AC 2011-120: USING THE PROCESSING PROGRAMMING ENVIRONMENT IN ENGINEERING EDUCATION

Ryan J Meuth, University of Advancing Technology

I graduated from UMR with a B.S. of Computer Engineering in 2005, after which I stayed at UMR (Now Missouri University of Science and Technology) to pursue and complete a Master's and PhD in computer engineering. I worked for Dr. Donald C. Wunsch at the Applied Computational Intelligence Laboratory in the Department of Electrical and Computer Engineering. There I worked on the Learning Applied to Ground Robotics project, developing a ground vehicle that can not only navigate unknown terrain, but be able to learn from experience with the world. During the summers since 2006 I worked at the Boeing Phantom Works in Seattle, WA, developing algorithms for adaptive control of a swarm of flying robots under varying environmental conditions and failures. I completed my PhD in the fall of 2009, and I am now a Professor of Robotics and Embedded Systems at the University of Advancing Technology in Tempe, AZ. My research interests include robotics, embodied intelligence, and meta-learning algorithm development.

Using the Processing Programming Environment in Engineering Education

Abstract
Processing is an open-source, Java-based programming environment designed for artists and visual designers. In this paper we explore the use of the Processing as a tool for constructing interactive and demonstrative applications that enhance the engineering classroom experience. A brief overview of the Processing environment and its application in the classroom is presented. We identify two primary modes of application – digital demonstration and virtual laboratory, and we explore the use of programming in these contexts. Our paper introduces guidelines for development of media-rich learning tools, addressing issues such as interactivity, simulation accuracy, and aesthetics. Examples of classroom experiences using Processing applications are given, including fundamentals of electricity, electronics, and an introduction to controls. Student performance is measured through standardized assessments. Partial results are presented, suggesting increased student performance.

Introduction

The Processing[1] programming language and integrated development environment were initially developed as a tool for the electronic arts and visual design communities. As an open-source project, Processing has gained a great deal of momentum, changing from a tool specifically for artists, to a highly flexible media-rich development environment. In education, Processing has typically been used as a development environment for algorithmic art, and as a tool for introducing programming concepts. There is great potential for the use of Processing in the classroom, particularly engineering education, as a tool for demonstrating dynamic processes and concepts in a visual environment. The simplicity and power of the environment allow these demonstrations to be developed very rapidly, and a thriving community enables the open exchange of developed applications. Additionally, several Processing based spin-off projects, Wiring[2], Arduino[3], and Fritzing[4], bring a similar design interface to embedded programming and circuit design.

In-classroom demonstrations are often utilized to bridge the gap between analytical world of theory and the physical world of application, attempting to imbue the student with a visceral understanding of the interactions of forces and variables in a system. The effect of these demonstrations are clearly beneficial, as they serve to both break the monotony of an endless stream of theory and equations, and to tie symbols to real-world phenomena, solidifying their meaning in the minds of pupils.[5]

Hands-on laboratory work is highly regarded as a method for reinforcing learning by exposing students to real-world applications and interactions. This is particularly important in the engineering disciplines, as there is a great deal of engineering culture that surrounds theoretical phenomena under study (e.g. resistor color codes, the use of compilers and tools, etc.). These initially mystifying and often confusing conventions must first be overcome before the nascent student can begin to explore the true object of their laboratory excursion. While this cultural knowledge is no doubt important, it can impede the student's progress towards theoretical understanding and flaws in application understanding (or even the laboratory equipment itself) can lead to major misunderstandings[6].

Many physical phenomena cannot be easily demonstrated, or the theory is lengthy and unwieldy. Thus, many educators most often make use of slide presentations and passive media in order to communicate ideas. This, in itself, is not a bad thing, as the added media of clear pictures, graphs, and schematics can be a great addition to the classroom experience. However, educators can overly rely on the slides, loading them with text and using the slides as their lecture notes, weakening the verbal and spatial reasoning of students[7,8].

The above teaching methods all point to an ever-increasing trend towards catering to visual-spatial learners in the classroom. Many students (particularly those attracted to the engineering disciplines) are not primarily reading-based learners, and though this skill is critical, the learning process can be accelerated greatly by the incorporation of multi-media, including demonstration, visualization, and simulation[8,9].

While MatLab, Simulink, and LabView provide similar simulation and experimentation capabilities, these environments are often extremely expensive, with steep learning curves and few portable skills. Processing provides a unique combination of cost (free), broad community support, extensibility, and as a Java based environment, skills learned in processing can be easily ported to other environments.

In the following sections we will summarize the capabilities of Processing, provide some guidelines for experience design, and explore two case studies on the use of Processing in the classroom, as a method of demonstration in one case, and as a virtual laboratory in a second case.

Processing Capabilities

Processing is a Java-based programming language and development environment targeted primarily to electronic artists and visual designers, but is becoming popular with hobbyists and educators, primarily for teaching introductory computer science. The programming environment was designed to make the programming experience as simple as possible, and to encourage exploration and understanding through immediate visual feedback. The Processing environment abstracts many of the steps necessary to produce and animate on-screen graphics, and much of the development environment itself, such that writing and compiling a program has very few steps and can be completed in a very short time.

This simplification comes with a cost – there are no debugging tools available in the default environment (making the development of large programs difficult) and as an open-source project there is no resource for rapid, personalized technical support that might be expected from a commercial software development environment.

Even with these limitations in mind, the Processing development environment is extremely capable, and particularly well suited to visualization. Both 2D and 3D primitives are available, making it a simple task to develop visualizations that would be involved and difficult on many other platforms. For example, a Processing code sample is given below that creates a drawing surface (contained in the setup function), and draws a square to the screen.

```
void setup()
{

// Size should be the first statement
  size(200, 200);  // Defines the size of the drawing surface, initializing
                   // height and width environment variables.

  stroke(255);     // Set stroke color to white
}

float y = 100; // Global Variable



// The statements in draw() are run until the
// program is stopped. Each statement is run in
// sequence and after the last line is read, the first
// line is run again.

void draw()
{
  background(0);   // Set the background to black
  rect(75, y, 50, 50);
  y = y - 1;
  if (y < 0) {
    y = height;
  }
}
```

Figure 1 - A simple Processing program.

By default, the Processing environment calls any defined `setup()` function first, and only once when the program starts. The `draw()` function is called repeatedly until the program stops, acting as built-in looping structure. The Processing environment is stateful, tracking the height and width of the drawing surface, the current stroke (border) and fill colors for drawing objects, and a stack of matrix transformations for translating and rotating drawn objects. In the example program, the `size(…)` function sets the size of the drawing surface, initializing the `height` and `width` environment variables.

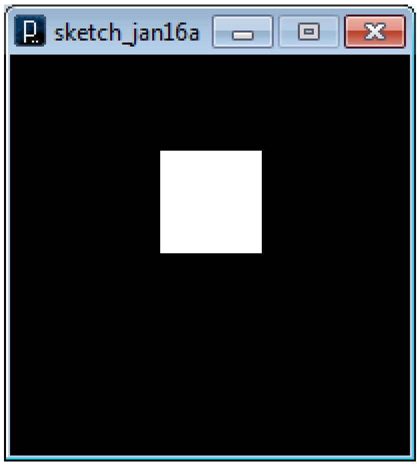


Figure 2 - Simple Processing program during execution

As can be seen in the code example above, the syntax is that of Java, with simplified access to drawing methods. In particular, the visual primitive function `rect(…)` draws a square to the screen at the given location. The output of the program is shown in Figure 2. Other visual primitives include ellipses, lines, points, vector shapes, and sprites, as well as individual pixel access. Interface primitives include keyboard and mouse input.

The entire Java environment and standard libraries are accessible from Processing, and advanced users can even use the Eclipse IDE[10] for more complex development projects. In addition to the standard libraries and visual primitives, access to audio and video capture and output is provided through the Minim and “video” interfaces, respectively. Networking and communication is supported through “net” and “serial” libraries, with simple, reliable interfaces. Many code examples can be found in the “Libraries” section of the Processing website.

Apart from extensive reference documentation and code examples, a thriving community shares programs (known as “sketches”) through the OpenProcessing.org website[11]. In this community, groups and individuals share and showcase their projects, and as Processing is Java-based, their projects execute directly in-browser. OpenProcessing.org is also a centralized location for hosting classrooms that use the Processing environment to teach programming, interaction, algorithmic art and many other topics. Campuses that have shared their online classrooms include UCLA, Harvard, and Yale in addition to many others.

As a free, open source project with ever-growing capabilities, Processing has a very low cost of entry. A thriving support and sharing community contribute helpfully to the environments already low learning curve. These are extremely desirable features for busy educators and students.

Experience Design
In engineering there are many complex phenomena that are difficult, if not impossible to demonstrate directly and visually in the classroom. An example of one of these "invisible" phenomena would be the changing electrical currents in a resistor-capacitor system driven by an oscillating voltage source. In this system, the currents and voltages can be measured and visualized indirectly using oscilloscopes and other tools, but these secondary measurements mask the actual motion of electrons. An intuitive knowledge about how charges behave in electrical systems may provide insight and understanding to students struggling with the abstract qualities of theory based on secondary measurements. In this case a dynamic visual simulation of the behavior of electrons in simple circuits may be extremely useful in the classroom. Such a simulation is possible, and indeed easy to create using Processing, as we will see in a following case study.

In addition to being able to simulate and visualize otherwise invisible phenomena, Processing enables the augmentation and instrumentation of real visible processes through the video and audio capture interfaces. Imagine a physics demonstration studying ballistic motion. The instructor executes a Processing sketch that takes input from a web-cam, tracking and recording the motion of a brightly colored ball, such as a tennis ball. The instructor then gently tosses the tennis ball across the camera's field of view. The Processing sketch marks the trajectory of the ball in every frame, automatically providing position, velocity, and acceleration measurements in a virtual coordinate frame. The instructor can then use these measurements to confirm pre-demonstration analysis, or post-demonstration verify acceleration due to gravity reinforcing theoretical knowledge with a media-rich application demonstration.

In the same way, Processing can function as a digital laboratory, providing inexpensive instrumentation for real processes, and simulation of processes that would be difficult to study in the budget-conscious engineering program. For example, a laboratory on controls typically requires sophisticated and expensive machinery, precise sensing equipment, and computing resources. Simulating these mechanisms in Processing vastly decreases the cost of equipment (students can perform the laboratory assignment on their home computers) and increases the possibilities for exploration, as the system under control can be perturbed by forces of arbitrary complexity, extensive and sophisticated instrumentation is possible, and there exists no machinery to maintain, wear out or break, removing many stumbling blocks from the laboratory experience.

As the educational field should adopt the "First, do no harm" principle from medical ethics, any demonstration or simulation should not mislead the student through over-simplification, erroneous representation, etc. An incorrect understanding is worse than lack of understanding, as the erroneous knowledge must be first exposed and overcome before the correct understanding can be communicated effectively. In computer simulation it is extremely easy to over-simplify phenomena, and thus impart partial understanding where the student believes they are receiving complete knowledge. It is vitally important that simulation reflects reality as accurately as possible – this is the burden of the experience designer. If available, a domain expert should be consulted. In addition to correct theory, the task of translating a complex, real-world process to computer simulation is often non-trivial, particularly for complex systems. An excellent overview of simulation problems and solutions is provided in[12].

Beside the above points, we suggest the following features for instruction designers:

1. Keep the visual style simple. Using shape primitives saves both development time and allows students to focus on the interaction and principles that are attempting to be communicated.
2. Use color to distinguish similar objects with different properties. For example, represent particles with opposite charge as circles with contrasting colors. See[13] for a tool for selecting attractive color schemes.
3. For dynamic processes, emphasize motion and environmental motion cues.
4. Instrumentation and representing data is critical. Provide continuously updating graphs and charts to tie visual representation to measured quantities.
5. Leverage time distortion. Processes do not need to occur at the same rate in simulation as they would in reality. For rapidly evolving dynamic systems, allow the simulation to progress at a slower rate to capture the nuances of interaction, and allow instrumentation to accurately measure the system for later analysis.
6. Use Processing's File I/O capabilities to automatically save captured data. The saved data can then be processed using more sophisticated tools like a spreadsheet program or statistical analysis programs like SAS/STAT[14] or R[15].
7. For digital laboratories, provide extensive user documentation on the use, structure, and modification of the program. Be very clear about what parts of the laboratory students are allowed to modify, and what parts are restricted.

Case Study – RC Circuit Simulation

To demonstrate the advantages of using Processing for in-classroom demonstration, a first case study is presented on the topic of Resistor-Capacitor circuits driven by AC signals. In this experiment, two sections of an introductory electronics course were presented with the same content in two different forms. One section utilized a static multimedia presentation in the form of presentation slides, acting as the control. The second section utilized a Processing-based electron-level circuit simulation. In both cases the function of the resistor and capacitor were explained, as well as frequency-filtering behavior, and the cutoff frequency was derived. Each course section consisted of 9 first-year undergraduate students, who were presented with material by the same instructor. The students were evaluated using identical homework and quizzes (see

Appendix A) including a verbal explanation of filtering behavior, and analytical problems designing high and low-pass filters for specific frequencies.

Experimental data for this case study is shown in Table 1. In both verbal and analytical evaluation, students presented with a Processing-based dynamic demonstration performed slightly higher than the control group. Further analysis is performed in Tables 2 and 3, showing analysis of variance for the experiment, and both metrics, with $\alpha < 0.05$. In each type of evaluation, both verbal and analytical, the difference between groups and sample size was not large enough to reject the null hypothesis; we cannot draw sound statistical conclusions from this data. However, this case suggests the need for additional experimentation.

	Verbal		**Analytical**	
	Control	Experiment	Control	Experiment
	0.563	1.000	1.000	0.500
	0.813	0.933	1.000	1.000
	0.250	1.000	1.000	0.200
	0.875	0.533	0.267	1.000
	0.938	1.000	1.000	0.600
	0.875	1.000	1.000	1.000
	0.813	0.667	0.800	0.700
	0.625	0.467	0.267	0.900
	1.000	0.533	0.133	0.900
Mean	0.750	0.793	0.719	0.756
Std. Dev.	0.234	0.237	0.380	0.279

Table 1 – Grade data for RC circuit case study, showing slightly increased performance for experimental group, in both verbal and analytical evaluation.

Anova: Single Factor - Verbal Responses						
Groups	*Count*	*Sum*	*Average*	*Variance*		
Control	9	6.75	0.75	0.054688		
Experiment	9	7.133333333	0.7925926	0.056049		
ANOVA						
Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.00816358	1	0.0081636	0.147441	0.706048	4.493998
Within Groups	0.88589506	16	0.0553684			
Total	0.89405864	17				

Table 2 – Analysis of variance for performance of control vs. experimental groups on verbal evaluation. Though increased performance is suggested, the sample size is not large enough for statistical significance (highlighted in gray).

Anova: Single Factor - Analytical Responses						
Groups	*Count*	*Sum*	*Average*	*Variance*		
Control	9	6.466666667	0.7185185	0.144198		
Experiment	9	6.8	0.7555556	0.077778		
ANOVA						
Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.00617284	1	0.0061728	0.055617	0.816554	4.493998
Within Groups	1.77580247	16	0.1109877			
Total	1.78197531	17				

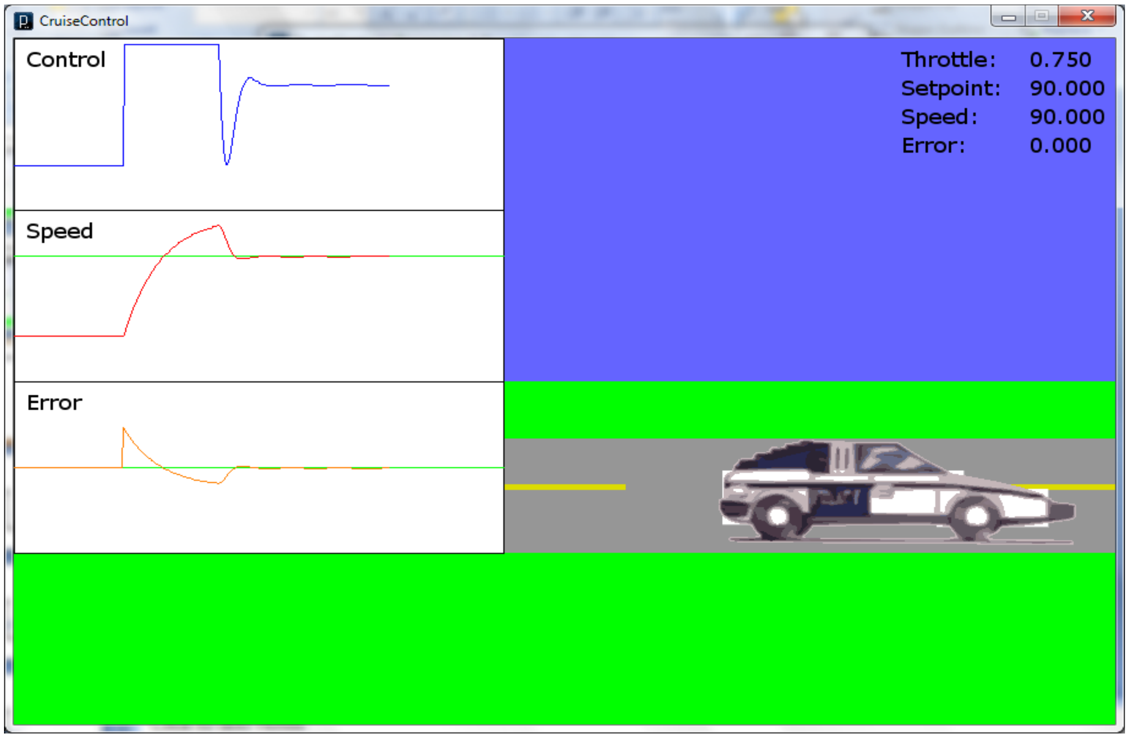
Table 3 – Analysis of variance for performance of control vs. experimental groups on analytical evaluation. Though increased performance is suggested, the sample size is not large enough for statistical significance (highlighted in gray).

Case Study - PID Control Digital Laboratory

In a second case study, a Processing-based digital laboratory is evaluated on a single class of students. The laboratory consists of a vehicle simulator, with environmental settings of level, down-hill and up-hill terrain, as well as 3 different set-points. The vehicle is instrumented with speed sensing, throttle position sensing, and error sensing, shown in Figure 2. Captured data can be exported to a spreadsheet compatible file format. The goal of the laboratory is to characterize the system, implement proportional, proportional-integral, and proportional-integral-derivative controllers for the simulated vehicle, tuning and evaluating the controllers at each step.

A class of 8 students was evaluated before and after the laboratory assignment with variants of an 18 question test consisting of 10 multiple choice, 5 analytical problems, and 3 short-answer questions, totaling 50 points. A partial sample of this evaluation battery is included in Appendix B.

The results in table 4 suggest that the laboratory produces a notable difference, though the experimental design does not compare a Processing-based digital laboratory to a traditional physical library. Further analysis of these results (Table 6) shows that this increase in performance is not statistically significant for an $\alpha < 0.05$ for the experiment size, again suggesting that further, more complicated experimentation is necessary.



Throttle: 0.750
Setpoint: 90.000
Speed: 90.000
Error: 0.000

Figure 3 – Screenshot of PID control digital laboratory software.

	Control	**Experiment**
	0.966667	0.98
	0.75	0.8
	0.666667	0.8
	0.866667	0.94
	0.85	1
	0.916667	1
	0.95	0.94
	0.833333	1
Mean:	**0.85**	**0.9325**
Std. Dev:	**0.101575**	**0.085482**

Table 4 – Control digital laboratory experiment results, showed marked improvement after interacting with the laboratory.

Anova: Single Factor - Controls Digital Laboratory

Groups	*Count*	*Sum*	*Average*	*Variance*
Control	8	6.8	0.85	0.010317
Experiment	8	7.46	0.9325	0.007307

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.027225	1	0.027225	3.089431	0.100636	4.60011
Within Groups	0.123372	14	0.008812			
Total	0.150597	15				

Table 5 – Analysis of variance for controls digital laboratory experiment. The experimental difference and sample size is not great enough to draw significant statistical conclusions, for an α < 0.05 (highlighted in gray).

Conclusion

We have presented an overview of the Processing programming environment, including a simple code example, references to libraries and related projects, and a review of Processing's capabilities. Additionally, we have explored a few design concepts, outlined implementation considerations, and suggested guidelines for experience design using Processing in the classroom. We have identified two primary modalities for applying Processing in engineering education – demonstration and digital laboratory, and explored some early and incomplete experimental results that suggest, but do not confirm, that Processing-based tools can be utilized effectively in the classroom. Future research will continue to explore these questions, but it is clear that Processing is a simple, inexpensive environment that has the potential to enrich the learning environment.

References

1. **Fry, Ben and Reas, Casey.** Processing.org. *Processing.org.* [Online] [Cited: January 17, 2011.] http://www.processing.org.

2. **Barragan, H.** *wiring.org.* [Online] [Cited: January 17, 2011.] http://www.wiring.org.co.

3. **Banzi, M.** *Arduino - Home Page.* [Online] [Cited: January 17, 2011.] http://arduino.cc/en/.

4. **Wettach, R.** *Fritzing.* [Online] [Cited: January 17, 2011.] http://fritzing.org.

5. *First Principles of Instruction.* **Merrill, M.D.** 3, s.l. : Educational Technology Research and Development, 2002, Vol. 50. 43-60.

6. *A wakeup call to Science Faculty.* **Alberts, B.** 5, s.l. : Cell, 2005, Vol. 123. 739-741.

7. **Tufte, Edward R.** *The cognitive style of PowerPoint.* Chesire, Conn. : Graphics Press, 2003.

8. *Using Learning Style Instruments to Enhanse Student Learning.* **Hawk, Thomas F. and Shah, Amit J.** s.l. : Descision Sciences Journal of Innovative Education, 2007.

9. *Attempted Validations of the Scores of the VARK: Learning Styles Inventory with Multitrait-Multimethod Confirmatory Factor Analysis Models.* **Leite, Walter L., Svinicki, Marilla and Shi, Yuying.** s.l. : SAGE Publications, 2009.

10. **Foundation, Eclipse.** [Online] [Cited: January 17, 2011.] http://www.eclipse.org.

11. **Processing, Open.** [Online] [Cited: January 17, 2011.] http://openprocessing.org.

12. **Perros, Harry.** Computer Simulation Techniques: The Definitive Introduction. [Online] [Cited: January 17, 2011.] http://www4.ncsu.edu/~hp/simulation.pdf.

13. **Stanicek, Petr.** *Color Scheme Designer.* [Online] [Cited: January 17, 2011.] http://colorschemedesigner.com.

14. *SAS - Statistical Analysis Software.* [Online] [Cited: January 17, 2011.] http://www.sas.com/technologies/analytics/statistics/stat/index.html.

15. *R-Project.* [Online] [Cited: January 17, 2011.] http://www.r-project.org.

Appendix A – RC Circuit Evaluation Quiz

1. Three capacitors are connected in series between nodes A and B. C1 is 0.1uF, C2 is 2uF, and C3 is 100uF. What is the capacitance measured between nodes A and B?
2. Three capacitors are connected in parallel between nodes A and B. C1 is 0.1uF, C2 is 2uF, and C3 is 100uF. What is the capacitance measured between nodes A and B?
3. A long-run serial communications line is experiencing problems with transmission errors. Investigation shows high-frequency noise on the line.

   Noise Frequency: 60MHz
   Serial Communication Frequency: 10KHz

   Design an RC circuit that removes the noise from the line without harming serial communication.

   In your solution, note what configuration of RC circuit is used (where is Vout measured?) and choose values of R and C.

   Show your work for full credit.

Appendix B – PID Controls Evaluation

1. Short Essay: Describe the limitations of proportional-only control.
2. Short Essay: Describe the tuning of a proportional-only controller, in the context of a critically-damped system.

3. A thermostat is an example of what kind of controller?

    a. Open Loop

    b. Feed Forward

    c. Proportional

    d. Bang-Bang

4. To what kinds of systems can classical controls be applied?

    a. Non-Linear

    b. Linear

    c. Discrete

    d. Transcendental

5. True / **False**: A measured quantity is always exactly the same as a real physical phenomena. For example, the measured speed of a vehicle is the same as the real speed of the same vehicle.

6. True / **False**: Open-Loop control systems utilize feed-back mechanisms

7. Which of the following can NOT be a characteristic of a linear system?

    a. The output of the system is strictly proportional to input.

    b. **Derivatives of the system result in no reduction in complexity.**

    c. For any set of real inputs and outputs of the system, there exists a constant that maps all inputs to all outputs.

    d. Derivatives of the system result in a reduction of complexity.

8. Essay: Describe each of the 3 components of a PID controller, and how they contribute to minimizing the output error of a system. Describe the limitations of a full PID controller.