Reading from Analog and Encoder Sensors. Outputting to PWM (for Nano33IoT)

Part 1: Reading an Analog Signal

Objective:

- o Read analog input position of the potentiometer
- o Control the brightness of a LED using pulse width modulation (PWM)
- o 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 12 bits (4096)

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 3.3v)?

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)*
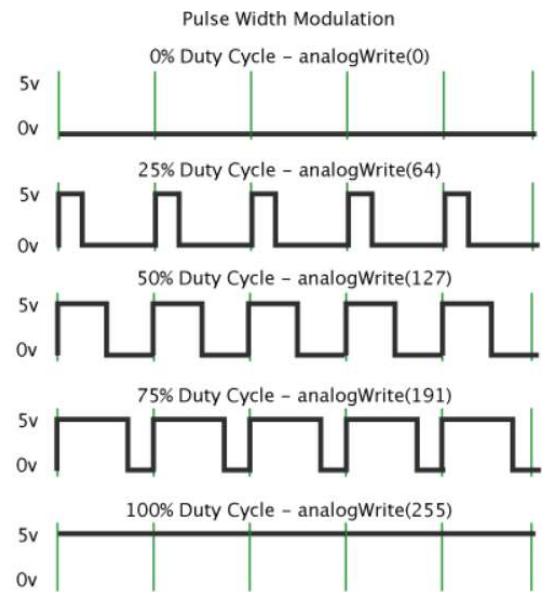
Simulink Model

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.

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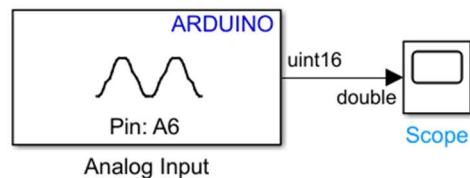
Analog Input

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:

Potentiometer on A6
Easy to break – do not
turn hard

USB Cable
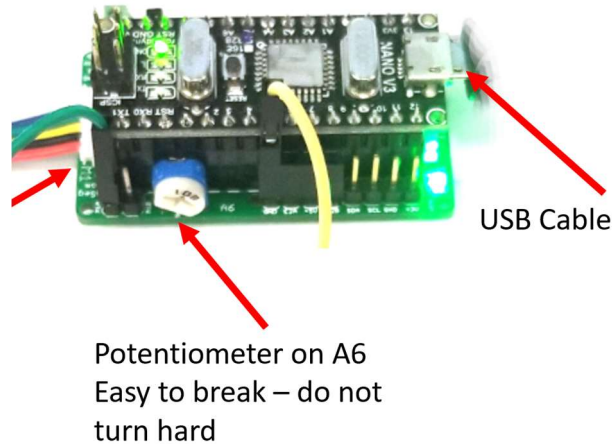
This potentiometer is fragile – do not try to turn it hard past its limit – look at the scope to view if it is approaching the limit: 0-4096

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

- **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).
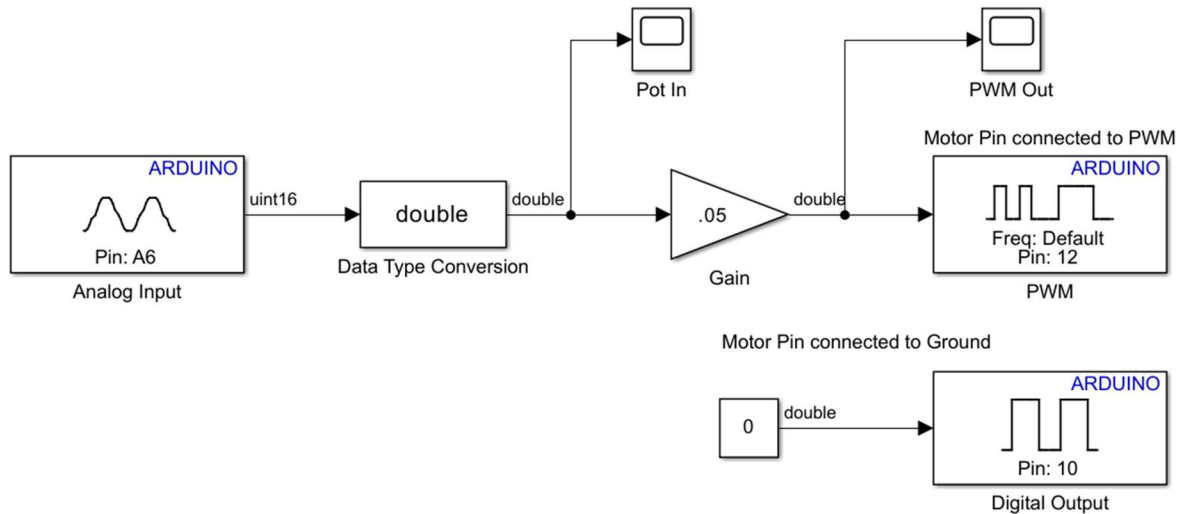
Observe the scopes while changing the position of the potentiometer wiper. Click the 'Autoscale' button to see the full range of the graphs.

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

PWM Output

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')

Pot In PWM Out

Motor Pin connected to PWM

ARDUINO

uint16 → double → .05

Pin: A6
Analog Input

double
Data Type Conversion

Gain

double

Freq: Default
Pin: 12
PWM

Motor Pin connected to Ground

0 double

Pin: 10
Digital Output

- **Configuration Parameters**: The "Fixed-step size" should be .03
- **Analog Input**: Set the Pin number to 6 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). You can find the pin number for your potentiometer written on the shield near the potentiometer.
- **PWM:** Set the PWM Pin to 12. This will be used to control the motor voltage (speed) by PWM
- **Digital Output:** Set the pin number to 10. This will ground the other motor wire.
- **Gain:** Start with the Gain at 0.05 and see the response as you gently turn the potentiometer. Calculate or determine what the gain needs to be so that the PWM gets to full value (the motor spins at full speed) when the potentiometer is in the max position. Set the gain to the correct value to so the analog input's full range will utilize the full PWM output
    - Hint: what is the maximum digital value for the PWM block and the maximum value for the ADC?
- **Data Type Conversion:** Go to Simulink-> Signal Attributes. Double click this block and set output data type to double. This is done so that floating point math is performed to avoid truncation and division errors associated with fixed point numbers.

Question: What is the correct value for the gain such that the full range of the potentiometer will utilize the full range of the PWM?

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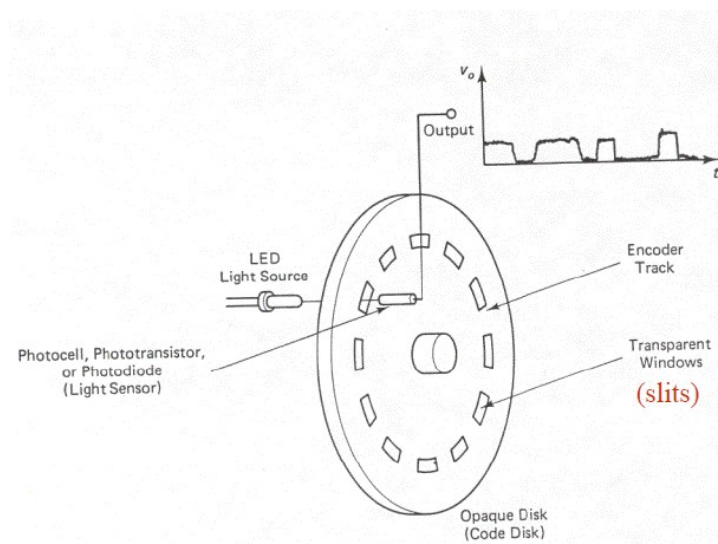
- The NXT encoder has a raw resolution of 12 counts. The figure on the left below show

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:

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

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 (5, 30, 50 or 100 etc. depending on your motor)
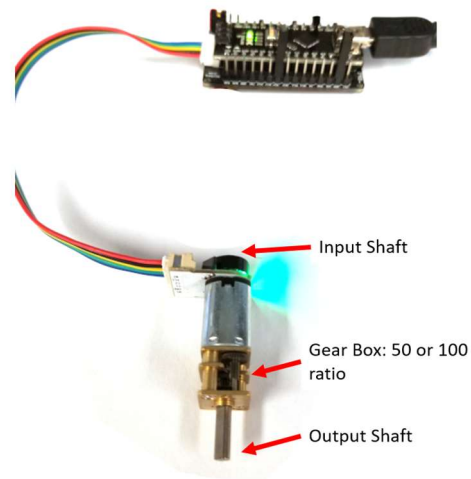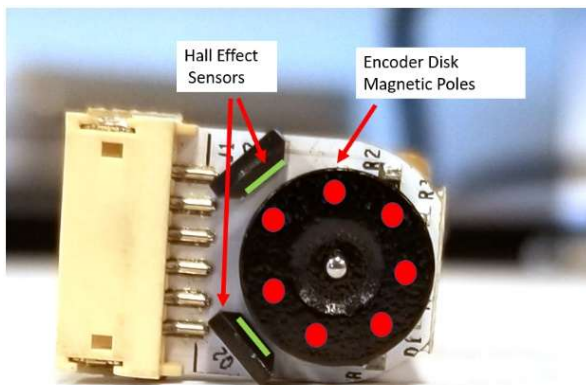
Simulink Model

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.

soEncoder   y

Pins 2 3
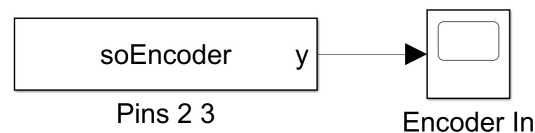
Encoder In

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?