Supplementary Material

An Embedded System based on Raspberry Pi for Effective Electrocardiograph Monitoring

Yusra M Obeidat[1], Ali Mohammad Alqudah[2]

[1]*Electronic Engineering Department, Yarmouk University, Irbid-Jordan*

[1]*Biomedical Engineering Department, Yarmouk University, Irbid-Jordan*

**Corresponding author: yusra.obeidat@yu.edu.jo*

S.1: The ECG Signal Generation

The following is the code used to generate the ECG signal.

```
Byte const
display[]={69,69,71,71,72,72,72,72,72,71,71,71,69,69,69,71,71,72,72,72,74,74,74,74,72,72,72,7
2,71,71,71,71,71,71,69,69,69,71,71,71,72,72,74,74,74,74,72,72,71,71,69,67,67,67,66,66,66,66,6
6,64,64,64,63,61,61,59,59,58,58,59,59,61,63,64,64,64,64,64,64,64,64,66,66,66,66,64,63,61,59,5
6,56,56,56,58,61,63,63,64,64,63,61,59,58,58,59,59,61,64,66,67,69,69,69,71,69,69,69,67,67,66,6
4,64,63,64,66,67,69,72,74,76,77,76,74,72,71,67,66,64,63,63,61,59,56,54,53,53,51,51,53,54,54,5
6,56,56,56,56,54,54,54,53,53,51,51,49,48,46,46,46,46,46,46,46,46,44,44,43,41,39,39,39,39,39,4
1,41,39,38,36,35,35,38,44,58,74,95,122,150,178,204,225,243,253,255,247,232,209,179,146,112,
79,49,26,10,2,0,3,12,20,30,36,41,44,44,43,41,38,38,36,36,36,36,38,38,38,36,36,36,35,35,35,35,3
5,35,36,36,38,39,39,41,41,41,41,41,41,41,41,39,39,39,39,41,41,43,43,44,46,46,46,46,46,46,46,4
6,46,48,49,51,53,54,56,58,58,59,59,61,63,64,66,67,71,72,72,74,74,74,74,74,76,77,79,81,84,87,8
9,92,94,95,95,97,97,97,97,97,99,99,100,102,104,105,107,107,105,104,100,95,92,89,87,86,84,82,
81,77,74,69,64,59,54,51,48,48};

    int i;
    void setup() {
      // put your setup code here, to run once:
    DDRD=255;
    PORTD=0;
    }
    void loop() {
      // put your main code here, to run repeatedly:

    for (i=0;i<=339;i++){
    PORTD=display[i];
```

```
delayMicroseconds(1500);
}
i=0;
}
```

S.2 The Gain of INA122

INA122 is a precision instrumentation amplifier that is used for accurate, low noise differential signal acquisition. It is known of its very low input bias current of only 25nA max, very low voltage noise of only 60nV/sqrt(Hz).

Based on the datasheet of the INA122 instrumentation amplifier:

$G=5+200k/R_G$

If R_G=4.7k, then the gain is 47.55

S.3 Filters and Peak Detector

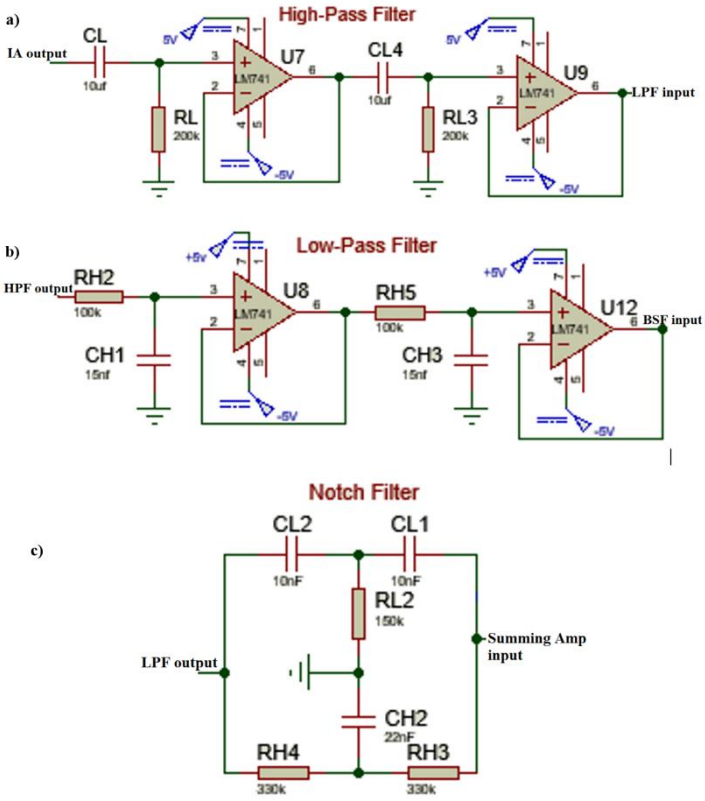


Fig.S.1 The second order HPF. **b)** The second order LPF. **c)** The second order BSF.

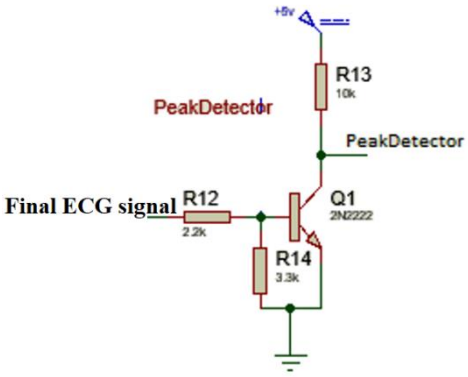


Fig.S.2 The Peak Detector

S.4 Circuit Connection on Proteus Professional Software

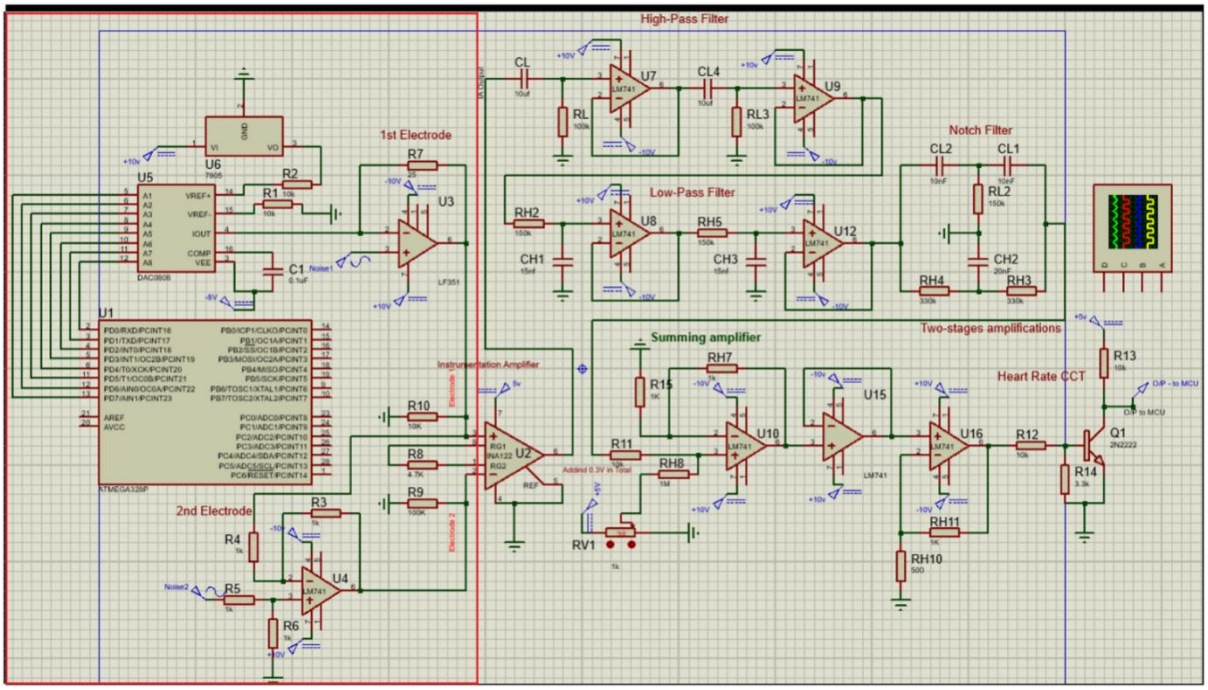


Fig.S.3 The ECG Circuit Simulation Using Proteus Professional Software.

S.5 The Printed Circuit Board Schematic and Layout on Altium

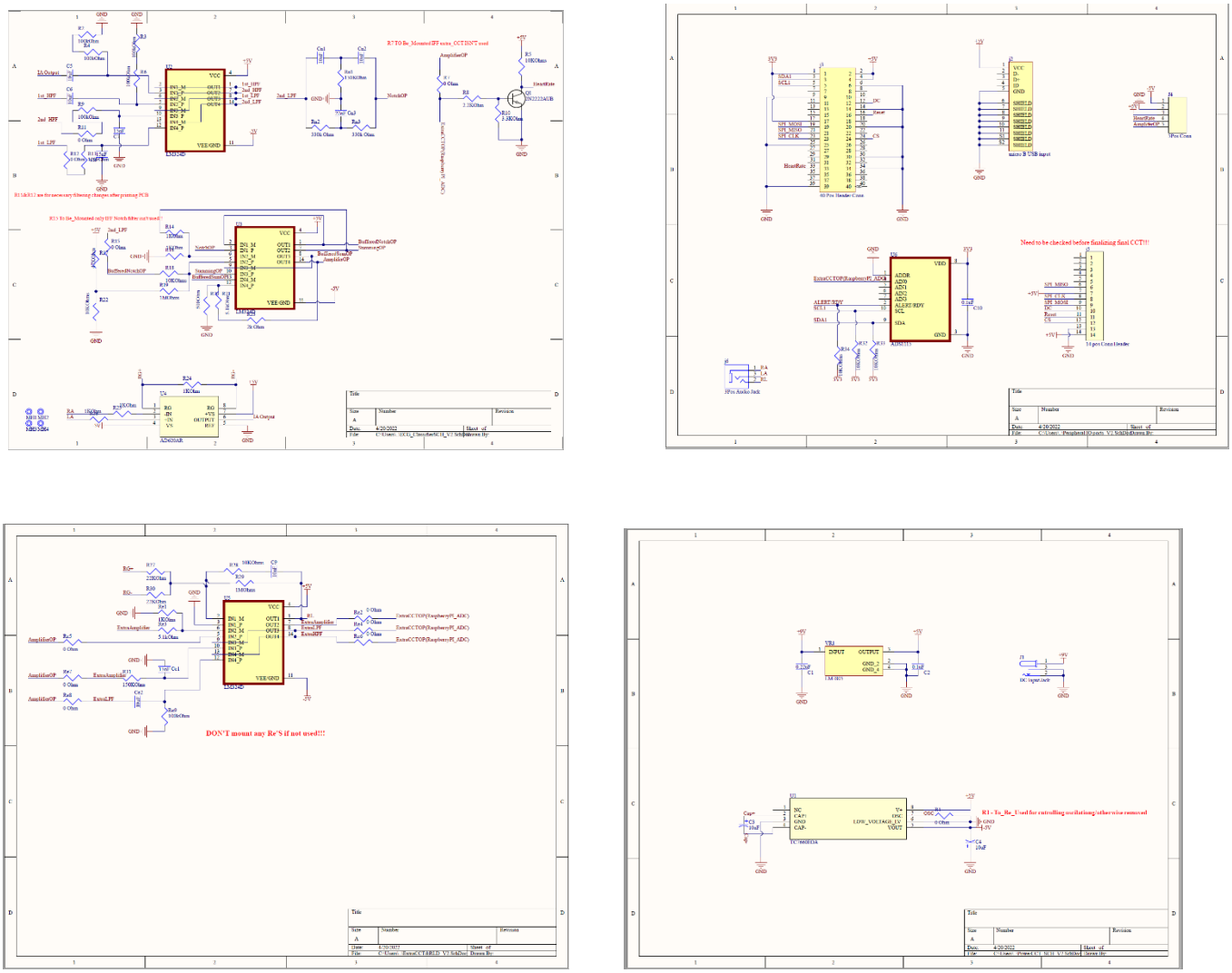


Fig.S.4 The Printed Circuit Board Schematic.

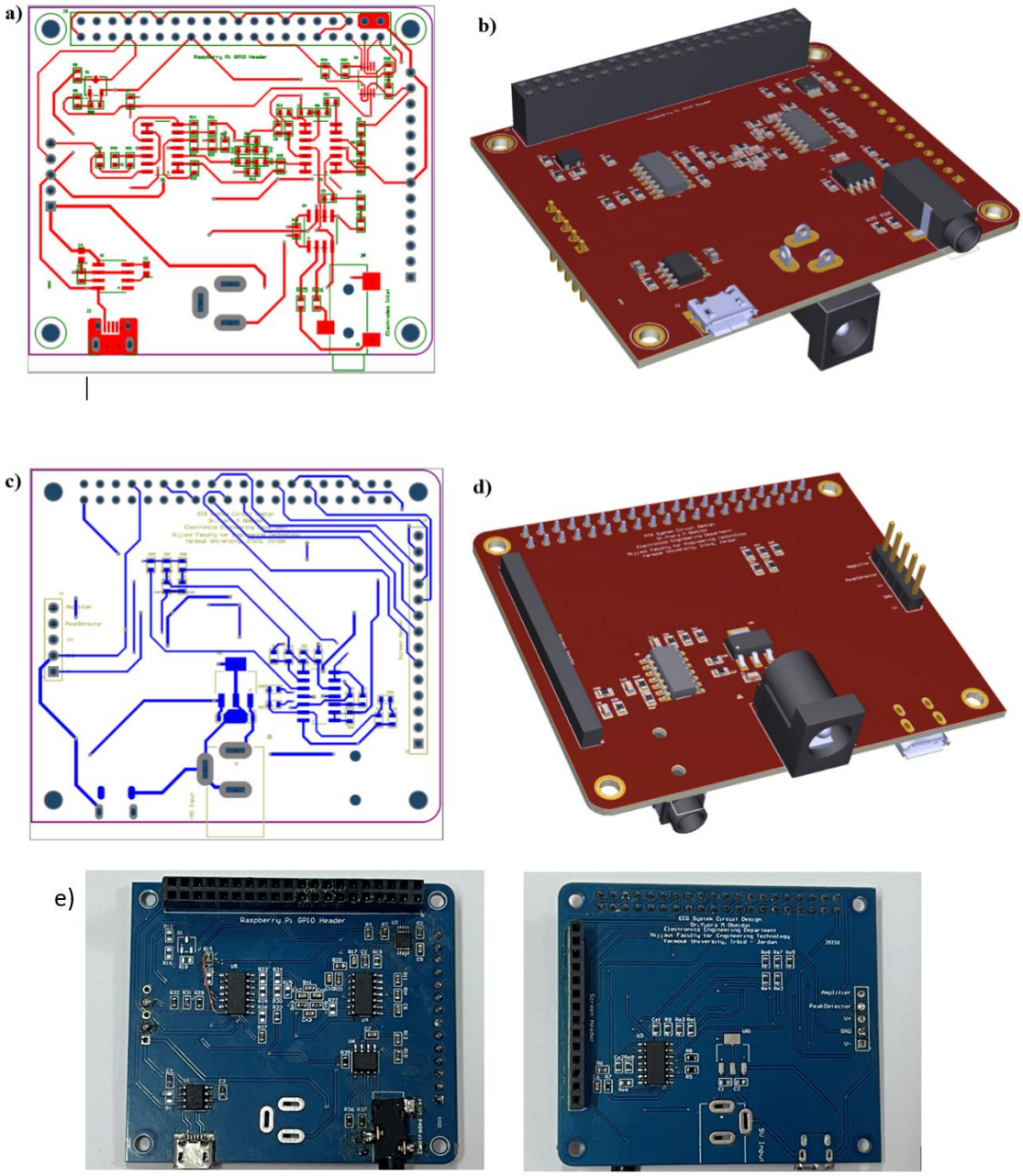
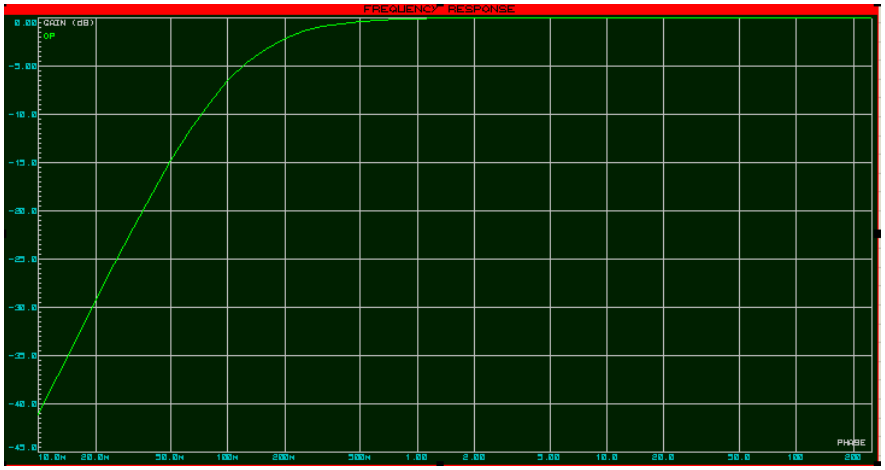


Fig.S.5 The ECG PCB Layout: **a)** The Top Layer (2D). **b)** The Top Layer (3D). **c)** The Bottom Layer (2D). **d)** The Bottom layer (3D). e) printed circuit board after soldering all necessary components

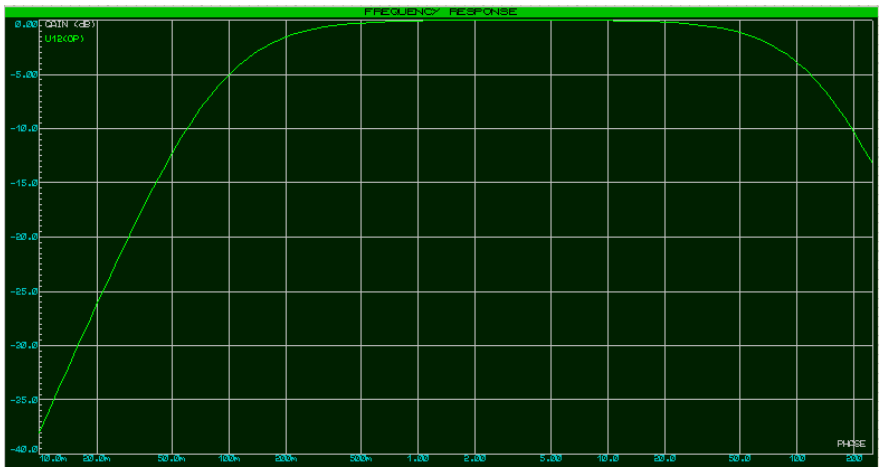
S.6 The Filters Bode Plots

a)

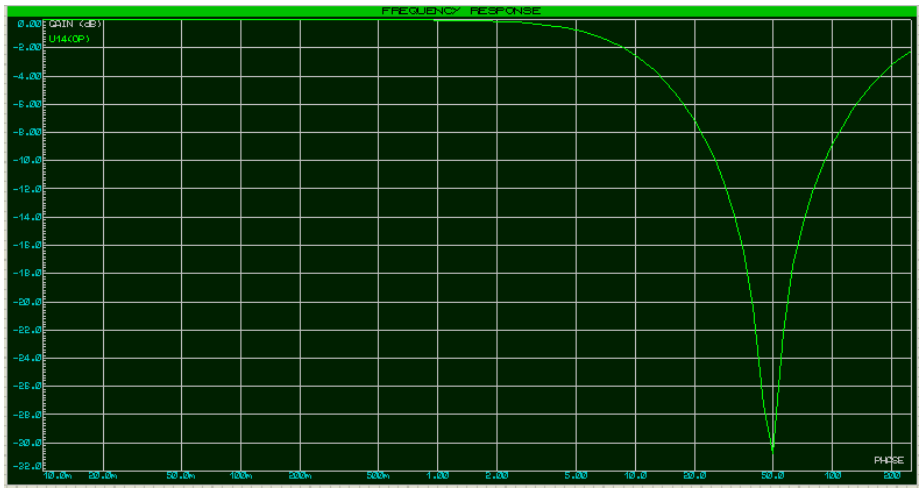


b)

c)



d)



e)

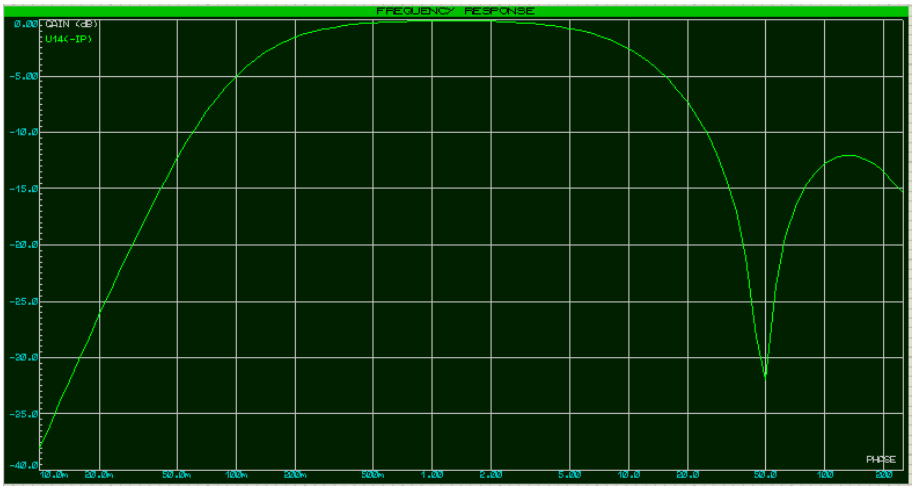


Fig.S.6: The Bode Plots for Filters: a) High Pass Filter. **b)** Low Pass Filter. **c)**Band Pass Filter. **d)** Notch Filter. **e)** All filters**.**

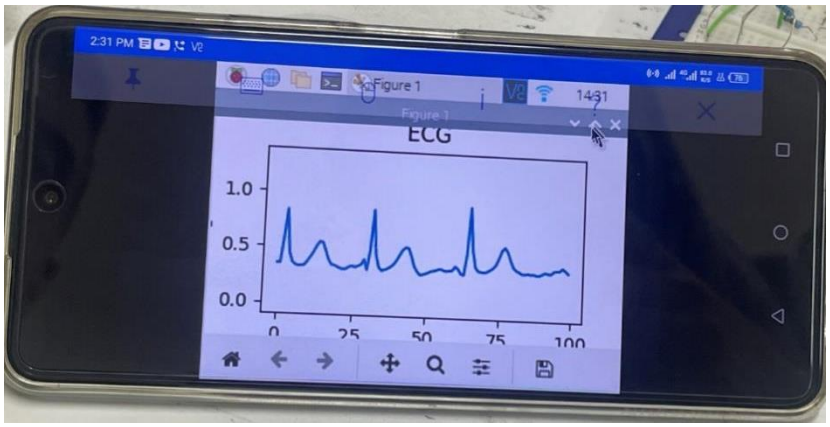


Fig.S.7: An ECG signal from human displayed on a smart phone through VNC viewer.

S.7 The programming code in python

```
fs=1000
x_len = 186         # Number of points to display
y_range = [-0.25, 1.5]  # Range of possible Y values to display
rate = 0
sample=[0] * 100
# load model
model = load_model('ECG_CNNLSTM_Model.h5')
# Create figure for plotting
fig = plt.figure(figsize=(3.3,1.7))
ECG = fig.add_subplot(1, 1, 1)
xs = list(range(0, 100))
ys = [0] * x_len
ECG.set_ylim(y_range)
# Create a blank line. We will update the line in animate
line, = ECG.plot(ys)
# Add labels
plt.title(' ECG ')
plt.xlabel(' Time(ms/18.75) ')
plt.ylabel(' Voltage(V) ')
###
class Pan_Tompkins_QRS():
    def band_pass_filter(self,signal):
    '''
```

```
Band Pass Filter
:param signal: input signal
:return: prcoessed signal
Methodology/Explaination:
Bandpass filter is used to attenuate the noise in the input signal.
To acheive a passband of 5-15 Hz, the input signal is first passed
through a low pass filter having a cutoff frequency of 11 Hz and then
through a high pass filter with a cutoff frequency of 5 Hz, thus
achieving the required thresholds.
The low pass filter has the recursive equation:
  y(nT) = 2y(nT - T) - y(nT - 2T) + x(nT) - 2x(nT - 6T) + x(nT - 12T)
The high pass filter has the recursive equation:
  y(nT) = 32x(nT - 16T) - y(nT - T) - x(nT) + x(nT - 32T)
'''
# Initialize result
result = None
# Create a copy of the input signal
sig = signal.copy()
            # Apply the low pass filter using the equation given
for index in range(len(signal)):
  sig[index] = signal[index]
  if (index >= 1):
    sig[index] += 2*sig[index-1]
  if (index >= 2):
    sig[index] -= sig[index-2]
  if (index >= 6):
    sig[index] -= 2*signal[index-6]
  if (index >= 12):
    sig[index] += signal[index-12]
            # Copy the result of the low pass filter
result = sig.copy()
# Apply the high pass filter using the equation given
for index in range(len(signal)):
  result[index] = -1*sig[index]
```

```
  if (index >= 1):
    result[index] -= result[index-1]
  if (index >= 16):
    result[index] += 32*sig[index-16]
  if (index >= 32):
    result[index] += sig[index-32]
# Normalize the result from the high pass filter
max_val = max(max(result),-min(result))
result = result/max_val
return result
```