SHIMMER 3

Basic fall detection

Written by Steffan Lildholdt (steffan@lildholdt.dk)

TABLE OF CONTENT

1. INTRODUCTION

Shimmer is a small sensor platform well suited for wearable applications. The integrated kinematic sensors, large storage and low-power standards based communication capabilities enable emerging applications in motion capture, long-term data acquisition and real-time monitoring. Shimmer offers the possibility for custom firmware integration so that the device can be tailored to fit many different applications. Shimmer3_FallDetection is an example of customized firmware.

The standard firmware “BtStream” transmits raw data from predetermined sensors with a rate of 50 Hz leaving the data processing to be carried out at a computer or mobile device. Transmitting this amount of data, yields relatively high power consumption. To extend the battery lifetime it is preferred to perform data aggregation locally on the device and only using Bluetooth transmission when a fall has occurred. This is the basis for this fall detection firmware.

Along with this firmware a simple C# console application has been developed which listen for the fall command from the Shimmer.

Section 2 and 3 describes the architecture and design of the firmware so an overview can be obtained quickly. Section 4 is about how to test the firmware along with expected outputs and section 5 explains the key areas of the code and how to replace the current fall detection algorithm.

2. FIRMWARE ARCHITECTURE

The software architecture of the fall detection firmware is a combination of layered and module based and is illustrated in Figure 2.1. Each element is explained in the following.

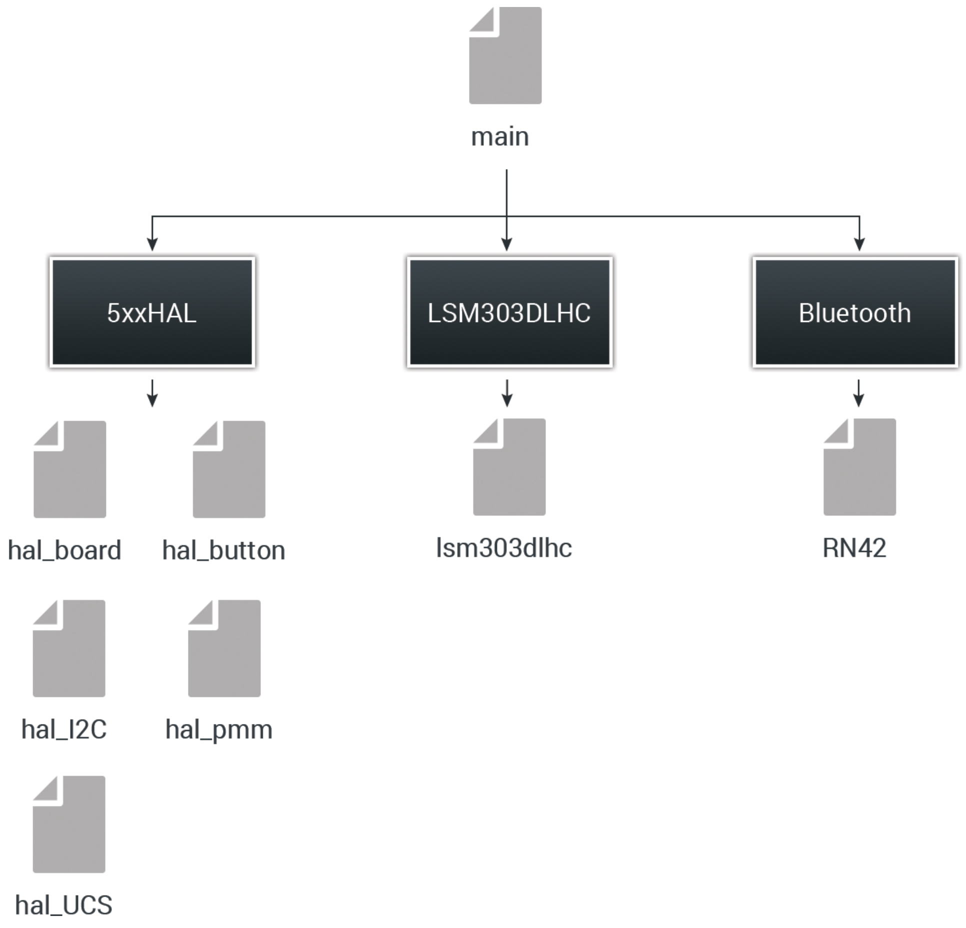


Figure 2.1 – Software architecture of fall detection firmware

Main
The main file is responsible for initializing all external components and controlling timers. Due to limited data processing all of this has also been placed in the main file.

Bluetooth
This is a driver for the RN42 Bluetooth module.

LSM303DLHC
This is a driver for the digital LSM303DLHC wide range accelerometer.

5xxHAL
A collection of various helper files. Hal_board is used to setup the board and control LEDs. Hal_button is a driver for the button. Hal_I2C is a driver for the I2C communication. Hal_pmm controls the power consumption (Power Management Module). Hal_UCS is driver for the timers (Unified Clock System).

3. FIRMWARE DESIGN

An overview of the fall detection algorithm is illustrated in the activity diagram in Figure 3.1.

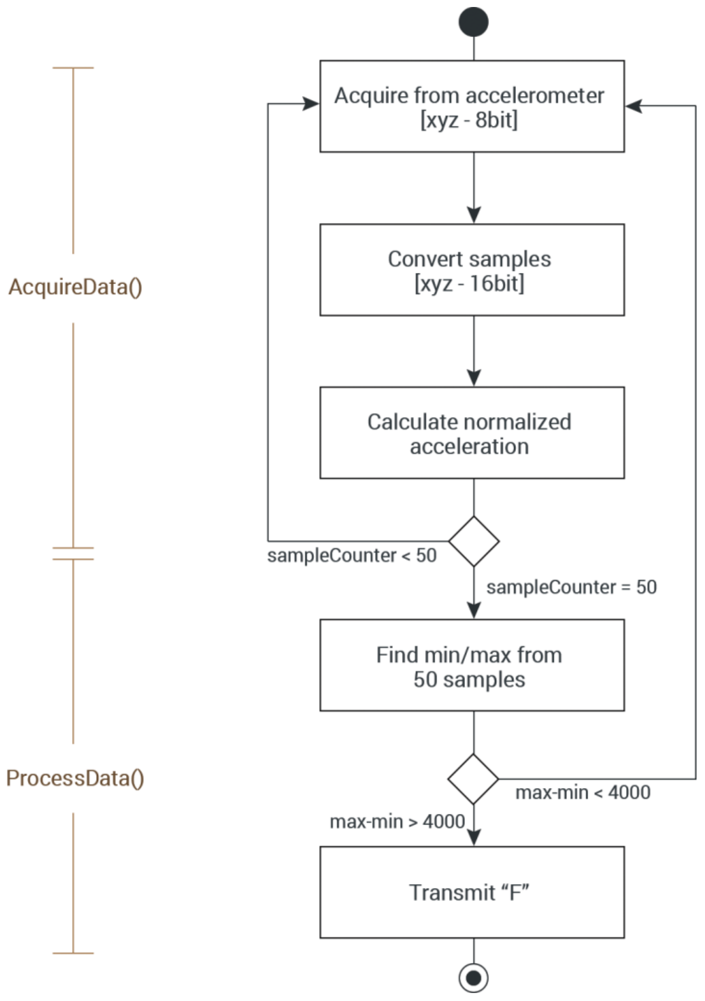

Acquire from accelerometer [xyz - 8bit]
Convert samples [xyz - 16bit]
Calculate normalized acceleration
sampleCounter < 50
sampleCounter = 50
Find min/max from 50 samples
max-min < 4000
max-min > 4000
Transmit "F"
AcquireData()
ProcessData()

Figure 3.1 – Activity diagram of the fall detection algorithm

The digital LSM303DLHC accelerometer provides six 8bit values as the acceleration of the x, y and z axis. The structure is as follows.

$$[X\_LSB\ X\_MSB\ Y\_LSB\ Y\_MSB\ Z\_LSB\ Z\_MSB]$$

This needs to be turned into three 16bit values so that data processing can be applied later on. When this is done the normalized acceleration is calculated using the following formula

$$norm\_acc = \sqrt{X^2 + Y^2 + Z^2}$$

Samples are acquired 50 times per second. When the 50 samples have been acquired the minimum and maximum sample value is found. The difference between these is compared to a fixed threshold value of 4000. If the difference is higher an "F" is transmitted via Bluetooth otherwise the samples are cleared and the acquisition starts over.

4. TESTING THE FIRMWARE

To test the firmware, follow these steps.

1. If the Shimmer3 is not added as unit on your computer then follow the instructions in this link http://windows.microsoft.com/en-us/windows7/add-a-bluetooth-enabled-device-to-your-computer

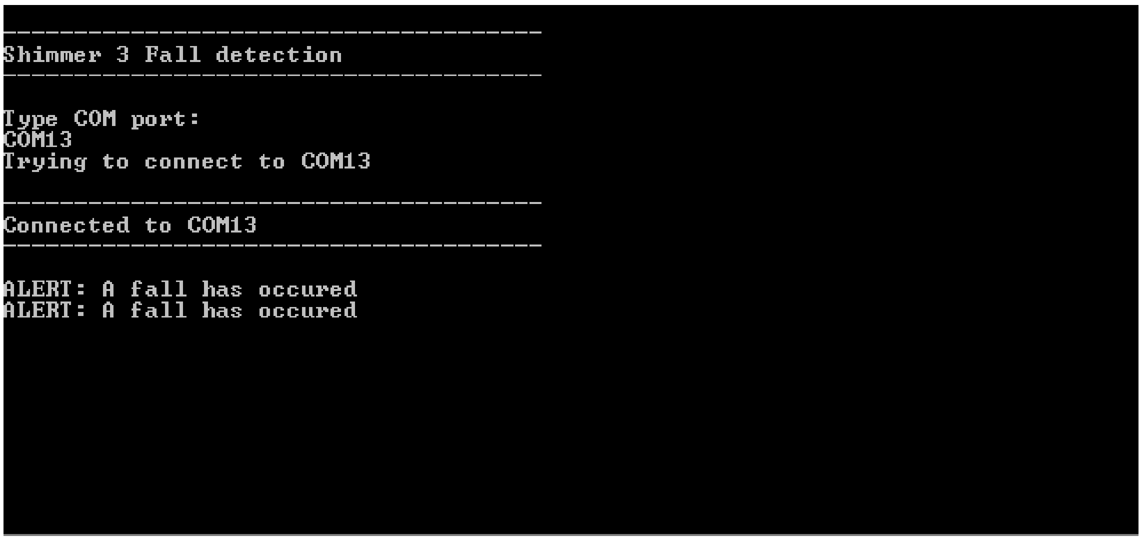
2. Load the bootstrap “Shimmer3_FallDetection.txt onto the Shimmer. The file is located in the folder “Resources/Firmware/”.
   (How to load custom firmware onto the Shimmer is described in tutorial 1)

3. Launch the PC application “Shimmer3_PCApp” located in the folder “Resources/PC Application/Shimmer3_PCApp”

4. Enter the COM port of the Shimmer Bluetooth connection (i.e. COM13)

5. The program should display the text “`Connected to COMxx`” and the blue LED on the Shimmer should be on. If the program keeps displaying the message “`Unable to connect to specified COM port`” try reload the firmware on the device.

6. Try to shake the Shimmer. The program should now display the text “`Alert: A fall has occurred`”.



```
----------------------------------------
Shimmer 3 Fall detection
----------------------------------------

Type COM port:
COM13
Trying to connect to COM13

----------------------------------------
Connected to COM13
----------------------------------------

ALERT: A fall has occured
ALERT: A fall has occured
```

Figure 4.1 - The program output at successful connection and detection

5. MODIFYING THE FIRMWARE

If the current fall detection algorithm seem insufficient according to a specific usage it is relatively easy to modify. There are three key areas which can be modified which are explained in the following.

5.1. Constants

Line 31-34 in the firmware:

```
// Defines
#define NUMBER_OF_SAMPLES                       50
#define FALL_THRESHOLD                          4000
#define SAMPLE_FREQUENCY                        50
```

The NUMBER_OF_SAMPLES constant determines how many samples to acquire before applying the fall detection algorithm.

The FALL_THRESHOLD is fixed limit used in the algorithm to determine when a fall has occurred. By lowering this value less acceleration is needed in order to determine if a fall has occurred.

The SAMPLE_FREQUENCY determines how many samples are acquired each second.

5.2. Initialization of accelerometer

Line 92-94 in the firmware:

```
// Setup of LSM303DLHC
LSM303DLHC_init();
LSM303DLHC_accelInit(LSM303DLHC_ACCEL_100HZ,ACCEL_16G,0,1);
```

The accelerometer can be initialized with different sampling frequencies and resolutions, which is passed through the first two parameters of the “LSM303DLHC_accelInit” function.

The first parameter (sampling rate) can be one of the following:

- *LSM303DLHC_ACCEL_POWER_DOWN*
- *LSM303DLHC_ACCEL_1HZ*
- *LSM303DLHC_ACCEL_10HZ*
- *LSM303DLHC_ACCEL_25HZ*
- *LSM303DLHC_ACCEL_50HZ*
- *LSM303DLHC_ACCEL_100HZ*
- *LSM303DLHC_ACCEL_200HZ*
- *LSM303DLHC_ACCEL_400HZ*
- *LSM303DLHC_ACCEL_1_620KHZ*
- *LSM303DLHC_ACCEL_1_344kHz*

The resolution (sensitivity of the accelerometer) can be one of the following:

- *ACCEL_2G*
- *ACCEL_4G*
- *ACCEL_8G*
- *ACCEL_16G*

5.3. Fall detection algorithm

The fall detection algorithm can easily be replaced. It is located in the function “ProcessData” and is called whenever the specified amount of acceleration samples has been acquired.

Line 210-232 of the firmware:

```
void ProcessData(void) {
    float min = 0, max = 0;

    // Find minimum and maximum of the sample buffer
    min = sampleBuff[0];
    max = sampleBuff[0];
    for (i = 1; i < NUMBER_OF_SAMPLES; i++) {
            if (sampleBuff[i] < min)
                    min = sampleBuff[i];
            if (sampleBuff[i] > max)
                    max = sampleBuff[i];
    }

    // Check if a fall has occurred
    if ((max - min) > FALL_THRESHOLD) {
            if (btIsConnected) {
                    BT_write("F", 1);
            }
    }

    // Reset sample counter
    sampleCounter = 0;
}
```

The acquired acceleration samples are stored in the array called “sampleBuff” (16 bit values). When replacing the body of this function remember to keep the last line which resets the sampleCounter.