SHIMMER 3

Basic pedometer

Written by Steffan Lildholdt (steffan@lildholdt.dk)

Version 1.0

TABLE OF CONTENT

1. Introduction 1
2. Firmware architecture 2
3. Firmware design 3
4. Testing the firmware 5
5. Modifying the firmware 6
  5.1. Constants 6
  5.2. Initialization of accelerometer 6
  5.3. Pedometer algorithm 7

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_Pedometer is an example of customized firmware.

A pedometer is a device which is able to count each step a person takes by detecting the motion of the body. It can be a motivation for people wanting to increase their physical activity but is also widely used in rehabilitation scenarios. Many commercial available pedometers undercount steps especially at slower speeds which often are the case when doing rehabilitation.

In order to read the steps taken the C# console application should be running on a PC. When this firmware is loaded onto a Shimmer device and the Shimmer is placed in the dock (connected to power, i.e. a computer), the Shimmer will automatically connect to the computer, transmit the number of steps taken and close the connection again.

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.

2. FIRMWARE ARCHITECTURE

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

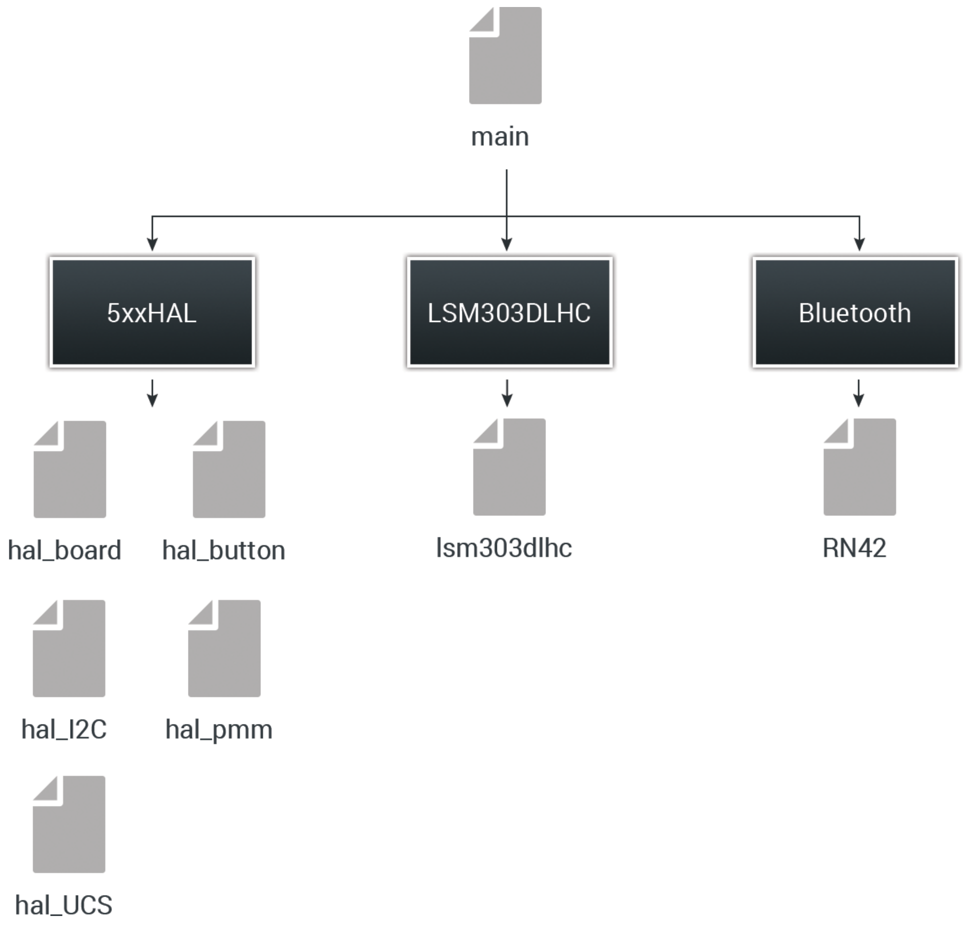


Figure 2.1 – Software architecture of fall detection firmware

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.

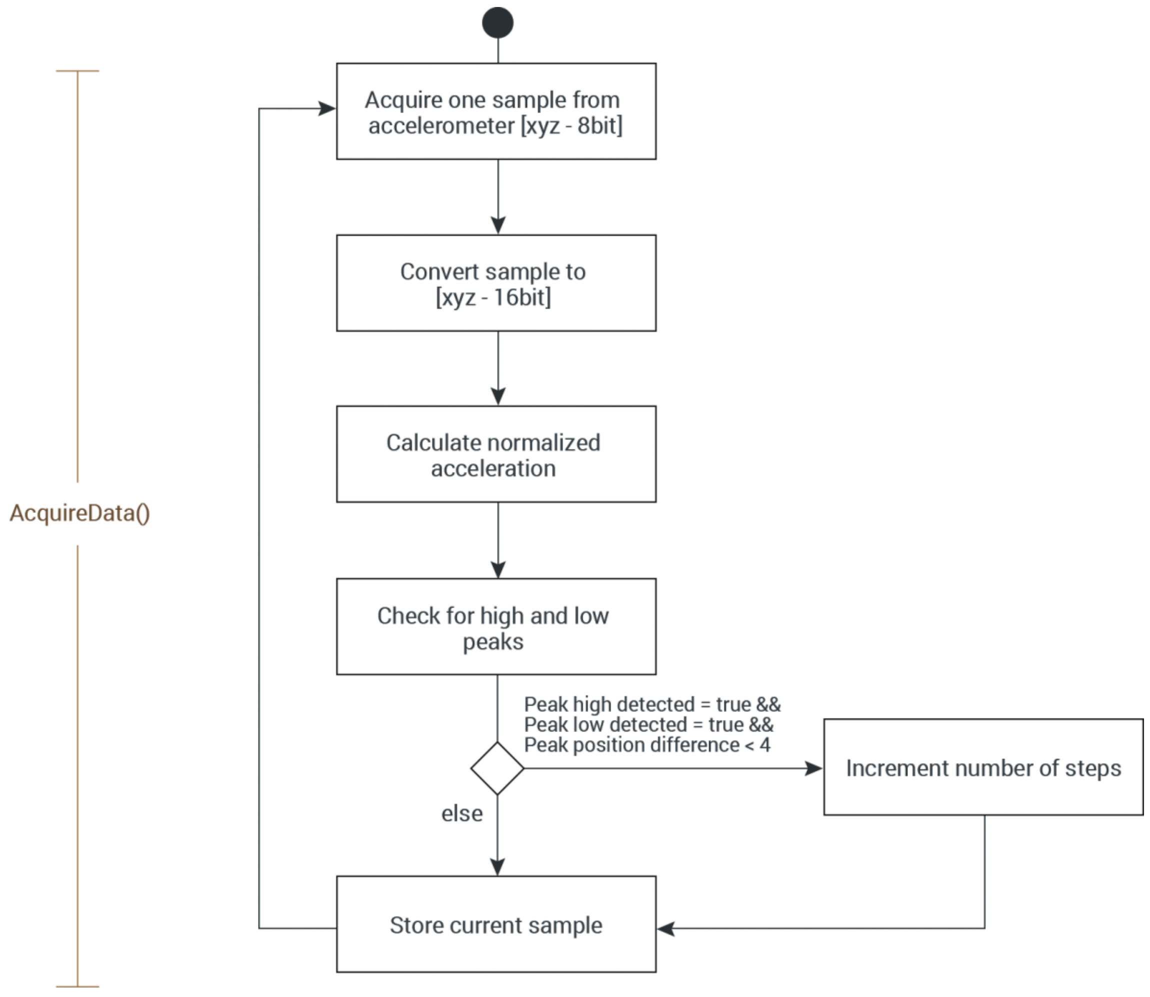
Bluetooth
This is a driver for the RN42 Bluetooth module.

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_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 pedometer algorithm is illustrated in the activity diagram in Figure 3.1.



Acquire one sample from accelerometer [xyz - 8bit]
Convert sample to [xyz - 16bit]
Calculate normalized acceleration
Check for high and low peaks
Peak high detected = true && Peak low detected = true && Peak position difference < 4
Increment number of steps
else
Store current sample
AcquireData()

Figure 3.1 – Activity diagram of the Alert button firmware

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 from the accelerometer are acquired at a rate of 10Hz. By subtracting the previous normalized vector from the current normalized vector over a given amount of time a characteristic as the one shown in Figure 3.2 will be generated.

Three parameters are used to determine if a step has taken place; peak high detected, peak low detected and peak position difference. A high peak should be followed by a low peak and the time difference between these peaks should be lower than a predetermined threshold (4 samples). If this is fulfilled the number of steps is incremented.

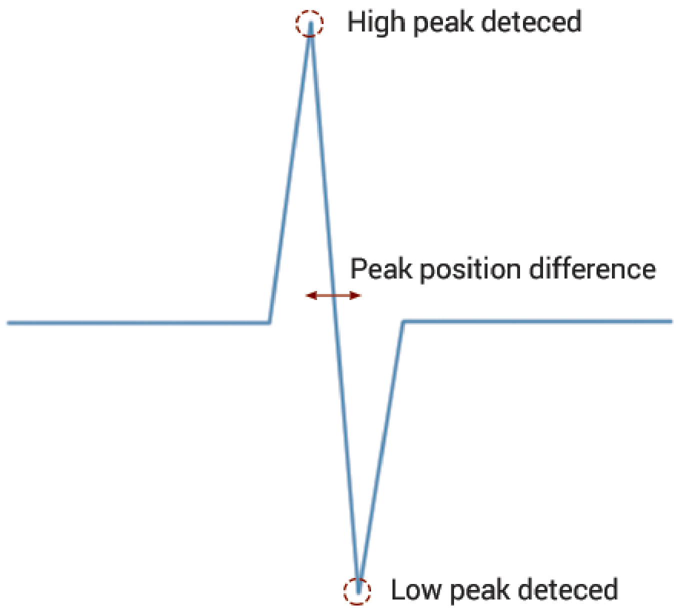


Figure 3.2 – Illustration of the characteristics of a typical step.

4. TESTING THE FIRMWARE

To test the firmware, follow these steps.

1. Load the bootstrap “Shimmer3_Pedometer.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)

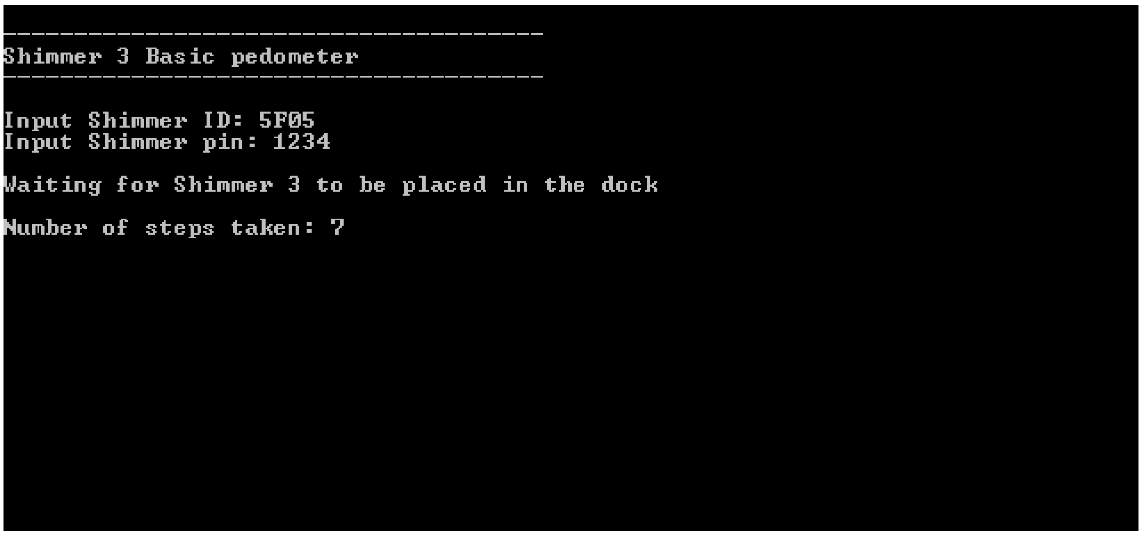
2. Launch the PC application “Shimmer3_PCApp” located in the folder “Resources/PC Application”

3. Type in the Shimmer ID (found on the back of the device)

4. Type in the pin (default: 1234)

5. Place the Shimmer3 in the dock to enable the Bluetooth module

6. The program now locates the Shimmer and receives the number of steps taken.



```
--------------------------------------
Shimmer 3 Basic pedometer
--------------------------------------

Input Shimmer ID: 5F05
Input Shimmer pin: 1234

Waiting for Shimmer 3 to be placed in the dock

Number of steps taken: 7
```

Figure 4.1 - The program output at successful connection of transmission

5. MODIFYING THE FIRMWARE

5.1. Constants

Line 35 in the firmware:

```
#define SAMPLE_FREQUENCY                        10
```

The SAMPLE_FREQUENCY determines how many samples are acquired each second.

5.2. Initialization of accelerometer

Line 108-109 in the firmware:

```
// Setup of LSM303DLHC
LSM303DLHC_init();
LSM303DLHC_accelInit(LSM303DLHC_ACCEL_50HZ,ACCEL_8G,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. Pedometer algorithm

The pedometer algorithm can easily be replaced. It is located in the function "AcquireData" and is called each time a sample from the accelerometer is ready.

Line 266-316 of the firmware:

```
void AcquireData(void) {

// Read sample from LSM303DLHC accelerometer
LSM303DLHC_getAccel(accel_8bit);

// 6 uint8_t are read from the accelerometer
// These are converted into 3 int16_t values
// Structure of accelBuff: [X_LSB X_MSB Y_LSB Y_MSB Z_LSB Z_MSB]
for (i = 0; i < 3; i++) {
     accel_16bit[i] = ((int16_t) accel_8bit[(2 * i) + 1] << 8) |
     (accel_8bit[2 * i] & 0xff);
}

// Calculating normalized acceleration
// sqrt(X^2 + Y^2 + Z^2)
accel_normalized = pow(accel_16bit[0], 2) + pow(accel_16bit[1], 2) +
pow(accel_16bit[2], 2);
accel_normalized = sqrt(accel_normalized);

currentSample = accel_normalized;

// Find high and low peak values
if(lastSample != 0)
{
     if((currentSample - lastSample) > 1000)
     {
            peakHighDetected = 1;
            peakPositionDifference = 0;
     }

     if ((currentSample - lastSample) < -1000)
     {
            peakLowDetected = 1;
     }

     if(peakHighDetected && !peakLowDetected)
     {
            peakPositionDifference++;
     }
```

```
	}

	// Check if a step has been taken
	if(peakLowDetected && peakHighDetected && peakPositionDifference < 4)
	{
		numberOfSteps++;
		peakHighDetected = 0;
		peakLowDetected = 0;
		peakPositionDifference = 0;
		Board_ledToggle(LED_RED);
	}

	lastSample = currentSample;
}
```