A Real Time Application to Identify Alcoholics from ECG Signals

AKARSH N KOLEKAR (1PI13EC009)
APOORV VATSAL (1PI13EC017)
RAKSHITH VISHWANATHA (1PI13EC075)

GUIDE: DR. B NIRANJANA KRUPA

STATUS – since synopsis (21st Jan)

- Raspberry Pi
- Feature Extraction
- Circuit Simulation
- Classifier

RASPBERRY PI

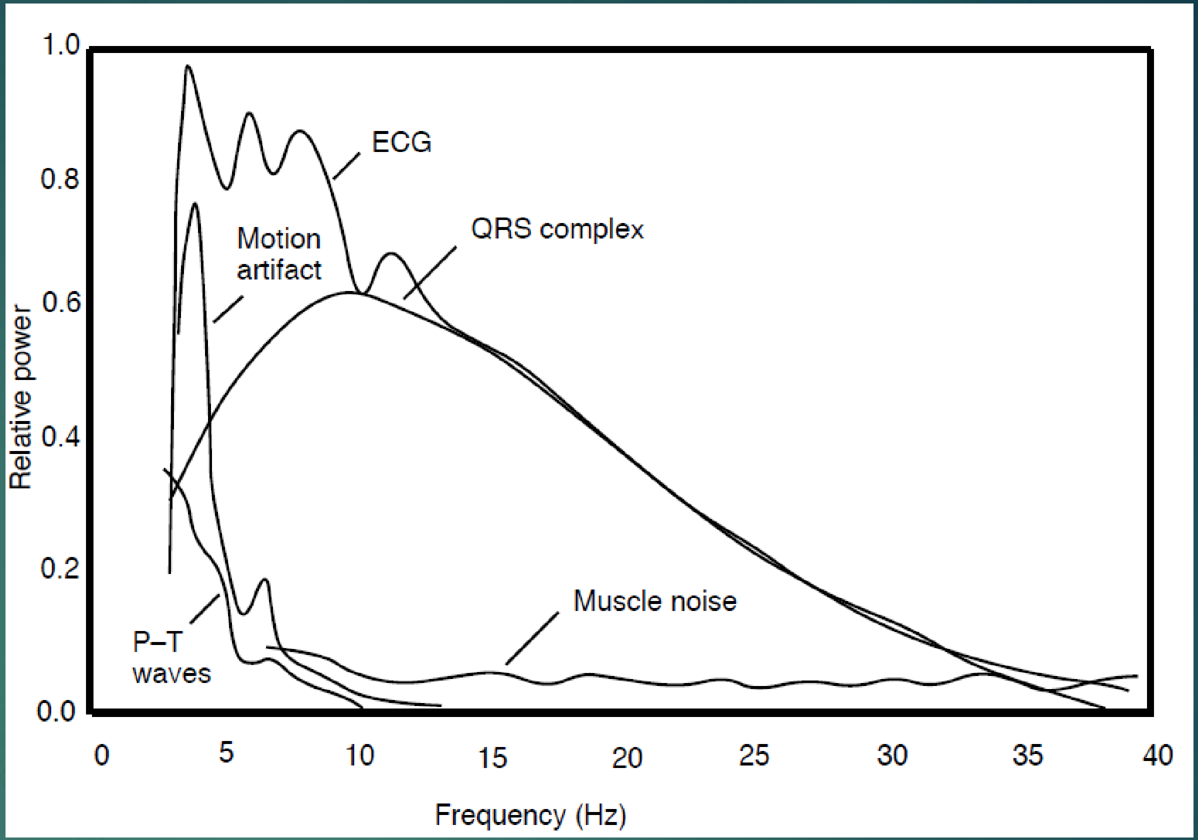
- Installed the OS
- Enabled ssh to accessed terminal
- Set static IP – cmdline.txt was used – IP and netmask provided

FEATURE EXTRATION

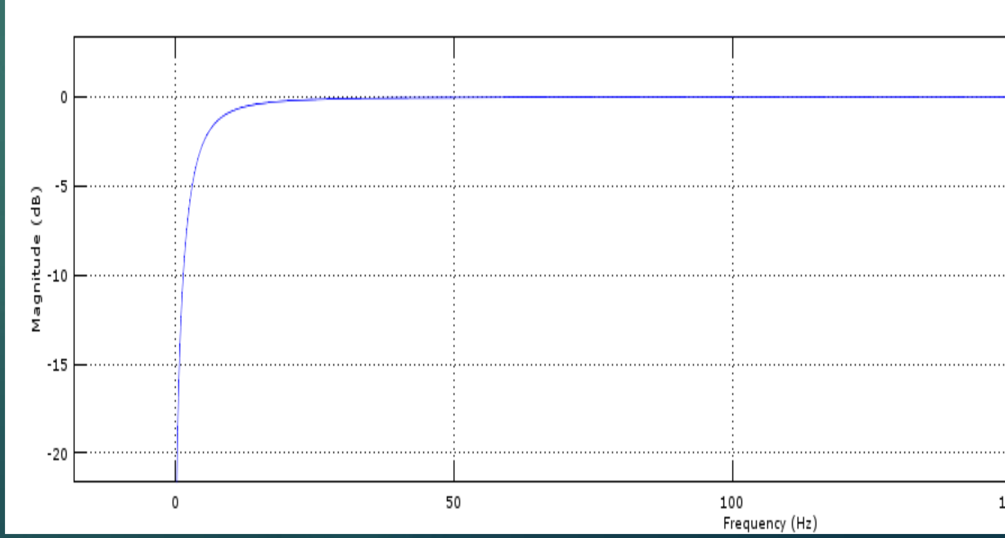
- Time domain
- Frequency domain
- Non-linear techniques

TIME DOMAIN

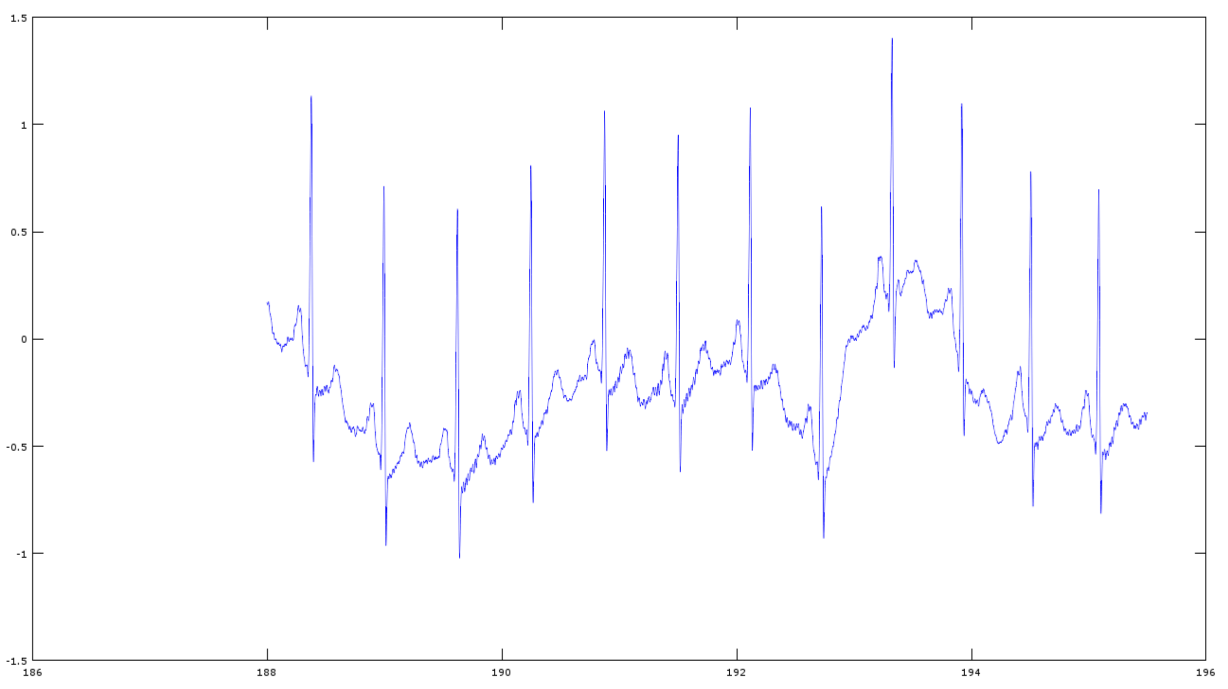
- Based on PSD of generic ECG signal, an IIR HPF filter was designed to remove the low frequency baseline wandering

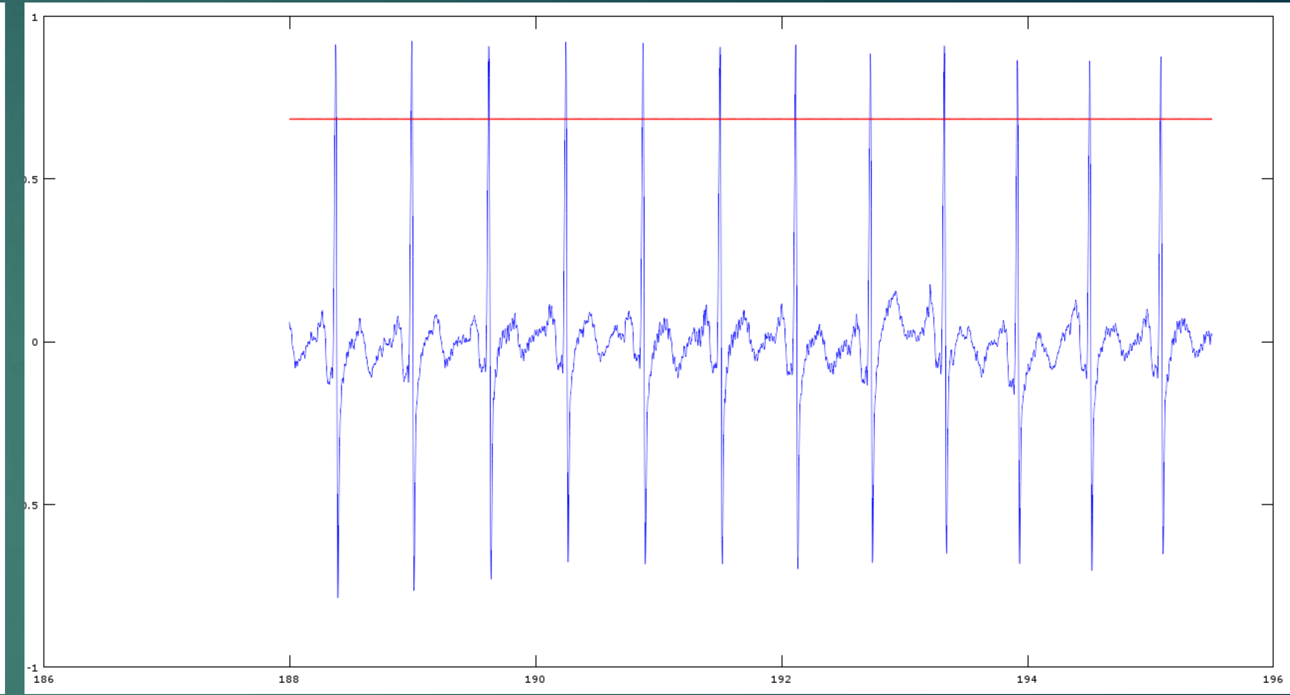



```
>> iir_butt
package loaded ...

Step 1 of 2: Reading requested records. (This may take a few minutes.)...
Step 2 of 2: Parsing data...
signal loaded ...


 Filter characteristics...
   1.00000   0.98574  -0.98574   1.00000  -0.97147
```



TIME DOMAIN






```
t_rr1 = [0, t_rr];
t_rr2 = [t_rr, 0];
t_rr_diff = [t_rr2 - t_rr1](2: end-1); % HOLDS THE RR INTERVALS
```

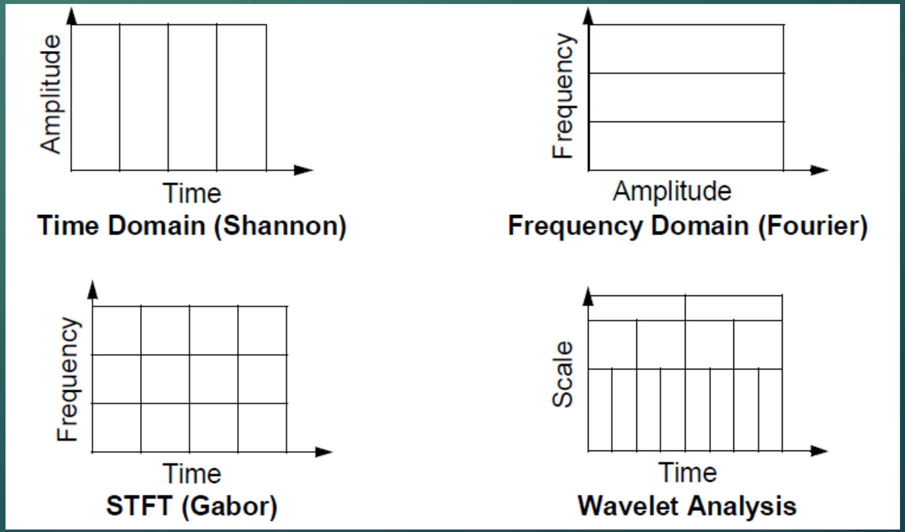
```
f_nn50 = length( find(t_rr_diff_diff >= 50/1000 ) );
f_pnn50 = 100*(f_nn50/length(t_rr_diff));
```

```
t_rr_diff1 = [0, t_rr_diff];
t_rr_diff2 = [t_rr_diff, 0];
t_rr_diff_diff = [t_rr_diff2 - t_rr_diff1](2: end-1);
```

```
f_mean = mean(t_rr_diff);
    f_mean_min = mean(t_rr_diff/(1000*60)); % SAME AS: t_mean/(1000*60);
f_std = sqrt(var(t_rr_diff));
    f_std_min = sqrt(var(t_rr_diff/(1000*60))); % SAME AS: f_std/(1000*60);
f_mean_HR = 1/f_mean_min;
f_std_HR = 1/f_std_min;
f_rms = sqrt( 1/length(t_rr_diff)*sum( (t_rr_diff).^2 ) );
```

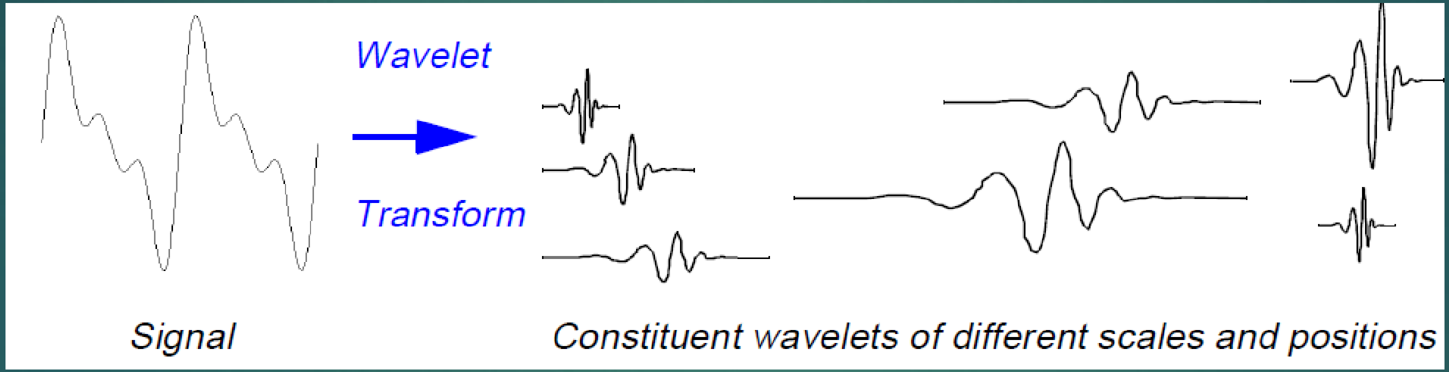
FREQUENCY DOMAIN

- Techniques possible:
  - Fourier Analysis – time information is lost
  - Short Time Fourier Analysis – limited precision based on window size
  - Wavelet Transform



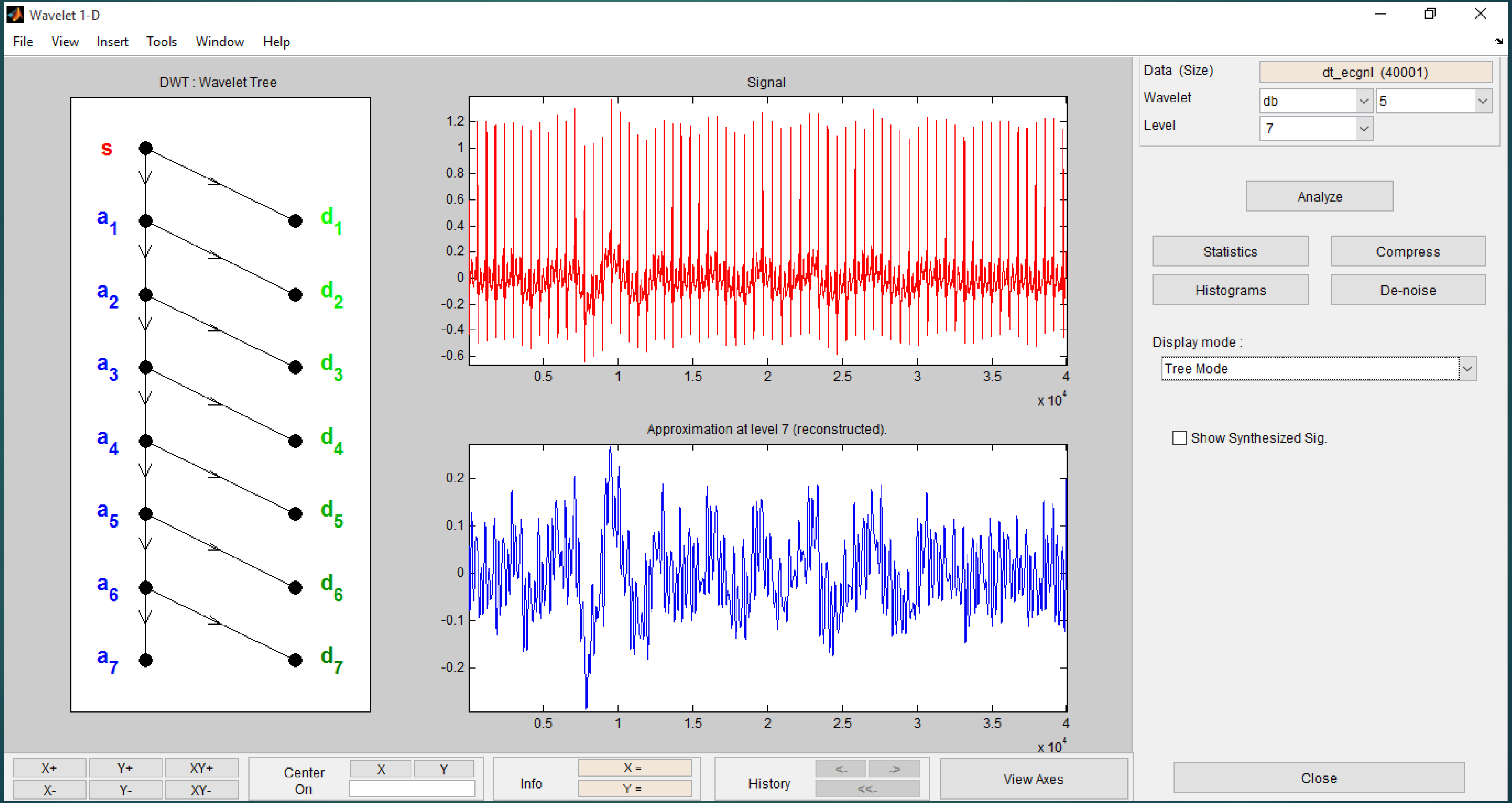
WHY WAVELET

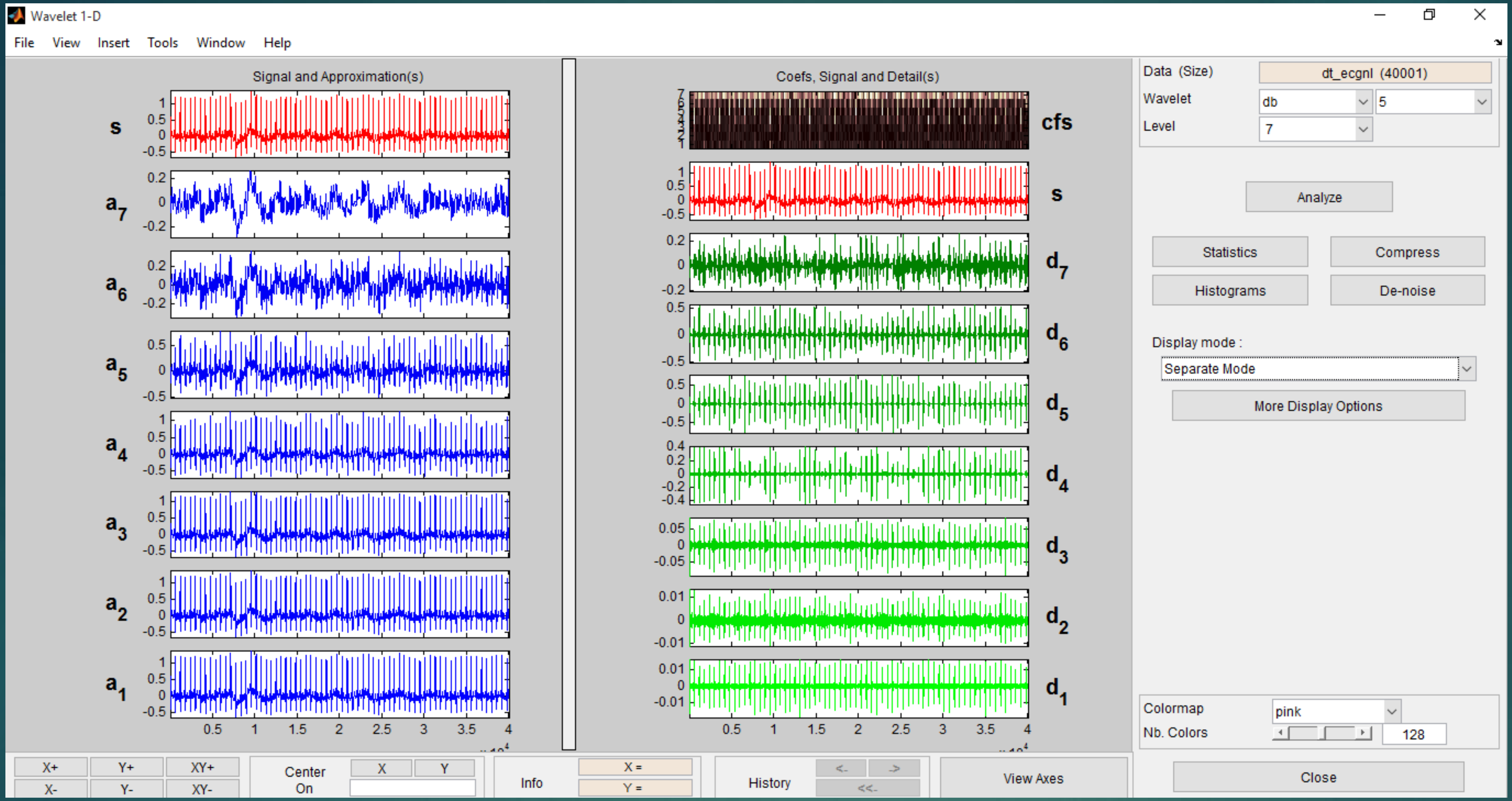
$$C(scale, position) = \int_{-\infty}^{\infty} f(t)\psi(scale, position, t)dt$$



- Contains both frequency and time domain information
- Splits parent signal to different freq components, allowing isolation of low frequency baseline wandering
- R peak detection and RR interval calculation done

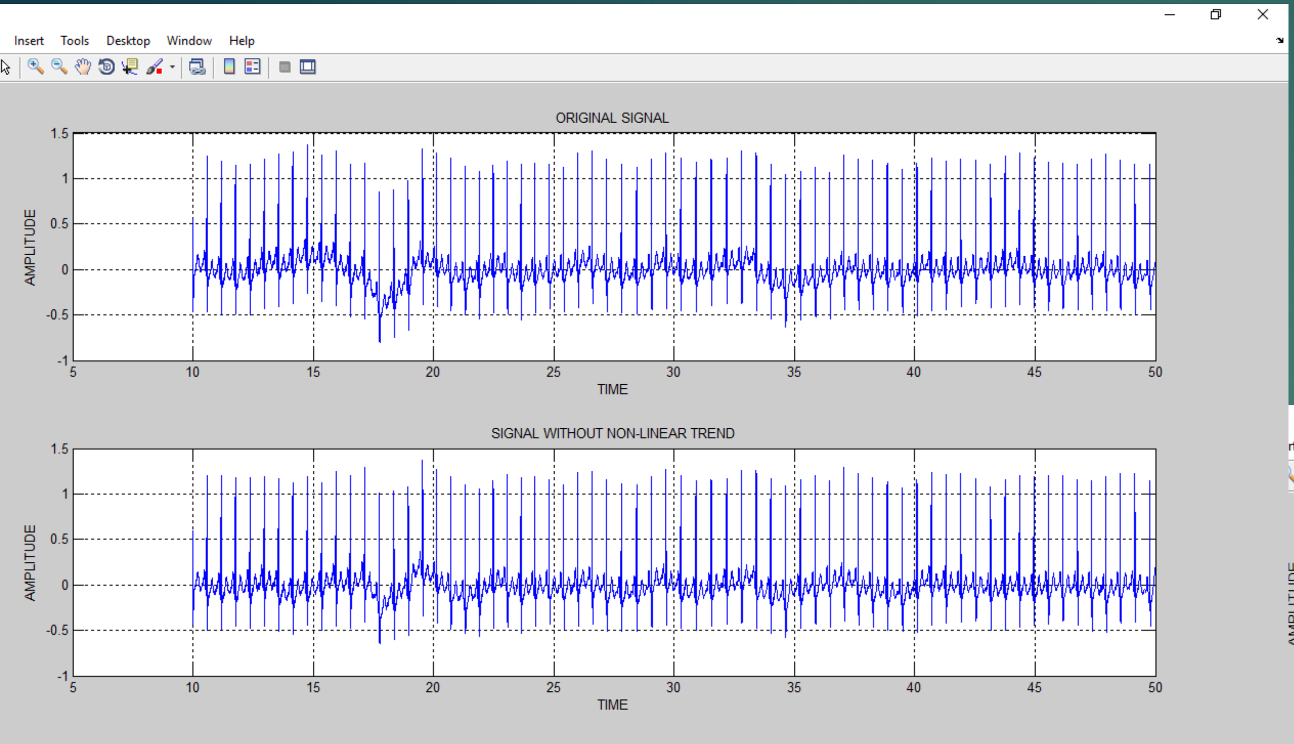
GRAPHS – MATLAB WAVELET TOOLBOX



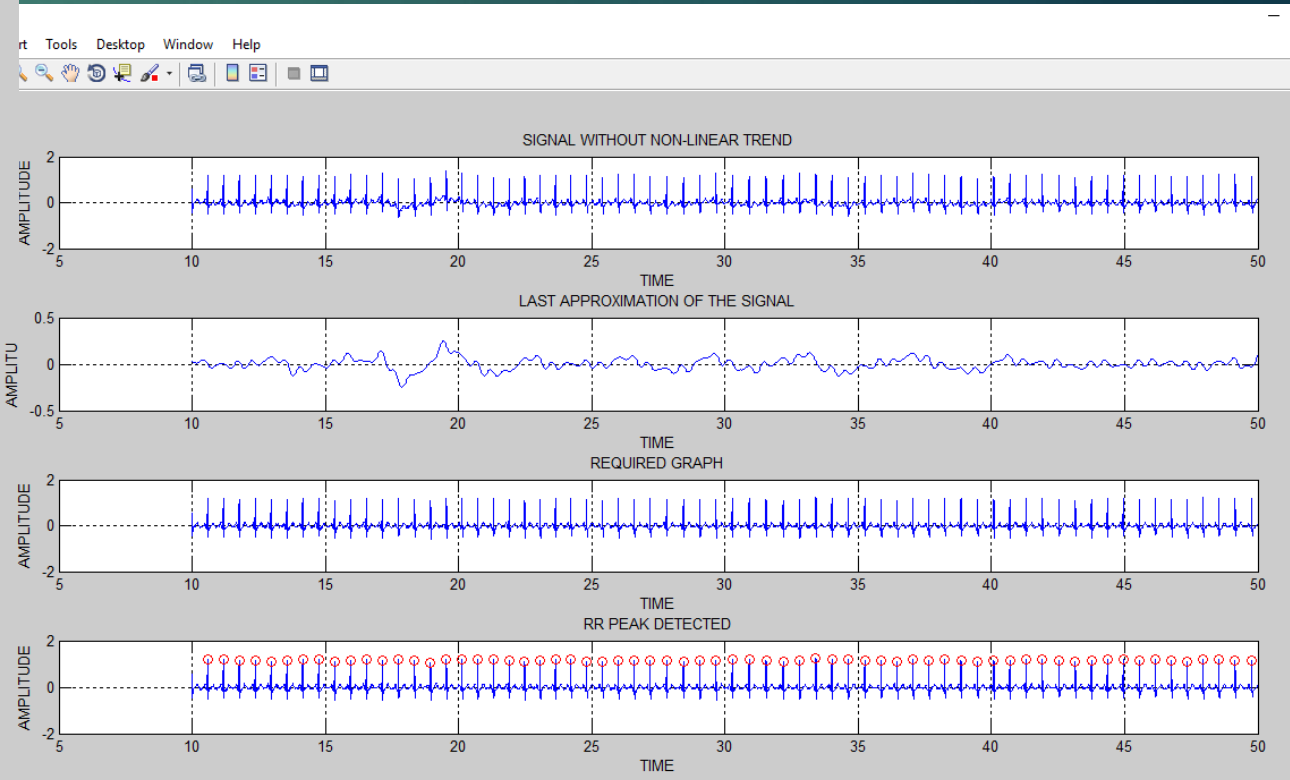


- Wavelet Toolbox of Matlab
- Daubechies (dB5) wavelet considered as mother wavelet
- Scale level used was 7

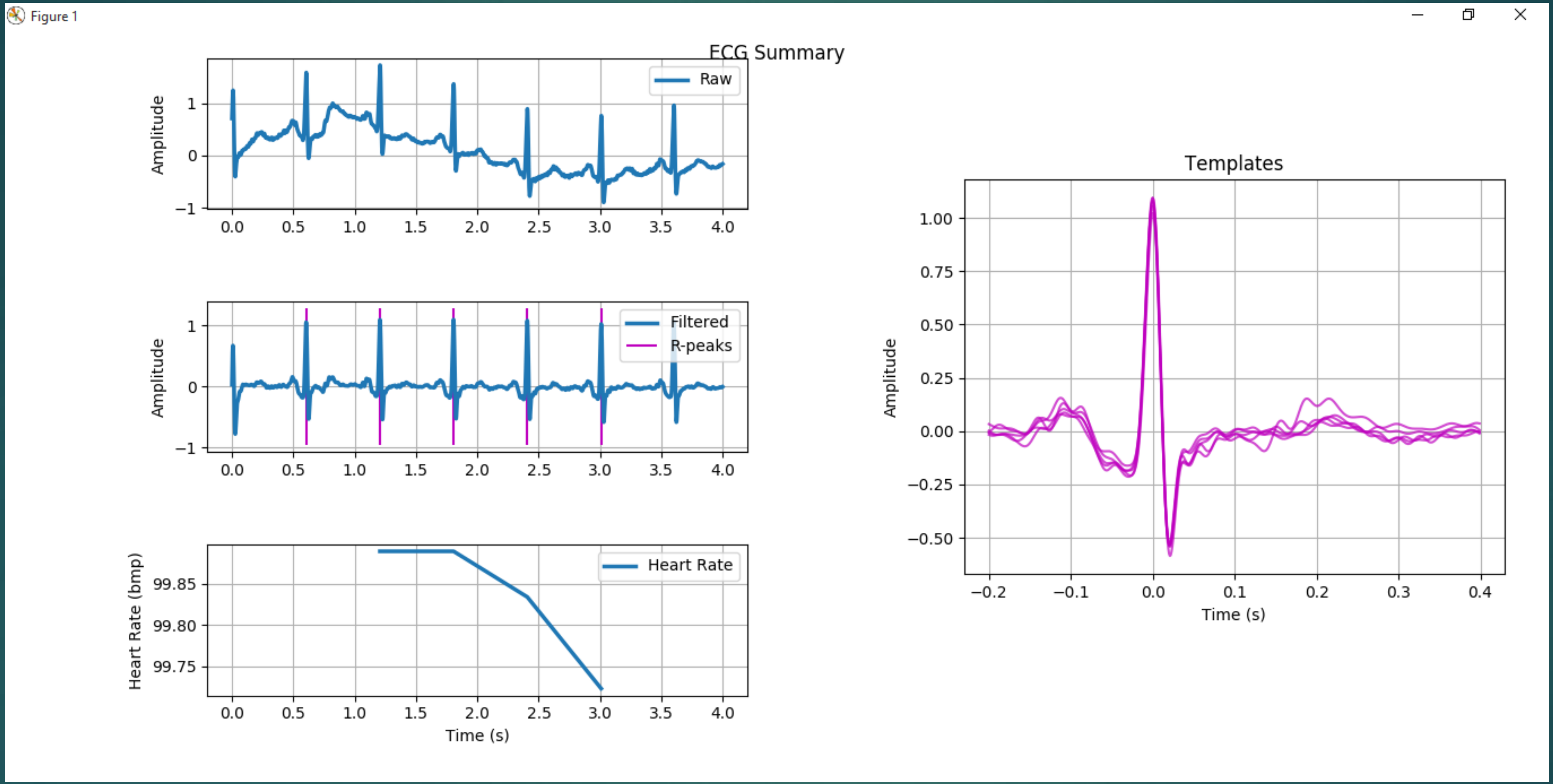
GRAPHS FOR ECG SAMPLE – MATLAB



- Base line wandering removed using Wavelet transform

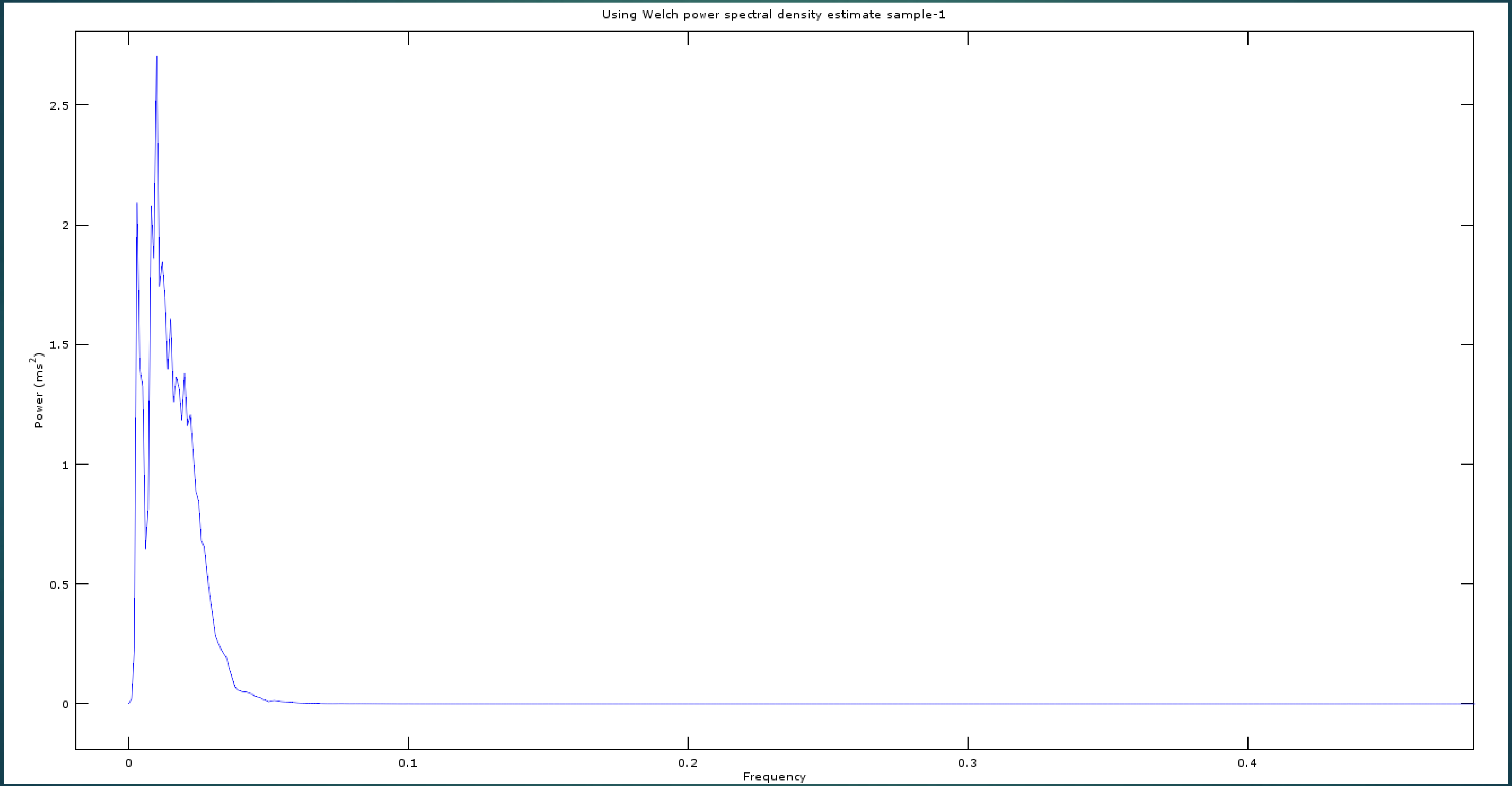


GRAPHS – PYTHON

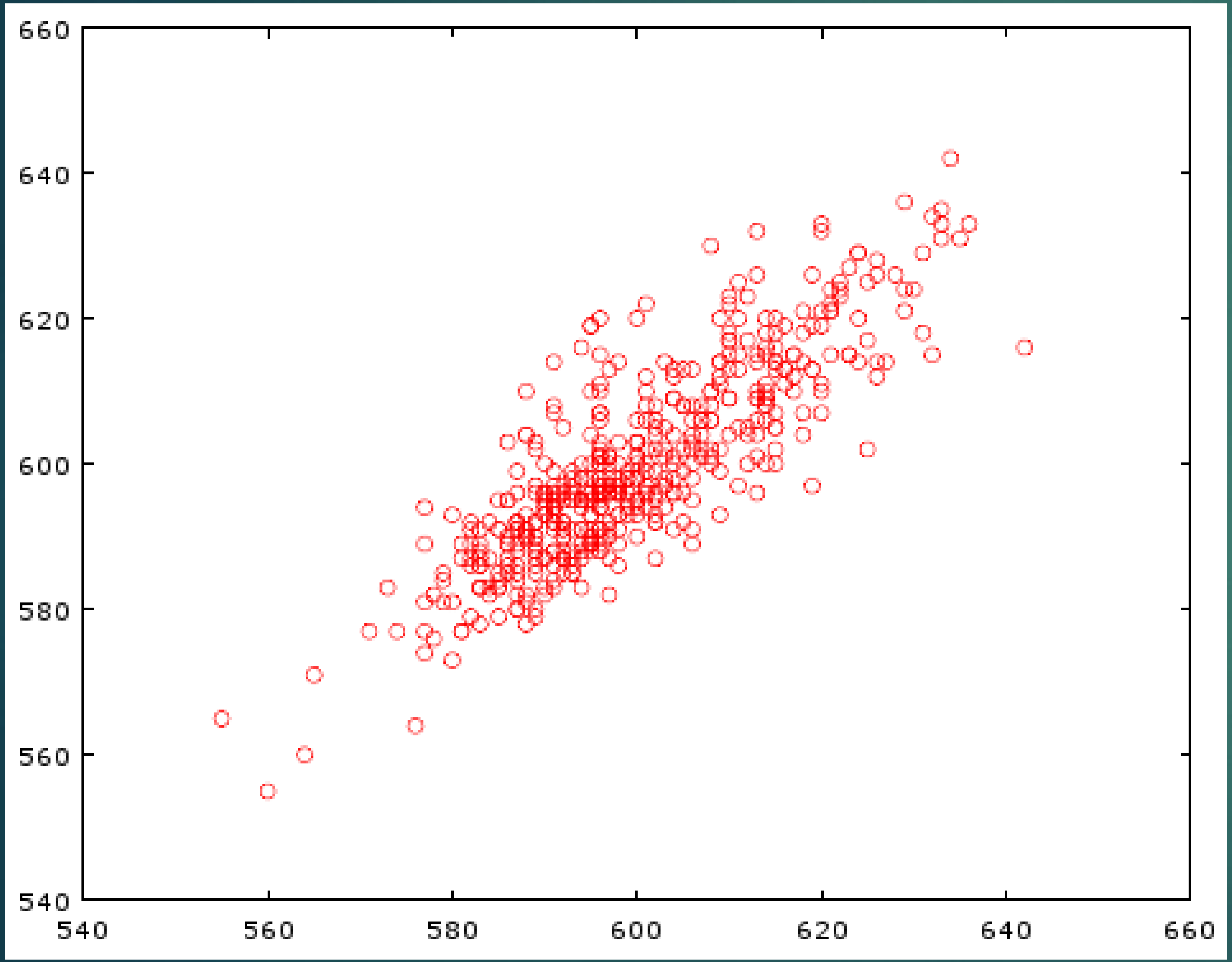


- De-trending the signal and RR peak detection was also implemented on Python

PSD OF RR INTERVAL SERIES



NON-LINEAR: POINCARE PLOT




```
%line x = y
a = 1;
b = -1;
dist = [];
for p = 1:size(points)(1)
  d = abs( (a*points(i, 1) + b*points(i, 2)) / (sqrt(a*a + b*b)) );
  dist = [dist, d];
end
sd1 = sqrt(var(dist));

%line x = -y
a = 1;
b = 1;
dist = [];
for p = 1:size(points)(1)
  d = abs( (a*points(i, 1) + b*points(i, 2)) / (sqrt(a*a + b*b)) );
  dist = [dist, d];
end
sd2 = sqrt(var(dist));
```

NON LINEAR: APPROX & SAMPLE ENTROPY

$$u_j = (\mathrm{RR}_j, \mathrm{RR}_{j+1}, \ldots, \mathrm{RR}_{j+m-1}), \quad j = 1, 2, \ldots N - m + 1$$

$$d(u_j, u_k) = \max\left\{ \left|\mathrm{RR}_{j+n} - \mathrm{RR}_{k+n}\right| \,\middle|\, n = 0, \ldots, m-1 \right\}$$

$$C_j^m(r) = \frac{\text{nbr of } \left\{ u_k \,\middle|\, d(u_j, u_k) \le r \right\}}{N - m + 1} \quad \forall\, k.$$

$$\Phi^m(r) = \frac{1}{N - m + 1} \sum_{j=1}^{N-m+1} \ln C_j^m(r).$$

$$\mathrm{ApEn}(m, r, N) = \Phi^m(r) - \Phi^{m+1}(r).$$

FEATURES EXTRACTED

- Time Domain
  - mean_rr, STD_rr, mean_hr, STD_hr, RMS_rr, rr_50, p_rr_50
- Frequency Domain
  - pk_freq, abs_pow, rel_pow, norm_pow, LF_pow/HF_pow
- Non-Linear
  - Poincare plot’s SD, ApEn, SamEn

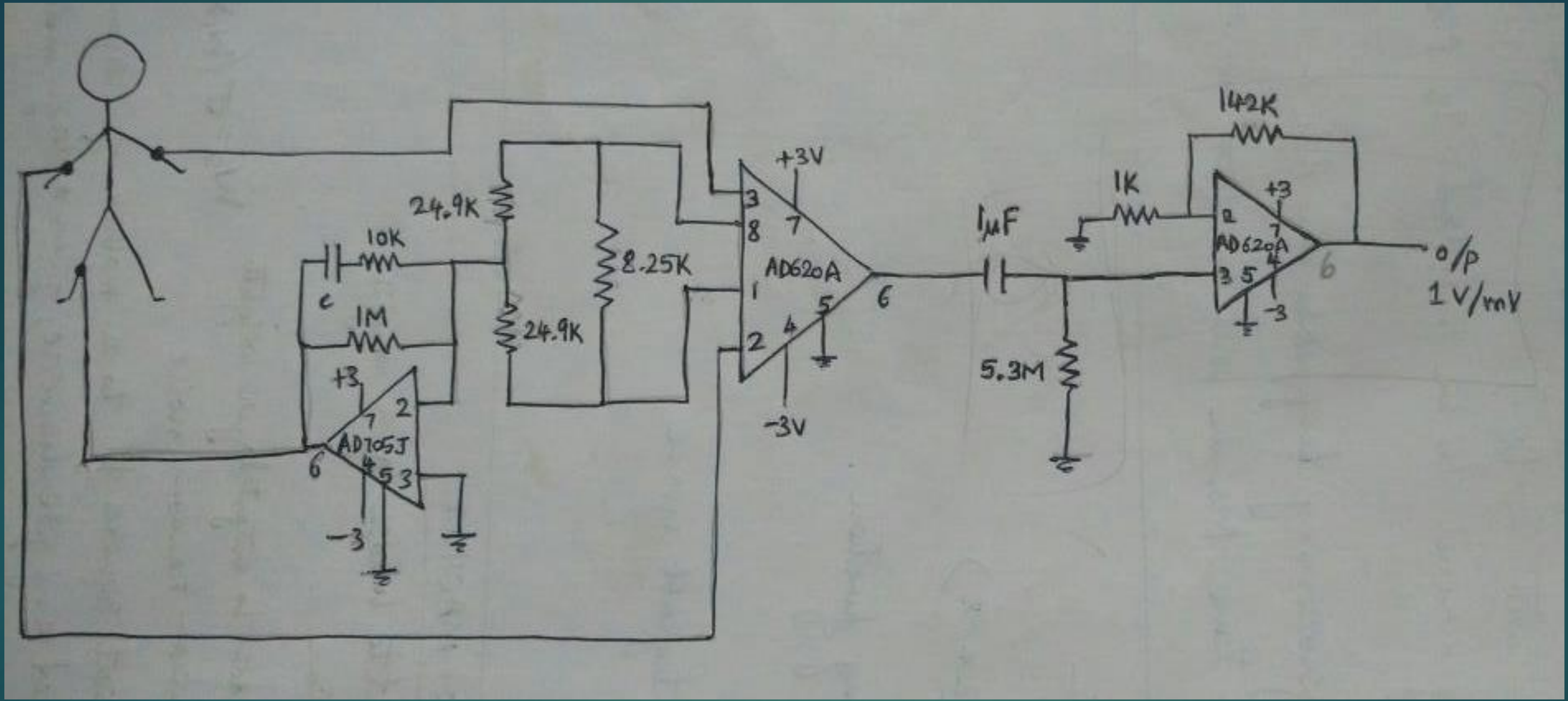
FEATURES WHICH CAN BE EXTRACTED

- Based on need as per classifier performance, following features can be added:
  - Time Domain
    - rr_hist, baseline_width_rr_hist
  - Frequency Domain
    - Sample Entropy – different ‘r’, ‘m’
  - Non-linear
    - DFA (Detrended Fluctuation Analysis), RPA (Recurrence Plot Analysis)

HARDWARE/CIRCUIT

- Studied various circuit designs available to acquire ECG Signals
- Met senior who had designed the circuit for his device
- Simulated circuits designed using sample base designs provided in AD620 Data-sheet, as well as circuits used by senior.
- Made modifications to circuit design as per requirements and issues faced.

ECG CIRCUIT FROM AD620 DATASHEET



- Simulated on MultiSim. The leg circuit to provide virtual ground did not work as expected.
- Erroneous outputs resulted in abandoning the circuit for other options.

MODIFIED CIRCUIT

- The circuit designed by senior was interfaced with a bluetooth module. We will not be using bluetooth.
- It also contained an integrator circuit in the second stage. We did not understand the use of such a part of the circuit, and resulted in erroneous output.
- Certain values of components(Resistors and Capacitors) were changed in order to obtain the necessary gain required.

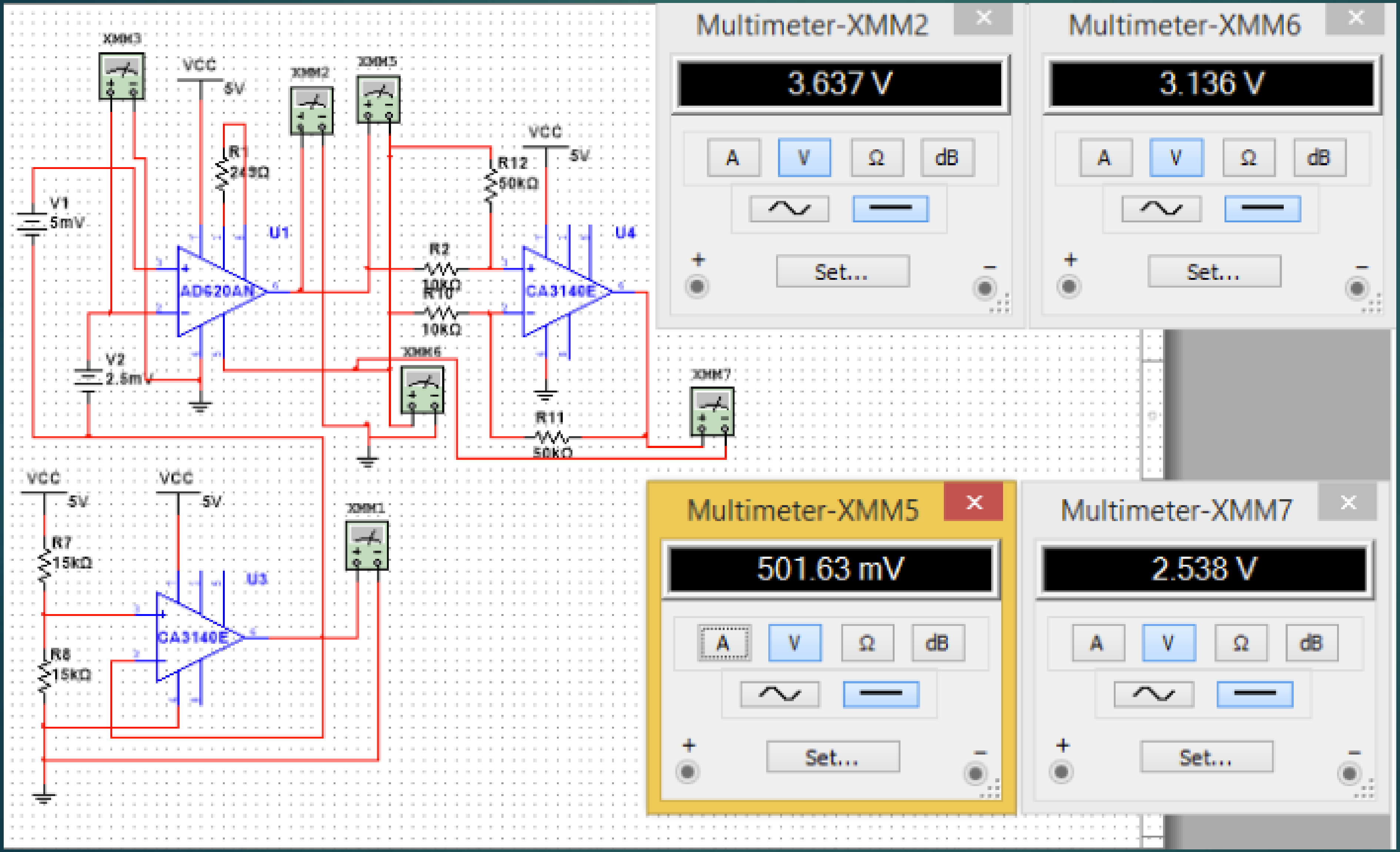
OVERALL IDEA OF CIRCUIT

- Virtual ground:
  A. To provide a reference with respect to which the voltage at the fingertips can be measured.
  B. The virtual ground will be provided to the thumb, so that the device can be hand held.

- Instrumentation amplifier:
  A. Used to measure the differential voltage between the two fingertips on both hands with respect to the virtual ground(thumb).
  B. This stage provides most of the amplification.

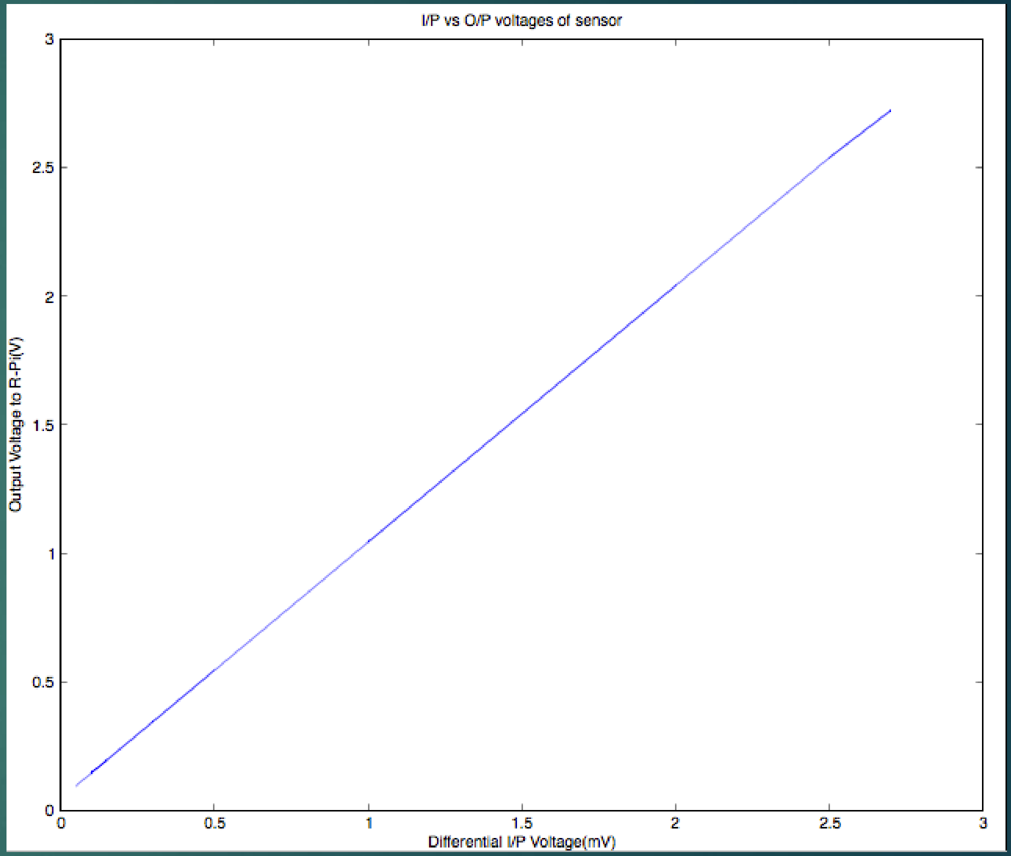
- Regular amp:
  A. To provide secondary amplification after the instrumentation amplifier, i.e., to spread out the gains.

MODIFIED CIRCUIT



- R_G=249Ohm
- G_1=200
- $V_{offset1}$=3.167mV
- $V_{offset2}$=45.8mV
- G_2=5

CIRCUIT DESIGN FOR R-PI COMPATIBILITY



- Max Differential I/P voltage=2.7mV
- Leads to saturated output of 2.78V
- Max I/P to each electrode=10.2mV
- R-Pi does not have analog I/Ps
- External ADC required(MPC3008)
- 10 bit ADC[0-1023]
- R-Pi can take I/P from 0-3.3V
- Resolution of R-Pi=3.22mV
- If we increase supply voltage from 5V, we can increase range of I/P

CLASSIFIER

- Found ELM documentation by developer.
- Currently understanding ELM in order to implement it.
- Algorithm:

$$\hat{\mathbf{Y}} = \mathbf{W}_2\sigma(\mathbf{W}_1 x)$$

W1 is the matrix of input-to-hidden-layer weights, σ is some activation function

W2 is the matrix of hidden-to-output-layer weights

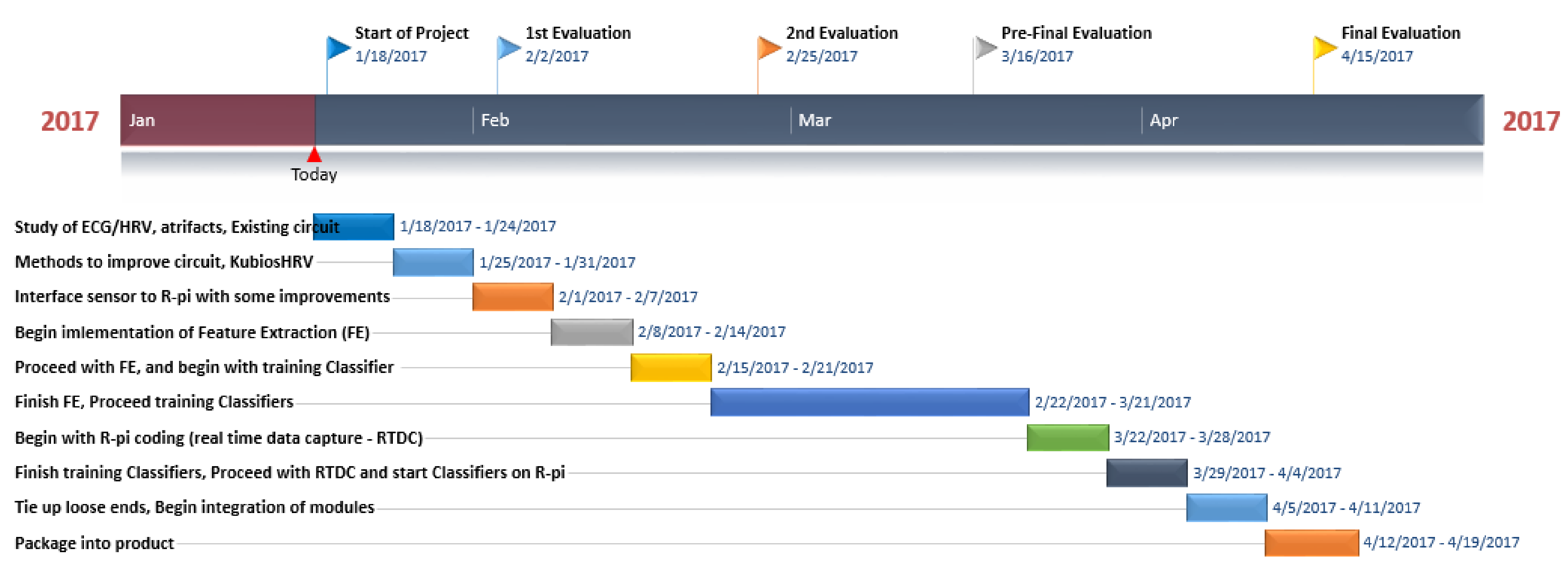
1. Fill W1 with Gaussian random noise;
2. Estimate W2 by least-squares fit to a matrix of response variables Y, computed using the pseudoinverse, given a design matrix X: $\mathbf{W}_2 = \sigma(\mathbf{W}_1\mathbf{X})^{+}\mathbf{Y}$

- Need to explore SVM
- Additional classifers to explore
  - RVM
  - parametric and non-parametric models, System Identification

WORK DONE

- RAKSHITH
  - Time domain feature extraction
  - Non linear feature extraction
- APOORV
  - Raspberry pi setup
  - Frequency domain feature extraction
- AKARSH
  - Circuit design/simulation
  - Introduction to ELM classifier

TIMELINE



- On schedule as per expected timeline

TARGETS FOR THE NEXT REVIEW

- RAKSHITH
  - Consolidate Features to CSV file
  - Explore SVM as a classifier
  - Construct and test circuit on breadboard
- APOORV
  - Port all MATLAB code to R-PI to remove toolbox dependency
  - Construct and test circuit on breadboard
- AKARSH
  - Train ELM classifier for consolidated features file
  - Construct and test circuit on breadboard