**SHIMMER 3**

Basic fall detection



Written by Steffan Lildholdt (steffan@lildholdt.dk)

**TABLE OF CONTENT**

[1. Introduction 1](#_Toc409594924)

[2. Firmware architecture 2](#_Toc409594925)

[3. Firmware design 3](#_Toc409594926)

[4. Testing the firmware 4](#_Toc409594927)

[5. Modifying the firmware 5](#_Toc409594928)

[5.1. Constants 5](#_Toc409594929)

[5.2. Initialization of accelerometer 5](#_Toc409594930)

[5.3. Fall detection algorithm 6](#_Toc409594931)

# 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.

# 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.

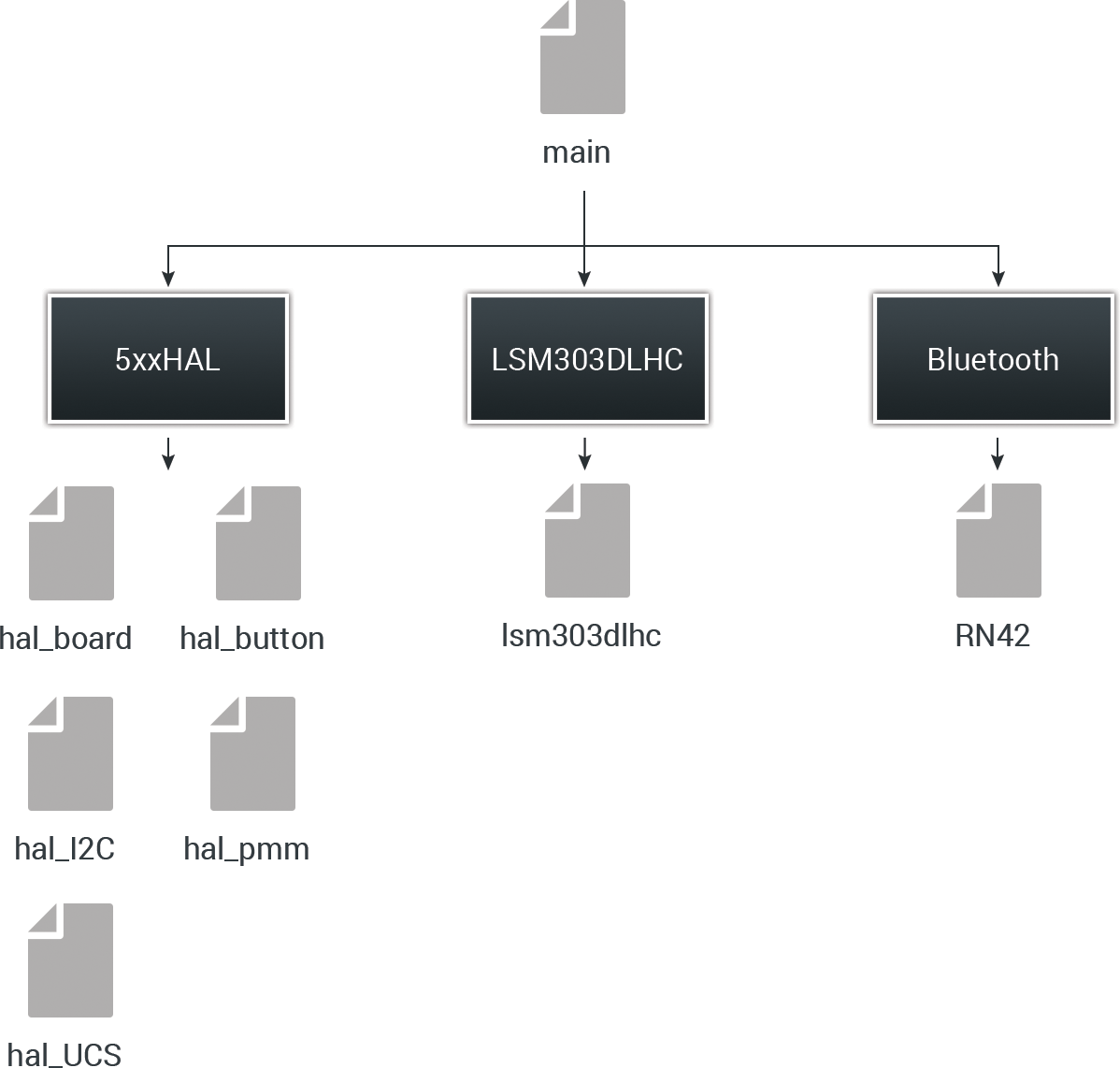


Figure 2.1 – Software architecture of fall detection firmware

**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).

# Firmware design

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

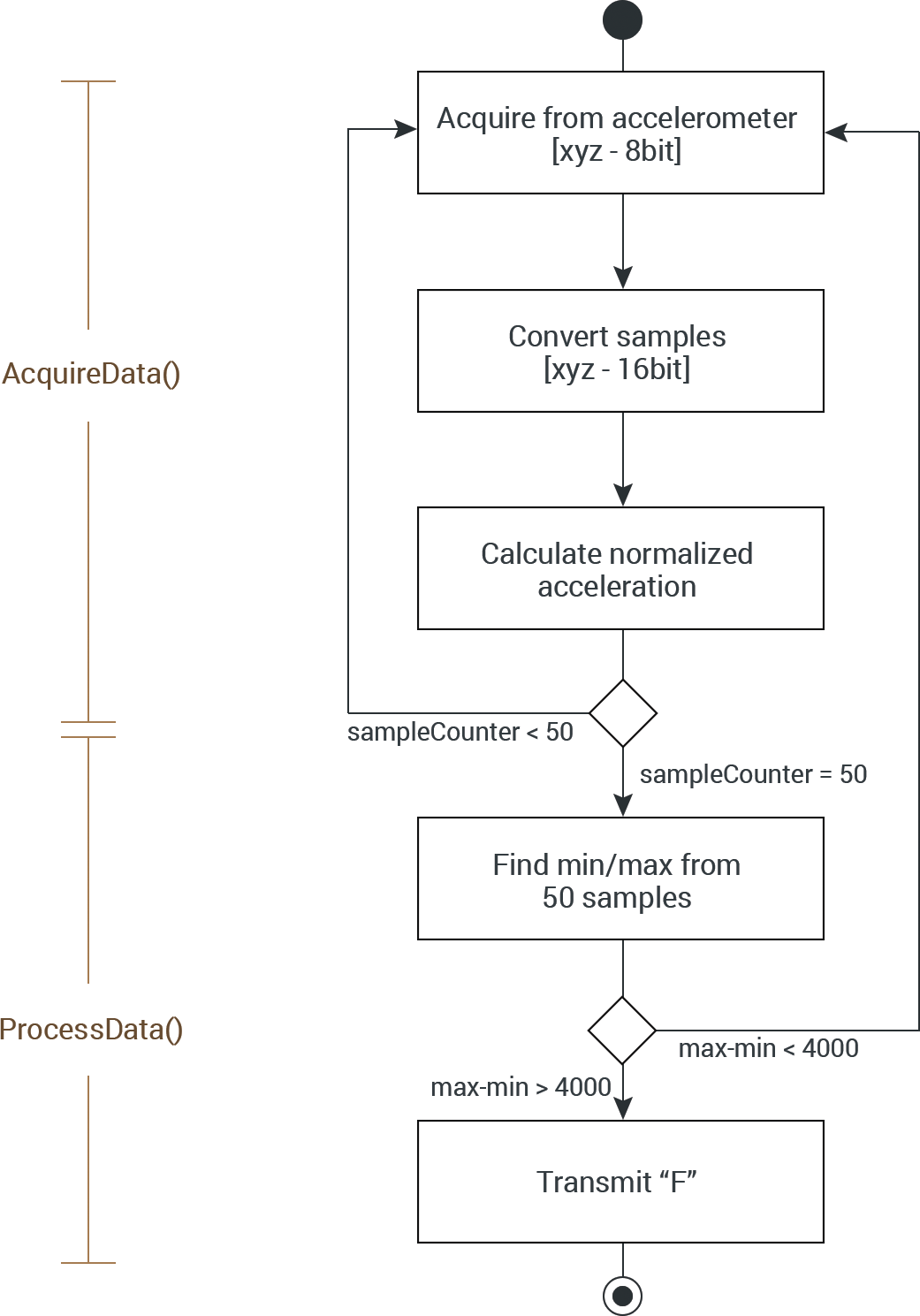


Figure 3.1 – Activity diagram of the fall detection algorithm

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

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

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.

# 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”.

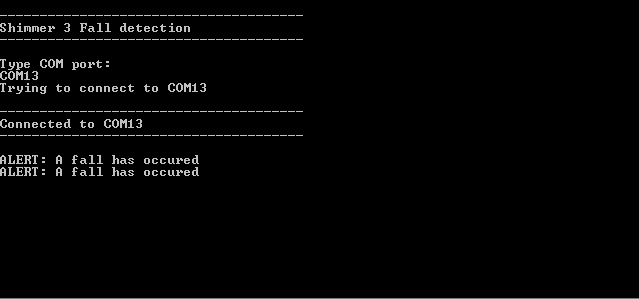


Figure 4.1 - The program output at successful connection and detection

# 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.

## Constants

Line 20-24 in the code:

// 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.

## Initialization of accelerometer

Line 81-83 in the code:

// 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*

## 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 203-225 of the code:

**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.