Lab8

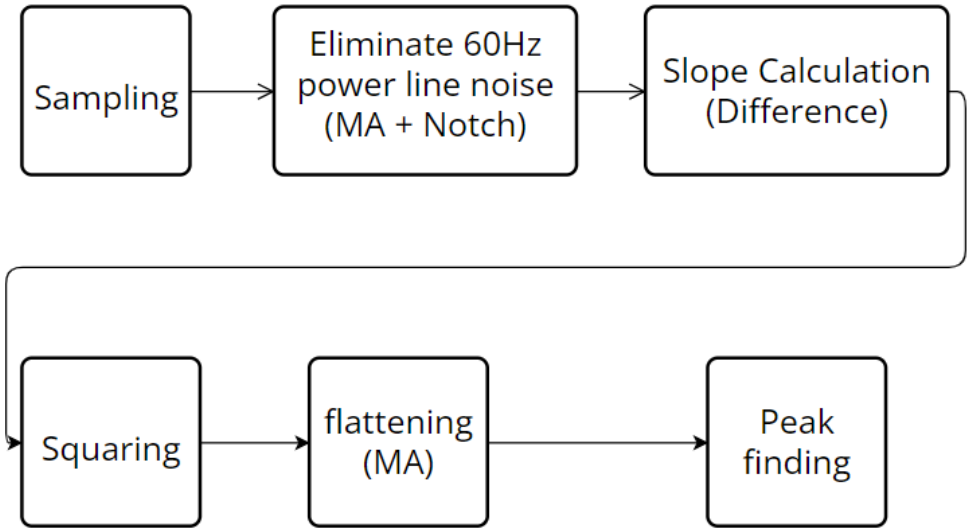
109061217 林峻霆

- ## Lab Objective

在前一個 Lab 中，我們使用 Digital filter 來處理量測 ECG 時會遇到的雜訊(60Hz power line noise 以及 baseline wander)。在這個 Lab，我們繼續進行訊號處理，找出 ECG 訊號中 R-peak 的位置，並時做一個 real-time 的心律測量。

- ## Design Implementation

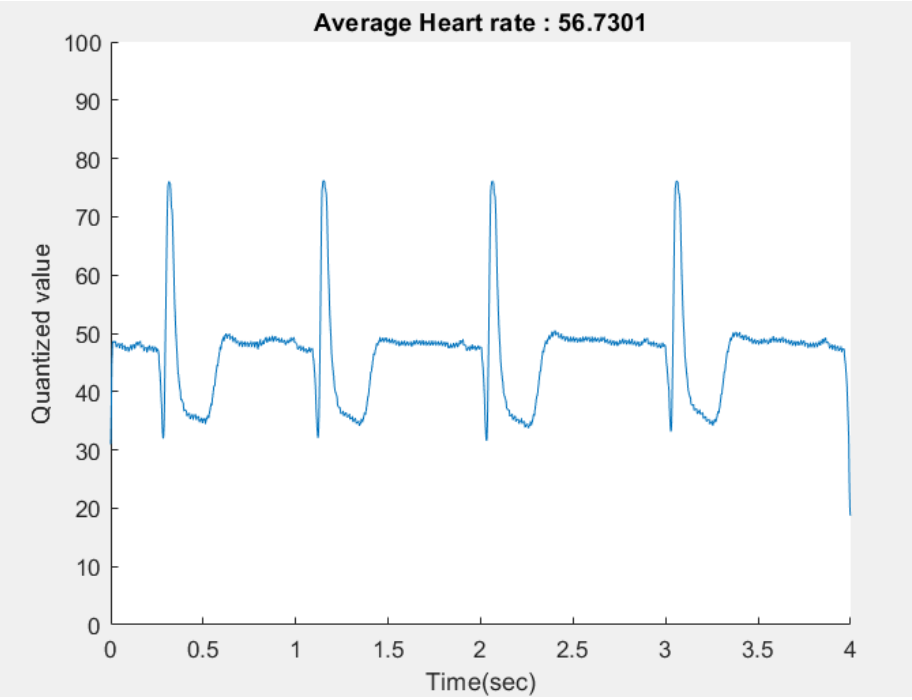
這部分我先解釋我的訊號處理流程為何，並分析各個步驟對 ECG 訊號的影響。我的 ECG 訊號處理的流程如下：



首先是 sampling，sampling 透過 Arduino 內部的 Analogread 來達成，這個 Lab 的 sampling rate(fs)設定為 500Hz，原因在於從前幾個 Lab 我們可以得知 ECG 訊號大部分的 frequency component 都掉在 200Hz 之內，所以 fs 要大於 400Hz，500Hz 相當足夠。

接著我們處理 60Hz power line noise。這部分我的作法沿用上一個 Lab，使用一個 Moving Average Filter 搭配一個 Notch filter。Moving average filter 的 order 設定為 8，使其 zero gain frequency 接近 60Hz。然而 Moving average filter 的再 60Hz 的壓抑量不太夠後面，因此我再接一個 Notch filter 將 60Hz Noise 再度壓抑。再做完這步驟之後，波形如下：

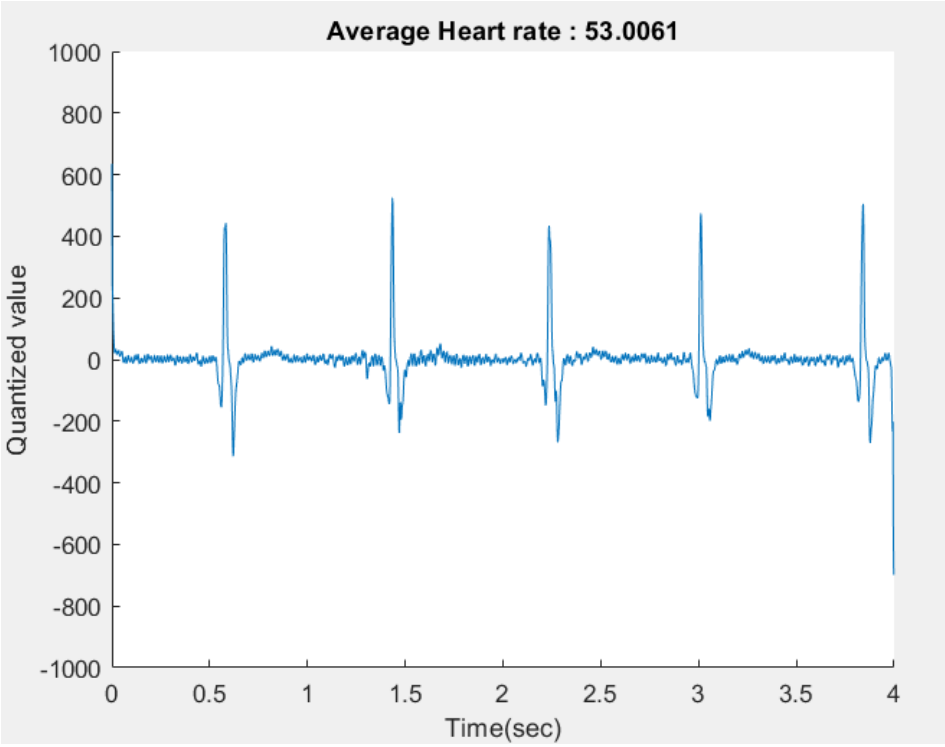
© Ma, Hsi-Pin, Lab for Reliable Computing (LaRC), EE, NTHU.



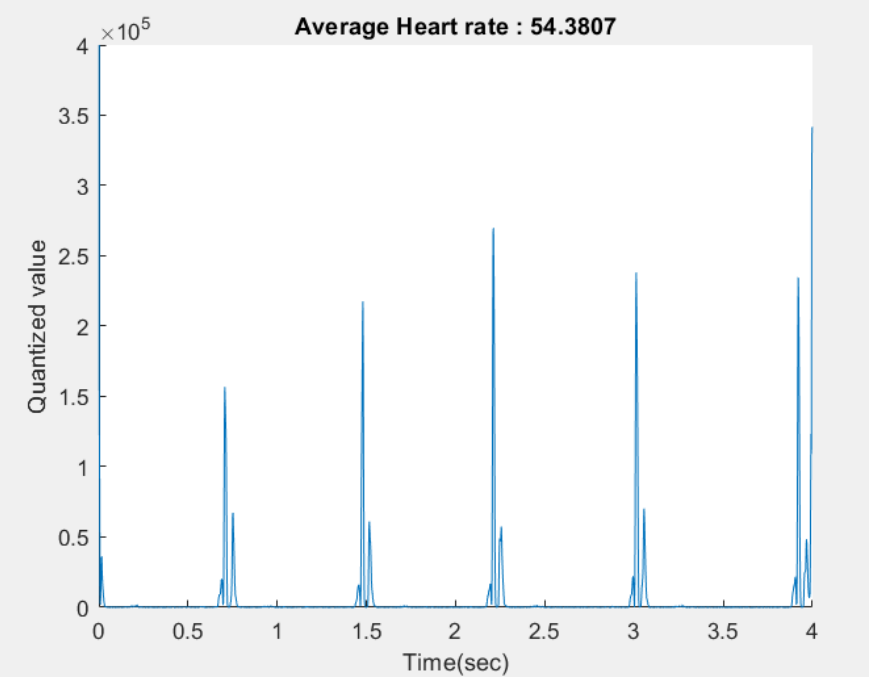
我們要怎麼找 R-peak 呢？一個想法就是去作微分，微分值為 0 的點就是最高點。再數位訊號上，微分就是再做差分(difference)，所以我使用一個 difference filter。Difference filter 的 impulse response 為

$$h[n] = \begin{cases} 2 & when\ n = 0 \\ 1 & when\ n = 1 \\ -1 & when\ n = 2 \\ -2 & when\ n = 3 \\ 0 & otherwise \end{cases}$$

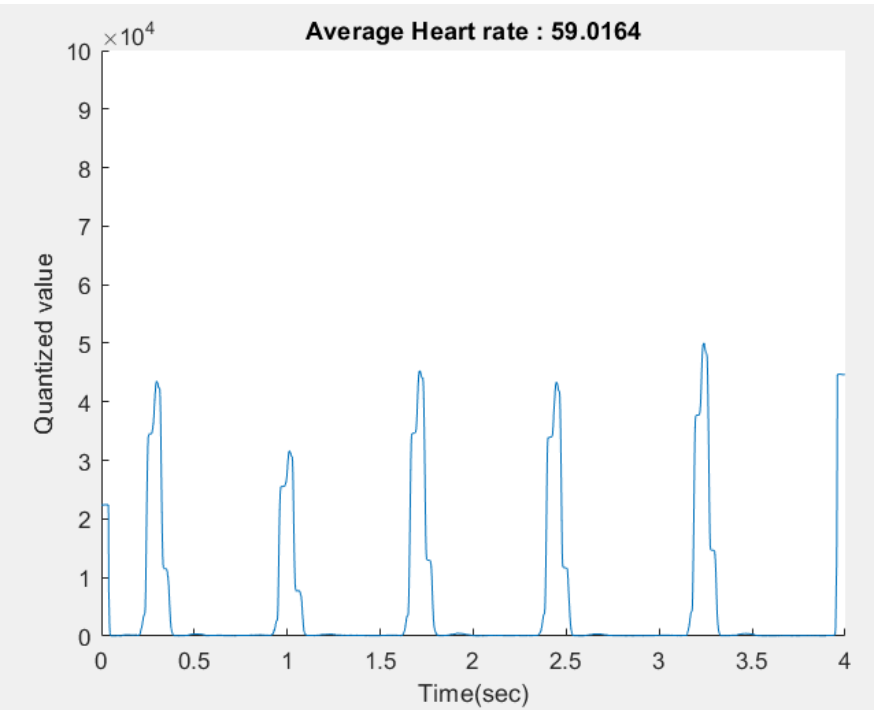
他就是一個 high pass filter，經過 difference filter 後波形圖如下：



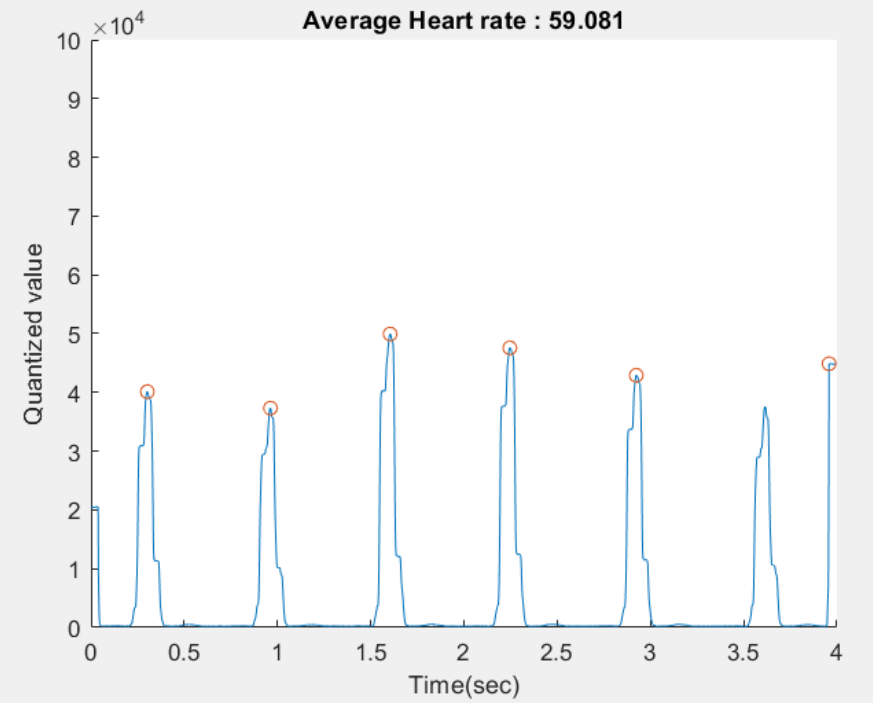
接著再通過一個 squarer，他將整個波的振幅進行平方，結果如下，可以看到有兩個小峰值這兩個峰值的中間點(0 點)就是微分為 0 的點，也就是我們想量測到的 R-peak 點。



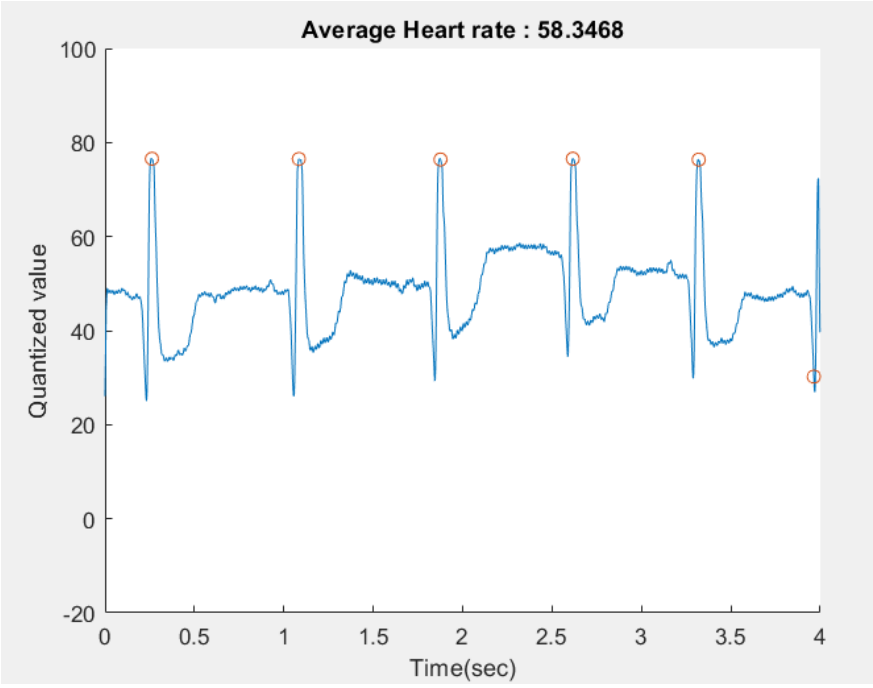
下一步，我們使用一個 order 為 40 的 moving average filter。Moving average filter 的功用在於讓上一張圖中的兩個 peak 中間的 0 點浮上來，這個點就是我們所量測到的 R-peak 的點。之所以 0 點會浮上來的原因在於：以零點為中心做平均的值最大，因為左右兩側有兩個峰值，其結果如下：



接著我們透過 MATLAB 內部的 findpeaks()將峰值找出來，標記如下：



最後，將剛剛用 findpeaks()找出的峰值標記回原先的 ECG 訊號(消除 60Hz power line noise 的)上，這部分要注意兩個地方，第一個是我們在經過 filter 時會有出現 delay，我們可以運用 MATLAB 中的 grpdelay()來找出各個 filter 的 delay，由於我們使用的 filter 的是 FIR filter，而且很幸運是我們這次使用的 filter 都是 linear phase(coefficient 是 symmetric)，所以 group delay 會是一個定值，這會讓我們相當好處理 delay，最後的 index 加一個常數即可。第二個問題出在我們做 convolution 時為了讓結果的 size 不變，並採用 steady state，所以我們往往會使用 conv(a, b, 'same')，這也會導致 index 上的誤差，他會造成 0.5 * filter size 的 index 誤差。綜上所述，為了正確標記 R-peak，我們要在最後畫圖的階段將這些 index 誤差考慮進去。結果如下：



到這邊，我們已經實作一個找出 R-peak 的演算法。接下來，我們來討論 real-time 的部分。要實作一個 real-time 的 heart rate estimation，我們需要注意到程式在執行時每一個步驟的執行時間，這部分可以運用 MATLAB 中的 profiler 來查看，並嘗試去進行優化。

▾ Lines that take the most time

Line Number	Code	Calls	Total Time (s)	% Time	Time Plot
115	drawnow;	1	0.580	87.9%	
94	[pks, locs] = findpeaks(ECG(1:display_length),1,'MinPeakDistance',300);	1	0.054	8.1%	
112	title(title_string);	1	0.020	3.0%	
116	profile viewer	1	0.002	0.3%	
84	ECG = conv(RESULT, df, 'same');	1	0.001	0.2%	
All other lines			0.004	0.6%	
Totals			0.660	100%	

從單一 cycle 的 profiler 結果來看，最多的時間花在畫圖(drawnow)以及 R-peak detection(findpeaks())上，為了更接近 real time 我們可以降低畫面更新率，不用每次取樣都畫圖。我最後採用 25 個取樣點(0.05s)更新一次。程式碼如下：

```
while 1
    tic
    %read data for Arduino
    data = fscanf(s1); % Read from Arduino
    data = str2double(data);

    % Add data to display buffer
    if i <= display_length
        display_buffer(i) = data;
    else
        display_buffer = [display_buffer(2:end) data]; % first in first out
    end


    if mod(i, 25) == 0
        for j = 1 : NSample
            if(isnan(display_buffer(j)))
                display_buffer(j) = 0;
            end
        end

        RESULT = conv(display_buffer, ma, 'same');          % moving average filter
        RESULT = conv(RESULT, FIR_notch, 'same');           % notch filter
        ECG = conv(RESULT, df, 'same');                     % difference filter
        ECG  = ECG .* ECG;                                  % squarer
        ECG = conv(ECG, ma2, 'same');                       % flattening
        RESULT = RESULT / 10;                               % normalize for suitable size for plot


       [pks, locs] = findpeaks(ECG(1:display_length),1,'MinPeakDistance',300);  % detect peak

       % setting title and plot
       len = length(locs);
       if len == 0
           title_string = sprintf('Average Heart rate : %g', 0);
       else
           title_string = sprintf('Average Heart rate : %g', 360*60 * (len - 1) / (locs(len) - locs(1)));
       end

       set(h_plot, 'xdata', t_axis, 'ydata', RESULT);
       set(h_plot2, 'xdata', t_axis(locs + 1 + gd), 'ydata', RESULT(locs + 1 + gd));
       title(title_string);
       drawnow;
   end
   i = i + 1;
   time = time + toc
end
```

從程式碼就可以看出，我們每個 Loop 都會從 Arduino 取資料，但每 25 個 Loop(取樣點)才會去進行 DSP 以及 R-peak detection，這樣就可以減少畫圖和找 R-peak 的次數，進而使資料處理的時間下降許多，讓結果更接近 real-time。

實作結果可以參考附檔影片，另外我用 MATLAB 中的 tic 和 toc 去量執行 100 次的平均時間，由於我所設定的畫面更新率是 25 點一次(取樣 25 次才更新一次畫面，並 detect 一次 R-peak，等同於一秒更新 20 次)，所以我觀察 2500 個 sample point(100 次畫圖)的總執行時間，去算平均每個取樣點要花的處理時間是多少。結果如下：

```
elapsed time of 2500 loops : 6.27707
average elapsed time : 0.00251083
```

理論上的 real time 是資料送進來就要馬上處理馬上將圖畫出來，但這會讓畫面更新率很高，每個 cycle 都要畫圖都要花約 0.6s(>>取樣時間 0.002s)左右(如上面的 profiler 所示)，這會導致我們在畫圖時漏掉很多取樣點，反而本末倒置，更加不是 real-time。透過降低畫面更新率，我們達到平均一個 cycle 約 0.0025s 與取樣的時間 0.002s 接近許多，再加上每秒更新螢幕 20 次對於人眼應該是相當足夠了。

- **Discussion**

這部分我們討論一些量測時遇到的問題以及可能的解決方法：

1. **為什麼有時候影片中找到的 peak 位置怪怪的，尤其是最後一個？**

   這個部分問題出在 MATLAB 中的 findpeaks()中，findpeaks()可以設定選取 peak 的條件，例如說我們可以設定 peak 最小的高度上限('minpeakheight')或者是 peak 之間的最小距離（'minpeakdistance'），以這次的 Lab 為例，我採用的是後者，原因在於若採用高度上限則容易發生找到多個 peak。我取 300 個 sample point(約 0.6s)代表任兩個 peak 之間至少相距 0.6s，這導致常常在最後就算沒有 peak 這個 function 也會取一個點來當 peak。一個解決方法就是同時設兩個限制(最小高度+最小距離)，但這樣會發生的問題就是晃將 average cycle time 拉長許多(如下圖)，會導致這個系統不太 real-time。

   ```
   elapsed time of 2500 loops : 11.2282
   average elapsed time : 0.00449126
   ```

2. **有哪些條件會影響到系統接不接近 real-time 呢？**

   除了畫面更新率之外，下面再舉幾個有影響的因素：

   a. **Filter 的 size**

      Filter 的長度、transfer function 的係數以及是 FIR or IIR 等都會有影響。其主要原因在於 Filter 長度越長，就有越長的 transient state，影響到量測結果，係數以及 FIR or IIR 會影響到 phase 進而造成 group delay。另外，filter 越長我們在做 convolution 時要花的計算時間越長，也會影響到 real-time。

   b. **Sampling rate(fs)**

      Sampling rate 太慢，導致我們量到的訊號不是即時的心電訊號。

   c. **Buffer**

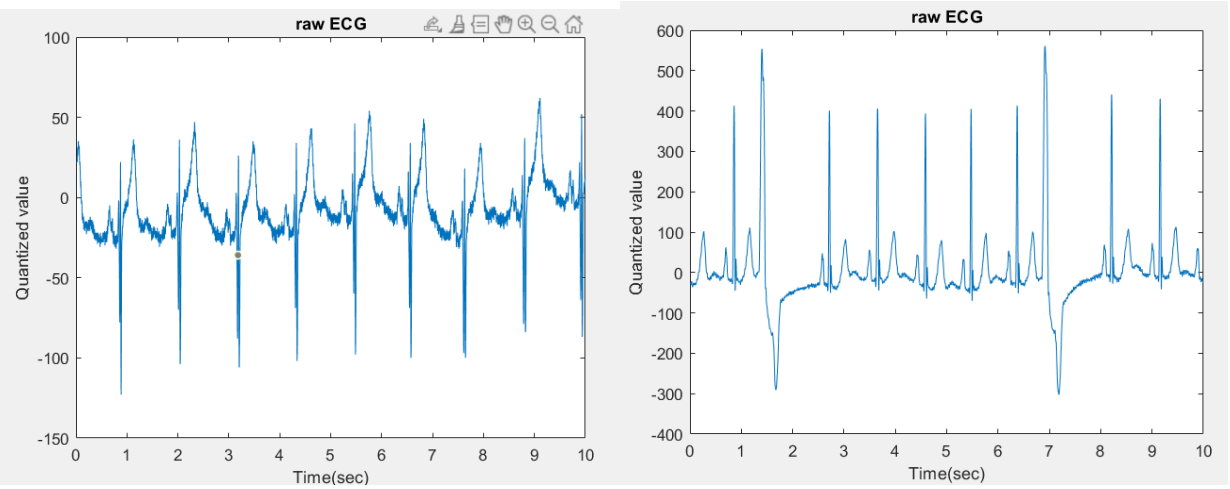
由於我們有用一個 buffer 來儲存從 Arduino 讀取的資料。然而 buffer 的 size 並不是無限大(這個 Lab 設定為 2000)，因此在 buffer 滿的時候要如何處理也會是一個重要的課題。以我的作法為例，當 buffer 滿的時候，buffer 的第一個資料丟掉將剛拿到的資料放在最後(FIFO)，由於我選的 buffer 是用一個 array 實作，所以我必須要做 1999 次左移；如果我的 buffer 是採用 linked list 那我就只需要將後面接上前面切斷即可，這樣就可以讓執行時間更短。

3. MIT-BIH database 的實作成果

MIT-BIH database 中紀錄了各種不同年齡以及性別的人的心電訊號。我們嘗試透過上課所學的訊號處理流程去找他 R-peak 的位置，並與 ground truth 進行比對。我做訊號處理的流程與前面相同，但其中有些參數有進行修改(例如 sampling rate 變成 360)。接著我先解釋我是怎麼將我的結果去與 ground truth 作比對的，將 ground truth 與未處理的 ECG 訊號做相比可以發現，兩者 R-peak 的標誌時間有時候會不太一樣，但相差不大。我認為這個原因來自於取樣的誤差，因為我們的 sampling rate 是 360，所以我們的 accuracy 就是 1 / 360 = 0.0028 秒，R-peak 就會出先在這 0.0028 秒之間。因此，我在將我的 prediction 與 ground truth 做比對時，只要 prediction 跟 ground truth 之間的差異 < 0.0028 秒，我就當作他是正確的 detection。下面為我有嘗試的幾組 testcase 以及進行 R-peak detection 的結果。

Easy			
	TP	TN	FN
100m	2259	14	14
103m	2084	1	1
112m	2532	7	7
117m	3	2978	2611
122m	2473	3	3
Mid			
	TP	TN	FN
107m	549	1590	1589
205m	229	2408	2406
219m	298	2039	2037
Hard			
108m	0	2046	1991
210m	105	2476	2458
230m	449	1868	1849

下面簡單分析一下 performance 不好的 case 的原因：可以發現在 Medium 和 Hard 中的 performance 都不太理想，於是我就去看了它們的波形，我發現 Mid 和 Hard 的 testcase 中往往有例如心律不整、R-peak 不明顯的現象，如下圖所示。



可以看到左邊這個 ECG 訊號他的 R-peak 的 amplitude 甚至比旁邊 interval 的 amplitude 還要小，這會導致我們的 difference filter 後出現 0 點的位置發生誤差，在做 flattening 時也會需要更高 order 的 moving average filter 才能將 0 點浮上來。右邊這張圖則可以看到他的心跳頻率和震幅不太正常，他的 ECG 訊號是 1 個較大的訊號搭配 5 個較小的訊號，這會導致我們在用 findpeaks()時容易標記到錯誤的點，進而產生誤差。

要解決以上的問題，我目前有兩種想法：首先，我們可以使用 iterative 的方式去找 R-peak，也就是多做幾次 peak detection 的流程，逐步縮小 R-peak 的 interval。然而這樣的缺點就在於不太適用於一個 real-time 的 system。另一個方法是修正我們所使用的 findpeaks()，多加一些限制條件讓 peak 標記的位置更準確。

- ## Conclusion

在這個 Lab 中，我們實作一個 real-time 的 heart rate estimation 系統。過程中，我們討論各種 filter 對於 peak finding 的幫助，透過 moving average filter、difference filter 以及 squaring，並搭配 MATLAB 的 findpeaks()可以讓我們找到 ECG 訊號中 peak 的位置。另外，在設計的過程中我們也要注意到 group delay、硬體、MATLAB function 的執行時間等來確保這確實是一個 real-time 的系統。

MIT-BIH database 則是測試我們所使用 DSP 流程的泛用性。實務上量測到的 ECG 訊號會根據被量測者得年紀、身體狀況而有很大的差異，以我目前的實力設計出的訊號處理流程在遇到一些特殊的 case 時(R-peak 不明顯、心律不整)，detection 的成果就不甚理想。希望未來在學習更多 DSP 流程以及更認識 ECG 訊號後能在重新挑戰這個問題。

在這四週的 Lab 中，我們一步一步實作一個 real-time heartrate estimation 的 system。從認識取樣、ECG 訊號開始，到接下來實作 filter 以及 real time system，過程中我們探討 filter 的設計分析以及一個 real-time 系統的注意事項，並嘗試設計數位訊號處理的流程來找出 R-peak 的位置。

- ## References

© Ma, Hsi-Pin, Lab for Reliable Computing (LaRC), EE, NTHU.

- 教授與助教的講義
- 李祈均教授的DSP講義
- 李夢麟教授的訊號與系統講義
- MATLAB : findpeaks(), grpdelay()