ECG Monitor Developing Report: Digital Part

Deng Luo Stu. No:11210004 Email: loud@mail.sustc.edu.cn

Introduction

We have finished the analog part of the ECG monitor before. But the result cannot be showed to user on a screen and we did not get the basic characteristics such as heart rate as well. And we may need some digital filter to process the original signal. So we need to develop another part of the monitor: the digital part. We chose to use MSP430G2553 and necessary accessory such as serial cable and Nokia 5110 LCD to finish this part, showing the ECG wave as well as the heart rate on screen.

Objective

Use the PCB board we soldered in Lab5&6 and MSP430 launchpad to finish the followings:

1. Acquire the ECG with the sample rate of about 500Hz.
2. Upload the data to computer (at least data of 4 continuous periods).
3. Realize the process of the data: filtering, calculating the heart and displaying the signal.
   a) Display the ECG wave and heart rate on computer
   b) Display the heart rate on LCD
4. Allow one LED to flicker with the heart beating.

Methods

1. Acquire the ECG with the sample rate of about 500Hz.

We use P1.7 as input pin. And because the MSP430 cannot read the signal under 0V, so we need to connect the -1.5V of the ECG PCB board to GND of MSP430. So the wiring is:

ECG output ------ P1.7

-1.5V ------ GND

We choose ADC10 as ADC converter. The reference we use is 2.5V to avoid saturation. Here is the relevant code and its comments.

```
ADC10CTL1 = INCH_7 + ADC10DIV_0;    // choose P1.7 as input
/*
  * set the basic parameters for ADC10
  * INCH_10: Input channel select, Selects Channel 10
  * ADC10DIV_0: ADC10 Clock Divider Select 0
  * We do not set the clock apparently for ADC10. Actually, the default value 0 means to use internal clock ADC10OSC. So we have set it. Check the frequency if needed.
  */
ADC10CTL0 = SREF_1 + ADC10SHT_3 + REFON + ADC10ON + REF2_5V;
/*
  * Set the parameters for the ADC sampling process
  * select REF: VR+ = VREF+ and VR- = AVSS
  * ADC10SHT_3: ADC10 Sample and Hold Time, 64 x ADC10CLKs
  * REFON: ADC10 Reference on，  we set it above and open it now. The open process takes 30us, so we need to delay 5 cycles.
  * ADC10ON: ADC10 ON
  */
_delay_cycles(5);                       // Wait for ADC Ref to settle
ADC10CTL0 |= ENC + ADC10SC;             // Sampling and conversion start
_delay_cycles(100);
ADC10CTL0 &= ~ENC;//Stop converting
ADC10CTL0 &= ~(REFON + ADC10ON);//close REF and ADC10
heartRaw = ADC10MEM;//Get value from ADC10MEM
```

To set the sampling rate, we use TimerA to control the awakening of ADC10 by its interuption. We use 1M digital clock as

MCLK and SMCLK, set TImerA use SMCLK, set TACCR0 equal to 2000+8 and use the up mode to trigger the interruption. If we set TACCR0 = 2000, then the sampling rate should be 1M/2000 = 500Hz. However when we use a sine wave to check the sampling rate there is a little bias. It may result from the Production Process, temperature, or the wires. So we add 8 to calibrate it. The calibrating figures are shown in results part. And the following is the code or this part.

```
    BCSCTL1 = CALBC1_1MHZ;                  // Set range
    DCOCTL = CALDCO_1MHZ;                   // Set DCO step +
modulation
    //When we do not set BCSCTL2 apparently, the default value is
0, which will set MCLK and SMCLK as DCO, and the divider is 1.
    BCSCTL2 = DIVM_0;
```

```
    TACCTL0|=CCIE;          //Enable the interruption of TimerA,
When time is up, it will set a flag.
    TACCR0=2000+8;              // CCR0 value for TimerA, which is the
value for the first timer.
    TACTL|=TASSEL_2+MC_1;      //set timerA's clock signal as SMCLK,
And set the mode as up mode. So we need to set SMCLK.
```

2. Upload the data to computer (at least data of 4 continuous periods).

We use serial transmission to upload data to computer and use Matlab to read the serial data. We use Baud rate of 9600 to send and read data, use SMCLK as the serial clock. Here is the code of this part.

CCS code:

```
void SetUART(void){
    //UART register
    P1SEL = BIT1 + BIT2 ;                       // P1.1 = RXD, P1.2=TXD
    P1SEL2 = BIT1 + BIT2 ;                      // P1.1 = RXD, P1.2=TXD
    UCA0CTL1 |= UCSSEL_2;                        // choose SMCLK as the
clock for serial port communication
    UCA0BR0 = 104;                              // 1MHz 9600 UCA0 Baud
Rate
    UCA0BR1 = 0;                                // 1MHz 9600 UCA0 Baud
Rate
    UCA0MCTL = UCBRS0;                           // Modulation UCBRSx
= 1
    UCA0CTL1 &= ~UCSWRST;                        // **Initialize USCI
state machine**
}


    //UART send
    while (!(IFG2&UCA0TXIFG));                  // Check whether
USCI_A0 TX buffer is ready
         UCA0TXBUF=heartC;
```

```
Matlab code:
clear;clc;
delete(instrfindall);
s= serial('COM7'); // Change the COM number if needed
s.DataBits=8;
s.BaudRate=9600;
s.Parity='none';
s.StopBits=1;
s.InputBufferSize=255;
fopen(s);

temp = fread(s,1);

fclose(s);
delete(s);
clear s;
```

3. Realize the process of the data: filtering, calculating the heart and displaying the signal.

a) Display the ECG wave and heart rate on computer

We read 100 data points and then plot them. We use Moving Average Algorithm with window size of 20 points and 50Hz notching filter to process the data. And we use a very clever way to display the data, splice the newest data with the old data making it look like a real ECG. As to calculating heart rate, we detect the highest the number in every 250 points and then set the threshold of "130" for a putative R wave of a ECG to count the heart beating, and then divide them by the time spent. Actually we tried to use the Pattern Recognition for this part. But the result is not good as mentioned in results and discussion part. Here is the code and comments for this part.

Matlab code (Only important code is mentioned here, please see the appendix for all)

```
   notched_y = filter(ones(1,windowSize)/windowSize,1,y);% Moving
Average
   notched_y = filter(Notching_50Hz, notched_y);

Fs = 1000;  % Sampling Frequency
N   = 10;  % Order
Fc1 = 48;  % First Cutoff Frequency
Fc2 = 52;  % Second Cutoff Frequency
% Construct an FDESIGN object and call its BUTTER method.
h  = fdesign.bandstop('N,F3dB1,F3dB2', N, Fc1, Fc2, Fs);
Hd = design(h, 'butter');


%Keep the signal of the last period at the tail, refresh the signal at
the head of the figure. spacing: 500points
      plot(( length(y) + 500 ):1:5000,old_y(( length(y) +
500 ):5000),'b',1:1:length(y),y,'b');

      % heart rate calculation and convert it to char
      heart_to_now = [notched_old_y notched_y(1:
floor(length(notched_y)/500)*500 )];
      k = 1;
      while ( k <= length(heart_to_now)-check_data_length + 1 )
         heart_max = [ heart_max
max(heart_to_now(k:k+check_data_length-1)) ];
         k = k+check_data_length;
      end
      heart_threshold = heart_max > 130;
      beat_no = sum(heart_threshold);
      heart_rate = 60*500*( beat_no - 1 )/( length(heart_to_now) );
```

b) Display the heart rate on LCD

This part is very difficult. Although we find an open source code for display on Nokia 5110 LCD by MSPG2553, we find it only recognize "char *" data type to display. However, our data is in int or long type. And we find that CCS do NOT support some of the Built-in functions of C such as sprint and itoa that can convert int to string. So we write a method to display int data based on the original open source code. Since this code works well, we did not dive into other details of displaying.

Wire connection

```
pin   description
1     VCC(3.3V)
2     GND
3     CE          //Pin2.3
4     RST         //Pin2.4
5     DC          //Pin2.2
6     DIN         //Pin2.1
```

```
7     CLK         //Pin2.0
8     Vlcd(+5V)
```

```
void Lcd_Write_Int(unsigned char X,unsigned char Y,int value)
{
    Lcd_Set_XY(X,Y);          //set Addr
// int a0 = value / 100;
// value = value % 100;
    int a1 = value / 10;
    int a2 = value % 10;
// a0 = a0 + '0';
    a1 = a1 + '0';
    a2 = a2 + '0';
// Lcd_WriteChar(a0);
    Lcd_WriteChar(a1);
    Lcd_WriteChar(a2);
}
```

For the heart rate calculating on MSP430, it is also not so easy. But we find a cunning method as following code shows. We detect the ADC mem data and see whether it is larger than the threshold and we use a “inpeak” to judge whether it should be count into the “heartCount” or not. And we use the nearest 10 seconds to calculate the heart rate.

```
  const int beatThreshold = 700;
  counter = 1;
  inpeak = 0;
  beatCounter = 0;
    //calculate heart rate,use whether it is inpeakto count the beat
number
    if(heartC > beatThreshold && inpeak == 0){
       beatCounter++;
       inpeak = 1;
       if(beatCounter > 5){
          heartRate = 60*500*(beatCounter-1)/counter;
          Lcd_Write_Int(0,4,heartRate);
       }
       else{
          Lcd_Write_String(0,4,"---");
       }
    }
    //retreat from inpeak
    else if(heartC < beatThreshold){
       inpeak = 0;
    }
    //Zero all parameters after 10 seconds for re-calculating
    if (counter == 5000){
       counter = 0;
       inpeak = 0;
       beatCounter = 0;
    }
    counter++;
```

4. Allow one LED to flicker with the heart beating.

This part is quite easy if you want to make it simple. We just use Peak Detection to decide whether the RED LED should be lightening or not.

```
    //Light the red LED when R weave reaches.
    if (heartC > beatThreshold){
       P1OUT = RED_LED;                    // red LED on
    }
    else if(heartC < beatThreshold){
        P1OUT = 0;                        // RED LED off
    }
```

Results

We made two videos to show our final results. One is “ECG_wave.avi” showing the Original ECG wave and the processed ECG wave and the heart rate. Another is “ECG_whole.avi” showing the whole experiment process, including the LED flickering and the LCD displaying. Please check those files.

Besides that, here we show other two things. One is sampling rate calibrating. We use 0.5Hz sine wave to calibrate it. **Figure 1** is the signal when we use 2000 for TACCR0. From the signal in red rectangle, it can be told that the sampling rate is not exactly equal to 500Hz. So we changed value of TACCR0. And we find 2008 is the best one as showing in **Figure 2**.

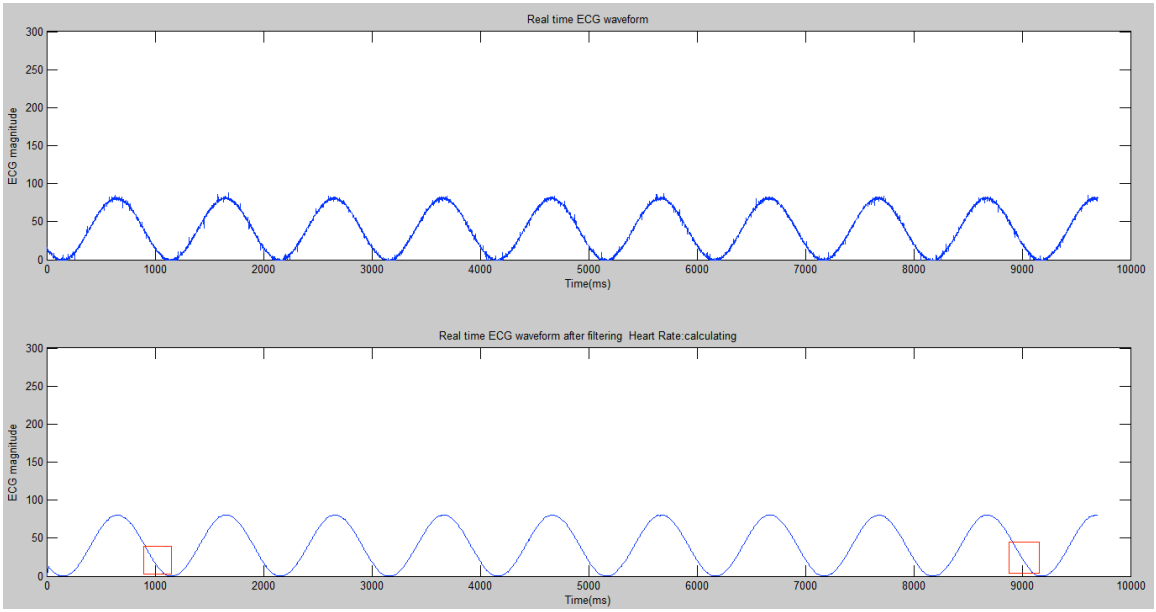

Figure 1 Sampling when TACCR0 = 2000

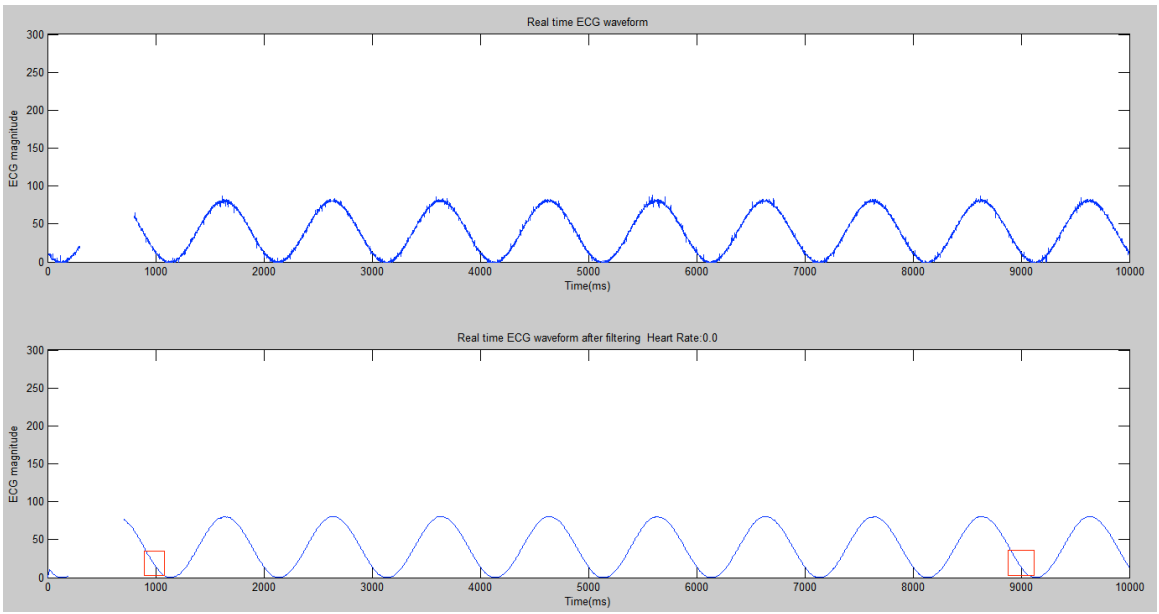

Figure 2 Sampling when TACCR0 = 2008

Another things need to be mentioned is that we actually tried to use a certain part of a ECG signal to do the Pattern Recognition when calculating the heart rate. We choose the R wave as the seed as shown in **Figure 3**. It contains 100 data points. We use the correlation coefficient between the seed and the real time data in a 100 points window. However the results is very bad. What’s more, It takes a lot of time for Matlab to do the computation and therefore resulting into the signal delay when displaying the ECG. So we gave it up and take the simple method to finish the heart rate calculating.

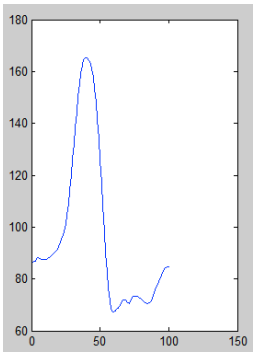

Figure 3 The R wave we tried to use as seed to do Pattern Recognition.

Discussion

It's a very complicated project and we learned a lot such as ADC converter, Matlab read serial data, digital filtering, making a running figure, and displaying technique for Nokia 5110 LCD, ect. It's not easy that we finished it. And in these days, we have some profound experience for this and other projects. Firstly, one should learn the most related knowledge throughly. So that you can understand what's going on and how to handle it when there is an error and need debugging. Secondly, if a project contains many steps, we should finish each step as excellent as possible. Because in this way, the following steps will be finished quickly, and you do not need to check the previous steps frequently. Last but not least, sometimes, the simplest way is the best way. Simple means stable and quickly. So we may need to find the simple ways to solve the problems rather than the complicated ways that you think it may produce good resuilts.