***Monitoramento de frequência cardíaca com msp430 (Oxímetro)***

|  |  |  |  |
| --- | --- | --- | --- |
|  | Gilvan Júnior Pereira Camargo  Universidade de Brasília - UnB Brasilia-DF, Brasil gilvan.jpc@gmail.com |  | Maria Carolina de Almeida da Silva  Universidade de Brasília - UnB Brasilia-DF, Brasil carolinallima20@gmail.com |

***Resumo*— Obtenção da frequência cardíaca a partir do processamento com o MSP430G2553 do sinal cardíaco obtido através da utilização de um oximetro, para monitoramento remoto em um aplicativo com transmissão bluetooth.**

***Keywords— Frequência cardíaca, Oxímetro, MSP430.***

1. INTRODUÇÃO

Faz-se necessária a medição dos sinais cardíacos para se obter informações cruciais para o mapeamento e conhecimento de aspectos de suma importância para a verificação do estado de saúde (pressão cardíaca, picos de BPM, e outros). Posteriormente, passou-se a utilizar esta medição também para monitoramento de certos aspectos durante a prática de atividades físicas (como exemplo a máxima frequência atingida, que mostra o nível de condicionamento físico).

Tendo em vista a necessidade de se conhecer a frequência cardíaca devido a estes aspectos, mesmo em condições não ideais a medição dos sinais cardíacos veio sendo realizada até metade da década de oitenta por um procedimento complicado de ser realizado, e que demandava de uma técnica invasiva e não contínua, além de que era realizada por um equipamento muito grande e pesado.

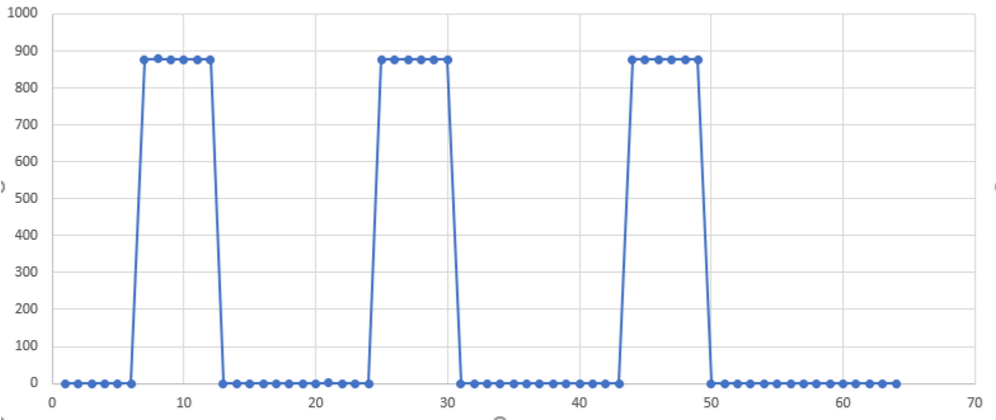
O monitoramento da frequência cardíaca se dá a partir da obtenção do sinal cardíaco através de um oxímetro, que é capaz de quantizar os pulsos cardíacos, e a partir do tratamento dessa entrada por um circuito eletrônico e tratando o sinal utilizando funções no MSP430, obtém-se o sinal discreto dos pulsos do coração, que nos dá a quantidade de batimentos cardíacos por minuto.

O problema chave na obtenção dos valores a partir do sinal cardíaco é, que além de o sinal possuir uma variação muito pequena, possui bastante ruído. Além disso, é irregular e possui picos aleatórios de ruído, que poderiam ser computados como pulso, mesmo não o sendo. Fatores estes, que dificultam a captação correta deste sinal.

1. DESENVOLVIMENTO

Em relação ao hardware, a solução para o tratamento do sinal cardíaco se deu pela amplificação e tratamento deste para se obter um sinal que pudesse ser lido, condicionado e analisado pelo MSP430G2553.

No MSP430 foi feita uma conversão analógico-digital para condicionamento do sinal para análise, além disso foi utilizado o UART para transmissão serial dos dados. Foi utilizado ainda, o Timer A programado para ler tanto bordas de subida quanto de descida (bits CMx setados em 11), espaçadas com certo intervalo para que não houvesse confusão do sinal do pulso com possíveis ruídos/glitches remanescentes. Isto gerou um sinal capaz de ser analisado e de transmitir as informações para cálculo da quantidade de batimentos cardíacos por minuto.



*Figura 1 - Amostras Condicionadas para análise.*

1. DESCRIÇÃO DO HARDWARE

Para a obtenção do sinal cardíaco utilizou-se um oxímetro, que é constituído de um led vermelho e um sensor infravermelho (fototransistor com configuração emissor comum). A partir deste, um sinal é gerado pelos pulsos que são obtidos pela resposta da luz do led no infravermelho, quando esta atravessa as células do sangue.

Seu funcionamento se dá basicamente pela variação da intensidade da luz vermelha, que é emitida pelo led posicionado paralelo ao sensor na extremidade vertical oposta do dedo, na qual, esta intensidade varia de acordo com a divergência na cor da hemoglobina (célula constituinte do sangue que transporta o oxigênio), que pode estar menos vermelha (mais oxigenada) ou mais vermelha (menos oxigenada), o que fará com que mais luz ou menos luz, respectivamente, chegue ao sensor infravermelho. Desse modo, obtêm-se o pulso e, consequentemente, o sinal cardíaco, uma vez que, o sangue arterial (rico em oxigênio) é o sangue que sai direto do coração para o corpo, e assim replica seu pulso de bombeamento pela corrente sanguínea. Pulso tal, que pode ser obtido em artérias (veia que sai direto do coração com sangue arterial – rico em oxigênio e, consequentemente menos vermelho).

O sinal cardíaco na saída do oxímetro apresenta variação de aproximadamente 100mV na faixa de tensão entre 3,8-4 V. Para deixar o sinal amostrado com sua variação a partir do zero, utiliza-se um capacitor para desacoplar o sinal e assim obter um sinal com sua variação indo de 0 a100mV. Então, este sinal é amplificado por um amplificador operacional com montagem para gerar um ganho de 10 no sinal.

A partir do sinal amplificado é feita uma comparação (utilizando um circuito comparador com amplificadores operacionais) da tensão do sinal desacoplado e amplificado com a tensão na saída do potenciômetro (faz-se um divisor de tensão para que a sua saída seja de 1,27 V). De modo que, se a tensão do sinal for maior do que a tensão no potenciômetro, a saída do comparador vai para V+ (que corresponde a alimentação do amplificador operacional, neste caso é igual a 3,6V) e, se menor, a saída do comparador vai para V- (0V). Isto faz com que o sinal na saída final do circuito eletrônico esteja dentro da faixa necessária para que o sinal seja recebido com êxito pelo MSP430G2553.

* Materiais Utilizados

- Resistores de 47 KΩ, 10 KΩ, 220 KΩ, 100 KΩ.

- Potenciômetro de 1 KΩ.

- Capacitor de 1µF.

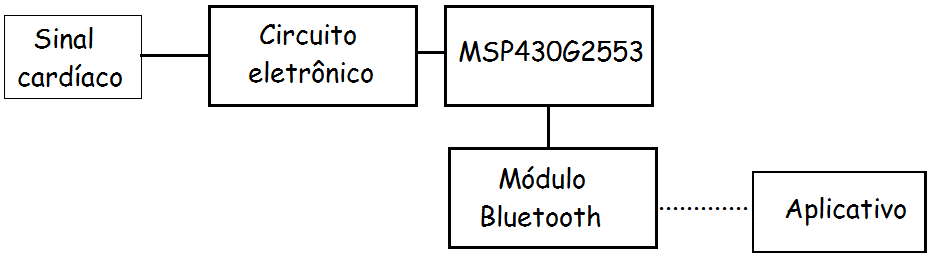
- Circuito Integrado Amplificador Operacional LM358.

- Módulo Bluetooth HC-05.

- Microcontrolador MSP430G2553.

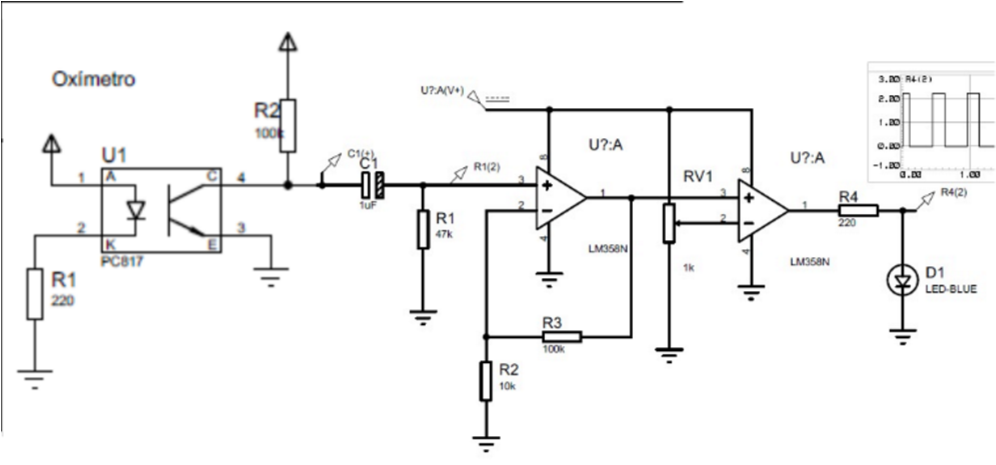
- Oxímetro.

* Diagrama de blocos



*Figura 2 – Diagrama de blocos do hardware para monitoramento do sinal cardíaco.*

* Esquemático



*Figura 3 - Esquemático do circuito para conversão dos pulso cardíaco em sinal capaz de ser analisado pelo MSP430G2553.*

1. DESCRIÇÃO DO SOFTWARE

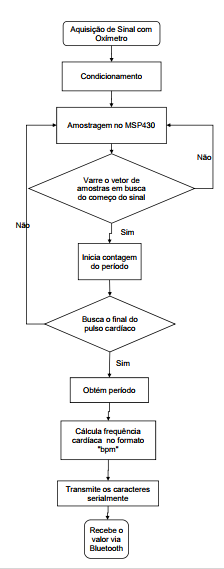
São criadas as variáveis globais que serão utilizadas no software como os marcadores de início e final do pulso cardíaco, vetor de amostras, “string” para transmissão e demais.

Primeiramente desativa-se o “Whatchdog Timer”, pois não é utilizado. A função “setClock(void);” configura o “Sub-System Master Clock” (SMCLK) em 1MHz e o “Auxiliary Clock” (ACLK), com suas divisões sucessivas em BCSCTL1 e BCSCTL2.

Em seguida chamam-se as funções que configuram o “Analog-to-Digital Converter” (ADC) e o “Universsal Serial Comunication Interface” (USCI), “adc\_Setup();” e “setUART();” respectivamente. O ADC é configurado para realizar 64 amostragens com frequência de amostragem 64Hz na porta P1.0, a flag de interrupção também é setada. Os registradores de transmissão são setados para uma transmissão serial UART, na frequência do SMCLK. O “TIMER A” é configurado em modo de Captura com borda de subida de descida, além do acionamento da “flag” de interrupção.

O laço do código consiste em amostrar o sinal de entrada em P1.0 e armazenar as amostras no vetor “adc”, são feitas interações no vetor para verificar-se o começo do sinal com borda de subida, então ativa-se a flag do “timer” para se começar a contagem das amostras, até o final do pulso cardíaco representado no vetor “adc”. Logo faz-se a subtração dos dois instantes “T1 – T0” para se obter o período armazenado na variável “period”. O valor do período somente é armazenado se a variável “period” fosse atualizado durante a varredura do vetor “adc”, pois, se não houvessem bordas indicativas de pulsos cardíacos no vetor o sinal deveria ser amostrado novamente.

O bpm é calculado com o inverso do período com o fator de contagem. Então este valor é alocado em um vetor de caracteres “word” e sua transmissão é feita com a inserção de cada caractere no “buffer” a ser de “TX”.



*Figura 4 – Fluxograma da descrição do software.*

Além do software do microcontrolador, também foi feita uma programação para criar o aplicativo, que recebe por bluetooth a quantidade de BPM e mostra este valor na tela do celular. Este aplicativo foi criado na plataforma App Inventor do MIT e sua programação é feita a partir de blocos de conexão.

1. RESULTADOS

Para a execução plena deste projeto, seria necessário uma obtenção rápida e eficaz do sinal cardíaco, sua amplificação pelo circuito eletrônico e seu tratamento, análise e cálculos para a determinação da frequência cardíaca através do MSP430G2553.

Para a obtenção da frequência cardíaca, era esperado que o sinal proveniente do oxímetro fosse facilmente obtido e, que o sinal fosse limpo e com variação maior do que a realmente apresentada por este. Com isso, como o sinal não era o esperado, foi necessário além dos outros componentes do circuito eletrônico que desacoplaram, amplificaram e diminuíram o ruído do sinal, um comparador que enviasse nível lógico alto com tensão entre 1,8 e 3,6 V, que corresponderia a sua alimentação como saída, caso sua comparação fosse verdadeira. Uma vez que, a tensão do sinal não varia dentro do intervalo de atuação do MSP430G2553, de modo que, não pode ser mandado direto para as portas do microcontrolador.

Com relação ao próprio oxímetro, este não funciona como esperado e nem com exatidão, uma vez que este é sensível a qualquer movimento, mesmo que sútil, gerando um ruído suficiente para atrapalhar o sinal de saída recebido pelo microcontrolador e assim, gerar um valor de frequência cardíaca incorreta.

Com relação ao executado pelo MSP430G2553 com o código escrito, o programa executou as funções, tratou e analisou o sinal conforme era desejado. Além de que, imprimiu o período a partir do sinal inserido e calculou a frequência cardíaca, enviando este valor para o módulo bluetooth, que por sua vez foi pareado com o celular android e através do aplicativo exibiu a frequência cardíaca em BPM que estava sendo medida no oxímetro.

1. CONCLUSÃO

Para a determinação da frequência cardíaca em batimentos por minuto (BPM), é necessária a obtenção do sinal, seu tratamento, análise e cálculos a partir dos valores obtidos do sinal. Para realização destas funções foram utilizados um circuito eletrônico que desacopla, amplifica e compara, enviando assim um sinal com tensão compatível com a que tem de ser recebida pelo microcontrolador. Foi utilizado o MSP430G2553 que foi codificado para tratamento do sinal, obtenção do período da onda e cálculo da frequência cardíaca. Após realizadas destas funções, a frequência calculada é enviada para o módulo bluetooth, que conectado ao celular andoid e envia os dados para um aplicativo desenvolvido na plataforma App Inventor disponibilizada pelo MIT.

Em suma, foram obtidos resultados satisfatórios com a realização deste projeto. Somente surgindo problema na obtenção do sinal no sensor, entretanto com o circuito utilizado foi possível obter o sinal necessário para trabalhar com as funções no microcontrolador, e por fim, somente restando erros na medição quando houvesse movimento no dedo em que estava sendo medido o sinal, e assim, gerando ruído suficiente para atrapalhar a contagem e, desse modo apresentando valores incompatíveis com a real frequência cardíaca.

REFERÊNCIAS

1. BARBOSA, Daniel - Intensidade de sessões de treinamento. Disponível em:<http://www.journals.usp.br/rbefe/article/view/16696> . Acessado em: 04 set. 2017.
2. Alves, Rodrigo Nascimento- Monitoramento e prevenção em atletas.
3. ROSERO, Oscar. Sistema móvel de monitoramento e treinamento para ciclista com smartphone android. Brasília, fev. 2012. Disponível em:< <http://repositorio.unb.br/bitstream/10482/11371/1/2012_OscarFernandoGaidosRosero.pdf>> . Acessado em: 28 nov. 2017.
4. Adolpho Ferreira, Artur F. C. Lemos, Lucas Mengarda, Tarcisio Leão. Estudo de oximetria de pulso em sistema de multiposicionamento via impressão 3D e monitoração por aplicativo de celular. São Paulo. Disponível em:< <http://www.obi2017.com.br/anais/PDF/01-022.pdf>> . Acessado em: 27 nov. 2017.

ANEXO – CÓDIGO DO MSP430G2553

//

/// Trabalho final da disciplina Microcontroladores e Miprocessadores - FGA

/// Professor: Diogo Caetano

/// Autores: Gilvan Camargo 14/0141537

/// Maria Carolina 14/0153403

///

/// Frequência Cardíaca com Oxímetro

/// Amostragem de sinal cardíaco e transmissão de batimentos por minuto

**#include** "msp430g2553.h"

// Global variables

**unsigned** **int** i=0; // Contador

**volatile** **unsigned** **int** enable=0, T0=0, T1=0; // Verificam início e final do pulso cardíaco

**volatile** **unsigned** **int** period\_in = 0, period\_out = 0, bpm = 0; // Calculam período do pulso

**int** adc[64] = {0}; //Vetor para armazenar amostras

**int** c = 0, d = 0, u = 0; // centenas, dezenas e unidades para armazena o "bpm"

**char** word[8] = "CDU BPM"; // Caracteres para BPM

// Function prototypes

**void** **setClock**(**void**);

**void** **adc\_Setup**(**void**);

**void** **adc\_Sam64**(**void**);

**void** **setUARTInterrupt**(**void**);

**void** **setUART**(**void**);

**void** **TimerA\_configure** (**void**);

**void** **Transmitir**(**volatile** **unsigned** **char** sinal);

**void** **get\_bpm**(**void**);

**void** **write\_on\_word** (**void**);

**void** **main**(**void**)

{

WDTCTL = WDTPW + WDTHOLD; // Stop the Watch dog

setClock();

adc\_Setup();// Fucntion call for adc\_setup

setUART();

**while** (1)

{

i=0;

adc\_Sam64(); // Function call for adc\_samp

TimerA\_configure ();

get\_bpm();

write\_on\_word();

setUARTInterrupt();

**while**(1)

{

Transmitir(word[i]); //RECEPTOR PUTTY COM5

i++;

**if**(i >= 7){

i=0;

**break**;

}

}

}

}

**void** **write\_on\_word** (**void**)

{

//Write Digits on string with ANSI

c = (bpm / 100) + 48;

**if** (bpm < 100)

d = (bpm / 10) + 48;

**else** {

d = bpm;

**while** (d>100){

d -= 100;

}

d = d/10 +48;

}

u = (bpm % 10) + 48;

word[0] = c;

word[1] = d;

word[2] = u;

word[7] = '\n';

}

**void** **get\_bpm**(**void**)

{

i=0;

enable = 0;

period\_in = period\_out;

**for** (i=1; i<65 ; i++)

{

**if** (enable ==2)

**break**;

**if**(adc[i-1]<30 & adc[i]>800){

enable++;

TA0CCTL0 ^= CCIS0; // Interrupt request (capture condition)

}

}

i=0;

**if** (period\_in == period\_out)

bpm = 0;

**else**

bpm = (600000/period\_out);

}

**void** **setUARTInterrupt**(**void**)

{

IE2 |= UCA0TXIE; // Enable the Transmit interrupt

IE2 |= UCA0RXIE; // Enable the Receive interrupt

\_BIS\_SR(GIE); // Enable the global interrupt

}

**void** **setClock**(**void**)

{

DCOCTL = 0; // Select lowest DCOx and MODx settings

BCSCTL1 = CALBC1\_1MHZ +XTS + DIVA\_3; // Set range

BCSCTL2 |= DIVA\_3; //32768/8 = 4,096kHz

DCOCTL = CALDCO\_1MHZ; // Set DCO step + modulation

}

// ADC set-up function

**void** **adc\_Setup**(**void**)

{

ADC10CTL1 = ADC10DIV\_7 + CONSEQ\_2 + INCH\_0 + ADC10SSEL\_1; // 4,096K/8 = 510Hz + ADC10SSEL\_3; // Repeat single channel, A0

ADC10CTL0 = ADC10SHT\_1 + MSC + ADC10ON + ADC10IE; // 510 /8 = 64Hz Sample & Hold Time + ADC10 ON + Interrupt Enable

ADC10DTC1 = 0x40; // 64 conversions

ADC10AE0 |= 0x01; // P1.0 ADC option select

}

// ADC sample conversion function

**void** **adc\_Sam64**(**void**)

{

ADC10CTL0 &= ~ENC; // Disable Conversion

**while** (ADC10CTL1 & BUSY); // Wait if ADC10 busy

ADC10SA = (**int**)adc; // Transfers data to next array (DTC auto increments address)

ADC10CTL0 |= ENC + ADC10SC; // Enable Conversion and conversion start

**\_\_bis\_SR\_register**(CPUOFF + GIE);// Low Power Mode 0, ADC10\_ISR

}

**void** **Transmitir**(**volatile** **unsigned** **char** sinal)

{

UCA0TXBUF = sinal; // Transmit a byte

\_delay\_cycles (5000);

}

**void** **setUART**(**void**)

{

P1SEL |= BIT1 + BIT2; // P1.1 UCA0RXD input

P1SEL2 |= BIT1 + BIT2; // P1.2 UCA0TXD output

UCA0CTL1 |= UCSSEL\_2 + UCSWRST; // USCI Clock = SMCLK,USCI\_A0 disabled

UCA0BR0 = 104; // 104 From datasheet table-

UCA0BR1 = 0; // -selects baudrate =9600,clk = SMCLK

UCA0MCTL = UCBRS\_1; // Modulation value = 1 from datasheet

//UCA0STAT |= UCLISTEN; // loop back mode enabled

UCA0CTL1 &= ~UCSWRST; // Clear UCSWRST to enable USCI\_A0

}

**void** **TimerA\_configure** (**void**)

{

TA0CCTL0 = CM\_3 + CAP + SCS + CCIS1 + CCIE; // Capture on rising edge + both edges capture + sync with the clock + interrupt flag.

**\_\_bis\_SR\_register**(GIE); // set the global interruptions

TA0CTL = TASSEL\_2 + MC\_2; // timer´s clock is system master clock, whithout division.

}

**#pragma** vector = USCIAB0TX\_VECTOR

**\_\_interrupt** **void** **TransmitInterrupt**(**void**)

{

//P1OUT ^= BIT0;//light up P1.0 Led on Tx

IFG2 &= ~UCA0TXIFG;

}

**#pragma** vector = USCIAB0RX\_VECTOR

**\_\_interrupt** **void** **ReceiveInterrupt**(**void**)

{

//P1OUT ^= BIT6; // light up P1.6 LED on RX

IFG2 &= ~UCA0RXIFG; // Clear RX flag

}

**#pragma** vector = ADC10\_VECTOR

**\_\_interrupt** **void** **ADC10\_ISR**(**void**)

{

**\_\_bic\_SR\_register\_on\_exit**(CPUOFF); // Clear CPUOFF bit from 0(SR)

}

**#pragma** vector=TIMER0\_A0\_VECTOR

**\_\_interrupt** **void** **Timer\_A**(**void**){

TA0CTL &= ~MC\_3; // Stop the timer (holds its value until the start)

**if**(enable == 1){

T0 = TA0CCR0; //reason: calling an interrupt and "stop timer command" demand clock cycles.

}**else** **if** (enable == 2){

T1 = TA0CCR0;

period\_out = T1 - T0;

TA0R = 0;

}

TA0CTL = TASSEL\_2 + MC\_2; // Reconfigure timer}

TA0CCTL0 &= ~CCIFG;

}