RENSSELAER MECHATRONICS

Reading data from a Gyroscope/Accelerometer with I2C

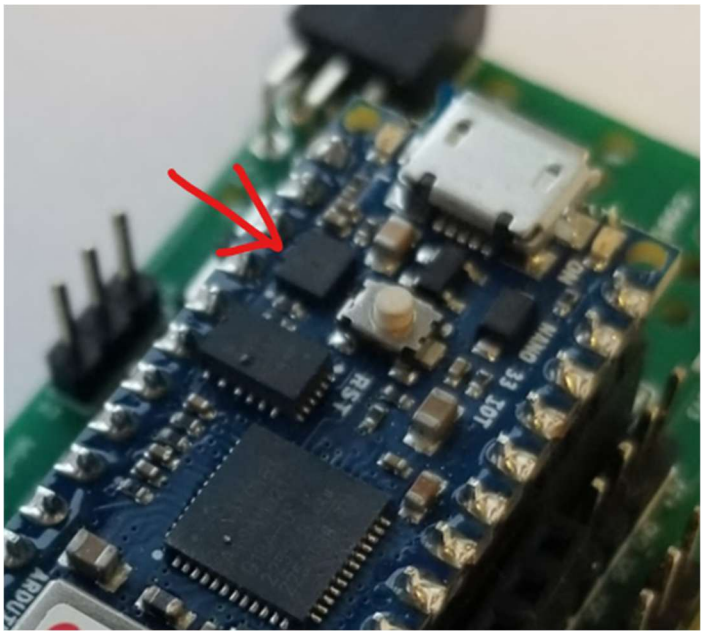
Part 1: Reading the Gyro

Objectives:

- Obtain data from a IC2 gyroscope
- Experimentally determine the conversion constant used to determine angle

Background Information:

This lab will utilize a single axis of the LSM6DS3 – 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:125, 250, ±500, ±1000, and ±2000 °/sec
  - Accelerometer ranges: $\pm 2g$, $\pm 4g$, $\pm 8g$, and $\pm 16g$

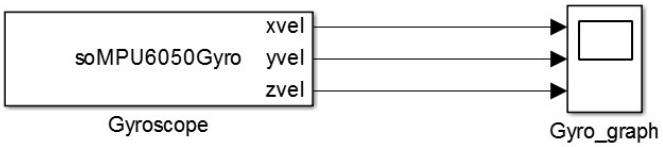
An Arduino library and sample code exists to make it easy to read from this device and set up the various options. This library is first tested in Arduino to ensure functionality, then ported to Simulink by "wrapping" the library into a system object block. This system object block is usually referred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

Gyroscopes:

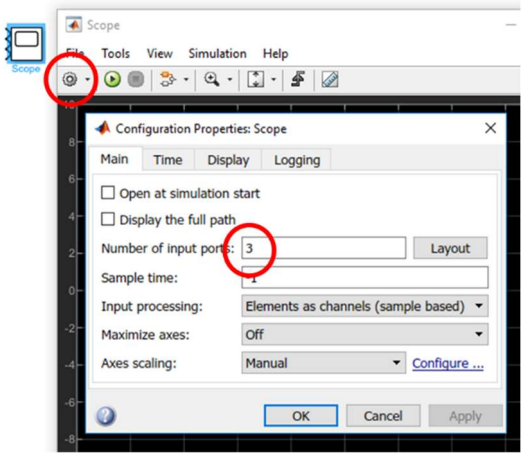
A gyroscope measures angular rate or 'angular' velocity. With a known initial condition the angular rate can be integrated to obtain angular position.

Simulink Model

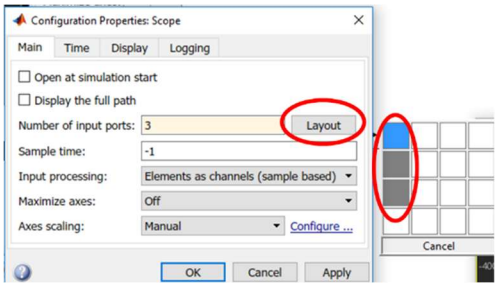
- Drag the Gyroscope from your demo file into your Simulink Model. Alternatively, it is easier to start with a working Simulink diagram that has all the settings to create this new diagram – open your demo file and delete all the blocks except the block with the gyroscope sensor.



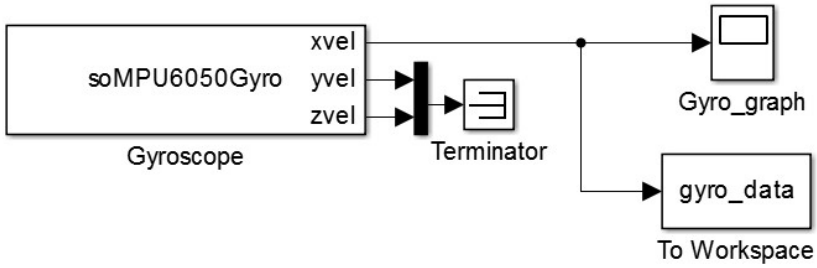
- Build and run the following Simulink diagram in external mode (using the green "play" button) with a step-size of 30ms to verify you can obtain the sensor data: (you can double-click the scope block to change the number axes to 3 by clicking on the small gear icon)



Then select "View", Layout, and highlight single rows to view each input channel on a separate graph.



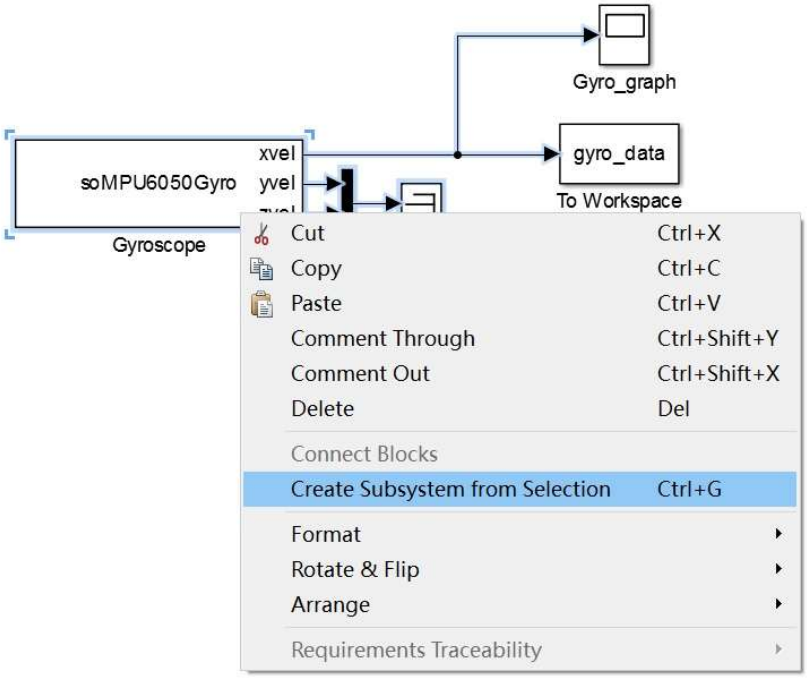
- Move the sensor around to determine each axis of the sensor. You should be able to verify the axis by looking at the documentation for the sensor.
- After verification, run the following Simulink diagram to store the data for plotting:

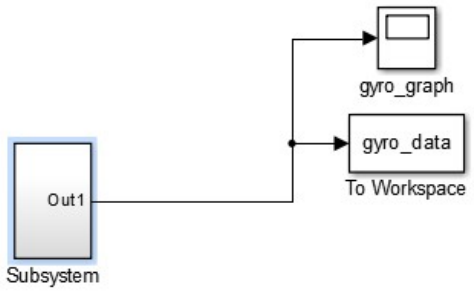


The Terminator is located under “Sinks”. Be sure the “To Workspace” block is set to log the data with a 2D array.

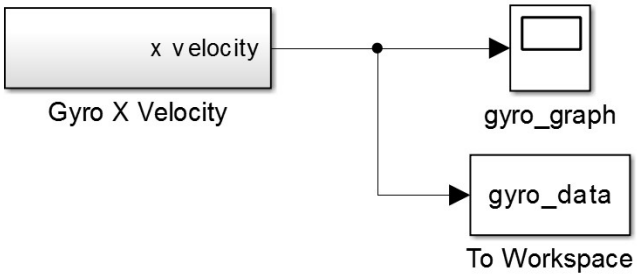
If the block names do not appear, you can right-click a block and select “Format”, “Show Block Name”, “On”.

Since we are only using one signal, to make the code more readable, use the mouse to select the driver block, the terminator blocks to create a subsystem:





Give the subsystem a descriptive name, then enter that subsystem and change the output port name to something useful to obtain:



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 1:

- 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]
- If you rotate the sensor very fast what happens to the sensor signal?

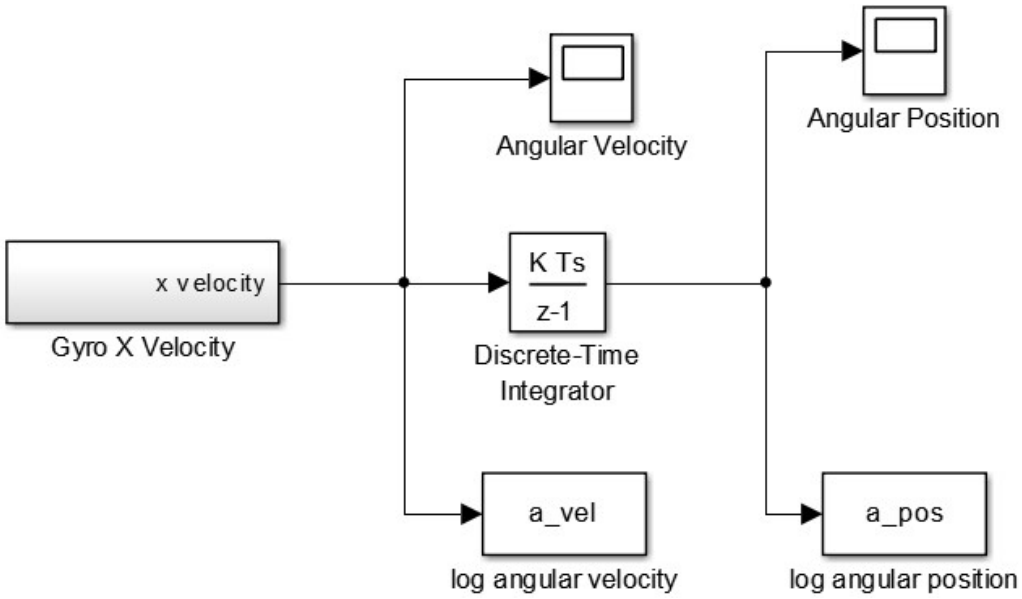
Note: The units are still unknown.

Part 2: Computing Angle from a Gyro

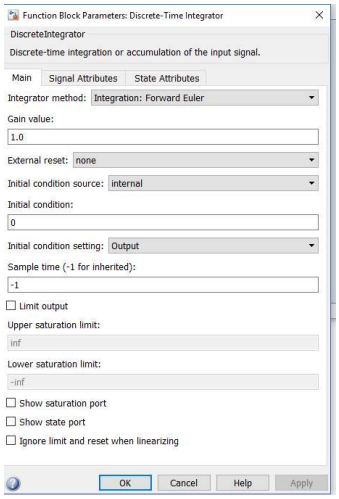
The gyro provides angular velocity – the change in angular position over time. To obtain the angle the angular velocity is integrated.

Simulink Model

Build and run the following Simulink diagram.



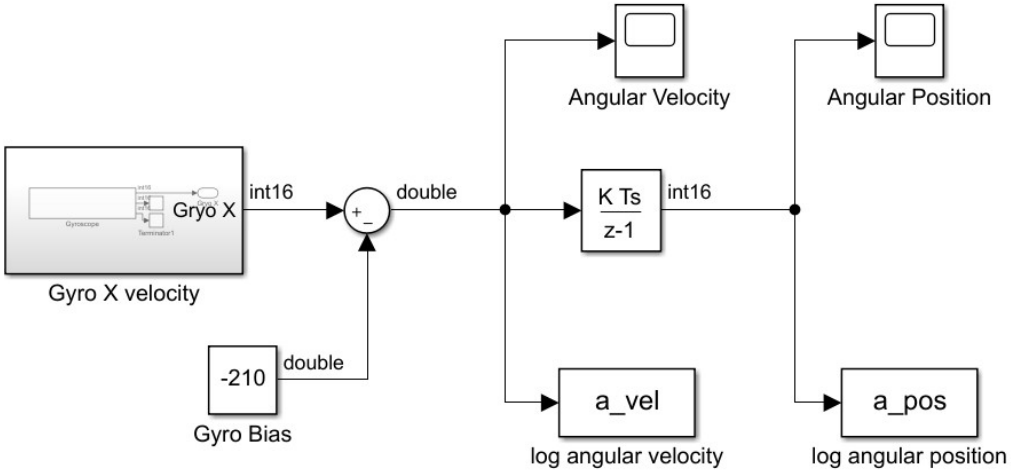
Change the sample time of the Discrete-Time Integrator to -1.



Question 2: 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.

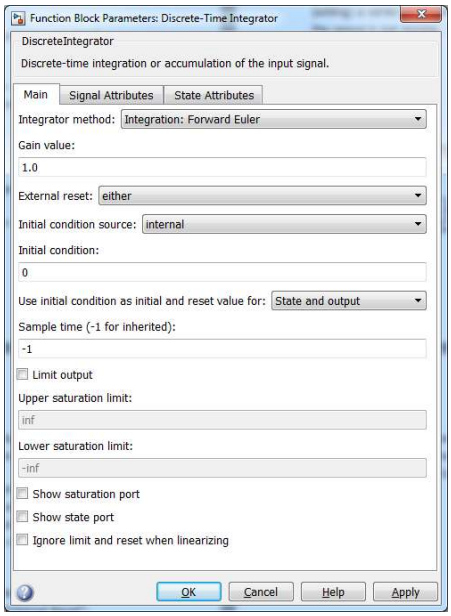
Next, to eliminate the position from changing due to a non-zero velocity when it is not moving, the bias is removed before integrating. Notice when you add the sum block and constant block it now chooses a “double” data type by default after the sum block.



Next, 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.

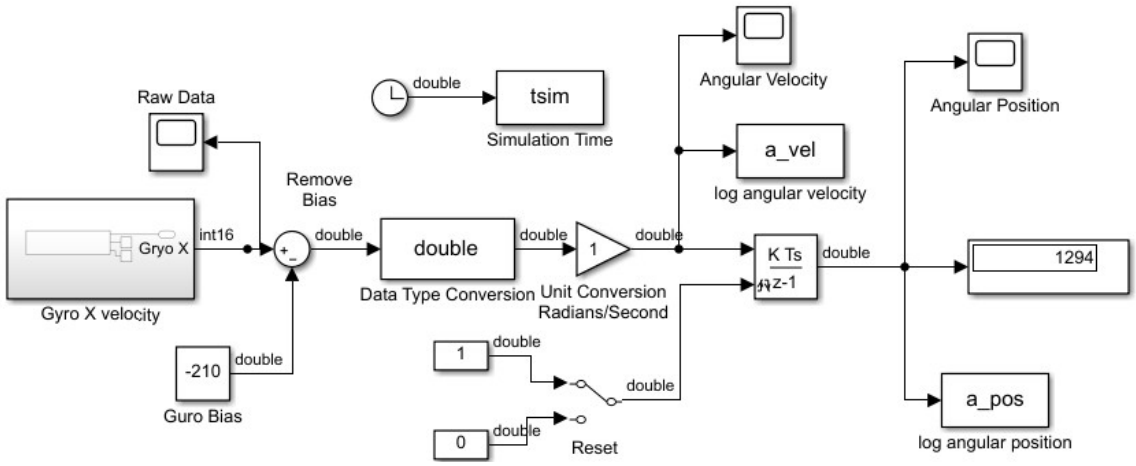
If there are any small errors in the bias, integrating these errors will eventually result in the position deviating from what is expected. In this case the only way to correct it is to start over again at a known point. This is done by resetting the integrator. When the code is first downloaded the 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”:



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.



Notice the data type is now being explicitly changed to double after the sum block to ensure the unit conversion will be performed with a data type of double. Depending on the memory and required speed all of these signal types could be chosen to optimize the coder performance. For example the constants 1 and 0 could be changed to a data type of uint8, etc.

It is also good coding practice to put the data type conversions in a separate block instead of doing in in a previous block as done previously with the integrator block. This makes the change in data type more visible.

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 3: 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. Include a screenshot of your final Simulink diagram.

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.