ECSE426 - Microprocessor Systems

Fall 2013

Lab 3: 3D Tilt Angle Measurement with Accelerometer

Lab 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 board's tilt angles by processing three readings of one dimensional accelerometers (that is; a single tri-axial accelerometer). You will be using ST accelerometer sensor LIS302DL. You will be reading (sampling) the accelerometer at a rate of 25Hz. Instead of using SysTick, you will be sampling at a rate set by a hardware timer peripheral. Similar to SysTick in lab 2, you will have to set up a timer interrupt-driven routine to control the sampling rate. You will also control the speed at which LEDs are flashing based on tilt range. Finally, you need to use the tap feature of the accelerometer to switch to PWM mode. Though this time, PWM functionality is achieved by setting it up through hardware timers instead of simulating it in software.

Lab requirements

(1) Tilt detection

For F4-Discovery board, you are required to design a system that calculates tilt angles in 3 dimensions with 4^{o} accuracy. The system should be calculating angles in real time, at least 25 times per second. During the demo, we will use triangular wooden blocks cut to specific angles to test for accuracy.

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, but 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). Luckily for you, ST has the drivers already written for you.

All required reference material for the accelerometer will be found in myCourses

(2) 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 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 one time and is not then to be used 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.

(3) Timers and interrupts

You are required to use ST's TIM4 peripheral module as means to generate the 25 Hz sampling clock for the device. You will also need to configure the 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 TIM4 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.

(4) Tapping feature and external interrupts

The LIS302DL you will be programming has a feature to detect single and double taps. You are required to configure the accelerometer to detect single taps and use them to switch between normal operation and PWM state. The difference now, is that such a signal is external to the microprocessor for the MEMS chipset is a different chipset and not an inside peripheral. For this, you will have to channel the signal through the accelerometer interrupt lines to the GPIO (consult the board schematics to see how is the MEMS hardwired to the microprocessor). You need to configure the GPIO, EXTI and NVIC features such that this detected signal acts as an external interrupt. Kindly read the application note on tap detection provided.

(5) Hardware PWM

Similar to what you did in lab 2, you are going to generate a PWM signal but this time in hardware. There are two ways to achieve this:

1. The simplest is by replacing your software timers in lab 2 by a hardware timer peripheral to do the same thing. You can associate interrupts to toggle the state of the LEDs and control the gradual change in LED brightness.
2. **(Bonus of 0.75 Marks)** The use of timers in PWM mode and OC (output compare) channels to generate the desired signal. In fact, the set of examples available with the ST Discovery board has a complete project on how to do this (Project TIM_PWM_Output). You can download the example files from this link http://www.st.com/web/en/catalog/tools/PF257904.

   Even though the example generates 4 different PWM signals at different duty cycles to control 4 different LEDs; the concept, procedure and flow of how to achieve this is similar to our case. The difference is that we are controlling all 4 LEDs simultaneously at varying duty cycles. You can study the example code, and modify it to achieve the lab requirements. For any technical terms which you might not be familiar with, you are advised to read the STM32F4xxx datasheet sections to understand basic functionality. Also you have to consult the drivers' documentation to see the list of available

commands, the required initialization sequence and link what you have read in the datasheet to the abstractions presented in the driver file.

(6) Calculating and observing measured data in real time (Angle Values and LEDs)

Since the sensor measurements change over time, you are expected to demonstrate the measurement results in real time, as per above specification, using the SWD debug interface. In addition to using the SWD debug interface, you should be ready to provide the evidence that your solution meets the specification.

You are required to divide the available angle space into regions (say divide the 0 - 180˚ to four or six regions or any reasonable choice). As the tilt angle goes up, the flashing frequency of the LEDs increases. This is true for both roll and pitch angles. You are also free to choose any different LED effects based on tilt angles. Or have an extra case for dealing with corner cases (like both roll and pitch increases simultaneously … etc.)

Hints

A. There is a set of functions acting as drivers for SPI, LIS302DL 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 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. **Even though this is a group project and you will need to divide tasks in between group members, it is expected that each student will understand and present in detail any part of the lab**

The final demonstration will be on Friday, Nov 1st. Exact demo time will be announced later.

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.