**Hands-on Lab:**

**LabVIEW – PID**

In previous labs, the step response and the PID simulation of the LEGO DCP were performed. The former provides a Labview framework to use the gains discovered from latter to achieved PID closed-loop control.

**Concept 1:** Experimental PID control

**Step 1:** Create Front Panel and Initial Block Diagram

From your System Identification lab open the file nxtLabviewLegoDcpSystemId1\_0.vi. Save this program as nxtLabviewLegoDcpPid1\_0.vi. This program provided the framework to acquire, graphically display, and save to file the DCP’s angle. This program also allowed one to command the motor-prop. Since Step Input is no longer needed, eliminate the toggle switch. Add numeric controls to the Front Panel. These controls will allow one to set PID gains, the desired angle. Also, add a numeric indicator to display the power going into the motor-prop (see Figure 1-1A left).

For the Block Diagram, delete the Case Structure that was associated with the toggle switch (but keep the NXT motor Block). Blocks will be added, so re-arrange the Block Diagram to provide suitable space (Figure 1-1A right).

**Step 2:** Create the error signal and proportional control

PID control works on the error signal. The error is the difference between the desired and actual angle (in degrees). The proportional component of PID control scales this error. As such, add a difference and multiplication block (Figure 1-2A).

**Step 3:** Differentiate the error and derivative control

The derivative component of PID control scales the rate change of error. Mathematically, this is for discrete-time systems where is the sample time is, is the current error and is the previous error. In Labview, shift registers were used to store the previous value in a loop. As such, add a shift register (Figure 1-3A) for the error signal.

In the Block Diagram, the loop uses a 25 millisecond Wait block. Thus, for this program . Add a division block to perform the differentiation (Figure 1-3B). Lastly, add a multiplication block that serves to multiply the error derivative with a derivative gain value.

**Step 4:** Integrate the error and Integration control

The integration component of PID control, integrates the error. Mathematically, this is for discrete-time systems. Thus, add a multiplication block to take the product of the current error and sample time seconds (Figure 1-4A).

Drag an addition block into the loop. Feed one input of the addition block with the. Next, wire the output of the addition block as shift register (Figure 1-4B).

Lastly, drag a Multiplication block to form the product of the error sum and the integration gain (Figure 1-4C).

**Step 5:** Form the control signal and dead-zoning

Drag 2 Addition Blocks and wire the summation of the proportional, integral, and derivative commands (Figure 1-5A).

Motor power is limited from 0% (stop) to 100% (full power). Dead-zoning is used to limit signals to within this range. Add the In Range and Coerce block (Programming – Comparison – In Range and Coerce)[[1]](#footnote-1). Right-click this block and make sure both Include upper limit and Include lower limit are checked (Figure 1-5B). Add and wire constants 0 and 100 to the lower and upper limits respectively, wire the PID command signal to the input of the In Range and Coerce Block (Figure 1-5C). The numeric indicator (Motor Power [%]) has been moved to this region of the loop as well.

Wire the PID command signal to Motor Power numeric indicator. This will allow one to monitor this value when tuning gains. From the In Range and Coerce block, wire the coerced(x) output in the input of the Motor Power block (Figure 1-5D). Save your program (e.g. nxtLabviewLegoDcpPid1\_0.vi)

**Step 6:** Execution

From the Simulink PID lab, one simulated that. Before executing the program, set these values in the appropriate numeric controls and set the desired angle to 35 degrees (Figure 1-6A). One can make these values default as well (Front Panel – Edit – Make Current Values Default). One can now run the program.

From simulation, one expects that with such gains, the DCP should slowly rise to the desired angle. Thus, the LEGO DCP should also slowly rise to 35 degrees if the Labview program is correct (Figure 1-6B left). Data was saved to a file (e.g. labviewPidOutput-111518-12H34M.csv) and plotted in Excel (Figure 1-6B right). One observes from the data plot, that slow increase in the pendulum’s angle which converges to 35 degrees.

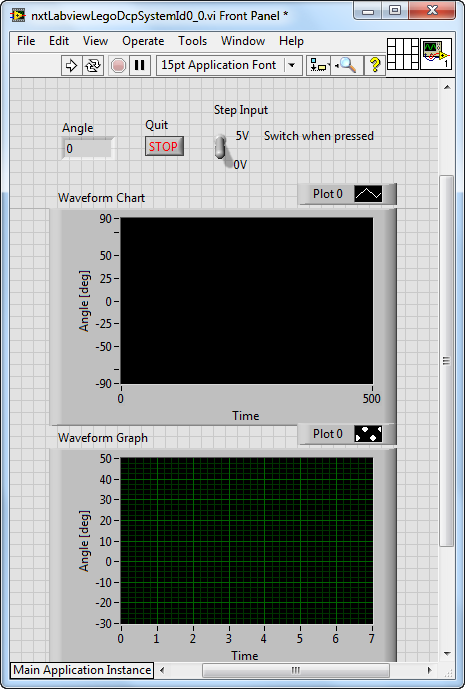
**Blah: Next time, start adding screen grabbed images to the above Steps.**

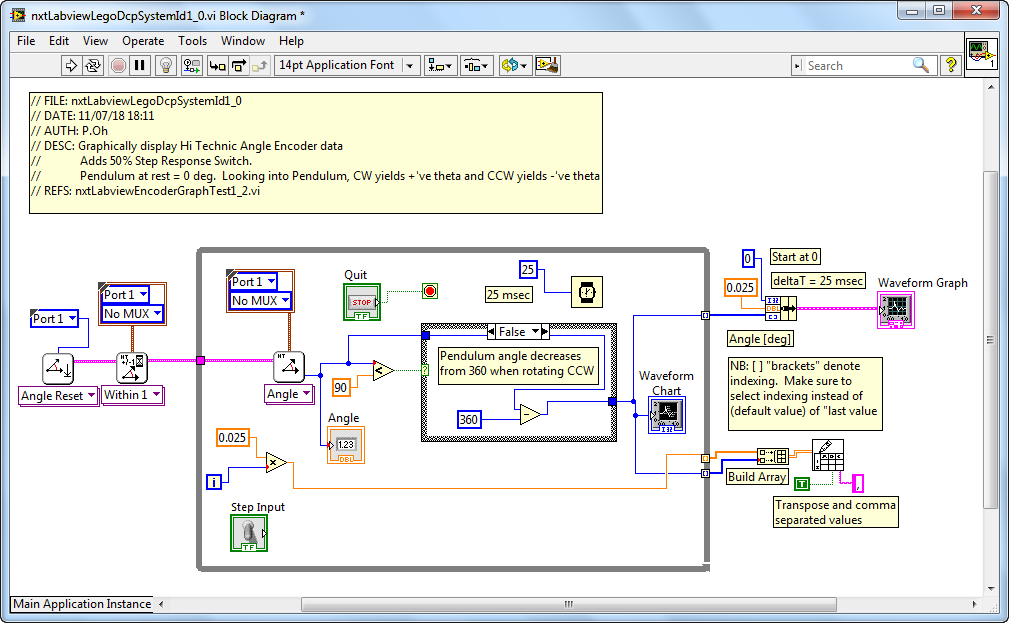
e nxtLabviewEncoderGraphTest1\_2.vi. Recall that you wrote this file in Exercise 2-2. This Labview program consisted of a waveform chart, an XY chart, and wrote angle data every 25 milliseconds to a CSV file.

Save this file as nxtLabviewLegoDcpSystemId1\_0.vi. Note **Errata 1 and Errata 2** in **Figure 1-1A right**.

Add a toggle switch (MINDSTORMS Robotics – Boolean – Vertical Toggle Switch) to the Front Panel and add text to this switch (**Figure 1A left**). Right click this switch to make sure that the Mechanical Action is set to Switch When Pressed. In the Block Diagram, drag the toggle switch so that it is inside the while-loop (**Figure 1A right**).

**Figure 1-1A:** The toggle switch is in the Front Panel (left) and inside while-loop of Block Diagram (right). **Errata 1:** The block diagram incorrectly compares the angle to 90. **This should be 180**. Also, **Errata 2**: the numeric indicator should be re-wired to be after the Case Structure (e.g. next to the waveform chart). Refer to **Figures 2B and 2C**, which are correct

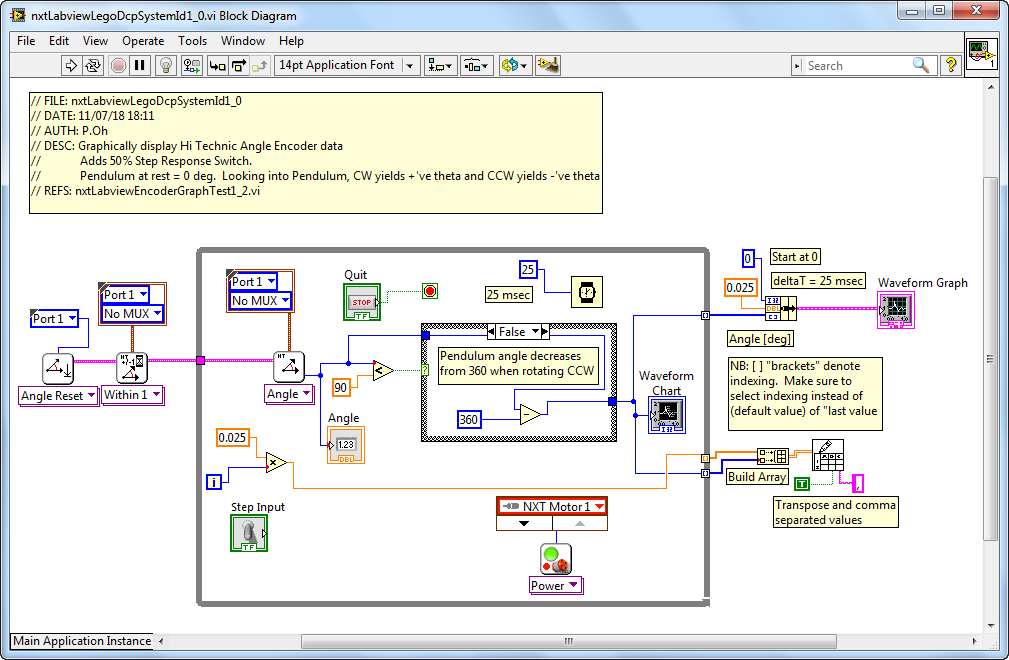




**Step 2:** Add a Motor Block to the Block Diagram

The toggle switch serves as a step input to the motor; when flipped on, a 50% motor power command will be fed into the motor. To achieve this, first add the NXT motor block (MINDSTORMS Robotics – I/O – Move Motors) into the while-loop. By default LEGO Mindstorms calls this block Power. Right click on the Power block and select Create Constant. Hovering over the created constant, select NXT Motor 1 (**Figure 1-2A**).

**Figure 1-2A:** Motor added to block diagram (called Power) with constant NXT Motor 1 as the assigned port. NB: The figure is incorrect, recall **Errata 1** and **Errata 2** from Figure 1A. Refer to **Figures 2B and 2C**, which are correct

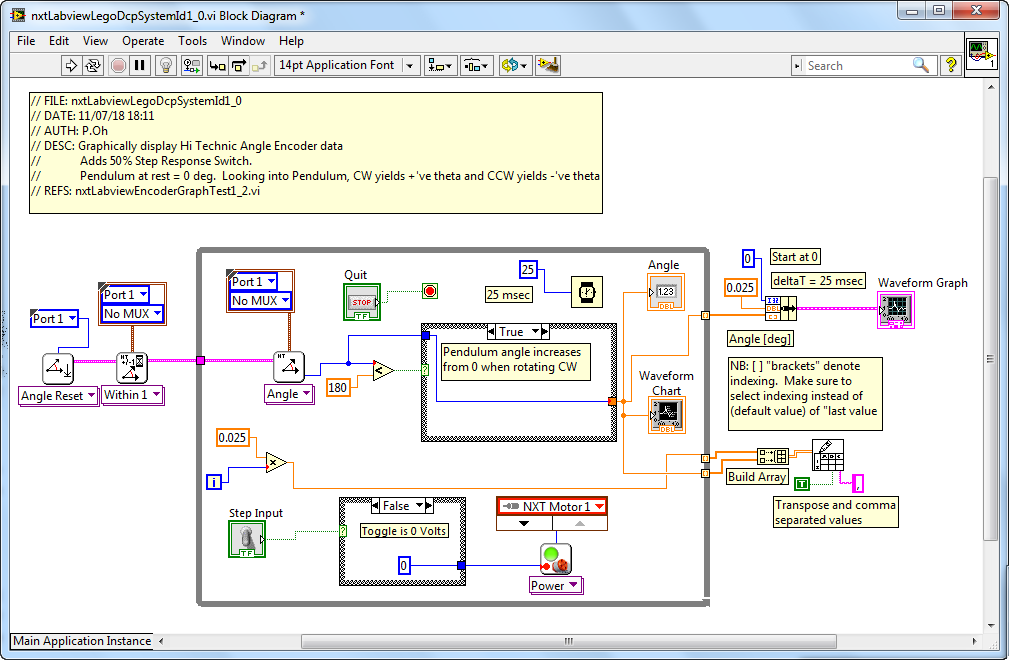


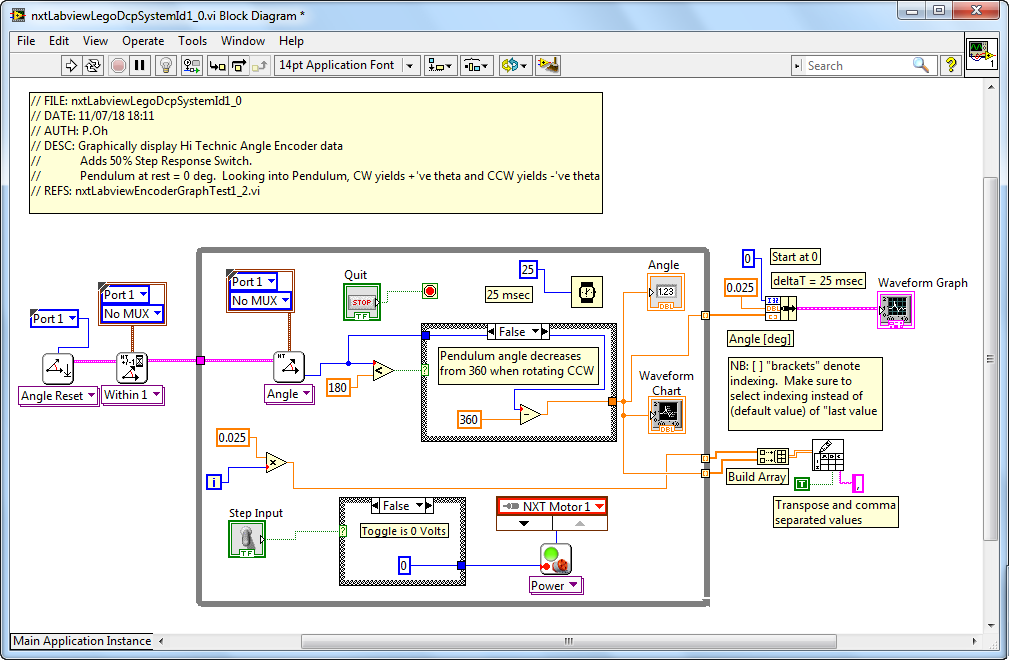
Next, place a Case Structure between the toggle switch and motor. For the True condition, add a numeric constant. Set this constant to 50, and wire it thru the Case Structure and into the motor’s Power input (**Figure 1-2B**).

For the False condition, add a numeric constant with the value of 0. Wire this constant to the Case Structure (**Figure 1-2C**). The net effect is when the Labview program is played toggling the switch will set the Case Structure to True and consequently issue a value of 50 into the motor power (and thus activating the motor). Conversely, when the switch is False, 0 is fed into the motor power (essentially deactivating the motor).

Save your program (called nxtLabviewLegoDcpSystemId1\_0.vi).

**Figure 1-2B:** A Case Structure is added into the Block Diagram. For the True condition, the numeric constant is fed into the input of the motor (Power block). The toggle switch is wired to the Case Structure. NB: NB: **The figure here is correct**; it incorporated the aforementioned errata





**Figure 1-2C:** For the Case Structure’s False condition, a numeric constant of zero is fed into the motor input (Power block). NB: **The figure here is correct**; it incorporated the aforementioned errata

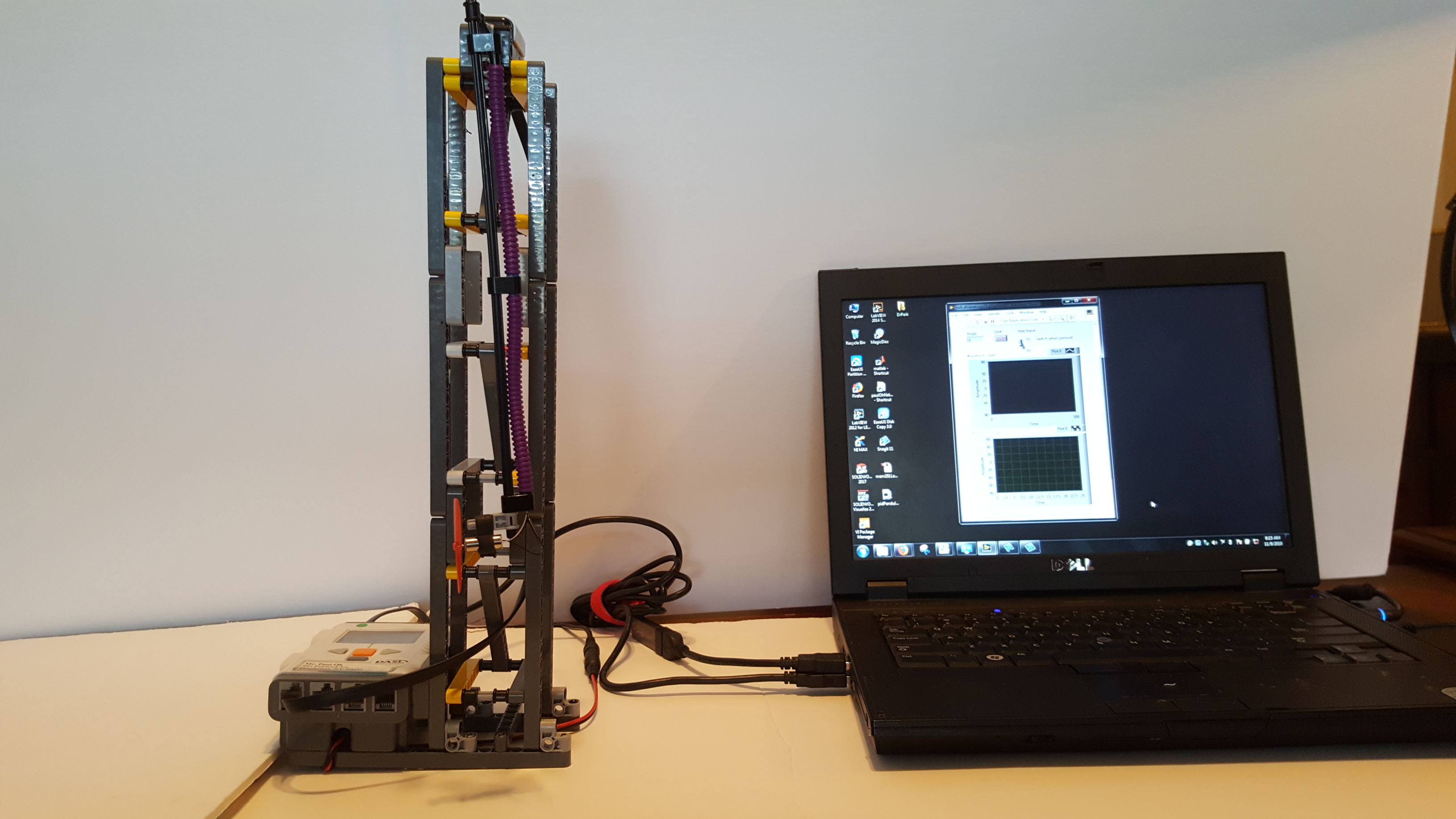
**Step 3:** Execution

A step response will be performed on the LEGO DCP (see **Figure 1-3A**). The above block diagram shows the angle sensor and motor connections. Thus, connect your NXT cable from the Hi Technic Sensor to the Port 1 of the NXT and your spliced NXT cable to Port A of the NXT.

Since this program will activate the motor-prop, make sure to clear any obstacles the pendulum could possibly hit. Execute the program (nxtLabviewLegoDcpSystemId1\_0.vi) and observe the waveform chart. Toggling on the switch will send 50% power to the motor and the pendulum will start to rotate clockwise. Toggling off the switch stops the motor. Repeat toggling on and off to ensure the LEGO DCP and your program is working as expected (**Figure 1-3B**).

Hit the front panel’s Stop button. A pop up box will prompt you to enter a file name. For example, saving as **systemId-110918-07H41M.csv** suggests the data was saved on 11/09/18 and 7:41 AM as comma-separated values (CSV) file format.

Opening this CSV file with Excel and plotting the values (**Figure 1-3C**) yields what was captured and displayed on the front panel’s XY chart

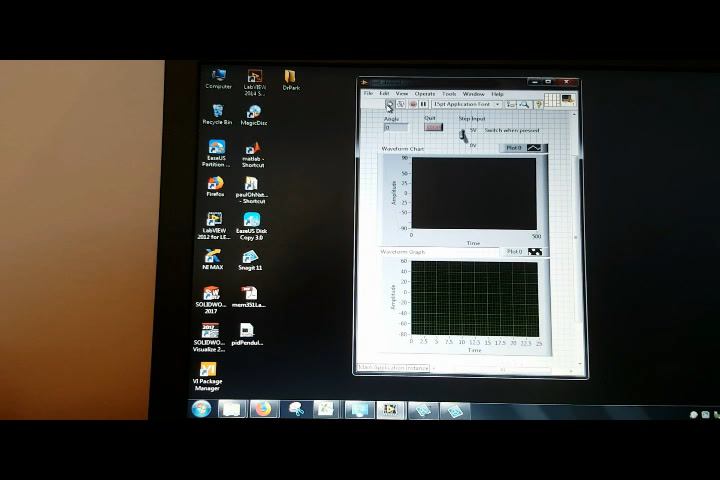


**Figure 1-3A:** LEGO DCP connected to laptop. Note: no obstacles in the pendulum’s pathway. Angle sensor and Motor are connected to the Brick’s Port 1 and Port A respectively.

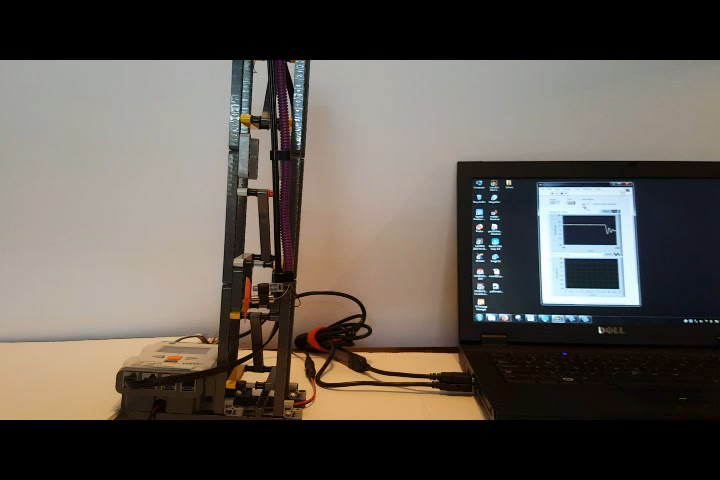
**Figure 1-3B:** Sequential photos from a video. Top Row: Program is executed (left); pendulum at rest (middle); and switch toggled on and activates motor-prop (right). Bottom Row: motor-prop generates thrust which raises pendulum and waveform chart provides real-time display of angle (left); toggling motor off, stops motor and pendulum returns to zero angle position (middle); toggling switch on again raises pendulum and waveform chart shows angle response (right).

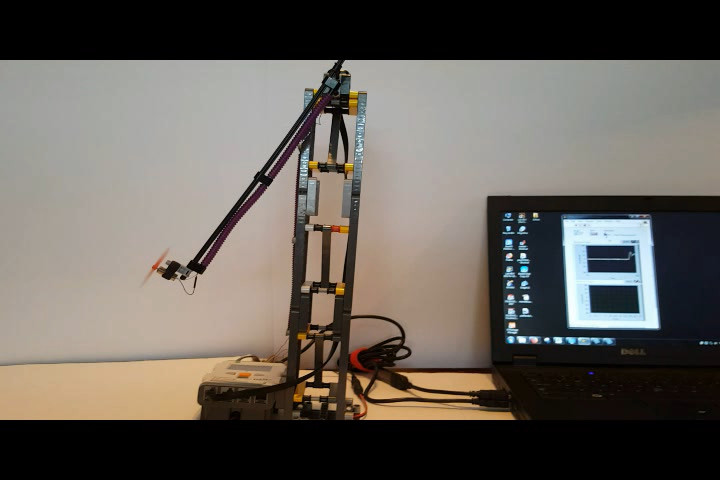




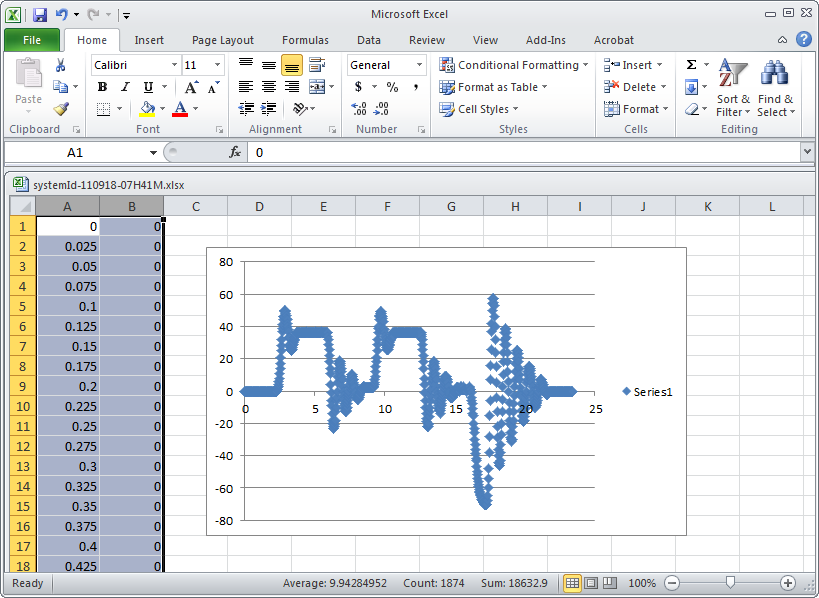








**Figure 1-3C:** Opening the CSV file in Excel, and performing a scatter plot, shows the LEGO DCP’s angular response to a step input. One sees (around 2.5 seconds) that apply 50% motor power (due to toggling the switch on), has the pendulum reach a peak angle (about 50 degrees) and oscillates to reach a steady-state angle of about 31 degrees.



**Exercise 1:**

* 1. Complete the following table

|  |  |  |  |
| --- | --- | --- | --- |
| Your Step Response Input (i.e. motor power) [%] | | Your Steady-State Angle [deg] | |
| e.g. 50% |  | e.g. 31 deg |  |

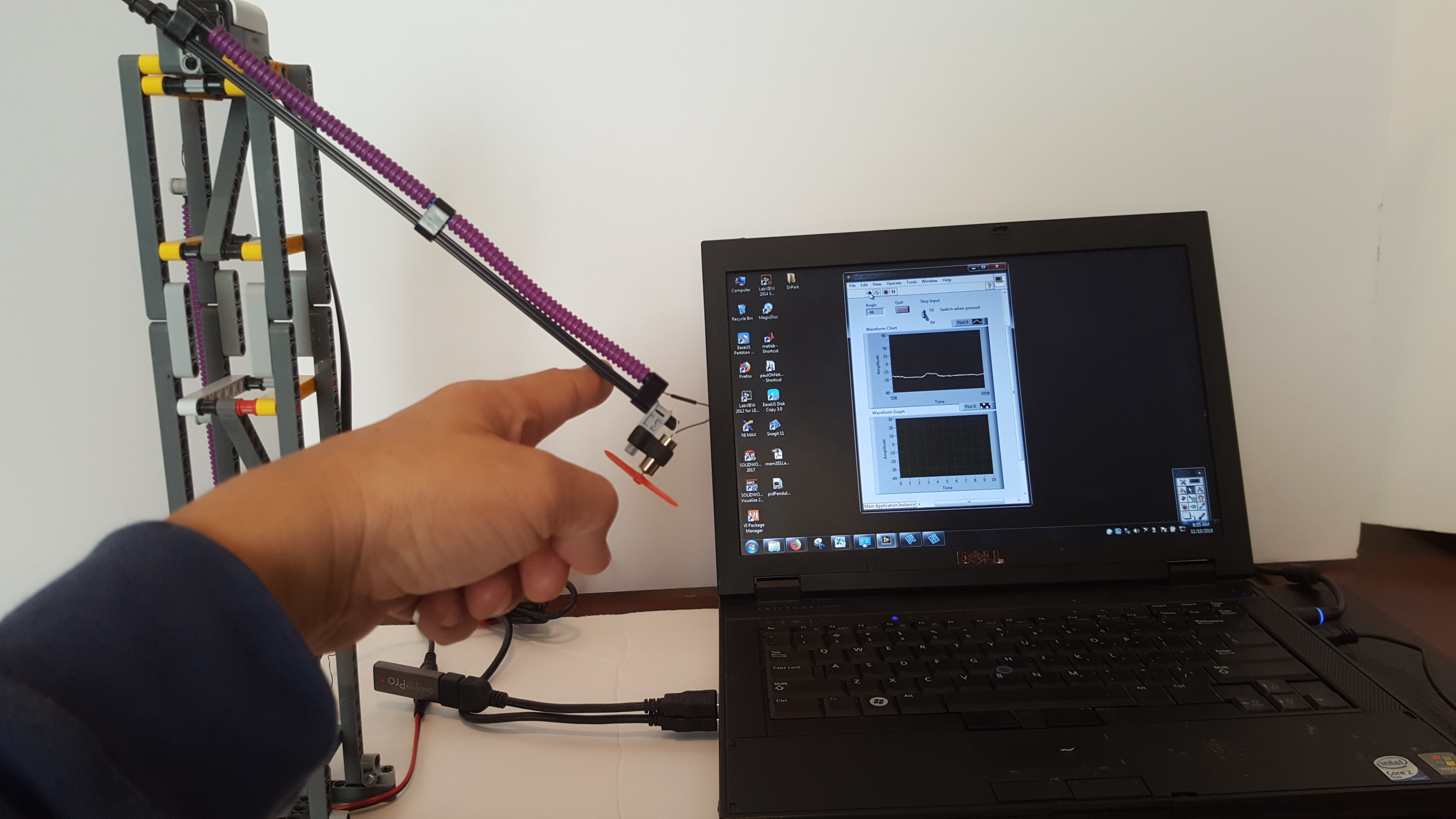
**Concept 2:** Free fall data capture and Logarithmic Decrement

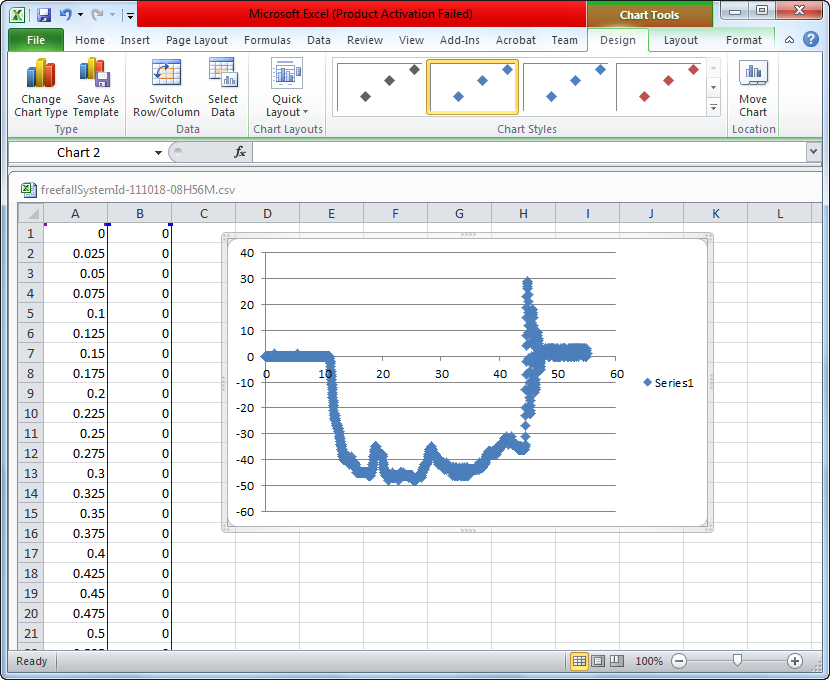
Pendulums, like the LEGO DCP, when released from an initial angle, will oscillate. The oscillations’ amplitudes become smaller as the pendulum swings back-and-forth, until the finally stop. This is a characteristic of damped second-order systems like the LEGO DCP. The amplitudes and period are related to the system’s damping constant and damped natural frequency. This relationship will be shown in this concept. Such system identification serves to find a mathematical model (called transfer function) of the system (i.e. plant).

**Step 1:** Execute nxtLabviewLegoDcpSystemId1\_0.vi

Here, toggling will not be used. Instead, run the program and rotate the pendulum counter clockwise (CCW) to an initial angle (e.g. -45 degrees) and release (**Figure 2-1A left**). Once the pendulum is motionless, hit the Stop button and save the data e.g. legoDcpFreefall-111018-08H56M.csv. Graph the data in Excel as a scatter plot (**Figure 2-1A right**).

**Figure 2-1A:** Program is executed on laptop and pendulum moved counter clockwise to about -45 degrees (left). Scatter plot of captured data in Excel (right).





**Step 2:** Label data points for peaks and calculate damping ratio and natural frequency

The data of interest in **Figure 2-1A right**, is the oscillatory peaks. Recall that the pendulum’s angle gets smaller as time elapses. The amplitudes of subsequent peaks are related to the pendulum’s damping ratio and natural frequency thru a logarithmic decrement (1):

(1)

Here, is the th peak and is the th peak (for any) are the amplitudes of subsequent peaks. The period of the oscillations is also related thru (2)

(2)

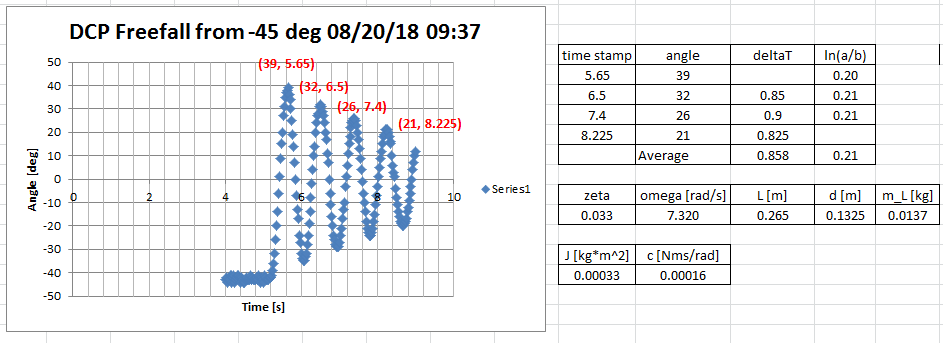
Using (1) and (2) and with the definition of one can show (3)

(3)

and

**Figure 2-2A left** focuses the scatter plot on the oscillatory peaks. Point your mouse over each peak to reveal the associated angle and time. For clarity, these angles and times are added as text boxes over each peak. The plot’s title and axes are also labeled. **Figure 2-2A right** implements equations (1)-(3) in Excel. Here, the average period and logarithmic decrement were used.

**Figure 2-2A:** From an initial angle of about -45 degrees, the pendulum oscillates for about 4 times. The angle and time of each peak was identified. The peaks are labeled with this data (left). The calculated logarithmic decrement, damping ratio, and coefficient of friction from equations (1)-(3) were calculated in Excel (right)



**Congratulations: LEGO Damped Compound Pendulum System ID Completed**

**Exercise 2:**

* 1. Complete the following table

|  |  |  |  |
| --- | --- | --- | --- |
| Your Damping Ratio | | Your Natural Frequency | |
| e.g. 0.033 |  | e.g. 7.320 |  |
|  |  |  |  |

* 1. Referring to the free body diagram, complete the following table
  2. Using compute the Moment of Inertia (e.g. 0.00033 )
  3. Using compute the coefficient of friction (e.g. 0.00016 )

thrustTesterFbd012905

|  |  |  |  |
| --- | --- | --- | --- |
|  | Pendulum Length [m] | e.g. 0.265 [m] | Your value |
|  | Pivot to CG distance [m] | e.g. 0.1325 [m] |  |
|  | Mass of Pendulum [kg] | e.g. 0.0137 [kg] |  |

1. The In Range and Coerce block is in the Labview Programmable Palette (not the Mindstorms Robotics palette). If needed, enable the Labview Program Palette (Function – Customize – Change Visible Palettes and check the Programming box). [↑](#footnote-ref-1)