ECSE426 - Microprocessor Systems

Winter 2017

Lab 3: MEMS Accelerometer, Timers and Interrupts

Lab Summary and Objectives

This exercise will introduce you to the peripherals attached to the processor in the F4-Discovery kit. It is aimed at designing a system that detects the board's tilt angles by processing readings of a tri-axial accelerometer. You will be using ST accelerometer sensor LIS3DSH (the MB997**C** version) to read (sample) the accelerometer at a rate of 25Hz. The MEMS sensor should generate an interrupt once data is ready and configure the associated GPIO to raise an interrupt in the microprocessor.

The board will be interfaced to a 4 by 4 keypad. Given the board orientation and the desired orientation, the 4 LEDs on the board will visualize the proximity of the entered angle from its current position by the intensity of the LED's light, such that if the board reaches to the desired angle, light intensity of LEDs should be the least (or OFF).

The user provides a desired target angle using the keypad. The angle can be anywhere from 0° to 180°. The player will provide a one, two or three digits **integer** angle input through the keypad. The user must follow the numerical sequence by pressing a keypad button. You should use **#** to simulate an ***Enter*** button and ***** to ***Delete*** the last digit was entered. The operation of entering the angle has two phases, first entering a desired roll angle, then by pressing "Enter" you get into phase two, which is entering a desired pitch angle. To restart the operation at any time, the user has to press * for longer time (around 2-3 sec). The actual values will be displayed on the 7-segment or using Printf function (whatever you prefer).

Once the "Enter" button is pressed, the processor will compare the current board orientation to the desired one. There are two scenarios:

1. The board angle is approaching to the target ***pitch***, so brightness of the two LEDs (like LD3 and LD6) in that direction should be reduced until you reach within the 5° difference range (should be almost OFF).
2. The same as in scenario one, but for the ***roll*** and two other LEDs (like LD4 and LD5) in the roll direction.

Lab requirements

(1) Tilt detection

You are required to design a system that calculates tilt angles in 2 dimensions with 4^{o} accuracy (use a protractor). The system should be calculating angles in real time, at 25 times per second. The MEMS accelerometer sensor measures the amount of acceleration the board is subjected to. Under no external force, the only force acting on the board is the gravitational pull. Given that the earth gravimetric pull is 1g (1000mg), we can determine the angles of the board through gravity force projections over the 3D Cartesian

axes. Those values are measured by the sensor and read by the user through driver API calls. The values are then used to measure orientation through simple equations (refer to **Doc_15 - Tilt angle application notes).** You will be able to measure angles along the x and y directions. The ones measured along the x axis are called the roll angles (α) and those along the y axis are called the pitch angles (β). For this part, you can choose which tilt angle (α or β) that you will base your application on in advance. You then hard code it in your code.

Please read the class notes and, the application notes by ST Microelectronics on how the tilt angles are measured from the projections of the gravity force. The notes also contain the explanation on how various imperfection sources (zero-*g* offset accuracy, temperature dependence etc.) get treated. Please note that the application notes are quite generic and you have to be cautious when you implement them for your specific MEMS chip. Also note that MEMS drivers we are using are not part of the STM32F4Cube HAL but in fact written by us. You have to include them on your own (or write them from scratch as we did for the Ver. C boards). In the base project, you will see the driver listed as LIS3DSH.

Side Note

It is worth noting that accelerometer sensors on their own are not accurate to measure tilt angles or generic dead reckoning applications. A more complex set of equations use readings from magnetometers and gyroscopes to determine 3D space orientation. In most modern smartphones/tablets, a single or a set of MEMS chipsets are provided for more application accuracy. For example, though accelerometers are simple to use for tilt angle measurement, they are often quite slow to react to angle changes. Gyroscopes are much more sensitive to angle changes but suffer from drifting values. Both cannot be used to determine headings accurately but magnetometers use earth's magnetic field as a way to do so. A system which uses all three sensors (9DOF or 9 degrees of freedom) is often used to take advantage of the best features of all sensors and offset their weaker points. **A newer version of the board we use in the lab was released in December 2014, the STM32F411 Discovery board has all three sensors; accelerometers, magnetometers and gyroscopes**. We also have at our disposal in ECE labs a high-end breakout board **LSM9DS1** which features all three sensors. Feel free to use that board but note you have to modify the driver files of LIS3DSH to make the compatible with the newer sensor.

---

(2) Sensor Calibration and filtering

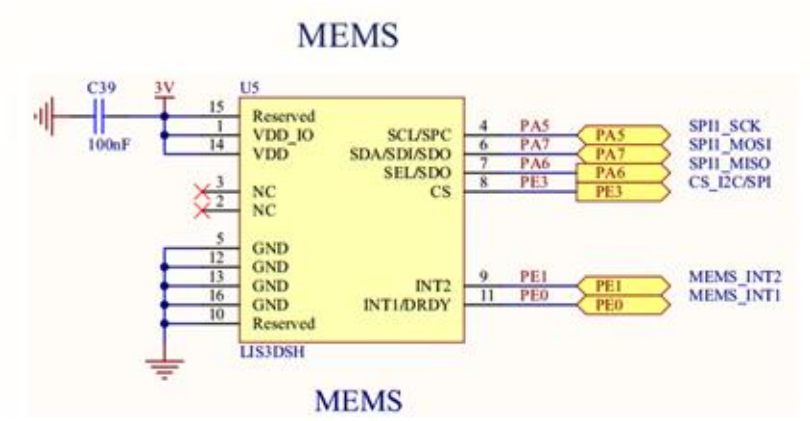
You are required to calibrate the sensor prior to their use in your application. You are advised to use offline calibration. That is, write the code which gets the calibration values for your board and then use those values in your implementation. This step is only to be done once and never be used again during application **run time**. **You are required to show and describe in detail the steps conducted in getting the calibration values.** Finally, do not forget to **filter** the data using the FIR filter. You have to *do an analysis on the parameters of the filter* so as to get decent filtering for the MEMS signals similar to what you did with the temperature sensor. The parameters are filter order and filter coefficients. In Lab 1 and 2, those parameters were fixed and given to you. You are required to find a best value for parameters based on your observations. Then, use any mathematical package of your choice to analyze and compare the original noisy data with different versions of the filtered data with various parameters. You have to present a solid convincing case to justify your choice. **We expect some visual analysis (i.e. graphs) of raw sensor data against filtered data with varying parameters.**

There are many ways to calibrate the data, one easy way is by taking averages to calculate the offset then use this offset to calibrate future readings as described in the tutorial. Another elaborate and a more accurate method is described in to **Doc_15 - Tilt angle application notes.**

(3) MEMS interrupts and NVIC

You are required to set up your MEMS sensor to generate an interrupt signal whenever a sample is ready; that is, your interrupt should occur at the same rate new acceleration samples are ready (25 Hz). However, this setup and configuration will only set up an internal flag and a signal on the interrupt pin of the MEMS sensor chipset. This chipset is an external chipset and its interrupts and action is isolated from your microprocessor inner workings. You need to configure your microprocessor to react to this external interrupt. The interrupt signal should be non-latched (pulsed) and active high. Refer to the schematics to see how the MEMS sensor is interfaced to your processor. Make sure you clear interrupt flags both from your processor's end and the sensor's end inside your interrupt service routine ISR (a.k.a interrupt handler).

IMPORTANT: Note that the data ready interrupt signal in the LIS3DSH sensor is only generated on INT1 line. This line is hard-wired to PORT E.0 of our processor. Since all PIN0 on all ports share the same external interrupt line 0, there might me be a conflict with the push button on PORTA 0, therefore, refrain from using the button in this lab. You can de-initialize and reset PA0 for maximum assurance.



(4) The alphanumeric keypads.

In order to read a 4*4 or 4*3 key pad given to you, a polling technique can be considered to perform this task. You should write the code such a way to minimize the bouncing effect. A tutorial on how to use keypads is uploaded on myCourses.

(5) The Timer in PWM mode

Configure the timer to generate the pulse at 2 KHz frequency on all GPIOs. To get the specific bus clock value which clocks TIM4 and subsequently used to derive the desired clock, refer to "**Doc_00 - STM32F4 Processor clock tree and configuration values".** The duty cycle (D.C.) of the timer is controlled by a value which is the difference between the target angle and the current angle.

(6) Hint for the next lab

Next lab (Lab 4) will be based on Lab 2 and Lab 3 (this lab). So, make sure both of your codes are working. We will merge both labs into one big project using real time operating system (RTOS) services. Moreover, we might replace/extend certain functions.

Putting all requirements in one perspective

Here we will summarize the programming requirements you need to do:

1. Setting up the accelerometer
    a. Setting up the accelerometer configuration and sampling data rate
    b. Setting up the accelerometer to generate interrupts when samples are ready
    c. Configure the GPIO to which the accelerometer interrupt output signal is hardwired
    d. Configure and enable the interrupt through NVIC module
2. Signal processing operations:
    a. Conduct experiments needed to calibrate sensor data for all sensor axes
    b. Filter the data through FIR filter and investigate filter parameters
3. Setting up the alphanumeric keypad
    a. Look for free (no conflict/hardwired) GPIO pins and configure them as GPIO as needed *(Avoid at all costs using PA13, PA14 and PB3. Reconfiguring these pins will damage the board as they are interfaced with the debugger/programmer. You will lose connectivity to the board and ST-Link will no longer detect the board)*
    b. Write the alphanumeric keypad scanning algorithm
    c. Write the code to handle button press debouncing
4. Setting up the timer (PWM)
    a. Set up the timer in the PWM mode
    b. Map the related GPIO pins to control the intensity of LEDs
5. Write the high level application which glues everything together toward the intended program

Hints for this lab

A. There is a set of functions acting as drivers for SPI, LIS3DSH and all other peripherals that are available with your board. You can rely on them. In the tutorial, you will be shown how these drivers are to be used as well as the steps needed to adjust those drivers for your needs.

You can identify the functions and the data structure for initiation of the accelerometer. Based on the specification of the problem, you need to provide the values for the fields of that data structure to select values such as: data rate, enable axes, define the scale of the accelerometer and the update mode of the device, as well as the potential filtering of the data. Please be aware of the interplay between your readouts and the data being updated, depending on the mode that might impact the performance, as well as the big/little endian selection importance for subsequent processing.

As communication to the accelerometer needs to be done via SPI, proper initialization of SPI and sending/receiving of the packets needs to happen first, followed by the actual setting of the registers in LIS3DSH via SPI and reading of the sampled data. ***Even though you will not be directly interacting with the SPI as it is being abstracted by the accelerometer driver, you still need to know what is going under the hood. That is, the SPI protocol, connections, frame structure and so on. You will be tested on your knowledge on this topic during the demo***.

B. The angles of the board relative to the vector of gravity force are to be calculated. Please note that among three such angles (roll, pitch and yaw/heading), the last one is not detectable, as the gravity force does not depend on the yaw.

Reading and Reference Material

- Tutorial III by Ashraf Suyyagh
- C Tutorial by Ben Nahill
- Doc_05 - STM32F4xxx Reference Manual
- Doc_10 - Discovery Kit F4 Rev.C
- Doc_13 - LIS3DSH datasheet
- Doc_14 - LIS3DSH 3-axis digital output accelerometer
- Doc_15 - Tilt angle application notes
- Doc_16 - Debugging with Keil
- Doc_17 - Introduction to Keil 5.xx
- Doc_19 - Documentation of STM32F4xx Cube HAL drivers
- Doc_24 - Doxygen Manual 1.8.2 (Optional)
- Doc_25 - ProGit 2nd Edition (Optional)
- Doc_32 - Generic Keypad and Hitachi HD44780 tutorial
- Of course you are free to refer to any other posted documents which you see useful.

Demonstration

You will need to demonstrate that your program is correct, and explain in detail how you guarantee the desired precision, and how you tested your solution. Your program should feature each and every requirement from 1 to 6 listed above. You will be expected to demonstrate a functional program and its operation performs in all situations.

The final demonstration will be on Friday, March 10th

Bonuses

1. **Early demo bonuses (6th – 9th March)**: Thursday demos have a bonus of 0.25, Wednesday's 0.375, Tuesday's 0.5 and Monday demos 0.625.

Report

No report for Lab 3 as it will be merged with the report for Lab 4