Lab 1: Blinking the LED

Part 1: Simulink and Arduino Installation

Objective:

For Part 1, you will check that you have installed the Arduino software and Simulink properly on your computer.

Simulink Arduino Library Installation:

- Open MATLAB 2013a and type "targetinstaller" in the MATLAB command window.
- The target installer window will appear. Select "Use internet". Afterwards, just select "Arduino" for the target installer.
- When prompted, log in using your MathWorks credentials.
- Follow the onscreen prompts to install the library.
- Once the installation process is finished, you can exit out by clicking on "Finish".

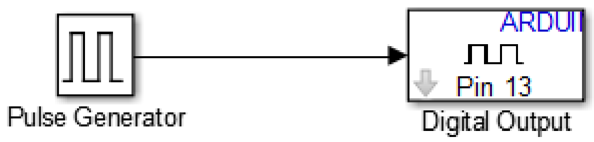
Arduino Mega 2560 Drivers' Setup:

- Connect your Arduino Mega board to the computer using the USB cable. Your computer will attempt to search for the necessary drivers online.
  - If your computer cannot find the right drivers, you'll have to do the following:
    - Go to Device Manager and look for your Arduino Mega board (Arduino Mega 2560). Your Arduino board will be under Ports (COM & LPT). Otherwise, it may be listed as "Unidentified Device" under 'Other Devices'.
    - Right click on Arduino Mega (or "Unidentified Device") and select "Update Driver and Software"
    - Select "Browse my computer for driver software"
    - Set the location to C:\MATLAB\SupportPackages\R2013a\arduino-1.0\drivers
    - You will receive a prompt letting you know Windows cannot verify the publisher of the driver, select "Install this driver software anyway."
    - Windows will then install the necessary drivers for the Arduino Mega from this folder. A message will appear when the driver software has installed.

The Arduino drivers can also be installed from the Arduino website (www.arduino.cc).

Simulink Setup:

In order to ensure that your computer can properly communicate with the Arduino board, please build and run the following Simulink diagram using the steps below:



1) Open MATLAB 2013a then open a new Simulink model.
2) Find and assemble the required blocks.
   - Pulse Generator is under View->Library Browser-> Simulink->Sources
   - Digital Output is under View->Library Browser-> Simulink Support Package for Arduino Hardware
3) Change the output pin to 13, which is connected to the onboard LED. This is done by double clicking on the Digital Output block.
4) Double click on the Pulse Generator to change the amplitude to 1, the Period to 10, and the pulse width to 50%.
5) Right click on the background and select 'Model Configuration Parameters' go to 'Solver' on the right hand menu and type 'inf' into the 'Stop time' spot so that the model will keep running. Make sure the solver options are set to "Fixed-step" and "discrete (no continuous states)". The "Fixed-step size" should be 0.05. This is how fast in seconds the control loop will execute. In this case every 50 milliseconds it will execute the Simulink code.
6) Under the tools menu go to 'Run on Target Hardware' and click 'Prepare to Run'. In the window that pops up you will need to select Arduino Mega 2560 in the drop down for Target Hardware.
7) Go to 'Tools-Run on Target Hardware-Run'.

At this point the on board LED connected to pin 13 should start blinking on and off. If not, check the debugging section below.

Debugging

- If an error occurs indicating that there is no driver, follow the directions on the screen.
- If MATLAB cannot find the board find COM port number from the device manager and in 'Configuration Parameters' set the port manually to the correct number.
- Some errors appear on the MATLAB workspace screen - if something is not working, check here.

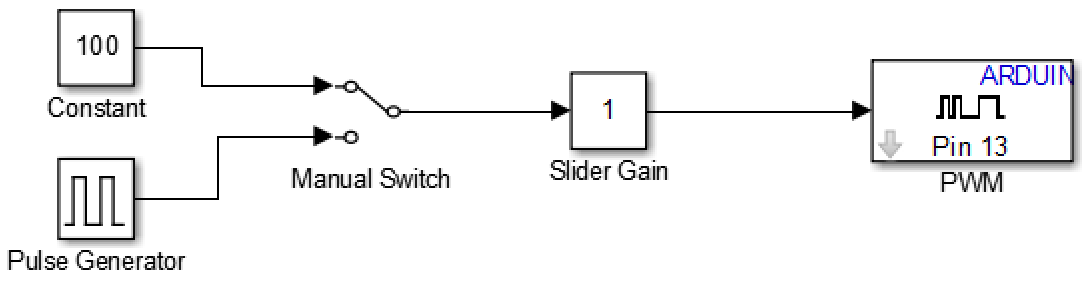
Part 2: External Mode, PWM, Manual Switches and Slider Gain

Objective

For Part 2, improve the diagram from part 1 to give the user the ability to change the parameters involved in blinking the LED.

Procedure

Modify your Simulink diagram to the following



- Pin 13 can also be set as a PWM instead of just an on/off digital output – replace the Digital Output block with the PWM block
- The "Slider Gain" can be found in View->Library Browser-> Simulink-> Math Operations
- The "Manual Switch" can be found in View->Library Browser-> Simulink ->Signal Routing
- The "Constant" can be found in View->Library Browser-> Simulink -> "Commonly Used Blocks"

Once the diagram is finished right click on the background of the diagram and click "Configuration Parameters" under the "Run on Target Hardware" tab make sure to enable external mode to allow you to change the values on the Slider Gain and the Manual Switch while the Program is running.

- When you use external mode you can change parameters while the system is running however this has a large impact on the performance.
  - Due to the communication bandwidth, the fastest you can run in this mode and still meet your control loop time is approximately 30 milliseconds
- The code can be run much faster (in the orders of 1 or 2 milliseconds) if external mode is not used.

Run the code and experiment with the setup. Observe the effects on the Arduino LED when changing the value of the Slider Gain, Manual Switch, and the Pulse Generator.

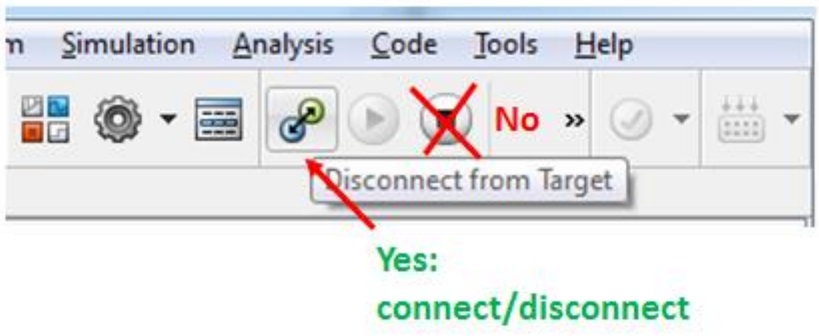
- While running the simulation, you can change the position of the switch by double clicking on the switch
- You can change the gain of the Slider Gain while running the simulation by double clicking it.
- Adjusting the Pulse Generator, experiment with all three parameters. Which increases brightness? What is the approximate range over which you can observe a difference? [Note: These parameters will be addressed in a later lab on PWM]

Stopping the Simulation: Do NOT use the stop button

When you are running the simulation in external mode you are connected to the device and are sending data and receiving data. When this is occurring do NOT use the stop button to stop the simulation:

- Always use the “Connect to Target” button to connect to the board. If you use the “Run on Target Hardware” menu then it will automatically download your code, connect to the board and run your code.

- Use “Disconnect from Target” if to make changes to the code or to unplug the USB cable. Do not unplug the USB cable before disconnecting from the target. Doing so will leave the serial port open on the computer and you will not be able to connect to the device without restarting Matlab or logging off your machine (to ensure the serial port gets closed).



Debugging – cannot connect to device or cannot download code

- If you cannot connect to the device or there is an error when trying to download the code usually this means external mode was enabled and the simulation was stopped with the stop button or the USB cable was unplugged while being connected to the device. In this case you have a couple things to try that may successfully close the serial port
    - Reset the device with the reset button or disconnect/connect the serial cable (this sometimes works)
    - Close and restart Matlab (this usually works)
    - Log off your machine, log back in, then restart Matlab (this almost always works)
- Make sure the COM port is correct in the “Model Configuration Parameters”

Checkpoint:

To complete Lab 1 you will need to show the ability to control the blink of the LED in one program with both the ‘manual switch’ and the ‘slider gain’.

Lab 2 – Reading data from a Gyroscope/Accelerometer with I2C.

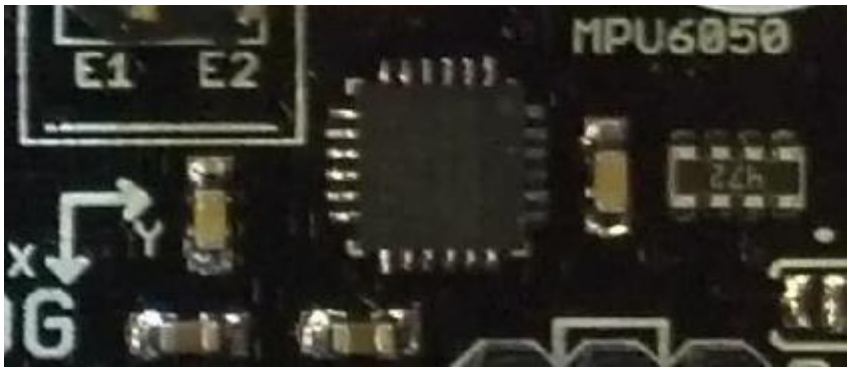
Part 1: Reading the Gyro

Objectives:

- Obtain data from a gyroscope with an I2C interface using a SFunction Driver
- Experimentally determine the conversion constant used to determine angle

Background Information:

This lab will utilize a single axis of the MPU6050 – an integrated circuit which contains a 3 axis gyroscope and a 3 axis accelerometer.



It has several noteworthy features:

- Configurable measurement ranges
    - Gyro angular rates: 250, ±500, ±1000, and ±2000°/sec
    - Accelerometer ranges: $\pm 2g$, $\pm 4g$, $\pm 8g$, and $\pm 16g$
- Configurable Low-pass and high pass filters

An Arduino library and sample code exists to make it easy to read from this device and set up the various options. This library is first tested in Arduino to ensure functionality, then ported to Simulink by "wrapping" the library into an SFunction block. This SFunction block is usually referred to as a "driver". In this case the "gyro driver" which is used to obtain the sensor data.

More information can be found by consulting the MPU6050 or by examining the actual library files included in this lab: `MPU6050.h` and `MPU6050.cpp`.

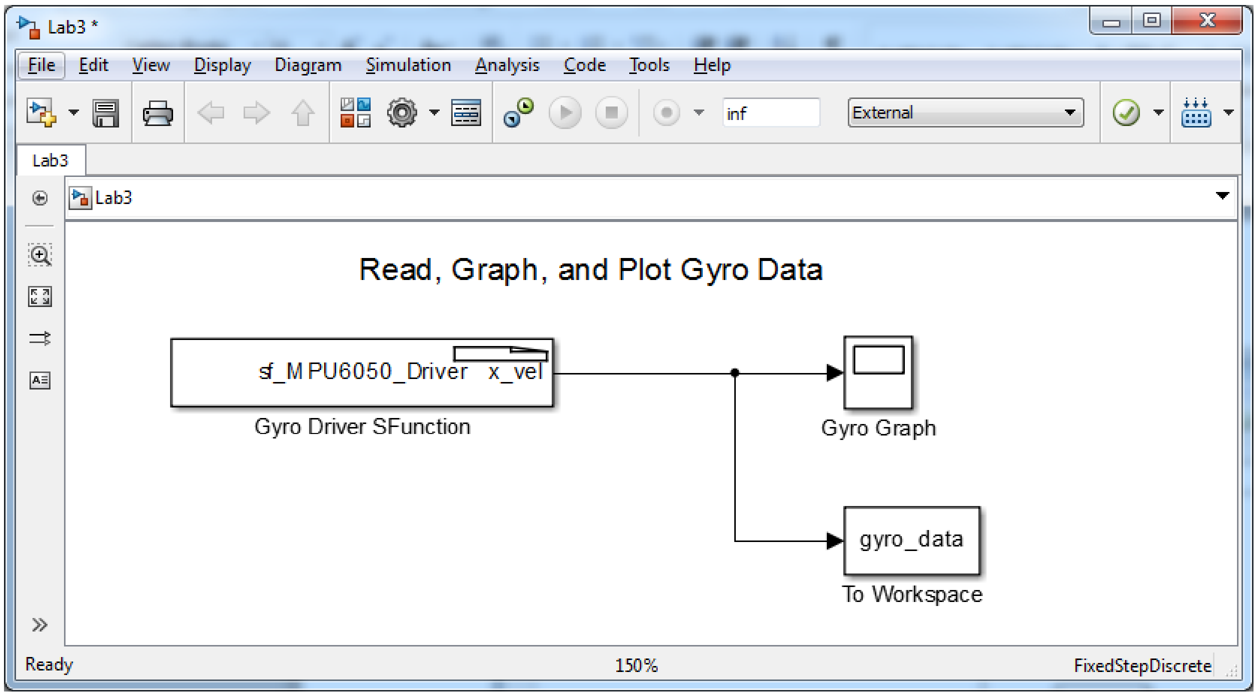
Question: What is the command needed to set the gyro low-pass filter to 42Hz? What is the delay associated with this? What is the command to set the gyro range to 250 degrees/second? (see `MPU6050.cpp`)

Gyroscopes:

A gyroscope measures angular rate or 'angular'. With a known initial condition the angular rate can be integrated to obtain angular position.

Simulink Model

- Copy the following files to your Matlab folder where you will build the Simulink diagram [Note: Leave these files in the default Matlab folder or your project folder. Do not put them in a sub-folder unless you add the path to Matlab with 'addpath' command.]
    - MPU6050_Blcck.slx
    - sf_MPU6050_Driver.mexw32
    - sf_MPU6050_Driver.mexw64
    - sf_MPU6050_Driver.tlc
    - sf_MPU6050_Driver_wrapper.cpp
    - I2Cdev.cpp (I2C Library files for MPU6050 communication)
    - I2Cdev.h
    - MPU6050.cpp (gyro library for configuration/communication with mpu6050)
    - MPU6050.h
    - Wire.cpp (Arduino wire library - support for I2C communication)
    - Wire.h
    - twi.c (Arduino wire library - support for I2C communication)
    - twi.h
- Build the following Simulink diagram.



Notice the descriptive text. Although it is graphical code, it is still code and should be properly commented. Get into the habit of properly commenting graphical code.

- **Parameters**: The step-size should be 30 milliseconds to run in external mode

Checkpoint 1:

Run the Simulink diagram and observe the results as you move the sensor (Arduino board) in your hand. Observe how the graph changes as you rotate about different axes.

Question: If the system is at rest – what is the value of the gyro reading? How much is it fluctuating? Use the data written to the workspace to compute the average gyro reading when the sensor is at rest for several seconds. [Hint: Make sure outputs to workspace such as 'gyro_data' are set to 'array'. Do this from now on in labs, unless mentioned otherwise]

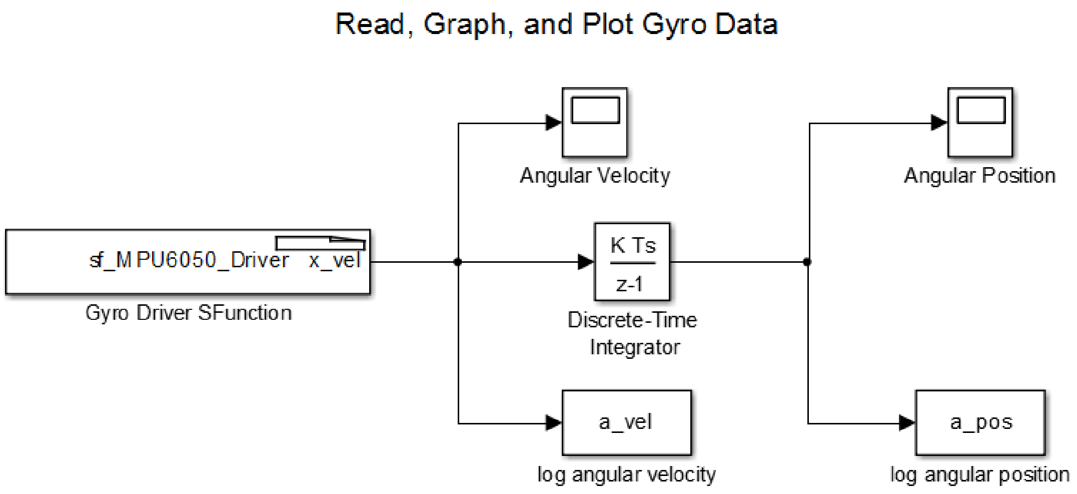
Note: The units are still unknown.

Part 2: Computing Angle from a Gyro

The gyro provides angular velocity – the change in angular position over time. To obtain the angle the angular velocity is integrated.

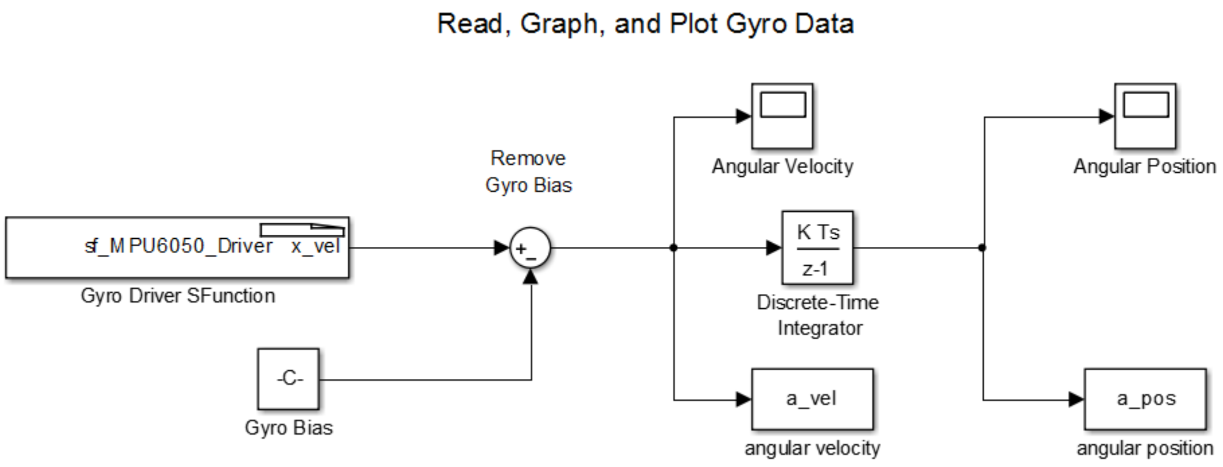
Simulink Model

Build and run the following Simulink diagram.



Question: What do you notice about the angular position and velocity when the sensor is sitting still? Does this make sense?

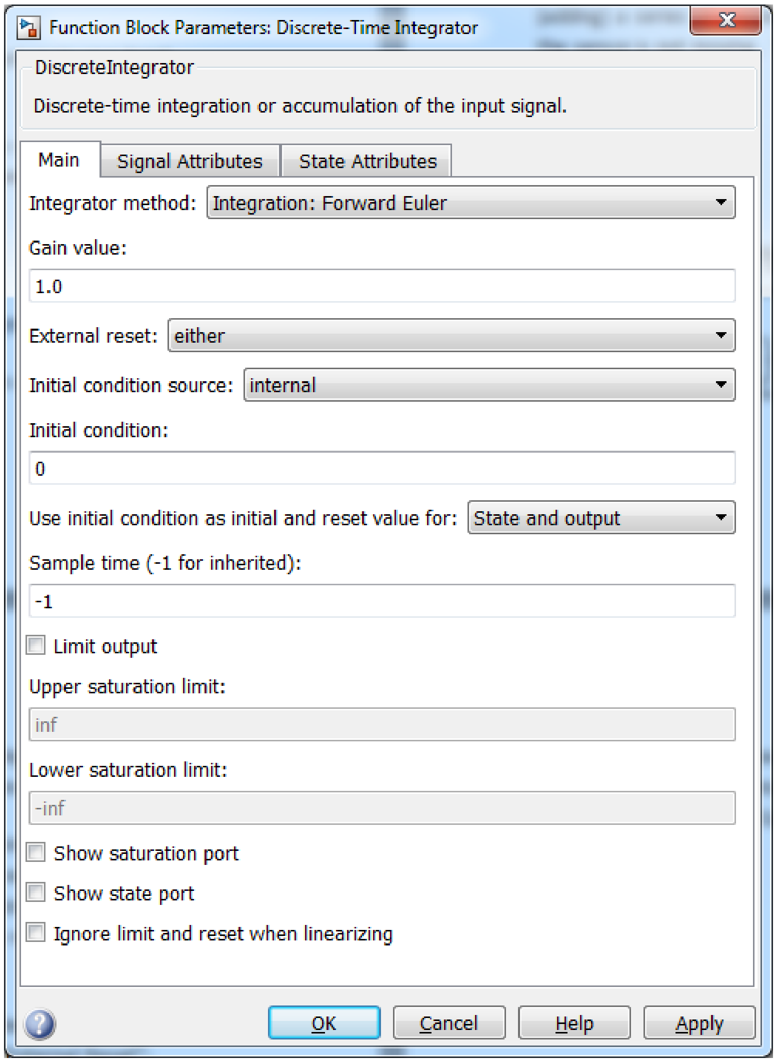
Since the sensor is not moving the values should be zero (no velocity). Any bias in velocity is integrating (adding) a series of nonzero numbers to the computed position even though the sensor is not moving. To eliminate this remove the bias before integrating.



Determine the bias which results in an angular velocity fluctuating about zero. Integrating this velocity will provide the angular position. The correct units are still unknown.

If there are any small errors in the bias, integrating these errors will eventually result in the position deviating from what is expected. In this case the only way to correct it is to start over again at a known point. This is done by resetting the integrator. When the code is first downloaded the integrator initial condition is zero. The only way to reset the integrator is to download the code again – this can be time consuming for initial testing.

To remedy this add an external reset to the integrator so it can be reset while running the simulation. Double click the integrator block and select “Either” for “External Reset”:


Function Block Parameters: Discrete-Time Integrator

DiscreteIntegrator

Discrete-time integration or accumulation of the input signal.

Main | Signal Attributes | State Attributes

Integrator method: Integration: Forward Euler

Gain value: 1.0

External reset: either

Initial condition source: internal

Initial condition: 0

Use initial condition as initial and reset value for: State and output

Sample time (-1 for inherited): -1

☐ Limit output

Upper saturation limit: inf

Lower saturation limit: -inf

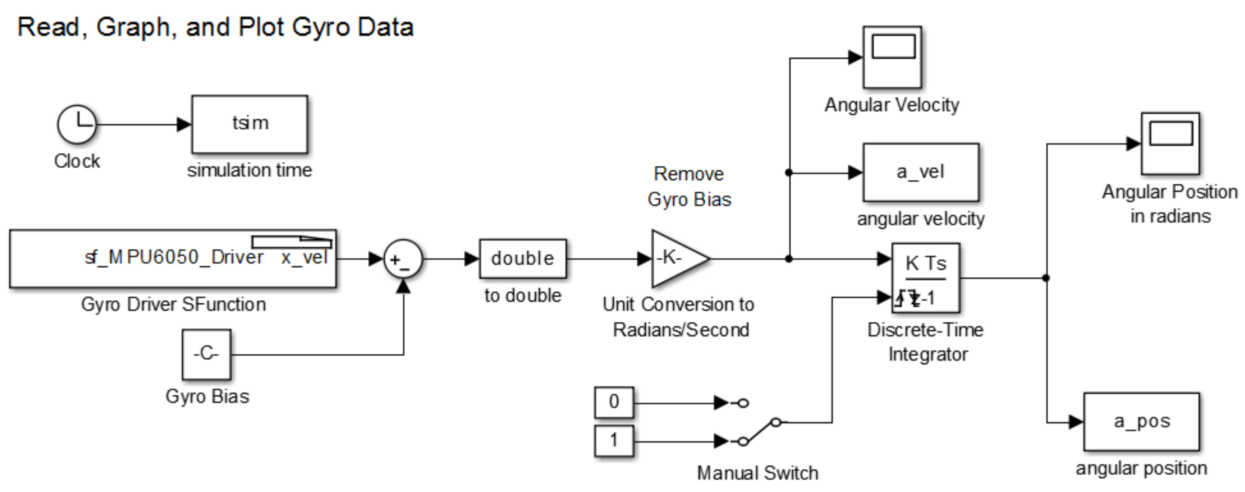
☐ Show saturation port

☐ Show state port

☐ Ignore limit and reset when linearizing

Add a switch in the simulation diagram so the integrator can be reset manually – this will reset the angular position to zero when the switch is changed.

A unit conversion must be performed to get the data into meaningful units. A gain block is added to accommodate this. Start with this value at initially at 1.



Determining the Unit Conversion:

- Build and run the Simulink diagram.
- Modify the bias until the velocity is fluctuating around zero.
- Toggle the switch to reset the angular position.
- Rotate the sensor about the x-axis 90 degrees
- Use the resulting angular position indicated to determine the unit conversion factor
- Test your results

Question: What is the proper unit conversion factor to obtain the results in radians/sec and radians? Create a plot showing both the angular position and angular velocity versus time as the sensor moves from 0 to 90 degrees and back to 0. The units for this plot should be radians, radians/second and seconds. Clearly label this graph.

Checkpoint 2:

Create a demonstration where the LED 13 will change from off at zero degrees to fully bright at 180 degrees. Be prepared to demonstrate this. Include a copy of your Simulink diagram with the rest of the lab questions.

Lab 4 – Reading from analog and encoder sensors. Outputting to PWM.

Part 1: Reading an Analog Signal

Objective:

For Part 1, the brightness of the LED will be controlled using pulse width modulation (PWM), which will be adjusted based on the position of the potentiometer's wiper.

Background Information:

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:

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

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

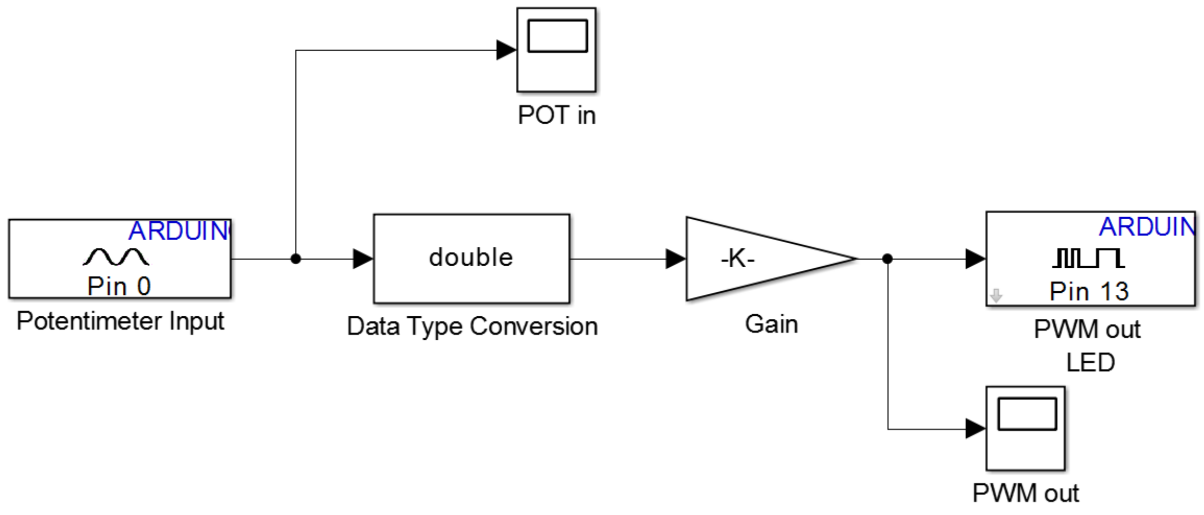
Question: If the value of the ADC on A0 of the Arduino is 512, what is the input voltage on A0?

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. In this lab the LED on pin 13 will be driven by a PWM. 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.

Simulink Model

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. (the fastest recommended size for external mode)
- **Analog Input**: Set the Pin number to 0 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)
- **PWM:** Set the Pin to 13 (the onboard LED).
- **Gain:** 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?

Make sure you set the simulation with the following options at the Model Configuration menu.

Checkpoint 1:

Observe the scopes while changing the position of the potentiometer wiper. Click the 'Autoscale' button ⛶ to see the full range of the graphs. At what potentiometer wiper position is the LED brightest? At what position of the potentiometer wiper is the LED completely dim?

Part 2: Reading Encoders

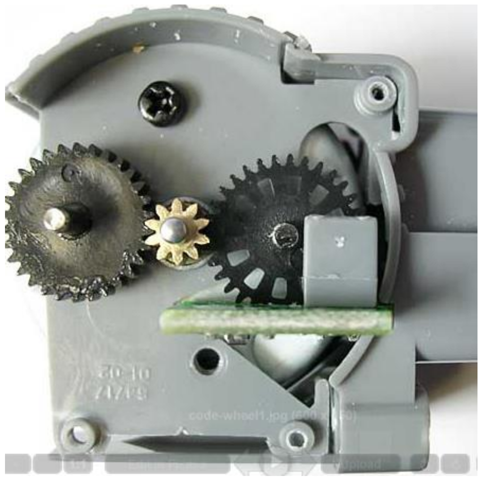
Objective:

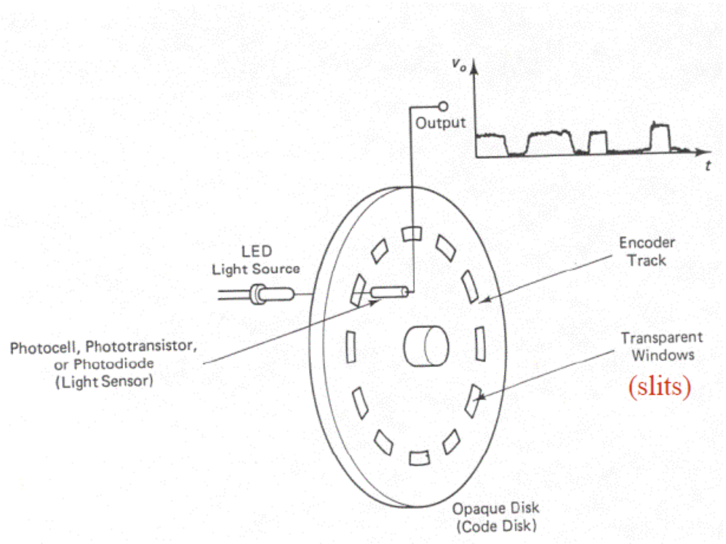
For Part 2, an encoder will be used to determine the position of the output motor shaft. The brightness of the LED will be controlled based on the position.

Background Information:

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 NXT's internal encoder will be used to track angular position.

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





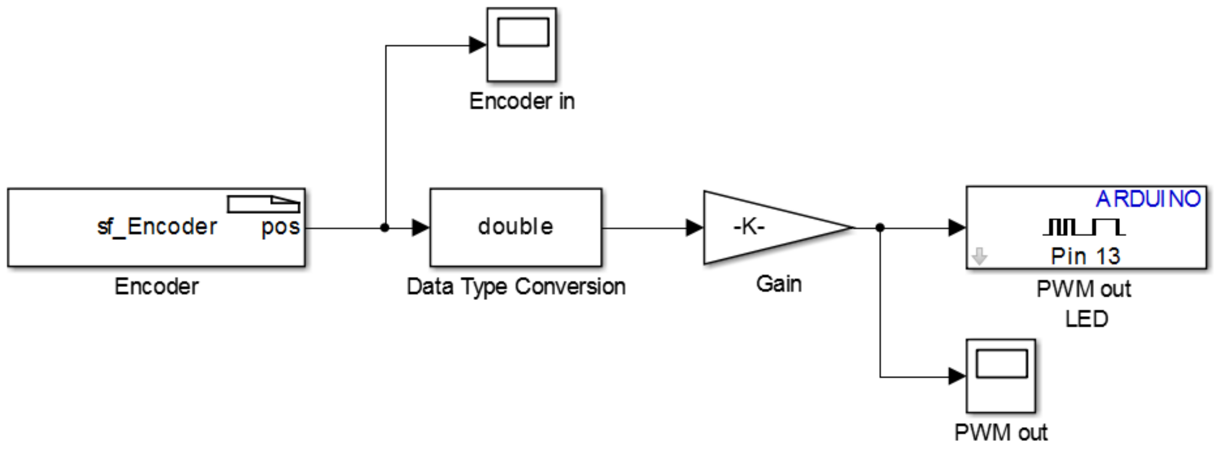
- If the output motor shaft moves 1 revolution the encoder wheel turns 15 revolutions – this means for one revolution of the motor output shaft there will be 15*12 = 180 pulses from the encoder.
- The encoder device provides two channels A & B. These channels are the encoder pulses with one channel shifted 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:

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

Simulink Model

- Copy the following files to your Matlab folder where you will build the Simulink diagram (all the these files and
  - sf_Encoder.mexw64
  - sf_Encoder.tlc
  - sf_Encoder_wrapper.c
  - Encoder_Block.slx

Build the following Simulink Model (the Encoder block is inside the Encoder_Block Simulink diagram)



Question: What value does the gain need to be so that 1 revolution of the encoder will obtain the maximum value of the PWM?

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 a position of 720. You can also add a 'Sinks -> Display' block to see exact values.

Lab 5 -DC Motor Step Response

Part 1: Motor Step Response

Objectives:

- Use the DRV8833 motor driver and obtain the motor step response
  - In External mode
  - Directly through the serial port
- Understand the logic needed to control the magnitude and direction of a motor
- Observe the different effects of different logic schemes to drive a motor

Background Information:

A motor driver chip converts a lower power signal (from the microcontroller: 5v, 40mA) to a high power signal (9v, 1.5A) to drive the motor. It can be thought of as a switch or relay to the driver supply voltage (9v or 5v in this case). Since the supply voltage is fixed this would result in a fixed motor speed. To regulate the speed a PWM signal is used to quickly switch the output on and off so that the average output voltage can be controlled.

The term motor "driver" is also commonly called "amplifier" or "chopper". The DRV8833 driver is capable of providing 2.7-10.8v at 1.5A RMS on each channel.

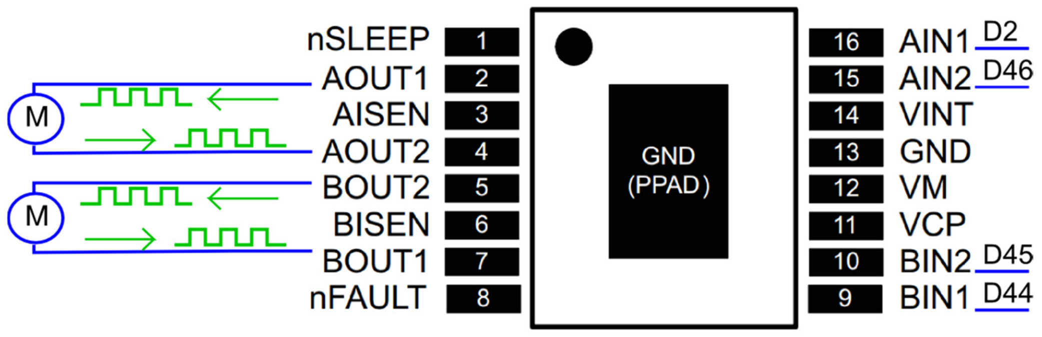


Figure 1: DRV8833 pinout/wiring diagram

". The SN754410 driver is capable of providing 3.5-36v at 1A on each channel.

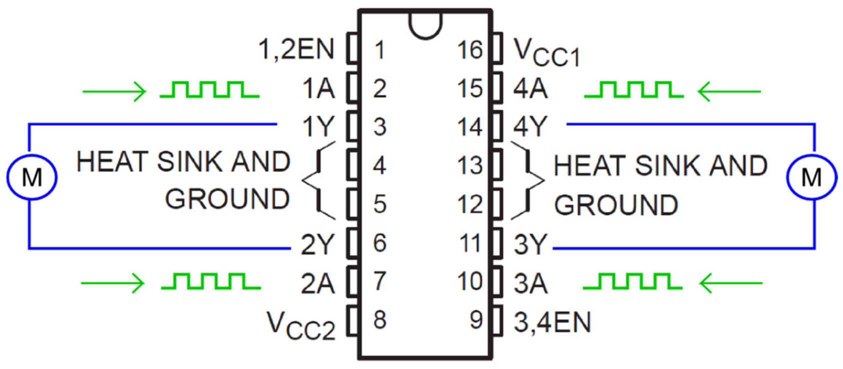
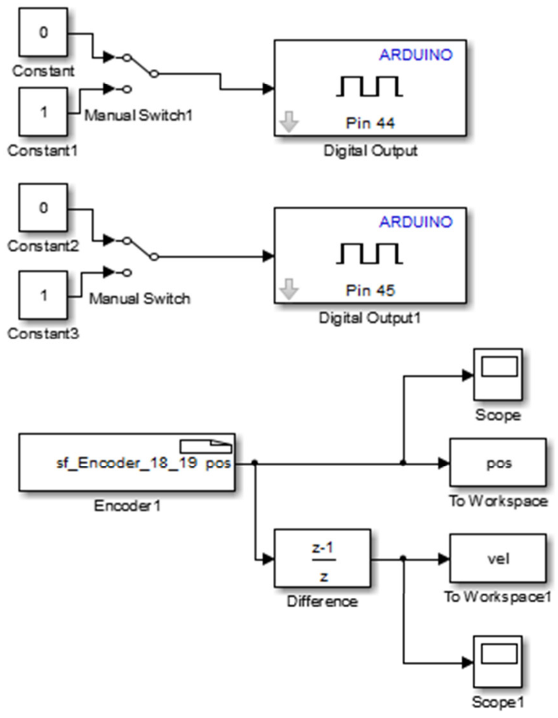


Figure 2: SN754410 pinout/wiring diagram

Simulink Model

Build and run the following Simulink diagram. Use the corresponding encoder/dual encoder block for your system, and the digital pins that do to your motor driver

- M1V4: motor on 4&5 or 9&11, encoder on 2&3 or 18&19
- M2V3: motor on 2&5 or 6&8, encoder on 18&19, or 15&A8
- MinSegMega: motor on 2&46 or 44&45, encoder on 18&19 or INT6&int7

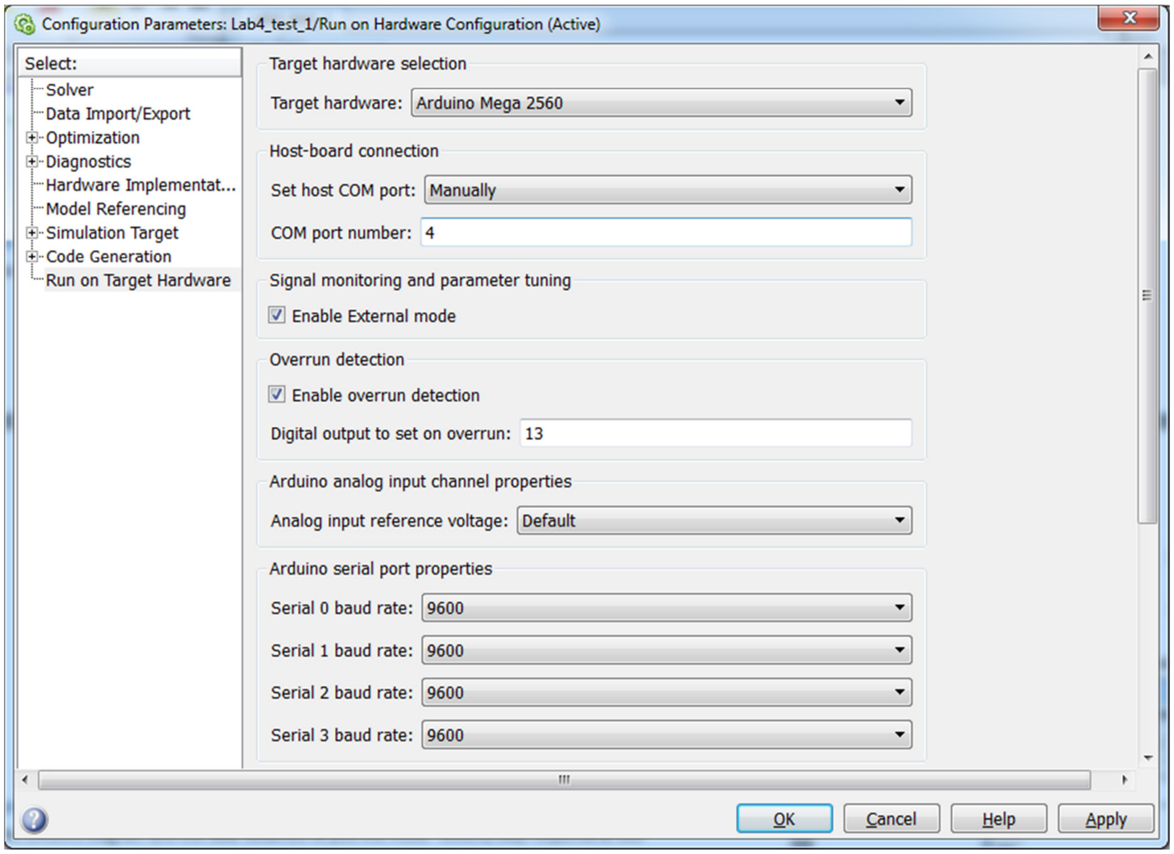


- Run the model in external mode to log the data

    - Sampling rate of .03 seconds
- Observe the response of the system
    - toggle the manual switches
    - toggle the On/Off switch on your board – this will determine the voltage source of the driver chip (VM):
        - OFF - USB voltage (~4.5 volts)
        - ON – Battery voltage (~9v fully charged)
- Stop/disconnect from the system, save the data then plot the results
    - The commented line below stores the variables ‘pos’ and ‘vel’ into the data file ext_dat.

```
%save ext_dat pos vel    % save the data
load ext_dat             % load the data
figure, plot(pos)
figure, plot(vel)
```

The velocity data obtained in external mode will be noisy. The primary reason seems to be that the time between samples is not actually fixed/constant. In the configuration parameters if you check the box “Enable overrun detection” then the digital pin you specify will be set high if the controller cannot execute your code in the sample time you specify. Check this box, select pin 13, and connect to the device. You will notice that the LED on pin 13 is always on. This indicates that in external mode the microprocessor has trouble meeting the sample time. This implies the time between each sample is not exactly 30 milliseconds. Any calculation assuming a fixed sample time, such as velocity, will then contain some error (noise) due to the sample time not being constant. Note that this noise is from the measurement system - not actual noise present in the signal or system.



Configuration Parameters: Lab4_test_1/Run on Hardware Configuration (Active)

Select:
- Solver
- Data Import/Export
- Optimization
- Diagnostics
- Hardware Implementat...
- Model Referencing
- Simulation Target
- Code Generation
- Run on Target Hardware

Target hardware selection

Target hardware: Arduino Mega 2560

Host-board connection

Set host COM port: Manually

COM port number: 4

Signal monitoring and parameter tuning

[✓] Enable External mode

Overrun detection

[✓] Enable overrun detection

Digital output to set on overrun: 13

Arduino analog input channel properties

Analog input reference voltage: Default

Arduino serial port properties

Serial 0 baud rate: 9600

Serial 1 baud rate: 9600

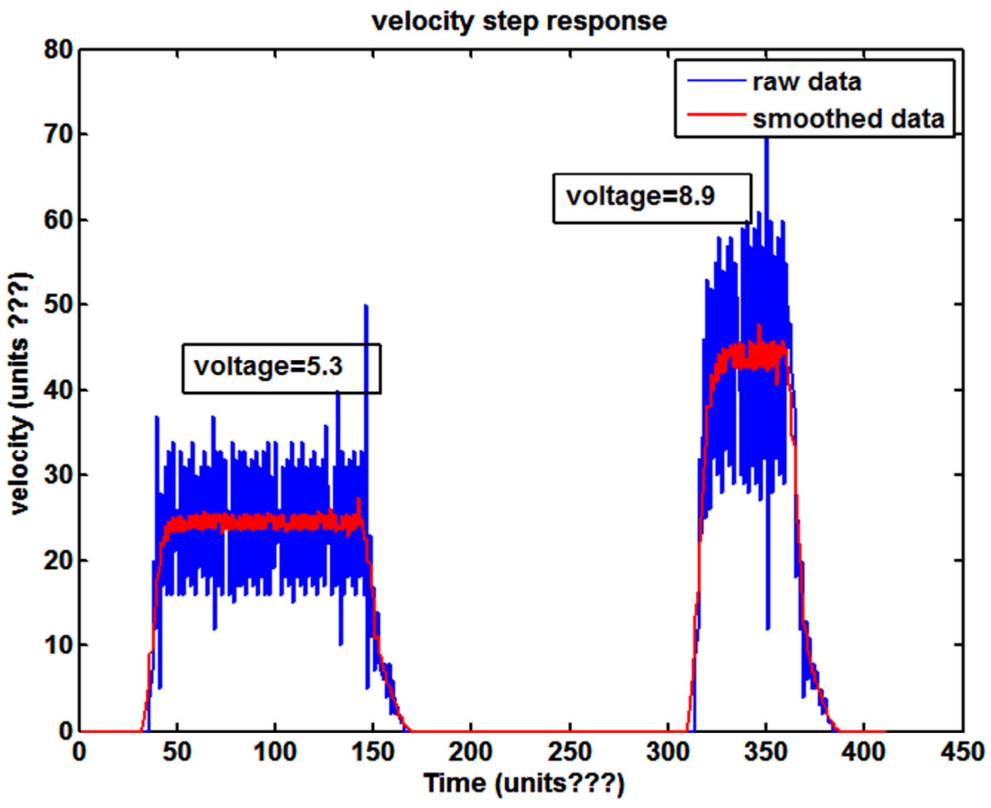
Serial 2 baud rate: 9600

Serial 3 baud rate: 9600

- The actual motor velocity will not be fluctuating like the "noisy" data indicates. To make the data more useful use the "`smooth`" command in Matlab. Type "`help smooth`" to obtain details on the smoothing method
  - `vel_smooth=smooth(double(Vel), 10)`
    - This function will use a moving average over 10 samples to smooth the velocity data
    - double – this converts the int16 values in Vel to doubles. If this is not done the function will complain

Questions:

- Provide a plot that contains the velocity step response. Provide the raw data and a smoothed response. Plot the velocity in RPM and time in seconds. An example plot (for two different voltage steps) is shown below for an unknown motor with unknown units:



Data for Parameter Identification (Optional)

The DC motor parameters can be determined from steady-state voltage and current measurements.

- From the pinout diagram for your system find the jumper that connects the driver to one of the motor terminals
    - Leave the jumper in so the motor is spinning at a constant velocity – use a multimeter to measure from the ground on the batter terminal block to the pins for the motor jumper. NOTE: the voltage will only be a meaningful value when the motor is spinning in one direction, in the other direction it will be closer to zero. If your voltage is not near the expected value reverse the direction of the motor. This is because the jumper is connect to only one of the motor leads.

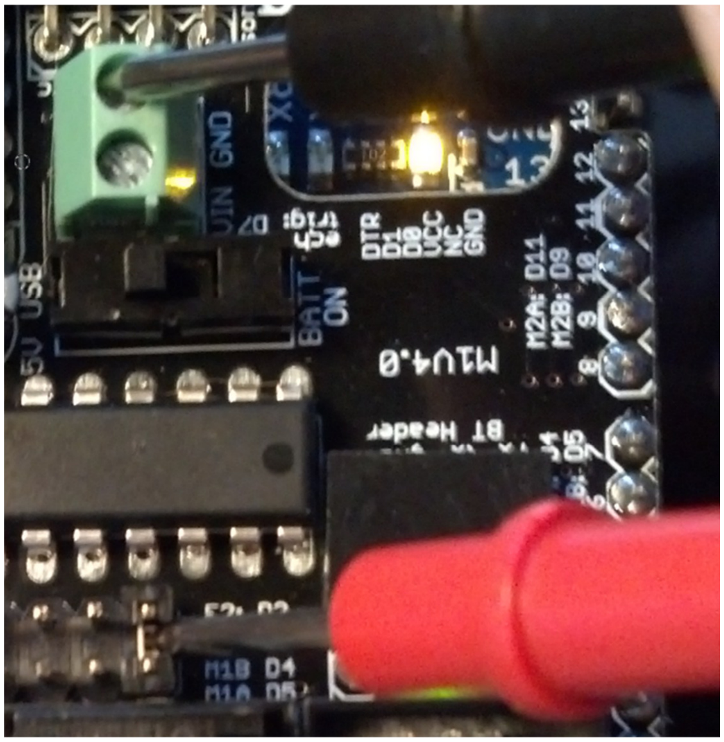


Figure 2: M1V4 Voltage measurement jumper

- Before you can measure the current determine the maximum current your multimeter can read. At steady-state most small DC motors will be less than 200ma.
- Remove the jumper and use a multimeter in current mode to complete the circuit and allow the motor to spin at a steady state speed and measure the current. Be careful not to short the multimeter probes.

- When the motor is running at a steady-state constant velocity record the steady state:
  - velocity average (with correct units)
  - voltage with a multimeter
  - current with a multimeter

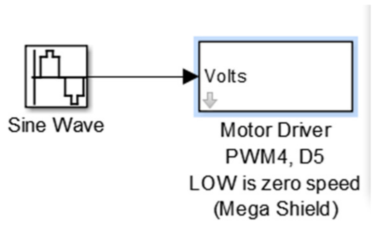
Questions:

- What is the maximum current your multimeter can read?
- When powered from USB what is the steady state speed, current, and voltage (include units)?

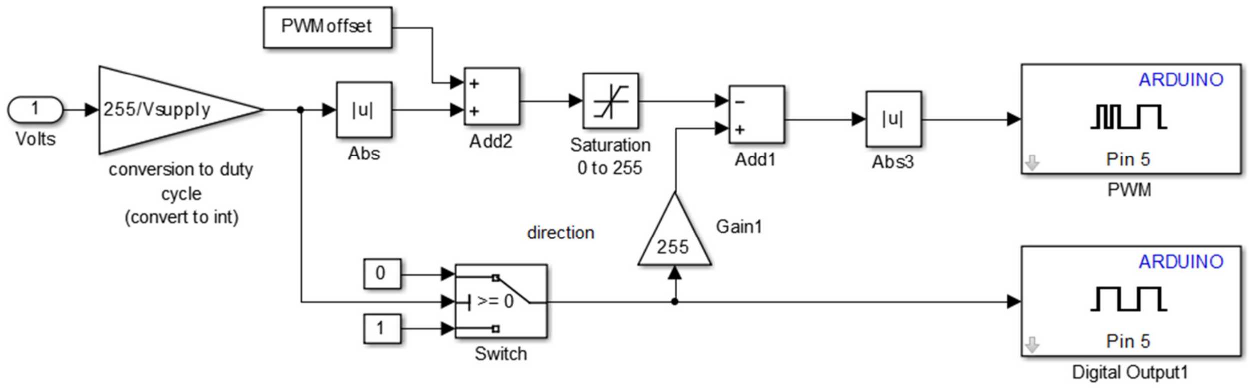
Part 2: Motor Logic: Direction and Magnitude

To regulate the speed a PWM signal is used to quickly switch the output on and off so that the average output voltage can be controlled. In addition the correct switching logic needs to be implemented so that the motor can change direction based on the sign of the input.

The subsystem block below is used to correctly output the correct magnitude and logic of the PWM if the only input is a scalar input voltage (which could be negative)

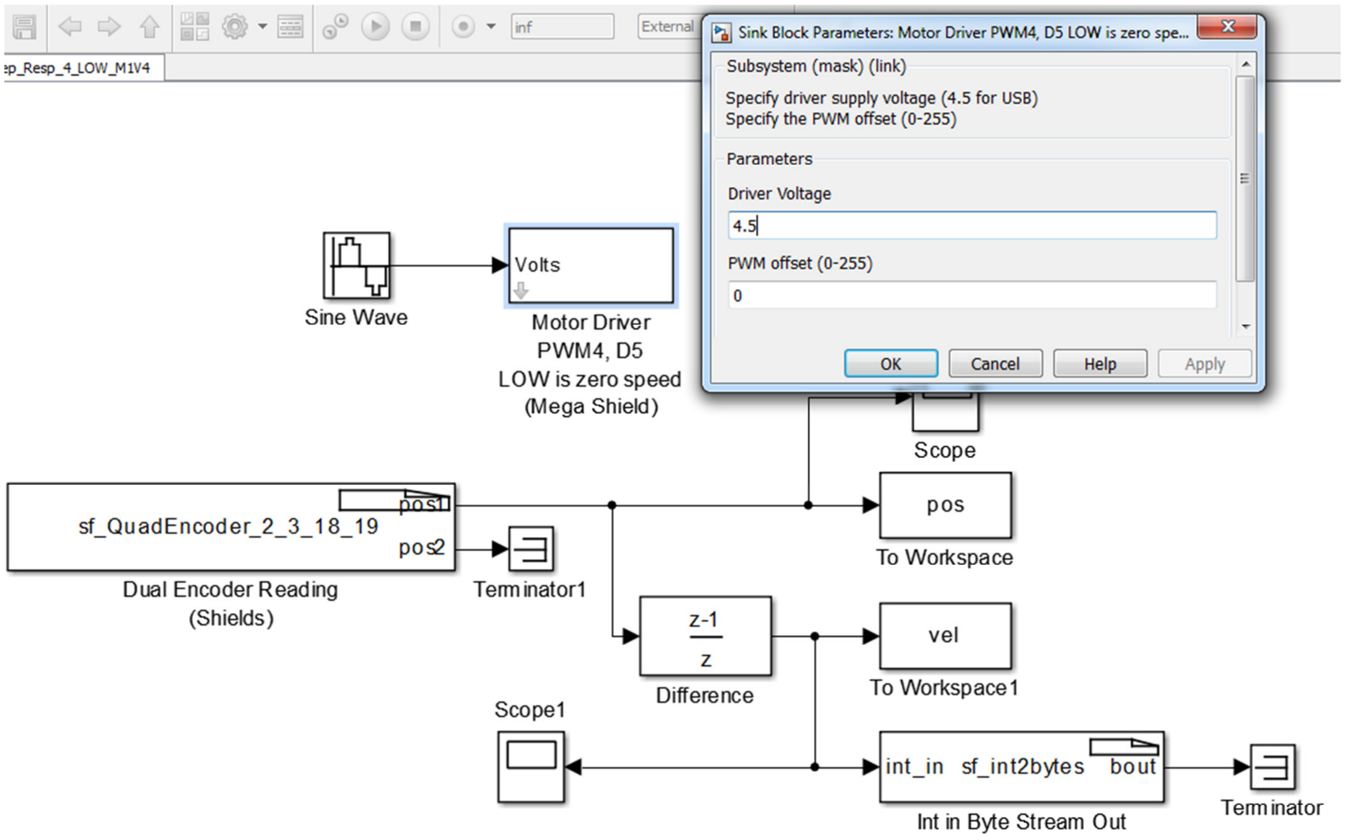


The contents of this subsystem are shown below:



- Use this subsystem block to generate a sinusoidal motor response:

cc


Sink Block Parameters: Motor Driver PWM4, D5 LOW is zero spe...

Subsystem (mask) (link)

Specify driver supply voltage (4.5 for USB)
Specify the PWM offset (0-255)

Parameters

Driver Voltage

4.5

PWM offset (0-255)

0

- Edit the Subsystem parameters to correctly specify the driver supply voltage (4.5 if power is from a USB) and a PWM offset – normally zero.

Questions:

- Provide a plot of the sinusoidal velocity in external mode.