Lab 3 Report

ECSE 426 – Microprocessor Systems

Group 7

Gregoire Martin – 260364773

Patrick White – 260353491

November 4th 2013

# Table of Contents

[1. Abstract 3](#_Toc369952480)

[2. Problem Statement 3](#_Toc369952481)

[3. Theory and Hypothesis 4](#_Toc369952482)

[4. Implementation 6](#_Toc369952483)

[5. Testing and Observations 10](#_Toc369952484)

[6. Conclusion 12](#_Toc369952485)

[Figure 1: Alpha and Beta Angles in 3D 4](#_Toc371340093)

[Figure 2: Pitch and Roll of Accelerometer (ST, 2010) 5](#_Toc371340094)

[Figure 3: PWM (Duty Cycles) 6](file:///C:\Users\pwhite8\Lab%203%20Report%20Group%207.docx#_Toc371340095)

[Figure 4: Flowchart of overall system 7](#_Toc371340096)

[Figure 5: LEDs updating algorithm 9](#_Toc371340097)

[Figure 6: Data point matrix 10](#_Toc371340098)

[Figure 7: Calibration matrix 11](#_Toc371340099)

[Figure 8: Origin 13](#_Toc371340100)

[Figure 9: Reverse Origin 13](#_Toc371340101)

[Figure 10: Maximum Roll 13](#_Toc371340102)

[Figure 11: Minimum Roll 13](#_Toc371340103)

[Figure 12: Maximum Pitch 13](#_Toc371340104)

[Figure 13: Minimum Pitch 14](#_Toc371340105)

[Figure 14: PWM pulse train (B, 2011) 14](#_Toc371340106)

# 1. Abstract

The primary goals of this experiment are to measure the acceleration of the STM32 discovery board and provide a simple graphical output using the board’s LEDs, as well as to implement a pulse width modulation (PWM) algorithm using the board’s timers and to provide an output on the board’s LEDs, and finally to provide the user a method of switching between the two modes of operation by tapping on the board. This experiment involved the use of an accelerometer, hardware timers and interrupts, basic output using LEDs, as well as some signal processing to calibrate the accelerometer.

# 2. Problem Statement

The end goals of this experiment are to have a simple LED display indicating the tilt angles of the board, to display PWM on the LEDs using a hardware timer as well as to provide a way of switching between modes of operation by tapping on the board. The problem can be broken down into five parts

* Acquiring data from the accelerometer
  + The accelerometer must be sampled at a rate of 25 Hz,
  + The sampling rate must be provided by a hardware timer using the Nested Vector Interrupt Controller (NVIC),
  + The data must be passed through a moving average filter
  + The accelerometer must be calibrated to provide an accurate tilt angle within 4 degrees
* Updating the LEDs according to the tilt angles
  + The pitch and roll of the board must be computed in real time for each accelerometer reading
  + The LED frequencies or pattern must be updated according to the pitch and roll of the board
* Generate the PWM signal in hardware
  + The PWM signal must be generated using a hardware timer,
  + Output compare mode must be used to demonstrate the PWM algorithm on the LED
* Providing the user a way of switching modes of operation via tapping the board
  + The external tap signal from the accelerometer interrupt lines must be channeled to a GPIO,
  + The board must switch between modes when a reasonable tap is recognized
* Display the sensor measurements and tilt angles in real time
  + The SWD debug interface must be used to print out the pitch and roll angles after each sample

These five aspects will allow the board to provide a simple display to the user that describes the tilt angles of the board, as well as demonstrating PWM on the board’s LEDs and allowing the user to easily select the desired mode of operation.

# 3. Theory and Hypothesis

The values obtained directly from the accelerometer will be composed of three mg readings, one for each axis. Since these values are already in units of gravity, no unit conversion is necessary.

Acceleration is a measure of the projection of the gravity vector on the sensitive axis. For this reason, the amplitude of any sensed acceleration can be used to calculate an angle, α, between the sensitive axis and the horizontal plane. Expanding this concept to 3D space, an angle,, can be calculated with respect to one sensitive axis (x-axis), while another angle, , can be calculated with respect to another sensitive axis (y-axis) as seen in Figure 1: Alpha and Beta Angles in 3D.

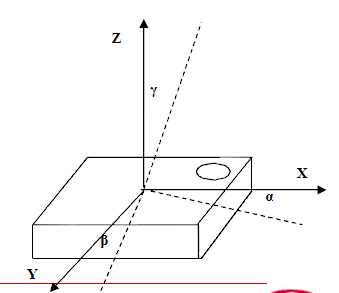


Figure 1: Alpha and Beta Angles in 3D

From Lecture 7, the formulas to obtain these angles, based on the magnitude of acceleration in each of the three axis, are as follows:

)

Equation 1: Alpha angle based on acceleration

)

Equation 2: Beta angle based on acceleration

The angles and will be calculated in radians. Pitch () is defined as the angle between the Xb axis and the horizontal plane (ST, 2010). Roll () is defined as the angle between the Yb axis and the horizontal plane (ST, 2010). Therefore, to obtain a value for the pitch and roll as seen in Figure 2: Pitch and Roll of Accelerometer of the board, the angles must be converted into degrees. The pitch of the board will correspond to the degree value of , while the roll of the board will correspond to the degree value of .

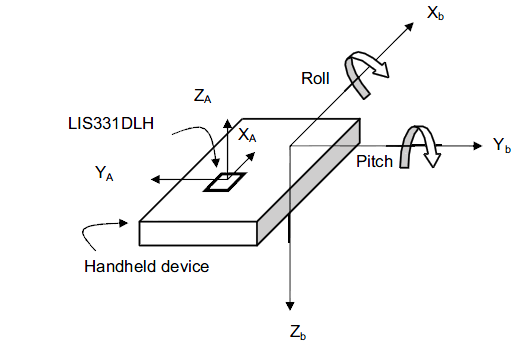


Figure 2: Pitch and Roll of Accelerometer (ST, 2010)

It is expected that the acceleration readings from the sensor will have unwanted fluctuations which will lead to inaccurate angle readings. This noise can be caused by many sources; electromagnetic interference, thermal noise, cross-axis sensitivity, zero-g bias levels among other things. The signal must be processed in such a way to filter as much of this noise as possible. A very simple moving average filter can be used. The moving average filter keeps a buffer of the D most recent samples, and takes the average of those values and outputs that average. When a new measurement is taken, the oldest measurement in the buffer is discarded, the new measurement added to the buffer, and the average recomputed.

In general, a smaller window (D) would be better suited to the task at hand in order to preserve the resources of the system, while reducing the noise in the signal to an acceptable level. For example, a filter depth of 10, as in experiment 2, may be desirable.

The accelerations returned will also contain some physical error. To ensure that the correct acceleration is observed in each case, the readings need to be calibrated for known accelerations in known orientations. There are multiple possible implementations for calibration. A high accuracy method is the least squares method, which will be discussed at length in Section 4: Implementation.

The LEDs on the board are driven by digital signals. For this reason, they are in either an “on” or “off” state. To display varying LED intensities, a duty cycle must be used. A longer duty cycle with respect to a known timer period will cause the LEDs to be “on” for a larger portion of the known timer period. This timer period must be small enough so that if the LEDs are toggling at its frequency, they will appear as always on. As demonstrated in Figure 2: PWM (Duty Cycles), with 100% duty cycle, the LEDs are never set to the off state, whereas for a duty cycle of 0% the LEDs are never set to the “on” state. Using varying sizes of duty cycles will result in varying intensity being displayed on the LEDs.



Figure 3: PWM (Duty Cycles)

The same concept applies directly to the PWM algorithm. Using two timers (one fast and one equivalent to the sampling rate) gives the ability to gradually increase a pulse length with respect to the faster timer every time the sampling timer interrupts. The pulse length is increased until its length is equivalent to the faster timer’s rate. While the fast timers’ count is less than or equal to the pulse length, the PWM signal is held high, otherwise, the PWM signal is low. When the pulse length is equivalent to the fast timers’ final count (period) the LEDs will be held on at all times.

# In order to obtain a reliable reading from the tap recognition of the accelerometer, some thresholds and timeout values will have to be computed. Threshold values indicate the intensity of acceleration observed required to be recognized as a tap, while the timeout value indicates the sampling rate of the tap recognition. A threshold value that is too small will result in small movements or change in position being recognized as a tap, while a value that is too large will require a large impulse to be recognized. A timeout value that is too small will not allow the full force of a tap to be properly recognized or may cause one tap to be recognized as two, while a timeout value that is too long will result in a slow tap response. These values will have to be computed experimentally, most likely by trial and error.

# 4. Implementation

The system designed to meet all the requirements of the problem statement can be seen in Figure 4: Flowchart of overall system. After initializing the accelerometer, the hardware timers, the LEDs and the filter window, the system enters into a continual loop. A simple two state machine is implemented to switch between displaying the tilt angle algorithm and displaying the PWM algorithm on the LEDs. The machine switches state on tap recognition.



Figure 4: Flowchart of overall system

The first step of implementation was the initialization of all system components. The initialization of the accelerometer and hardware timers will be further discussed in the next sections. The LEDs are configured in output compare mode as described in Section 3: Theory and Hypothesis. The timer used for this mode is TIM4, which has been programmed to have a frequency of 100Hz. This frequency was selected because it is the frequency at which toggling the LEDs is no longer noticeable (appears always on). The frequency was calculated based on equation (3).

|  |  |
| --- | --- |
| Period | 65535 |
| Prescaler | 13 |
| SystemCoreClockFrequency | 168MHz |
| Frequency | 100Hz |

Table 1: Parameters for TIM4

Equation 3: Frequency of a timer

Each LED is configured on its own channel in output compare mode with respect to the hardware timer. Because of this, each channel can be programmed to have different pulse length values (with respect to a period of 65535). These programmed values can be updated and changed at any point in the process. This can be used to control the duty cycle, and ultimately the intensity, of each LED. At 100Hz, any toggling of the LEDs occurs faster than human perception (i.e. even if it is toggling at 100Hz, it will still look as though it is always on).

In order to enable tap recognition on the accelerometer, click interrupt mode has to be enabled. An NVIC is used. First, the accelerometer must be programmed for the correct mode of operation (SingleClock\_Axes = CLICKINTERRUPT\_XYZ\_ENABLE). Then, the control registers for the thresholds and timeout period of the click interrupt are programmed (maximum values in this case). These values control the sensitivity of the tap recognition. Then, since the interrupt from the accelerometer is external from the MEMS chip, a GPIO pin must be configured in floating mode, this pin will be used to input the interrupt from the accelerometer to the systems core. The priority of this interrupt is set to 2 (1 being for sampling interrupt). Then we simply enable the clock on the bus that will be used (APB2Periph\_SYSCFG) and force the interrupt onto the GPIO pin. An interrupt service routine (ISR) has been implemented to execute whenever this external interrupt is driven high. The ISR simply clears the interrupt bit and sets an interrupt flag to true. In the main body of the code, if this flag is detected, we switch modes of operations, clear the flag and we reset the latch bits in the config register. No reconfiguration of timers or GPIO is required when changing modes.

|  |  |
| --- | --- |
| XY Threshold | 0xFF |
| Z Threshold | 0xFF |
| Timeout Period | 0xFF |
| Latency | 0x7F |

Table 2: Click registers parameters

The first mode of operation involves tilt angle calculation. The acceleration values per axis can be directly sampled from the accelerometer, no conversion is necessary. The accelerometer has been configured to its lowest data rate, 100Hz. Since it will only be sampled at 25Hz, this is acceptable. The measurement range of the accelerometer is set to +- 2.0 g. Therefore, the values returned by the accelerometer for each axis will range from -2000mg to 2000mg. Although this range does not provide the ability to recognize large accelerations, it provides a higher precision for the smaller accelerations which is critical for tilt calculation. The range of the accelerometer does not affect the size of the data it is returning. For this reason, a higher precision reading does not consume more system resources.

|  |  |
| --- | --- |
| Data Rate | 100Hz |
| Acceleration range | +- 2g |

Table 3: Accelerometer parameters

In accordance with the specification, the sensor was to be sampled at 25 Hz. To get this sampling frequency, a hardware timer was implemented and set to interrupt using the NVIC. The parameters required to generate this frequency were calculated using Equation 3: Frequency of a timer. The period and prescaler values of a timer are programmed into 16-bit registers for that specific timer. For this reason, the maximum possible value for either of them is 2^16 – 1 = 65535. A larger value will overflow the registers. To generate an interrupt every 25Hz, a period of 65535 and a prescaler of 50 were selected. The timer will raise an interrupt flag every 25Hz. The ISR will be executed every 25Hz and will clear the interrupt bit.

|  |  |
| --- | --- |
| Period | 65535 |
| Prescaler | 50 |
| SystemCoreClockFrequency | 168MHz |
| Frequency | 25Hz |

Table 4: Parameters for TIM3 (Sampling Rate)

When the interrupt is received, it signals that a new reading from the accelerometer should be taken. At this point, 3 data points are taken from the accelerometer, 1 for each axis. This data is then passed into the moving average filter, where a new average acceleration in each axis is computed. The filter was implemented as a C struct. The filter is essentially a ring buffer, which allows new values to be inserted directly into the buffer, rather than having to loop through and shift all buffer entries down to be able to insert a new entry at the end of the buffer. The size of the ring buffer implemented is of size D (discussed later), the window depth of the filter. These new average values are then used to calculate the angles and using Equation 1: Alpha angle based on acceleration and Equation 2: Beta angle based on acceleration respectively and subsequently the pitch and roll of the board.

Once the pitch and roll have been computed, the LEDs can be updated based on their magnitude. Since the LEDs are set in the output compare mode, they can be easily set to varying intensities by simply programming a new pulse length values with respect to their hardware timers period. A simple implementation is to ramp up the pulse length (duty cycle) of the LEDs based on its angle. For example, an absolute range of 0 to 90 degrees in either pitch or roll would correspond to a duty cycle range of 0% to 100%. Therefore, as the angle increases, so to would the LEDs intensity. However, a more complex algorithm for updating the LEDs was implemented. Only the LED in the direction of the tilt will be ramped up. Therefore, two opposing LEDs are used to display pitch and two opposing LEDs are used to display roll. For example, a roll range of 0 to -90 degrees will cause a ramp on one LED, while holding the other LED off. Similarly, a roll range range of 0 to 90 degrees will ramp the other LED while keeping an LED off. A graphical representation of the algorithm can be seen in Figure 5: LEDs updating algorithm.



Figure 5: LEDs updating algorithm

To optimize the moving average filter, a Matlab model was used to compare different buffer lengths and the resulting quality of data. The moving average filter was applied to several data sets from the accelerometer, and the buffer length was varied. After running several steady state data sets through the Matlab model, it was determined that a buffer of length 10 would be used. For the specifics of the testing and optimization, see Section 5: Testing and Observations.

To calibrate the accelerometer, the least squares method was used. The method involves multiplying accelerometer values by a calibration matrix with parameters ACCij. The equation can be written as

= \*

OR

Y = w \* X

Equation 4: Calibration calculation

Where Y is a vector of the resulting accelerometer values, w is a vector containing the values read from the accelerometer, and X is a matrix containing the calibration parameters. The matrix X is obtained from rearranging the equation into

X = [wT \* w]-1 \* wT \* Y

Equation 5: Calculating the calibration matrix

In this case, Y will represent the ideal accelerometer readings (the Earth gravity vector). In order to obtain the most accurate X matrix possible, large amounts of data ( > 1000 data points) are collected at 6 different accelerometer orientations. These orientations are

|  |  |  |  |
| --- | --- | --- | --- |
| Orientation | X acc. (g) | Y acc. (g) | Z acc. (g) |
| Zb up | 0 | 0 | -1 |
| Zb down | 0 | 0 | 1 |
| Yb up | 0 | -1 | 0 |
| Yb down | 0 | 1 | 0 |
| Xb up | -1 | 0 | 0 |
| Xb down | 1 | 0 | 0 |

Table 5: Orientations used for calibration

Six vectors, w1, w2, …, w6, are synthesized from the data points in the form

Figure 6: Data point matrix

Where Axs, Ay, Az are the raw values obtained from the accelerometer in that orientation.

The corresponding Earth gravity vectors y1, y2, …, y6 are also used. For example, in Zb up position, the Earth gravity vector would be

These vectors are arranged into matricies

From here, X can be solved for.

Using the data acquired from the 6 different orientations, the calibration matrix was determined to be

X =

Figure 7: Calibration matrix

All of the accelerometer vectors are multiplied by this matrix upon being read and before being passed to the filter. This ensures that the values used by the rest of the application are more reliable than the raw accelerometer values.

The second mode of operation is the PWM mode. Since the LEDs have been configured in output compare mode, this is extremely easy to implement. A simple hardware counter is used as a time base. In this case, the same timer is also used for the output compare mode on the LEDs. By programming values with respect to the hardware timers period into the pulse length of each output compare channel, varying duty cycles can be generated. Then by simply programming successively larger pulse lengths beginning at 0 at each 25Hz interrupt, the LEDs can be gradually increased in intensity until the point at which they are always on. After a brief hold time (25 interrupts = 1 second) at max intensity, a value of 0 is programmed back into the pulse lengths and the entire process is repeated.

This current implementation performs a 4 second duty cycle. Within those 4 seconds, 100 interrupts will have occurred. By programming the pulse length of each channel to be the hardware timers period divided by 100 multiplied by the number of interrupts we have experienced, we perform a simple linear ramp of the LEDs. In this case, the number of interrupts we have experienced also corresponds to the percentage of our duty cycle.

# 5. Testing and Observations

First, it is useful to ensure that the frequencies of the timers, mainly the sampling rate timer, are in fact what are expected. This can be tested using another GPIO pin and testing one timer at a time. When the timers’ ISR is executed, we toggle the bit on the GPIO pin. This essentially generates a “clock signal” with a period twice as large as our timers’ period. Then, using an oscilloscope, the period of this “clock signal” can be observed, which in turn, provides us with period of our timer. The frequencies of both timers were confirmed using this method.

INSERT FILTER STUFF SOMEWHERE AROUND HERE, YOUR CALL WHERE

Then, we need to test the functionality and accuracy of the system.

In this first mode, we can simply test the accuracy of the tilt angles reported with respect to known inclines or tilts. The orientations that were tested correspond to the six orientations used for calculating the calibration matrix, as seen in Table 5: Orientations used for calibration. Visual depictions of these orientations, as well as a brief description of the expected output, are shown in Table 6: Testing orientation of board.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Figure 8: Origin | In this orientation both pitch and roll should be equivalent to 0. The only acceleration on the board is in the downwards z direction. This orientation produces tilt angles roughly equal to 0, but never exceeding 1 degree.   |  |  | | --- | --- | | Roll | 0 | | Pitch | 0 | |
| Figure 9: Reverse Origin | In this orientation both pitch and roll should again be equivalent to 0. However, the board has been flipped upside down with respect to the z-axis. Again, this orientation produces tilt angles roughly equal to 0, but never exceeding 1.   |  |  | | --- | --- | | Roll | 0 | | Pitch | 0 | |
| Figure 10: Maximum Roll | In this orientation, the pitch should be 0, while the roll of the board should be equivalent to 90 degrees.   |  |  | | --- | --- | | Roll | 90 | | Pitch | 0 | |
| Figure 11: Minimum Roll | In this orientation, the pitch should be 0, however, unlike the above orientation; the roll of the board will be -90 degrees.   |  |  | | --- | --- | | Roll | -90 | | Pitch | 0 | |
| Figure 12: Maximum Pitch | In this orientation, the roll should be 0, while the pitch of the board should be equivalent to 90 degrees.   |  |  | | --- | --- | | Roll | 0 | | Pitch | 90 | |
| Figure 13: Minimum Pitch | In this orientation, the roll should be 0, however, unlike the above orientation; the pitch of the board will be -90 degrees.   |  |  | | --- | --- | | Roll | 0 | | Pitch | -90 | |

Table 6: Testing orientation of board

The second mode can be tested in two ways. The simple, less accurate, method involves observing the LEDs ramp up to their maximum intensity on the board. This provides a simple confirmation that the duty cycle of the PWM is in fact increasing in duration. Then, if we wish to fully justify that the PWM algorithm is functional, an oscilloscope can again be used. Since the LEDs are already ported to a GPIO pin, we can simply monitor their output using a probe. A PWM pulse train should be observed on the pin. An example of a PWM pulse train can be seen in Figure 6: PWM pulse train.

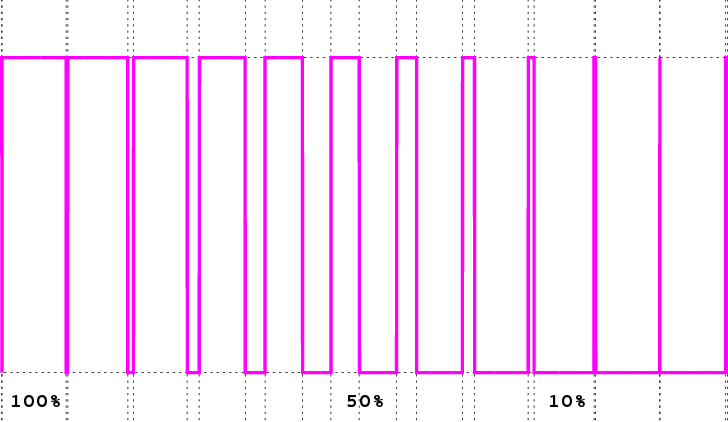


Figure 14: PWM pulse train (B, 2011)

Note how the above performs an entire PWM ramp within 12 equivalent chunks of time. Similarly, our PWM algorithm performs an entire PWM ramp within 100 chunks of time (4 seconds of 25Hz clock pulses). In our case, each chunk represents a 1% change in PWM from the last chunk. Since it is difficult to trace 4 seconds of data on an oscilloscope, we simply traced one chunk worth of data. Each successive chunk had the LED signal being held high for a longer percentage of the chunk, until finally, the final chunk had an LED signal that is always held high.

The final test involves tap detection. This test is much less formal than the others. If the board responds to a reasonable tap, and modes can be seen to switch, then the tap recognition is functional. If the board is too sensitive to taps, lower threshold values need to be programmed in.

# 6. Conclusion

The accelerometer of the STMFsdfsdf discovery board can be effectively used to report tilt angle of the board to a high degree of accuracy. In our case, a deviation of less than 1 degree was observed on known inclines. The tap recognition procedure is extremely sensitive to external impulses. Any slight jitter to the board can cause an interrupt. This is mainly due to the fact that extremely high tap thresholds had to be used to successfully implement the recognition of a light tap on a very stable surface. This is mainly due to the fact that very little acceleration is observed downwards when the board is tapped while sitting on a stable surface (no give to move down) since the board cannot physically accelerate downwards.

# Appendix

B, C. (2011, 05 29). *Principle of the intersective pulse width modulation (PWM).* Retrieved 04 2013, from

ST. (2010, April). AN3182 - Application Notes. Retrieved October 2013

STMicroelectronics. (2013). STM32F405xx & STM32F407xx Datasheet DocID022152 Rev 4.

Lecture 7 of course notes