# Microprocessadores e Microcontroladores

Relatório final: Eletrocardiógrafo

Bruno Alves Ferreira Camargos
Programa de Engenharia Eletrônica
Faculdade Gama – Universidade de Brasília
15/0120117
E-mail: bruno.ferreirasg@hotmail.com

Gabriel de Matos Souza
Programa de Engenharia Eletrônica
Faculdade Gama – Universidade de Brasília
15/0126204
E-mail: gabriel.matos.s@hotmail.com

***Resumo*— Este é um projeto de um eletrocardiograma a ser feito com o auxílio de uma placa Msp430 com os conhecimentos adquiridos na disciplina microcontroladores emicrocomputadores na Universidade de Brasília.**

**Palavras chave — Msp, ECG, Eletrocardiograma;**

## I. INTRODUÇÃO

O eletrocardiograma (ECG), é um exame que utiliza eletrodos em contato com a pele para avaliar a atividade elétrica do coração. Essa atividade é possível ser notada através da variação na quantidade de íons de sódio dentro e fora das células musculares cardíacas e o seu resultado é registrado em gráficos detectando o ritmo do coração e o número de batimentos por minuto (Eletrocardiograma (ECG): o que é, para que serve e como é feito o exame, Redação CR).

Esse exame é capaz de identificar algumas avarias na saúde das pessoas (no caso atletas) como arritmias, infartos do miocárdio, distúrbios na condução elétrica do órgão, problemas nas válvulas do coração, entre outros.

Também pode ser indicado para exames preventivos para verificar a saúde do coração quando o paciente apresenta outras condições, como pressão alta, colesterol alto, diabetes, histórico familiar de doença cardíaca precoce.

Existem três diferentes tipos de eletrocardiograma, todos têm a mesma finalidade e são capazes de atingir o mesmo fim. A principal diferença entre cada tipo são as formas como cada um é feito.

a) ECG padrão (repouso ou de superfície)

Os eletrodos são conectados em partes específicas do corpo, como peitos, braços e pernas. Nos braços e pernas os eletrodos são fixados por braceletes e no tórax por uma espécie de ventosa de borracha.

b) ECG de esforço (teste ergométrico)

Nesse tipo de eletrocardiograma o teste é realizado enquanto o paciente executa algum tipo de exercício físico, geralmente em esteira ou na bicicleta.

c) Holter (monitorização de ECG ambulatorial)

Nesse tipo de exame um dispositivo registra a atividade cardíaca do paciente durante 24 horas.

## II. JUSTIFICATIVA

Tendo como objetivo a otimização do treino de um atleta, o presente projeto vem como uma melhoria na forma como os mesmos são avaliados quanto ao seu treinamento. Desta forma, o projeto utilizando um eletrocardiograma, irá determinar como a pessoa se comporta quando submetida a atividades físicas, assim sendo, a partir da análise dos dados gerados, um personal trainer poderá analisar de forma geral e verificar os esforços os quais o atleta poderá submete-se.

A ideia deste projeto nasceu tendo em vista que alterações nos batimentos cardíacos estão ligados diretamente com atividades físicas e o grupo havia pensado em criar um projeto relacionado a exames de ECG, utilizando a MSP430. Portanto, como a medição de batimentos cardíacos está ligada as atividades físicas, o projeto fora proposto.

## III. OBJETIVOS

O presente projeto tem como finalidade pegar um sinal de ECG proveniente do chip AD8232, amostra-lo em 500 Hz e transferi-lo por meio de um módulo bluetooth HC05 para o computador para que seja plotado no ambiente python, o qual servirá para irá monitorar um atleta em determinada atividade física exercida. O indivíduo terá o acompanhamento demonstrado pelo aparelho, o qual irá gerar dados os quais serão analisados pelo profissional responsável pelo treino do mesmo, desta forma a partir da análise dos dados poderá ser construído um treino o mais ideal possível para o atleta, inclusive poderão ser determinados novos limites de treino, os quais poderão mudar com a evolução do atleta. Desta forma o atleta estará protegendo a sua saúde, bem como otimizando sua atividade.

## IV. REQUISITOS

Para o funcionamento completo do projeto é requerido que se tenha um conhecimento dentro da área de programação e tecnologias e uma breve interação com microcontroladores, sensores e componentes eletrônicos. Também é de suma importância conhecimento na área de processamento de sinais uma vez que o projeto necessita da aplicação de filtros, sejam eles analógicos ou digitais.

## V. BENEFÍCIOS

O projeto permite um monitoramento e acompanhamento preventivo pessoal a saúde, algo que geralmente é conquistado através de exames, consultas e afins todas sob a prescrição de um médico.

## VI. Parte analógica

O projeto conta com um chip de captação de dados do ECG (AD8232). O esquemático do chip está disposto na figura 1.

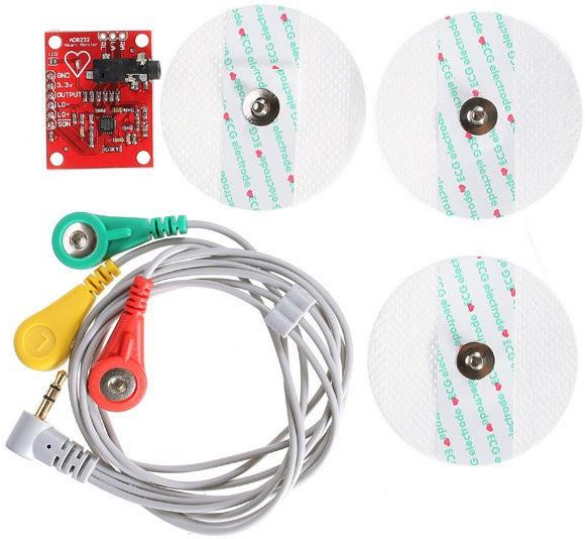

Figura 1 – Chip AD8232 (Fonte: - Electrofun).

O kit acima dispõe de um chip que filtra os sinais captados pelos eletrodos que são dispostos em lugares específicos indicados na figura 2.

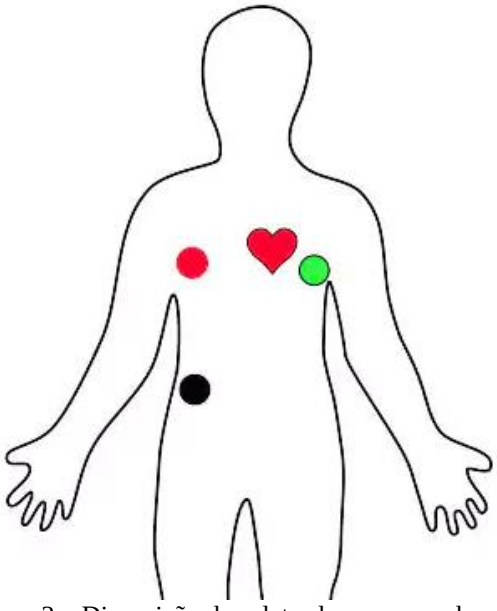

Figura 2 – Disposição dos eletrodos no corpo humano (Fonte: Gearbest).

O chip possui três saídas, sendo elas, duas digitais, e uma analógica, a qual apresenta os dados necessários. Entretanto, sua saída possui uma tensão de 3.3V, sendo que a MSP430 suporta, ocasionalmente, 2.5V, então será necessária uma divisão de tensão para que haja a conexão.

## VII. Parte digital

- Display LCD

Serve como interface entre o programa e o usuário, primeiramente ele disponibiliza a opção de iniciar o programa. Com o sistema inicializado ele permite pará-lo e também mostra o batimento cardíaco do usuário.

- Código em C

Analisando o código nomeado por main.c que está em anexo é possível perceber primeiramente a descrição do funcionamento do display Lcd tal como a função que determina seu modo de operação.

A função configura adc faz a conversão de sinais analógicos em sinais digitais por meio de aproximação com determinada frequência de amostragem, armazenando-os em ADC10. O Time A será citado a 1 ms ou a cada 500 Hz

O Timer A está definido como trigger, contando até TACCR0, quando este valor ser alcançado, será setado o valor 1 no TimerAOut2, ativando a conversão adc.

Para a comunicação, O módulo de blueetooth foi utilizado sendo conectado por via UART no modo HC-05. O módulo vem com a configuração de baudrate em 9600, porém o projeto precisava de uma baudrate de 115200, para realizar essa mudança teve que alterar sua configuração. Para configurar a baudrate do módulo é necessário que primeiramente execute o código da msp e mande para ela mesma, em seguida retirar a alimentação somente do módulo, depois desligar a msp por completo, ao ligar a msp, deve-se segurar um botão no módulo ao mesmo tempo em que é ligada sua alimentação, feito isso o módulo piscará um led com frequência de 0,5 Hz avisando que está em modo de configuração. Com isso, é pressionado o botão reset da msp para o código rodar de novo, desse modo o módulo estará configurado pois no código há uma string a ser enviada.

No código, sua função (Init uart) é trabalhada com uma taxa de transmissão. Ainda dentro da área de comunicação, foi utilizada uma função para o envio de dados (send data), que funciona recebendo oito bits, sendo os oito primeiros dígitos do número a ser enviado via bluetooth e os outros oito restantes o resto do número a ser enviado, pois o módulo não transmite mais do que oito bits por vez. Os dados são presos num buffer que são transmitidos ao fim de um laço de while.

As funções Lower byte e Uper byte são máscaras que são aplicadas nos sinais guardados no ADC10. A função Lower byte salva em si o valor do resultado de A AND 255, ou seja, seleciona apenas os oito primeiros bits, já a função uper byte desloca os primeiros bits para a direita e capta o restante dos números que ainda não foram enviados

A int main, além de conter os dados de controle da Msp430, inicia o Lcd, chama a função configura adc, chama a função configura timer A, inicia a comunicação UART com a taxa de transmissão de 115200, ativa as interrupções e define o modo de funcionamento como modo de baixo consumo.

A função Interrupt ADC10 é chamada após o ADC10 realizar uma conversão, logo depois é chamada a função lower byte que vai pegar os oito primeiros bits de ADCmain (registrador de 10 bits) e a função uper byte salvará os dois últimos bits restantes, em seguida a função send data é chamada para efetuar o envio tanto do lower byte quanto do

uper byte. Um led é utilizado para ficar alternando toda vez que houver uma chamada.

Para a lógica da interrupção do botão é utilizada a função interrupt p1, que fica presa num while enquanto o botão não for solto e após pressionado o ADC10 passa por uma NOT, logo, a conversão para.

- Código em Python

Para a aquisição de dados do ECG foi utilizado o código "ECG_AQUISITION".

Primeiramente, o "ECG aquisition" apresenta entre suas bibliotecas utilizadas, uma com o nome de serial que tem como função realizar a comunicação serial do python com o computador.

Mais abaixo no mesmo código, a linha iniciada com "ser" permite abrir uma porta seguindo as configurações disponíveis entre parênteses, como a baudrate, bit serial, 8 bits, paridade.

O primeiro laço "try" testa se a porta está aberta para poder continuar efetuando o código, caso não esteja, o sistema apresenta erro e para de funcionar.

Considerando o fato da porta estar aberta, o código entra num laço "while" dentro do "if", salvando na string data dois bytes que foram lidos por "ser.read". Em seguida ele converte cada byte que é hexadecimal em números inteiros com a função "struct.unpack", o que acaba se tornando um vetor de duas posições que é salvo em "data1". Logo após a conversão esse vetor é separado em "uper", que desloca o bite em 8 posições pegando somente os bits mais significativos, e em "lower", que recebe os bits menos significativos. Por fim, "dado_pronto" recebe "uper or lower", ou seja, soma os dois bites formando a informação novamente.

Por fim, os dados recebidos são salvos em um arquivo .txt no computador.

Para a plotagem do gráfico, localizada no código 'graph_ECG.py" foram utilizadas as bibliotecas "numpy", que é usada em cálculos com arrays e/ou números inteiros e a matplotlib para a plotagem em si.

O código inicia com a leitura dos dados salvos no arquivo .txt, salvando-os em um grande vetor nomeado de data conforme o que for lido. Em seguida, o valor do vetor data é salvo em data1 para garantir que não haja perca de informação. Por fim, a função "plt.plot()" pega amostras de 0 a 1000 do vetor data1, plotando em um gráfico.

## VIII. Resultados

Após a realização de todos os códigos e testes, os eletrodos foram conectados ao corpo seguindo a figura 2. A seguir veremos alguns resultados e suas explicações.

No primeiro teste, o sinal foi adquirido dentro de um laboratório com bastante interferência da rede elétrica (60 Hz) e com o notebook conectado a tomada, o que ocasionou em um gráfico com bastante ruído.

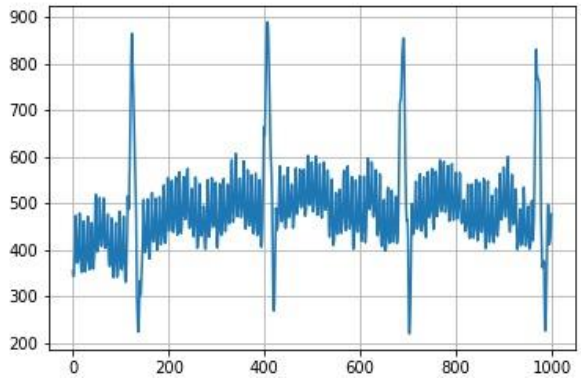

Figura 3 – Gráfico do sinal de ECG com bastante ruído

O próximo teste foi feito fora do laboratório e sem o notebook estar conectado à rede elétrica, o que ocasionou em um gráfico de melhor interpretação, ou seja, com menos ruídos.

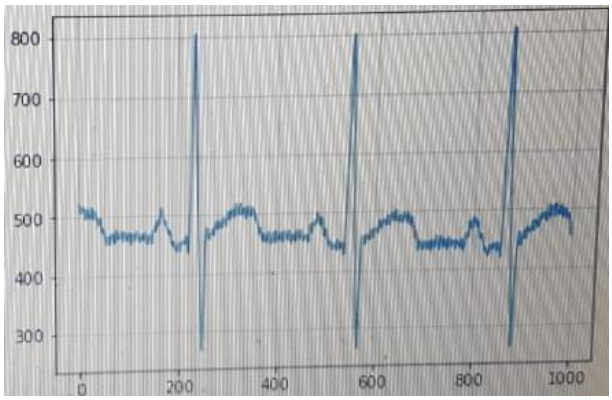

Figura 4 – Gráfico do sinal de ECG com menos ruído

## IX. Conclusão

A partir dos códigos efetuados para este projeto foi possível plotar um gráfico com as informações de um ECG vindas de um chip que fora conectado no corpo, portanto essas informações não foram passadas em tempo real para o gráfico, uma vez que as mesmas são atrasadas por serem salvas antes de ser plotadas. Não houve essa representação em tempo real pela falta de prática com a linguagem em python.

## X. Referências

[1] Redacão CR. Eletrocardiograma (ECG): o que é, para que serve e como é feito o exame. Disponível em: < https://minutosaudavel.com.br/eletrocardiograma-ecg-o-que-e-para-que-serve-e-como-e-feito-o-exame/> Acesso em 03 de abril de 2018.

[2] Scott. DIY ECG Machine On The Cheap. Disponivel em : <https://www.swharden.com/wp/2009-08-14-diy-ecg-machine-on-the-cheap/> Acesso em 03 de abril de 2018.

[3] https://github.com/DiogoCaetanoGarcia/Microcontroladores

[4] HC-03/05 Embedded Bluetooth Serial Communication Module

[5] Eletrocardiograma usando Placa de Som – ECG Caseiro, Nova eletrônica. Disponível em: < http://blog.novaeletronica.com.br/eletrocardiograma-usando-placa-de-som-ecg-caseiro/ >

## XI. CÓDIGOS DO PROJETO

- Códigodo projeto (Main.c)

```
#include <msp430g2553.h>
#include <legacymsp430.h>
#define BAUD_9600   0
#define BAUD_19200  1
#define BAUD_38400  2
#define BAUD_56000  3
#define BAUD_115200 4
#define BAUD_128000 5
#define BAUD_256000 6
#define NUM_BAUDS   7

#define BTN  BIT3
#define RX BIT1
#define TX BIT2

#define lcd_port        P2OUT
#define lcd_port_dir    P2DIR

#define LCD_EN      BIT4
#define LCD_RS      BIT5

void lcd_reset()
{
   lcd_port_dir = 0xff;
   lcd_port = 0xff;
   __delay_cycles(20000);
   lcd_port = 0x03+LCD_EN;
   lcd_port = 0x03;
   __delay_cycles(10000);
   lcd_port = 0x03+LCD_EN;
   lcd_port = 0x03;
   __delay_cycles(1000);
   lcd_port = 0x03+LCD_EN;
   lcd_port = 0x03;
   __delay_cycles(1000);
   lcd_port = 0x02+LCD_EN;
   lcd_port = 0x02;
   __delay_cycles(1000);
}

void lcd_cmd (char cmd)
{
   // Send upper nibble
   lcd_port = ((cmd >> 4) & 0x0F)|LCD_EN;
   lcd_port = ((cmd >> 4) & 0x0F);

   // Send lower nibble
   lcd_port = (cmd & 0x0F)|LCD_EN;
   lcd_port = (cmd & 0x0F);

   __delay_cycles(4000);
}

void lcd_init ()
{
   lcd_reset();        // Call LCD reset
   lcd_cmd(0x28);      // 4-bit mode - 2 line - 5x7 font.
   lcd_cmd(0x0C);      // Display no cursor - no blink.
   lcd_cmd(0x06);      // Automatic Increment - No Display shift.
   lcd_cmd(0x80);      // Address DDRAM with 0 offset 80h.
   lcd_cmd(0x01);      // Clear screen
}

void lcd_data (unsigned char dat)
{
   // Send upper nibble
   lcd_port = (((dat >> 4) & 0x0F)|LCD_EN|LCD_RS);
   lcd_port = (((dat >> 4) & 0x0F)|LCD_RS);

   // Send lower nibble
   lcd_port = ((dat & 0x0F)|LCD_EN|LCD_RS);
   lcd_port = ((dat & 0x0F)|LCD_RS);

   __delay_cycles(4000); // a small delay may result in missing char display
```

```
}


void display_line(char *line)
{
   while (*line)
      lcd_data(*line++);
}


void atraso(volatile unsigned int i)
{
   while((i--)>0);
}

configura_adc(void)
{
   ADC10CTL1  = SHS_1 + CONSEQ_2 + INCH_0+ ADC10DIV_1;                    // timerAout1 dispara - leitura consecutiva - canal 1 leitura
p1.1
   ADC10CTL0  = SREF_1 + ADC10SHT_2 + REFON + ADC10ON + ADC10IE;    // tensao referencia - clks para aproximçao - referencia
tensao interna on - adcon - interrupt on
   ADC10CTL0 &= ~ENC;
   ADC10AE0 |=0x0;

}
configurar_timerA(void)
{
   TACCR0 = 250-1;                           //conta ate este valor e zera
   TACCTL1 = OUTMOD_3;                              // leva a saida para quato contar 2047
   TACCR1 = TACCR0-1;                          //
   TACTL = TASSEL_2 | ID_3 | MC_1;                 //clk aux - contar up
}

void Init_UART(unsigned int baud_rate_choice)
{
   unsigned char BRs[NUM_BAUDS] = {104, 52, 26, 17, 8, 7, 3};
   unsigned char MCTLs[NUM_BAUDS] = {UCBRF_0+UCBRS_1,
                                     UCBRF_0+UCBRS_0,
                                     UCBRF_0+UCBRS_0,
                                     UCBRF_0+UCBRS_7,
                                     UCBRF_0+UCBRS_6,
                                     UCBRF_0+UCBRS_7,
                                     UCBRF_0+UCBRS_7};
   if(baud_rate_choice<NUM_BAUDS)
   {
      // Habilita os pinos para transmissao serial UART
      P1SEL2 = P1SEL = RX+TX;
      // Configura a transmissao serial UART com 8 bits de dados,
      // sem paridade, comecando pelo bit menos significativo,
      // e com um bit de STOP
      UCA0CTL0 = 0;
      // Escolhe o SMCLK como clock para a UART
      UCA0CTL1 = UCSSEL_2;
      // Define a baud rate
      UCA0BR0 = BRs[baud_rate_choice];
      UCA0BR1 = 0;
      UCA0MCTL = MCTLs[baud_rate_choice];
   }
}

void Send_Data(unsigned char c)
{
   while((IFG2&UCA0TXIFG)==0);
   UCA0TXBUF = c;
}

void Send_String(char str[])
{
   int i;
   for(i=0; str[i]!= '\0'; i++)
      Send_Data(str[i]);
}


unsigned char lower_byte(unsigned int A)
{
  return A & 255;
```

```
}

unsigned char uper_byte(unsigned int A)
{
   return A >> 8;
}

int adc=0,j=1;

int main(void)
{

   WDTCTL = WDTPW | WDTHOLD;                  // stop watchdog timer
   BCSCTL1= CALBC1_1MHZ;
   DCOCTL=  CALDCO_1MHZ;
   P1DIR |=  BIT6;                        //definindo o led3
   P1OUT &= ~BIT6;                          //estado inicial desligado

   P1DIR &= ~BTN;                          //para o botão
   P1REN |= BTN;
   P1OUT |= BTN;
   P1IES |= BTN;
   P1IE |= BTN;

   lcd_init();
   lcd_cmd(0x80); // select 1st line (0x80 + addr) - here addr = 0x00
   display_line("Pressione Botao");
   lcd_cmd(0xc0); // select 2nd line (0x80 + addr) - here addr = 0x40
   display_line("  Para iniciar");

   configura_adc();
   configurar_timerA();
   // Send_String("AT+UART=115200,0,0,\r\n");
   Init_UART(BAUD_115200);


   _BIS_SR(LPM0_bits+GIE);
   return 0;
}

//a interrupçao é chamada sempre uma nova converção é realizada
interrupt(ADC10_VECTOR) ADC10_ISR(void)
{

   unsigned char l = lower_byte(ADC10MEM), u = uper_byte(ADC10MEM);

   Send_Data(u);
   Send_Data(l);
   P1OUT ^= BIT6;  //a cada converçao o led troca o estado
   //UCA0TXBUF=0;

}

interrupt(PORT1_VECTOR) Interrupcao_P1(void)
{
   while((P1IN&BTN)==0);
   ADC10CTL0 ^= ENC;
         j ^=BIT0;

   if (j==0)
   {
      lcd_init();
      display_line("Sistema Online");
      lcd_cmd(0xc0); // select 2nd line (0x80 + addr) - here addr = 0x40
      display_line("       :)");
   }
   else if (j==1)
   {
      lcd_init();
      display_line("Sistema Offline");
      lcd_cmd(0xc0); // select 2nd line (0x80 + addr) - here addr = 0x40
      display_line("       :|");

   }
   P1IFG &= ~BTN;
}
```

- Códigos em python

- ECG_AQUISITION

```
import serial
import struct
x=0
ser = serial.Serial(port='COM5', baudrate=115200, bytesize=serial.EIGHTBITS, parity=serial.PARITY_NONE,timeout = 2)

try:
   ser.isOpen()
   print("Porta Aberta")
except:
   print("error")
   exit()

if(ser.isOpen()):
    try:
       while(1):
          x+=1;
          data=ser.read(2)
          data1=struct.unpack('!BB', data)

          uper=data1[0] << 8
          lower=data1[1]
          dado_pronto= uper | lower
          file=open('recebidos.txt', 'a+')
          file.write("%i\n"  % (dado_pronto))

          print(dado_pronto)

    except Exception:
       print("error")
else:
   print("nao foi possivel conectar")
```

- Graf_ECG

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

data = pd.read_csv("recebidos.txt", header=None, delimiter=r"\s+")

data1 = np.asarray(data)

plt.plot(data1[0:1000])
plt.grid()
plt.show()
```