Thread in Motion Project

Name of Project: **CPR Compression Feedback with Accelerometer**

Introduction

Project is mainly about obtaining cardiopulmonary resuscitation (CPR) pulses with accelerometer and by using this acceleration signal generate chest compression depth. Obtained compression depth will be used to give feedback in real time. Also, it tells user when to apply pressure and by that chest compression rate is fixed. In project there are 3 main components which are MCU, accelerometer sensor and a display to show compression in millimeters.

During CPR, the early and persistent application of chest compressions and ventilations to the patient artificially maintains a minimal flow of oxygenated blood. This delays brain damage and generates myocardial blood flow, essential to the restoration of a perfusing rhythm [1]. Because of that a device that can give feedback in real time is very helpful to paramedics or whom do the first aid. Evidences shows that taking feedbacks in time of incident or practicing with feedback enhances quality of chest compressions [2].

Using only acceleration data as a feedback input requires much more computational power because to obtain compression depth from acceleration the one should double integrate the chest acceleration. Therefore, first devices have used force sensors for giving feedback. Nowadays there are accelerometer-based devices and these commercial accelerometer-based feedback systems uses pressure sensors and accelerometers at the same time or uses additional reference signals to fix boundary conditions for the integration process [1]. For protection purposes of force sensors this kind of devices should have to be rigid and bulky, this condition can give damage patient's chest [3]. In contrast implementation of this project can be cheaper, easier to use.

This project uses algorithms in [4] to achieve integrations and can be used in a glove for users. This device can be located in first aid kits etc. With the use of display on the device the user can receive feedback about his/her compression depths and can change his/her pressure. Like in the Figure-1 but instead of a box, there will be a glove with accelerometer.

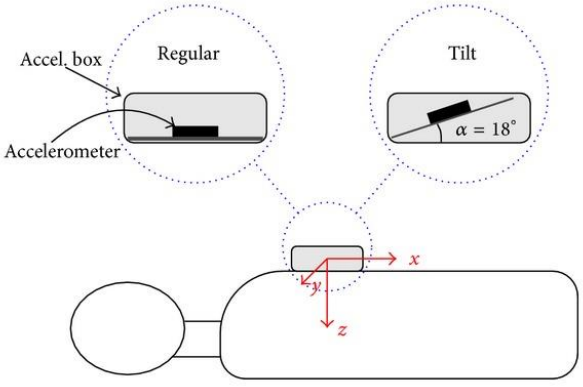


Figure 1 Implementation of device

Selecting Units

MCU

As I mentioned before, there are some computational restricts. Mathematical approach of this project requires fast Fourier transform (FFT) process and for this reason I decided to use an ARM Cortex-M4 based MCU. I think Cortex-M4 will fast enough do computations and show results in real time. As I looked to the FFT performances of STM32 ARM Cortex MCUs it clearly shows that M4 requires less cycles to compute FFT than M3 and M0[5] and M4 cheaper than M7s. Also, there is FPU in M4 and it will be useful because we are using float acceleration data.

For selecting MCU I have checked out NXP LPC540 series and Texas Instruments MSP432 but I could not be sure because I could not that much recourses and examples. Because there are lots of recourses on the internet that can be access easily, I have chosen to use STM32 series MCU that I am familiar with. Maybe this project firstly can be tried on development kit hence I have selected STM32f407vg which is used on commonly sold STM32f4 Discovery Board. Its reference manual can be found at [6].

Sensor

For acceleration sensor I have checked about sensors that can be used. In [7] Analog Devices ADXL330 MEMS sensor so I have chosen much more enhanced version of this accelerometer, ADXL355 which has built-in 20bit ADC and it can be achieved via SPI.

ADXL355 has 25 µg/√Hz noise density so with 20bit ADC its effective bits can be approximately 13-14bits which is perfect for accuracy. Its reference manual can be found at [8].

Display

I have wanted to use a generic display to show processed sensor value and I did not want to use basic 2x16 LCD screen so I have chosen ST7735 TFT display.

Mathematical Approach of Project

I have used mathematical model from [4]. In this model assumes acceleration and compression depth signal are almost periodic. These periodic representations can be modelled by using first N harmonics of their Fourier series decomposition:

$$a(t) = \sum_{k=1}^{N} A_k \cos(2\pi k f_{cc} t + \theta_k) \ \left(\mathrm{m/s^2}\right), \tag{1}$$

$$s(t) = \sum_{k=1}^{N} S_k \cos(2\pi k f_{cc} t + \phi_k) \ (\mathrm{mm}). \tag{2}$$

In this equations f_{cc} is fundamental frequency. And we know that displacement can be obtained by second integral of acceleration:

$$a(t) = \frac{d^2 s(t)}{dt^2} \tag{3}$$

If we use quasiperiodic approximation using Fourier series transform (3) equation yields the following relations between the amplitudes and phases of their harmonics

$$S_k = \frac{1000 A_k}{(2\pi k f_{cc})^2}, \qquad \phi_k = \theta_k + \pi, \quad \text{for} \quad k = 1,2,\ldots,N. \tag{4}$$

These equations can be used to reconstruct *s(t)* once f_{cc}, A_k, and θ_k are obtained from the acceleration signal.

In this project I have used 2 seconds of acceleration data window and first 3 harmonics like in the [4] article. Therefore, I can calculate rate and depth like this:

$$\begin{aligned} \text{rate}\ (\text{min}^{-1}) &= 60 \cdot f_{cc}, \\ \text{depth}\ (\text{mm}) &= \max\{s(t)\} - \min\{s(t)\}. \end{aligned} \tag{5}$$

I do not calculate rate because I use fixed 120 cpm rate to easily calculate timers.

Implementation of Project

Firstly, I have chosen to use 2 Timers (TIM1, TIM3) and 2 timer interrupts. First timer is used for toggling second timer so second timer sets acceleration reading flags. Acceleration data should be obtained in a high frequency. Thus, second timer interrupt occurs in a higher frequency than first timer interrupt. By using first timer interrupt, we can determine CPU's and sensor's sleep interval.

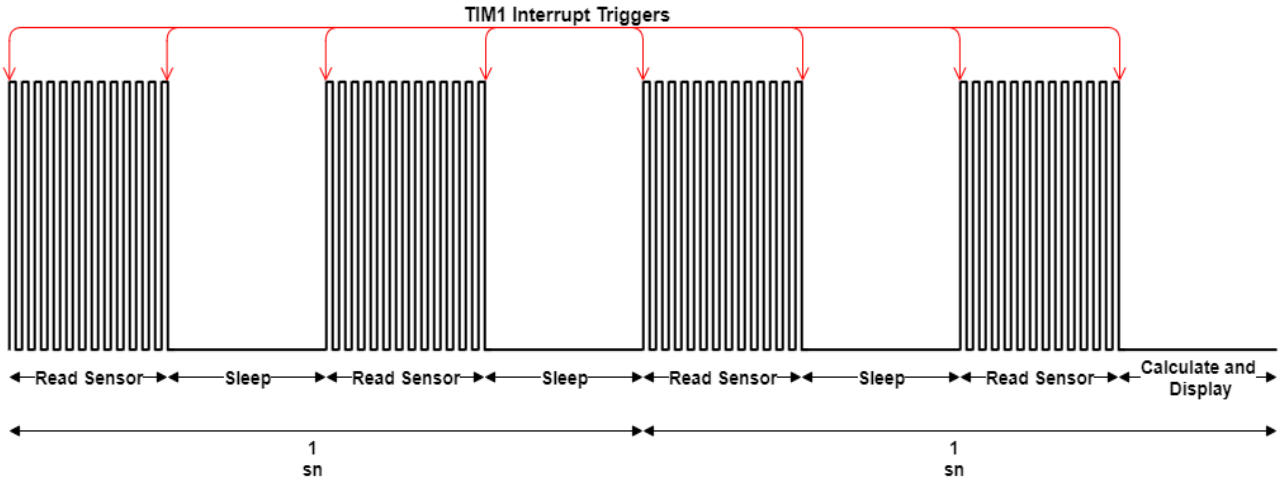


Figure 2 Timing diagram of project

Figure-2 briefly shows that when timer interrupts are occurred and when CPU goes to sleep. TIM1 interrupt triggers 4 times in 1 second (every 250ms) and toggle TIM3, so TIM3 interrupt triggers only in "Read Sensor" interval as it can be seen in the Figure-2. TIM3 interrupt triggers 1050 times in 1 second (every 952 us) if it is set.

TIM1 runs on APB2 bus and its clock frequency is 168Mhz, TIM1 runs on APB1 bus and its clock frequency is 84Mhz. "Pre-scaler" and "Period" values are calculated according to that.

Eventually, after 2 seconds we will have 262 * 4 = 1048 samples to do 1024 samples FFT and calculate compression depth.

As you can see in the Figure-2 there is only 250 milliseconds for CPU to sleep. I took the idea of sleeping for such a short interval from example in [9]. In this application note example CPU and the sensor are in sleep mode for almost 180 milliseconds.

Every time when device goes into read sensor mode it set a GPIO output pin with a LED or a buzzer user can know when to apply pressure.

Sensor driver uses HAL API SPI driver and I when I am writing sensor driver, I used demo example as a reference which is developed by Analog Devices for their own EVAL-ADICUP360 evaluation board in [10]. I have made driver compatible with STM32f4, changed some of the functions, written new functions and added structs using ADXL355 Datasheet to make it more efficient and modular. Communication with ADXL355 sensor is made with Full-Duplex SPI (Master STM32F4, Slave ADXL355).

After 2 seconds of reading process is completed input array which contains acceleration data is used for calculations. FFT step of this calculations executed by ARM CMSIS DSP library. In application note [5] it can be seen that ARM CMSIS DSP library is one of the fastest approaches and I this application note and library explanations in [11] while using the library and creating "CPR_Math.c".
When calculation process is over compression depth in millimeters send to the ST7735 display to show user. I have used ST7735 library from [12] and deleted unnecessary functions for this project and I have checked and compared with Waveshare demo code in [13] just to be sure that the library is correct. Communication with ST7735 Display is made with Half-Duplex SPI (Master STM32F4, Slave ST7735).
I did not want to use sprint function since it is very slow and takes too much space in memory. Hence, for showing the result value in display, I have used "float to string" example in [14] as a reference and I have enhanced that function to use. It can be seen in the "main.c" file.

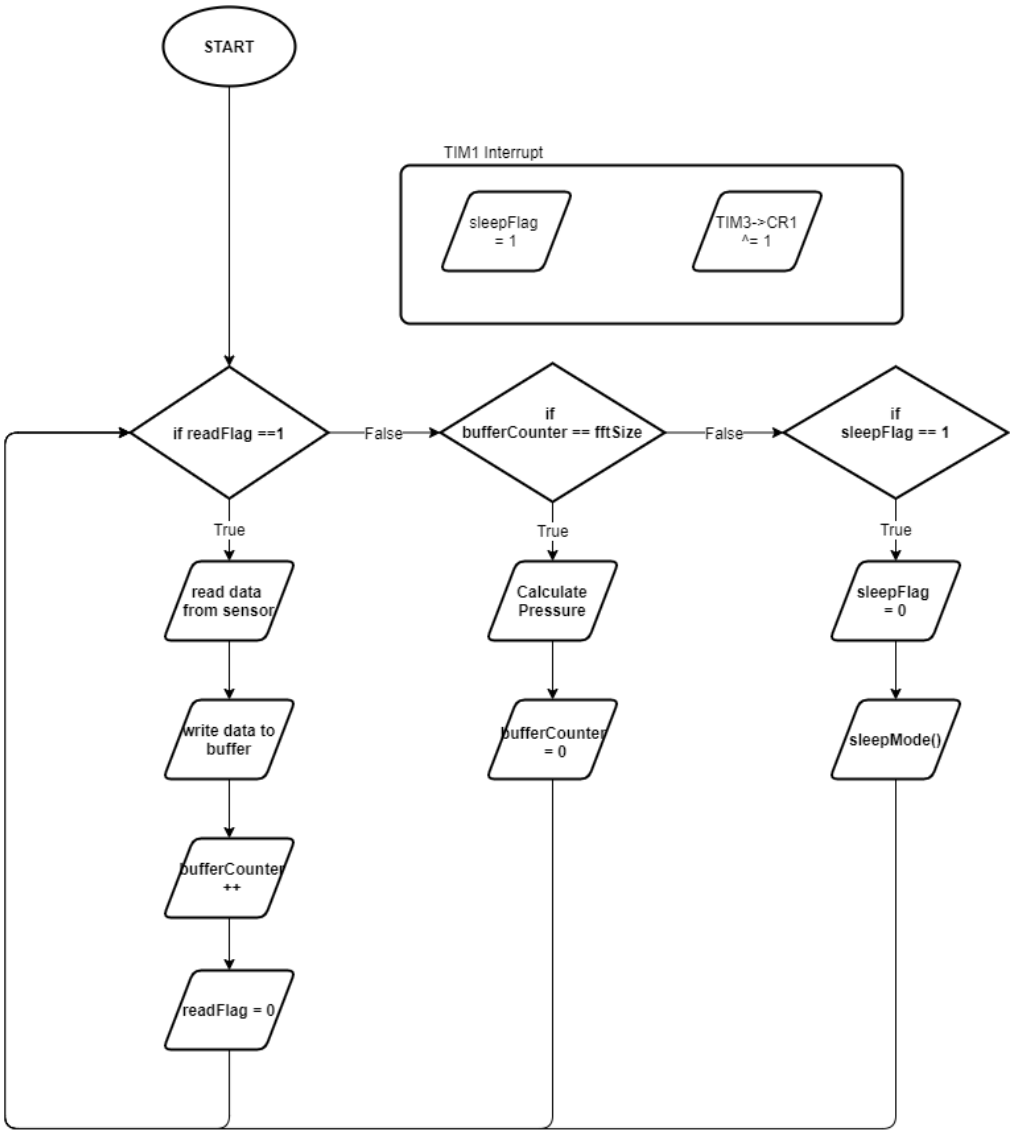


Figure 3 Flowchart

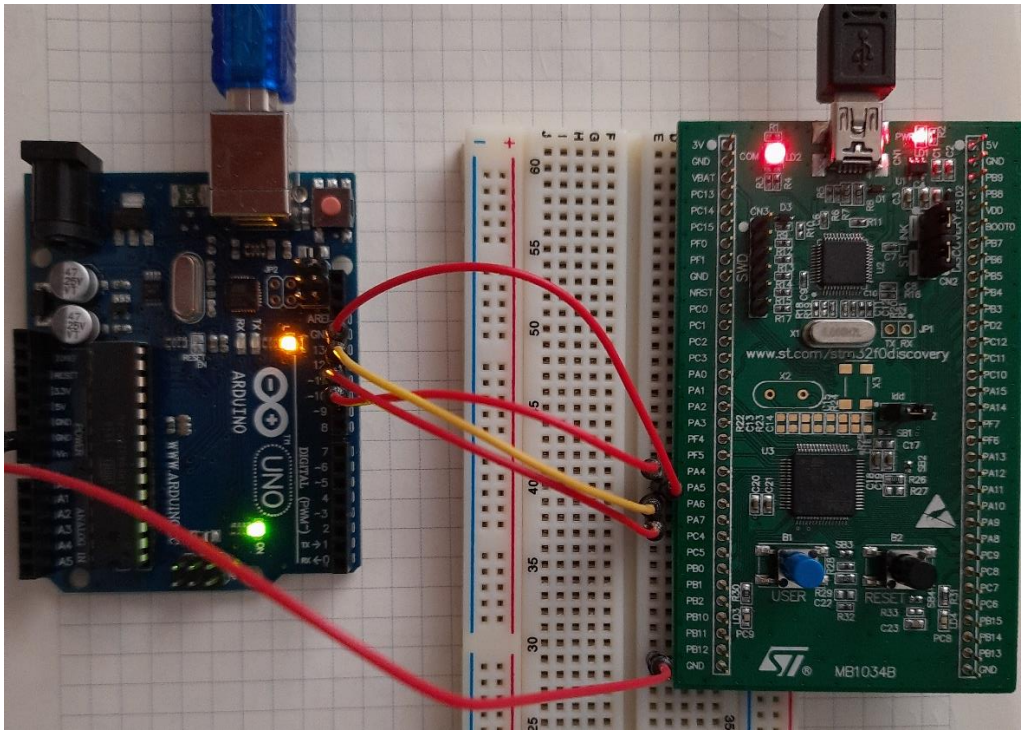
Testing on STM32F0 Discovery Board

As I do not have any of the main components for this project, I have created another project and I used my STM32F0 Discovery Board for testing the main code.

First of all, I tested “PWR” functions in HAL API to sleep and I checked it with toggling onboard LEDs. So, I can say that “Sleep_Mode” function works perfectly fine with mentioned intervals. Of course, I had to change Timer “Pre-scaler” and “Period” values for STM32f0 clock frequency to obtain same intervals.

Secondly, I generated a cosine function in “Read Sensor” block and calculations are done with that cosine functions. I checked every calculated value like fundamental frequency, phase angle and they were all true.

Finally, I used an Arduino UNO to generate cosine function and send it with SPI connection. Master STM32F0 send a dumb value and take cosine value from Slave Arduino UNO. Test worked perfectly find and I took same results as generated cosine in STM32f0.



References

[1] González-Otero DM, Ruiz JM, Ruiz de Gauna S, et al. Monitoring chest compression quality during cardiopulmonary resuscitation: proof-of-concept of a single accelerometer-based feedback algorithm. PLoS One 2018;13:e0192810.

[2] Abella BS, Edelson DP, Kim S, Retzer E, Myklebust H, Barry AM, et al. CPR quality improvement during in-hospital cardiac arrest using a real-time audiovisual feedback system. Resuscitation. 2007;73(1):54–61. doi: 10.1016/j.resuscitation.2006.10.027

[3] Gruber J, Stumpf D, Zapletal B, Neuhold S, Fischer H. Real-time feedback systems in CPR. Trends in Anaesthesia and Critical Care. 2012;2(6):287–294. doi: 10.1016/j.tacc.2012.09.004

[4] González-Otero DM, Ruiz JM, Ruiz de Gauna S, Irusta U, Ayala U, Alonso E. A new method for feedback on the quality of chest compressions during cardiopulmonary resuscitation. BioMed research international. 2014;2014.

[5] Digital signal processing for STM32 microcontrollers using CMSIS
link: st.com/resource/en/application_note/dm00273990-digital-signal-processing-for-stm32-microcontrollers-using-cmsis-stmicroelectronics.pdf

[6] STM32F407 Reference Manual
link: st.com/resource/en/reference_manual/dm00031020-stm32f405415-stm32f407417-stm32f427437-and-stm32f429439-advanced-armbased-32bit-mcus-stmicroelectronics.pdf

[7] Ruiz de Gauna S, González-Otero DM, Ruiz J, Gutiérrez J, Russell JK. A Feasibility Study for Measuring Accurate Chest Compression Depth and Rate on Soft Surfaces Using Two Accelerometers and Spectral Analysis. BioMed research international. 2016;2016 doi: 10.1155/2016/6596040

[8] ADXL355 Datasheet
link: analog.com/media/en/technical-documentation/data-sheets/adxl354_355.pdf

[9] Using STM32F4 MCU power modes with best dynamic efficiency
link: st.com/resource/en/application_note/dm00096220-using-stm32f4-mcu-power-modes-with-best-dynamic-efficiency-stmicroelectronics.pdf

[10] ADXL355 demo code
link: github.com/analogdevicesinc/EVAL-ADICUP360/tree/master/projects/ADuCM360_demo_adxl355_pmdz

[11] CMSIS DSP Software Library
link: keil.com/pack/doc/CMSIS/DSP/html/

[12] ST7735 reference code
link: github.com/afiskon/stm32-st7735

[13] ST7735 LCD module page
link: waveshare.com/wiki/1.8inch_LCD_Module

[14] Stack Overflow answer
link: stackoverflow.com/a/56173153