ME – 190 MECHATRONICS SYSTEM DESIGN

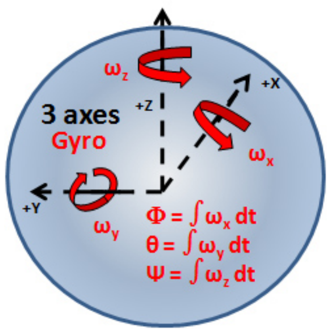
Lab 4[*]: Reading data from a Gyroscope/Accelerometer and determining Angle using Complimentary Filter

Objectives:

- Determine angle using a gyroscope
- Determine angle using an accelerometer
- Create a "Complimentary Filter" utilizing weighted input from both sensors

Background (Gyroscopes):

Gyroscopes utilize the principle of angular momentum to accurately measure rotation rate about an axis. Thus, multiple gyroscopes can be used to measure the angular rotational rate of an object in space. However, gyroscopes cannot inherently tell which way is "up" in the world, and techniques for computing position may result in "drift".


3 axes
Gyro
ω_z
+Z
+X
ω_x
+Y
ω_y
$\Phi = \int \omega_x \, dt$
$\theta = \int \omega_y \, dt$
$\Psi = \int \omega_z \, dt$

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

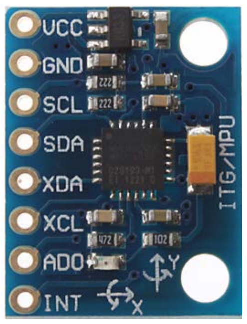


Figure 1: Generic MPU6050 breakout board

[*]Courtesy of Dr. Joshua Hurst at RPI/MinSeg

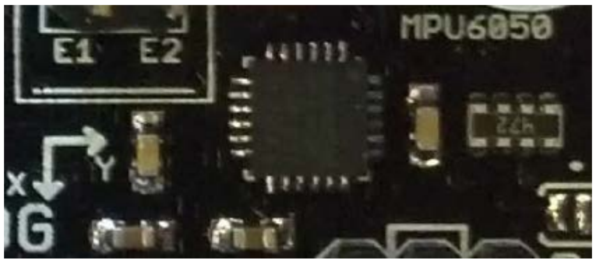


Figure 2: MPU6050 on MinSegMega

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 a 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`.

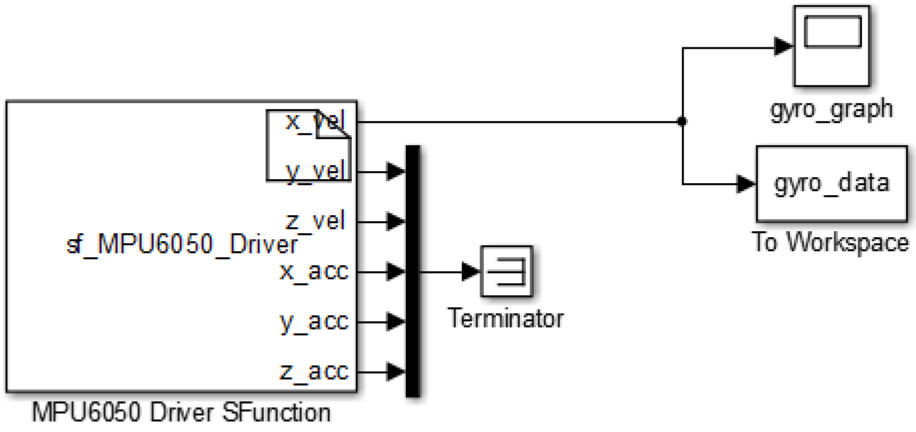
Question: What is the command needed to set the gyro low-pass filter to 42Hz? What is the delay associated with this? What is the command to set the gyro range to 250 degrees/second? (see`MPU6050.cpp`)

Gyroscopes:

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

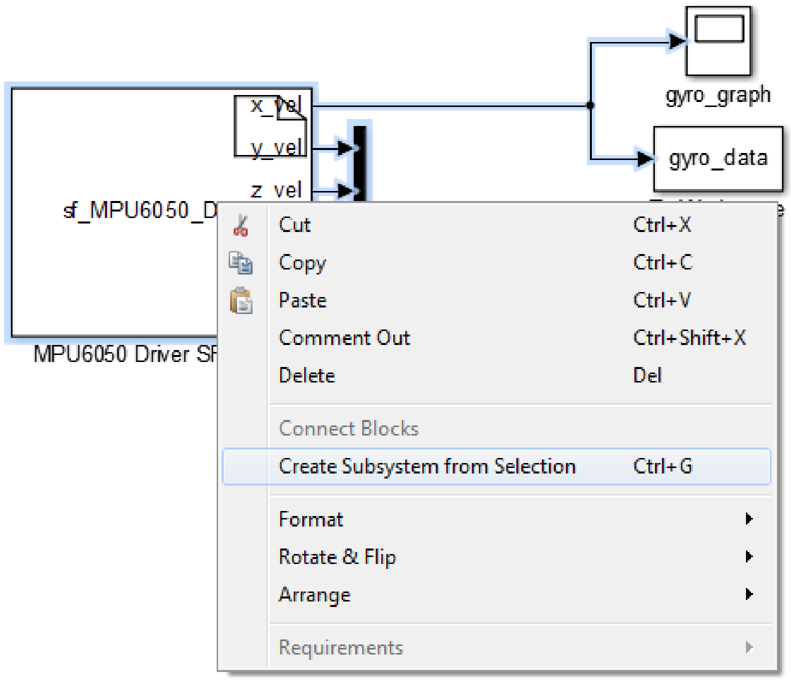
Simulink Model

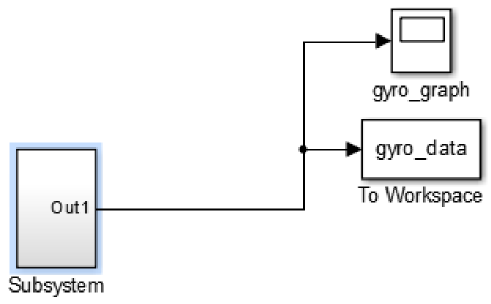
- Drag the MPU6050 block into your Simulink Model from RASPlib
- First run the following Simulink diagram in external mode (use a step-size of 30ms) to verify you can obtain the sensor data: (you can double-click the scope block to change the number of axes to 6 by clicking on the small gear icon)
- After verification build the following Simulink diagram.



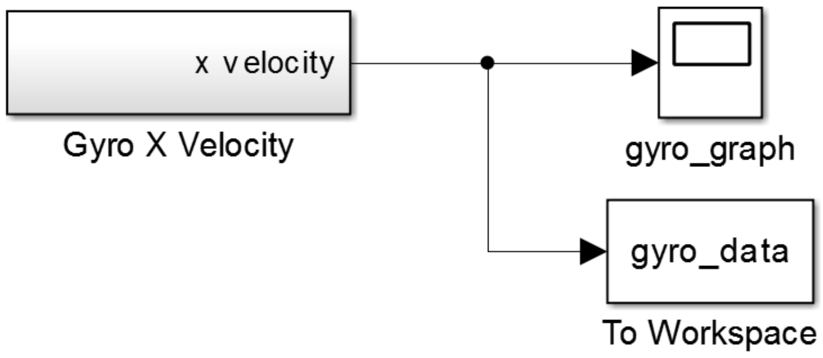
The mux block is found under ‘Signal Routing’, and the Terminator under “Sinks”

Since we are only using one signal, to make the code more readable, use the mouse to select the sf_MPU6050_Driver block, the mux block and the terminator block 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: 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]

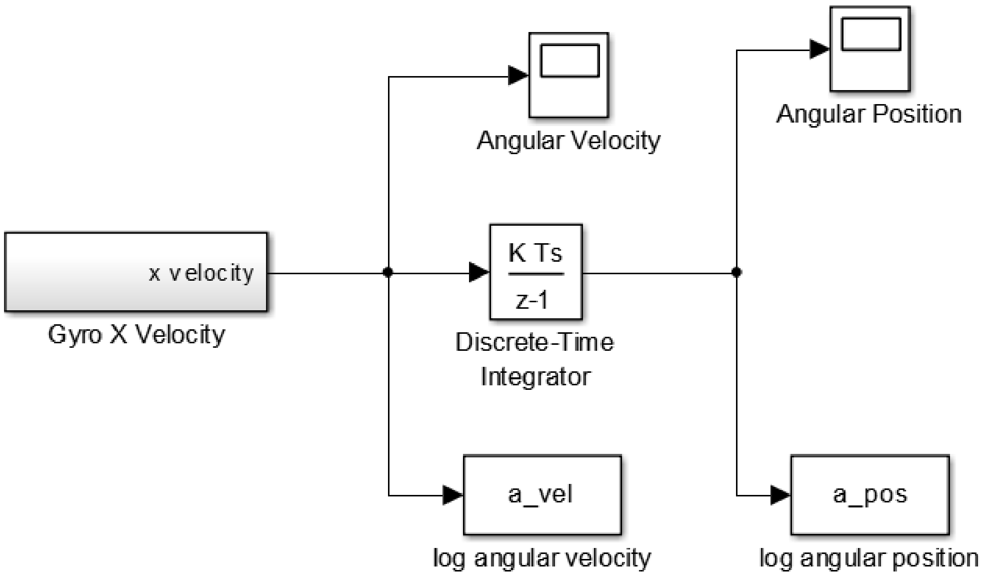
Note: The units are still unknown.

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.

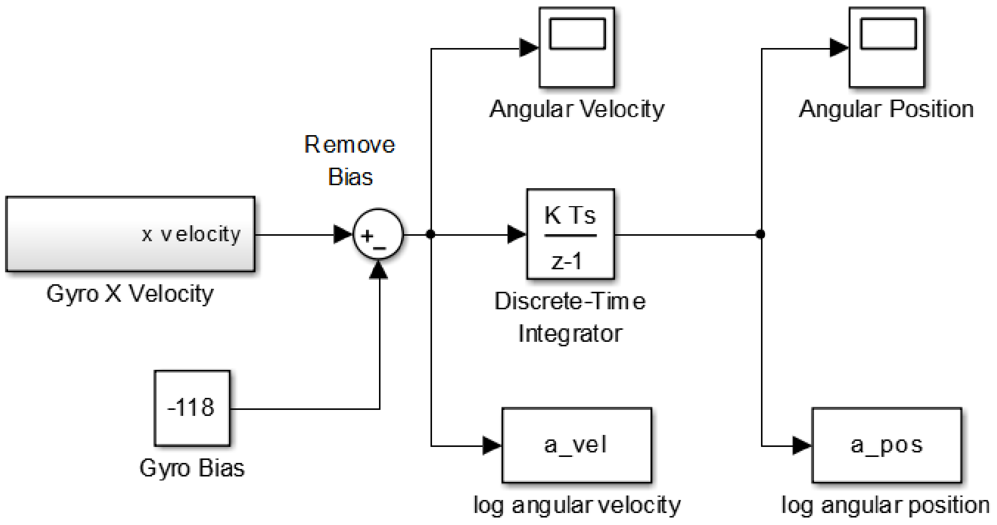
Simulink Model

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?

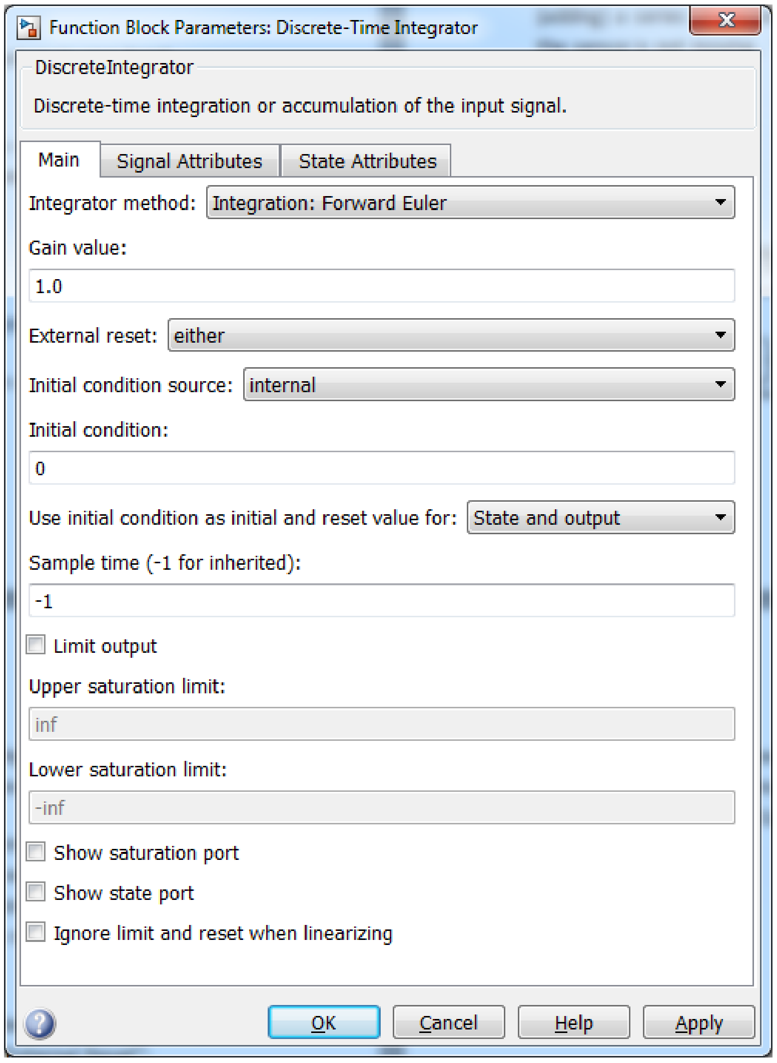
Since the sensor is not moving the values should be zero (no velocity). Any bias in velocity is integrating (adding) a series of nonzero numbers to the computed position even though the sensor is not moving. To eliminate this, remove the bias before integrating.



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”:



DiscreteIntegrator

Discrete-time integration or accumulation of the input signal.

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

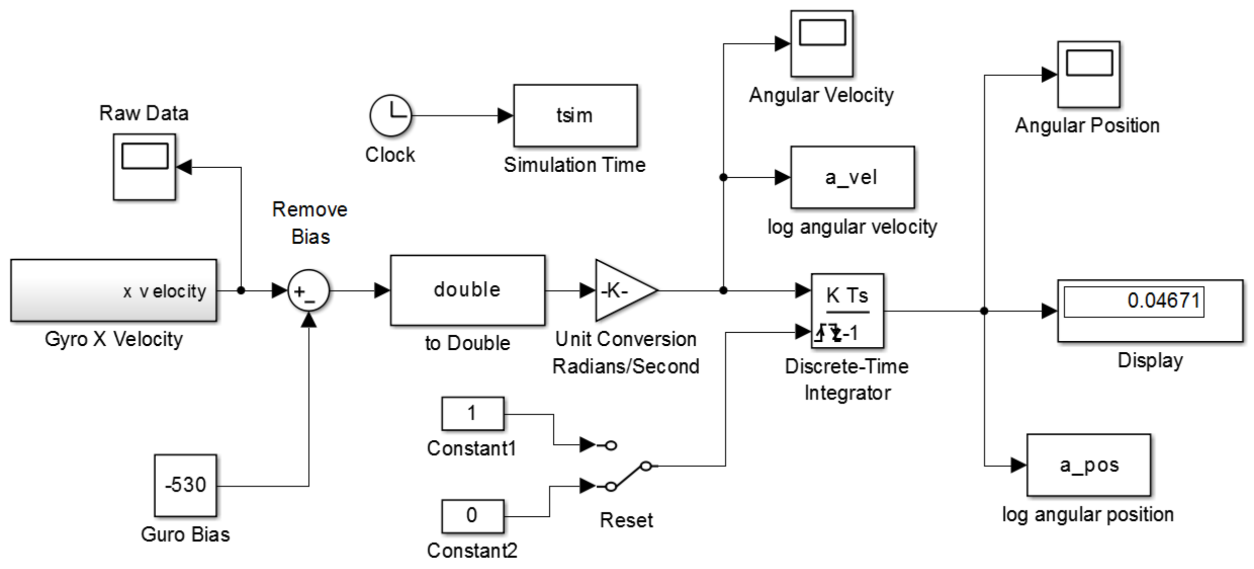
☐ Show saturation port

☐ 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

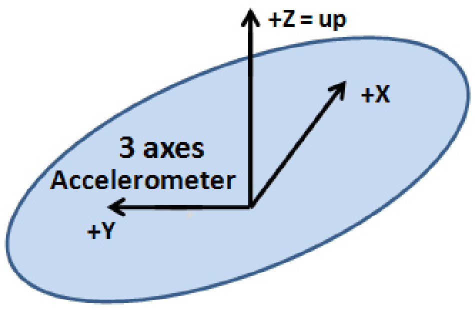
Question: What is the proper unit conversion factor to obtain the results in radians/sec and radians? Create a plot showing both the angular position and angular velocity versus time as the sensor moves from 0 to 90 degrees and back to 0. The units for this plot should be radians, radians/second and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

Background (Accelerometers):

Accelerometers measure the stresses caused by the movement of masses during linear acceleration or deceleration. Utilizing an array of these elements in three axes, we can effectively track linear movement in a 3D space. Because of the constant downward acceleration force of Earth’s gravitational field, accelerometers can be used to determine “down” in the world.



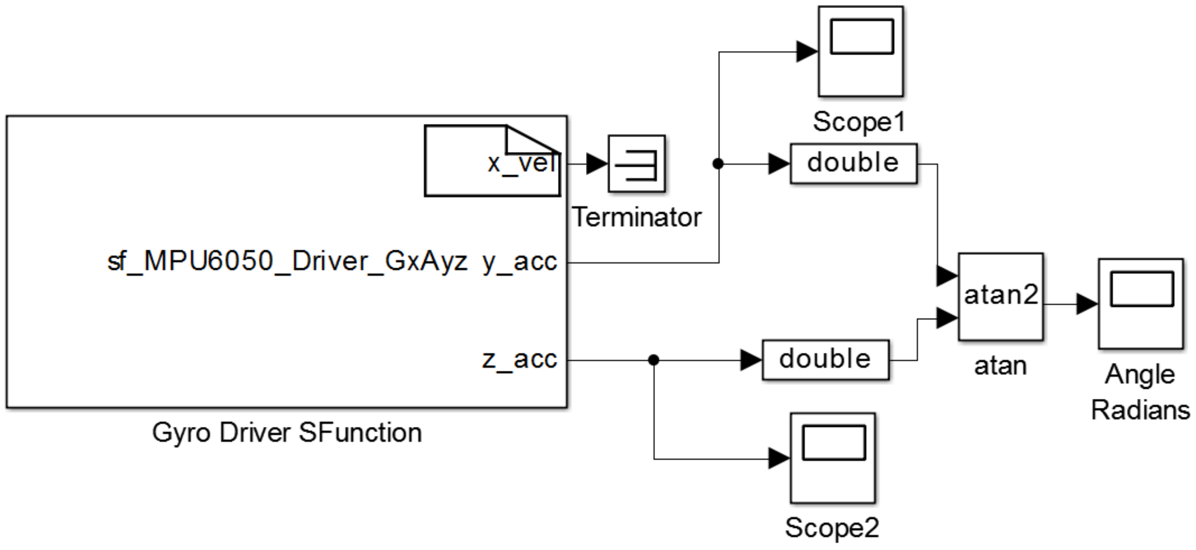
Using both sensors in one application can leverage the strengths of both sensors, while eliminating the drawbacks. One real world example of the use both sensors is the Nintendo® Wii Remote. After its launch in 2006, many users observed that the “WiiMote“ was great for sensing up waggle, but couldn’t accurately track more complex motions including rotation. This was because the WiiMote only relied on a 3-axis accelerometer for motion tracking. In 2009, Nintendo released a “Wii Remote Plus” add-on that finally enabled “True 1:1 motion tracking” in 6 DOF. The core of this add on was a gyroscopic sensor.

Part 3: Computing Board Angle with Accelerometer Data

In this exercise board angle is calculated using the accelerometer data by observing the components of the constant gravitational force (which is always “down”) on each of the accelerometers axes.

Data from 2 acceleometer axes (Y_acc and Z_acc) are used to calculate the angular position. The downward acceleration from the earth will always be fairly constant: if the board measures a Z-component of 9.81m/s^2 but nothing in the Y-component, it knows it is lying flat. If the Z-acceleration decreases, but the Y accelerometer measures some portion of this downward, we can calculate the angle using an arctangent block.

Create the Simulink diagram below.



Calculating Angle from Accelerometer Data

Checkpoint 3:

Run the system. Rotate the board 90°, and observe the result in the scope to verify that the angle is calculated correctly.

Discussion Questions:

- Do the measurements seem valid for all angles through a rotation?
- Repeately move the board from 0 to 90 degrees – is the angle measuement still accurate?
- How does the noise of the signal compare to that of the gyroscope function? (Plot the concurrent data from two methods on the same figure for comparison)

Part 4: Creating the Complimentary Filter

The previous two measurement circuits can be combined into one measurement system to utilize the characteristics of the two sensors: the gyroscope can measure quick changes in rotation, but has steady state error. The accelerometer can accurately obtain the steady state angle. A simple method of combining these two measurements is called a Complimentary Filter.

Create the Simulink diagrams below.

Note: The “GoTo” blocks are used to connect signals without actually having the “wires” drawn on the Simulink diagram.

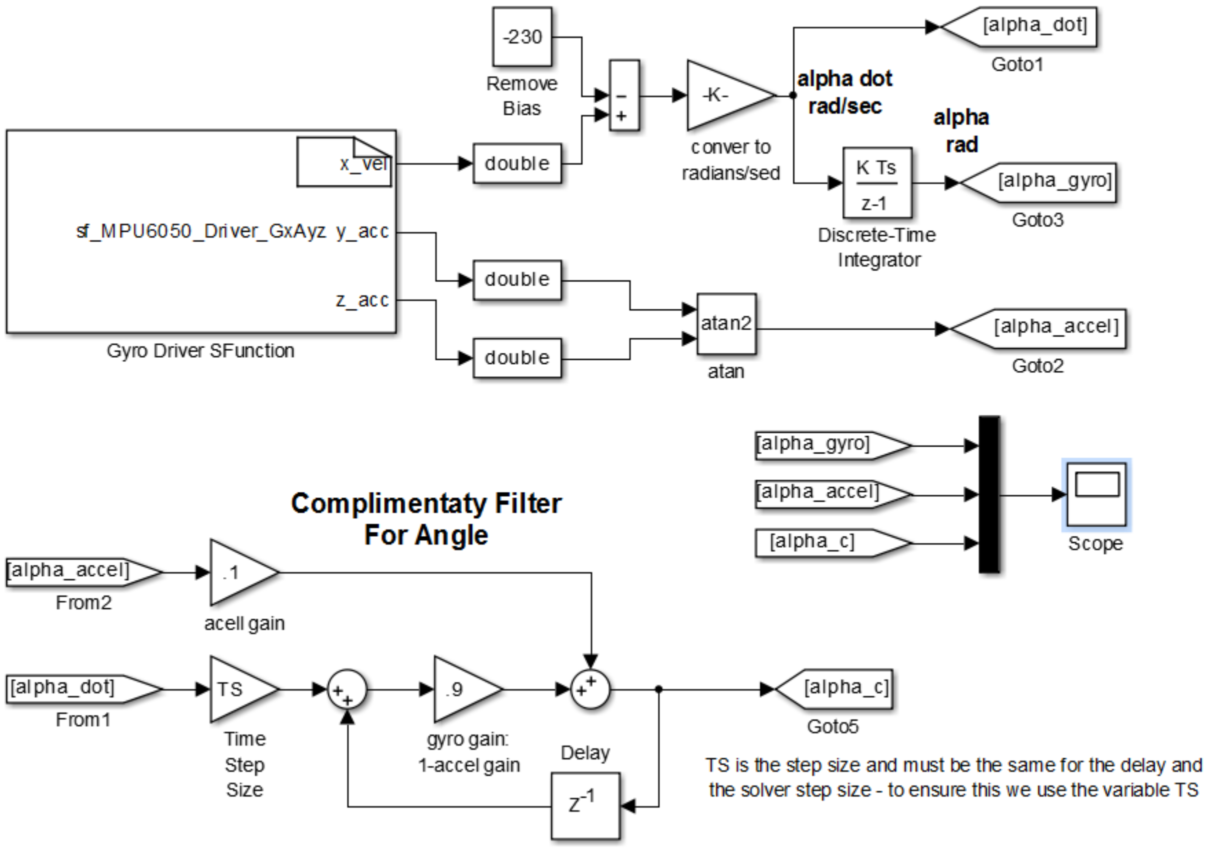
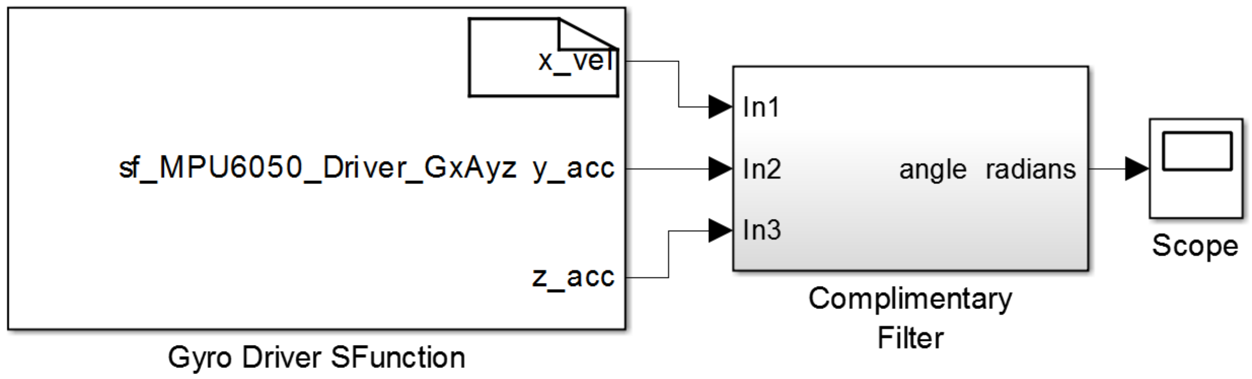


Figure 3: Components of a complimentary filter (detailed view)

The gyroscope only needs a small correction to compensate for drift accumulation. So most of the angle is measured from the gyroscope, in this case 90%. But to account for any drift that may accumulate 10% of each calculation comes from the accelerometer. The weights (0.1 and 0.9) can be adjusted to provide a trade off between speed of resposne, and steady state error, but their sum must add up to 1.

Now, select the components of the detailed diagram above, then right click and create “Create subsystem from selection”. This will create a block that can be easily inserted into code for other applications.



Complimentary filter with subsystem block

Checkpoint 4:

Run the system. Rotate the board in several directions, and observe the outputs.

Discussion Questions:

- What happens to the readings as you adjust the accelerometer and gyroscope gains? What is the system like if their values are reversed (0.9 and 0.1 resp.)?
- Provide a plot showing all 3 angle calculations as you rotate the board.