Experiment A11
Chaotic Double Pendulum
Procedure

Deliverables: Checked lab notebook, Brief tech memo

Background

Measuring and controlling the angular position and velocity of rotating mechanical parts is often a critical aspect of mechanical design. In this lab you will learn how to use a sensor called a **quadrature rotary encoder** and an **Arduino microcontroller** to measure the angular position of a chaotic double pendulum. The quadrature encoder contains a slotted wheel, which is mounted to the rotating shaft. Illustrated in Fig. 1, the slotted wheel rotates through two photogates (not unlike the Photogates in the second lab), and an electronic pulse train is produced. The Arduino microcontroller is used to count the number of pulses, and the number of counts in proportional to the angle of rotation. Using two Photogates (A and B) allows it to determine the direction of rotation. The encoders you will use in this lab have 2000 slots, so a full 360° rotation results in 2000 counts.

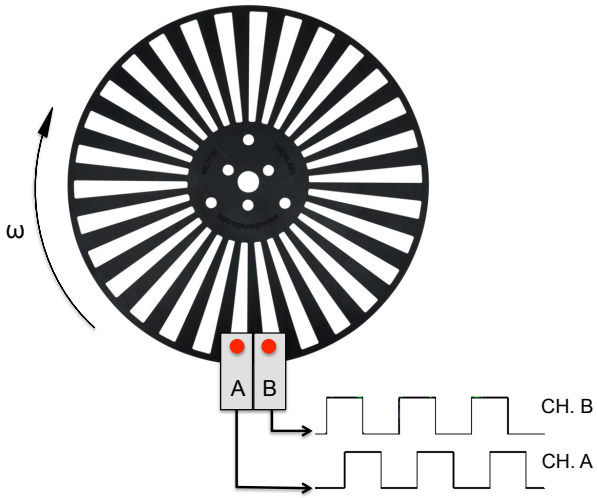

Figure 1 – A schematic illustrating the operation of a quadrature encoder.

In this lab, you learn how to used quadrature encoders to measure the angular position of a rotating mechanical part. You will then examine the predictable behavior of a single pendulum and compare it to the erratic behavior of a chaotic double pendulum.

Part I: Sensor Implementation

Please follow the instructions below to connect the encoder to the Arduino and initialize data collection from the Arduino.

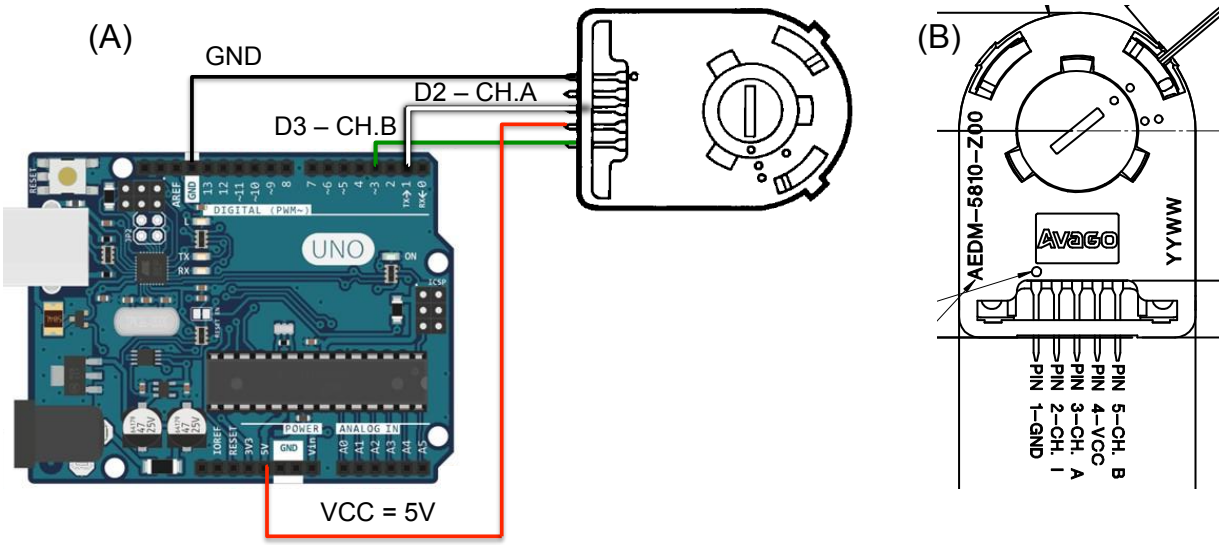

Figure 2 – (A) A wiring diagram illustrating how to connect the rotary encoder to the Arduino. **(B)** A pinout for the rotary encoder. Pin 1 is connected to ground (GND – BLACK) on the Arduino. Pin 2 is not connected. Pin 3 (CH. A - WHITE) is connected to digital input D2. Pin 4 (VCC - RED) is the 5V supply voltage. Pin 5 (CH. B - GREEN) is connected to digital input D3.

1. Make sure the locking bolt is in the pendulum, such that θ_2 cannot rotate. **Do not over tighten the bolt! You will wreck the bearings!**
2. Make sure the USB cable is NOT connected to the PC.
3. Connect the 5 pin female connector of the cable to the θ_1 encoder such that the black wire is on top. Connect the male pins on the other end of the cable to the Arduino board as shown in Fig. 2. Black is ground, red is 5V, green is pin 3, and white is pin 2.

   WARNING: Incorrect wiring can severely damage or break the encoder! Make sure it is properly connected before moving on to the next step.

4. Connect the Arduino to the lab bench PC with the USB cable. A green light should appear and remain on the Arduino. If it does not, please ask the lab instructor for assistance.
5. Open the folder on the desktop titled "rotary_encoder" and double click the program "rotary_encoder". By default, this should open with the Arduino IDE software. Take a moment to admire the code and try to understand what it will do.
6. Click on the "Tools" menu in the Arduino IDE software. On the drop down menu, go to "Port", and select the one that says COM# "(Arduino/Genuino Uno)", where # will be a number corresponding to the COM port being used for the Arduino. **Write down the number of the COM port corresponding to the Arduino in your lab notebook.**

7. Click on the button with a checkmark (✔) symbol to compile the program, then click the button with an arrow (➔) next to it, to upload the program on the Arduino.
8. Next, you will use a program called PuTTY to save the serial output from the Arduino to a text file.
9. Launch the PuTTY software from the desktop. Select "Session" on the left and check "Serial" as the connection type. Make sure the baud rate is 9600, if it is already not set to the value. Then, enter the COM# you wrote down earlier in step 6.
10. Select the "Logging" tab right below session on the left side of the PuTTY app. In the option below "Session logging" select "Printable output". Click "Browse" to choose the file name and destination. Give it an appropriate name and save it on the computer as a .txt file.
11. Click the "Open" button, and a black terminal window should appear. Give the pendulum a tap and allow it to swing to and fro for a few cycles. You should see numbers appear in the terminal corresponding to the encoder count, time in ms, and displacement angle in degrees.
12. Close the window, and find the .txt file you specified in step 10. Open it, and check to make sure it contains the data that was printed to the terminal.
    a. The column on the left is the number of counts. (This particular encoder has 2000 counts per revolution.)
    b. The middle column is the angle in degrees.
    c. The far right column is the time in milliseconds is in the far right column.
13. Make a rough plot of θ_1 vs. t in Matlab or Excel just to check that the data looks good. (It should like a damped harmonic oscillator if you did it correctly.)

Part II: Single Pendulum

Now that you have the sensor correctly wired and collecting data, it is time to explore the physical behavior of the pendulum.

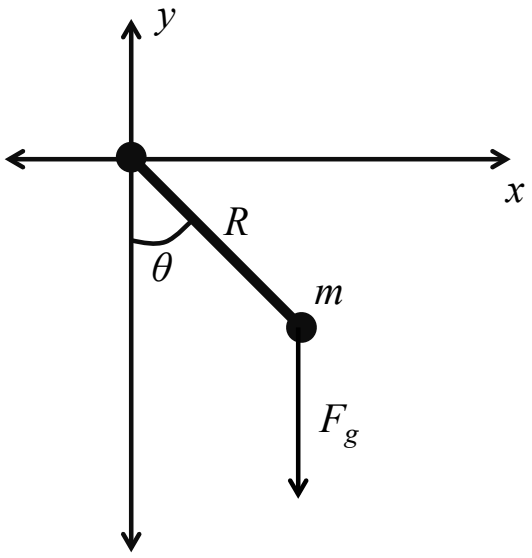

Figure 3 – A schematic illustrating the geometry of the single pendulum.

With the locking bolt in place, we essentially have a single rigid pendulum with only one degree of freedom θ. Illustrated in Fig. 3, it has a mass m with a center of mass a distance R away from the axis of rotation. Balancing the angular acceleration with the torque from gravity and a viscous drag force from the surrounding air yields the equation of motion for the single pendulum

$$mR^2\ddot{\theta} = -mgR\sin\theta - \gamma R^2\dot{\theta}, \quad (1)$$

where γ is the viscous drag force coefficient. If we assume the θ is small, then $\sin\theta \approx \theta$, and Eq. (1) can be re-written as

$$\ddot{\theta} + \frac{\gamma}{m}\dot{\theta} + \frac{g}{R}\theta = 0. \quad (2)$$

If you have taken differential equations, you should recognize this as the equation for a damped harmonic oscillator. As we have seen in previous labs, this equation can be easily solved, and the behavior is fairly predictable. You will now use the angle encoder to examine the motion of the pendulum and compare it to the solution to Eq. (2).

1. Repeat steps 7 – 11 of Part I to measure the angle vs. time for the pendulum as it oscillates, but rather than tapping the pendulum, lift it up to the stopper and let it go. Repeat this *three* times, and save your data in three separate files. Email the files to your lab partner.
2. Use the encoder counts on the far left column of the data set to recompute the angle θ_1 in units of radians. Recall that the encoder has 2000 counts per revolution, so each encoder count corresponds to $2\pi/2000$ radians of rotation.

3. With your favorite of the three data sets, use a first order finite difference to numerically compute the angular speed

$$\omega_1^i = \frac{\theta_1^i - \theta_1^{i-1}}{t^i - t^{i-1}} \qquad (3)$$

where i is the integer index of the i^{th} data point and $\omega_1{}^i$ is the angular speed at the time t^i. (Note that you will have to begin you calculation with the 2nd data point.) Make sure the final units of angular speed are rad/s.

4. Use the 'yyaxis' command to plot the angle θ_1 (degrees) vs. time t (s) on the left y-axis, and the angular speed ω_1 (rad/s) vs. time (s) on the right y-axis. Zoom in on the data, so you are not plotting "dead time" at the beginning or end of the data, where the pendulum is not doing anything.

Part III: Chaotic Double Pendulum

You will now add an extra degree of freedom θ_2 by removing the locking bolt. Adding this extra degree of freedom will result in *chaotic* behavior. This means that the long-term behavior of the system is extremely sensitive to initial conditions, and the tiniest perturbation in the initial angle will result in completely different behavior. In pop-science, this is often referred to as "the butterfly effect". This effect makes it difficult to control many complex systems, such as a robotic arm or a large financial system.

The chaotic double pendulum in this portion of the lab is simple at a glance. Illustrated in Fig. 4, it is essentially a normal rigid pendulum with a second rigid pendulum mounted to the end of it, hence the name "double pendulum".

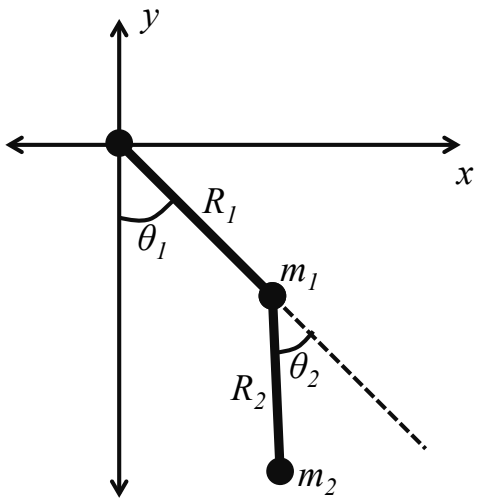

Figure 4 – A schematic illustrating the geometry of the chaotic double pendulum.

The equations of motion for the double pendulum are quite complex. Balancing the various torques with the angular accelerations results in a system of two coupled, non-linear differential equations

$$(m_1 + m_2)R_1\ddot{\theta}_1 + m_2R_2\ddot{\theta}_2\cos(\theta_2 - \theta_1) = m_2R_2\dot{\theta}_2^2\sin(\theta_2 - \theta_1) - (m_1 + m_2)g\sin\theta_1 \qquad (3)$$

and

$$R_2\ddot{\theta}_2 + R_1\ddot{\theta}_1\cos(\theta_2 - \theta_1) = -R_1\dot{\theta}_1^2\sin(\theta_2 - \theta_1) - g\sin\theta_2\,, \qquad (4)$$

where g is the acceleration of gravity. There is no known analytic solution to these equations, so they are often solved numerically to yield the transient behavior $\theta_1(t)$ and $\theta_2(t)$. Interestingly, the numerical solution is highly sensitive to the initial condition. That is, if you perturb the initial angles at $t = 0$ by just a fraction of a degree, it will result in completely different behavior at later times. This mathematical and physical behavior, known as *chaos*, was only discovered in the last 50 years and has had a profound impact on the way scientists and engineers view the world and analyze complex systems.

1. With the locking bolt removed, repeat the steps from the previous sections to initialize data collection with the PuTTY software. Be sure to give your data set a descriptive file name.
2. Rotate θ_1 up to the metal bar and rotate θ_2 to 180°, such that the lower pendulum blade is inside the upper blade.
3. Release the pendulum, and numbers should appear in the PuTTY terminal. When the pendulum is finished moving, close the terminal and check that the data was properly saved. Repeat this three times. Check the data and email it to your lab partner.
4. As you did in the previous section, compute the angular speed in units of rad/s using a first order finite difference.
5. Use the 'yyaxis' command to plot the angle θ_1 (degrees) vs. time t (s) on the left y-axis, and the angular speed ω_1 (rad/s) vs. time (s) on the right y-axis. Zoom in on the data, so you are not plotting "dead time" at the beginning or end of the data, where the pendulum is not doing anything.
6. Qualitatively compare the plots of θ_1 vs. t and ω_1 vs. t for the single pendulum and double pendulum.

Data Analysis and Deliverables

Using LaTeX or MS Word, make the following items and give them concise, intelligent captions. Make sure the axes are clearly labeled with units. Plots with multiple data sets on them should have a legend. **Additionally, write 1 – 3 paragraphs separate from the caption describing the plots/tables. Any relevant equations should go in these paragraphs.**

1. For the single pendulum, use the 'yyaxis' command to plot the angle θ_1 (degrees) vs. time t (s) on the left y-axis, and the angular speed ω_1 (rad/s) vs. time t (s) on the right y-axis. Describe the behavior in the caption.
2. For the double pendulum, use the 'yyaxis' command to plot the angle θ_1 (degrees) vs. time t (s) on the left y-axis, and the angular speed ω_1 (rad/s) vs. time t (s) on the right y-axis. Describe the behavior in the caption.

Talking Points – Please address the following questions in your paragraphs.

- Compare and contrast the trajectory of the double pendulum and single pendulum.
- Does the numeric derivative have more noise or less noise than the original signal? How does this compare to the numeric integration in the A9? Did the numeric integration in A9 increase or decrease the amount of noise?

Appendix A

Equipment

- Double pendulum with Avago AEDM-5810-T06 encoders
- Lab stand with pendulum mount and stopping bar
- Arduino UNO microcontroller
- Workstation PC with Arduino IDE and PuTTY software
- Cable to connect encoder to Arduino
- USB cable to connect Arduino to workstation PC