Report Due on Tuesday, October 14.

Sensor Tile Lab 2 Report: sensor system signal acquisition, event detection and configuration

CEE-445 Embedded System, Fall 2019

Student Name: Jacob Hillebrand

Introduction: Dr. Cheng Liu

In this lab, we will be further exploring the sensortile system by examining the code used in the first lab and modifying it to fit our own criteria in an effort to gain a better understanding of the functionality behind this embedded device.

Experimental data:

1. Examining the sensor sampling rate and output data rate:
   1. Calculate the difference between 5 successive timestamps (i.e. t1 = T2-T1, t2 = T3-T2,

Timestamp 1: 26.62s

Timestamp 2: 26.73s

Timestamp 3: 26.82s

Timestamp 4: 26.92s

Timestamp 5: 27.03s

Timestamp 6: 27.12s

delta2-1: .11s

delta3-2: .09s

delta4-3: .10s

delta5-4: .11s

delta6-5: .09s

* 1. Average these differences (t1 + t2 + t3 + t4 + t5)/5. What is this average

difference? See Figure 2 in the lab pdf.

Averagedelta: .10s

* 1. Explain what this code does: **msTick = HAL\_GetTick();**

This code will use the HAL\_GetTick() function to retrieve the time, in msecs, that has elapsed since the sensor began operation. It will then assign this value to the msTick variable.

1. Modifying the USB data output:

Calculate the difference between 5 successive timestamps (i.e. t1 = T2-T1, t2 = T3-T2, …, t5 = T5-T4). Average these differences (t1 + t2 + t3 + t4 + t5)/5. What is this average difference? See Figure 2 in the lab pdf. Does this agree with your expectations?

Timestamp 1: 35.99

Timestamp 2: 36.99

Timestamp 3: 37.99

Timestamp 4: 38.99

Timestamp 5: 39.99

Timestamp 6: 40.99

Delta 2-1: 1s

Delta 3-2: 1s

Delta 4-3: 1s

Delta 5-4: 1s

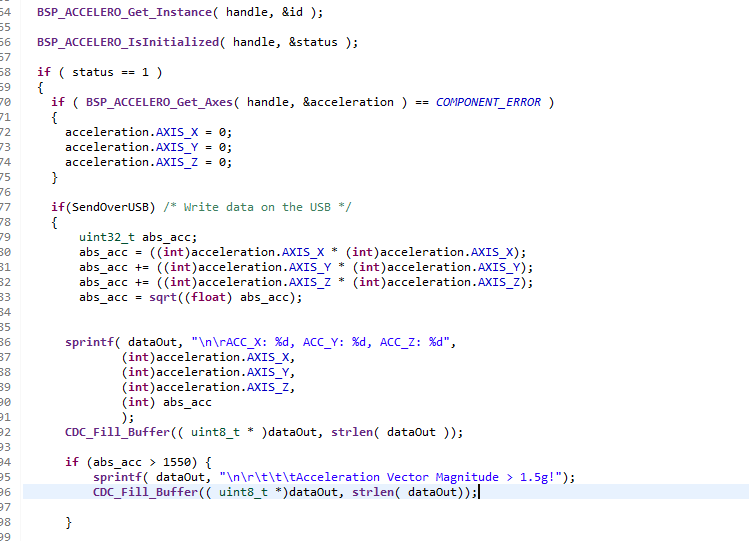
Delta 6-5: 1s

Average Delta: 1s

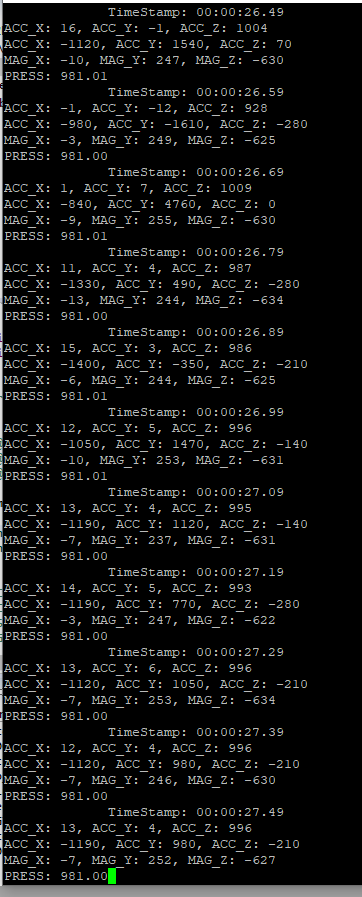
This matches my expectations, as I changed the polling rate from 100 milliseconds to 1000 milliseconds (1 second).

1. Creating new data, new messages, and event detection:
   1. Computing Vector Magnitude Acceleration:

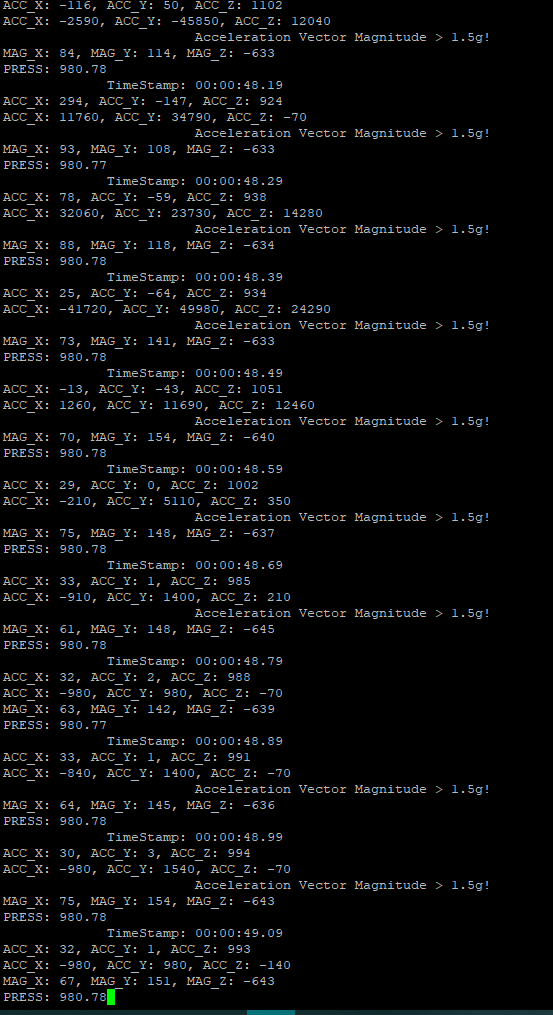
// during this lab exercise, you added some accelerometer code that writes its output data //over the USB to inform the user of the magnitude of the acceleration vector. Take a //screen shot of your code, cut and paste to the space below.



* 1. Take a screen shot of the messages you receive over the USB Serial port.



1. In the section of detecting Events: Acceleration Vector Magnitude Threshold Crossing, examine the messages received by your personal computer over the serial connection, shake the sensor gently, and take a screenshot the message appearing over the USB Serial port.



Conclusion:

Sensor data acquisition and event detection both have plenty of uses in real-world applications. For example, let’s look at car airbags. During a crash, a car will experience significant g-forces as it comes to a halt, which can be detected by an accelerometer on the car. If these forces are significant enough, this can trigger an event to indicate the car is crashing. Then upon the event trigger, the airbags can deploy to keep the driver and passengers safe.

For a second example of sensor data acquisition and event detection in the real world, we can consider the sensors inside of a modern x86 CPU chip. The temperature sensors will monitor the CPU and provide data on how warm the chip is becoming. If it becomes too hot, an event can be triggered, and the CPU will know to slow itself down to prevent any damage from overheating. This will protect the CPU from damage and extend its service life considerably. Both this and the air bags are perfect examples of how sensor data acquisition and event detection can be used in real-world applications.