Detector de Mentiras

Disciplina: Eletrônica Embarcada - 2/2019 , Prof.Dr: Gilmar

Luísa Caroline Alves 16/0134587
Programa de Engenharia Eletrônica
Faculdade do Gama - Universidade de Brasília
St. Leste Projeção A - Gama Leste,
Brasília- DF, 72444-240
email: luisacarollinne@gmail.com

Manuella Cristina Panza 16/0135290
Programa de Engenharia Eletrônica
Faculdade do Gama - Universidade de Brasília
St. Leste Projeção A - Gama Leste,
Brasília- DF, 72444-240
email: manuellapanza@gmail.com

***Resumo*—O relatório consiste em apresentar o projeto final da disciplina Eletrônica Embarcarda, sendo esse um detector de mentiras: um dispositivo que mede a resistência da pele, e por meio da variação dessa resistência devido a razões fisiológicas seus resultados podem indicar se a pessoa está mentindo ou não. Para o projeto em questão essa análise será feita em um microcontrolador MSP430 e a demonstração do protótipo funcional com a resposta em um LCD.**

***Palavras-chaves:* MSP430, Microcontrolador, Resistência Galvânica da Pele, Eletrodo, Detector de Mentiras.**

I. Introdução

As pessoas contam mentiras pelos mais variados motivos. Na maioria das vezes, mentir é um mecanismo de defesa usado para evitar problemas com a família, com os chefes ou com as autoridades. Às vezes, é possível distinguir quando alguém está mentindo, mas na maioria das vezes, isso não é tão fácil. Polígrafos, mais conhecidos como detectores de mentira, são instrumentos que monitoram determinadas reações de uma pessoa. O corpo humano emite vários sinais quando é submetido a algum estado de pressão ou excitação, assim podendo variar a resistência da pele, batimentos cardíacos e outros aspectos, onde especialistas conseguem analisar e revelar a possível mentira.

O aparelho funciona de forma simples: existe um efeito chamado resposta galvânica da pele (GSR) que pode ser descrito como sendo um circuito que mede a impedância da pele por meio de um eletrodo de toque cujo o valor varia por uma conexão entre uma resistência pré-definida e a resistência obtida através de dois eletrodos posicionados nos dedos das mãos de um pessoa, um no indicador e outro no dedo médio, já que esses pontos são localizados as glândulas sudoríparas. [3]

Conforme a pessoa é estimulada psicologicamente, a quantidade de secreção (suor), liberada nesses pontos monitorados pelos eletrodos varia, alterando dessa forma a resistência medida. Portanto, quanto mais estimulado o sistema nervoso central estiver, mais suor as glândulas sudoríparas vão produzir e menor será a resistência medida nos eletrodos, visto que, o suor é um condutor elétrico, e dessa forma o aparelho acusa quando essa condutividade aumenta, uma vez que aumenta também a amplitude do sinal de saída do circuito.

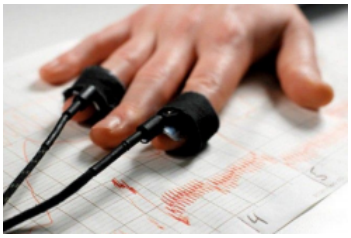

Figura 1. Detector de mentiras por resposta galvânica da pele.

O intuito dessa escolha para a disciplina foi acrescentar informações do indivíduo sobre inocência ou culpa por um interrogatório através desse equipamento, e, mesmo não sendo aceito como prova definitiva, ele pode ter uma determinada relevância em julgamentos.

Neste projeto, iremos utilizar como base o livro "30 Projetos Com Arduino - Série Tekne - Monk,Simon 2ª Ed. 2017", onde na página 157, demonstra o desenvolvimento de um detector de mentira, como lista de materiais, sketch do código e outros. Como na disciplina estudamos o microcontrolador MSP430, então, o projeto teve que ser adaptado para cumprir os requisitos da matéria. [2]

II. Desenvolvimento

Na realização desse projeto temos um circuito montado com os devidos componentes para obter o efeito GSR e a saída esperada. Assim, a medição dessa resistência foi analisada através de eletrodos de toque por meio de entrada analógica. Ao receber os sinais analógicos, o circuito possui um microcontrolador MSP430G2553 que realiza a conversão desses sinais para digital e os envia através de uma porta tipo USB, para o computador [3]. E por um LED, pode-se indicar a resposta como verdade ou mentira de uma forma digital, e através de um buzzer sinalizar sonoramente que a pessoa está mentindo.

A. Descrição do hardware

1) Lista de materiais:

- Microcontrolador MSP430: Dispositivo capaz de realizar controle de máquinas e equipamentos eletrônicos através de programas.

Figura 2. MSP430G2553.

- Resistores, para auxiliar a medida de resistência da pele.
- Buzzer, para indicar sinais de mentira.
- LED RGB cátodo comum, para indicar verdade.
- Jumpers, para fazer conexões entre os componentes.
- Protoboard, para fazer a montagem do circuito.
- Potenciômetro (trimpot), para ajustar o LED no início do teste.
- LCD 16x2

2) Montagem: O seguinte fluxograma foi feito para demonstrar uma ideia geral do projeto.

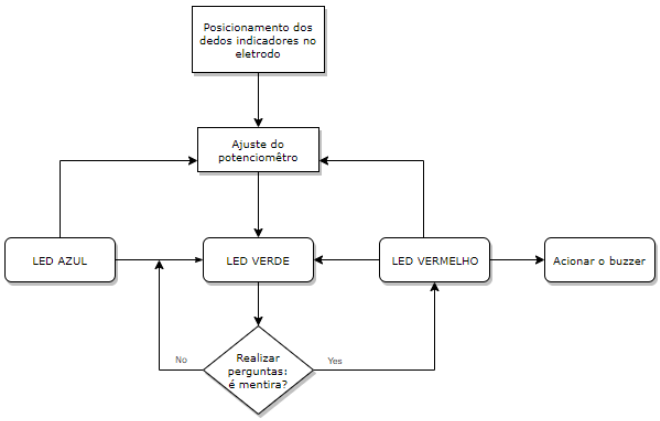

Figura 3. Fluxograma do projeto (hardware).

O desenho do esquemático na Fig. 4 apresenta todas as conexões dos componentes na MSP430.

3) Limitação: Sabendo que é possível que o interrogado talvez consiga manipular os resultados de um teste de detector de mentiras, esse aparelho acaba se tornando pouco confiável por si só. Além disso, ele mede os fatores fisiológicos associados não apenas à mentira, mas também ao nervosismo e isso pode ser um problema, já que qualquer pessoa pode ficar nervosa ao ser interrogada, inclusive aquelas que são inocentes.

4) Funcionamento: Usamos um LED multicolor RGB que acende a cor vermelha com o intuito de indicar uma mentira, a cor verde para uma verdade e a cor azul para indicar que o detector de mentira deve ser ajustado girando o resistor

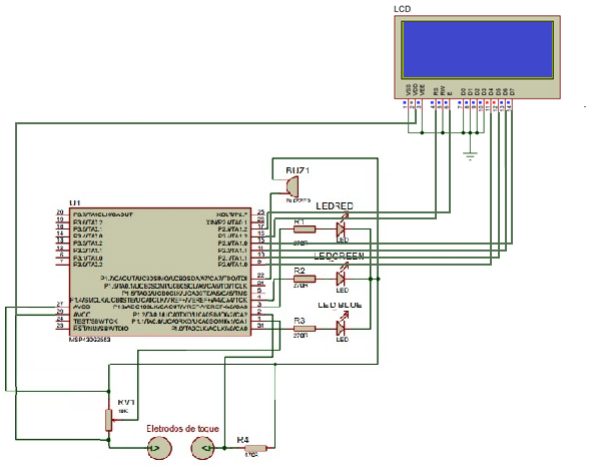

Figura 4. Desenho esquemático do circuito conectado a MSP430G2553.

variável. Este ajuste servirá para calibrar o ponto de ajuste da resistências. Os eletrodos de toque são dois percevejos metálicos inseridos na protoboard para detectar a resistência da pele.

Para medir essa resistência, a própria pessoa será utilizada como um dos resistores de um divisor de tensão. Já o outro resistor será um valor fixo de 470 Ω. Quanto menor for a resistência da pessoa, mais a entrada analógica 0 será puxada em direção ao Vcc e quanto maior a resistência, mais próximo estará do 0V. O buzzer consome pouca corrente e pode ser acionado em um pino digital da MSP430.

Utilizaremos dois periféricos analógicos com a finalidade de comparar as tensões entre eles. Se forem aproximadamente iguais, o LED ficará verde. Se a tensão vinda do sensor do dedo (em um periférico) for bem maior que a tensão do outro, o resistor variável irá indicar uma diminuição da resistência da pele, o LED ficará vermelho e o buzzer irá soar. Assim, se o outro periférico for significativamente menor do que o outro, o LED ficará azul, indicando um aumento de resistência da pele.

O LCD adicionado serve para leitura dos resultados de acordo com a saída obtida pela resistência, funcionando em conjunto com o LED para a mentira, verdade e ajuste.

B. Descrição do sofware

Para leitura da resistência ajustável do potenciometro, juntamente com a resistencia da pele, é captada pelo eletrodo de aço latonado, é usado o periférico ADC10 (Conversor Analógico Digital), no qual foi configurado para realizar mais de uma conversão consecutiva (CONSEQ_3), em múltiplos canais [1]. Após a conversão, os valores são armazenados nas variáveis sensor e poten (potênciometro). No sensor é feita uma média de valores para que obtenha uma melhor leitura e análise dos dados, utilizando 20 valores lidos.

Após encerrar a conversão entra-se em uma interrupção, onde são analisadas as condições que seleciona o resultado. Para sensor $>$ (poten + band), o resultado é que o interrogado

pode estar mentindo, quando sensor $<$ (poten - band) o detector deve ser ajustado e caso contrário o interrogado pode estar falando a verdade, onde band é sensibilidade (resistência pré definida). Todos os resultados obtidos na interrupção é mandado para o LCD (mentira, ajuste e verdade). O display LCD é configurado através da função InitLCD() que se encarrega de configurar as portas P2 ligá-lo. Para escrever no LCD é utilizada a função Send_String, que manda caracter por caracter da palavra a ser escrita para o display. Na função Atraso_us foi configurado o Timer_A para gerar os atrasos necessários para o correto funcionamento do display LCD.

A rotina assembly foi implementada em um arquivo buzzer.asm que foi colocado no projeto principal e onde era chamado como uma função. O código funciona basicamente ativando e desativando o buzzer.

Para melhor entendimento da lógica do projeto, foi desenvolvido o seguinte fluxograma:

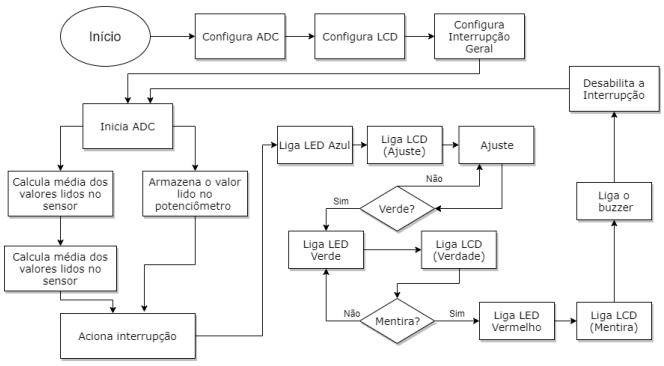

Figura 5. Fluxograma do projeto (software).

III. Resultados

Portanto, tem-se como resultado obtido desde etapas de desenvolvimento anteriores no funcionamento, conseguiu-se implementar o circuito do detector de mentiras observada na Fig. 4 com o display LCD em paralelo. Também foi concluída a estrutura para alocar todo o circuito juntamente com o display, como pode ser visto na Fig. 6 e Fig. 7, o protótipo funcional finalizado.

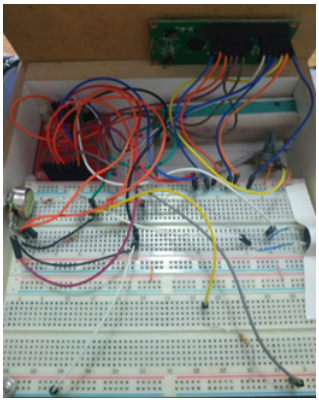

Figura 6. Circuito conectado a MSP430G2553.

Figura 7. Caixa do detector de mentiras.

A implementação do projeto ocorreu de maneira que a pessoa coloca dois dedos indicadores sobre os percevejos metálicos, e com isso é preciso ajustar o resistor variável até que o LED fique verde. Depois, ao fazer perguntas para o interrogado, observar se o LED vai ficar vermelho ou azul (necessitando do ajuste), e após cada pergunta e mudança da cor do LED ajusta-se o resistor variável novamente.

Verificou-se que o sistema funcionou um pouco mais preciso do que esperado, um assunto abordado antes devido às limitações do detector de mentira em si, como também os componentes utilizados em questão, temos o exemplo dos percevejos latonados (usado na medição da resistência da pele) não ser bem eficiente.

A interrupção juntamente com a conversão AD foram realizadas com sucesso. Também queríamos implementar uma comunicação I2C, porém foram encontradas certas complicações através do LCD, e além disso, houve dificuldades na inserção do código Assembly ao código em C.

Para a finalização do projeto os problemas foram resolvidos, com a junção do código assembly por meio de uma arquivo em Assembly feito tendo a possibilidade de chamar uma função para esse arquivo ao código em C e a conexão do LCD em pararelo, visto que tínhamos pinos suficientes. Após todas as mudanças no código, conseguimos um funcionamento satisfatório.

Conclusão

A partir do projeto, foi possível observar como funciona a resposta galvânica da pele, uma das variáveis fisiológicas de um polígrafo, e como essa reação pode ser manipulada ou equivocada, o que se torna um grande problema para uma análise real.

De forma a deixar o projeto com um funcionamento mais adequado, os eletrodos devem ser de alterados, pois o utilizado tem uma baixa precisão, tornando assim os resultados muitas vezes inconsistentes, sendo necessário o uso de filtros. Além disso, pode-se integrar ao projeto o módulo I2C onde se reduz o número de portas utilizadas, principalmente quando se trata do display LCD, um componente que geralmente consome muitas portas.

Uma outra forma de deixar o funcionamento do projeto mais adequado é utilizar PWM, de forma que, com a variação

da largura do pulso de cada um dos pinos do LED RGB, conseguisse uma gama maior de cores, fazendo com que a transição entre as 3 cores principais do projeto (verde, vermelho e azul) fosse feita de forma mais suave, passando por diversas outras cores, melhorando ainda mais a etapa de ajuste e calibração.

Portanto, pode-se concluir que, por ser um protótipo, o projeto obteve sucesso, pois todos os objetivos iniciais e adicionais foram executados com êxito, além de que, os conhecimentos adquiridos sobre as diversas funções da MSP430 foram em sua maioria aplicados.

REFERÊNCIAS

[1] John H. Davies. *MSP430 Microcontroller Basics*. Newnes, ago. de 2008.

[2] Simon Monk. *30 Projetos com Arduino (Tekne) (Portuguese Edition)*. Bookman, jan. de 2014.

[3] Geciane de Souza Pereira e Julio César Marques de Lima. "MRGP-Monitoramento da Resistência Galvânica da Pele". Em: *Revista da Graduação* 3.1 ().

APÊNDICE

Listing 1. Código principal

```
#include <msp430.h>
#include <legacymsp430.h>

#define LEDRED BIT3 // P1.3
#define LEDGREEN BIT4 // P1.4
#define LEDBLUE BIT0 // P1.0
#define N 20
//LCD
#define BTN BIT3
#define LCD_OUT P2OUT
#define LCD_DIR P2DIR
#define D4 BIT0
#define D5 BIT1
#define D6 BIT2
#define D7 BIT3
#define RS BIT4
#define E  BIT5
#define DADOS 1
#define COMANDO 0
#define CMND_DLY 1000
#define DATA_DLY 1000
#define BIG_DLY  20000
#define CLR_DISPLAY  Send_Byte(1, COMANDO
    , BIG_DLY)
#define POS0_DISPLAY Send_Byte(2, COMANDO
    , BIG_DLY)

// AD
#define CANAIS_ADC 2 // vetor com A0 e A1
#define IN_AD BIT1 // P1.1 potenciometro
#define IN_AD1 BIT2 // P1.2 sensor


int resis[CANAIS_ADC];

int sensor, poten ; // estara incluso em
    resis

int band = 230; //sensibilidade

int i,media = 0;

void Atraso_us(volatile unsigned int us);
void Send_Nibble(volatile unsigned char
    nibble, volatile unsigned char dados,
    volatile unsigned int microsegs);
void Send_Byte(volatile unsigned char
    byte, volatile unsigned char dados,
    volatile unsigned int microsegs);
void Send_Data(volatile unsigned char
    byte);
void Send_String(char str[]);
void Send_Int(int n);
void InitLCD(void);

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;    // stop
        watchdog timer

    P1DIR |= LEDRED;
    P1OUT = LEDRED;

    P1DIR |= LEDGREEN;
    P1OUT = LEDGREEN;

    P1DIR |= LEDBLUE;
    P1OUT = LEDBLUE;

    ADC10CTL0 = MSC + SREF_0 + ADC10SHT_0
        + ADC10ON + ADC10IE;
    ADC10CTL1 |= INCH_2 + ADC10SSEL_3 +
        CONSEQ_3; //multiplas convers es
    ADC10AE0 |= IN_AD + IN_AD1; // bits
        correspondentes a cada pino da msp
    ADC10DTC1 = CANAIS_ADC;
    ADC10CTL0 |= ENC + ADC10SC;
    InitLCD();
    _BIS_SR(GIE);

   while (1){
    poten = resis[0];
    for (i = 0; i < N-1 ; i++) {

    media+=resis[1]/N;   // faz uma media
        dos valores obtidos
    }
    sensor = media;
    media = 0;

    ADC10CTL0 &= ~ENC; //encerra a
        convers o AD

    while (ADC10CTL1 & BUSY); // garantir
        que o conversor AD n o esteja
        ocupado para iniciar novas
        convers es

    ADC10SA = (unsigned int)resis;//
        endere o do registrador que est
        guardando as convers es;

    ADC10CTL0 |= ENC+ADC10SC ;   // inicia
        a convers o AD
    _BIS_SR(GIE);

          }
```

```
}

#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void)
     {

    CLR_DISPLAY;
    POS0_DISPLAY;

      if ( sensor > poten + band){
        Send_String("    Mentira");
        P1OUT = LEDRED;
        //  ativar buzzer em assembly
        buzzer();

      }
      else if (sensor < poten - band) {
        Send_String("    Ajustar");
        P1OUT = LEDBLUE;

      }
      else {

        Send_String("    Verdade");
        P1OUT = LEDGREEN;

      }
      __bic_SR_register(GIE);
     }

void Atraso_us(volatile unsigned int us)
{
    TA1CCR0 = us-1;
    TA1CTL = TASSEL_2 + ID_0 + MC_1 +
        TAIE;
    while((TA1CTL & TAIFG)==0);
    TA1CTL = TACLR;
    TA1CTL = 0;
}

void Send_Nibble(volatile unsigned char
    nibble, volatile unsigned char dados,
    volatile unsigned int microsegs)
{
    LCD_OUT |= E;
    LCD_OUT &= ~(RS + D4 + D5 + D6 + D7);
    LCD_OUT |= RS*(dados==DADOS) +
        D4*((nibble & BIT0)>0) +
        D5*((nibble & BIT1)>0) +
        D6*((nibble & BIT2)>0) +
        D7*((nibble & BIT3)>0);
    LCD_OUT &= ~E;
    Atraso_us(microsegs);
}

void Send_Byte(volatile unsigned char
    byte, volatile unsigned char dados,
    volatile unsigned int microsegs)
{
    Send_Nibble(byte >> 4, dados,
        microsegs/2);
    Send_Nibble(byte & 0xF, dados,
        microsegs/2);
}

void Send_Data(volatile unsigned char
    byte)
{
    Send_Byte(byte, DADOS, DATA_DLY);
}

void Send_String(char str[])
{
    while((*str)!='\0')
    {
        Send_Data(*(str++));
    }
}

void Send_Int(int n)
{
    int casa, dig;
    if(n==0)
    {
        Send_Data('0');
        return;
    }
    if(n<0)
    {
        Send_Data('-');
        n = -n;
    }
    for(casa = 10000; casa>n; casa /= 10)
        ;
    while(casa>0)
    {
        dig = (n/casa);
        Send_Data(dig+'0');
        n -= dig*casa;
        casa /= 10;
    }
}

void InitLCD(void)
{
    unsigned char CMNDS[] = {0x20, 0x14,
        0xC, 0x6};
    unsigned int i;
    // Atraso de 10ms para o LCD fazer o
        boot
    Atraso_us(10000);
```

```
    LCD_DIR |= D4+D5+D6+D7+RS+E;
    Send_Nibble(0x2, COMANDO, CMND_DLY);
    for(i=0; i<4; i++)
        Send_Byte(CMNDS[i], COMANDO,
            CMND_DLY);
    CLR_DISPLAY;
    POS0_DISPLAY;
}
```

Listing 2. Código em Assembly

```
      .cdecls C,NOLIST,"msp430.h"       ;
          Processor specific definitions

      .global   buzzer                  ;
         Declare symbol to be exported
      .sect ".text"                     ;
          Code is relocatable
buzzer:    .asmfunc
    bis.b   #BIT7, &P1DIR
    bis.b   #BIT7, &P1OUT
    mov.w   #54000,R4

apaga:
    dec.w   R4
    jnz     apaga
    xor.b   #BIT7,&P1OUT
    .if ($defined(
        __MSP430_HAS_MSP430XV2_CPU__) |
        $defined(
        __MSP430_HAS_MSP430X_CPU__))
         reta
   .else
         ret
   .endif
          .endasmfunc

         .end
    jmp     apaga
```