Lab 3: 3D Tilt Angle and Tap Detection using Accelerometer

Contents

Abstract

This project involves the construction of a system that calculates tilt angles in 3 dimensions with 4 degree accuracy on the STM32F4DISCOVERY processor. Using the ST LIS302DL and the on-board digital accelerometer, the acceleration is detected at 100 Hz. By measuring the static acceleration due to gravity, we can find out the angle the device is tilted at. The system is able to detect a single tap and turns on/off the 4 user LEDs at each tap. The accelerometer is calibrated to operate on any inclined surface, and the data is filtered using a moving average filter before being displayed on the 4 LEDs.

1 Problem Statement

We were required to develop a program with two major functions: angle measuring and tap detection.

1.1 Angle measuring

Using the acceleration sensors, we were required to determine the current pitch and roll of the discovery board. It isn't possible to measure the yaw while restricting oneself to the accelerometer sensor values because of the nature of trigonometry (the surface created by any sweep of the yaw angle is normal to the gravity vector [ELABORATE]). The accelerometer sensors had to be calibrated off-line and during program execution, their values were to be piped through a moving average filter. The error in the calculated angles wasn't allowed to exceed 4 [degrees]. These angles had to be measured and printed to the debug screen at least 100 times per second. A hardware timer (TIM3) was to be responsible for ensuring this timing. It had to be set to the desired frequency (over 100Hz) and was to trigger an interupt which starts the angle computation.

1.2 Tap detection

The accelerometer has built in capability to detect "taps". A tap is when the device reads a sharp and transient increase in acceleration. This can be acheived by tapping the device with one's finger. We were required to configure the accelerometer to trigger interupts upon detecting a tap. The state of the LEDs (Off or On) was to be toggled when the microprocessor receives this interupt. This interupt was to have a lower priority than the TIM3 interupt. All LEDs shared the same state so a tap would turn them all on, and a subsequent tap would turn them all off.

> Students should not be tapping the board like 10 times in order to make it respond
>
> — Anonymous (Probably Ashraf) *Microprocessors course news feed*

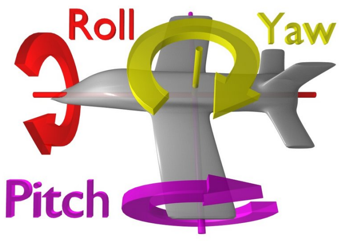


Figure 1: Yaw, Pitch and Roll

2 Theory and Hypothesis

2.1 Theory

Looking at the theory behind the conversion between acceleration and the tilt angles:

Tilt angles are used to describe how an object is oriented in 3D space relative to a global XYZ frame. The XY plane is horizontal to the earth, and the Z plane is perpendicular to it. When rotated around this frame, the accelerometer changes angles with respect to the coordinate system. We will refer to these angles as roll (between the X axis and the horizontal plane), pitch (between the Y axis and the horizontal plane) and yaw (with respect to the Z axis). Figure 1 shows these angles. Note that yaw cannot be detected because gravity force does not depend on it.

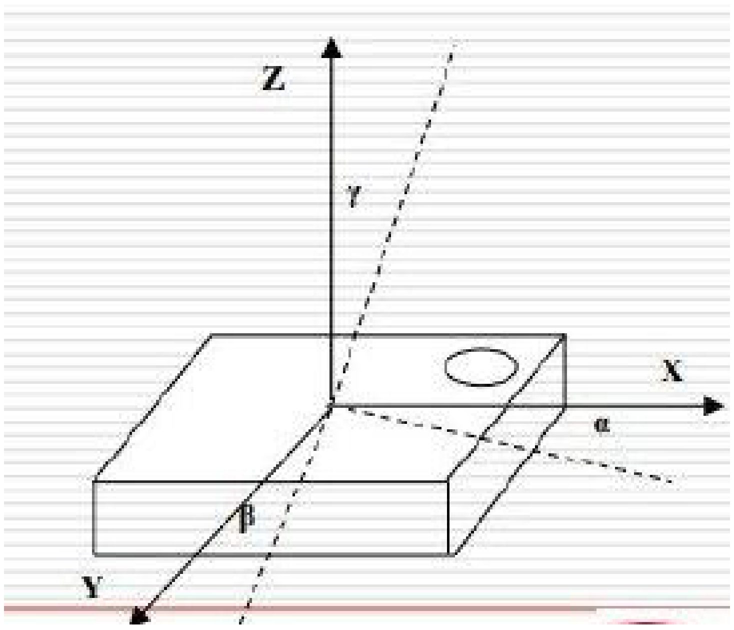


Figure 2: Yaw (beta), Pitch (gamma) and Roll (alpha)

The accelerometer returns a digital value representing static forces caused by a constant force like gravity, or a dynamic one like moving orvibrating the accelerometer [1]. By measuring the static acceleration due to gravity, we can find out the angle of the device. Equations 1 and 2 allow us to convert the acceleration detected in each direction into a tilt angle.

$$\alpha = \arctan\left(\frac{a_X}{\sqrt{(a_Y)^2 + (a_Z)^2}}\right)$$

Figure 3: Computing roll form accelerations

$$\beta = \arctan\left(\frac{a_Y}{\sqrt{(a_X)^2 + (a_Z)^2}}\right)$$

Figure 4: Computing pitch form accelerations

Equation 1 and 2 return angles between 0 and 90 degrees. In order to obtain a range of 0 to 180 and 0 to -180, the formulas in Figures 2 and 3 were used depending on the sign of the X Y and X accelerations.

2.1.1 Accelerometer

The ST LIS302DL 3-axis digital accelerometer is used in this experiment. The measured acceleration is provided to the user through I2C/SPI serial interface.

Parameters The ST LIS302DL 3-axis digital accelerometer is used in this experiment. The measured acceleration is provided to the user through I2C/SPI serial interface.

The data rates at which acceleration samples can be produced include 100Hz and 400 Hz. For the purpose of this lab, a 100 Hz data rate was chosen. All axes were configured since all 3 accelerations are needed to determine the tilt angles. The LIS302DL has dynamically user selectable full scale measurement range of +-2.3g and +-9.2g. We chose a +-2.3g range since we are only concerned with static acceleration. Moreover, a lower range is associated with more accurate readings. A self-test capability allows the user to check the functioning of the sensor without moving it. In this lab, we turn off this functionality.

Calibration Because the sensor readings aren't perfect, calibration is required to correct these deviations. In this experiment, we apply an off-line calibration technique where the process of calibrating the parameters is done only once (not every time the board is powered on). Calibration is done by putting the board in three different directions and taking a sequence of measurements in each direction.

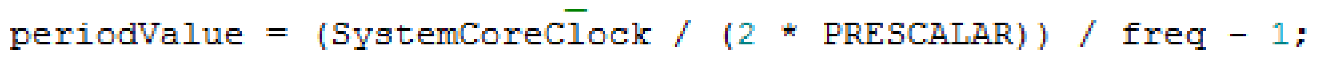
We start by placing the board in the direction orthogonal to the horizontal plane (see Figure 2 a), and take a sequence of measurements in that direction. The offset is the value which centers the reading in in each direction. After applying the offset, we get two values centered around zero, but with non-unity magnitude. A scalar multiplier is used to correct for this. We call this value the scalar multiplier. This calibration method is used for every direction, and stored to reapply to each sample.

Reading The LIS302DL_ReadACC function is used to read the outputs of the accelerometer. The function reads the LIS302DL output register, calculates the acceleration and writes the value to a 3 value array passed as a parameter. The first cell of the array holds the X acceleration, the second the Y acceleration, and the last one the Z acceleration.

2.1.2 Timer

In this experiment, we use the general purpose internal hardware timer TIM3 to implement an interval timer. This timer is used to control the interval of the accelerometer readings.

Parameters The timer's initializing structure TIM_TimeBaseInitTypeDef must be created in order to modify the parameters of the timer. This structure can be found in the stm32f4 library. The timer has its own clock that must be enabled before being used. The structure allows us to choose a prescaler value (integer clock divider to reduce the tick rate of the timer) as well as a reload value (timer generates an event upon hitting zero and reloads some initial value). Using the initializing structure, we can select the period of the timer. The period was calculated by dividing the APB1 clock by a prescaler and the desired frequency of the timer as seen in equation 3.



```
periodValue = (SystemCoreClock / (2 * PRESCALAR)) / freq - 1;
```

Figure 5: period computation

The prescalar and period specified as parameters must be between 0x0000 and 0xFFFF. Because of this, we chose our prescalar to be 42,000. The APB1 clock operates at a frequency of about 42Mhz [3] giving us a value of 1000 for the counter. This means that one second is equivalent to when the counter counts down from 1000 to zero. We then divide 1000 by the desired frequency. The frequency specified for this lab is 100 Hz, so dividing 1000 by 100 gives us 10. This means that the counter must count down to one hundredth of a 1000at each tick. We finally subtract 1 from this value to . . .

2.1.3 Interrupts

The TIM update Interrupt source is checked to determine when the down counter reaches zero. If the bit status is set (meaning an interrupt has occurred), the interrupt pending bit is cleared so that the system is ready to wait for the triggering of another interrupt rather than staying in the interrupted state.

Tap detection The accelerometer is also used to trigger an interrupt when the board is tapped.

2.2 Hypothesis

The accelerometer is noisy due to inaccuracies introduced by design limitations, manufacturing technologies, external noise and other sources of error. Although we will be calibrating the sensor through off-line calibration as well as using a moving average filter, we do not expect to get perfect readings or perfect tap detection. We do believe that our sensor will be able to detect tilt angles within 4 degrees of inaccuracy and that the tapping will work relatively well. We are anticipating the LED peripherals to respond as expected.

3 Implementation

todo

4 Testing and Observations

todo

5 Conclusion

todo

6 Example Bibliography

6.1 References

[1] [taoup] Eric Steven Raymond. The Art of Unix Programming. Addison-Wesley. ISBN 0-13-142901-9.

[2] [walsh-muellner] Norman Walsh & Leonard Muellner. DocBook - The Definitive Guide. O'Reilly & Associates. 1999. ISBN 1-56592-580-7.

A Example Appendix

AsciiDoc article appendices are just just article sections with *specialsection* titles.