Sensobed signal processing using Raspberry Pi

v1.0

Mari Carmen Pardo Martínez
Universidad de Granada
mcpardo@correo.ugr.es

Antonio Javier Pérez Ávila
Universidad de Granada
mraeto@correo.ugr.es

3 October 2020

Abstract

This first version of the report broadly covers the work done for the implementation of Sensobed processing in Raspberry Pi.

Índice

1. Introduction

This project focuses on the sampling and processing of the signals coming from the monitoring system which measures physiological parameters, created in the contract Sensobed2.

2. Hardware

In respect of the hardware elements, an ADC converter (ADS1115 by Texas Instruments) and a Raspberry Pi 4 model B.

- ADS1115: This IC is a 16-bit, I^{2}C-compatible analog-to-digital converter. Its simplified block diagram can be seen in Fig. 1. Since it is able to sample bipolar signals, the real resolution for a positive signal is 15 bits. It includes an output pin called ALERT/RDY, which will be configure as RDY in order to generate a positive pulse every time a conversion is ready to be read, as shown in Figure 2.
- Raspberry Pi 4 Model B: This device is a miniaturized computer which will be in charge of the communication through I^2C protocol with the ADC and of the processing of the transferred samples. Furthermore, the board has communication capabilities such as Bluetooth and Wi-Fi, that will be required later on to sent the results to an Android smartphone.

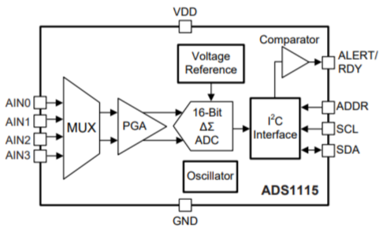


Figura 1: Simplified block diagram of the ADS1115 ADC converter.

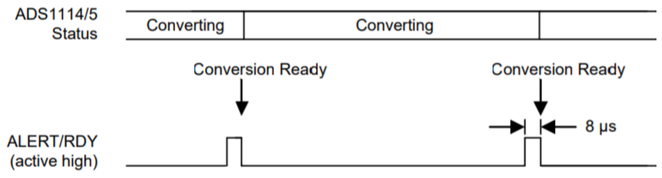


Figura 2: Time diagram showing the conversion-ready pulse generated by the ADS1115 when a conversion is completed.

3. Firmware

3.1. Introduction

The goal of the firmware is to process the breathing and heart signals in time slots in order to refresh the heart and breathing rate every slot time. Inside of every slot time, two functions appear: the signal sampling and the signal processing, which are required to be concurrent.

This idea (shown in Fig. 3) is aimed to sample the signal from the measurement system while the processing of the last slot time is being carried out. In a slot, there will be as well a period of time to prepare the next slot, in which the acquired samples are copied to the processing array. A general flowchart of the firmware is shown in Fig. 4 for the sake of making this idea as straightforward as possible.

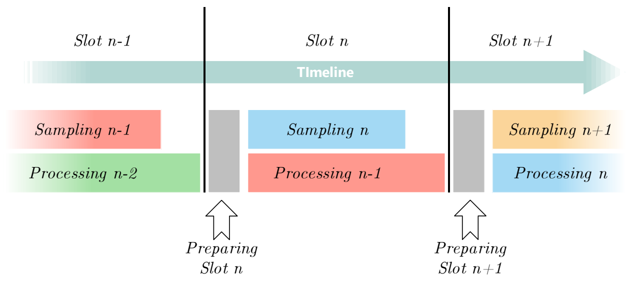


Figura 3: General approach of the methodology employed to sample and process the signal.

- In the initialization, the parameters for the ADC (data rate, gain, I^2C channel) are configured, as well as the variables are declared.
- After that, there is a loop in which two processes take place simultaneously: sampling of a new group of samples in the slot n and processing of the prior group of samples $n-1$. Both these processes will be described in greater depth in the next section.
- Only when both of them come to an end, the slot is considered to be finished so that the samples in n move to $n-1$, n is emptied and a new slot is started.

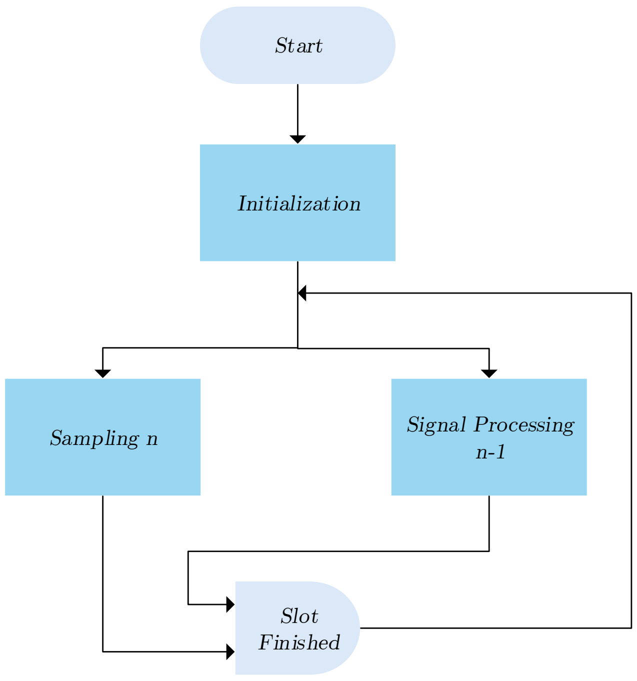


Figura 4: Flowchart representing the firmware.

3.2. Firmware description

The program was written in Python 3.8. This section will be focused on the further description of the firmware itself.

3.2.1. Initialization

The following Python libraries were used:

- RPi.GPIO
- board
- busio
- numpy
- scipy
- Modified ADS1115 Adrafruit Library

The most significant variables declared are the following:

- *n*: Array where the new samples are storaged.
- *n*_1: Array where the samples from the prior time slot are storaged.
- *slot*: Integer that identifies the time slot.
- *conv*: Integer that identifies a conversion inside the current time slot.
- *endSampling*: Boolean variable that turns 1 to stop the sampling.

Apart from the declaration variables, the initialization also includes the ADC configuration and the definition of an ISR (Interrupt Service Routine). In the sampling section, these will be described with more detail.

3.2.2. Sampling

As previously mentioned, sampling was made through the ADC ADS1115. The most relevant parameters used to set up this device (employing a custom modified ADS1115 Adafruit Library) has been the following:

- ADC object creation: firstly, an ADC object from the mentioned library is defined, indicating the $\mathrm{I}^2\mathrm{C}$ channel. In this definition, the output ALERT/RDY has been configured as RDY. This has required the modification of the Adafruit Library.
- Sampling rate: the possible values for this parameter are in the range $[8, 860]$ SPS. The chosen one has been 250 SPS.
- Conversion mode: contrary to single-shot mode, continuous conversion mode has been enabled in order to let the ADC sample continuously at the stated rate.
- ADS gain: this ADC has an integrated PGA and it is used to select the correct FSR (Full Scale Range). From the possible values for this parameter, a FSR of $\pm 6,144$ V has been picked, which needs the ADS1115 to be supplied with 5 V.

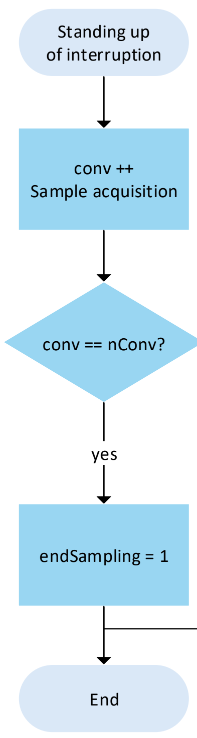
The sampling procedure is drafted on Fig. 5b.

1. Standing up of interuption: first of all, in the slot beginning the bethe interruption is declared and associated with the edge of the signal coming from the output ALERT/RDY of the ADC. Thus, when there is a rising edge in such pin (meaning a conversion is ready) the ISR will be executed.

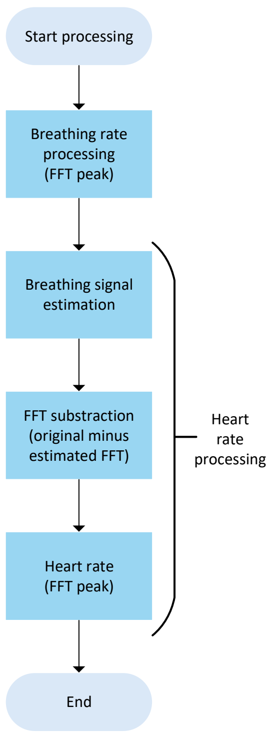
2. Sample acquisition: inside the ISR, a sample is acquired and added in the n array, which will be processed in the next slot. That way, the n array is gradually filled up.

3. Checking of n array fulfillment.

4. Only when the number of conversions $conv$ reaches the limit imposed $nConv$ (indicating the end of the slot) the end sampling flag $endSampling$ is turned on. Otherwise, $endSampling$ keeps its value on 0, meaning the normal flow of interrruptions will take place.



(a) Flowchart representing the sampling method.


Start processing
Breathing rate processing (FFT peak)
Breathing signal estimation
FFT substraction (original minus estimated FFT)
Heart rate (FFT peak)
Heart rate processing
End

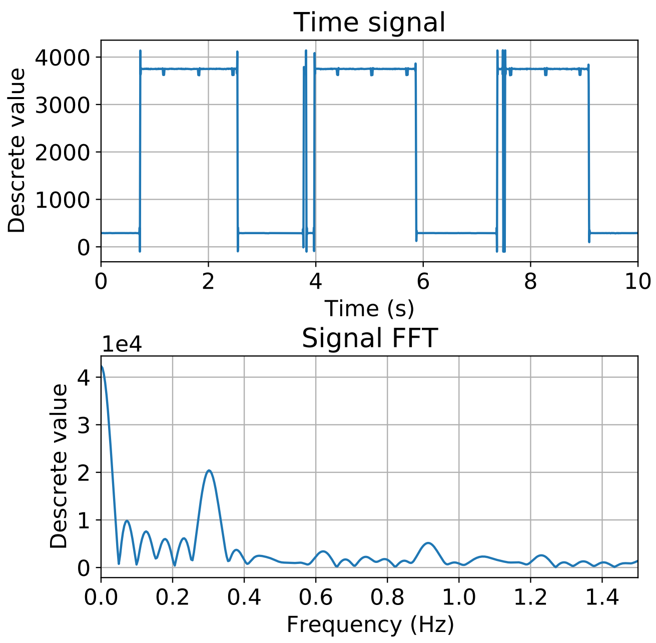
(b) Flowchart representing the processing method.

Figura 5

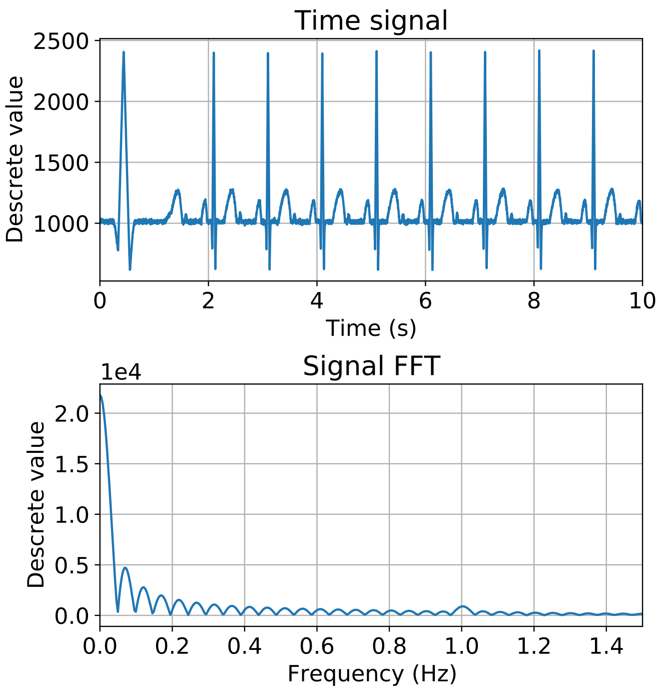
3.2.3. Processing

In this section, the processing part of the firmware will be illustrated. The aim of the processing is to obtain heart and breating rates from the samples acquired via the ADC.

The starting point for the creation of this process was to generate a signal which would resemble the real signal provided from the sensor as much as possible. For that purpose, real breathing and heart signals (shown in Fig. 6a and Fig. 6b) have been used. Furthermore, they have been combined and white gaussian noise was added to the resultant signal, as it can be seen in Fig. 7.



(a) Test breathing signal.

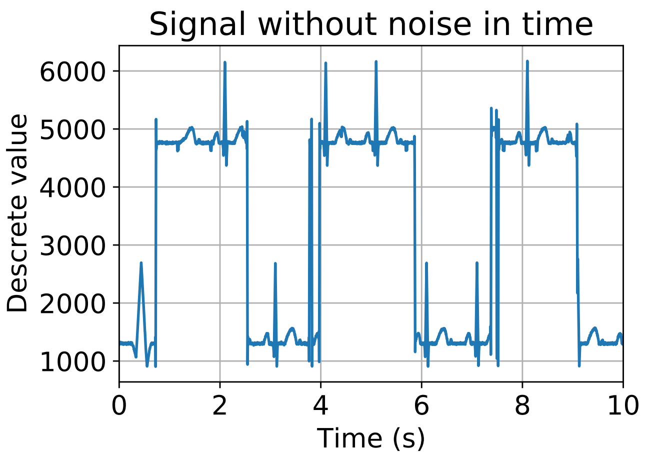


(b) Test heart signal.

Figura 6

Once the complete test signal was created, the signal processing showed in Fig. 5b is applied to it.

1. Breathing rate processing: the signal is filtered with a third order IIR Butterworth band-pass filter made up of SOS (Second Order Section) filters. The upper and lower cutoff frequencies were selected so that the frequencies corresponding to normal breathing rates would not be degraded. As normal sleeping breath rates, the range in $[10, 50]$ breaths per minute was taken. Fig. 8 shows the frequency response of the filter, both in amplitude and phase. By applying this filter to the signal, the result showed in Fig. 9 was



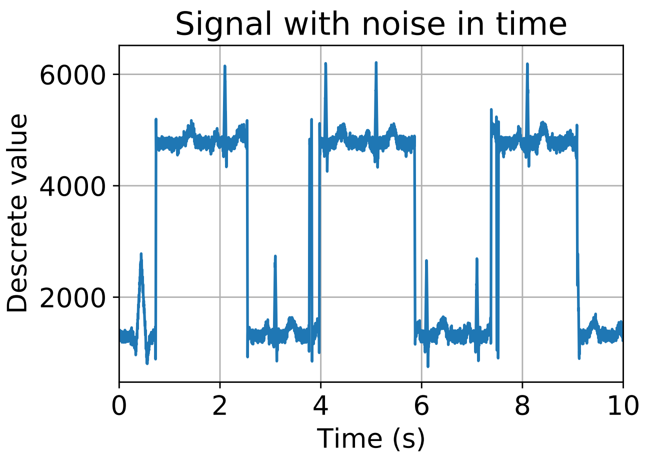
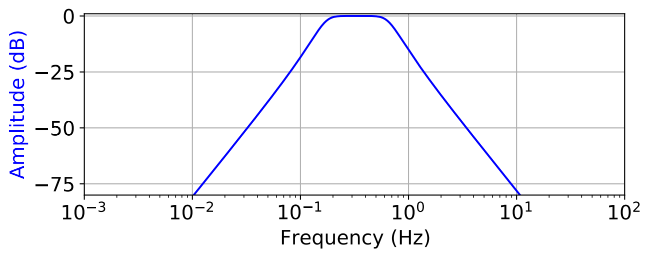


Figura 7: Combination of breathing and heart signals, with and without noise.

achieved.



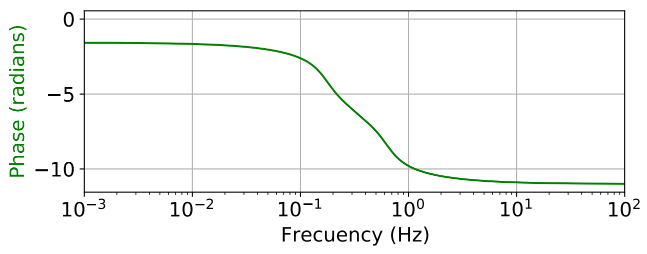
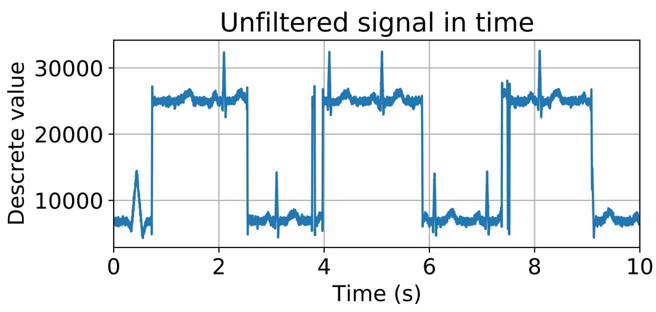
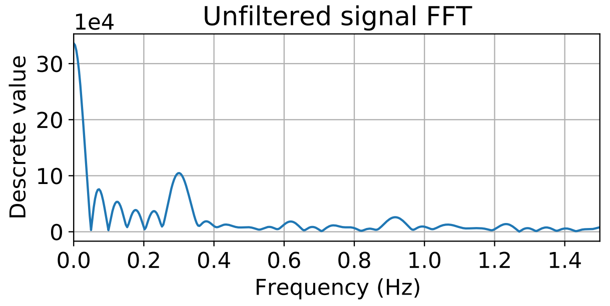


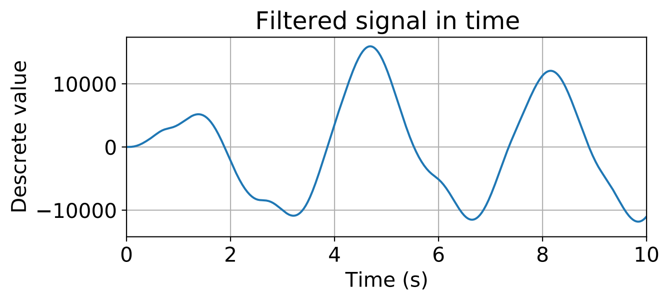
Figura 8: Frequency response of the IIR filter for breathing rate obtainment.

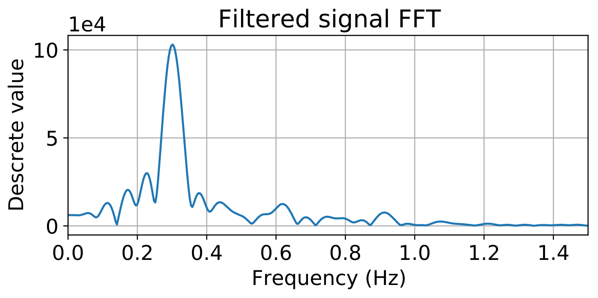
Subsequently, breath rate was figured out from the filtered signal FFT, where a maximum peak in a specific frequency can be easily detected. From the used signal, the peak is found in $f = 0,301$ Hz, which corresponds to a breath rate of $18,082$ Hz.





(a) Unfiltered signal.





(b) Filtered signal.

Figura 9

2. Breathing signal estimation: this is the first step that undertakes heart rate detection. Breathing and heart signals are mixed in the same signal coming from the sensor, being the first much more significant than the second. This fact raised the need to dissociate the two components to obtain the heart rate. The method to meet that need was based on the breathing signal estimation: the unfiltered signal in Fig. 9 is separated in two discrete values to compose the typical rectangular breathing shape, resulting in the signal in Fig. 10.

3. FFT subtraction: once the estimated breathing signal is created, it is possible to take advantage of the FFT linearity by the subtraction of the estimated breathing signal FFT to the complete signal FFT to obtain an estimated heart signal FFT.

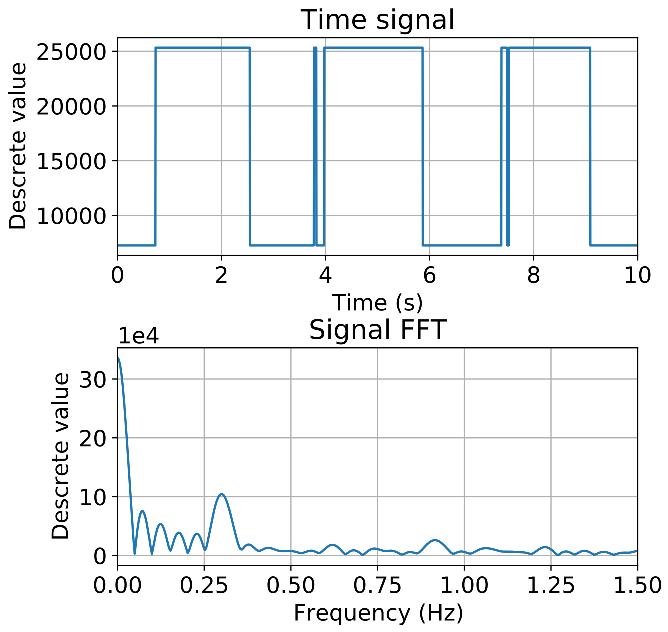


Figura 10: Estimated breathing signal in time and frequency domains.

4. Heart rate: this frequency can be detected by looking for the maximum peak in the estimated heart signal FFT in the frequency interval between 50 and 200 bpm (Beats Per Minute).

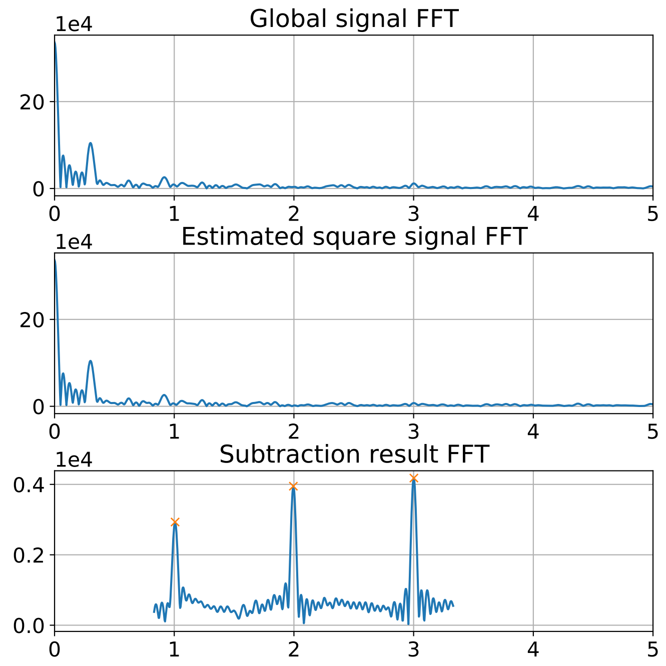


Figura 11: Combination of breathing and heart signals, with and without noise.

Firmware working conditions

The commented signal processing works, provided that the breathing signal estimation was similar to the original breathing signal. So, a big heart rate signal (respectively to breathing one) leads to estimation errors, and therefore, non-coherent FFT subtraction. When the heart rate signal increases over a threshold, the estimation is no longer good enough, henceforth the FFT subtraction point is not going to be successful.