

Lab 3: MEMS Accelerometer, Timer and Interrupts

ECSE 426 – Microprocessor Systems



McGill University

Winter 2015

Group 4 : Sidney Ng 260507001

Meng Yin Tao 260480207

Due March 16th 2015

# 1. Abstract

# 2. Problem Statement

The goal of this experiment is to create a system that detects the orientation of the STM32F4 discovery board. The rotation of the board around its axes is detected using the built-in accelerometer. However, the accelerometer cannot determine the yaw (rotation around the board’s z-axis) since it does not result in a change in acceleration. Only the roll (rotation around the board’s x-axis) and pitch (rotation around the board’s y-axis) can be found using the accelerometer. When the accelerometer’s raw data becomes available, the processor will calibrate it using a pre-built calibration matrix and filter it using a Kalman filter whose parameters are experimentally defined. In order to make sense of the data, it needs to be converted into an angle in degrees. This conversion is done using the arctan function available in the math library. Once the angle is available, it can be compared against the user input. The user is prompt to enter a target angle for both the roll and the pitch using a 4x4 external keypad. As the user tilts the board, an indication on the 7-segment displays an animation to help the user directing the board toward the target angle. Once the user is within five degree of the target roll angle, the 7-segment display will show the current angle captured by the accelerometer. This measured angle is expected to be within four degrees of accuracy compared to the actual angle

# 3. Theory and Hypothesis

## 3.1. Accelerometer

### 3.1.1. Orientation

--- Relationship between 3 axis and 3 angles (one of them cannot be detected)

One of the objectives of the lab was to display the tilt angles of the board using the accelerometer on the board. Since we only measure the tilt angles, the only acceleration force applied to the board is the gravitational force on the board. At sea level, the force should be 1g. Using only the accelerometer, we are able to calculate the pitch and the roll angles. The pitch angle is the angle between the X axis of the board and the horizontal plane, and is a rotation around the Y axis of the board, as seen in Figure 1. When pitch is 0°, the X axis of the board is on the same plane as the horizontal plane as in Figure 1. When pitch is 90°, the X axis of the board is perpendicular to the horizontal plane. The equation for the pitch is given by:

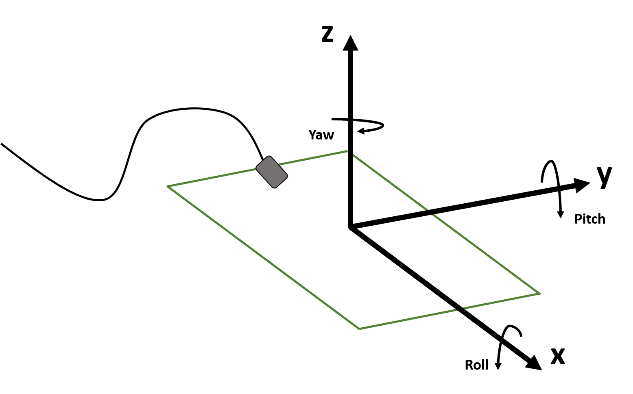
(1)

The roll angle is the angle between the Y axis of the board and the horizontal plane, and is a rotation around the X axis of the board, as seen in Figure 1. When roll is 0°, the Y axis is in the same plane as the horizontal plane. When roll is 90°, the Y axis is perpendicular to the horizontal plane. The equation for the roll is given by:

(2)

The yaw angle cannot be computed with just the accelerometer sensor. The pitch and the roll equations depend on the change in acceleration in the Z direction, gravity, while the other axis remains constant. However, since there is no change in the force of gravity on the board while rotating around the Z axis, the yaw cannot be determined without using other sensors.

--- Figure showing the axis and angles



### 3.1.2. Acceleration

--- How the angle is quantified

## 3.2. Data Calibration and Filtering

--- When data ready (signaled by an interrupt flag), perform calibration

### 3.2.1. Data Normalization and Calibration

During the process of manufacturing the board, the MEMS accelerometer is not perfectly soldered to the board with its axes aligned. As a result, any acquired data must be normalized before using the data. The equation converting the raw data to the normalized acceleration is expressed as:

(1)

The [Am] matrix represents the misalignment between the accelerometer axes and the device’s axes, ASCirepresents the sensitivity (scaling) and AOSi is the zero-g level offset. Equation (1) is equivalent to:

(2)

(3)

The matrix X in equation (3) is equivalent to matrix [ACCij] in equation (2). The matrix X is the normalization vector multiplied to the raw data vector w in equation (2) to obtain the normalized data in matrix Y in equation (3).

Since the soldering of the sensor is different on every board, the normalization values (ACCij) must be calibrated. To acquire these values, the least square method is used to solve equation (3) for the matrix X:

(4)

For the values of matrices w and Y, the raw data values and corresponding normalized values are used from the positions Xup, Xdown, Y­up, Ydown, Z­up, Zdown. Z­up is when the board is facing up in the horizontal plane as in Figure 1. Y­up is when the Y axis of the board is where the Z axis is in Figure 1. Xup is when the X axis of the board is where the Z axis is in Figure 1. The down counterparts are when the axes are in the opposite direction of the Z axis in Figure 1. The Y matrix is equal to:

(5)

The first row of the matrix corresponds to Zup, the second to Zdown, the third to Yup, the fourth to Ydown, the fifth to Xup, and the sixth to Xdown. The first column holds X values, the second Y values, and the third holds Z values. The w matrix follows the same format of the Y matrix. However, it has an additional fourth column of 1’s.

--- 6 positions expected (not normalized?) acceleration (0 0 1, 0 0 -1, 0 1 0, 0 -1 0, 1 0 0, -1 0 0)

--- Calibration operation (need equation)

### 3.2.2. Data Filtering

--- Kalman filter (what’s the purpose of filtering?)

### 3.2.3. Data Interpretation

--- Normalize acceleration

--- Convert from acceleration to angle in radian (need equation)

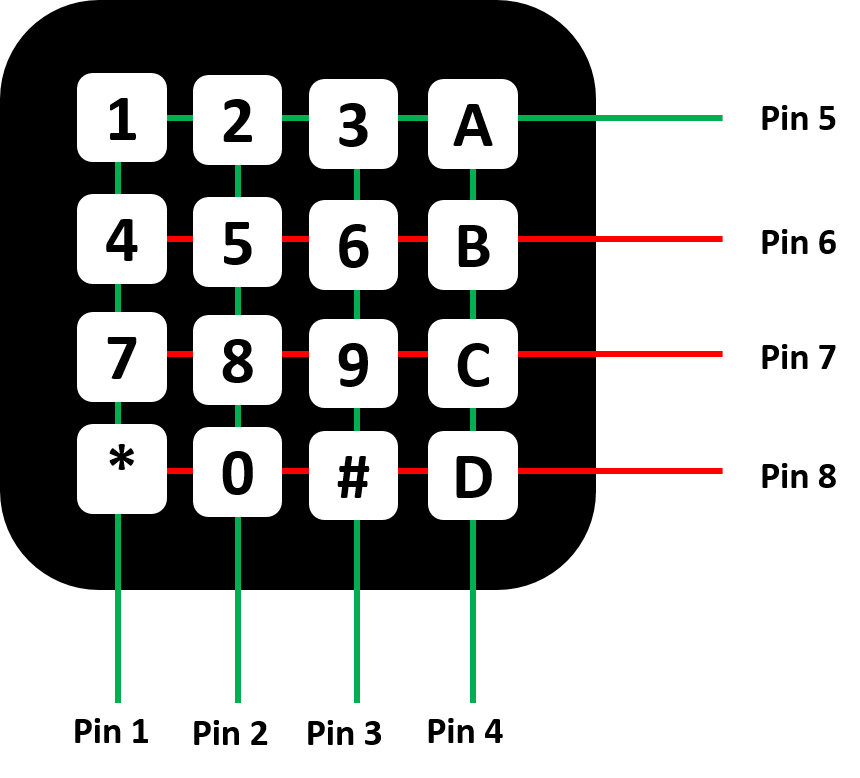
--- Alternative way of getting the angle (lookup table)

## 3.3. External Keypad

### 3.3.1. Circuit Layout

--- Determine connections (experimental)

--- Figure showing the layout



### 3.3.2. Data Acquisition

--- Set, detect, reverse, detect

### 3.3.3. Handling Key Bouncing

--- Sample the same digit multiple times

--- Compare with previous value to detect changes

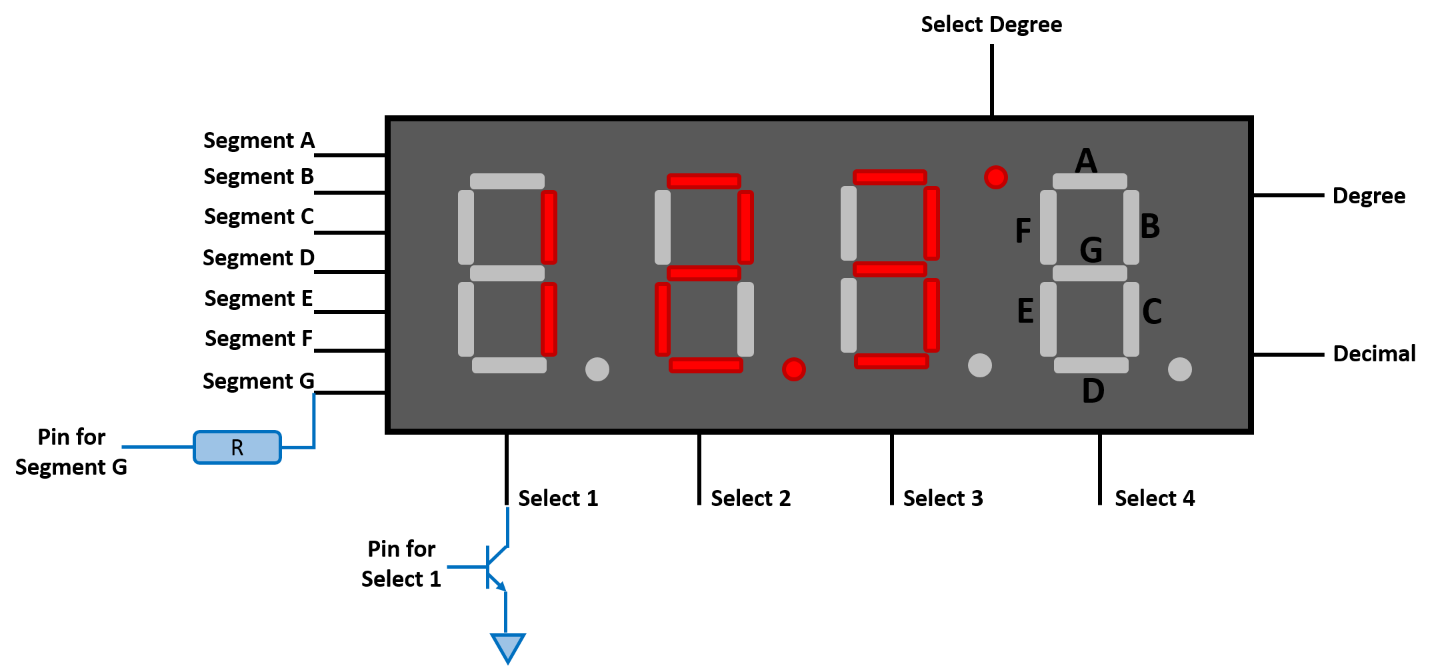
--- Ignore short period NO INPUT signal

## 3.4. External 7-Segment Display

### 3.4.1. Circuit Layout

--- Resistors and transistors to protect the circuit against possible high current flow

--- Figure showing the layout



--- Use of resistors and transistors

### 3.4.2. Data Display

--- Select digit, display, next digit

## 3.5. Timing

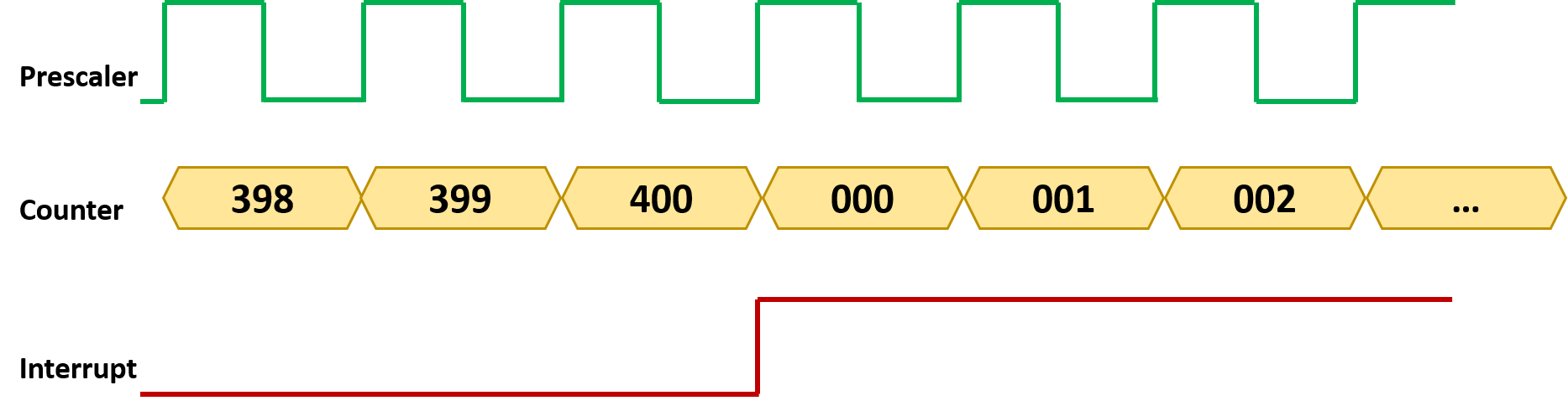
### 3.5.1. Timing Based on Sample Rate

--- Interrupt when sample is ready

### 3.5.2. Timing Based on Hardware Timer

--- Clock frequency, Prescaler and Period 🡪 Interrupt (show equation)

--- Figure to show the relationship



# 4. Implementation

## 4.1. Component Configuration and Initialization

### 4.1.1. GPIO Configuration

--- Includes PE0 for accelerometer

--- 7-segment pins

--- On board LEDs

### 4.1.2. Accelerometer Configuration, Accelerometer Interrupt and Interrupt Handler

--- Pin and channel

--- Frequency of the interrupt

### 4.1.3. Timer Configuration, Timer Interrupt and Interrupt Handler

--- Frequency of the interrupt

--- Interrupt variables (TIM3\_Interrupt and TIM3\_Interrupt\_Count)

## 4.2. Collect User Input

### 4.2.1. Initialize Keypad

### 4.2.2. Return User Input

### 4.2.3. Improvement on Keypad

## 4.3. Tilt Angle Calculation

The process of calculating the tilt angle involved sampling the accelerometer data, normalizing the data, filtering the normalized values, then performing the calculations.

### 4.3.1 Data Sampling

As per the requirements, the accelerometer sampled values at a frequency of 100Hz. To avoid continually polling the accelerometer for data, we only acquired data when an interrupt was generated when the accelerometer indicated it had data ready. Since the accelerometer is not hardwired to the processor, we used EXTI (External Interrupt) to generate the interrupt and NVIC (Nested Vector Interrupt Controller) to channel the interrupt.

To detect the interrupt, we used a global variable as a flag. In the interrupt handler, the flag is set. In our main loop, if the flag is set, then we proceed to reset the flag and read the new data values. Then the data is processed.

--- Signal processor when data is ready

### 4.3.2. Data Processing

Processing the data requires three steps. The first step is to normalize the data using the normalization matrix. The matrix values are stored in the tilt\_detection.h header file as an array of 12 elements. In order to perform matrix operations on the data, the array is stored within an arm\_matrix\_instance from the CMSIS DSP API. When normalizing the data is stored as a 1x4 matrix and multiplied with the normalization vector as in equation (x). The normalization parameters we used are given in Table 1. After normalizing, the output is passed to the Kalman filter.

Table 1s

|  |  |  |
| --- | --- | --- |
| -0.001007704542329 | -0.000023576795574 | -0.000020500331472 |
| -0.000005549672267 | -0.001001988618073 | -0.000000555682904 |
| 0.000001981760320 | -0.000006760475090 | 0.000964570821189 |
| -0.520517014244453 | 0.477262260417352 | -0.440876330953729 |

--- Show calibration data in **Appendix A**

--- Kalman parameters, show Matlab simulation results in **Appendix B**

The second step is to filter the data. We used the Kalman filter taken directly from lab 2. We determined the parameters experimentally as described in section 5.2. For this application, we required the use of three different instances of a Kalman filter state, one for each of the accelerometer values. Using only one Kalman filter would have resulted in incorrect final calculations. Each individual input would have advanced the iteration when all three accelerometer values should be in the same iteration for different states. The filtered values are then passed to tilt angle calculation.

The final step is to perform the calculation. The pitch angle we determined by using equation (x) and the roll angle we determined by using equation (y). However, the result of these values only range from -90 to 90. As a result, we included calculations to determine the quadrant of the angle given the sign of the z acceleration and the sign of the calculated degree.

### 4.3.3. Data Comparison

---

## 4.4. Result Display

### 4.4.1. Visual Feedback

--- Animation to direct user to move in the right direction

### 4.4.2. Display Current Angle

--- Only displaying the beta angle

## 4.5 Continuous Process

--- Continuous process until user presses reset

# 5. Testing and Observation

## 5.1. Accelerometer Calibration

To calibrate our sensor, we performed offline calibration as described in section 3.2. First we collected samples of raw accelerometer values at the six positions as described in section 3.2. Then we averaged the acceleration values and formed the w matrix in accordance with its format as described in section 3.2. Then in Matlab, we solved for equation (4). Taking the output values, we then hardcoded them to an array. The normalization measured the tilt angles within 4 degrees of accuracy.

--- Within 4 degree accuracy

--- Low accuracy, therefore could use lookup table instead

--- Using complex math functions on floating points --- more power hungry

--- Using lookup table --- less accurate --- more memory needed --- less power

## 5.2. Kalman Filter

The Kalman filter initial state parameters we chose were q = 0.025, r = 5, x = k = p = 0. x is the estimated value, and since each acceleration value will be in the range of 0 to 1, we set x to 0. To get the other values of the parameters, we plotted a comparison of filtered values with the raw data with varying parameter values. We held r constant at 5 and increased q from 0.0025 to 2500 as seen in Figures 4 to 11. Just by comparing visually, as the value of q rose, the signal became noisier. The optimal value we found at q = 0.025 and r = 5 (Figure 5). The figures show the comparison for only the X values, but the filters for the Y and Z values gave the same result.

--- Choice of constants

# 6. Conclusion

# Reference

# Appendix A – Calibration Data

