ECSE426 - Microprocessor Systems

Fall 2016

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. Your goal is to design a system that (i) detects the board's tilt angles by processing readings of a tri-axial accelerometer; and (ii) detects the strength of a strike for the piezoelectric sensor. You will be using the ST accelerometer sensor LIS3DSH (the MB997C version) to read (sample) the accelerometer at a rate of 25Hz. The MEMS sensor should generate an interrupt once data is ready and you should configure the associated GPIO to raise an interrupt in the microprocessor.

You will use this core embedded design within a higher-level application. The board will be put into a certain initial orientation and the user will use a keypad to enter a desired board's tilt angle. The board is to be connected to a 3x4 alphanumeric display for the user to provide his/her input. The board will be interfaced to a 4-digit 7-segment display. Given the initial board orientation and the desired orientation, the display will visualize the direction the board should move from its current position to meets the target orientation. The visual effects are left for you to decide. Once the board is within 5^{o} of the target orientation, the user input angle is shown on the display.

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).**

Please read the class notes and the application notes by ST Microelectronics to understand how the tilt angles are measured from the projections of the gravity force. The notes also contain an explanation of the impact of various approximations (zero-*g* offset accuracy, temperature dependence etc.) and how they can be addressed. 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 the 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 MEMS chipsets are used to improve the 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. Neither can be used to determine headings accurately but magnetometers can use the earth's magnetic field to do this. 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. This has all three sensors: accelerometers, magnetometers and gyroscopes**. We also have at our disposal in ECE labs a high-end breakout board **LSM9DS1** that 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) Piezoelectric sensor

You are required to take a reading from the piezoelectric sensor. This has been discussed in the tutorials and you have been given example code illustrating how to do this using the GPIO. You will need to use a Hardware Timer in order to sample the Piezo via polling the ADC. Configure the timer to generate interrupts at the desired sampling frequency. You will need to decide what the sampling frequency for the piezo should be (**hint:** using the oscilloscope will be helpful).

(3) Sensor calibration and filtering

You are required to calibrate the sensors prior to their use in your application. You are advised to use offline calibration. That is, write the code that chooses appropriate calibration values for your board and then use those values in your implementation. This step is only to be done once. It should not be used again during application **run time**. You are required to show and describe in detail the steps conducted in deciding upon the calibration values.

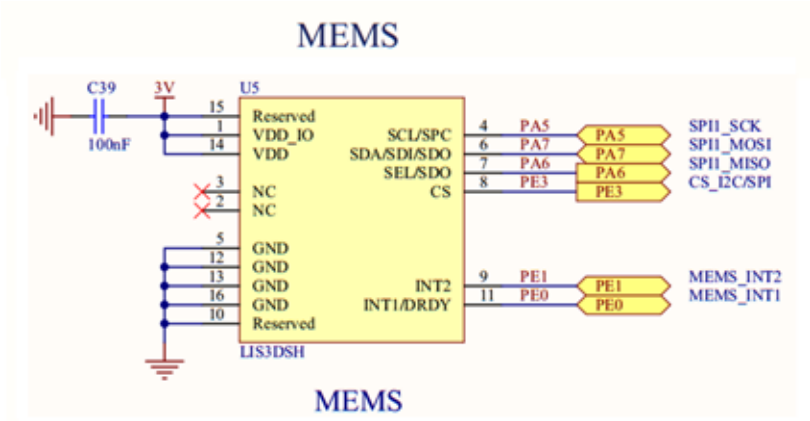
For the piezo sensor, you will need to determine the minimum force that leads to a saturation value and the minimum force that provides a measurable output. You will need to establish a method of converting from the voltage to an appropriate measure of force and you will need to determine the range of measurable forces. Normalize the final piezo output to be between 0-100, with endpoints corresponding to the minimum force values determined above.

Finally, do not forget to **filter** the sensor data using the Kalman filter (you will need to decide if one is necessary or sensible for the piezoelectric sensor). You will need to *conduct an analysis of the parameters of the filter* so as to achieve good filtering performance for the MEMS signals. This should be similar to what you did for the temperature sensor in Experiment 2. There are many ways to perform the calibration. One straightforward approach is to take averages to calculate the offset and then use this offset to calibrate future readings as described in the tutorial. Another elaborate and a more accurate method is described in **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 that new acceleration samples become available (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 are 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 the interrupt flags **both** from your processor's end and the sensor's end inside your interrupt service routine (ISR).

IMPORTANT: Note that the data ready interrupt signal in the LIS3DSH sensor is only generated on the INT1 line. This line is hard-wired to PORTE PIN0 of our processor. Since all PIN0s 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 to be sure to avoid such a conflict.



(4) The 7-segment displays and alphanumeric keypads.

You will be able to measure angles along the x and y directions. The angles measured along the x axis are called the roll angles (α) and those measured 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 can then hard code it in your code.

The user first chooses: piezo or tilt. You can assign keys to indicate this choice. You may wish to display something on the 7 segment display to provide instructions.

Piezo

The user enters a desired strength of strike in a range of 1 to 100 using the keypad. The user must follow the numerical sequence by pressing a keypad button (any non-numerical button of your choice) to simulate an ***Enter*** button.

Once the designated "Enter" button is pressed, the processor will wait to detect a strike from the piezo sensor. It will evaluate the strike and provide an indication of softer or harder back to the user via the 7-segment display. You can choose what the indication should be.

Once the strike is within ±5 of the target piezo value, the target value should be displayed on the 7-segment display.

Tilt

Once you have placed the board at a certain angle, the user provides a desired target angle using the keypad. The angle can be anywhere from 0° to 180°. The user will provide a one, two or three digit **integer** angle input through the keypad. The user must follow the numerical sequence by pressing a keypad button (any non-numerical button of your choice) to simulate an ***Enter*** button.

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

1. The target angle is larger, and thus the board needs to be tilted in an upward direction.
2. The target angle is smaller, and thus the board needs to be tilted in a downward direction.
3. The target angle is already within the 5° difference range.

For the initial two cases, you will show certain visuals on the board to direct the user towards the desired orientation. The choice of the visuals is left to you. However, you should use some simple animations. In any case, the visualization should be meaningful and helpful towards its purpose. Once the orientation of the board is within 5° from the target orientation, the user-provided angle is shown on the display. For the third case where the current and desired orientation are within the range of each other, the application should skip directly to showing the angle.

The actual measured value will be displayed on the 7-segment display in the form of XXX°, XX.Y° or X.YY° depending on the actual angle (i.e. 119°, 59.3°, 7.55°). The degrees symbol is also shown on the display. The application is reset by pressing the board reset button.

(6) Hardware Timers and 7-Segment displays

Since you have been asked to display your measured tilt angles on the 7-segment display, you will need timers in order to set up the 7-segment multiplexing operations (described in the tutorial). For this, instead of relying on software timers for switching delays, you are asked to replace them by hardware timers. Hardware timers are more efficient and lead to a more robust application.

You are required to use ST's TIM3 peripheral module as a means to generate the necessary delay. You will also need to configure the associated Nested Vector Interrupt Controller (NVIC) to set up the interrupt line and priority for this timer. Consult the driver files for steps of operation and available functions. To get the specific bus clock value which clocks TIM3 and subsequently used to derive the desired clock, refer to "**Doc_00 - STM32F4 Processor clock tree and configuration values".** The basic equation to set up your desired rate is:

Desired rate = TimerClockingFrequency / (Period × prescalar).

Make sure that your choice for the period and prescalar do not overflow the registers (16-bits). The maximum value is 65536.

A tutorial on how to use the keypads will be uploaded on myCourses. The 7-segment interfacing schematic is provided at the end of this document.

(7) Hint for the next lab

The next lab (Lab 4) will be based on Lab 2 and Lab 3 (this lab) and you will only have one week to do it. So please ensure now that all of your code is working. We will merge both labs into a larger project using real time operating system (RTOS) services.

Summary of programming requirements

1. Set up the accelerometer
    a. Set up the accelerometer configuration and sampling data rate.
    b. Set 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. Setting up the piezoelectric sensor
    a. Configure the GPIO so that you can take a reading from the piezoelectric sensor.

3. Signal processing operations:
    a. Conduct experiments needed to calibrate sensor data for all sensor axes.
    b. Filter the data through Kalman filter and investigate filter parameters.

4. Setting up the alphanumeric keypad and 7-segment display
    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. Set up a timer to count the delay time between moving into 7-segment display digits.
    c. Configure the timer to generate interrupts and configure it using the NVIC.
    d. Write the high level code which handles digit multiplexing (no visible flickering allowed).
    e. Write the code which handles visual effects to be displayed on 7-segment display.
    f. Write the alphanumeric keypad scanning algorithm.
    g. Write the code to handle button press debouncing.
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 device attributes such as: the data rate, the enabled axes, the scale of the accelerometer and the update mode of the device. You also need to decide on potential filtering of the data. Please be aware of the interplay between your readouts and the data being updated. Depending on the mode of operation this might impact the performance. The big/little endian selection may be important.

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, because it is being abstracted by the accelerometer driver, you still need to know what is going***

on under the hood. You need to understand 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 need 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.

C. A piezo strike will appear as a quick voltage pulse to the ADC. You will need to sample it fast enough to catch multiple points on this pulse, and then implement something along the lines of a peak detector to detect the highest point of the pulse.

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_31 - Clock Display Datasheet
- Doc_32 - Generic Keypad and Hitachi HD44780 tutorial

Demonstrations

Demo 1 (on or before Friday Oct. 21)

In the first demo, you will need to demonstrate the capability to:

(1) read values from the accelerometer;
(2) read values from the piezoelectric sensor;
(3) display the normalized strike reading (0-100) on the 7-segment
(4) receive and respond to input from the keypad.

Demo 2 (on or before Friday Oct. 28)

In the second demo, 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 5 listed above.

You will be expected to demonstrate a functional program and ensure that it satisfies the application requirements.

Early demo bonuses (Mon. – Thurs.): Monday 0.8; Tuesday 0.6; Wednesday 0.4; Thursday 0.2

Report

There is no report for Lab 3 as it will be merged with the report for Lab 4

Common-cathode 4-digit 7-segment interfacing