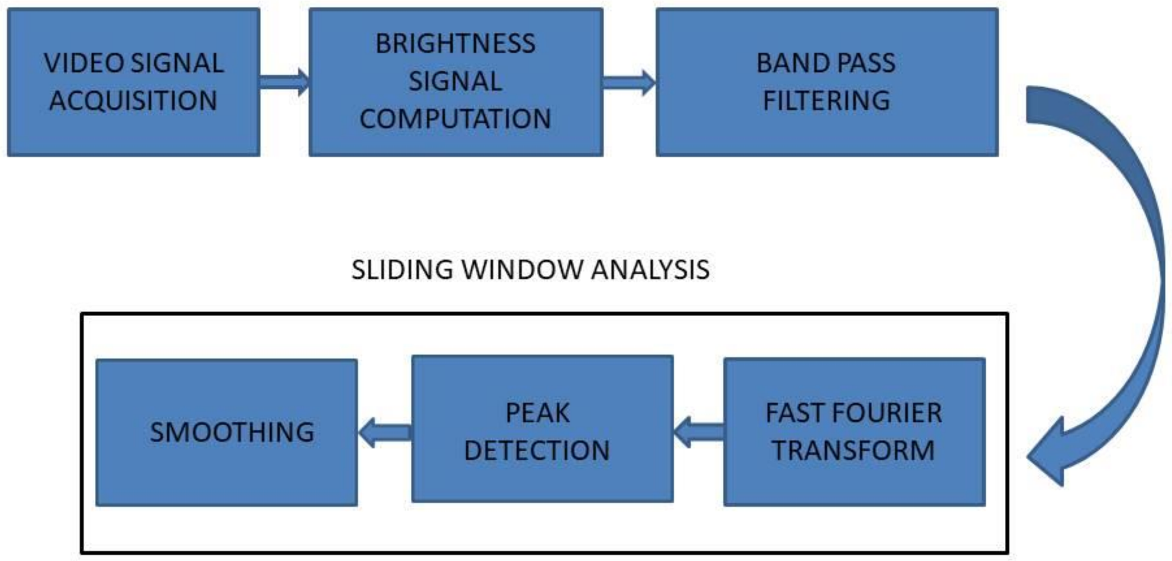
CONTENTS

SOFTWARE AND HARDWARE REQUIREMENT

1. MATLAB 2017 VERSION WITH IMAGE PROCESING TOOLBOX
2. SMARTPHONE CAMREA WITH MINIMUM 5 MEGAPIXEL CAMERA
   WITH VIDEO RECORDING AT 30fps. (We have used Samsung Galaxy A5(2016)).

BLOCK DIAGRAM



VIDEO SIGNAL ACQUISITION

The first thing to take into account when it comes to sampling a signal is bandwidth. The normal Human heartbeat is between 60 and 200 beats per minute (bpm), depending on age, fitness Condition and the physical activity that the subject is doing. In our experiments, we have generously assumed that the target signal can be found between 40 and 230 bpm, i.e. 0.667 and 3.833 Hz. According to Nyquist, the sampling frequency should be at least twice the highest value (i.e 7.667 Hz) to catch the whole range of heartbeat frequencies without aliasing. With the Moto G4 Plus camera, the videos are recorded at 30 frames per second.

RED BRIGHTNESS COMPUTATION

The signal we want to process is the brightness of the skin over time. We cannot ensure that all pixels in the image will contain the brightness variation that we are looking for and we want the rest of the processing pipeline to stay computationally light. Therefore, we chose to combine all pixels into a single average brightness value per frame. On the other hand, as we were thinking about implementing the algorithm in real time, we have skipped the common image brightness computation —combining the red, green and blue planes— in favor of a simple average of all the pixels in the red plane. This is computationally much cheaper and gives very similar results because almost all the image energy is in the red plane. So, the nth sample of the red brightness function can be expressed as :

$$b\,[n] \;=$$

$$\frac{1}{W \cdot H} \mathop{}_{x}^{H} \mathop{}_{y}^{W} v\,[n, x, y, 1]$$

Where:

W : width of the image in pixels

H : height of the image in pixels

v [n, x, y, 1] : light level of the red plane (index 1) at [x, y] coordinates of frame n in the video

BAND PASS FILTERING

After the signal acquisition, a band-pass filter attenuates frequencies outside the interest band. This reduces the noise in later processing steps (peak fine-tuning) and makes the resulting heart rate signal smoother. For our case, a second-order Butterworth filter is designed. The cutoff frequencies have been set to contain our band of interest: 40-230 bpm.

The initial piece of 1 seconds is cut off the filtered signal. It is an approximation of the time it takes for the filter to completely remove the constant signal offset. If this initial piece is not removed, we might get bad readings of the heart beat during the signal stabilization time.

Other filter types can be tested. We chose Butterworth because:

1. It is an IIR filter and the order required for a given bandwidth is much lower than with a FIR filter. Lower order usually means less computations.
2. It has flat pass-band and stop-bands compared to other IIR structures that show ripples. This avoids favoring certain frequencies over others in the valid range.

SLIDING WINDOW ANALYSIS

In order to give a continuous estimation of the heart rate, the FFT and the following two steps (peak detection and smoothing) are repeated every 0.5 seconds. This computation is always performed over a window containing the last 6 seconds of signal samples. Virtually, the window moves over the signal and this is why it is called "sliding" or "moving" window.
The 6-second length is not arbitrary. The window length directly affects frequency resolution and, thus, the accuracy of our estimation. The FFT of a signal sampled N times at a sampling frequency Fs is N bins long. All the bins together cover a bandwidth of Fs. So, the frequency difference between two consecutive bins is Fs / N. This is the frequency resolution (Fr). As the sampling frequency can be written as the number of window samples divided by the total time it took to sample them (the window time duration), we can say that:

$$F_r = \frac{F_s}{N} = \frac{\frac{N}{T_w}}{N} = \frac{1}{T_w}$$

Where:
Fr : frequency resolution
Fs: sampling frequency
N : number of window samples
Tw : window time duration

Therefore, the higher the window duration, the better the frequency resolution. The accuracy will be better, too, as it is half the resolution in this case.

However, increasing the window duration decreases the time accuracy. In the trivial case in which the whole signal length is picked as the window length. If a peak is detected in the FFT, it is impossible to tell when that tone started within the signal or how long it lasted.

Whatever number we give will be a maximum of a window length away from the real value. Another problem of a long window is that it will force the user to wait for an equally long period to get a first reading after starting up the measurements.

We move the window in 0.5-second steps. Computing an estimate every 0.5 seconds does not improve the time accuracy of the output, but it increases the time resolution of the reading. It produces more heart rate output samples per second that will be later smoothed to provide a more continuous and frequent reading. With this, we are incrementing the time resolution of the reading but not its accuracy, which stays limited by the time resolution of the FFT.

FAST FOURIER TRANSFORM

The Discrete Fourier Transform (DFT) is used to translate the signal from the time domain to the frequency domain. The Fast Fourier Transform (FFT) algorithm was used to save processing time when computing the DFT. While the computational complexity of the DFT is O(N2) for a set of N points, the FFT gets the same results with O(N · log2(N)), which means a huge speed-up when N is high.

PEAK DETECTION

Once the FFT is computed for the current sliding window contents, magnitude peaks in the interest band are spotted thanks to the findpeaks function in Matlab. A sample is taken as a peak if it is either larger than its two neighbors or equal to infinity. Among the resulting peaks, the highest peak position is sought with the max function. Finally, it is translated to the corresponding frequency in the FFT vector.

SMOOTHING

At this stage, an approximate location for the most powerful tone in the frequency band has been found, but the possible outcomes are a discrete set in 10 bpm increments because of the frequency resolution produced by the 6-second window. We would like that the heart rate readings look more continuous, with 1 bpm frequency resolution instead. To achieve this, the signal window is correlated with a series of tones in phase and quadrature around the FFT peak in 1 bpm increments. The tones lie in the uncertainty interval around the peak caused by the FFT frequency resolution. The result of each signal-tone correlation is a complex number representing a phase-magnitude pair. The frequency that corresponds to the highest magnitude is taken as the smoothed heart rate.

$$c_k = \sum_{n=0}^{N-1} b_n \, e^{j 2\pi n \, (f_p - 0.5 \, F_r + k \, F_r') / F_s}$$

$$\mathrm{HR} = f_p - 0.5 \, F_r + F_r' \underset{k \in \mathbf{U}}{\arg\max} \, |c_k|$$

Where:

HR: Smoothed heart rate

bk: kth sample of brightness signal

N: Window length in samples

fp: FFT peak frequency

Fr: FFT frequency resolution

Fr': Smoothing frequency resolution

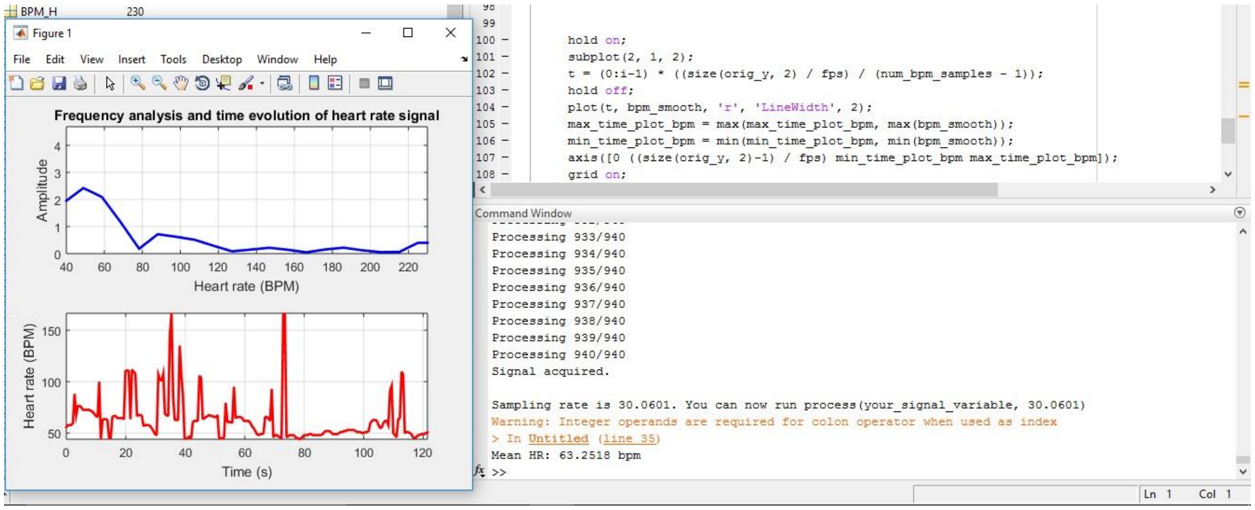
INCREASING SNR

A good Signal-to-Noise Ratio (SNR) is essential to accurately detect the heart rate signal. It can be improved by either reducing the noise or increasing the signal power. In our case, the noise affecting the measurement comes from following sources:

1. Image Noise: By averaging all the pixels for the brightness calculation, we are filtering out a big part of its spatial component. And also by band pass filtering we are removing the unwanted part of signal hence, easing our computation.
2. User Behavior: The pressure variation of the fingertip against the camera lens may show up in video. So it is advisable to use the finger of the hand holding the phone.
3. Lighting Changes: Since we are holding the camera from back, any changes in light may introduce noise in video so,we can overcome this problem by holding the phone gently with keeping the LED flashlight ‘ON’.

RESULTS

We have compared the heart rate of two people in their normal state and other after running a for 10-15 mins and the results achieved are as follows:


BPM_H 230
Figure 1
File Edit View Insert Tools Desktop Window Help
Frequency analysis and time evolution of heart rate signal
Amplitude
Heart rate (BPM)
Heart rate (BPM)
Time (s)
Command Window

```
    hold on;
    subplot(2, 1, 2);
    t = (0:i-1) * ((size(orig_y, 2) / fps) / (num_bpm_samples - 1));
    hold off;
    plot(t, bpm_smooth, 'r', 'LineWidth', 2);
    max_time_plot_bpm = max(max_time_plot_bpm, max(bpm_smooth));
    min_time_plot_bpm = min(min_time_plot_bpm, min(bpm_smooth));
    axis([0 ((size(orig_y, 2)-1) / fps) min_time_plot_bpm max_time_plot_bpm]);
    grid on;
```

```
Processing 933/940
Processing 934/940
Processing 935/940
Processing 936/940
Processing 937/940
Processing 938/940
Processing 939/940
Processing 940/940
Signal acquired.

Sampling rate is 30.0601. You can now run process(your_signal_variable, 30.0601)
Warning: Integer operands are required for colon operator when used as index
> In Untitled (line 35)
Mean HR: 63.2518 bpm
>>
```

Fig: Person ‘A’ in his normal state (Mean HR : 66 bpm)

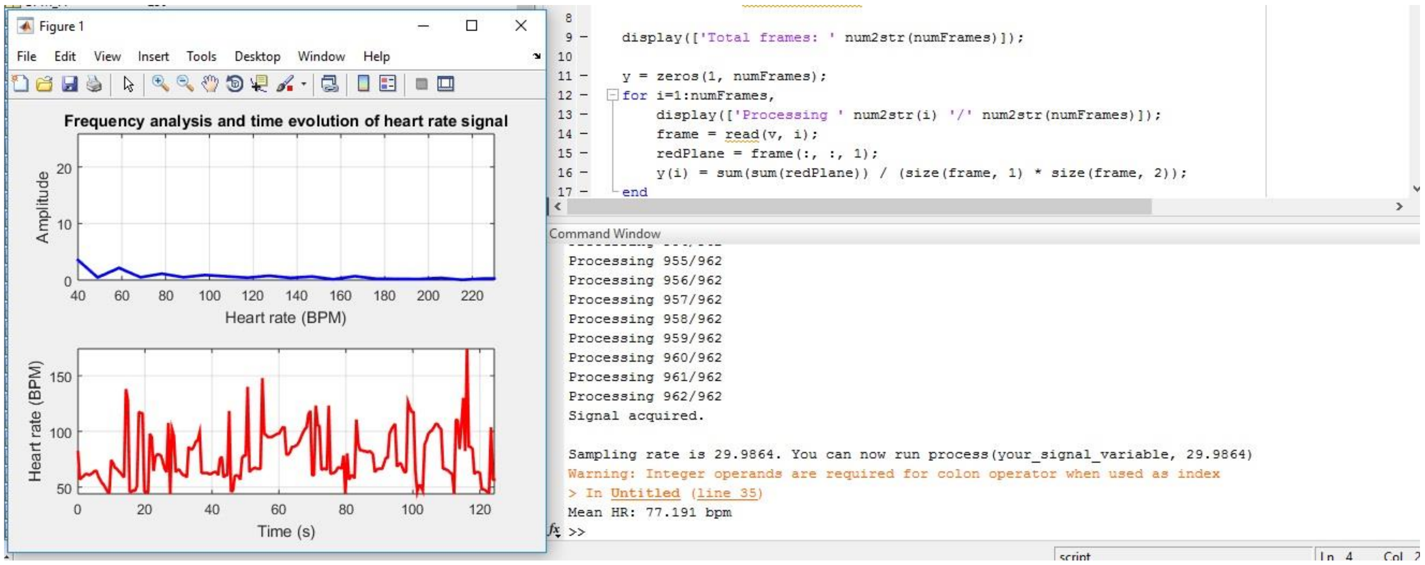
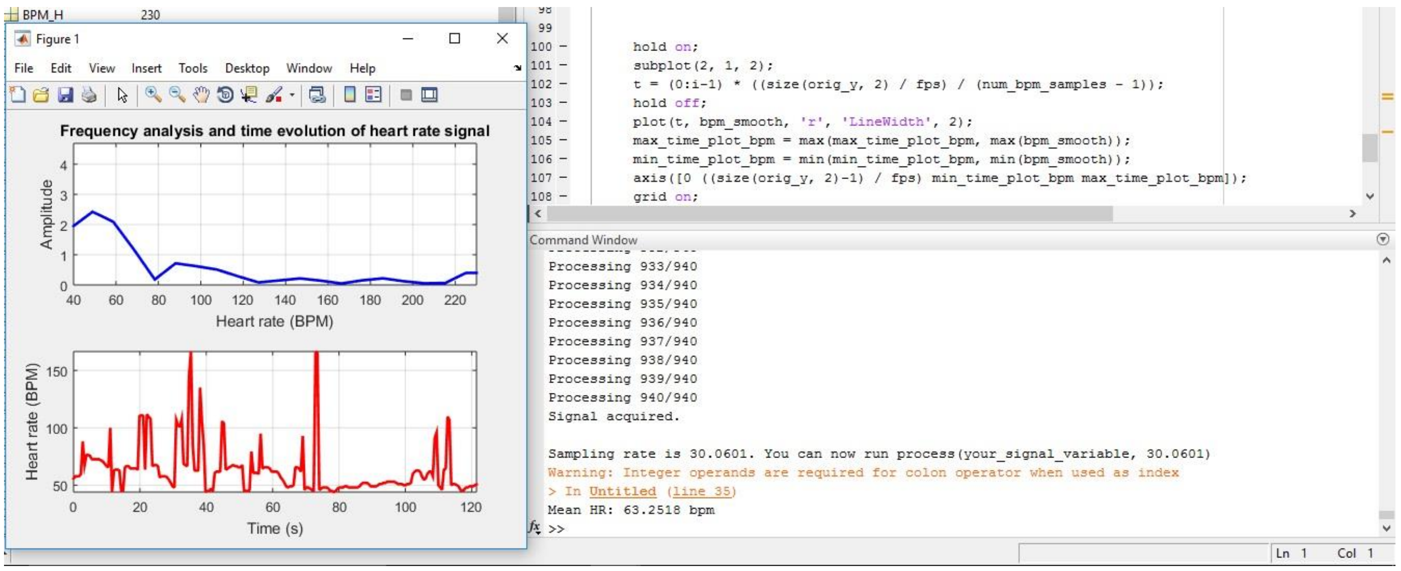


Figure 1
File Edit View Insert Tools Desktop Window Help
Frequency analysis and time evolution of heart rate signal
Amplitude
Heart rate (BPM)
Heart rate (BPM)
Time (s)
Command Window

```

display(['Total frames: ' num2str(numFrames)]);

y = zeros(1, numFrames);
for i=1:numFrames,
    display(['Processing ' num2str(i) '/' num2str(numFrames)]);
    frame = read(v, i);
    redPlane = frame(:, :, 1);
    y(i) = sum(sum(redPlane)) / (size(frame, 1) * size(frame, 2));
end
```

```
Processing 955/962
Processing 956/962
Processing 957/962
Processing 958/962
Processing 959/962
Processing 960/962
Processing 961/962
Processing 962/962
Signal acquired.

Sampling rate is 29.9864. You can now run process(your_signal_variable, 29.9864)
Warning: Integer operands are required for colon operator when used as index
> In Untitled (line 35)
Mean HR: 77.191 bpm
>>
```

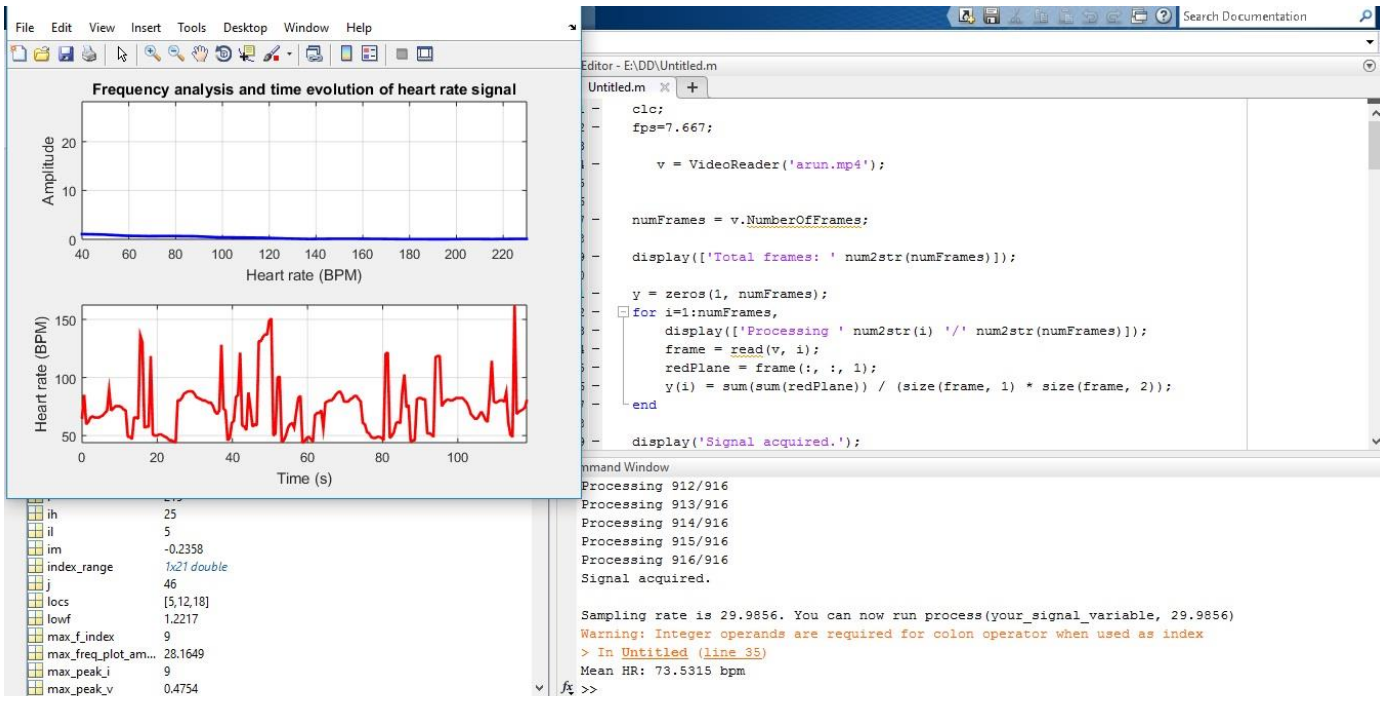
Fig: Person ‘A’ after running (Mean HR : 77 bpm)


BPM_H 230
Figure 1
File Edit View Insert Tools Desktop Window Help
Frequency analysis and time evolution of heart rate signal
Amplitude
Heart rate (BPM)
Heart rate (BPM)
Time (s)
Command Window
Ln 1 Col 1

```
    hold on;
    subplot(2, 1, 2);
    t = (0:i-1) * ((size(orig_y, 2) / fps) / (num_bpm_samples - 1));
    hold off;
    plot(t, bpm_smooth, 'r', 'LineWidth', 2);
    max_time_plot_bpm = max(max_time_plot_bpm, max(bpm_smooth));
    min_time_plot_bpm = min(min_time_plot_bpm, min(bpm_smooth));
    axis([0 ((size(orig_y, 2)-1) / fps) min_time_plot_bpm max_time_plot_bpm]);
    grid on;
```

```
Processing 933/940
Processing 934/940
Processing 935/940
Processing 936/940
Processing 937/940
Processing 938/940
Processing 939/940
Processing 940/940
Signal acquired.

Sampling rate is 30.0601. You can now run process(your_signal_variable, 30.0601)
Warning: Integer operands are required for colon operator when used as index
> In Untitled (line 35)
Mean HR: 63.2518 bpm
>>
```

Fig: Person ‘B’ in his normal state (Mean HR : 63 bpm)


File Edit View Insert Tools Desktop Window Help
Frequency analysis and time evolution of heart rate signal
Amplitude
Heart rate (BPM)
Heart rate (BPM)
Time (s)
Search Documentation
Editor - E:\DD\Untitled.m
Untitled.m
Command Window

```
    clc;
    fps=7.667;

       v = VideoReader('arun.mp4');


    numFrames = v.NumberOfFrames;

    display(['Total frames: ' num2str(numFrames)]);

    y = zeros(1, numFrames);
    for i=1:numFrames,
        display(['Processing ' num2str(i) '/' num2str(numFrames)]);
        frame = read(v, i);
        redPlane = frame(:, :, 1);
        y(i) = sum(sum(redPlane)) / (size(frame, 1) * size(frame, 2));
    end

    display('Signal acquired.');
```

ih	25
il	5
im	-0.2358
index_range	1x21 double
j	46
locs	[5,12,18]
lowf	1.2217
max_f_index	9
max_freq_plot_am...	28.1649
max_peak_i	9
max_peak_v	0.4754

```
Processing 912/916
Processing 913/916
Processing 914/916
Processing 915/916
Processing 916/916
Signal acquired.

Sampling rate is 29.9856. You can now run process(your_signal_variable, 29.9856)
Warning: Integer operands are required for colon operator when used as index
> In Untitled (line 35)
Mean HR: 73.5315 bpm
>>
```

Fig: Person ‘B’ in his normal state (Mean HR : 73 bpm)

FUTURE WORKS

There is scope of improvement in this project in various aspects. Some of them are listed below.

1. Portability: We can add a feature in which we can monitor the data remotely by using Think Speak server so that doctors can monitor their patient’s condition even at midnight when he is not present at hospital.
2. Real time Operation: We can make this project more efficient by saving the measurement time; this can be achieved by making some changes that can enable us to get readings while we are recording video.
3. Mobile Application: We can implement the algorithm in Eclipse to make an Android app and for iOS we can implement it on Swift Platform.