Proceedings of the 5th International Conference of Control, Dynamic Systems, and Robotics (CDSR'18)
Niagara Falls, Canada – June 7 – 9, 2018
Paper No. 124
DOI: 10.11159/cdsr18.124

A Do-and-See Approach for Learning Mechatronics Concepts

Anoush Sepehri, Alireza A Asadi, Gustavo K. Costa, Nariman Sepehri
Department of Mechanical Engineering, the University of Manitoba
Winnipeg, MB, Canada

Abstract - This paper describes a low-cost approach for teaching mechatronic concepts to undergraduate engineering students. The laboratory unit includes a wide range of mechatronics related subjects, including computer interfacing, design, prototyping, controller implementation, mechanism synthesis and analysis and, hardware-in-the-loop simulations. The students are given a shoebox filled with all needed materials with the goal of completing several projects with clear objectives. The students can plan and accomplish the goals in many ways, and will gain greater technical skills sooner and become technically more mature. For the experiments described in this paper, prototyping and control of a 5-link parallel robot to follows a planar trajectory is described as a project.

Keywords: Mechatronics, Training Unit, Arduino, Hardware-In-The-Loop, Controller Design.

1. Introduction

The science of mechatronics is built upon the creation of optimal designs for electronic-controlled mechanical systems [1,2]. These systems embody four major fields of knowledge: Electrical Engineering, Mechanical Engineering, Computer Science and Information Systems. Typically, a mechatronic system consists of sensors to provide real-time information for controller that is to be designed to produce output signals to actuators. The controller also sends out information to graphical displays such as LCDs LEDs, or printers. Fig. 1 shows typical signal flow for mechatronic systems.

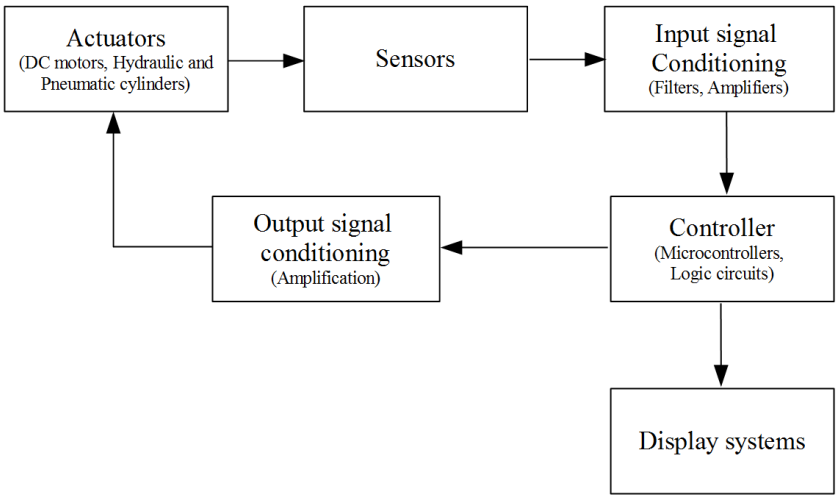

Fig. 1: Typical signal flow in a mechatronic system (adapted from reference [1]).

The traditional way of teaching mechatronics cannot adequately serve the need for practical knowledge in this area, which heavily depends on a cross-disciplinary set of projects. It is therefore very essential that students be extensively involved with hands on experiments to develop a better understanding of the issues concerning this area. Furthermore, it is also well known that increased exposure to real-world problems requires shifting the students' learning experience from a dominant lecture mode to an active learning approach. Project-based learning approach allows traditional lectures to be partially replaced with group learning via working on practical applications. In this paper, we present a laboratory-based environment equipped with non-traditional modular hardware devices and extensive use of computer simulations and

interfacing, allowing the students to learn as they need and independently design, build and evaluate typical mechatronics devices. Many aspects and issues related to mechatronics will be explored in this unique team- and self-learning environment, which is believed to be more effective than the pure traditional lectures and conventional laboratory sessions.

2. Project Activities

The components of this lab include an Arduino UNO (a microcontroller that can be programmed to accomplish various tasks), a motor shield, which allows the Arduino to supply adequate voltage signals, DC motors, basic electrical components, potentiometers and encoders. The lab also includes several acrylic sheets for prototyping parts. For simulation and prototype creation, students use third-party programs, Matlab and Solidworks. An Arduino IDE (which is an open source software) is also needed as a platform where students can create their programs. Having described the basic components, we now proceed to explore six tasks for students to undertake.

2.1. Task 1

The first task is to build an open-loop speed-controlled DC motor. In order to accomplish this, students will have to write a C script that will allow Arduino to communicate with the motor. Here, students will learn as they code. Within the Arduino IDE, there are many built-in examples describing what each part of the code does. This makes it easier for students to grasp the basics of Arduino C-programming. As a result, understanding the built-in scripts is a parallel task to be performed by the students.

In this task, a knob potentiometer is used to produce an input signal to the Arduino. The signal is then read through the program created by the students, and an output signal is sent to the motor, which controls the speed and direction of rotation. Fig. 2 shows a typical screenshot containing an example of a C program, which was uploaded to Arduino to accomplish this task. The program reads a signal from an actual knob potentiometer and sends a voltage output to the DC motor. The variables declared (potpin, brake and direction) are all used according to the hardcoded pins on the motor shield. The direction and brake are declared as output signals, which are sent to the motor. The knob potentiometer sends the analog signal (0-1023 bytes) to Arduino, in which a Boolean expression is used that reads it and sends out a pulse width modulated signal to the motor to control speed and direction (clockwise or counter clockwise).

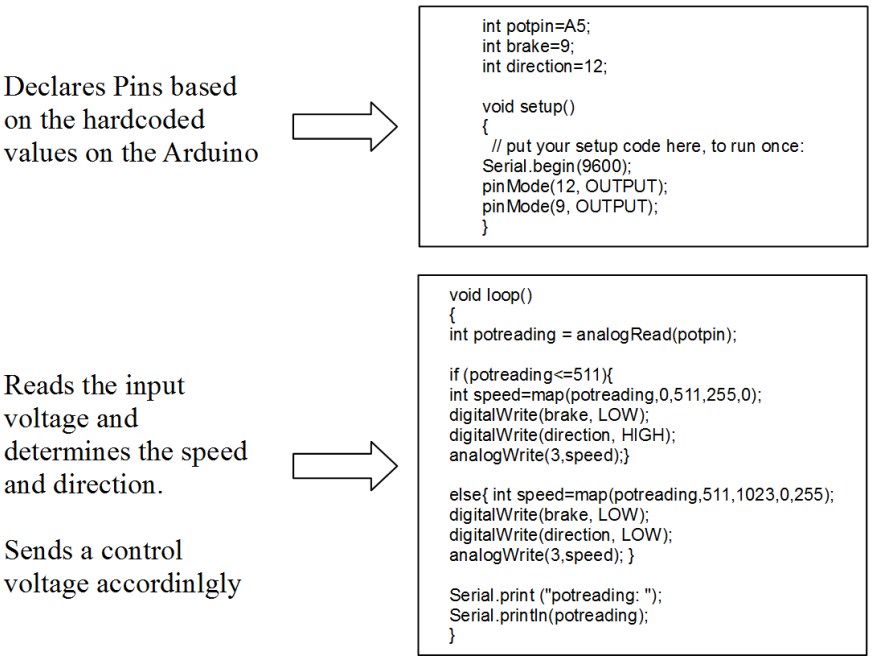

Fig. 2: Typical C script for open-loop speed and direction control of DC motor.

Another aspect of this exercise is that students must design a single link to be connected to the motor shaft. The design is done using Solidworks. Once the design is finished, the parts are to be built. In this exercise, the parts are all produced with only a laser cutter. Figs. 3 and 4 show the design and the prototyped assembly. As seen in Fig. 4, a set of acrylic disks are placed in between the rotating link and the motor shaft. An outside capsule has also been added to make the whole set more visually appealing. Acrylic was the material used because it is relatively cheap, easy to prototype and is lightweight. With the completion of Task 1, students will have acquired a basic understanding of how to use the Arduino IDE and Solidworks program. This task forms the foundation for future tasks the students will conduct.

Fig. 3: Solidworks design of single link.

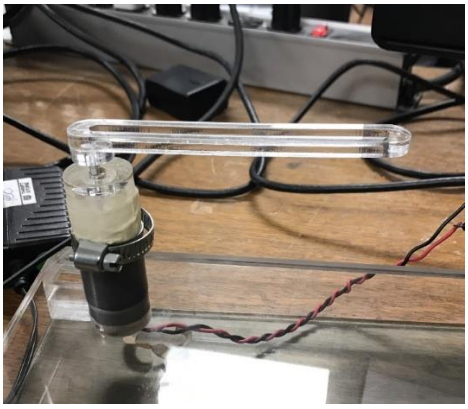

Fig. 4: Prototyped unit of single link.

2.2. Task 2

This task is divided into four steps (activities). In the first activity, students are required to use the Matlab-Simulink tool to simulate the velocity and position control of the same DC motor they operated in Task 1. The first simulation is performed assuming an ideal DC motor. Using the dynamic equations, derived by students, block diagrams are created, as seen in Fig. 5 - the input voltage goes through a series of gain ports, representing resistance, motor torque constant and inertia. This determines angular acceleration, which can sequentially be integrated to obtain the angular velocity and position. The next step is to make the simulation more realistic by adding friction as seen in Fig. 6. Students must learn about different types of friction and the mathematical models representing them. The dynamic equations for the block diagrams in both Figs. 5 and 6 are based on Eqs. (1) to (3) where T is the motor torque, V is the applied motor voltage, K_e is the constant back emf, w is the angular velocity, K_t is the constant motor torque, R is the electrical motor resistance, F_f is the friction force, F_c is the Coulomb friction, c_w is a Stribeck friction related coefficient [5], f is the viscous friction coefficient, F_{brk} is the breakaway friction, I is the moment of inertia, and a is the acceleration.

$$T = (V - K_e w)\frac{K_t}{R} \tag{1}$$

$$F_f = [F_c + (F_{brk} - F_c) \cdot \exp(-c_w|w|)]\text{sign}(w) + fw \tag{2}$$

$$a = \frac{T - F_f}{I} \tag{3}$$

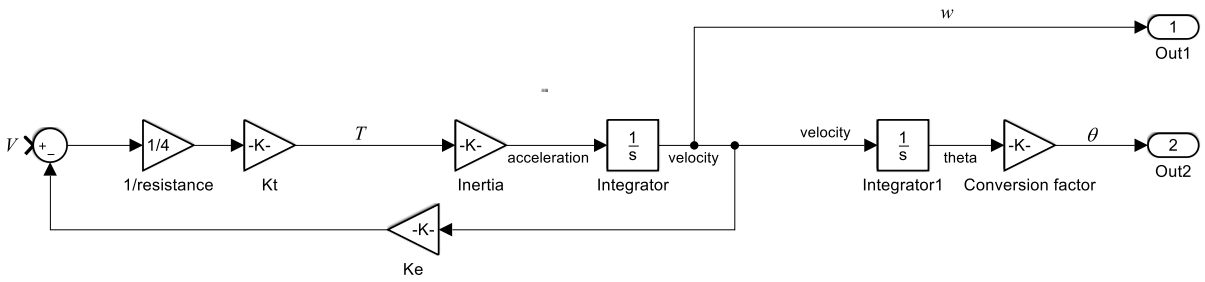

Fig. 5: Simulink block diagram of an ideal DC motor.

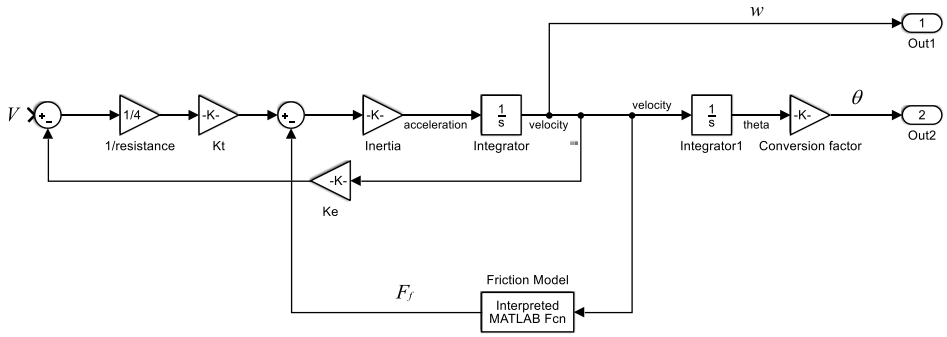

Fig. 6: Simulink block diagram of DC motor experiencing viscous and coulomb friction.

The third activity is to incorporate a controller in the simulation model. Students begin by building a proportional-derivative-integral (PID) control scheme. As shown in Fig. 7, the angular position of the motor shaft is read and compared to an input representing the desired motor shaft angular velocity. The resulting error is an input to the PID controller that sends an adjusted signal into the motor system. This output signal also goes through a saturation block of $\pm$9V, which represents the maximum voltage the DC motor can handle.

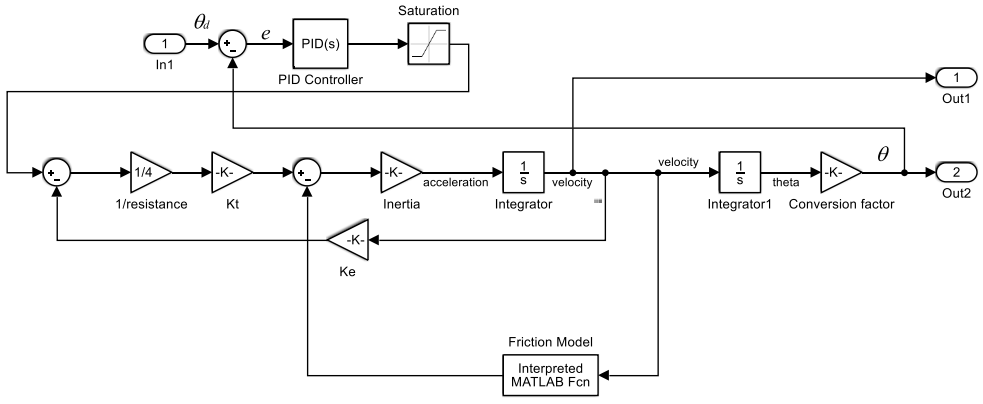

Fig. 7: Simulink block diagram for angular position control of DC motor using PID controller; *e* is the error between the desired and the actual angle.

The PID controller shown in Fig. 7 consists of three components [3]. The proportional part calculates the output based on the difference between the desired and the actual position of the motor. The integral part calculates a signal to remove the steady-state error between the desired and the actual positions due to external factors such as friction and gravity. The derivative part calculates its output by the rate of change of the motor position [4] and can affect the transient response. The students will study the influence each component of the controller on the overall response. Typical results are shown in Figs. 8 to 10.

$$u(t) = \overbrace{K_p e(t)}^{\text{Proportional}} + \overbrace{K_i \int_0^t e(\tau)d\tau}^{\text{Integral}} + \overbrace{K_d \frac{d}{dt} e(t)}^{\text{Derivative}} \tag{4}$$

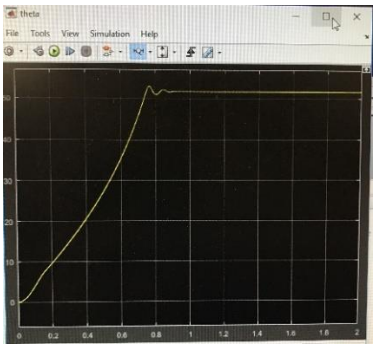

Fig. 8. Simulation response: P control.

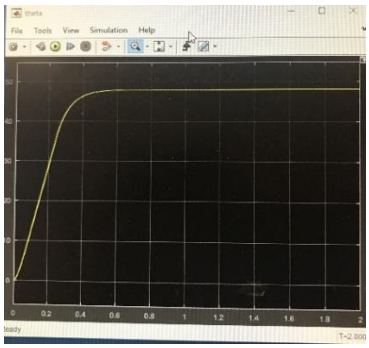

Fig. 9. Simulation response: PD control.

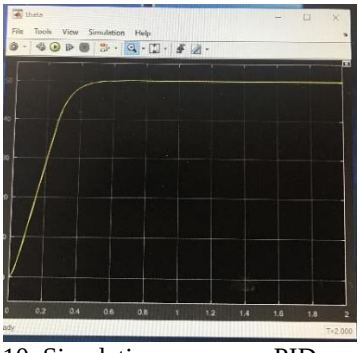

Fig. 10. Simulation response: PID control.

The last activity involves integrating the Arduino with the simulation model of DC motor holding a link to form a hardware-in-the-loop (HIL) simulation platform. In doing so, students must interface the Arduino running the PID controller with the Simulink simulation of DC motor and see if the controller running on the Arduino is suitable before it is applied to the actual physical system. With reference to Fig 11, the circled Serial-Receive and Serial-Send blocksare for sending and receiving data from the Arduino that implements the PID controller. The blocks in between the Serial Receive and Send are to simulate the DC motor. The Simulink software requires single typedata, while the Arduino requires int8 type data. Thus, data type conversions and saturation blocks have to be used to change the data into a workable form for both the computer and the Arduino.

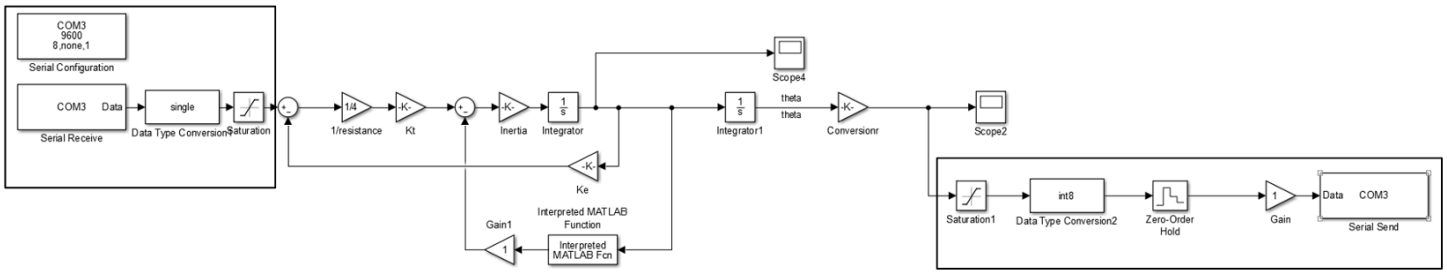

Fig. 11: Hardware-in-the-loop interfacing of Arduino of PC through serial communication. The box on the left is to receive data from the Arduino and the box on the right is to send data to the Arduino.

Once the hardware-in-the-loop simulation is developed, the same tests as in full simulation shown in Figs. 8 to 10 are conducted to ensure that a PID controller on the Arduino is a suitable application for a physical DC motor. Figs. 12 to 14 show typical results. The differences between these results and the ones shown in in Figs. 8 to 10 are largely due to the resolution and accuracy of the Arduino board. The rate at which data was sent through the USB was significantly lower when compared to the computer simulation. Here the students can investigate all these factors in the results.

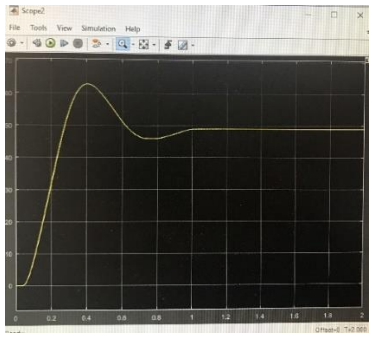

Fig. 12: HIL simulation response: P control.

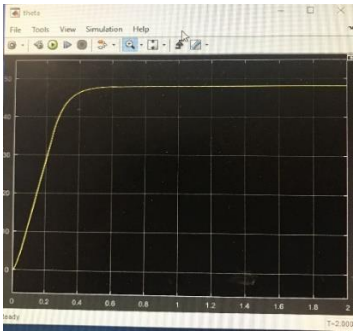

Fig. 13: HIL simulation response: PD control.

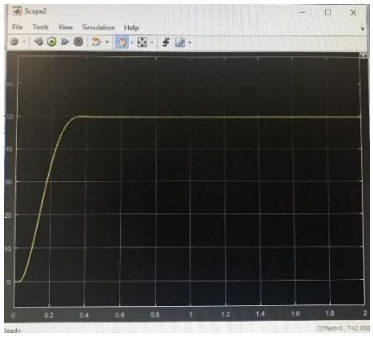

Fig. 14: HIL simulation response: PID control.

2.3. Task 3

In this task, a rotary potentiometer is attached to the shaft of the DC motor, which allows sensing and sending data back to the Arduino. To accommodate this, students have to design parts that hold the potentiometer in place. Fig. 15 shows one design that only needs laser cutting to make all needed parts. Students are now able to move the link in the prescribed manner.

Fig. 15: Motor attached to the inner band of the potentiometer.

2.4. Task 4

The final activity is to put together two position-controlled arms to construct a 5-bar linkage mechanism [7]. Students will have to first understand the theory behind the 5-bar mechanism and design it in Solidworks. At design stage, students must ensure functionality of the mechanism, limitations and the range of motion. Figs. 16 and 17 show details of a typical Solidwork model that was conceived to be entirely constructed using only laser cut technology.

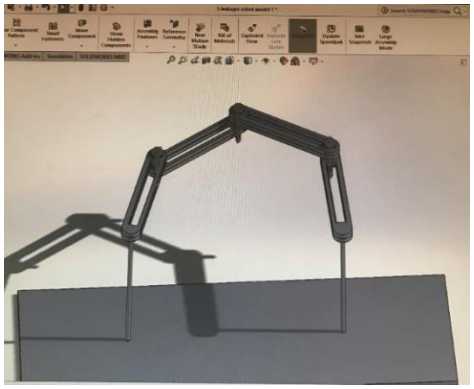

Fig. 16: Screenshot of linkage mechanism with Solidworks.

Fig. 17: Exploded view of bearings arrangement

Figure 18 shows a typical 5-bar linkage. The students at this stage should be able to program the 5-bar linkage to for example hold a pen and draw signatures. In doing so, they need to also understand forward and inverse kinematics concepts and derive/obtain equations describing the relation between the implement (to be described by them) and motor angles. Equations that are used to calculate the motor angles knowing the desired end-effector coordinates are given by Eqs. (5) and (6) [5].

$$\theta_1 = atan2(x, y) - atan2(k_2, k_1) \tag{5}$$

$$\theta_3 = atan2[\pm\sqrt{1 - \left(\frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1 l_2}\right)}, (x^2 + y^2 - l_1^2 - l_2^2)/(2l_1 l_2)] \tag{6}$$

where $k_1 = l_1 + l_2 cos\theta_3$ and $k_2 = l_2 sin\theta_3$. x and y are the coordinates of the endpoint (implement), l_1 and l_2 are the lengths of the each of the two bar linkages connected to each motor (see Fig. 19). θ_2 and θ_4 can be calculated in similar fashion as θ_1 and θ_3 where x is replaced by (x-L).

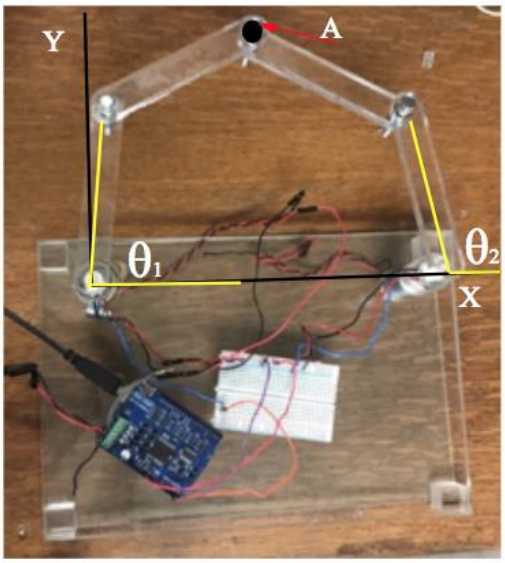

Fig. 18: Assembled mechanisms after laser cutting.

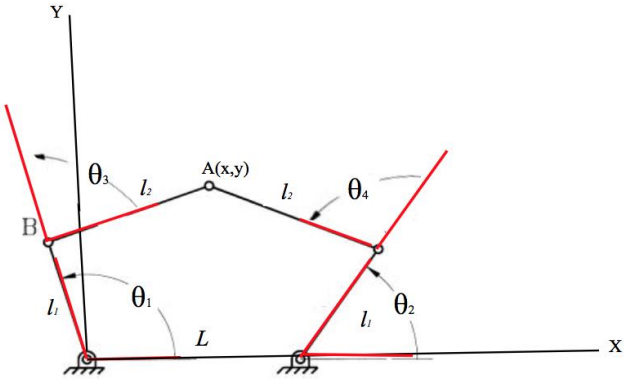

Fig 19: Schematics in which the kinematic Equations of 5 bar linkage are based on.

3. Conclusion

In this paper a low-cost project-based laboratory, activity was described that allows students to learn fundamental aspects of mechatronics. Going through this activity, students learn how to program in C to process information, design and prototype mechanical devices using engineering tools, model and simulate mechatronic systems. Furthermore, students learn how to implement hardware-in-the-loop testing based on the simulations, and then deploy their development into actual systems. Last, students are given the opportunity to integrate various engineering concepts and use/design structures, sensors and actuators to build a fully functional mechatronic device. In completing the tasks (sub-projects) as described in this paper, students have to independently learn various important mechatronics topics including efficacy of feedback control systems, modelling, simulation, creating prototypes and the integration of physical systems, sensors and control programs. Furthermore, students will develop their problem-solving skills and the ability to analyze problems so that they can develop their own solutions. These types of problem solving activities are more beneficial, in comparison with the standard university laboratories students are so often required to do since it also promotes team working and life-long learning needed in today's workforce [7].

Acknowledgments

The authors would like to thank Dr. Ehsan Jalayeri for his assistance with laser cutting and prototyping of the parts and Dr. David Khun (Head, Department of Mechanical Engineering) for the support and encouragement throughout the course of this study, which took place in summer 2017.

References

[1] D. Shetty, R. A. Kolk, *Mechatronics System Design*. PWS Publishing Co., Boston, MA, 1997.
[2] W. Bolton, *Mechatronics: Electronic Control Systems in Mechanical and Electrical Engineering*. Addison Wesley Longman Limited, Harlow, Essex, England, 1999.
[3] K. J. Astrom, T. Hogglund, *PID Controllers: Theory, Design and Tuning.* Instrument Society of America, 1995.
[4] N. Sepehri, "Demonstration of an aspect of data acquisition in mechatronics education," *Int. J. of Engineering Education*, vol. 17, pp. 538-545, 2001.
[5] The MathWorks Inc., *Matlab help documentation*. Natick, MA, 2013.
[6] N. Sclater, *Mechanisms and Mechanical Devices Sourcebook,* 5th ed. McGrew-Hill, 2011.
[7] B. J. S. Barron et al., "Doing with Understanding: Lessons from Research on Problem- and Project-Based Learning," *J. of Learning Sciences*, vol. 7, pp. 217-311, 1998.