Autonomous Vehicle Design, Fall 2020

Homework 4, MPU-6050 Sensor

Total Points, 100

**1. Bias Offset Calibration (40 points)**

Implement the lesson 6 sensor bias calibration for the gyroscope and accelerometer. Store the offset values as MPU 6050 class data members. This will be easier and likely more successful than storing them in offset registers.

Use conditional compilation for serial debugging print statements in your calibration function. Print each set of mean values calculated, each set of offset values as they are updated, and the number of tests that pass the error level test with each loop iteration. When the calibration is complete, print the final offset values. This debugging information should be transmitted to a python program. The python program should print a table showing the iteration number and mean values. You may plot this data as well. Tables and plots should be displayed on the computer screen as well as written to a file.

Experiment with the error level convergence. Start with larger error levels to ensure your algorithm does converge, then try making them smaller. Find an error level that converges for your sensor.

You don’t have to use the 1000 sample size for calibration. You may choose another sample size but may want to try this first as the Internet wisdom seems to support it. If it works, consider it good and move on.

Provide a written summary of the error values tested while trying to find the values that converged for your sensor. Include the tables and/or plots in your summary. You do not have to document every error value tested but should include at least 2 for comparison.

You are not required to use the source code from lesson 5. You may write your own MPU 6050 class.

**2. Measured Error after Calibration (60 points)**

1. Add the following functionality to your MPU6050 class.
   1. Function that returns the measured gyro x, y, z values, unscaled, but corrected for the bias offset.
   2. Example: read the gyro x measurement register, adjust the value with the calibration offset, return that value. Do the same for gyro y and z.
   3. Function that returns the measured accelerometer x,y,z values, unscaled, but corrected for the bias offset.
   4. Function that returns the gyro x, y, z values, scaled to degrees/sec. (Remember to adjust measurements by the bias offset before scaling.)
   5. Function that returns the accelerometer x, y, z values, scaled to g. (Remember to adjust measurements by the bias offset before scaling.)
   6. Function that returns the temperature in degrees C.
2. Write an Arduino program that samples 1000 gyroscope and accelerometer measurements after calibration. Transmit the data in real time, after reading each set of measurements. In other words, do not take 1000 samples, store them in an array, and then transmit. Take one sample, transmit it, looping through 1000 times.
3. Write a python program to receive the 1000 samples of the offset adjusted measurements and save the data in a file or files. You may wish to write a separate python program to perform the data analysis.

The python program(s) should produce the following information:

* 1. Files - Save received data measurements in file(s).
  2. Error calculations
     1. min, max, mean, and standard deviation for each of the six unscaled measurements. Units are LSB.
     2. Average errors in units of g and deg/sec?
  3. Acceleration Measurement Plot – create a subplot for each of the 3 axes.
     1. Example: subplot 1 is x axis measurement data. Subplot 2 is y axis measurement data. Subplot 3 is z axis measurement data.
  4. Gyroscope Measurement Plot – create a subplot for each of the 3 axes.

1. The MPU-6050 should be placed on a flat level surface and not moved while conducting this experiment. This allows you to calculate the average error because you know what the expected value should have been for each measurement.
2. Write a brief summary of how well the calibration worked. Your analysis should include a table of the offset values used, the average error, min error, and max error calculations and your plots. You may generate additional plots if desired.

Submit your program source code files in addition to your written analysis.

**Submission Summary**

|  |  |  |
| --- | --- | --- |
| Problem 1 | Source Code: Arduino & python | Written analysis |
| Problem 3 | Arduino & python source code | Written analysis |

***If any program is not complete or if it is not working correctly, submit a text file detailing what works, what is missing, and what is broken.***