Biped Patrol - Task 1.1

Task Instructions

Problem Statement:

In this task you will be implementing the frequency filters through coding in Octave. Before starting, please read the content provided in **“Task_Theory.pdf”** and make sure you understand the concept of frequency filters. Once you understand the concepts, open the **“read_write.m”** file. You will see skeletons of some functions. The expected performance of those functions are described later in this document.

Your task is to understand the requirements of those functions and then write the code for these function in **“read_write.m”** file. After writing the code based on the instructions provided later in this document, RUN the **“read_write.m”** file, which will generate a file named **“output_data.csv”.**

In the task folder you will find two files named **“sensor_data.csv”** & **“read_write.m”**.

“sensor_data.csv” contains readings from sensor module named GY-87 which is a 9-DOF motion sensor. These readings are obtained from 12 registers named as below:

Sr. No.	Register Name	Sr. No.	Register Name
1	ACCEL_XOUT_H	**7**	GYRO_XOUT_H
2	ACCEL_XOUT_L	**8**	GYRO_XOUT_L
3	ACCEL_YOUT_H	**9**	GYRO_YOUT_H
4	ACCEL_YOUT_L	**10**	GYRO_YOUT_L
5	ACCEL_ZOUT_H	**11**	GYRO_YOUT_H
6	ACCEL_ZOUT_L	**12**	GYRO_YOUT_L

Table 1: Register Names

Warning: Do not edit "sensor_data.csv". If edited then this will affect the output of the code written in read_write.m file.

These readings are taken at an interval of **10 milliseconds**. These readings help us find out the **tilt of the body on which the sensor is mounted**.

Readings of registers from Sr. No. 1 to 6 in the Table 1 are from the Accelerometer present in the sensor module. Similarly, readings of registers from Sr. No. 7 to 12 in the Table 1 are from the Gyroscope. Accelerometer and Gyroscope in the GY-87 sensor module gives us readings for all three axes (X, Y and Z).

In order to get data for a particular axis from Accelerometer or Gyroscope, you have to combine readings from its respective HIGH and LOW registers.

For example: if you want to get Accelerometer reading for Y axis, then you have to combine values from ACCEL_YOUT_H & ACCEL_YOUT_L. **As the High and Low registers are 8-bit in nature, therefore the combination of both will be a 16-bit signed data.** This data can thus take values from **-32767 to +32767**.

In GY-87 sensor module you have MPU-6050 sensor which integrates 6-axis MotionTracking device that combines a 3-axis Gyroscope, 3-axis Accelerometer, and a Digital Motion Processor (DMP) all in a small 4mmX4mmX0.9mm package. With its dedicated I2C sensor bus, it directly accepts inputs from an external 3-axis compass to provide a complete 9-axis MotionFusion output.

While recording the data for **"sensor_data.csv"** we have configured the sensor module with certain settings. For this we wrote following values to designated registers in the sensor:

Register Name	**Register Address**	**Value Written**
PWR_MGMT_1	0x6B	0x01
GYRO_CONFIG	0x1B	0x00
ACCEL_CONFIG	0x1C	0x00

Table 2: Configuration for sensor readings

To understand the effect of these settings, read the datasheet and register map of the MPU6050 provided. These files are named as **"MPU-6050_DataSheet.pdf"** & **"MPU-6050_Register-Map.pdf"** respectively.

This will also help you to find the **scaling factors** for function mentioned further in this document.

Once data for individual axis is obtained, it further needs to be passed through the HIGH-PASS/LOW-PASS filter and then through COMPLEMENTARY filter to obtain the actual tilt angle.

read_write.m

In this file you will find following functions:

read_accel	**read_gyro**
lowpassfilter	**highpassfilter**
comp_filter_pitch	**comp_filter_roll**
execute_code	

Table 3: Functions in read_write.m file

global A = csvread(' sensor_data.csv'); is the first line of the code, which reads the **sensor_data.csv** and puts the data in the array **"A"**. The dimension of **"A"** will be 8000x12. These 12 columns contain data for the registers from Table 1 in the same order.

Function description:

1. read_accel & read_gyro:

These two functions are going to be used for combining the High and Low values of the registers mentioned in Table 1 for respective axis. *read_accel* function will work on readings of registers from Sr. No. 1 to 6 in the Table 1. Similarly, *read_gyro* will work on readings from Sr. No. 7 to 12 in the Table 1. Then the output of **read_accel** & **read_gyro** are passed as the argument to **lowpassfilter** & **highpassfilter** respectively.

2. lowpassfilter & highpassfilter:

These two functions are going to be used for filtering the data of the sensors as the data is highly noisy. One of the arguments in these two functions is *f_cut* i.e. cut off frequency. **Keep *f_cut* as 5.** You already understand that cut off frequency is the point which defines the point of blocking/passing of the signal based on their frequency.

3. comp_filter_pitch & comp_filter_roll:

Use these two functions to compute Pitch and Roll by implementing the concept of Complementary Filter on the values filtered by **lowpassfilter** & **highpassfilter**. Both of these functions take six (6) values as argument.

For calculating the Pitch: use the changes in values of accelerometer readings of Y-axis & Z-axis and gyroscope readings of X-axis.

For calculating the Roll: use the changes in values of accelerometer readings of X-axis & Z-axis and gyroscope readings of Y-axis.

4. execute_code:

In this function, you will call all of the above mentioned function to calculate the Pitch & Roll. Once they are calculated, you will save them in an array named **“B”**. This array will be of size 8000x2.

First column of the array “B” must save value of Pitch calculated from the values from respective row of elements from array “A” from respective row. Similarly, the second column of “B” must save value of Roll.

For example: At 343rd row and 2nd column of array **“B”** must contain the value of Roll which will be calculated using the values of elements from 343rd Row and 1st to 12th column of array **“A”.**

Last line of this function is csvwrite(‘output_data.csv’,B); write the Array **“B”** in a csv file. Do not edit this line.

Do not open or edit this file. You have to submit this auto-generated **output_data.csv** as it is.

That's it !! Task 1.1 is complete!! You can move on to the Submission Instructions given in **"Read Me.pdf"** to complete submission of task.