RENSSELAER MECHATRONICS

Reading from Analog and Encoder Sensors. Outputting to PWM (for Nano)

Part 1: Reading an Analog Signal

Objective:

- Read analog input position of the potentiometer
- Control the brightness of a LED using pulse width modulation (PWM)
- Understand how to read a quadrature encoder works

Background Information:

Analog to Digital Converter (ADC):

Arduino's onboard *analog to digital converter (ADC)* converts an analog signal to a digital value. The following equation relates the input voltage to the converted digital value:

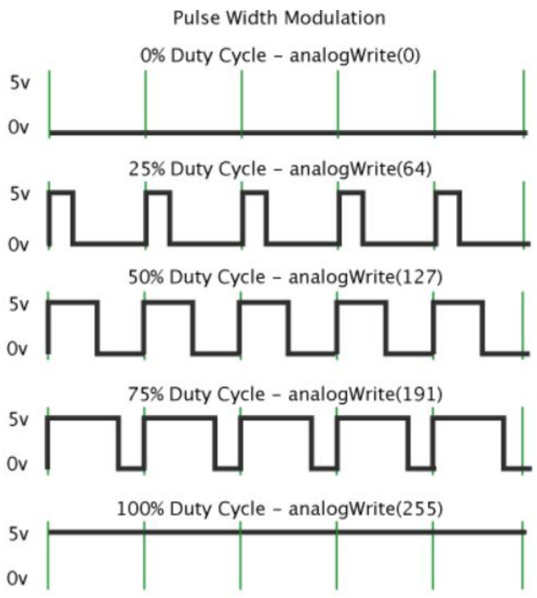
$$\frac{A/D_{result}}{ADC_{resolution}} = \frac{V_{in}}{V_{ref}}$$

- V_{in} is the input voltage – typically from a sensor or in this case a potentiometer
- V_{ref} is the reference voltage (usually 5v or 3.3v)
- $ADC_{resolution}$ is the resolution of the ADC conversion, in this case it is 10 bits (1023)

Question: If the value of the ADC on A0 of the Arduino is 512, what is the input voltage on A0 (assume the Arduino operates at 5v)?

Pulse Width Modulation (PWM)

If the microcontroller does not have Digital to Analog Converter (DAC) then it can only output a binary high or low voltage (5v or 0). *Pulse width modulation (PWM)* is used to create an *average* voltage by varying the duty cycle, or the amount of time the signal is on/off. A 100% duty cycle corresponds to fully powering the LED with 5v and a 0% duty cycle means the LED will receive 0v. A 50% duty cycle means *on average* the voltage seen by the LED is 2.5v. The value assigned to the PWM block can be from 0 to 255: 100% duty cycle corresponding to 255 and 0% duty cycle corresponding to 0.



PWM duty cycle (https://www.arduino.cc/en/Tutorial/PWM)

Simulink Model

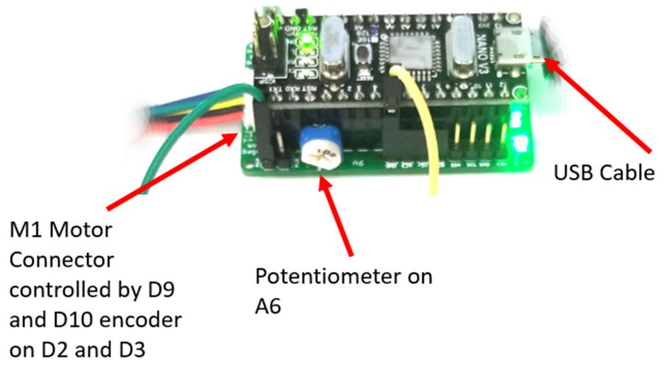
It is easiest to start with a Simulink model that is already set up for hardware implementation. The Demo files for your hardware should be used for this purpose. You can always run the demo file if you think you might have a hardware problem.

After installing RASPlib, right click on the Rensselaer Arduino Support Package to open the library, under demos open the demo for your hardware, save it to another location under a useful name.

Delete everything in this diagram except the blocks you want to use. (these blocks have different names depending on your hardware system).

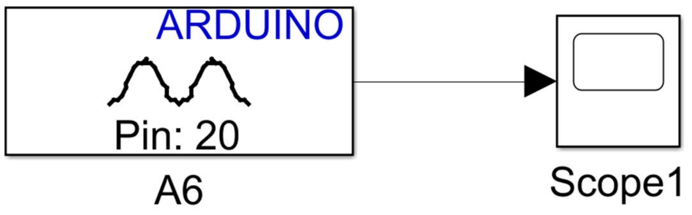
Analog Input

The potentiometer on the MinSegNano is the white circle located on the side of the Shield. Rotating it changes the resistance/voltage read by pin A6:


USB Cable
M1 Motor Connector controlled by D9 and D10 encoder on D2 and D3
Potentiometer on A6

Build and run the following Simulink diagram with the following settings.

(Note: The “POT in” and “PWM out” blocks both use the ‘Scope’ block found under ‘Sinks’)



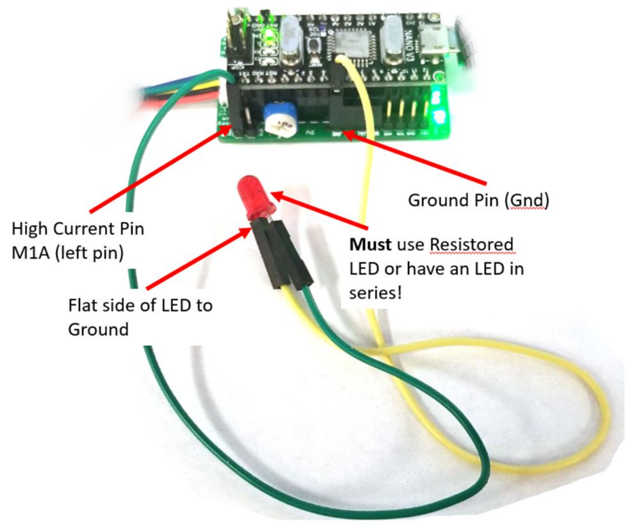
- **Configuration Parameters**: The “Fixed-step size” should be .03 seconds.
- **Analog Input**: Set the Pin number to 6 and the sampling time to -1. When the sample time is -1 the block will inherit the sample time used in the simulation settings (in this case .03).

Observe the scopes while changing the position of the potentiometer wiper. Click the ‘Autoscale’ button to see the full range of the graphs.

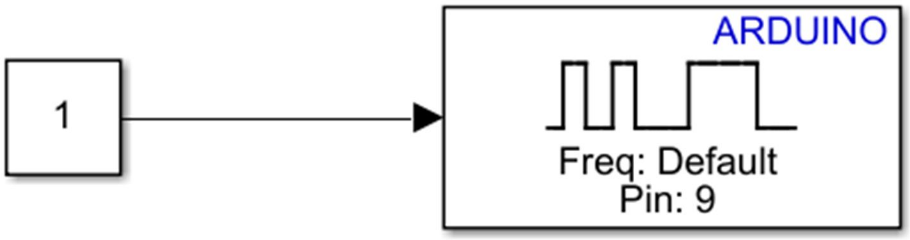
Questions: Provide a plot of the analog input as you change the potentiometer position. What is the max and minimum values received?

PWM Output

For the MinSegNano the onboard LED connected to pin 13 is a digital pin, and not a PWM pin. For this lab an external LED needs to be connected. Connect a resistor with the ground pin (flat side of LED) to GND, and with the opposite side connected to the M1A jumper pin as shown. Please note that the M1A pin is powered high current pin. It is driven by Pin 9 on the nano and amplified by the motor drive. This means that the LED used must have a resistor placed in series with it or you could burn out the LED. The resistor shown in the diagram below has a built in resistor.


Ground Pin (Gnd)
High Current Pin M1A (left pin)
Must use Resistored LED or have an LED in series!
Flat side of LED to Ground

Build and run the following Simulink diagram with the following settings.



- **PWM:** Set the Pin to 9.
  - **Note**: If pin 13 is used for an output you might have to go the configuration settings and unselect “detect on overrun”

Questions: Change the values of the constant and observe the change in brightness. What is the max number that can be specified? What is the smallest? What happens if a negative number is used – explain what is occurring? What happens if you specify numbers above the max – explain what is occurring?

Part 2: Reading Encoders

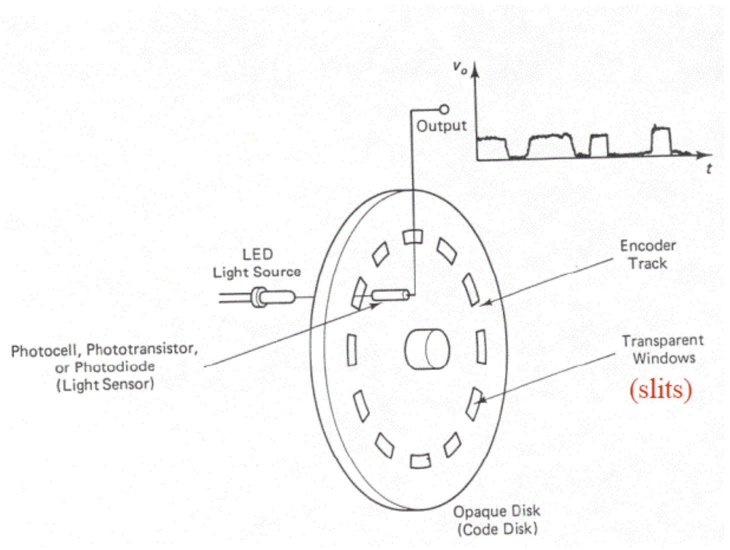
Objective:

For Part 2, an encoder will be used to determine the position of the output motor shaft. The brightness of the LED will be controlled based on the position.

Background Information NXT Motor Encoder:

A Lego NXT motor has encoder wheel inside that measures the rotation of the motor shaft. An encoder is a device that can be used to determine position. Most encoders use optical sensors to output a pulse train which can be decoded to determine position and direction. In this lab, the encoder will be used to track angular position.

Figure 1: Lego NXT Motor



- The NXT encoder has a raw resolution of 12 counts. The figure on the left below show

Figure 2: Example of Optical Encoder signals

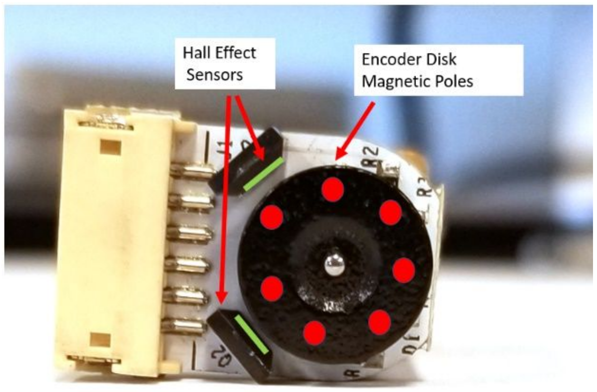
the actual encoder wheel in the NXT motor. It has 12 holes. The optical sensor will output a high or low (1 or 0) depending on the position of this encoder. Each revolution will produce 12 pulses.

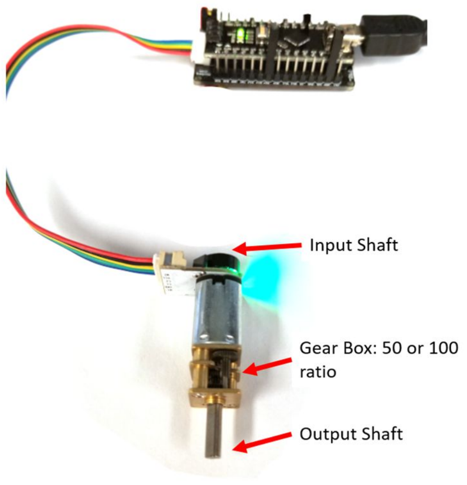
- If the output motor shaft moves 1 revolution the encoder wheel turns 15 revolutions – this means for **one revolution of the motor output shaft there will be 15*12 = 180 pulses from the encoder.**
- The encoder device provides two channels A & B. These channels are the encoder pulses with one channel shifted by 90 degrees. By monitoring these two channels both the position and direction of the motor shaft can be determined. **If quadrature decoding is used the resolution can be increased by 4 times**.
- The Encoder block included with this lab performs quadrature decoding so the total resolution is:

$$\boldsymbol{Anglular\ Resolution = 15 * 12 * 4 = 720\ pulses\ per\ revolution\ of\ output\ shaft}$$

Background Information Micro Gear Motor:

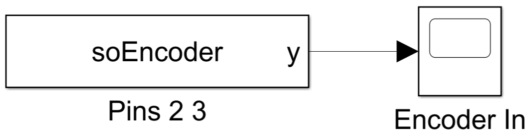
The encoder uses an encoder disk that has magnetic poles placed in the round disk. There are 7 for this encoder disk. The two hall effect sensors measure the poles as they pass the sensor they proved an On/Off signal to the Arduino. For every revolution of the **input** shaft the **quadrature decoded output** will be 4*7 = 28. For every revolution of the **output** shaft the **quadrature decoded output** will be 4*7*Gr, where Gr is the gear ratio of the gearbox (30, 50 or 100 depending on your motor)





Simulink Model

Build and run the following Simulink diagram. The encoder block can be obtained from the Demo file for your hardware located in the Examples folder of RASPLib.

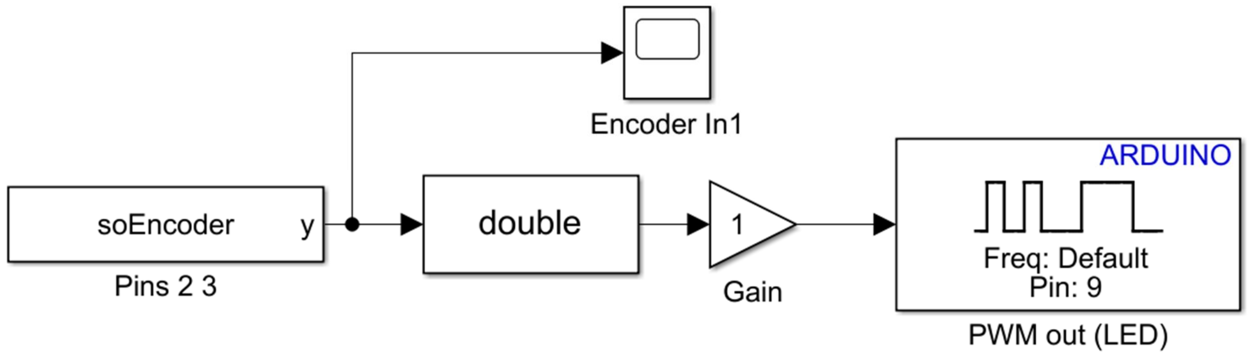


Turn the motor output shaft (or the small encoder wheel attached directly to the motor). Observe that the scope captures the change in the motor position.

Questions: Rotate the input motor shaft 1 revolution (approximately). How much does recorded position change? Rotate the output shaft 1 revolution (approximately). How much does the position change? What is the gear ratio for your motor?

Simulink Model

Build the following Simulink model.



Question: What value does the gain need to be so that 1 revolution of the motor output shaft will obtain the maximum value of the PWM? What happens if the motor is turned more than 1 revolution? What happens when the encoder position is negative?

Checkpoint 2:

Run the simulation and observe the scope and the LED when you rotate the encoder's wheel clockwise and counter-clockwise.

Verify that one revolution of the output motor shaft results in the value obtained for your gear ratio. You can also add a 'Sinks -> Display' block to see exact values.

Be prepared to demonstrate LED brightness changes to fully bright after 1 revolution.