS12 microcontroller and embedded systems using Assembly and C with CodeWarrior, Pearson/Prentice Hall.

4- Daniel J Pack and Steven F. Barrett. (2008). Microcontroller Theory and Applications: HC12 and S12, 2nd Edition, Pearson/Prentice Hall.

5- Bjarne Stroustrup (2009). Programming: Principles and Practice Using C++, Addison Wesley.

6- Jens Graf (2013) PID Control, Create Space Publisher.

7- Kok K. Tan. NEW Advances in PID Control (2011), Springer.

8- Michael Johnson and Mohammad Moradi. PID Control (2010), Springer.

9- Antonio Visioli (2006). NEW Practical PID Control, Springer.

10- www.digilentinc.com. Digilent Robotic Start Kit Reference Manual.