MCGILL UNIVERSITY

ECSE-426

Lab 3: 3D Tilt Angle Measurement with Accelerometer (Group 11)

Tanuj Sharma (260367436), Benjamin Swearingen (260398370)

11/4/2013

Table of Contents

Contents

Abstract

This lab report describes the purpose, functionality, testing and conclusions of the third lab. The goal of this lab was to use the STM32F4DISCOVERY board's accelerometer sensor to design a system that detects board's tilt angles by processing three readings of one dimensional accelerometers. The accelerometer is set to a sampling rate of 25Hz with the use of a hardware timer peripheral to control the sampling rate. The brightness of the LEDs are based on the current pitch and roll. Finally, a tap feature is implemented on the accelerometer to switch to PWM mode using hardware timers. At the end of this lab, the system was able to communicate tilt angle changes, the direction of the change and tap detection through the LEDs.

Problem Statement

Several design questions had to be addressed for the efficient measurement of the tilt angle using accelerometer and LED display functionality:

- The first task involved the configuring and initializing of the accelerometer with correct settings to give readings of tilt angles. The system designed, must calculate tilt angles in 3 dimensions with an accuracy of 4 degrees. These readings are then converted to zero-g format.
- The system must be calibrated prior to the use in application. This is achieved by doing a six point calibration using least squares. The offset values are added to initial sensor values converted to zero-g to accurately calibrate the accelerometer. The data should then be filtered using a moving average filter and analysis is done in MATLAB on the depth of filter size. This is explained more in detail in the section "Theory and Hypothesis".
- The ST's TIM4 peripheral module is used to generate the 25Hz sampling clock. This is achieved by configuring the Nested Vector Interrupt Controller (NVIC) to set up the

interrupt for the specific time. The choice of period and pre-scalar must be designed such that the registers must not overflow.

- The system has a feature to detect single and double taps by programming the LIS302DL. The accelerometer is to be configured to detect single taps and use them to switch between normal operation and PWM state. The taps are used generate interrupt and switch mode. This means the signal will have to be channeled through the accelerometer interrupt lines to the GPIO. GPIO, EXTI, and NVIC will need to be configured so that this detected signal acts as an interrupt.

- Furthermore, a PWM signal must be generated using hardware. This is to be done with the use of timers in PWM mode and OC (output compare) channels to generate the desired signal. The system designed must be able to control all four LEDs simultaneously at varying duty cycles.

- Finally, the system must be able to calculate and measure observed data in real time. This is done using the SWD debug interface. In addition, the system must be able to provide evidence that the solution meets the specification. The LEDs function is based on pitch and roll rather than discrete regions divided.

Theory and Hypothesis

To measure the 3D tilt angle of the accelerometer we must be able to measure the pitch and roll of the processor body.

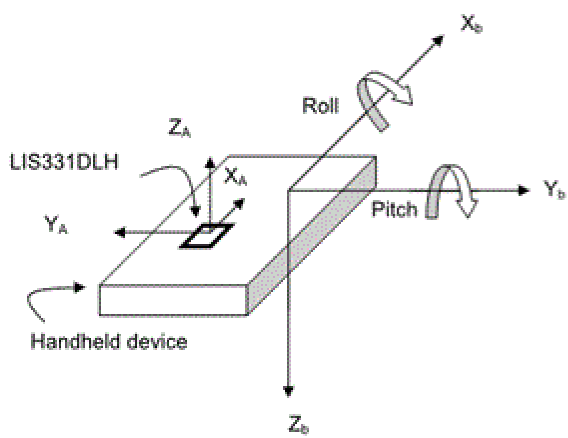


Fig. 1 – Accelerometer Axes

From Fig.1, the Xb, Yb and Zb are the device body axes with a forward-right-down configuration. XA, YA and ZA are the accelerometer sensing axes, respectively. The Pitch and Roll angles are referenced to the local horizontal plane.

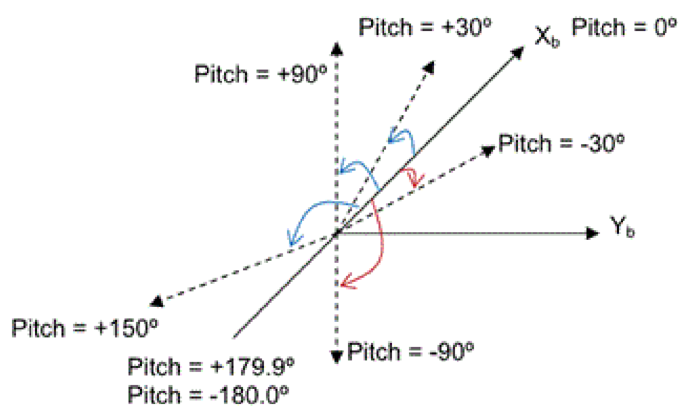


Fig. 2 – Pitch Measurement

The Pitch is defined as the angle between the Xb axis and the horizontal plane. For example, from Fig. 2 we see that Yb is fixed, Xb is rotating from Pitch = 0° to +30°, +90°, +150° and +179.9° for a positive direction.

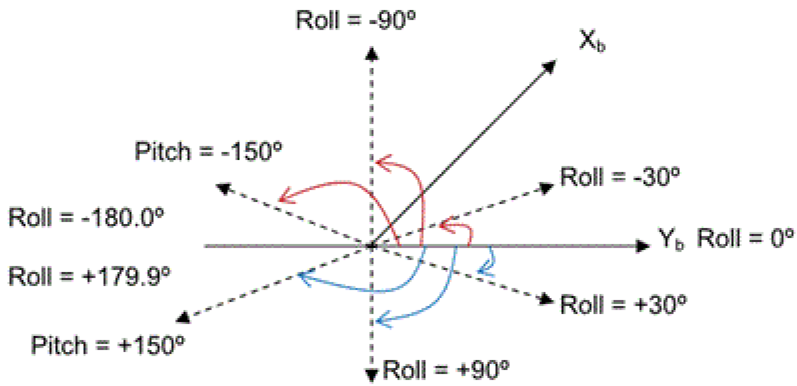


Fig. 3 – Roll measurement

The Roll is defined as the angle between the Yb axis and the horizontal plane. For example, from Fig. 3 we see that Xb is fixed, Yb is rotating from roll = 0° to +30°, +90°, +150° and +179.9° for a positive direction.

The Pitch and Roll angles are measured using the following equations:

$$pitch = \arctan\left(\frac{A_x}{\sqrt{A_y^2 + A_z^2}}\right)$$

Equation: 1 – Pitch equation

$$roll = \arctan\left(\frac{A_y}{\sqrt{A_x^2 + A_z^2}}\right)$$

Equation: 2 – Roll equation

Table: 1- raw sensor data at 6 different stationary positions

Stationary position	Accelerometer (signed integer)		
	A_x	A_y	A_z
Z_b down	0	0	+1 *g*
Z_b up	0	0	-1 *g*
Y_b down	0	+1 *g*	0
Y_b up	0	-1 *g*	0
X_b down	+1 *g*	0	0
X_b up	-1 *g*	0	0

The accelerometer is then calibrated using a six point calibration method by least squares method. Table 1 shows the sign definition of the raw sensor data at 6 stationary positions with respect to the known Earth gravity vector. For example, in fig. 1 Xb and Yb are level and Zb is pointing down. Therefore, Ax = Ay = 0, Az = +1 g.

The readings of raw measurements are taken at 6 different positions Zb down, Zb up, Yb down, Yb up, Xb down, Xb up respectively. These measurements are then taken in a matrix form with a 1 at the last column to get the desired offset value. The following equation is used to define the matrix X whose values are used to calculate the offset.

$$X = [w^T \cdot w]^{-1} \cdot w^t \cdot Y$$

Equation. 3 – A 12 point calibration matrix equation

Where,

- Matrix Y is the known Earth gravity vector.
- Matrix w is the sensor data LSBs collected at all the six stationary positions.
- Matrix X is the 12 calibration parameters that need to be determined.

The calculations were performed in MATLAB. See “Observation and Testing” for the results of sensor calibration.

Furthermore, The ST’s TIM4 peripheral module is used to generate the 25Hz-sampling clock. This is obtained from the following equation of the desired rate:

$$Rate_{desired} = \frac{TimerClockingFrequency}{Period * prescaler}$$

Equation: 4 – Desired rate equation

Implementation

The first step in this lab was the configuration and initialization of the accelerometer. Configuration and initialization occurs in the ACCEL_Config method. The power mode is set to LOWPOWERMODE_ACTIVE. LOWPOWERMODE_SHUTDOWN was the only other option for a power mode, so the ACTIVE mode was used. The data rate is set to 100Hz rather than 400Hz. As the accelerometer is only read at 25Hz, there is no need for a 400Hz data rate. The X, Y, and Z-axes are enabled. The scale is set to 2_3. In this mode the accelerometer can sense changes of ±2 gravities. In tilt measurement there will be no accelerations greater than ±1 after calibration. Self_Test is set to normal, which means that self-test is not occurring. Self-testing involves the accelerometer applying a known force to the sensors, and then measuring the deflections. This would block the acceleration of gravity that was being measured.

After configuring the accelerometer, data was read from it. Reading starts at register 0x29, OUT_X, and continues for 4 more registers. Registers separate OUT_X, OUT_Y, and OUT_Z, so 5 registers in total are read. The results are put into an array. The numbers output from the accelerometer are how many 18 milligravities are being applied to the axis. As such, the output values are multiplied by 18 to transform them into miligravities. These values are then passed into Equations 1 and 2 to determine pitch and roll.

To ensure that data was being read from the accelerometer at the correct rate, a hardware timer (TIM2) was configured to generate an interrupt with a frequency of 25Hz. The timer itself was configured with the following equations.

$$Prescaler = \frac{TIM_2}{Desired\ Hz_1} - 1$$

$$TIM_2 = 2 * APB1\ Clock$$

$$APB1\ Clock = \frac{SystemCoreClock}{4}$$

$$Prescaler = \frac{\frac{SystemCoreClock}{2}}{1000000Hz} - 1$$

Equation. 5 – Equation and derivation for determining the prescaler.

$$Period = \frac{Desired\ Hz_1}{Desired\ Hz_{Final}} - 1$$

$$Period = \frac{1000000Hz}{25Hz} - 1$$

Equation. 6 - Equations for determining the period.

There is no clock division occurring, so TIM_ClockDivision is set to 0. The counter mode is set to Up, but counting down works as well.

After configuring the timer, the Nested Vector Interrupt Controller was configured to allow the timer to generate an interrupt. This interrupt is given a priority of 0, and a subpriority of 1, which allows any (0, 0) interrupt to preempt it. In the interrupt, the tick bit is set to 1, which keeps the length of the interrupt to a minimum.

Calibration was implemented using the six point calibration method by the least square method. The offset values calculated and added to the raw sensor values. This was done externally and the offset values were passed to the program. A brief description of the calibration procedure is given under the section “Theory and Hypothesis” and a detail of the results obtained in MATLAB is under the section entitled “Observations and Results”.

With the accelerometer properly calibrated, the X, Y, and Z outputs were put through individual filters to remove any noise. For a detailed analysis of the filters, see “Lab 2 Report.pdf.” The filter depths will be discussed in the “Observation and Testing” section of this report.

To successfully detect taps, the accelerometer’s interrupt was configured. It was set to generate an interrupt on a single click on the Z-axis. All double click axes were disabled. The Latch_Request was set to Latched. This means that the CLICK_SRC_REG remains high after a click until it is read. After this, it is unlatched and returns to low. CTRL_REG3 is set to 0x7, or 0111, a value which is taken from a truth table on page 9 of the “LIS302DL 3-axis digital MEMS accelerometer translates finger taps into actions” paper. The Z threshold is set to 0x05, which is in increments of 0.5g, so the threshold is 2.5g. The maximum tap length is set to 0ms. Higher values resulted in behavior similar to bouncing on a switch.

With the accelerometer configured, the interrupt was configured. As it occurs on the accelerometer, it is an external interrupt. It is set to occur on line 0, and to trigger on the rising edge. The NVIC was then configured with a priority of (0, 1). The CLICK_SRC_REG is then read to ensure that it is in the low position, and not latched high. On a tap, the integer mode is toggled, to signal a change in mode.

To implement PWM through hardware rather than software, TIM4 was configured with a similar method to TIM2, but with a frequency of 1.5 KHz. Then Output Compare was configured. It was set to be in mode PWM1 with output state enabled, and high polarity. All four channels are then enabled. In the main method, when PWM is running the CCR values of TIM4 are updated every 10000 cycles. This is done without spin locking, as that should be avoided. It is a linear increase and decrease from 0 to 1500 and back down to 0 again. The number of cycles off is increased or decreased by one every update.

The last portion of the project that was implemented was the LED angle display. The lab description said that the LED toggle rate should be controlled by the angle, but any effect could be used so long as the angle was somehow displayed. Rather than using blinking, the already implemented PWM was modified so that when in the display mode the CCR values are updated based on the pitch and roll values.

$$CCRN = \begin{cases} (0.43033 * pitch)^2 & if\ pitch < 0 \\ 0 & if\ pitch > 0 \end{cases}$$

$$CCRS = \begin{cases} (0.43033 * pitch)^2 & if\ pitch > 0 \\ 0 & if\ pitch < 0 \end{cases}$$

$$CCRE = \begin{cases} (0.43033 * roll)^2 & if\ roll > 0 \\ 0 & if\ roll < 0 \end{cases}$$

$$CCRW = \begin{cases} (0.43033 * roll)^2 & if\ roll < 0 \\ 0 & if\ roll > 0 \end{cases}$$

Equation. 7 – CCR value equations

These equations result in changes to the LED brightness when the pitch and roll change. As the equations are nonlinear, the resulting changes in the LED brightness appear to be linear. There are no real corner cases in this implementation, as the CCR values are recomputed each cycle.

Observations and Testing

The calibration is implemented using the least squares method. The calculation is performed in MATLAB and is shown below:

```
>> y=[0, 0, 1; 0, 0, -1; 0, 1, 0; 0, -1, 0; 1, 0, 0; -1, 0, 0]

y =

     0     0     1
     0     0    -1
     0     1     0
     0    -1     0
     1     0     0
    -1     0     0
```

Fig. 4 – Matrix Y

From Fig. 4 we can see that the Y matrix consists of the known normalized Earth gravity vector

```
>> w=[-21,-38,994,1; -4,-3,-972,1; -19,949,3,1; -19, -1026, 15,1; 961,7,24,1; 993,-24,12,1]

w =

   -21     -38     994     1
    -4      -3    -972     1
   -19     949       3     1
   -19   -1026      15     1
   961       7      24     1
   993     -24      12     1
```

Fig. 5 – Matrix w

From Fig. 5 we can see that the Matrix w is sensor raw data LSBs collected at all the six stationary positions.

```
>> X=inv(w'*w)*w'*y

X =

   -0.0000   -0.0000   -0.0000
    0.0000    0.0010    0.0000
    0.0000    0.0000    0.0010
    0.0081    0.0293   -0.0102
```

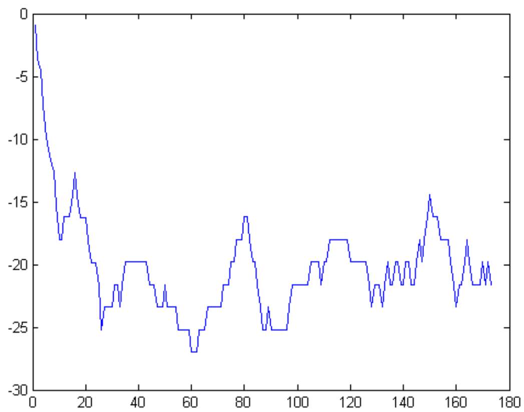
Fig. 6 – Matrix X

From Fig. 6 we can see that the Matrix X is the 12 calibration parameters that need to be determined

This Matrix X is determined using the equation 3 (section "Theory and Observations")

As seen from Fig. 6 the last row to the X matrix gives us the offset values. These offset values are then added with the raw sensor measurements at zero-g to give us the calibrated sensor values.

Most of the filter testing occurred in Lab 2, however the depth of the filter was reevaluated in this lab. With the LED display being used to show the values of pitch and roll, filter response was much more apparent in this lab. With that in mind, a depth of 13 was chosen as it applies an acceptable amount of smoothing to the data, but doesn't result in an unnecessarily long response time.



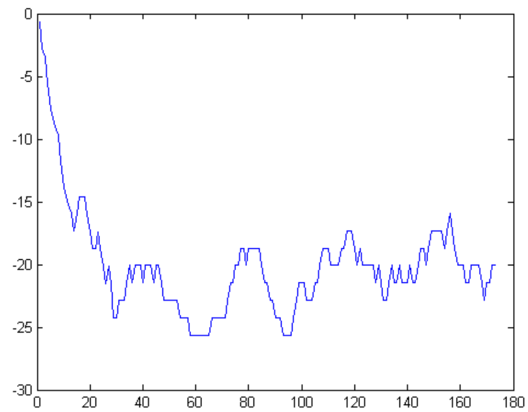
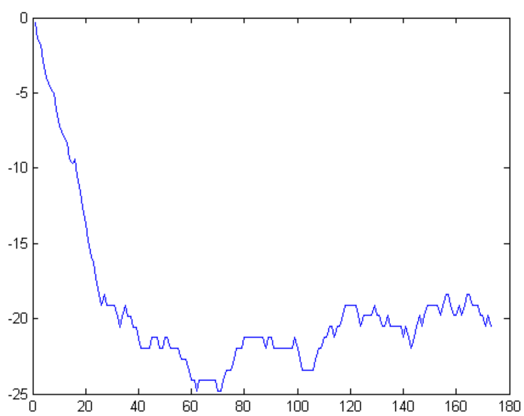


Fig. 7 -Window size of 10

Fig. 8 - Window size of 13



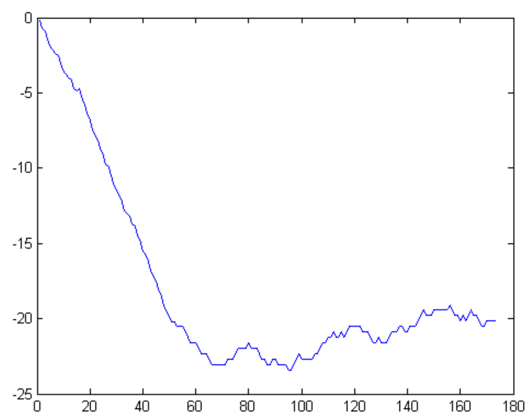


Fig. 9- Window size of 25

Fig. 10 - Window size of 50

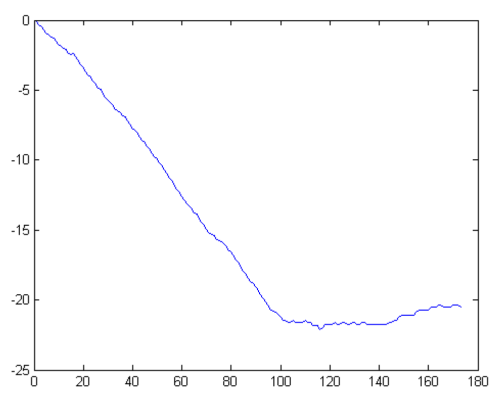


Fig. 11- Window size of 100

Angle testing was difficult as there was no access known angles other than 90 degrees. Testing consisted of testing all possible ±90 degree positions, which resulted in correct angles. Testing other angles by eye showed both the correct angles, and smooth increases from 0 degrees to ± 90. This was enough testing to ensure that the correct values for pitch and roll were being determined.

At first the single tap detection had no debouncing occurring. After successfully triggering the tap interrupt (several times with the same tap), a time-out was put into place to act as a simple debouncer. This was not done with a busy loop, so no spin locking occurs. With the switch debounced, the threshold was fine-tuned from an initial state that was reading Z-axis taps when the board was moved across the slightly textured lab bench to a more reasonable level. The maximum tap length was initially set to the current value, but increased levels were tested. They merely undid the work of the debouncer, and caused bouncing over long periods of time. As this was not a desired quality, the value was reset.

As PWM testing occurred in Lab 2, very little testing occurred in this lab. With the initial implementation of a linearly increasing and decreasing CCR value effectively demonstrating PWM, even if it was an ugly PWM, the choice was made to move on to the rest of the lab rather than spending time making a purely aesthetical change.

Although originally not intended to act as any sort of testing for PWM, the testing of the LED display did end up as a sort of secondary test of the PWM system. Using the PWM functionality to display pitch and roll data meant that the linearly increasing CCR from the PWM mode could not be used, as it was impossible to differentiate levels of brightness at the highest levels. This led to equation 7 being used instead.

Timer testing was the most problematic section of the lab. The initial intent was to toggle a GPIO pin in TIM2's interrupt, which should have shown something resembling a 25Hz square wave. This was not achieved. The initial testing used the wrong ground pin, and resulted in a non-functioning board. After obtaining a new board, a second round of testing was performed. The same idea was used, but a different ground pin was used. While this round of testing did not end with a non-functioning board, all pins that were read from showed the same values. Reading from the toggling pin, non-toggling pins, the pulsing LEDs, and even the fully lit system LEDs all showed the same results. As such, there is no empirical evidence that TIM2 was operating at 25Hz. As the configuration equations for TIM2 came from an example project, it is assumed that it is correctly operating at a frequency of 25Hz.

Conclusion

The experiment involved in measuring the 3D tilt angle of the processor by processing three readings of a one dimensional accelerometer. The accelerometer is designed to sample readings at a rate of 25Hz. This is implemented using a timer interrupt-driven routine to control the sampling rate. The LEDs are monitored to control the speed of flashing based on the change of tilt range. A tap feature is implemented in the accelerometer to switch to PWM mode and back. This was achieved by setting the PWM functionality through hardware timers. These observations and results are supported by various figures that are contained in the section titled “Observations and Testing”.

The end result of this experiment was a successfully working device that communicated tilt angle changes to the user and had a second mode that showed the capabilities of the LEDs. As we did not use the discrete angle regions, the 3D tilt angle can still be measured using the information from the Pitch and Roll as opposed to discrete regions where the regions have to be manually defined beforehand.

Appendix

- "Lab 2 Report.pdf," available on myCourses
- "LIS302DL 30axis digital MEMS accelerometer translates finger taps into actions," http://www.st.com/web/en/resource/technical/document/application_note/CD00163557.pdf
- "Tilt measurement using a low-g 3-axis accelerometer,"http://www.st.com/web/en/resource/technical/document/application_note/CD00268887.pdf