**Checklist for Lab06:**

1. **A computer installed with MATLAB compatible with Arduino.**
2. **Download the state flow model in Canvas/Laboratory Handouts**
3. **Uno or others board**
4. **Breadboard & jumper wires**
5. **MPU-6050**

**Lab has free MPU-6050 available, and amazon purchase link is available in the Announcement section of Canvas system.**

1. **Might need to install “Embedded Coder,” “Simulink Coder,” and “MATLAB Coder” from MATLAB Add-Ons**

In lecture, we have discussed orientation measurement via an MPU-6050 sensor. The measurement data can be fed to the I2C pins onto the Arduino Uno board. The MPU-6050 sensor is a 6-degree of freedom (DOF) inertial measurement unit (IMU) used to read acceleration and angular velocity in all three dimensions. It incorporates a gyroscope unit and an accelerometer unit and is accessible using the I2C communication bus. You can read the sensor data in real-time in MATLAB and visualize the data in a Simulink scope.

**Section 1. connect MPU-6050 to Arduino Uno**

The MPU-6050 works on normal voltage at 5V. (3.3V is also acceptable). You can insert the MPU into a letter column on the breadboard to help with wiring (Figure 1). The MPU lights up to indicate when it is receiving power from the UNO. Laying the MPU flat instead of upright will help prevent the MPU from popping out of the breadboard.

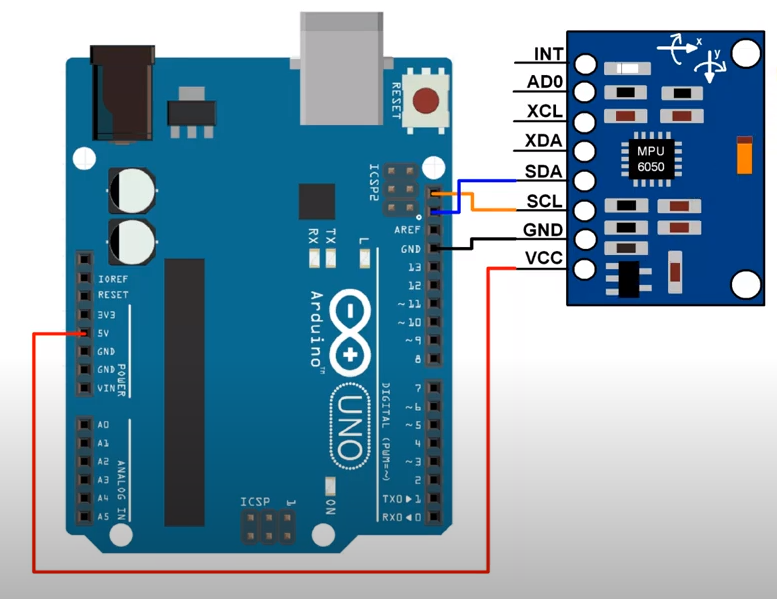


Figure . MPU-6050 Wiring

**Section 2. Simple sensor reading extraction with pre-built packages**

**Exercise 2.1:**

Simulink Support Package for Arduino Hardware / Sensors library provides an integrated block for an MPU-6050.

Text, letter

Description automatically generated

Figure . MPU block

Pull out an MPU-6050 block and connect the acceleration output to a scope and angular rate output to another scope. Translate & Rotate the IMU on the bread board, identify the , , , , , and states using the scope legend, and show 6 scope pics showing you can activate each individual state with your imu (or pass off with the TA).

🟉Consider using a Demux block to separate each state from the MPU-6050 block. See Figure 9 and Figure 11 for reference.

🟉The scope can add more inputs when you drag and hover a 2nd signal arrow over the scope. Assigning labels to the scopes’ input signals will correspond to the legend names in the scope)

🟉You will likely have to use “Connected IO” mode with “Run with IO” instead of “Run on board” mode with “Monitor & Tune”

🟉The direction of the three, orthogonal coordinate system is shown in Figure 10.

🟉It’s ok if the Simulink blocks turn red when you run the program

🟉Viewing the scope with a time span around 200 with wrap overrun action is a good view setup.

🟉Debug: Make sure the Uno’s & Simulink COM #s are consistent.

🟉Debug: As will be discussed later in this document, you may need to change the MPU6050 block’s I2C address if you are getting build errors.

🟉Debug: too much noise? Try letting the sensor stay in resting position for a minute. Try stopping and restarting the Simulink program

🟉Debug: 2+ states activating at the same time? isolating/decoupling signals is one of many sensor problems in engineering. Lifting the UNO and sensor off the desk might help. Nevertheless, in this lab, focus more on getting the desired state to activate with higher magnitude compared to the other state signals.

Accel Scope w/ 3 states & legend names

- activation:

- activation:

- activation:

Ang Rate Scope w/ 3 states & legend names

- activation:

- activation:

- activation:

**Section 3. Detailed sensor reading extraction with state flow**

Let’s try to understand what Simulink’s MPU block is doing “under the hood.” We will restart and repeat our objective of extracting sensor readings, but this time we will see the internals using state flow blocks.

The MPU-6050 needs to be initialized once it is powered up and stable. For initialization, we need to send the I2C address of the peripheral (slave) with the write command and then send the internal register number you want to write to. Simulink provides the I2C Controller Write Block (I2C Master Write) block, Figure 3, which enables us to write data to an I2C slave device or a device register. This block is used inside the **class-provided MPU-6050 Simulink model in Canvas**. Find this block in the class-provided model.

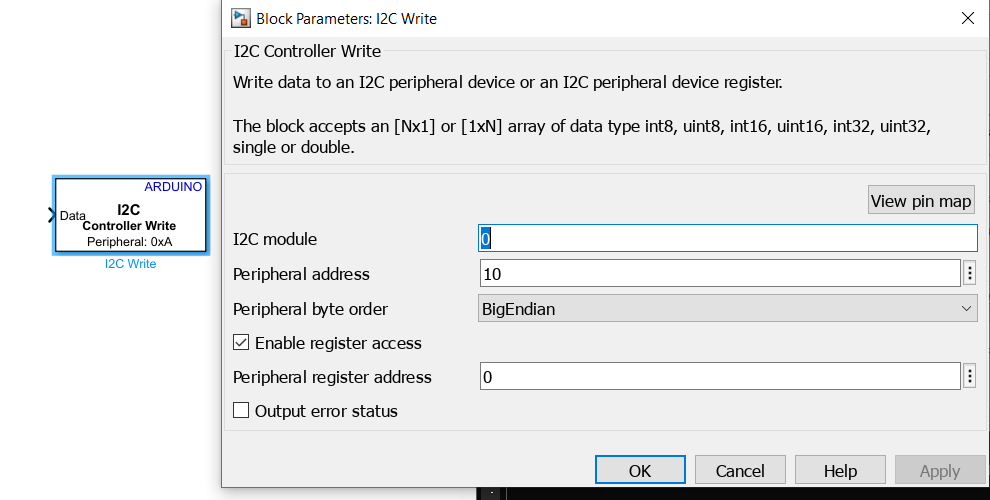
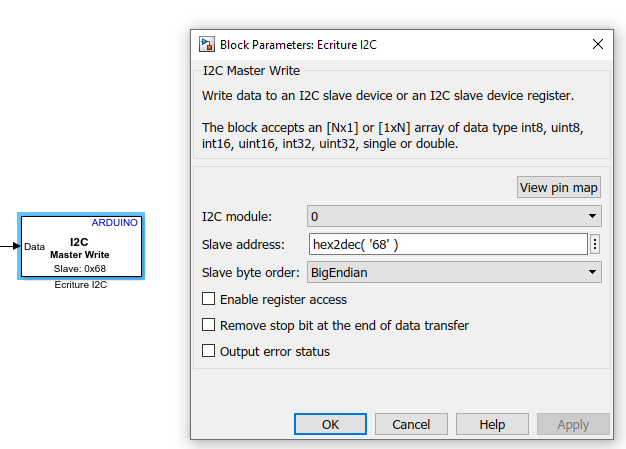
 

Figure . Two versions of the I2C Write Block in Simulink

The peripheral (slave) address of MPU-6050 can be found either from a data sheet or by typing the command as shown in Figure 4 in MATLAB after you connected your Arduino board and MPU-6050 to a computer.

Setup the Arduino with ***a = arduino*** and then type ***imu = mpu6050(a)*** to get the I2C address. Verify the MPU-6050 address and the I2C write block peripheral address match.

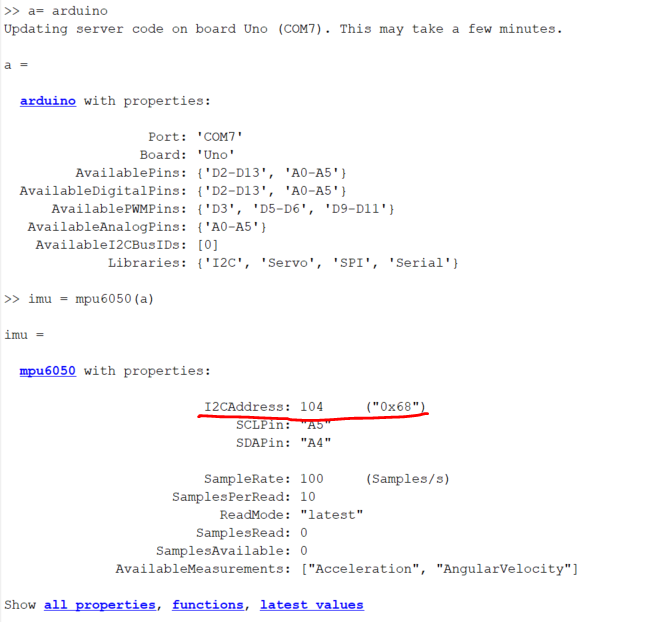


Figure . I2C Address of MPU6050

In this Simulink model, we use a State Flow Chart (the yellow box) which is a software package that extends MATLAB and Simulink with tools for modeling logic and event-driven systems. When active, the contents of the state are executed sequentially. The “entry” (also called “en”) action is executed immediately upon entering this state. After the entry phase and for as long as this state is active, the commands in the “during” section are executed; the code for “during” can be thought of as the body of a "for" loop. Then, as the state becomes inactive, the exit commands are executed. The Simulink Functions (a function that is defined using Simulink blocks) are called from within the State flow chart to write and read data for the MPU-6050 (using the I2C Simulink blocks).

The MPU-6050 provides a programmable full-scale range of ±2g, ±4g, ±8g and ±16g for acceleration and full-scale range of ±250, ±500, ±1000, and ±2000°/sec for X-, Y-, and Z-Axis angular rate sensors for Gyro. During the initialization process, the range ±4g and 1000 °/sec was selected by writing commands to the specific register as shown in Figure 5.

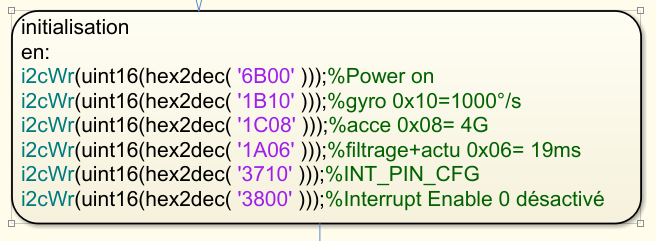


Figure . Initialization of MPU-6050

The specific register to read and write is in the data sheets but once it's configured and running, you can read the data. The gyro data starts at register 0x3b. The register map and data sheet of MPU-6050 are available online and screenshots attached for your reference.

**You shouldn’t need to change any other parameters within the state flow block.**

Go to **Modeling/Model Explorer/Edit/Properties/Callbacks/InitFcn** to confirm the initialization function, as shown in Figure 6 (This is yet another way you can define variables in Simulink).

A screenshot of a computer

AI-generated content may be incorrect.

Figure . Model Explorer

Referring to Figure 9, the outputs from the state flow model need to be scaled based on the sensitivity scale factors. From the datasheet, the sensitivity scale factor can be found for the gyroscope, accelerometer and temperature based on the range we have selected. An example of the gyroscope scale factor is shown in Figure 7.

For example, in Figure 5, we selected gyro measurement range is 1000/second. Referring to the MPU-6050 datasheet, the conditions FS\_SEL = 2 corresponds to TYP with 1000/second. Then the sensitivity scale factor can be found to be 32.8. Hence, we need to put a gain 1/32.8 to the measurement output from gyro sensor reading.

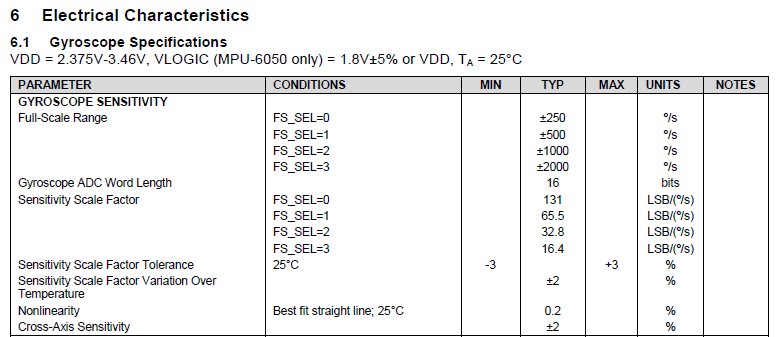




Figure . Sensitivity Scale Factor for Gyro Example

**Exercise 3.1: find the scale factor for accelerometer**

Refer to Figure 8, the datasheet for accelerometer sensitivity,

Suppose we are selecting the full-scale range of 4g, what is the scale factor for this scale range?

Scale factor (not gain): \_\_\_\_\_\_\_\_\_\_\_\_\_\_

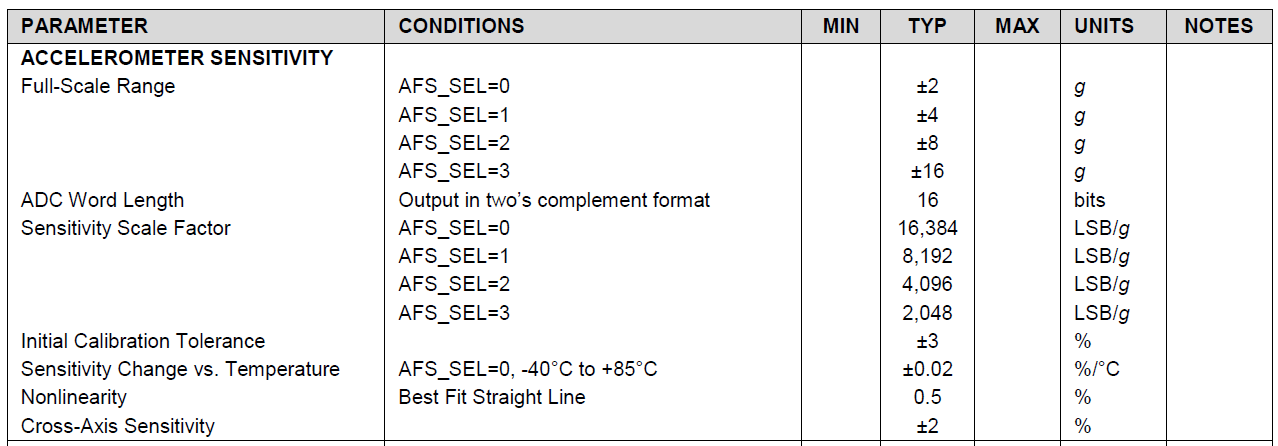
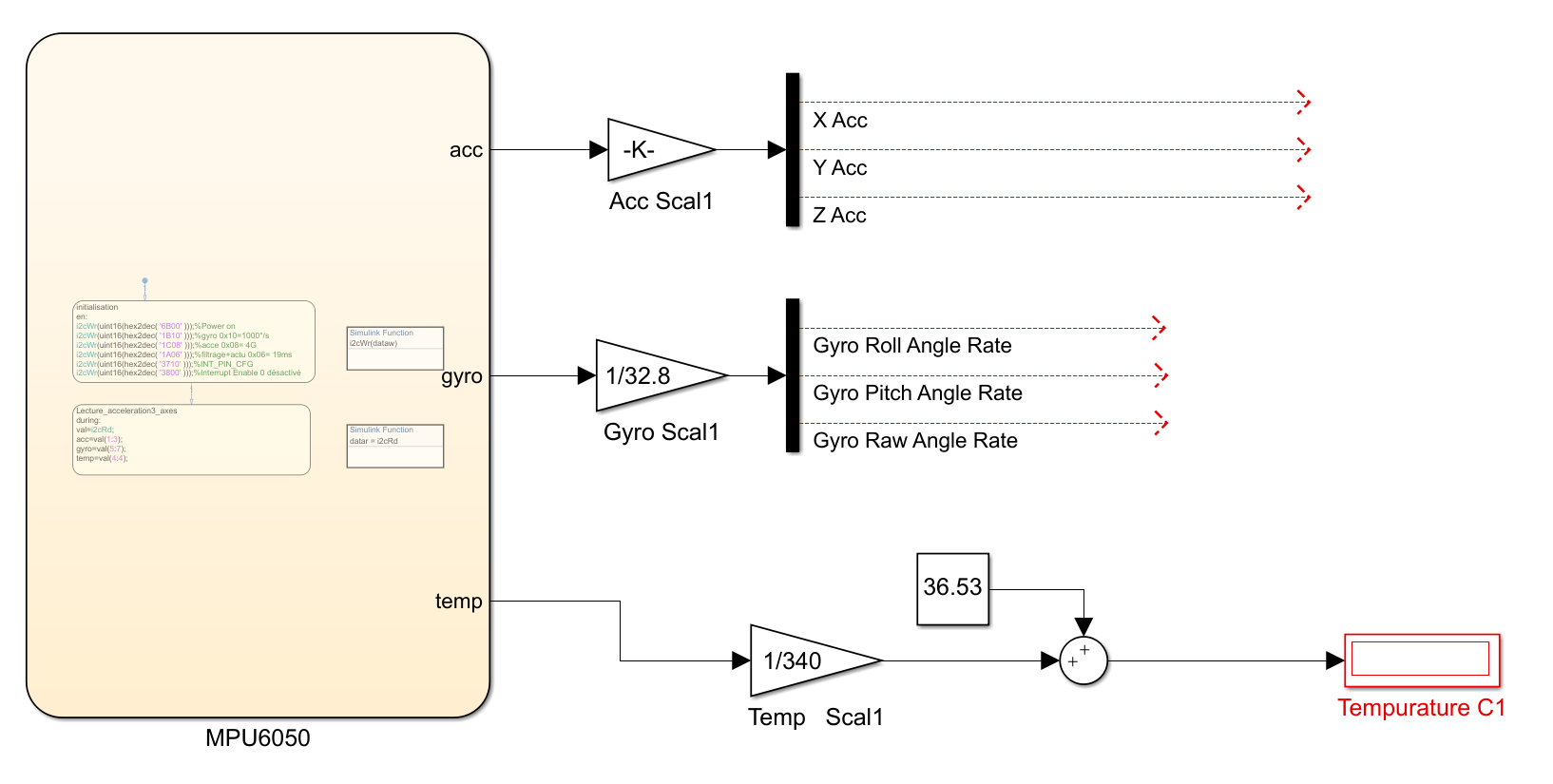




Figure . Sensitivity Scale Factor for Accelerometer

We can calculate acceleration and angular velocity by dividing the sensor’s raw data with their respective sensitivity scale factor. One example is shown below for the gyro in Figure 9. A **Demux block** will be used to split signals relative to three axes, respectively.

**



Temp. calibration

Can be changed.



Figure . Calculation Based on Scale Factor

Connect the acceleration and gyro outputs to scopes (temp output optional but might be useful for debugging or other applications) and translate/rotate the MPU-6050 along each axis manually to confirm sensor values are being received (similar to Exercise 2.1).

**Section 4. Extended sensor utility**

Now that we know how to acquire sensor readings and know a little more about what is happening “under the hood,” let’s exploit one indirect way to use our sensor. We can use our sensor readings to determine more than just acceleration and angular velocity. We will practice manipulating our sensor readings with complementary filters to determine angular position too!

As introduced in the lecture, the pitch angle can be calculated from either accelerometer signals or gyro signals. Using the accelerometer signals, the pitch angle can be calculated by

and using the gyro signals, the angle was calculated from angle rate as below by a discrete transfer function

Using Figure 11 and Figure 12 as reference (figure wirings are not necessarily correct nor concise), develop the pitch, yaw, roll angle signals from both the accelerometer and gyroscope readings (E.g., for the accelerometer pitch, choose Trigonometric Function from math library and select atan2 for function in parameter dialog. Connect this block to X acc and Z acc, then the pitch angle can be calculated based on the accelerometer).

Connect these angle outputs to new scopes and rotate the sensor along each axis manually so that you can identify which angle corresponds to which Segway axis. Again, the direction of three, orthogonal coordinate system is shown in Figure 10.

**Exercise 4.1: Identify rotation axis for pitch, yaw and roll**

4.1 Place the sensor on the desk facing you as shown in Figure 10, and answer the questions on which rotation axes (x,y,z) are for roll, pitch and yaw movement.

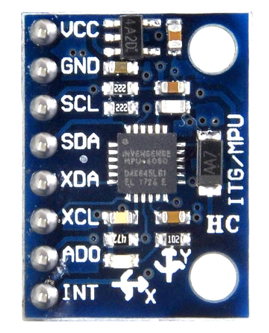


Figure . Definition of Orthogonal Axis System. X, Y axis are marked on chip, Z axis is the vertical direction orthogonal to X,Y.

Which axis is the rotation axis of pitch movement? \_\_\_\_\_\_\_\_\_\_

Which axis is the rotation axis of roll movement? \_\_\_\_\_\_\_\_\_\_

Which axis is the rotation axis of yaw movement? \_\_\_\_\_\_\_\_\_\_

**Exercise 4.2: Compare sensors**

4.2 Attach a screenshot of your Simulink model and attach screenshots of your scopes showing you can activate angle readings from both the gyro and accelerometer (or pass off with the TA).

- Simulink Model:

Screenshot of Accelerometer’s angle scope (with 3 angle states & legend names) and Gyroscope’s angle scope (with 3 angle states & legend names) side by side both activating the same angle

- Pitch angle activation:

- Roll angle activation:

- Yaw angle activation:

🟉Debug: Sensors aren’t zeroed out (bias/initial offset)? signal drifting or oversensitive? Units not intuitive? Gyro signal sign opposite compared to accelerometer signal? These are also some of the many sensor problems in engineering. For this 3.2.2 exercise, we won’t worry too much on correcting these, but we will explore one way of compensating for some sensor errors by using sensor fusion in the next section.

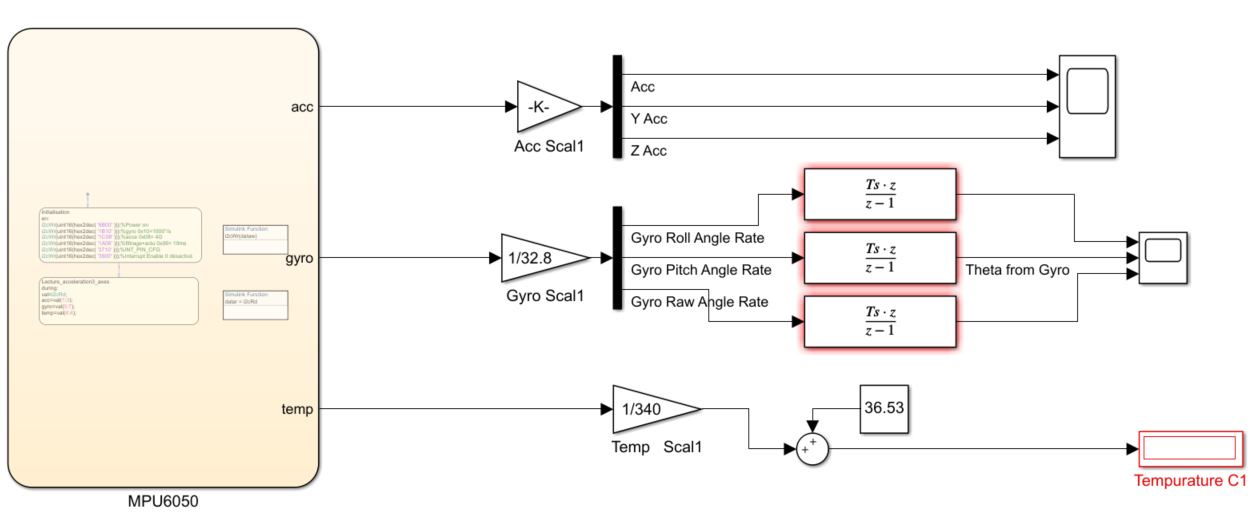


Figure . Reference Model. The gain K of Acc Scal = 1/scale factor, the scale factor is from the answer in Exercise 3.1.

**Section 5. SENSOR FUSION FOR PITCH ANGLE**

After finding out what the angles are, we will now experiment with sensor fusion theory using the pitch angle. Note that the pitch calculated using the accelerometer signal is accurate at low frequency and pitch angle based on the gyroscope is accurate at high frequency. The complementary filter will be adopted, and the basic idea behind the filter is to combine the two outputs obtained with the accelerometer and the gyroscope to obtain a good estimate of the orientation angle in both high and low frequency ranges (Note: the data type in all Gain blocks in the Simulink model need to be chosen as ‘double’).

1. Add a gain block to convert the units of the accelerometer angle outputs in Step 2 from radian to degree so that the unit will be consistent with the output from gyro. Make a new scope that will compare the accelerometer’s pitch angle and the gyro’s pitch angle on the same plot.
2. Run the model in the connected IO or external mode with infinite time. Describe the difference of pitch angle from two different sensors.

**Exercise 5.1: Describe difference:**5.1 Describe the differences between the Acclerometer and Gyro pitch angle outputs (~3 sentences w.r.t. sensitivity, drift, and bias).

-Differences:

1. Add complementary filters into the model to set up a sensor fusion effect we learned in class.
2. Run the model with a complementary filter and compare the pitch angle signal before and after sensor fusion.

**Exercise 5.2:**

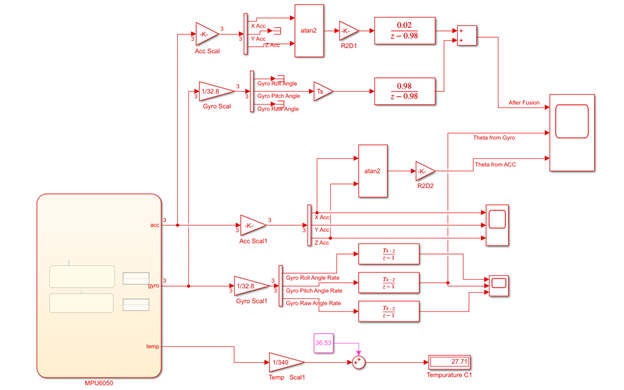
Show the results in the scope comparing the responses of complementary filtered signal and original signals from gyro and accelerometer.

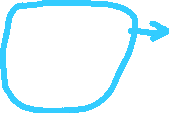
-Scope (accelerometer, gyro, and fusion signals with legend):

Compare the fusion signal with the accelerometer signal and gyro angle signal (~2 sentences).

-Compare:

The example model and result are shown in Figure 12 and Figure 13, respectively.





Complementary filter

Figure . Reference Model for Sensor Fusion. (Gain of R2D1 and R2D2 block after atan2 block is (-180/pi)), which is used to convert from radian degree to degree °.

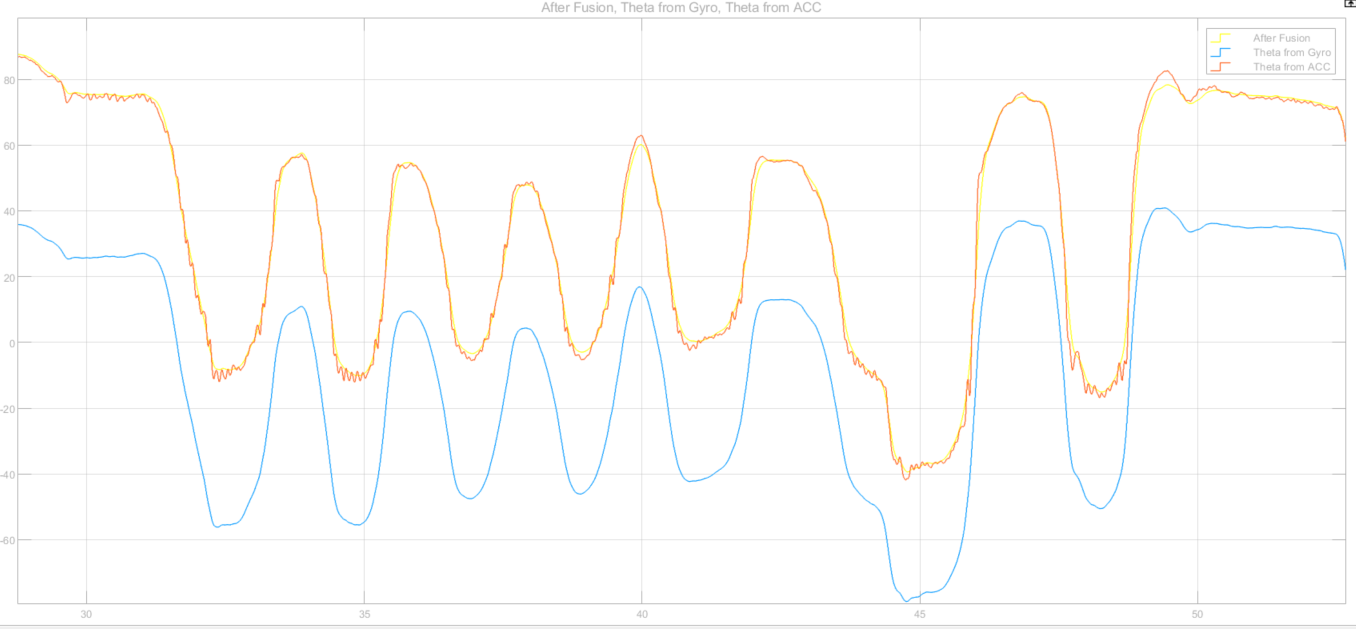


Figure . Reference Results for Sensor Fusion. Fused sensor data is yellow and it can be observed after enlarging the photo.

All done! Disassemble your hardware and clean up your workstation.