Lab 5 report – TI TivaC/ADC compiling and booting

Student name: Vu Nguyen (71141)

Date: 05.03.2015

Assumptions on knowledge of the reader

It is assumed that a reader should know some basic programming skills in C/C++ in order to understand the source code. A basic knowledge of hardware such as processors, LEDs, pin ports, and accelerometer should be known.

Description of the task

The task of lab 5 is to construct software for reading a 3-axis accelerometer. In normal condition (acceleration 9.81m/s^2), the green LED on the expansion board is lit. In high acceleration (> 12 m/s^2), the yellow LED is lit. In higher acceleration, the red LED is turned on.

Description of the equipment used

The equipment used consists of a TivaC TM4C123G LaunchPad evaluation kit by Texas Instruments, a homebrew daughtercard (expansion board) with 3 LEDs: red, green and yellow, an ADXL327B accelerometer. A microB to A type USB cable is used for powering the platform and programming it. Regarding the development environment, Arduino-based Energia IDE is used.

Description of the performed work

The accelerometer measures acceleration in 3 axis (x, y, and z), and is read using the ADC converter of the TivaC development kit. The LEDs (green, yellow, red) are accessed through the port pins PB2, PE0, and PB7, respectively. The total acceleration is calculated using the following equation.

$$r = k * \sqrt{(rX^2 + rY^2 + rZ^2)}$$

where k is the gain constant of the accelerometer.
r is the total acceleration or the magnitude of the force vector the accelerometer is measuring
rX, rY, and rZ are projections of vector r on x, y, and z axes respectively

In fact rX, rY and rZ are the voltage shifts from zero-g voltage. From the datasheet of the ADXL327B accelerometer, the zero-g voltage is Vdd/2. The power supply to the accelerometer is given from the development kit, and it is 3.3V. Therefore, the zero-g voltage is 1.65V. Since there are imperfections during chip manufacturing process, the accelerometer has its offset. This offset needs to be calibrated. Therefore, function calibrate_ADXL327B is used to calibrate the accelerometer. This function determines the gain constant k, the zero-g voltages from x, y, and z axes. The function takes as argument the number of samples and uses it for calculating the average values. After calibration phase is done, the software will enter an infinite loop in which it calculates the total acceleration, converts the acceleration values from g to m/s². After that it compares the calculated acceleration value with the acceleration levels (normal, high, very high) and turn on/off the LEDs accordingly.

Implementation of calibration function

Function prototype: void *calibrate_ADXL327B(unsigned int nsamples)*

The function turns on the green LED on the expansion board to signify that it is in the process of measuring the zero-g voltages of the x and y axes. To start this process, a button press is expected from the user. During this process, it calculates zero-g voltages as follows:

```
/* Turn on green LED to signify calibration of x and y axis begins */
digitalWrite(LED_GREEN, HIGH);

/* Wait for a press button from the user to start calibrating */
while(digitalRead(BUTTON_A2) == HIGH);

/* Calibrating x and y axis */
for (i = 0; i < nsamples; i++) {
  xZero += (analogRead(XPIN) * (float)VMAX) / (float)DMAX;
  yZero += (analogRead(YPIN) * (float)VMAX) / (float)DMAX;
}
xZero = xZero / nsamples;
yZero = yZero / nsamples;
```

After calculation is done, the green LED is turned off. It is important to note that when the green LED is turned on, the user has to lay the accelerometer flat so that the axes are zero degree to the horizontal.

Next, the function call *delay* function to avoid button bouncing problem caused by pressing mechanical button. When the yellow LED on the expansion board is turned on, measuring zero-g voltage on the z axis begins after the button is pressed. It is implemented in the following code snippet.

```
/* Turn on yellow LED to signify calibration of z axis begins */
digitalWrite(LED_YELLOW, HIGH);

/* Wait for a press button from the user to start calibrating */
while(digitalRead(BUTTON_A2) == HIGH);

for (i = 0; i < nsamples; i++) {
  zZero += (analogRead(ZPIN) * (float)VMAX) / (float)DMAX;
}
zZero = zZero / nsamples;

/* Turn off green LED to signify that calibration of z axis is done*/
digitalWrite(LED_YELLOW, LOW);
```

When the yellow LED is turned on, the user has to adjust the orientation of the accelerometer so that the z axis is perpendicular to the vertical. Pressing the button on the expansion board will start this process. This process ends when the yellow LED is turned off. Once zero-g voltages for x, y, and z voltages have been determined. It continues by calculating the gain constant k. Then red LED is turned on to let the user know that it is about to calculate the constant k. The job of the user in this phase is to lay the accelerometer flat the way it was done when calculating zero-g voltages for x and y axes. After calculation is done, it turns off the red LED. Implementation of this is shown below

```
/* Turn on red LED to signify that calculation of k is in progress */
digitalWrite(LED_RED, HIGH);

/* Wait for a button press from the user */
while(digitalRead(BUTTON_A2) == HIGH);

/* Calculate gain constant k */
rX = (analogRead(XPIN) * (float)VMAX / (float)DMAX - xZero) /
       SENSITIVITY;
rY = (analogRead(YPIN) * (float)VMAX / (float)DMAX - yZero) /
       SENSITIVITY;
rZ = (analogRead(ZPIN) * (float)VMAX / (float)DMAX - zZero) /
       SENSITIVITY;
k = 1 / sqrt(pow(rX, 2) + pow(rY, 2) + pow(rZ, 2));

/* Turn off red LED after calculation is done */
digitalWrite(LED_RED, LOW);
```

Implementation of total acceleration calculation and comparison

After zero-g voltages from x, y, z and the gain constant k have been calculated during the calibration process, the software enters the infinite loop. In this loop it calculates the total acceleration values by BUFFER_SIZE times, and calculates the average value as the final total acceleration. The reason for this is that the value of total acceleration fluctuates significantly in a short period of time. So taking average is a simple way to smooth out the acceleration value. Once the total acceleration value is determined, it continues by converting this value from g to m/s² and comparing the value in m/s² with the predefined acceleration levels (normal, high, very high). The LEDs are turned on/off based on the ranges. This can be seen in the following code snippet.

```
void loop() {
  int i;

  /* Calculate total acceleration */
  /* Calculate the average value of r to smooth out its total value since it
fluctuates a lot */
  for (i = 0; i < BUFFER_SIZE; i++) {
    rX = (analogRead(XPIN) * (float)VMAX / (float)DMAX - xZero) / SENSITIVITY;
    rY = (analogRead(YPIN) * (float)VMAX / (float)DMAX - yZero) / SENSITIVITY;
    rZ = (analogRead(ZPIN) * (float)VMAX / (float)DMAX - zZero) / SENSITIVITY;
    arr_r[i] = k * sqrt(pow(rX,2) + pow(rY,2) + pow(rZ,2));
    Serial.println(arr_r[i]);
  }
  r = 0;
  for (i = 0; i < BUFFER_SIZE; i++) {
    r += arr_r[i];
  }
  r = r / BUFFER_SIZE;

  /* Convert from total acceleration g to m/s^2 */
  r_ms2 = r * 9.81;

  /* Turn on LEDs according to accleration values */
  if (r_ms2 >= 0 && r_ms2 <= NORMAL_ACC) {
    digitalWrite(LED_GREEN, HIGH);
    digitalWrite(LED_RED, LOW);
    digitalWrite(LED_YELLOW, LOW);
  } else if (r_ms2 > NORMAL_ACC && r_ms2 <= HIGH_ACC) {
    digitalWrite(LED_YELLOW, HIGH);
    digitalWrite(LED_RED, LOW);
    digitalWrite(LED_GREEN, LOW);
  } else {
    digitalWrite(LED_RED, HIGH);
    digitalWrite(LED_YELLOW, LOW);
    digitalWrite(LED_GREEN, LOW);
  }
}
```

Achieved results

It can be concluded the device works basically. The acceleration value still fluctuates a lot over a short period of time, which results in the green LED and the yellow LED being turned on/off in a short time even the accelerometer is laid flat on the testing table. However, this does not happen very often. Perhaps for further improvements, some kind of digital filter could be implemented to filer noise and smooth out the signal. Besides, errors during calibration could be the problem which in turn leads to incorrect total acceleration value when it is calculated in the infinite loop.