Matlab®Code for ECG peak detection

```
clear all
close all
clc

%% Connect to Arduino
delete(instrfindall)
s = serial('COM1');
set(s,'BaudRate',57600);
try
    fopen(s);
catch err
    fclose(instrfind);
    error('Make sure you select the correct COM Port where the Arduino is connected.');
end

%% input parametres
Fs=200;                                     % sampling frequency
N=3;                                        % order of bandpass filter
band=[5 15];                                % bandpass frequency

%% calculate filter coefficients
[B,A] = butter(N,band/(Fs/2),'bandpass');
bbuffer = zeros(length(B),1);               % buffer for bandpass filter

if Fs ~= 200
    int_c = (5-1)/(Fs*1/40);
    b = interp1(1:5,[1 2 0 -2 -1].*(1/8)*Fs,1:int_c:5);
else
    b = [1 2 0 -2 -1].*(1/8)*Fs;
end

dbuffer = zeros(length(b),1);               % buffer for derivative filter

MA=0.15*Fs;
MA=ones(1,MA)./MA;
mbuffer=zeros(length(MA),1);                % buffer for moving average filter
%% define parameters

i =0;
fs = 200;
trained = 0;
leng = round(0.075*fs);
MAecg_section = zeros(1,leng);
filtecg_section = zeros(1,0.15*fs); %0.15
global THR_SIG THR_NOISE SIG_LEV NOISE_LEV THR_SIG1 THR_NOISE1 SIG_LEV1 NOISE_LEV1
Beat_C = 0;                                 % Raw Beats
Beat_C1 = 0;                                % Filtered Beats
skip = 0;
```

```
m_selected_RR = 0;
mean_RR = 0;
locs = 0;
pks = 0;
Slope1 = 0;
Slope2 = 0;
slope_all = [];
fil_ecg_train = [];
MA_ecg_train = [];
locs_all = [];
pks_all = [];
count_down = 0;
QRSPEAK_all = [];

if rem(round(0.03*fs),2) == 0
    mov_win_size = round(0.03*fs)+1;
else
    mov_win_size = round(0.03*fs);
end

mov_win = zeros(1,mov_win_size);
MA2=ones(1,mov_win_size)./mov_win_size;
m2buffer=zeros(mov_win_size,1);
Ts = 1/Fs;
TMAX = 40;
ecg = 0;
t = 0;
tic
j = 1;
while toc <= TMAX

     while j<500
        out = fgetl(s);
        ecg = str2double(out);
        trash(j) = ecg;
        j = j+1;
     end

    i = i+1;
    out = fgetl(s);
    ecg = str2double(out);
    ecg_store(i)=ecg;

    % bandpass
    [fil_ecg bbuffer_new] = BandPass(B, A, ecg,bbuffer);
    bbuffer_new(1,end)=fil_ecg;
    bbuffer=bbuffer_new;
    fil_ecg_all(i) = fil_ecg;

    % derivative and square
    [deri_ecg dbuffer_new]= Derivative(fil_ecg,b,dbuffer);
    dbuffer_new(1,end)=deri_ecg;
    dbuffer=dbuffer_new;
    deri_ecg_all(i) = deri_ecg;

    % moving average
    [MA1_ecg mbuffer_new]= MavgFilter(deri_ecg,MA,mbuffer);
    mbuffer_new(1,end)=MA1_ecg;
    mbuffer=mbuffer_new;
    MA1_ecg_all(i) = MA1_ecg;
    % MA2
    [MA_ecg m2buffer_new]= MavgFilter(MA1_ecg,MA2,m2buffer);
    m2buffer_new(1,end)=MA_ecg;
    m2buffer=m2buffer_new;
    MA_ecg_all(i) = MA_ecg;

    MAecg_section = [MAecg_section(2:end) MA_ecg];
    filtecg_section = [filtecg_section(2:end) fil_ecg];

    if trained ==1 && count_down==0

        mov_win = [mov_win(2:end) MA_ecg];
        if mov_win((mov_win_size-1)/2)<mov_win((mov_win_size+1)/2) && mov_win((mov_win_size+3)/2)<mov_win((mov_win_size+1)/2)          %local maximum

                locs = i-(mov_win_size+1)/2+1;
                pks = mov_win((mov_win_size+1)/2);
                locs_all = [locs_all locs];
                pks_all = [pks_all pks];
                [y,x] = max(filtecg_section);          %locate the corresponding peak in the filtered signal

        end

     %% ================= update the heart_rate ===================== %%
        if Beat_C >= 9
            diffRR = diff(qrs_i(Beat_C-8:Beat_C));                         % calculate RR interval
            mean_RR = mean(diffRR);                                        % calculate the mean of 8 previous R waves interval
            comp =qrs_i(Beat_C)-qrs_i(Beat_C-1);                           % latest RR

            if comp <= 0.92*mean_RR || comp >= 1.16*mean_RR
         % ------ lower down thresholds to detect better in MVI --------- %
                    THR_SIG = 0.5*(THR_SIG);
                    THR_SIG1 = 0.5*(THR_SIG1);
            else
                m_selected_RR = mean_RR;                                   % The latest regular beats mean
            end

        end
    %% ===================  find noise and QRS peaks =================== %%
            if pks >= THR_SIG
              % ------ if No QRS in 360ms of the previous QRS See if T wave ------%
               if Beat_C >= 3
                  if (i-qrs_i(Beat_C)) <= round(0.3600*fs)
                      Slope1 = mean(diff(MAecg_section));       % mean slope of the waveform at that position
```

```

          if abs(Slope1) <= abs(0.5*(Slope2))
                                          % slope less
              then 0.5 of previous R
             skip = 1;

                % T wave identification
             % ------ adjust noise levels ------- %
             NOISE_LEV1 = 0.125*y + 0.875*NOISE_LEV1;
             NOISE_LEV = 0.125*pks + 0.875*NOISE_LEV;
          else
             skip = 0;
          end

       end
    end
    %----------- skip is 1 when a T wave is detected
       -------------- %
    if skip == 0
      Beat_C = Beat_C + 1;
      qrs_i(Beat_C) = locs;
    %--------------- bandpass filter check threshold
       --------------- %
      if y >= THR_SIG1
          Beat_C1 = Beat_C1 + 1;
          count_down = round(0.2*fs);
          QRSPEAK = 1;
          fprintf(s,'%c',QRSPEAK);
                                             %QRS
             detected, generate pulse
          Slope2 = mean(diff(MAecg_section)); % mean
             slope of previous R wave
          SIG_LEV1 = 0.125*y + 0.875*SIG_LEV1;
                                   % adjust threshold
             for bandpass filtered sig
          ecg_section_all{i} = filtecg_section;
      end
     SIG_LEV = 0.125*pks + 0.875*SIG_LEV ;
                                 % adjust Signal level
    end
elseif (THR_NOISE <= pks) && (pks < THR_SIG)
     NOISE_LEV1 = 0.125*y + 0.875*NOISE_LEV1;
                               % adjust Noise level in
         filtered sig
     NOISE_LEV = 0.125*pks + 0.875*NOISE_LEV;
                              % adjust Noise level in
        MVI
elseif pks < THR_NOISE && pks~=0
    NOISE_LEV1 = 0.125*y + 0.875*NOISE_LEV1;
                               % noise level in
       filtered signal
    NOISE_LEV = 0.125*pks + 0.875*NOISE_LEV;
                              % adjust Noise level in
       MVI
end
%% ================== adjust the threshold with SNR
   ============= %%
if NOISE_LEV ~= 0 || SIG_LEV ~= 0
    THR_SIG = NOISE_LEV + 0.25*(abs(SIG_LEV -
       NOISE_LEV));
```

```
                THR_NOISE = 0.5*(THR_SIG);
            end
            %------ adjust the threshold with SNR for bandpassed signal -------- %
            if NOISE_LEV1 ~= 0 || SIG_LEV1 ~= 0
                THR_SIG1 = NOISE_LEV1 + 0.25*(abs(SIG_LEV1 - NOISE_LEV1));
                THR_NOISE1 = 0.5*(THR_SIG1);
            end
        %--------- take a track of thresholds of smoothed signal -------------%
            SIGL_buf(i) = SIG_LEV;
            NOISL_buf(i) = NOISE_LEV;
            THRS_buf(i) = THR_SIG;
            %-------- take a track of thresholds of filtered signal ----------- %
            SIGL_buf1(i) = SIG_LEV1;
            NOISL_buf1(i) = NOISE_LEV1;
            THRS_buf1(i) = THR_SIG1;
            % ----------------------- reset parameters -------------------------- %
            skip = 0;
            not_nois = 0;
            ser_back = 0;
            pks = 0;
            locs = 0;
            QRSPEAK_all(i) = QRSPEAK;
    end

    QRSPEAK = 0;

    if length(fil_ecg_all)>5*fs && length(MA_ecg_all)>5*fs && trained==0
        trainning(MA_ecg_all(1,2*fs:end),fil_ecg_all(1,2*fs:end),fs);
        trained = 1;
    end


    if count_down ~= 0
    count_down = count_down -1;
    end


end

fclose(s)
delete(s)


%% Filter Functions
function [data bbuffer_new] = BandPass(b, a,value,bbuffer)
    k = 1;
    while(k<length(b))
        bbuffer(k) = bbuffer(k+1);
        k=k+1;
    end
    bbuffer(length(b)) = 0;
    k = 1;
    while(k<(length(b)+1))
```

```
        bbuffer(k) = bbuffer(k) + value .* b(k);
        k=k+1;
    end

    k = 1;
    while(k<length(b))
        bbuffer(k+1) = bbuffer(k+1) - bbuffer(1) * a(k+1);
        k=k+1;
    end

    data = bbuffer(1);
    bbuffer_new=bbuffer;
end

% Derivative and square
function [data,dbuffer_new] = Derivative(value,b,dbuffer)
    k = 1;
        N = length(b);
    while(k<N)
        dbuffer(k) = dbuffer(k+1);
        k=k+1;
    end
    dbuffer(N) = value;
    k = 1;
    while(k<N+1)
        dbuffer(k) = dbuffer(k) + value .* b(k);
        k=k+1;
    end
    dbuffer_new=dbuffer;
    data = dbuffer(1)^2;
end

function [data mbuffer_new] = MavgFilter(value,MA,mbuffer)
    k = 1;
    N=length(MA);
    while(k<N)
        mbuffer(k) = mbuffer(k+1);
        k=k+1;
    end
    mbuffer(N) = 0;
    k = 1;
    while(k<N+1)
        mbuffer(k) = mbuffer(k) + value.*MA(k);
        k=k+1;
    end
    mbuffer_new=mbuffer;
    data = mbuffer(1);
end

function  trainning (ecg_m,ecg_h,fs)

global THR_SIG THR_NOISE SIG_LEV NOISE_LEV THR_SIG1 THR_NOISE1 SIG_LEV1 NOISE_LEV1

%% initialize the training phase (2 seconds of the signal) to determine the THR_SIG and THR_NOISE
THR_SIG = max(ecg_m(1:2*fs))*1/3;                                   % 0.25 of the max amplitude
```

```
THR_NOISE = mean(ecg_m(1:2*fs))*1/2;                                        % 0.5 of the mean signal is considered to be noise
SIG_LEV= THR_SIG;
NOISE_LEV = THR_NOISE;

%% Initialize bandpath filter threshold(2 seconds of the bandpass signal)
THR_SIG1 = max(ecg_h(1:2*fs))*1/3;                                              % 0.25 of the max amplitude
THR_NOISE1 = mean(ecg_h(1:2*fs))*1/2;
SIG_LEV1 = THR_SIG1;                                                                        % Signal level in Bandpassed filter
NOISE_LEV1 = THR_NOISE1;          % Noise level in Bandpassed filter

end
```