# Motor Control Lab Manual

This document will describe a basic servo control apparatus that can be used to implement digital controllers. The first part will explain the hardware set-up, while the second part will expand on the GUI and controller software. Finally, the last part will briefly detail the modelling of the motor. The overall goal is to implement digital controllers for position control of a DC motor using encoder feedback.

## 1 Quick Startup

1. Connecting power connectors and USB for the myRIO. [Sec. 2.2]
2. Installing drivers (a pop-up window will tell if this step is necessary). [Sec. 2.2]
3. Opening "MotorLab.lvproj". [Sec. 3.2]
4. Right-clicking "myRIO-1900 (172.22.11.2)" >>"Connect". [Sec. 3.2]
5. Opening and executing "InvPendCtrlSample.vi". [Sec. 3.2]
6. Holding the inertia to the desired initial position. [Sec. 3.2]
7. Pressing the red enable button [Sec. 2.3]
8. Clicking "Run Control". Your first experiment is now running! [Sec. 3.2]
9. Downloading the experimental data and importing to Matlab. [Sec. 3.4]

## 2 Hardware Setup

### 2.1 Schematic of the Hardware

As shown in Figure 1, the hardware setup is fairly typical consisting of a Maxon DC motor being driven by a Digilent motor driver. On the motor shaft an Avago HEDL-5540-C01 encoder is attached to measure the shaft position which will be used for feedback. The encoder signals, digital outputs for the motor drive, and the controller is being implemented

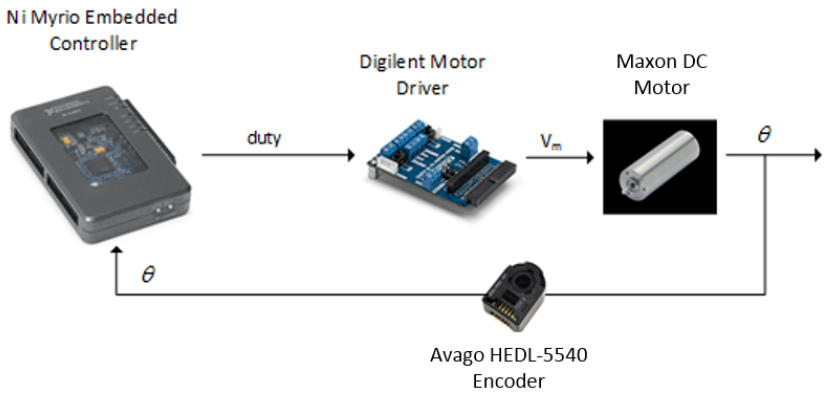


Figure 1: Overall Schematic of the Motor Control Apparatus

on a National Instruments myRIO Embedded Controller. All the wiring should be done, only two power cords and a USB connector needs to connected.

## 2.2 myRIO

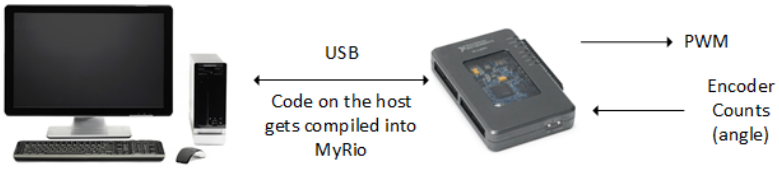


Figure 2: The myRIO is programmed using USB

The myRIO is the controller used for the motor kit. It is based on the Zync System on a Chip, which is a powerful chip that integrates processor and FPGA capabilities. Additionally, the myRIO can be programmed using Control Design and Simulation that is similar to Simulink, so the learning curve to programming is much less. For our purposes, the Myrio will be programmed to implement controller, acquire data, and send reference commands. The Myrio is programmed using a USB connected to the host computer. To drive the motor, a pwm is needed; while, an encoder counter is needed to read the position. Luckily, the myRIO has built in capability to do both of these tasks.

The kit should have came with two power connectors and USB for the myRIO. To connect the hardware, make sure the terminals shown in Figure 3 is hooked up. Once the Myrio and Digilent Driver is powered LEDS on the devices should turn on. Once these three connections are made and your operating system detects the new device, a window will pop-

 

up (Figure 4). For the first setup of the myRIO, using the "Getting Started Wizard" to help diagnose and install the recommended drivers. Alternatively, the user may manually install the required software listed on `http://www.ni.com/product-documentation/14603/en/`. After installing the drivers properly, the software is ready to be ran (See Sec. 3.2).

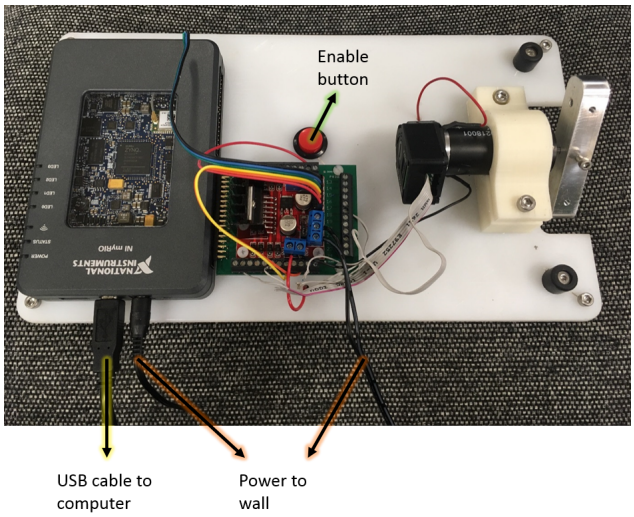


Figure 3: The Three Connections Needed to Run the Hardware

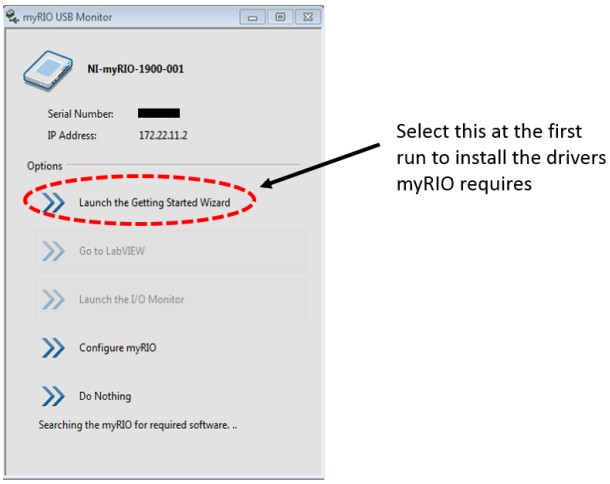


Figure 4: The auto pop-up window after plugging-in the USB.

## 2.3 Motor and Inertia

The two main parts of the plant is the motor and inertia. For modelling purposes later, it is useful to view the motor as the electrical part and the inertia as the mechanical part. The motor is a brushed permanent magnet DC motor like the one shown in figure 5. There is

permanent magnet on the stator (the non-rotating part) that creates a fixed field. The input voltage on the terminals causes current to flow through the windings which creates a torque on the armature (the moving part) that causes it to move. To make sure the torque keeps the armature moving, the current in the windings must flow in one direction this is done by the mechanical commutator. The brushes are connected to the terminal and is the part that makes or breaks contact with the different commutator regions. Therefore, to make a DC motor move, all is needed is an input DC voltage at the terminals.

Besides, a red enable button (Figure 3) is implemented in this motor kit for experimental safety. This red button opens the circuit and is closed when the button is press down. To run an experiment on the DC motor, make sure you manually press the button all the times.

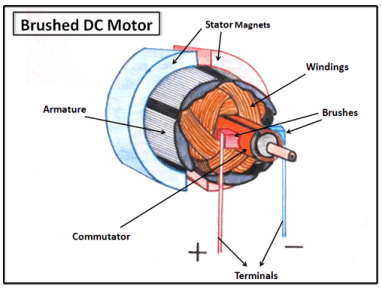


Figure 5: The DC brushed Motor (http://www.cvel.clemson.edu/auto/actuators/motors-dc-brushed.html)

The inertia is a rod that should have came with the kit and has two set screws, one in the middle and the other on the end. As shown in figure 6, three different mechanical plants are possible using the rod. Throughout the course of this manual, we will use these three different plants to highlight some of the topics in the course. As one may expect, these plants have different transfer functions and unique analysis.

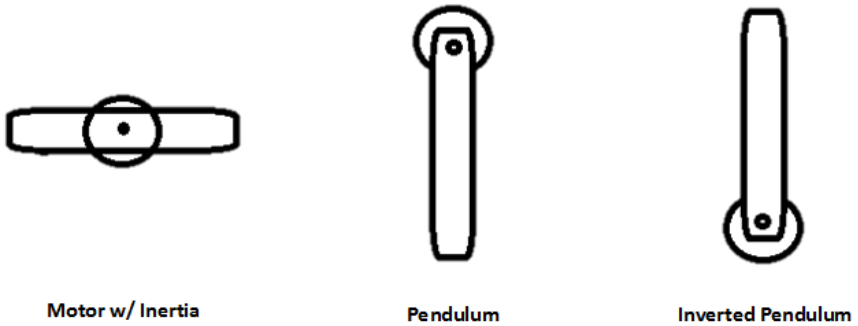


Figure 6: The Three Possible Plants

## 2.4 H-bridge

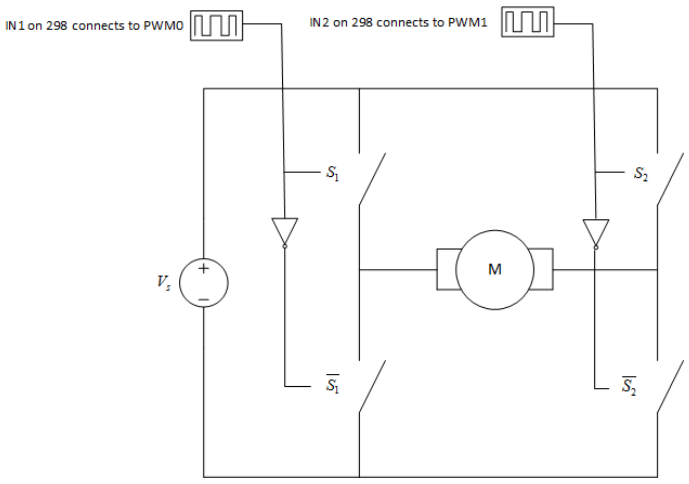


Figure 7: The H-bridge circuit to drive a DC motor

The myRIO is **only** able to output low power signal ($< 4$ mA), but the DC motor input is high power. How is this remedied? The answer is an H-bridge. An H-bridge can take a low power PWM signal from the myRIO and control the voltage input to the motor. How this is done is shown in figure 7. There are two legs in a H-bridge and we can control each individually. The first fundamental rule is the two switches in any leg cannot be closed at the same time since this will cause a short circuit, thus most H-bridges avoid this by feeding the complement to the switches. Under this constraint, there are four possible states of the H-bridge, as shown in figure 8.

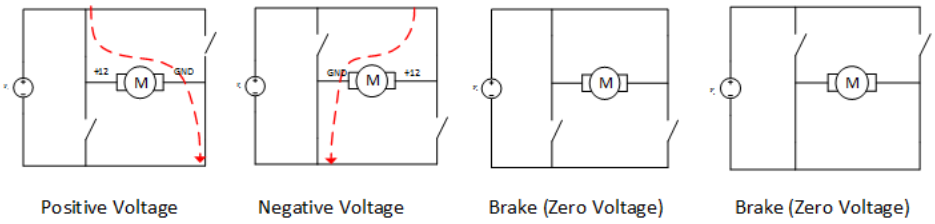


Figure 8: The four possible H-bridge states

By closing the switches correctly, the motor can have $+V_s$,$-V_s$, or 0 across its leads. Moreover, by using a correct PWM scheme a linear relationship between input duty cycle and voltage can be obtained. In equation 1, d varies from +1 to -1. Therefore, it seen that an H-bridge simply amplifies a signal.

$$V_m = V_s d \tag{1}$$

 

## 2.5 Encoder

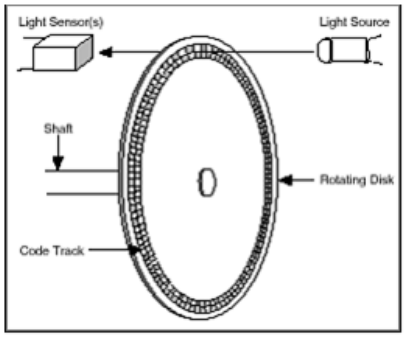


**Figure 4-10. Optical Encoder**

Figure 9: Measuring Angle with Optical Encoder

In order to analyze systems from an analytical point of a view, a measurement of the output is needed. In our case, an angular mesurement of the shaft will suffice. To that end, an Avago HEDL-5540-C01 optical encoder is attached to the motor shaft for position measurement. From figure 9, an optical uses slits on a circular disk with light sensing to measure angle. In particular, as the disk is rotating, the disk either passes or rejects the light and gives a signal as shown in figure 10.

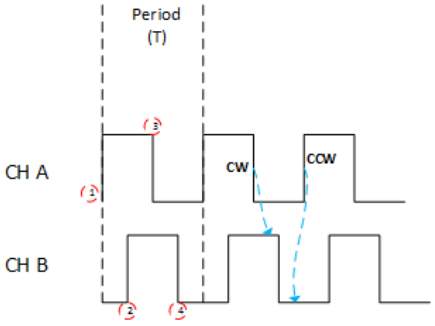


Figure 10: Output of a encoder. A count is incremented or decrement at each rising an falling edge for 4x counting. Direction is known from state of CH B at each falling edge of CH A

An encoder gives two square pulse outputs that are 90 degrees apart. By counting both the following and rising edges of each channel, the resolution can be increased 4x. For example, the encoder resolution for the kits are 100 counts per revolution, but using 4x counting the resolution can be increased to 400 counts per revolution. Another advantage of using both CH. A and B for counting, the rotation direction is known. An easy way to see this is by looking at the falling edge of CH A. If at a falling edge of A, CH B is HIGH

then the shaft is rotating one way and if B is LOW then it is rotating the other.

Note, there is no fixed "home" angle for the motor kit. The initial position when running the Labview program is used to reset the encoder value and defined as zero output angle.

# 3 Software Description

## 3.1 General Overview

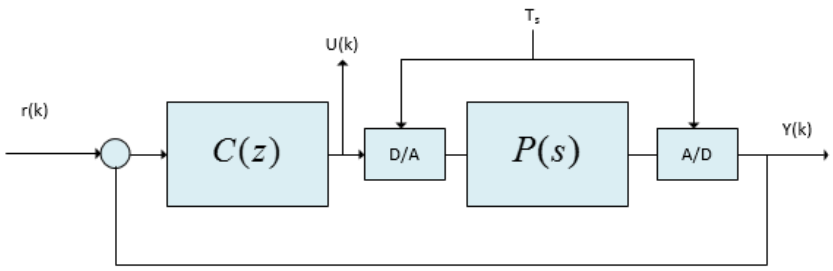


Figure 11: Control Structure Implemented on the myRIO

The controller implemented on the myRIO has the general structure shown in Figure 11. The user has control over various parameters that are fundamental to digital control design. Mainly, the controller transfer function, $C(z)$. Also, the sample time $T_s$ can be varied. Moreover, the reference command $r$ can be assigned either sinusoidal, square wave, sawtooth (ramp) with a desired frequency and amplitude, or other customized signal. The open loop motor transfer function is given in continuous time so the user can descretize based on the sample time. The next section is going to describe how to successfully run the software.

## 3.2 Opening/Using the Software

To run the software, open the "MotorKitControl_MAE171b" folder given; then open the main project "MotorLab.lvproj" file as shown in Figure 12.

Then on project explorer (Figure 13), right click the myRIO >> "Connect". The green LED should come on, then open the sample VI "InvPendCtrlSample.vi" under myRIO.

Shown in Figure 14 is the sample VI. This sample VI implements a PID controller in continuous-time domain to regulate the inverted pendulum. The sampling rate is set to be 1kHz. To start with, first run the VI before changing the parameters.

 

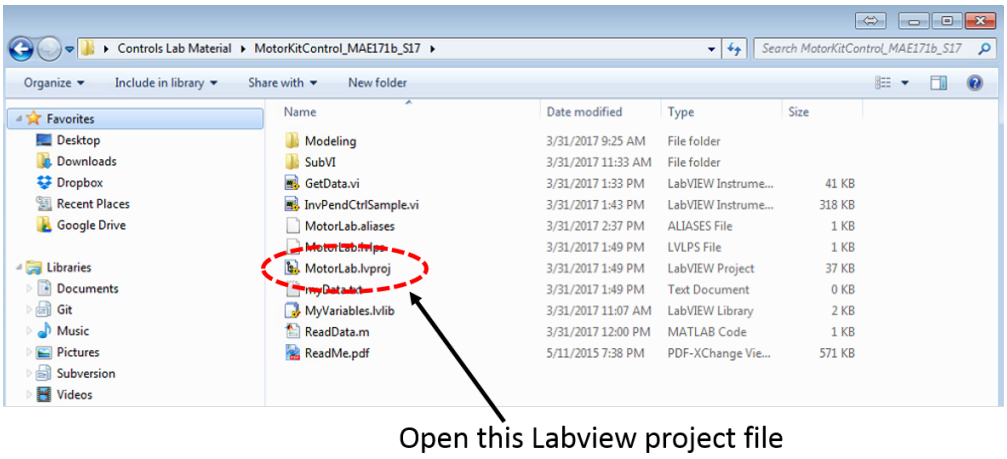


Figure 12: Open the project file

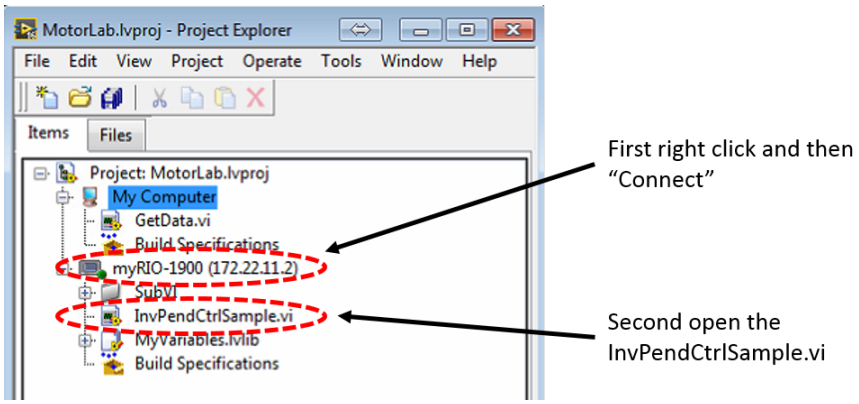


Figure 13: Connect to the myRIO and open the sample program

Next, hold the inertia to the upright position, press the red enable button (Figure 3), and then press the "Run Control" button. In this sample VI, the control reference is selected to be the last position of the inertia before clicking the "Run Control" button. The controller is then activated and the inertia holds at the initial position without assisted from any external force. The user can try to push the inertia and "feel" the torque generated from encoder feedback control. The motor can be deactivated by disabling "Run Control" button, or it will stop after default 10 seconds.

As shown in Figure 15, the real-time response data are drawn on the GUI. The upper chart shows the reference $r$ (green line) and the output $y$ (light blue line); the lower chart illustrates the commanded duty cycle $u$ (red line). These data are automatically stored in the myRIO, and can be downloaded for further analysis (See Sec. 3.4).

 

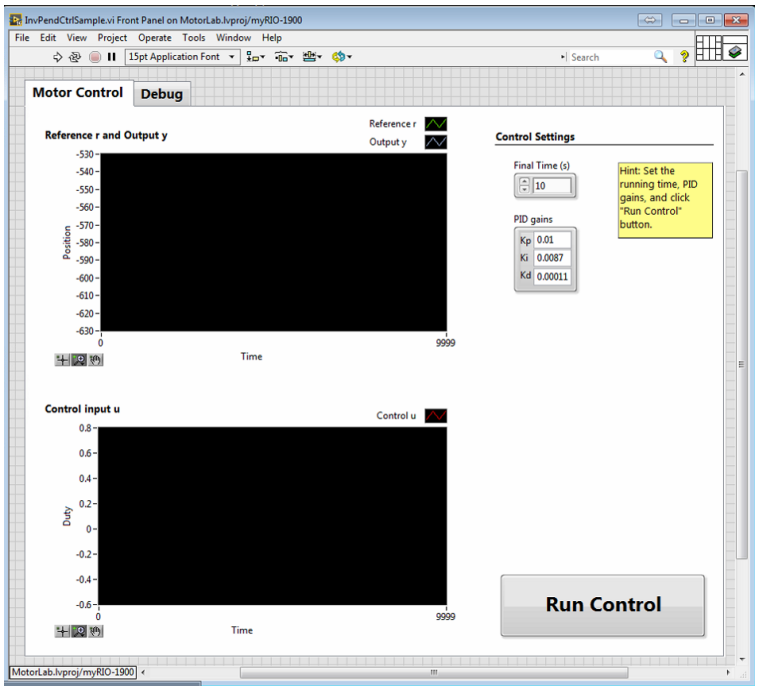


Figure 14: The sample VI - PID control for inverted pendulum.

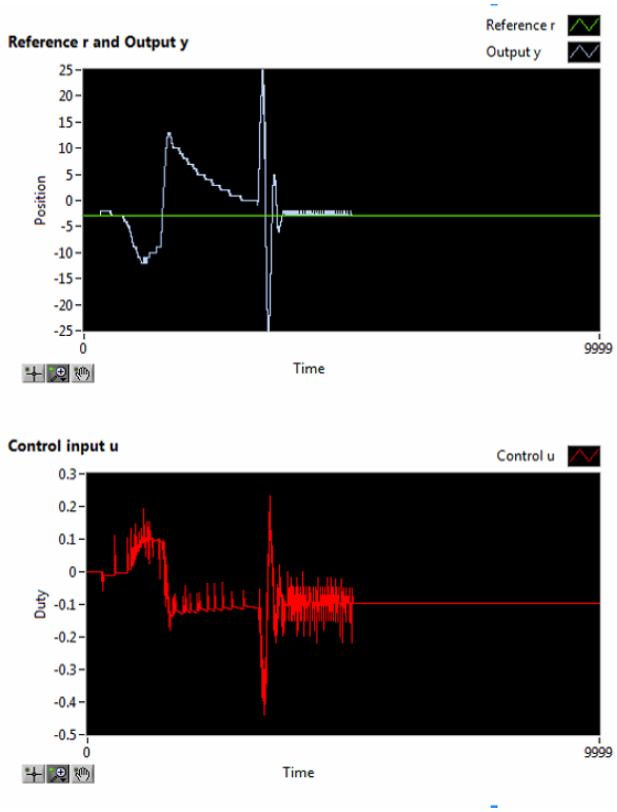


Figure 15: The real-time response charts.

### 3.3 LabVIEW Control Design and Simulation Module

Press Ctrl-E and switch to Labview block diagram. The main control loop of the myRIO is implemented under the LabVIEW Control Design and Simulation Module (CD&Sim) environment, as shown in Figure 16. Similar to Matlab Simulink, CD&Sim allows engineers build block diagrams to graphically represent dynamic systems, acquire measurements, and perform real-time feedback control. The controller can be implemented in either continuous-time domain ($s$-domain) or discrete-time domain ($z$-domain), but the sampling rate must be specified. More details about CD&Sim could be found on NI website: `http://www.ni.com/white-paper/10685/en/`.

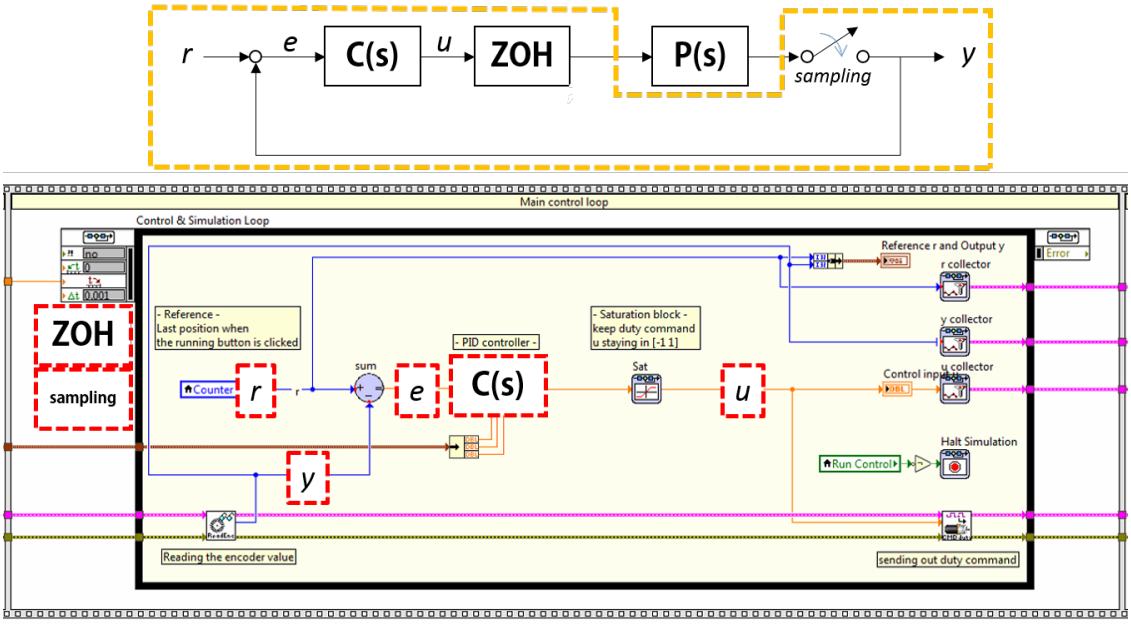


Figure 16: The main control loop

For position control of the DC motor, the main control loop contains a sub-VI for reading encoder value $y$ and a sub-VI for sending duty command $u$ to the motor driver. To keep the command $u$ staying in the designated range [-1,1], a saturation block is connected right after the PID controller.

The user is allowed to implement his/her own controller by replacing the PID control block and/or adding other blocks in the CD&Sim main loop. Here listing some useful blocks for the user's reference: Simulation/Controllers, Simulation/Signal generation, Simulation/Discrete Linear Systems, Simulation/Nonlinear Systems (Figure 17). All the blocks must be placed inside the "Control and Simulation Loop".

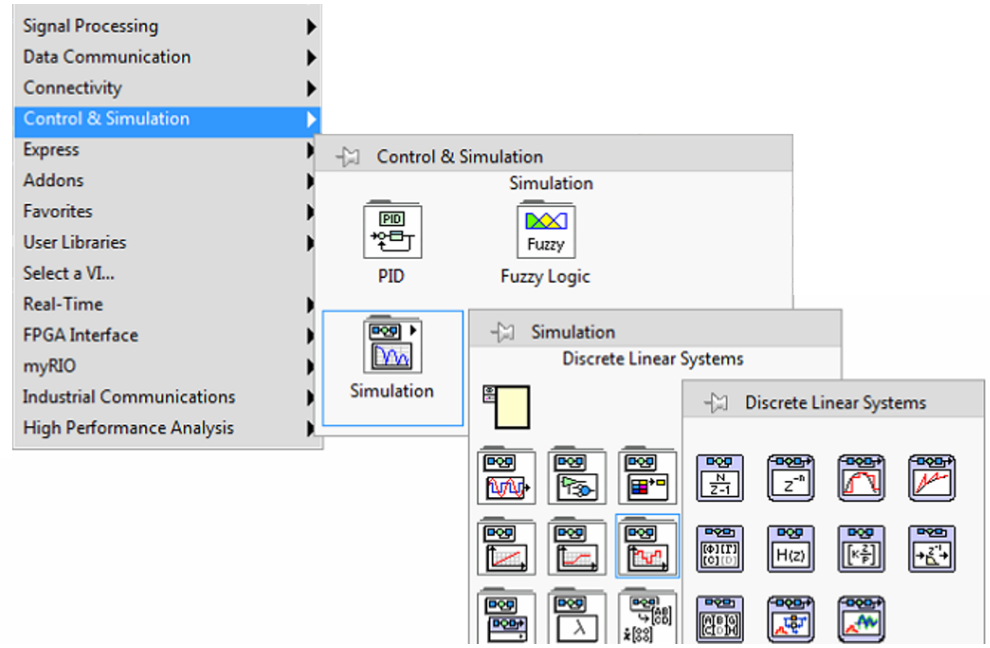


Figure 17: Blocks available in CD&Sim module

### 3.4 Data Logging and Importing to Matlab

All of the real-time response data are automatically saved on the myRIO after running the control experiment. As the example shown in InvPendCtrlSample.vi, time $t$, reference $r$, output $y$, and control input $u$ are recorded to a 2D double array (Figure 18). Other interested vectors can be also appended.

To download the data from the myRIO, go back to the project explorer (Figure 19) and then run the "GetData.vi" under My Computer. A file dialog will pop-up and ask for the desired file path.

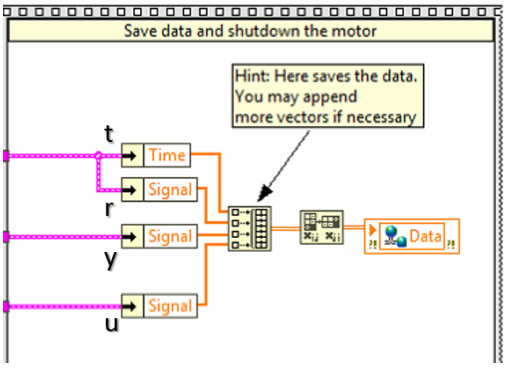


Figure 18: Data logging in myRIO

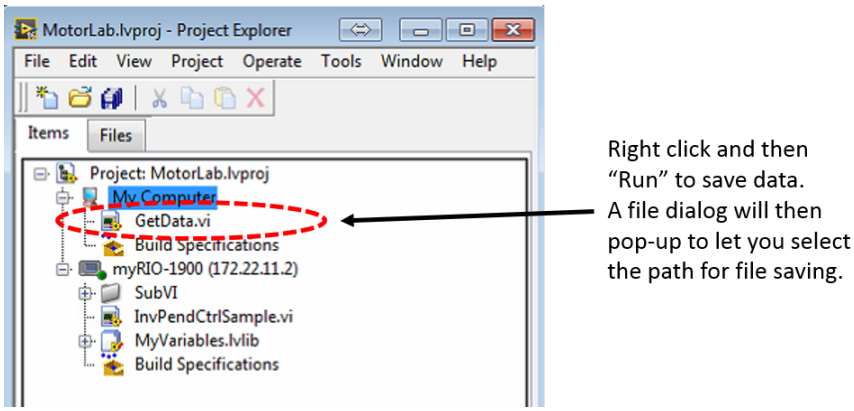


Figure 19: Download data from the myRIO

Most of people analyze experimental data in Matlab. Use "ReadData.m" to load the data into Matlab workspace. Remember to change the filename before running it.

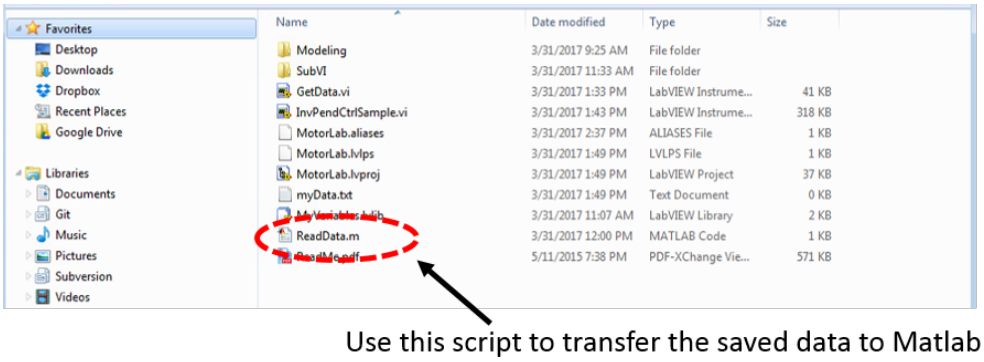


Figure 20: Run the script to load data into Matlab

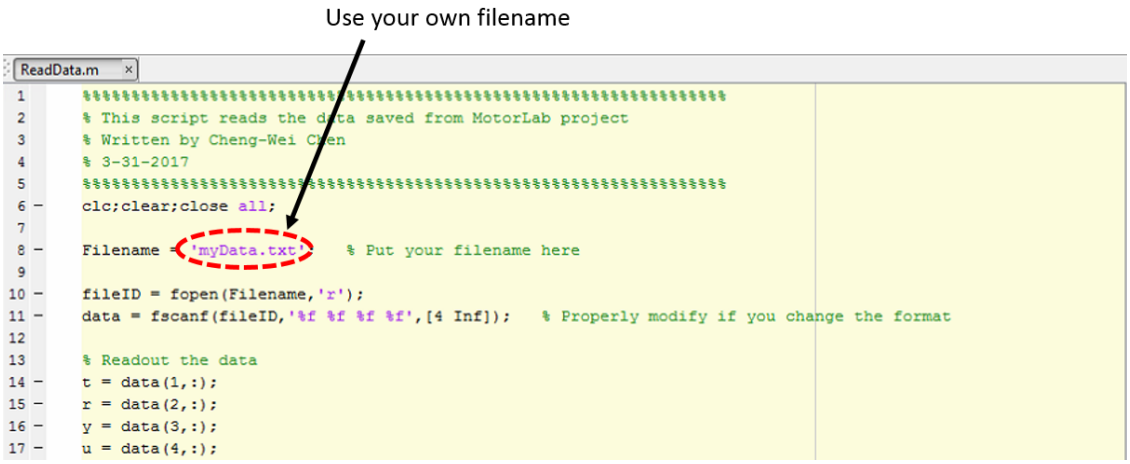


Figure 21: The Matlab script for importing experimental data stored in the myRIO.

# 4 Motor Model

In product development, system characterization is an important part of the design and control process. Many times we need to know whether the designed system meets the specifications set. In control applications, a good model is essential to test out control algorithms without endangering the actual system.

From first principles we can develop a model for the DC motor. The brushed DC motor has mechanical dynamics, electrical dynamics and the electromechanical transduction described by:

$$J\frac{d\omega}{dt} = -B\omega + T_m - T_d \tag{2}$$

$$L\frac{di}{dt} = -Ri + V_m - V_a \tag{3}$$

$$T_m = K_m i \tag{4}$$

$$V_a = K_b \omega \tag{5}$$

$$K_m = K_b = K(\text{for SI unit}) \tag{6}$$

where

- $J$ is the inertia attached to the motor
- $\omega$ is the motor shaft speed
- $B$ is viscous damping coefficient
- $T_m$ is motor torque
- $T_d$ is external torque applied to motor shaft
- $L$ is motor coil winding inductance
- $R$ is motor winding resistance
- $i$ is motor current
- $V_m$ is voltage across motor terminals
- $V_a$ is back e.m.f. voltage in the motor armature
- $K_m$ is motor torque constant
- $K_b$ is motor back e.m.f. constant

 

The particular motor drive used here takes an input with a certain pulse duration and generates a pulse width modulated (PWM) output to the motor with corresponding duty cycle and direction. The commanded value varies between -1 and 1, where 1 means 100% duty cycle, so a good controller will need to make sure the input doesn't exceed the maximum or else the controller will saturate. Also, since the 15 kHz carrier frequency of the PWM waveform is much higher than the motors dynamics, the PWM can be averaged out and considered as an analog voltage input to the motor, where the voltage input value is the supply voltage times the duty cycle:

$$V_m = V_s u \tag{7}$$

- $V_m$ is the voltage applied to the motor
- $V_s$ is the supply voltage applied to the amplifier circuit
- $u$ is the control output (-1, 1) representing PWM duty cycle with direction

The open loop plant transfer function can be calculated:

$$\omega(s) = \frac{KV_s u(s) - (Ls + R)T_d(s)}{(Js + b)(Ls + R) + K^2} \tag{8}$$

The motor inductance $L$ and damping $B$ are often negligible and this simplifies the above to a first order system:

$$\omega(s) = \frac{KV_s u(s) - RT_d(s)}{JRs + K^2} \tag{9}$$

For feedback control design, the plant model can be written as

$$\frac{\omega(s)}{u(s)} = \frac{\kappa}{\tau s + 1} \tag{10}$$

where $\tau$ represents the time constant and $\kappa$ represents the plant gain.

Notice, however, the above transfer function is from duty to speed. Since we are interested in position control (unit in rad), an integrator has be to added to make the overall transfer function

$$\frac{\theta(s)}{u(s)} = \frac{\kappa}{s(\tau s + 1)} \tag{11}$$

For the motor apparatus, the two parameters were identified using a closed loop sinusoidal frequency sweep

$$\kappa = 350$$

$$\tau = 0.6$$

 