Heart Rate Monitoring with Emotion Detection using PPG & GSR Sensors

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

EEE 426
BIOMEDICAL INSTRUMENTATION LABORATORY

REPORT ON

Heart Rate Monitoring with Emotion Detection using PPG & GSR Sensors

Submitted to:-

I. K. M. Reaz Rahman

Lecturer, Dept. of EEE, BUET.

Arik Subhana

Lecturer, Dept. of EEE, BUET

Submitted by:

Shirajum Munira (1406134)

Farhat Binte Azam (1406135)

Imran Fahad (1406137)

Dipannita Ghosh (1406138)

Md. Zesun Ahmed Mia (1406141)

Md. Julkar Nine (1406142)

Md. Istiaq Ansari (1406143)

Mishkatul Islam(1406149)

Forwarding letter

February 5, 2019

I. K. M. Reaz Rahman, Lecturer

Arik Subhana, Lecturer

Department of Electrical and Electronic Engineering

Bangladesh University of Engineering and Technology, Dhaka.

Subject: A report on "Heart Rate Monitoring with Emotion Detection using PPG & GSR Sensors".

Sir,

With due respect we would like submit you a report on "Heart Rate Monitoring with Emotion Detection using PPG & GSR Sensors". Heart diseases are common phenomena all around the world. Most of people suffer from these diseases because they cannot monitor their heart condition regularly. Heart rate monitoring devices are not available to mass people. Therefore, we are developing a low cost and reliable heart rate monitoring device using photoplethysmogram (PPG) sensor. By detecting rapid emotional change of a person, we can notify him for taking safety measures. Galvanic skin response (GSR) sensor can help him detect emotional changes properly.

We would like to thank you for providing us with your suggestions and help us completing the project. We earnestly apologize for the mistakes on our parts and request you to kindly consider such cases and oblige thereby.

Sincerely yours,

Shirajum Munira (1406134)

Farhat Binte Azam (1406135)

Imran Fahad (1406137)

Dipannita Ghosh (1406138)

Md. Zesun Ahmed Mia (1406141)

Md. Julkar Nine (1406142)

Md. Istiaq Ansari (1406143)

Mishkatul Islam(1406149)

Level-4 Term-2

Department of Electrical and Electronic Engineering

Abstract

Heart diseases are common phenomena all around the world. Most of people suffer from these diseases because they cannot monitor their heart condition regularly. Heart rate monitoring devices are not available to mass people. Therefore, we are developing a low cost and reliable heart rate monitoring device using photoplethysmogram (PPG) sensor.

In modern times, emotions of a human being change rapidly. This rapid change is very dangerous for patients with abnormal heart and brain conditions. By detecting rapid emotional change of a person, we can notify him for taking safety measures. Galvanic skin response (GSR) sensor can help him detect emotional changes properly.

Wearable sensors, just as the name implies, are integrated into wearable objects or directly with the body in order to help monitor health and provide clinically relevant data for care. Wearable sensors are increasingly taking part in daily activities, not only because of the recent society health concern, but also due to their relevance in the medical industry. Therefore, we are developing a low cost 'Heart Rate Monitoring with Emotion Detection' wearable device using PPG & GSR Sensors.

Table of Contents

Introduction:

Heart diseases are common phenomena all around the world. Most of people suffer from these diseases because they cannot monitor their heart condition regularly. Heart rate monitoring devices are not available to mass people. Therefore, we are developing a low cost and reliable heart rate monitoring device using photoplethysmogram (PPG) sensor.

In modern times, emotions of a human being change rapidly. This rapid change is very dangerous for patients with abnormal heart and brain conditions. By detecting rapid emotional change of a person, we can notify him for taking safety measures. Galvanic skin response (GSR) sensor can help him detect emotional changes properly.

Wearable sensors, just as the name implies, are integrated into wearable objects or directly with the body in order to help monitor health and provide clinically relevant data for care. Wearable sensors are increasingly taking part in daily activities, not only because of the recent society health concern, but also due to their relevance in the medical industry. Therefore, we are developing a low cost 'Heart Rate Monitoring with Emotion Detection' wearable device using PPG & GSR Sensors.

Components:

1. MAX30100 (PPG sensor)
2. GSR sensor
3. HC-05 Bluetooth module
4. Arduino Uno
5. Resistors
6. Jumper wires
7. Printed circuit board

Software used:

1. Matlab 2018a and
2. Arduino 1.6.8

Here is a short description of the components that we have used:

MAX30100:

The MAX30100 is an integrated pulse oximetry and heartrate monitor sensor solution. It combines two LEDs, a photodetector, optimized optics, and low-noise analog signal processing to detect pulse oximetry and heart-rate signals. The MAX30100 operates from 1.8V and 3.3V power supplies and can be powered down through software with negligible standby current, permitting the power supply to remain connected at all time. Data is stored in a FIFO buffer. It can store up to 16 measurements, where each sample is size of 4 bytes. First two bytes are for IR measurement and last two bytes are for RED measurement.

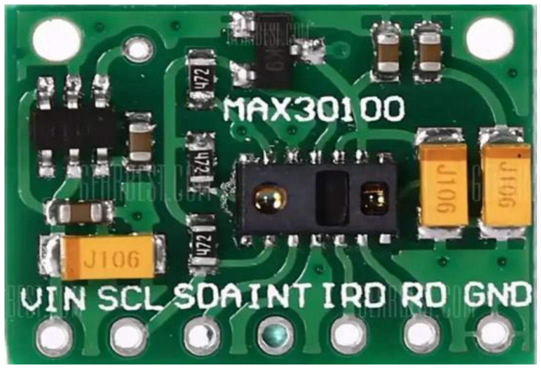

Figure 01: MAX30100 sensor

GSR sensor:

GSR stands for galvanic skin response, is a method of measuring the electrical conductance of the skin. GSR sensor allows us to spot strong emotions by simply attaching two electrodes to two fingers on one hand. Its operating voltage is 3.3-5 volt and input signal is resistance. Nickel is used for finger contact material.

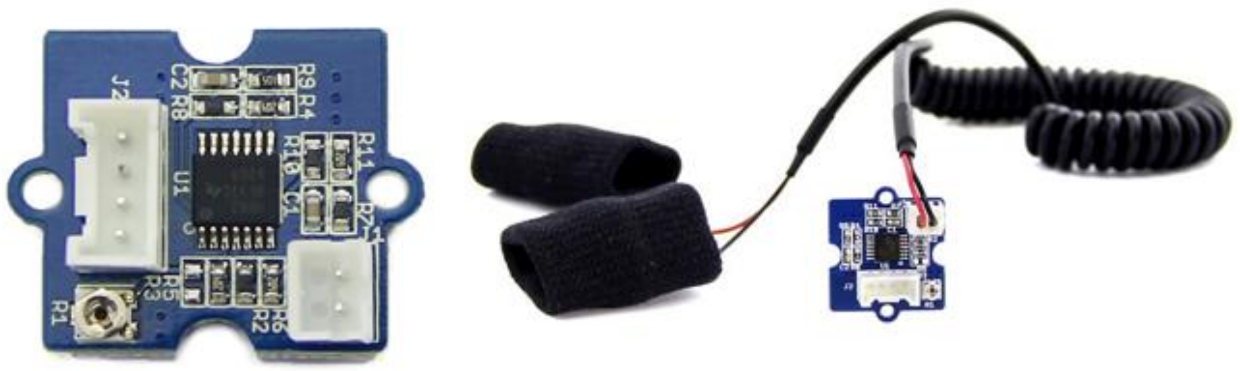

Figure 02: GSR sensor

HC-05 Bluetooth module:

HC-05 Bluetooth Module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. Its communication is via serial communication, which makes an easy way to interface with controller or PC. This module can be used in a master or slave configuration. It has six pins, which are EN, VCC, GND, TXD, RXD and STATIC.

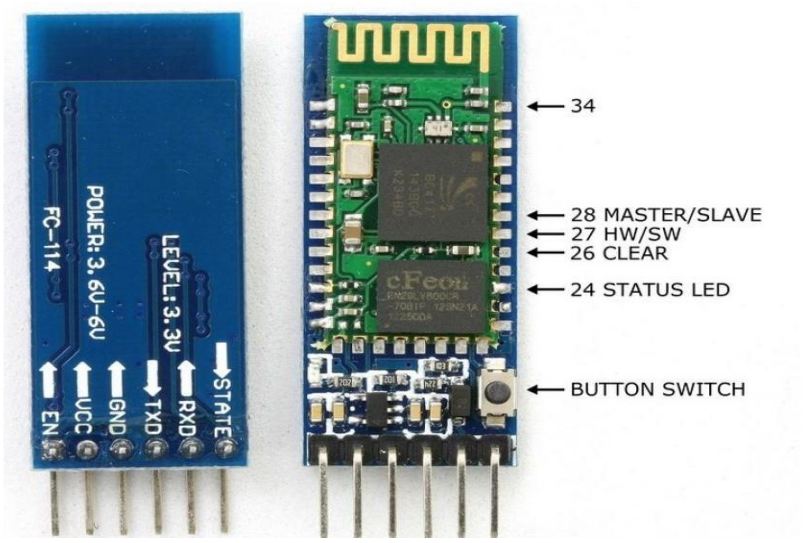

Figure 03: HC-05 Bluetooth module.

To integrate all parts and make a wearable device we have designed PCB. Design is given below

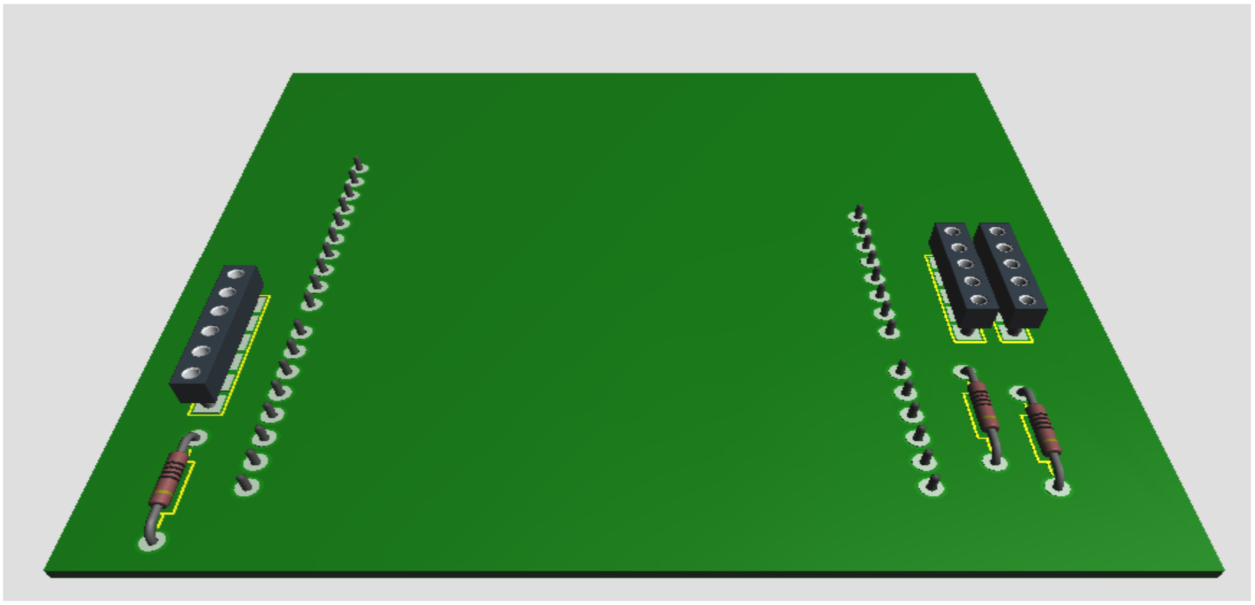

Figure 04: Visualization of PCB

Part: 1 Heart Rate Measurement from Photoplethysmographic (PPG) signal

Monitoring of health parameters during physical exercise is an important aspect of both sports and Photoplethysmography (PPG) is routinely employed for heart rate (HR) measurement in low-cost medical devices, since PPG signal periodicity corresponds to cardiac rhythm. PPG is an optical technique that detects the changes in blood volume occurring in tissues at the microvascular level. A pulse oximeter measures changes in the intensity of light reflected from a subject skin when it is illuminated by a light-emitting diode (LED). The resulting waveform presents a pseudoperiodic pulse component attributed to changes in blood volume occurring with each heartbeat. Further components, related to respiration, nervous activity, and thermoregulation, are superimposed on it.

Components:

1. Pulse Oximeter and Heart-Rate Sensor IC (MAX30100):

   The MAX30100 is an integrated pulse oximetry and heartrate monitor sensor solution. It combines two LEDs, a photodetector, optimized optics, and low-noise analog signal processing to detect pulse oximetry and heart-rate signals. The MAX30100 operates from 1.8V and 3.3V power supplies and can be powered down through software with negligible standby current, permitting the power supply to remain connected at all times.

A. Benefts and Features:

   - Complete Pulse Oximeter and Heart-Rate Sensor Solution Simplifies Design
   - Integrated LEDs, Photo Sensor, and
   - High-Performance Analog Front –End
   - Tiny 5.6mm x 2.8mm x 1.2mm 14-Pin Optically Enhanced System-in-Package
   - Ultra-Low-Power Operation Increases Battery Life for Wearable Devices
   - Programmable Sample Rate and LED Current for Power Savings
   - Ultra-Low Shutdown Current (0.7µA, typ)
   - Advanced Functionality Improves Measurement Performance
   - High SNR Provides Robust Motion Artifact Resilience
   - Integrated Ambient Light Cancellation

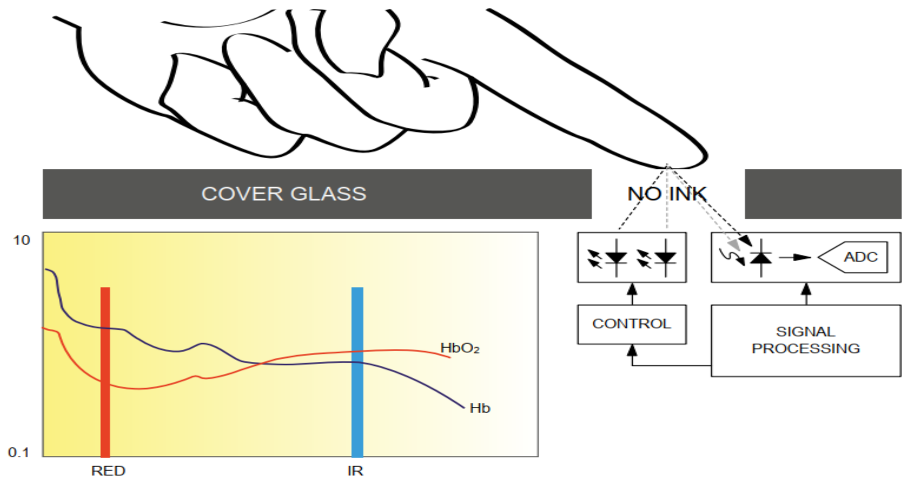

Figure 05: System Block Diagram.

B. SpO2 Subsystem

The SpO2 subsystem in the MAX30100 is composed of ambient light cancellation (ALC), 16-bit sigma delta ADC, and proprietary discrete time filter. The SpO2 ADC is a continuous time oversampling sigma delta converter with up to 16-bit resolution. The ADC output data rate can be programmed from 50Hz to 1kHz. The MAX30100 includes a proprietary discrete time filter to reject 50Hz/60Hz interference and low-frequency residual ambient noise.

C. Temperature Sensor

The MAX30100 has an on-chip temperature sensor for (optionally) calibrating the temperature dependence of the SpO2 subsystem. The SpO2 algorithm is relatively insensitive to the wavelength of the IR LED, but the red LED's wavelength is critical to correct interpretation of the data. The temperature sensor data can be used to compensate the SpO2 error with ambient temperature changes.

D. **LED Driver**

The MAX30100 integrates red and IR LED drivers to drive LED pulses for SpO2 and HR measurements. The LED current can be programmed from 0mA to 50mA (typical only) with proper supply voltage. The LED pulse width can be programmed from 200µs to 1.6ms to optimize measurement accuracy and power consumption based on use cases.

Table: Mode Control

MODE[2:0]	MODE
000	Unused
001	Reserved (Do not use)
010	HR only enabled
011	SPO_2 enabled
100–111	Unused

Table: Sample Rate Control

SPO2_SR[2:0]	SAMPLES (PER SECOND)
000	50
001	100
010	167
011	200
100	400
101	600
110	800
111	1000

Table: LED Pulse Width Control

LED_PW[1:0]	PULSE WIDTH (µs)	ADC RESOLUTION (BITS)
00	200	13
01	400	14
10	800	15
11	1600	16

Table: Heart-Rate Mode

SAMPLES (per second)	PULSE WIDTH (µs)			
	200	400	800	1600
50	O	O	O	O
100	O	O	O	O
167	O	O	O	▨
200	O	O	O	▨
400	O	O	▨	▨
600	O	O	▨	▨
800	O	O	▨	▨
1000	O	O	▨	▨
Resolution (bits)	13	14	15	16

Methodology

PPG Signal Acquisition:

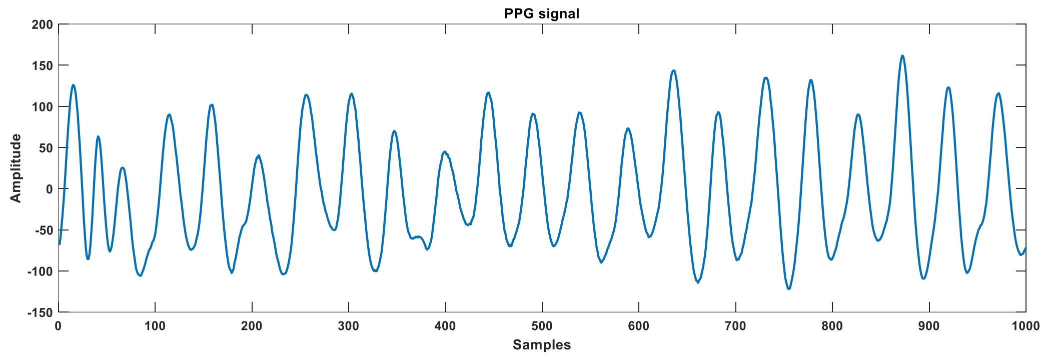

Figure 06: PPG signal recording.

PPG Denoising:

The trace library employed for trials provides a comprehensive set of test cases. Overall, each test lasts about 300 s, producing between 35 000 and 40 000 samples per trace. The database includes 12 traces from male subjects aged from 18 to 35, recorded while running, and 10 traces from further eight subjects, seven male and one female, with ages ranging from 19 to 58 years, recorded during rehabilitation exercises and boxing. All males are healthy, and the female subject is affected by a cardiovascular disease.

The test setup described in is composed of two pulse oximeters, using a green LED at 515 nm, and tri-axial accelerometer, carried with a wrist band. Ground-truth data are provided by an associated electrocardiographic (ECG) trace acquired by electrodes placed on a chest band. Each acquisition includes the two PPG records and the outputs of the three accelerometer channels. All signals are sampled at a frequency of Fs = 125 Hz. We analyze each signal employing an observation window of length TW = 8 s, resulting in a segment of N = 1000 samples. Consecutive windows have a 75% overlap (i.e., 6 s), so that the algorithm provides a new HR estimate every TR = 2 s, corresponding to a reporting rate of 0.5 Hz.

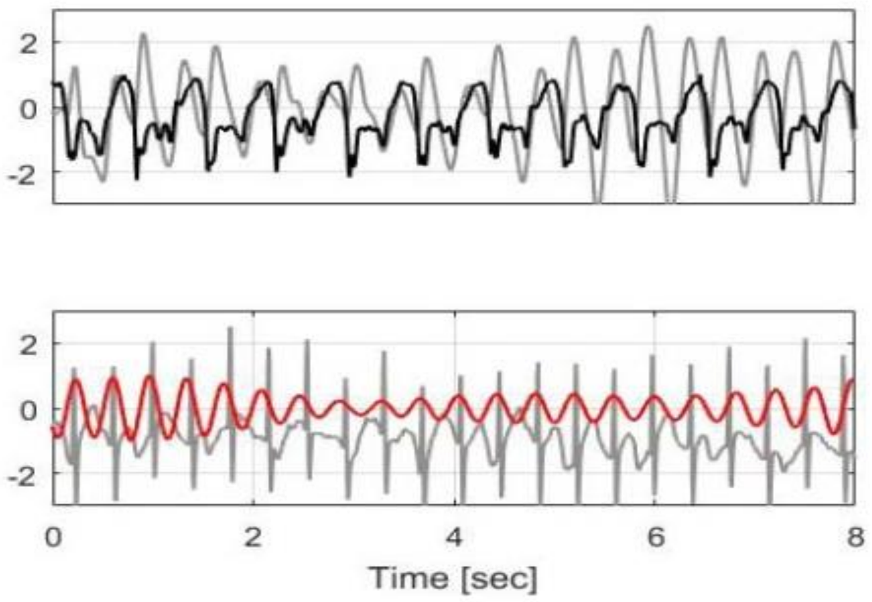

Figure 07: Comparison between the PPG signal and accelerometer. Both signals are periodic with almost the same pseudo period. ECG and reconstructed PPG signal.

Heart Rate Estimation and Tracking

A. *Frequency Analysis*

The initial HR measurement is obtained by frequency analysis of the denoised PPG signal *x(nTs)*. We aim at the ability of measuring to a resolution of one beat per minute (±1 bpm) that translates into the need to detect a minimum frequency shift $f < (1/60)$ Hz. This is 7.5 times better than the frequency grid step obtained by computing a DFT with the 8-s observation window. On the other hand, extending this length to the required 60 s would be impractical. For this reason, data are zero-padded to extend DFT vector length to 8192 samples. Zero padding allows to obtain a finer grid step $F \sim = 15$ mHz with a moderate increase in computation. Of course, the capability to resolve closely spaced frequency components remains lower bounded by the reciprocal of the sample sequence length TW. Since the spectrum of PPG signals is mostly made up of harmonic components, this should not cause problems, their separation being usually good enough to resolve them.

B. *Dynamic HR Model and Kalman Filtering*

HR measurements provided by DFT-based frequency analysis are still regarded as “raw measurements.” The algorithm stage that follows is based on a simple signal model of HR variation and deals with validation, filtering, and smoothing, ultimately providing a considerably more accurate HR measurement. Considerations based on physiology, combined with the need to employ simple equations in the interest of robustness, led us to the representation of the dynamic behavior of human HR by a discrete-time model based on the discrete-time random walk process.

Measurement noise and uncertainty of raw HR estimates are represented by the zero-mean white noise random process *v(k)*, whose finite variance is *σv*2. Careful KF tuning is critical to performance, and for this reason, the values of KF parameters *σw*2 and *σv*2 have been determined from the analysis of experimental data. The process noise variance *σw*2 accounts for the goodness of fit between actual HR evolution and the model introduced in (7). An evaluation of the contribution of *w(kTR)* can be obtained by considering HR reference values and extracting the statistical properties of the difference time series. This analysis was carried out on the set of 10 acquisitions that compose the SP Cup “testing” set, whose difference time series is characterized by greater variability. IKF equations for refined HR estimation enough to represent changes related to the intensity of physical exercise, but rules out unrealistically sudden variations.

Similarly, the variance of measurement noise process *v(kTR)* can be evaluated by considering the time series corresponding to the difference between the reference HR values and the estimated values provided at the output of the SVD algorithm. Performances of the algorithm are unaffected by slight variations of *σv*, as long as the order of magnitude is correct. Consequently, based on the histogram in Fig. 3, we calculated the standard deviation of values within ±50 bpm from zero, to discard possible outliers. This yields *σv* ∼ = 10 bpm. The whole set of KF equations are summarized in Table I for conciseness and, from here on, time interval *TR* is dropped from equations for simplicity. At the end of the *k*th algorithm iteration, the KF-smoothed HR measurement is, of course, the *a posteriori* state estimate *θ+(k)*.

RESULTS

We refer in this section to traces from, employed in as well as in a number of subsequent works for performance evaluation. Those traces cover a set of subjects of various ages, undertaking physical exercises of different intensity. HR varies in accordance with physical stress, though individual responses are closely related to specific physical conditions of the subject. Reference HR values, obtained from the simultaneously recorded ECG trace and indicated in the following as B*ref(k)*, were compared with the estimate B*meas(k) = θ+(k)*. It should be remembered that, for this kind of HR measurement, uncertainty is primarily a consequence of MAs. Thus, even though the set of conditions provided by does not cover all combinations of gender, age, and health conditions, the variety of MAs found in the traces is enough to support the claim that reported performance is representative.

1) The average *absolute* deviation from reference values:

$$E1 = \frac{1}{\mathrm{N}} \sum_{1}^{N} |Bmeans(k) - Bref(k)|$$

2) The average *relative* deviation from reference values

$$E2 = \frac{1}{N} \sum_{n=1}^{N} \left| \frac{Bmeans(k) - Bref(k)}{Bref(k)} \right|$$

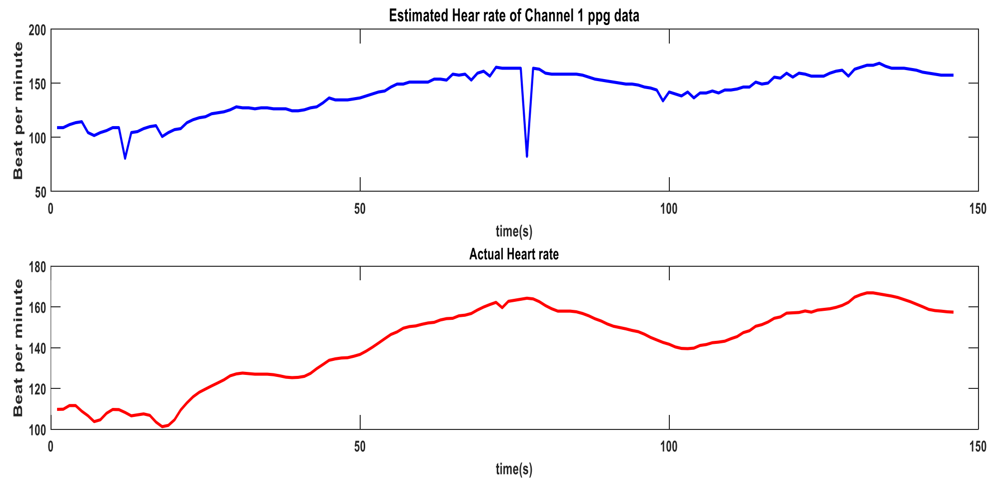

Figure 08: Comparison of Beat per minute of estimated (channel 1) and actual heartbeat.

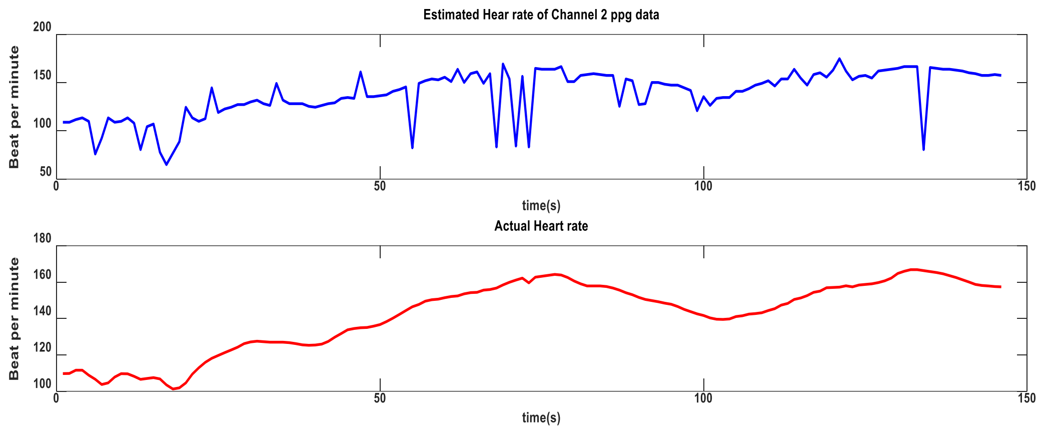

Figure 09: Comparison of Beat per minute of estimated (channel 2) and actual heartbeat.

Part: 2 Emotion detection from GSR(Galvanic Skin Response) signal

GSR:

The term Galvanic Skin Response (GSR) refers to changes in the electrical properties on the surface of the skin in response to sweat secretions. These secretions are due to an increment in sudomotor innervation, caused by the Sympathetic Nervous System (SNS) that results in changes in the GSR signals as the body responds to different daily circumstances: stress, temperature, anxiety, exertion situations, etc.

Hence, the sympathetic activity can be measured by analyzing the GSR signals. For instance, this relation between SNS acts and the observed skin conductance, which determines the sympathetic innervation of the sweat glands, has been extensively used in stress detection applications.

GSR signals carry information about SNS activity, but are also influenced by other factors, like temperature changes or sweating due to aerobic exercise.

Methodology:

The methodology for emotion detection using GSR sensor is given below:

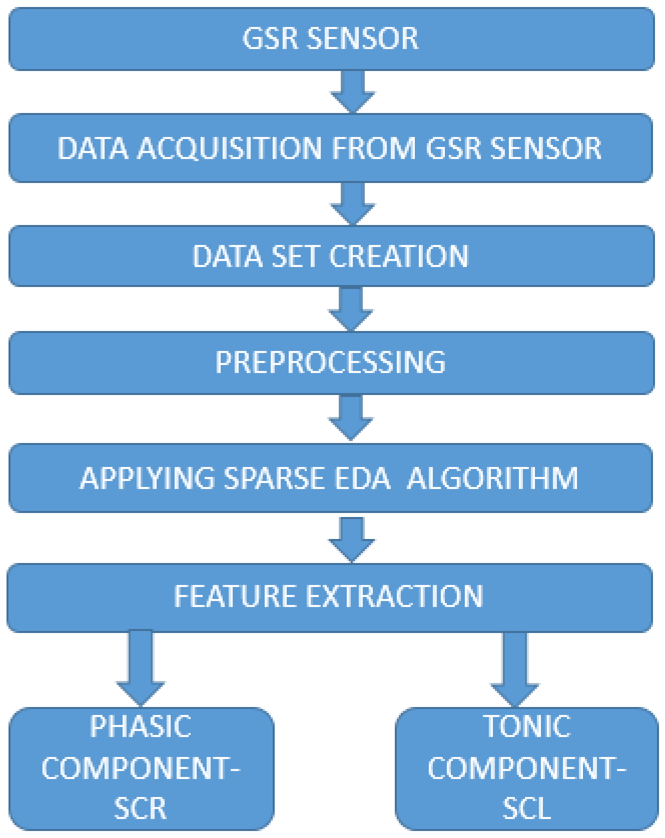

Figure 10: Flow chart for emotion detection

Data acquisition:

We have collected data from the GSR sensor. Resistance, conductance, and time these data were taken from the GSR sensor using Arduino IDE software.

Data set creation:

We have collected three sets of data for eight different people. All of them were shown three different types of videos and data was taken. Therefore, we have created 24 data set.

Preprocessing:

Continuous and unobtrusive measurement of GSR using wearable devices makes the signal collected vulnerable to several types of noise and artifacts. Artifacts can be generated from electronic noise or variation in the contact between the skin and the recording electrode caused by pressure, excessive movement or adjustment of the device. If these artifacts remain in the signal when it is analyzed, they can easily be misinterpreted and skew the analysis; for example, they may be easily mistaken for an SCR.

In order to remove these artifacts, machine-learning algorithm SVM (Support Vector Machine) has been used to detect automatically electro dermal activity (EDA) artifacts. In order to perform model selection we tested a range of settings for the parameters of SVM, including both a Radial Basis Function (RBF), polynomial, and linear kernel, and selected the settings that produced the highest accuracy on the validation set.

This method takes segments of 5 seconds and classifies them either as artifacts or as valid GSR signals. In those slots that are classified as artifacts, the signal is replaced by a polynomial regression of order 1 between the last non-artifact point in previous slots and the first non-artifact sample in subsequent slots, and no further processing is performed.

Besides the artifact removal stage, the complete signal is resampled to 8 Hz if the sampling frequency, fs, is higher than 8 Hz. This resampling does not have any influence on the signal's quality, since SCR waveforms can be perfectly represented using fs = 8 Hz (or even fs = 4 Hz), but allows us to reduce notably the computation time.

Continuous-mode Operation:

The final goal of the SparsEDA method is being able to extract EDA-related features continuously using shorter signal sets of $N = W + M + 1 < L$ samples, where M is the length of the waveforms used to construct the SCR dictionary and L is the total number of samples available of the GSR signal. This continuous-mode operation allows us to deal with large signals of arbitrary length (since the approach is applied on segments of fixed length) and even to process signals on-line (i.e., as they are being acquired) in real-time or quasi-real-time.

Continuous-mode operation SparsEDA algorithm that we have used finally is:

1. Extend the signal by adding M samples before the first one and W samples after the last one.
2. Set the first N samples of the extended signal as the active set.
3. Subtract the value of the first sample, and apply the SparsEDA method on the current active set, thus obtaining an estimate of the SCL and SCR components. Add back the value of the subtracted sample to obtain the final solution.
4. Keep shifting the active set by W=3 samples and repeating the previous stage until all the samples in the original GSR signal have been processed.
5. Discard the SCL and SCR components corresponding to the samples added at the beginning and the end of the signal.

Feature extraction:

The SparsEDA algorithm allows us to extract two GSR components (features) which are interesting for SNS studies:

- Tonic component: It is also called Skin Conductance Level (SCL), which is a slow changing signal. The SCL is related to several non-SNS activity factors but also to the level of attention of the subject, even in the absence of instantaneous stimuli.
- Phasic component: It is also called Skin Conductance Response (SCR), which is the reaction to sporadic SNS stimuli. The SCR, which is superimposed on top of the tonic component, includes higher frequency components and appears only within specific time windows whose length typically lasts from one to five seconds.

Result:

After applying the sparse EDA algorithm to our data sets the following results were observed:

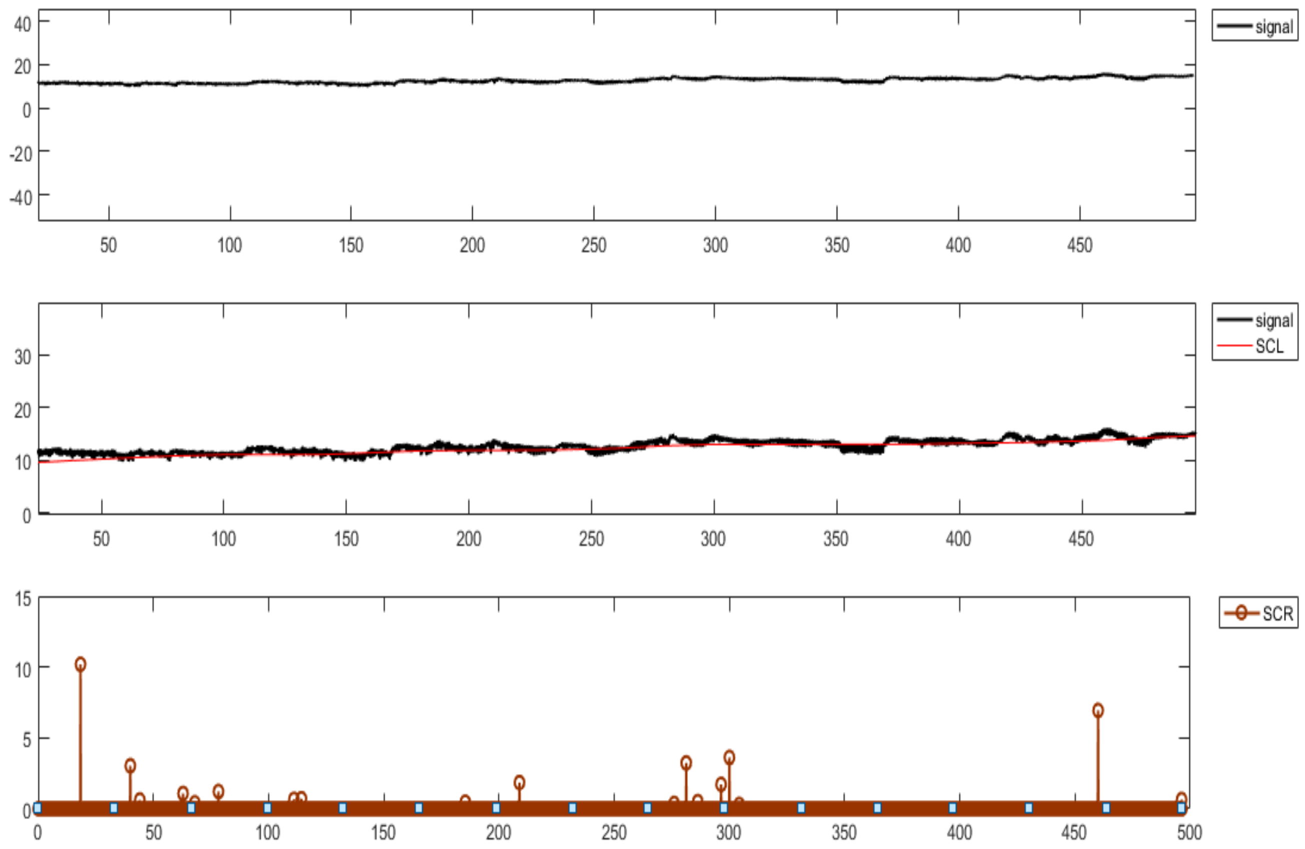

Figure 12: (top) GSR signal, (middle) GSR signal with SCL components, (bottom) SCR components of the SCR signal.

Cost:

Component's name	*Price (Taka)*
MAX30100 PPG sensor	750
Grove GSR sensor	1000
HC-05 Bluetooth module	300
Arduino Uno	350
PCB	300
Jumper wire	80

Conclusion:

We have made a wearable device, which enable us to collect GSR signal and PPG signal simultaneously. We have made a data set for eight different people. After removing noise by filtering and applying artifacts removal algorithm we have measured heart rate (HR) from the PPG signal and detected SCR and SCL components of the GSR signal which help us to determine level of attraction and sudden change in emotion.

References:

1. http://eda-explorer.media.mit.edu/
2. Feature Extraction of Galvanic Skin Responses by Non-Negative Sparse Deconvolution - Francisco Hernando-Gallego, David Luengo, Member, IEEE, and Antonio Artes-Rodriguez,SeniorMember, IEEE
3. Measuring Heart Rate During Physical Exercise bySubspace Decomposition and Kalman Smoothing -Alessandra Galli, Claudio Narduzzi , and Giada Giorgi
4. PREHEAT: Precision heart rate monitoring from intense motionartifact corrupted PPG signals using constrained RLS and wavelets Md. Shofiqul Islam, Md. Shifat-E-Rabbi, Abdullah Mohamed Ali Dobaie, Md. Kamrul Hasan.

Appendix

PPG code:

```
clc
close all
clear all

filename = '.\DATABASE\Training_data\DATA_05_TYPE02.MAT';
matObj = matfile(filename);
Tracce=matObj.sig;

fn='.\DATABASE\Training_data\DATA_05_TYPE02_BPMtrace.MAT';
matObj=matfile(fn);
BPM0=matObj.BPM0;

if(min(size(Tracce))==6)    %training trace
    ECG=Tracce(1,:);
    PPG1=Tracce(2,:);
    PPG2=Tracce(3,:);
    AX=Tracce(4,:);
    AY=Tracce(5,:);
    AZ=Tracce(6,:);
else                           % competition trace
    PPG1=Tracce(1,:);
    PPG2=Tracce(2,:);
    AX=Tracce(3,:);
    AY=Tracce(4,:);
    AZ=Tracce(5,:);
end

N=length(PPG1);
K=1000;
DK=250;
i=1; fin=0;
n=length(Tracce);
while fin < n - DK

    in=(i-1)*DK+1;
    fin=in+K-1;
    ppg1=PPG1(in:fin);
    ppg2=PPG2(in:fin);
    accx=AX(in:fin);
    accy=AY(in:fin);
    accz=AZ(in:fin);
    [temp1,temp2]=signal_denoising(ppg1,ppg2,accx,accy,accz);
    [z_ppg1(i) z_ppg2(i)] = frequency_analysis(ppg1,ppg2,temp1,temp2);
    i=i+1;
end

%% Accuracy Measurement

% channel 1
temp = abs(BPM0-z_ppg1')./BPM0;
tolerance1 = sum(temp)/length(temp)*100
accuracy1 = 100 - tolerance1
```

```
% channel 2
temp = abs(BPM0-z_ppg2')./BPM0;
tolerance2 = sum(temp)/length(temp)*100
accuracy2 = 100 - tolerance2

%% BPM plot
subplot(211),plot(z_ppg1)
xlabel('time(s)')
ylabel('Beat per minute')
title('Estimated Hear rate of Channel 1 ppg data')
subplot(212),plot(BPM0)
xlabel('time(s)')
ylabel('Beat per minute')
title('Actual Heart rate')

figure
subplot(211),plot(z_ppg2)
xlabel('time(s)')
ylabel('Beat per minute')
title('Estimated Hear rate of Channel 2 ppg data')
subplot(212),plot(BPM0)
xlabel('time(s)')
ylabel('Beat per minute')
title('Actual Heart rate')


clc
clear
close all

TOT_EST=[];
TOT_TRUE=[];
PPG1=[];
PPG2=[];

%% Result calculation

for k=1:12

    eval(['load(''./DATI_INTERMEDI/TR',num2str(k),''');'])
    [res]=kalman_filter(z_ppg1,z_ppg2);


    errore.v1(k)=mean(abs(BPM0(1:end)-res(1:end)'));
    errore.v2(k)=mean(abs(BPM0(1:end)-res(1:end)')./(BPM0(1:end)))*100;
    errore.v3(k)=max(abs(BPM0(1:end)-res(1:end)'));
    errore.v4(k)=std(abs(BPM0(1:end)-res(1:end)'));

    errore.r1_2s(k)=mean(abs(BPM0(1:end-1)-res(2:end)'));
    errore.r2_2s(k)=mean(abs(BPM0(1:end-1)-res(2:end)')./(BPM0(1:end-
1)))*100;
    errore.r3_2s(k)=max(abs(BPM0(1:end-1)-res(2:end)'));
    errore.r4_2s(k)=std(abs(BPM0(1:end-1)-res(2:end)'));

    errore.r1_4s(k)=mean(abs(BPM0(1:end-2)-res(3:end)'));
    errore.r2_4s(k)=mean(abs(BPM0(1:end-2)-res(3:end)')./(BPM0(1:end-
2)))*100;
    errore.r3_4s(k)=max(abs(BPM0(1:end-2)-res(3:end)'));
    errore.r4_4s(k)=std(abs(BPM0(1:end-2)-res(3:end)'));

    TOT_EST=[TOT_EST res(3:end)];
    TOT_TRUE=[TOT_TRUE, BPM0(1:end-2)'];

    PPG1=[PPG1 z_ppg1(3:end)];
    PPG2=[PPG2 z_ppg2(3:end)];


end

 function [ xr_1,xr_2 ] = signal_denoising(ppg1,ppg2,accx,accy,accz)
 %% Construction of Hankel matrices and SVD
    H1=Hk(ppg1);
    [U1,S1,V1]=svd(H1);
    eig1=diag(S1);

    H2=Hk(ppg2);
    [U2,S2,V2]=svd(H2);
    eig2=diag(S2);

    H_x=Hk(accx);
    [U_x,S_x,~]=svd(H_x);
    eig_x=diag(S_x);

    H_y=Hk(accy);
    [U_y,S_y,~]=svd(H_y);
    eig_y=diag(S_y);

    H_z=Hk(accz);
    [U_z,S_z,~]=svd(H_z);
    eig_z=diag(S_z);


    ppg_aut_1=auto_grandi(eig1);
    ppg_aut_2=auto_grandi(eig2);
    acc_aut_x=auto_grandi(eig_x);
    acc_aut_y=auto_grandi(eig_y);
    acc_aut_z=auto_grandi(eig_z);

    %% Correlation matrices
    Matrix_Cor_1x=(U1')*(U_x);    % ppg1 e accx
    Matrix_Cor_1y=(U1')*(U_y);    % ppg1 e accy
    Matrix_Cor_1z=(U1')*(U_z);    % ppg1 e accx
    Matrix_Cor_2x=(U2')*(U_x);    % ppg2 e accx
    Matrix_Cor_2y=(U2')*(U_y);    % ppg2 e accy
    Matrix_Cor_2z=(U2')*(U_z);    % ppg2 e accz


    Matrix_Cor_1x=Matrix_Cor_1x(1:ppg_aut_1,1:acc_aut_x);
    Matrix_Cor_1y=Matrix_Cor_1y(1:ppg_aut_1,1:acc_aut_y);
    Matrix_Cor_1z=Matrix_Cor_1z(1:ppg_aut_1,1:acc_aut_z);
    Matrix_Cor_2x=Matrix_Cor_2x(1:ppg_aut_2,1:acc_aut_x);
    Matrix_Cor_2y=Matrix_Cor_2y(1:ppg_aut_2,1:acc_aut_y);
```

```
Matrix_Cor_2z=Matrix_Cor_2z(1:ppg_aut_2,1:acc_aut_z);

Matrix_Cor_1x=(Matrix_Cor_1x.^2)';
Matrix_Cor_1y=(Matrix_Cor_1y.^2)';
Matrix_Cor_1z=(Matrix_Cor_1z.^2)';
Matrix_Cor_2x=(Matrix_Cor_2x.^2)';
Matrix_Cor_2y=(Matrix_Cor_2y.^2)';
Matrix_Cor_2z=(Matrix_Cor_2z.^2)';

SUM_1x=max(Matrix_Cor_1x);
SUM_1y=max(Matrix_Cor_1y);
SUM_1z=max(Matrix_Cor_1z);
SUM_2x=max(Matrix_Cor_2x);
SUM_2y=max(Matrix_Cor_2y);
SUM_2z=max(Matrix_Cor_2z);

SUM_1=max([SUM_1x; SUM_1y; SUM_1z]);
SUM_2=max([SUM_2x; SUM_2y; SUM_2z]);

%% Reconstruction

SOGLIA=0.6;
Sr_1=zeros(size(S1)); % matrix 701x300
Sr_2=zeros(size(S2)); % matrix 701x300

% ------------------ PPG1 --------------------
cont=0;
for i=1:length(SUM_1)
    if SUM_1(i)<SOGLIA
        Sr_1(i,i)=eig1(i);
        cont=cont+1;
    end
end

if cont==0
    for i=1:length(SUM_1)
       Sr_1(i,i)=eig1(i);
    end
end

mH_1=U1*Sr_1*V1';

for i=1:ppg_aut_1
       temp(i,:)=mH_1(:,i)/sqrt(eig1(i));
end

if size(temp,1)>1
    xr_1=sum(temp);
else
    xr_1=temp;
end

% ------------------ PPG2 --------------------
cont=0;
for j=1:length(SUM_2)
    if SUM_2(j)<SOGLIA
        Sr_2(j,j)=eig2(j);
```

```
                cont=cont+1;
            end
        end
        if cont==0
            for j=1:length(SUM_2)
               Sr_2(j,j)=eig2(j);
            end
        end

        mH_2=U2*Sr_2*V2';

        for j=1:ppg_aut_2
               temp2(j,:)=mH_2(:,j)/sqrt(eig2(j));
        end
        if size(temp2,1)>1
            xr_2=sum(temp2);
        else
            xr_2=temp2;
        end
end

%PPG function code

function[res]=kalman_filter(z_ppg1,z_ppg2)

    Q=4^2;
    R=10^2;

    count=0;
    countmax=5;
    imax=3;

    xposteriori=(z_ppg1(1)+z_ppg1(1))/2;
    res(1)=(z_ppg1(1)+z_ppg1(1))/2;
    Pposteriori=0;

    for i=2:length(z_ppg1)

        xpriori=xposteriori;
        Ppriori=Pposteriori+Q;

        S=Ppriori+R;
        gain=Ppriori/S;
        %--------------------------------------------------------------------
-
        inn1=z_ppg1(i)-xpriori;
        inn2=z_ppg2(i)-xpriori;

        VAR1=var(z_ppg1(max(1,(i-90)):i));
        VAR2=var(z_ppg2(max(1,(i-90)):i));

        %check
        if(isnan(inn1) || abs(inn1)>2*sqrt(S))
            flag1=1;
        else
            flag1=0;
        end
        %check
        if(isnan(inn2) || abs(inn2)>2*sqrt(S))
            flag2=1;
        else
            flag2=0;
        end

        if (VAR1*2)<VAR2  && i>20
          flag2=1;
        elseif   (VAR2*2)<VAR1  && i>20
           flag1=1;
        end

        if((flag1==0 && flag2==0)||((count==countmax || i<imax) &&
(~isnan(inn1) && ~isnan(inn2))))

            if(abs(inn1)<abs(inn2))
                inn=inn1;
            else
                inn=inn2;
            end
            xposteriori=xpriori+gain*inn;
            Pposteriori=(1-gain)*Ppriori;
            if(count==countmax )
                xposteriori(1)=(z_ppg1(i)+z_ppg2(i)+xpriori(1))/3;
            end
            count=0;
        elseif((flag1==0 && flag2==1)||((count==countmax || i<imax) &&
~isnan(inn1)))

            inn=inn1;
            xposteriori=xpriori+gain*inn;
            Pposteriori=(1-gain)*Ppriori;
            if(count==countmax )
                xposteriori(1)=0.5*(z_ppg1(i)+xpriori(1));
            end
            count=0;
       elseif((flag2==0 && flag1==1) ||((count==countmax || i<imax) &&
~isnan(inn2)))

            inn=inn2;
            xposteriori=xpriori+gain*inn;
            Pposteriori=(1-gain)*Ppriori;
            if(count==countmax )
                xposteriori(1)=0.5*(z_ppg2(i)+xpriori(1));
            end
            count=0;
        else
            count=count+1;
            xposteriori=xpriori;
            Pposteriori=Ppriori;
        end

      res(i)=xposteriori(1);
    end
end
```

```
function[p1,p2]=frequency_analysis(ppg1,ppg2,xr_1,xr_2)

    fS=125;
    Nfft=2^13;
    F=fS/Nfft;
    w=hann(length(xr_1))';
    w2=hann(length(xr_2))';

    Xr_1=abs(fft(xr_1.*w,Nfft)/sum(w));
    Xr_2=abs(fft(xr_2.*w2,Nfft)/sum(w2));

    start=round(60/60/F);
    fine=round(180/60/F);
    [~, ind_pk_1]=findpeaks(Xr_1(start:fine),'sortstr','descend');
    [~, ind_pk_2]=findpeaks(Xr_2(start:fine),'sortstr','descend');

    w=hann(length(ppg1))';
    w2=hann(length(ppg2))';
    X1=abs(fft(ppg1.*w,Nfft)/sum(w));
    X2=abs(fft(ppg2.*w2,Nfft)/sum(w2));
    [~, ind1]=findpeaks(X1(start:fine),'sortstr','descend');
    [~, ind2]=findpeaks(X2(start:fine),'sortstr','descend');


    if size(ind_pk_1)>0
        p1=(ind_pk_1(1)+start-2)*F*60;
        picco1=(ind1(1)+start-2)*F*60;
        if abs(2*picco1-p1)<10
            p1=picco1;
        end
    else
        p1=NaN;
    end

    if size(ind_pk_2)>0
        p2=(ind_pk_2(1)+start-2)*F*60;
        picco2=(ind2(1)+start-2)*F*60;
         if abs(2*picco2-p2)<10
             p2=picco2;
         end
    else
        p2=NaN;
    end
end



function [H]=Hk(x)
L=400;
N=1000;
K=N-L+1;
xc=x(1:K);
xr=x(K:N);
H=hankel(xc,xr);
```

```
end



function [num_aut]=auto_grandi(eigen)

eig1=eigen(1:10);
dif=-diff(eig1);
media=mean(dif);
index=find(dif>media);
ind=index(end);
p=dif(ind);
if p>2*(eigen(1)-eigen(2))
    num_aut=ind;
else
    num_aut=10;
end
num_aut=max(4,num_aut);
end
```

GSR code:

```
%% clears
clc
clear all
close all


%% DATA 1
% ivn07_16_matlab.mat should be in the current folder to load the gsr
% signal

%load('data1.mat')
signal_pre = csvread('our_data_1.csv')';
signal = signal_pre(1,:);
%signal= data.conductance';
sr = 16;
graphics = 1;
epsilon  = 1;
Kmax    = 40;
dmin = 1.25*sr;  % Minimum distance between activations
rho = 0.025;

%%
[driver, SCL, MSE] = sparsEDA(signal,sr,graphics,epsilon,Kmax,dmin,rho);
% figure(2)
% subplot(3,1,2)
% plot(MSE);
% subplot(3,1,1)
 %plot(SCL);
% subplot(3,1,3)
% plot(driver);
```

%% GSR function code Sparse EDA algorithm

```
function [driver, SCL, MSE] =
sparsEDA(signalIn,sr,graphics,epsilon,Kmax,dmin,rho)
    % Authors, F. Hernando-Gallego, D. Luengo-Garcia and A. Artes-Rodriguez
    % Dpt. of Signal Theory and Communications
    % Universidad Carlos III de Madrid
    % -----------------------------------------------------------------------
    % Abstract. This method provide the low component and the driver
    % response of a galvanic skin response signal longer than 70 seconds.
    % -----------------------------------------------------------------------
    % signal  [vector]: galvanic skin response
    % sr         [int]: sample rate
    % graphics   [0/1]: show graphics (1) or not (0)
    % epsilon [double]: step remainder
    % maxIters   [int]: maximum number of LARS iterations
    % dmin    [double]: maximum distance between sparse reactions
    % th      [double]: minimun threshold of sparse reactions
    %
    % OUTPUTS
    % driver  [vector]: driver responses, tonico componet
    % SCL     [vector]: low component
    % MSE     [vector]: reminder of the signal fitting


    %% 0. Exceptions
    if (length(signalIn)/sr < 80)
        display('Signal not enought large. May be longer than 80 senconds')
    end

    if (sum(isnan(signalIn))>0)
        display('Signal contains NaN')
    end
    %% 1.PREPROCESSING


    % addition start and end to process the signal completly
    signalAdd = ...
        [signalIn(1)*ones(20*sr,1); signalIn; signalIn(end)*ones(60*sr,1)];

    if (sr>8)
        [P,Q] = rat(8/sr);
        signalAdd = resample(signalAdd,P,Q);
        Nss = floor(8*length(signalIn)/sr);
        sr = 8;
    else
        Nss = length(signalIn);
    end

    Ns  = length(signalAdd);
    b0 = 0;

    pointerS = (20*sr) + 1;
    pointerE = pointerS + Nss;
    signalRs = signalAdd(pointerS:pointerE);

    %% Overlap save
    % Loop Analysis
    durationR = 70;  % Slots 70 seconds (60s + 10s for deconvolution)
    Lreg = 20*sr*3;  % 60 seconds
    L = 10*sr;       % pulse
    N = durationR*sr;% 70 seconds
    T = 6;           % CSL additional columns

    % Toeplitz Matrix
    % SCR
    Rzeros = zeros(N+L,Lreg);
    srF = sr*[0.5 0.75 1 1.25 1.5]; % Multianalysis
    for j=1:length(srF)
        t_rf = 0:(1/srF(j)):10; % 10 seconds
        taus = [0.5, 2, 1];
        rf_biexp = exp(-t_rf/taus(2)) - exp(-t_rf/taus(1));
        rf_est = taus(3)*rf_biexp;
        rf_est = rf_est/sqrt((sum(rf_est.^2)));
        Rzeros(1:N,1 + (j-1)*Lreg:j*Lreg) =...
            toeplitz([rf_est zeros(1,N-length(rf_est))],zeros(1,Lreg));
    end
    R = Rzeros(1:N,1:5*Lreg);
    % SCL
    R = [zeros(N,T) R];
    R(1:Lreg,1) = (+1)*linspace(0,1,Lreg)';
    R(1:Lreg,2) = (-1)*linspace(0,1,Lreg)';
    R((Lreg/3)+1:Lreg,3) = (+1)*linspace(0,2/3,(2*Lreg)/3)';
    R((Lreg/3)+1:Lreg,4) = (-1)*linspace(0,2/3,(2*Lreg)/3)';
    R((2*Lreg/3)+1:Lreg,5) = (+1)*linspace(0,1/3,Lreg/3)';
    R((2*Lreg/3)+1:Lreg,6) = (-1)*linspace(0,1/3,Lreg/3)';
    Cte = sum(R(:,1).^2);
    R(:,1:6) = R(:,1:6)/sqrt(Cte);

    %% Loop
    cutS = 1;
    cutE = N;
    slcAux   = zeros(1,Ns);
    drverAux = zeros(1,Ns);
    resAux   = zeros(1,Ns);
    aux      = 0;

    while (cutE<Ns)
        aux = aux + 1;
        signalCut = signalAdd(cutS:cutE);

        if(b0==0)
            b0 = signalCut(1);
        end

        signalCutIn = signalCut - b0;
        [beta,~, activationHist,~,~,~] =
lasso(R,signalCutIn,sr,Kmax,epsilon);
        % display(activationHist)
        signalEst = R*beta + b0;

        remAout = (signalCut - signalEst).^2;
        res2 = sum(remAout(20*sr+1:(40*sr)));
        res3 = sum(remAout(40*sr+1:(60*sr)));
        jump = 1;
        if (res2 < 1)
            jump = 2;
            if (res3 < 1)
                jump = 3;
            end
        end

        SCL = R(:,1:6)*beta(1:6) + b0;
        SCRline = beta(7:end);
        SCRaux = zeros(Lreg,5);
        SCRaux(:) = SCRline;
        driver = sum(SCRaux');

        b0 = R(jump*20*sr,1:6)*beta(1:6) + b0;

        drverAux(cutS:cutS+(jump*20*sr)-1) = driver(1:jump*sr*20)';
        slcAux(cutS:cutS+(jump*20*sr)-1) = SCL(1:jump*sr*20)';
        resAux(cutS:cutS+(jump*20*sr)-1) = remAout(1:jump*sr*20)';
        cutS = cutS + (jump)*20*sr;
        cutE = cutS + N -1;
    end

    SCRaux    = drverAux(pointerS:pointerE);
    SCL       = slcAux(pointerS:pointerE);
    MSE       = resAux(pointerS:pointerE);

    %% POST-PROCESSING: REMOVING UNNECESSARY SCR COMPONENTS


    ind = find(SCRaux>0);
    scr_temp = SCRaux(ind);
    [scr_ord, ind2] = sort(scr_temp, 'descend');
    scr_fin = scr_ord(1);
    ind_fin = ind(ind2(1));
    for i = 2:length(ind2)
        if all(abs(ind(ind2(i))-ind_fin)>=dmin)
            scr_fin = [scr_fin scr_ord(i)];
            ind_fin = [ind_fin ind(ind2(i))];
        end
    end
    driver = zeros(size(SCRaux));
    driver(ind_fin) = scr_fin;

    scr_max = scr_fin(1);
    threshold = rho*scr_max;
    driver(driver < threshold) = 0;

        %%
    if (graphics)
        f1 = figure('Position',[-1900 8 1890 990]);
        t = 0:(1/sr):(length(signalRs)/sr)-(1/sr);
        subplot(311)
        plot(t,signalRs,'k','Linewidth',1.5)
        legend('signal','Location','NorthEastOutside')
        subplot(312)
        plot(t,signalRs,'k','Linewidth',1.5)
        hold on
        plot(t,SCL)
        legend('signal','SCL','Location','NorthEastOutside')
        subplot(313)
        plot(t,driver,'-o','Linewidth',1.5)
        legend('SCR','Location','NorthEastOutside')
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [beta, numIters, activationHist, duals, lambda, res] = lasso(R, s,
sr,maxIters,epsilon)

    N = length(s);
    W = size(R,2);

    OptTol = -10;
    solFreq = 0;
    resStop2 = .0005;
    lambdaStop = 0;

    % Global variables for linsolve function
    global opts opts_tr zeroTol
    opts.UT = true;
    opts_tr.UT = true; opts_tr.TRANSA = true;
    zeroTol = 1e-5;

    x = zeros(W,1);
    x_old = zeros(W,1);
    iter = 0;

    % Correlation
    c = R'*s;

    % Maximun correlation value
    lambda = max(c);

    if (lambda < 0)
        error('y is not expressible as a non-negative linear combination of
the columns of X');
    end

    % argmax de correlacion
    newIndices = find(abs(c-lambda) < zeroTol)';

    collinearIndices = [];
    beta = [];
    duals = [];
    res = s; % res en principio la seÃ±al

    % Check stopping conditions
    if ((lambdaStop > 0) & (lambda < lambdaStop)) | ((epsilon > 0) &
(norm(res) < epsilon))
        activationHist = [];
        numIters = 0;
        % return;
    end
    % Initialize Cholesky factor of A_I
    R_I = [];
    activeSet = [];
    for j = 1:length(newIndices)
        iter = iter+1;
        [R_I, flag] = updateChol(R_I, N, W, R, 1, activeSet, newIndices(j));
        activeSet = [activeSet newIndices(j)];
    end
    activationHist = activeSet;


    %%%%%% LOOP %%%%%%
    done = 0;
    while  ~done

        if (length(activationHist) == 4)
            lambda = max(c);
            newIndices = find(abs(c-lambda) < zeroTol)';
            % R_I = [];
            activeSet = [];
            for j = 1:length(newIndices)
                iter = iter+1;
                [R_I, flag] = updateChol(R_I, N, W, R, 1, activeSet,
newIndices(j));
                activeSet = [activeSet newIndices(j)];
            end
            activationHist = [activationHist activeSet];
        else
            lambda = c(activeSet(1));
        end

        % Compute Lars direction - Equiangular vector
        dx = zeros(W,1);
        % Solve the equation (A_I'*A_I)dx_I = sgn(corr_I)
        z = linsolve(R_I,sign(c(activeSet)),opts_tr);
        dx(activeSet) = linsolve(R_I,z,opts);

        v = R(:,activeSet)*dx(activeSet);
        ATv = R'*v;

        % For  Lasso, Find first active vector to violate sign constraint
        gammaI = Inf;
        removeIndices = [];

        % Find first inacti ve vector to enter the active set
        inactiveSet = 1:W;
        inactiveSet(activeSet) = 0;
        inactiveSet(collinearIndices) = 0;
        inactiveSet = find(inactiveSet > 0);

        if (length(inactiveSet) == 0)
            gammaIc = 1;
            newIndices = [];
        else
            epsilon = 1e-12;
            gammaArr = (lambda - c(inactiveSet))./(1 - ATv(inactiveSet) +
epsilon);
```

```
            gammaArr(gammaArr < zeroTol) = Inf;

            [gammaIc, Imin] = min(gammaArr);
            newIndices = inactiveSet(find(abs(gammaArr - gammaIc) <
zeroTol));
         end


        gammaMin = min(gammaIc,gammaI);

        % Compute the next Lars step
        x = x + gammaMin*dx;
        res = res - gammaMin*v;
        c = c - gammaMin*ATv;

        % Check stopping condition
        if ((lambda - gammaMin) < OptTol) | ((lambdaStop > 0) & (lambda <=
lambdaStop)) | ((epsilon > 0) & (norm(res) <= epsilon))
            newIndices = [];
            removeIndices = [];
            done = 1;

            if ((lambda - gammaMin) < OptTol)
                display(num2str((lambda - gammaMin)))
            end
            if ((lambdaStop > 0) & (lambda <= lambdaStop))
            end
            if ((epsilon > 0) & (norm(res) <= epsilon))
            end

        end

        if (norm(res(1:sr*20)) <= resStop2)
            done = 1;
            if (norm(res(sr*20:sr*40)) <= resStop2)
                done = 1;
                if (norm(res(sr*40:sr*60)) <= resStop2)
                    done = 1;
                end
            end
        end

        % Add new indices to active set
        if (gammaIc <= gammaI) && (length(newIndices) > 0)
            for j = 1:length(newIndices)
                iter = iter+1;
                % Update the Cholesky factorization of A_I
                [R_I, flag] = updateChol(R_I, N, W, R, 1, activeSet,
newIndices(j));
                % Check for collinearity
                if (flag)
                    collinearIndices = [collinearIndices newIndices(j)];
                else
                    activeSet = [activeSet newIndices(j)];
                    activationHist = [activationHist newIndices(j)];
                end
            end
        end

        % Remove violating indices from active set
        if (gammaI <= gammaIc)
            for j = 1:length(removeIndices)
                iter = iter+1;
                col = find(activeSet == removeIndices(j));

                % Downdate the Cholesky factorization of A_I
                R_I = downdateChol(R_I,col);
                activeSet = [activeSet(1:col-1),
activeSet(col+1:length(activeSet))];

                % Reset collinear set
                collinearIndices = [];
            end

            x(removeIndices) = 0;  % To avoid numerical errors
            activationHist = [activationHist -removeIndices];
        end

        if iter >= maxIters
            done = 1;
        end

        if (find(x<0)>0)
            x = x_old;
            done = 1;
        else
            x_old = x;
        end

        if done | ((solFreq > 0) & (~mod(iter,solFreq)))
            beta = [beta x];
            duals = [duals v];
        end
end

numIters = iter;
clear opts opts_tr zeroTol
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [R_I, flag] = updateChol(R_I, n, N, R, explicitA, activeSet,
newIndex)
% updateChol: Updates the Cholesky factor R of the matrix
% X(:,activeSet)'*X(:,activeSet) by adding X(:,newIndex)
% If the candidate column is in the span of the existing
% active set, R is not updated, and flag is set to 1.

global opts_tr zeroTol
flag = 0;


newVec = R(:,newIndex);

if (length(activeSet) == 0)
    R_I = sqrt(sum(newVec.^2));
else
```

```
    if (explicitA)
        p = linsolve(R_I,R(:,activeSet)'*R(:,newIndex),opts_tr);
    else
        AnewVec = feval(R,2,n,length(activeSet),newVec,activeSet,N);
        p = linsolve(R_I,AnewVec,opts_tr);
    end
    q = sum(newVec.^2) - sum(p.^2);
    if (q <= zeroTol) % Collinear vector
        flag = 1;
    else
        R_I = [R_I p; zeros(1, size(R_I,2)) sqrt(q)];
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function R = downdateChol(R, j)
% downdateChol: `Downdates' the cholesky factor R by removing the
% column indexed by j.

% Remove the j-th column
R(:,j) = [];
[m,n] = size(R);

% R now has nonzeros below the diagonal in columns j through n.
% We use plane rotations to zero the 'violating nonzeros'.
for k = j:n
    p = k:k+1;
    [G,R(p,k)] = planerot(R(p,k));
    if k < n
        R(p,k+1:n) = G*R(p,k+1:n);
    end
end

% Remove last row of zeros from R
R = R(1:n,:);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Arduino code for converting GSR signal to resistance value

```
#include <Arduino.h>

#include <math.h>

const int GSR=A0;

int sensorValue=0;

float gsr_average=0;

float resistance=0;

float conductance=0;


void setup(){
  Serial.begin(9600);
  int startMillis = millis();
}

void loop(){
  long sum=0;
  for(int i=0;i<10;i++)           //Average the 10 measurements to remove the glitch
      {
      sensorValue=analogRead(GSR);
      sum += sensorValue;
      delay(5);
      }
   gsr_average = sum/10;
   resistance=((1024+2*gsr_average)*10000)/(abs(512-gsr_average));
   conductance=(1/resistance)*1000000;//micro Siemens;

   Serial.print(gsr_average);
   Serial.print(",");
   Serial.print(resistance);
   Serial.print(",");
   Serial.print(conductance);
   Serial.print(",");
   Serial.println(millis());
}
```

//Human Resistance = ((1024+2*Serial_Port_Reading)*10000)/(512-Serial_Port_Reading), unit is ohm, Serial_Port_Reading is the value display on Serial Port(between 0~1023)