ECSE426 - Microprocessor Systems

Fall 2015

Lab 3: MEMS Accelerometer and Guess the Angle Game

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 LIS302DL (the MB997B board version) or LIS3DSH (the MB997C version) to read (sample) the accelerometer at a rate of 100Hz. The MEMS sensor should generate an interrupt once data is ready and you have to channel this signal to raise an interrupt in the microprocessor through EXTI and NVIC modules. This core embedded design will serve a higher-level application, specifically a game we will call "Guess the angle". The board will be put into a certain orientation and the user will use a keypad to enter his guess of the board's tilt angle. The board is to be connected to a 3x4 alphanumeric display for the player to enter his guess. The board will be interfaced to a 4-digit 7-segment display which will displays the player guessed input / measured tilt angles. The board LEDs will provide feedback for the user about his guess (i.e. lower, higher, success or a loss).

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 least 100 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.

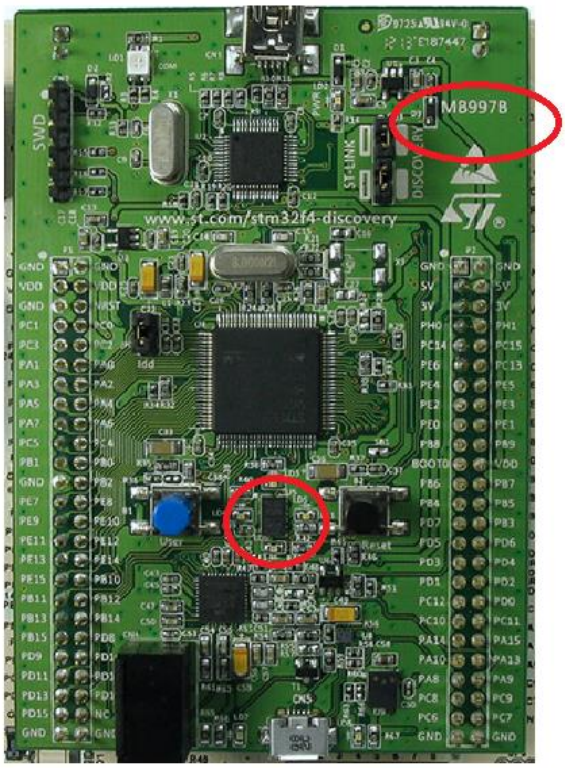
Please read the class notes and, the application notes by ST Microelectronics on how the tilt angles are measured from the projections of the gravity force. The notes also contain the explanation on how various imperfection sources (zero-*g* offset accuracy, temperature dependence etc.) get treated. 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 are not part of the standard peripheral library. You have to include them on your own (or write them from scratch as we did for the new boards). In the base project, you will see two drivers, one for the LIS302DL provided by ST, and LIS3DSH which was written by the TAs for

this new sensor). Refer to the figure below to see which MEMS sensor version you have and use the correct driver to work with your board. All required reference material for the accelerometer will be found in myCourses.

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.

LIS3DSH



LIS302DL

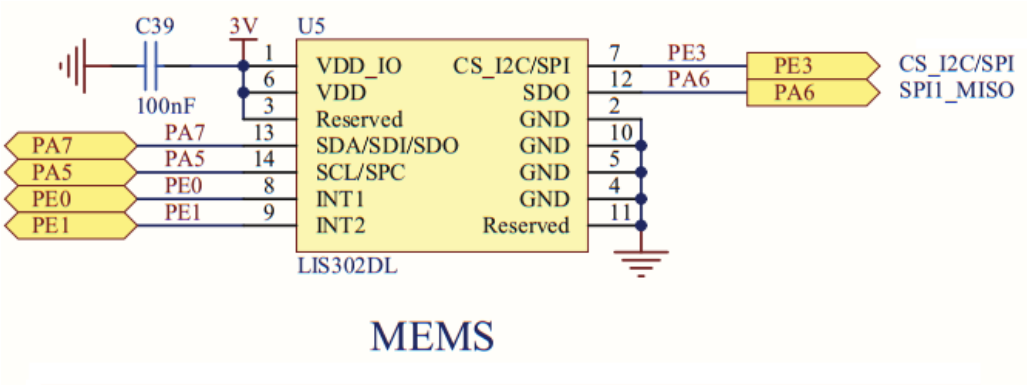
(2) Sensor Calibration and filtering

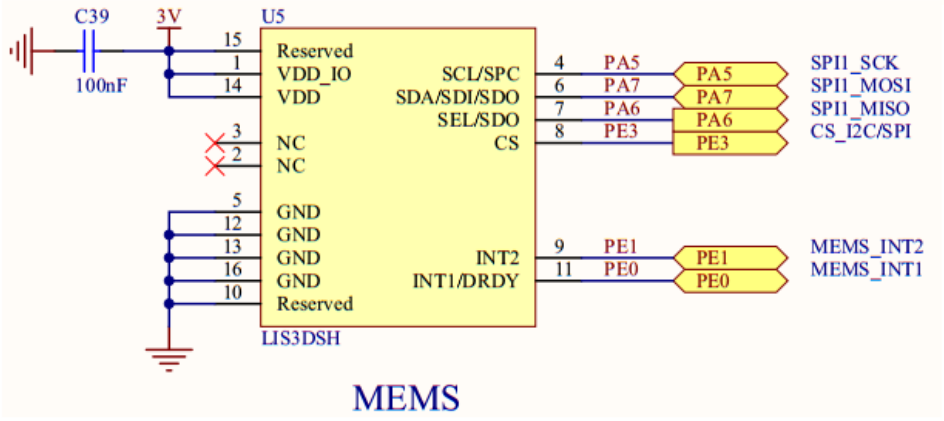
You are required to calibrate the sensor prior to their use in your application. You are advised to use offline calibration. That is, write and present the code which gets the calibration

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 moving average filter. You will have to *do a similar analysis on the depth of the filter* so as to get decent filtering for the MEMS signals similar to what you did with the temperature sensor. 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. Another elaborate and a more accurate method is described in the accelerometer document uploaded to myCourses.

(3) MEMS interrupts, EXTI 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 (100 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 by channeling it through by using the EXTI interrupt and NVIC modules. 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).





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

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 you game on in advance. You hard code it in your code.

Once you have placed the board in a certain angle, the game starts by allowing the player to enter his guess. The user has a predetermined number of trials (3 to 5) to guess the tilt angle of the board. The player will provide a two or three digit **integer** angle guess through the keypad. The user must follow the numerical sequence by pressing a keypad button (any non-numerical button of your choice) to simulate and ***Enter*** button. The input angle is shown on the 7-segment display digit-by-digit as it is being entered.

Once the designated "Enter" button is pressed, the processor will compare the player guess to the measured angle to within 4° more and less. If the guess is below the exact measured value - 4°, the blue LED will light on. If it is more than the exact measured value + 4°, the red LED will light on. The game terminates either by the player unsuccessfully using all the allowed number of trials or by correctly guessing the angle. If the player provides a correct guess, the actual measured value will be displayed on the 7-segment display in the form of XX.Y°, 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 green LED will light on in the case of success. You can choose whatever visual feedback you wish to display in the case of a loss. The game is reset by pressing the board reset button.

(6) Hardware Timers and 7-Segment displays

Since you have been asked to display your guessed/measured tilt angles on the 7-segments clock display, you will need timers in order to setup the 7-segment multiplexing operations (described in the tutorial). For this, instead of relying on software timers for switching delays, you are asked to them by hardware timers, Hardware timers are more efficient and allow for robust responsive overall application.

You are required to use either ST's TIM3 or TIM8 peripheral module as means to generate the necessary delay. You will also need to configure the associated Nested Vector Interrupt Controller (NVIC) to set up the interrupt 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, you might wish to consult the RCC drivers. The basic equation to set up your desired rate is TimerClockingFrequency / (Period × prescalar) = desired rate. Make sure that your choice for the period and prescalar do not overflow the registers.

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

(7) 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.

Hints for this lab

A. There is a set of functions acting as drivers for SPI, LIS302DL/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 LIS302DL/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.

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, October 30th.

Report

Your technical report should comply with the guidelines posted on myCourses. Always refer to the notes on your previous report submissions to avoid making the same mistakes and

losing marks over them. The report is due on Monday Nov. 2nd. Late submission penalties apply.

Bonuses

1. **Early demo bonuses**: Demos taking place on Wednesday, Oct. 28th and Thursday Oct. 29th will be awarded 0.25 points. Demos taking place on and earlier than Oct. 27th will be awarded 0.5 points.
2. **Bonus (driver – 0.75 points)**: *You are highly encouraged to do this bonus as it constitutes a **large component of the project**. Starting **early** on this part gives you three more weeks before the actual project start date to work on an integral part. Even if you do not get this part working for now, at least you will have saved much precious time off your limited project duration. Best case, you will be done with an integral component and get bonus for that. Worst case, you have done some work of the project and debugged many issues that could delay your final submission.*

For this part, you are required to pick up the 9DOF MEMS extension board LSM9DS1 from ECE labs counter. The LSM9DS1 is a sensor package encompassing an accelerometer, gyroscope, magnetometer and temperature sensors. The packaged accelerometer is quite an advanced model with high-end specs similar to the LIS3DSH.

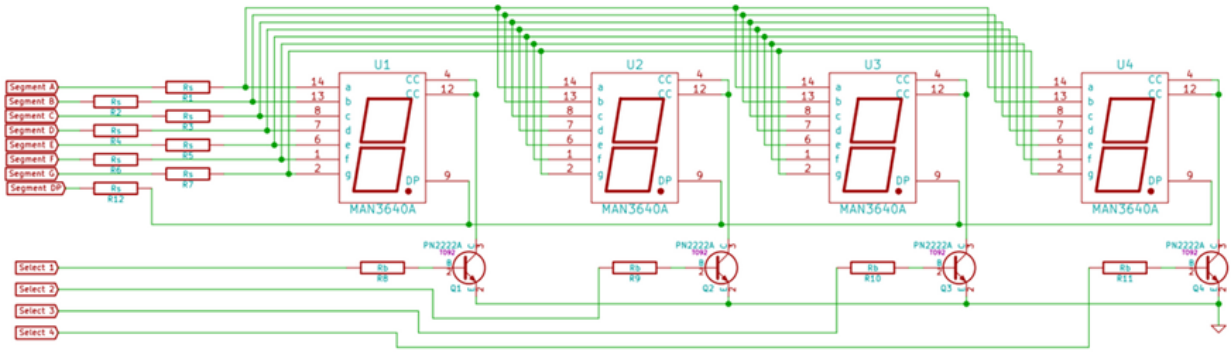
In this bonus, you are required to re-base the lab to use this external accelerometer board instead of the on-board MEMS sensor. Your duties will be to interface the extension board to the main board using the **SPI** interface, write two driver files (header and C file) for the LSM9DS1 chipset, and make your code call your functions to read data from the new sensor.

Hints:

1. You should study how the on-board LIS3DSH drivers are written (since it is closer in specs). Your code will be very similar to them in terms of defining addresses, options, configuring SPI, and your function implementations. You might even reuse many of the code base if no changes are necessary, or have little modifications for some parts.
2. The on-board sensor uses SPI 1 of the processor SoC. You have the option to use SPI 2 or SPI 3 to interface the extension board or you can reuse SPI 1 by changing the Chip Select (CS) pin to point to the GPIO interfaced to the external sensor.
3. Do not forget to configure/modify the GPIOs to be used with the SPI module you use to the ones interfaced to the extension board. In this case, some will be in alternate function mode (for SPI operation) and others as output (chip select).
4. The new chipset offers advanced features like tap detection among others that do not need. Only write a driver to implement the basic functionality of configuring the accelerometer, reading and writing to its internal control registers and accessing the acceleration data. Ignore any features that are remotely different from those of our current drivers. Include all options that are necessary for a working driver. We will leave which ones to include to your better judgment and understanding of the datasheet.
5. One main difference is that this sensor has buffered output that could store multiple values of X, Y and Z axes at a time. There are few modes of operation like FIFO,

continuous and bypass modes among other. You should only implement one of those options. Choose the one that is easiest for you yet logical from an operational point of view. If there is one that does not use the FIFO, go for it as it would be straightforward to code.

6. In your header file, you need to define register addresses as well as values to the parameters to be passed to the functions in a similar fashion to the drivers in hand. Similarly, you need to define at least one struct for configuration options. Finally, you are to define the function prototypes that you will call in your program.
7. In your C file, you need to implement the functions defined in the header files. For the most part, you might be able to reuse the implementations of previous drivers. You have to compare both datasheets to each other and how the implementation of one driver will be mapped to the other.
8. Finally, the project will further use the gyroscope sensor. Even though this bonus does not include the gyroscope, if you could start working on the gyro drivers as well, you will save yourself much needed time. Most of the time, both the gyro and accelerometer share same configurations registers and their options, so it would not be much of a trouble to do so.
9. You need to **redo the calibration and filtering analysis** for the new extension board
10. You are **not** required to write codes for the magnetometer or temperature sensors.



Common-cathode 4-digit 7-segment interfacing