Dispositivo para Controle de Medicação

Universidade de Brasília - Faculdade do Gama
Beatriz Freitas Calheira, *160003032*
Giovanna Amorim de Farias, *160007356*

Resumo: **O presente relatório descreve o desenvolvimento e resultados do projeto final realizado para conclusão da disciplina de Eletrônica Embarcada. O projeto tem como objetivo implementar um hardware e um software de um dispositivo de controle de medicação. O dispositivo é configurado para alertar o usuário dos medicamentos e liberá-los nos horários corretos. Ele emite um sinal sonoro e um sinal luminoso nos horários pré-estabelecidos e libera os medicamentos correspondentes na saída.**

***Palavras chaves*: medicação; controle; idosos, alerta.**

I. Introdução

O uso de medicamentos requer segurança e cuidado. Para tanto, é importante manipulá-los de forma segura e precisa. No Brasil, a utilização de grande número de medicamentos é amplamente observada entre indivíduos com 60 anos ou mais, já que esses apresentam maior incidência de doenças crônicas. [1]

O dispositivo pode ser utilizado por pessoas de qualquer faixa etária e até mesmo em instalações médicas. No entanto, o público alvo do projeto são pessoas idosas, que, em geral, necessitam de maior atenção, e apresentam dificuldades de memória.

Para auxiliar a elaboração deste projeto, serão utilizados como base projetos sobre automação de casinhas para pets, como sistemas de alimentação e de água. O princípio será o mesmo, um dispositivo que irá liberar algo dentro de um período de tempo determinado pelo usuário. Com isso, a reposição de remédios e checagem do nível da reserva será feita por intermédio do microcontrolador, assim como nos sistemas de alimentação.

O projeto consiste em um dispositivo de controle de medicação. O dispositivo é configurado para alertar ao paciente dos medicamentos e liberá-los nos horários corretos. Ele emite um sinal sonoro e um sinal luminoso nos horários pré-estabelecidos e libera os medicamentos correspondentes na saída.

II. Desenvolvimento

O objetivo do projeto é desenvolver um dispositivo que auxilie os pacientes idosos que fazem o uso de medicações controladas a manipular os medicamentos de forma adequada. E, consequentemente, favorecer uma melhor qualidade de vida a este grupo etário.

Para construção do protótipo, foram necessários os seguintes materiais:

- MSP430G2553 (1)
- Sensor HC-SR04 (1)
- Servo SG90 (1)
- LEDs (2)
- Buzzer (1)
- Suporte
- Tubos de silicone
- Protoboard (1)
- Jumpers

A escolha do microcontrolador MSP430 se deu pelo fato de ser um projeto de baixo consumo de energia. Além disso, sua flexibilidade em relação à sua arquitetura de portas facilita a utilização de sensores para entrada e saída de informações. Sendo este fator fundamental para implementar o processo de liberação dos medicamentos de acordo com os parâmetros e horários determinados;

A partir do objetivo exposto, o sistema irá permanecer em modo de baixo consumo até a interrupção determinada com base no horário em que a medicação deve ser administrada. Ao atingir este horário, o led na cor amarela irá ligar e um sinal sonoro será emitido. Enquanto o comprimido estiver no dispositivo, o sensor manterá este led ligado.

Quando o sensor detectar que o recipiente está sem o medicamento, irá voltar para o modo de baixo consumo, desligará o led amarelo e ligará o led verde. Este ciclo irá se repetir periodicamente.

Para implementação do dispositivo, foi desenvolvido um código em linguagem C. Este código permite o controle do sistema de alerta, leds e buzzer, e o sistema de liberação de medicamento, constituído pelo servo motor.

Por meio do sensor ultrassônico, foi implementada uma condicional para verificar se o medicamento ainda

se encontra no dispositivo. Se essa condição for verificada, o led amarelo e o buzzer são acionados. Senão, o led verde permanece aceso.

O sistema de liberação de medicamento funciona a partir de um período determinado, intervalo entre um medicamente e outro, quando este período é zerado, por meio de uma função de decremento, o motor é ativado. Com isso, o comprimido será liberado.

Somado a isto, o circuito que realiza a comunicação com o microcontrolador MSP430G2553 foi implementado, utilizando todos os componentes para comunicação com o usuário.

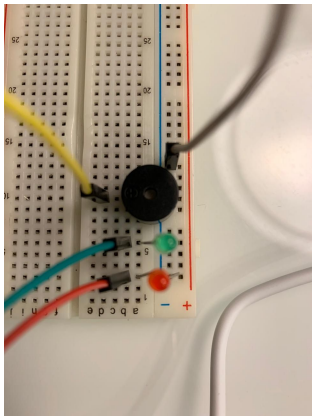

Figura 01: Sistema de Alerta.

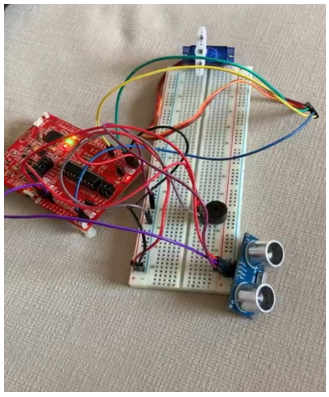

Figura 02: Sistema com periféricos.

Inicialmente, o Watchdog Timer é suspenso, são definidos os clocks, as flags são limpas e as interrupções habilitadas. Após estas etapas, os leds são definidos e a interrupção global é habilitada.

Em seguida, o sensor para verificação do comprimido na base tem seus TRIGGER e ECHO definidos como saída e entrada, respectivamente. Seu funcionamento é a partir de picos de clock, tendo como referência o clock da placa. Este sensor está diretamente atrelado ao acionamento do led amarelo e do buzzer, constituindo o sistema de alarme.

Para utilizar os dados de entrada do sensor, foi transformado o valor captado em centímetros. Este valor permite a verificação se o medicamento ainda se encontra na base. Por fim, foi definido a função de saída do servo motor.

Os códigos implementados são apresentados nos apêndices A e B, bem como descrição e explicação de cada uma das partes por meio de comentários nas linhas de código. Sendo o código em c do projeto final e o código em assembly do sistema de alarme, respectivamente.

III. Resultados

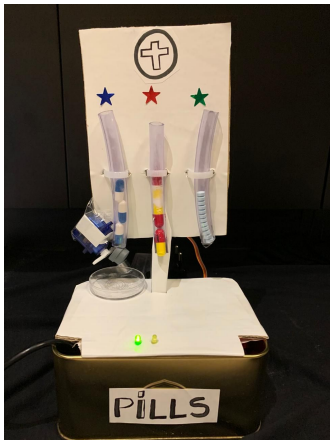

Figura 01: Protótipo Funcional 1.

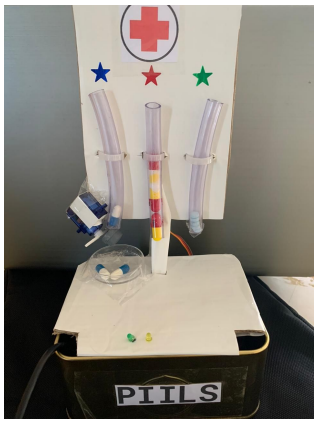

Figura 02: Protótipo Funcional 2.

Para construção do protótipo, utilizou-se uma estrutura construída com uma base de papelão e tubos de silicone, para armazenamento das medicações dos usuários. Além disso, definiu-se o posicionamento do motor, do sistema de alerta e do depósito de comprimidos.

No desenvolvimento do protótipo funcional, foi possível observar a necessidade de implementação de outros módulos de liberação de medicamento. Os demais módulos possuem funcionamento semelhante ao módulo

apresentado nesta etapa.

O refinamento do código em C foi dado a partir da definição da temporização das funções de interrupção. O tempo entre as interrupções é definido a partir da necessidade do usuário. No entanto, para apresentação do projeto definiu-se um tempo de 10 segundos entre uma rotação do motor e outra. E o acionamento direto do motor por intermédio de um botão.

IV. Conclusão

Inicialmente, encontrou-se dificuldades quanto à adequação da MSP430 ao projeto proposto. O número de portas e implementação de uma rotina em assembly foram os desafios iniciais.

Outro desafio encontrado foi a calibração do sensor, pois o mesmo não apresenta uma sensibilidade adequada para percepção do medicamento. Após a transformação do sinal captado em centímetros e teste para posicionamento do sensor no protótipo funcional, o problema foi solucionado.

Ao final do projetos, alguns pontos de melhoria se tornaram perceptíveis. Para um projeto futuro, a implementação de um Display LCD indicando o nome do remédio liberado facilitaria o conhecimento do usuário quanto ao medicamento a ser manipulado. Somado a isto, o uso de um sensor com uma sensibilidade mais significativa facilitaria a sinalização da liberação do medicamento pelo sistema.

No entanto, projeto se mostrou viável e aplicável. O sistema de alarme e liberação de medicamento funcionou conforme o esperado, apresentando bom desempenho e eficácia. O sistema de alerta e liberação obtiveram bons resultados nos testes, facilitando o desenvolvimento do protótipo funcional.

V. Referências

[1] DA SILVA, ANDERSON L. Utilização de medicamentos por idosos brasileiros, de acordo com a faixa etária: um inquérito postal. SciELO, Saúde Pública, Rio de Janeiro, junho, 2012.

[2] Automatic Pet Watering System. Disponível em <https://create.arduino.cc/projecthub/SindreKragsrud/automatic-pet-watering-system-9bfc46?ref=tag&ref_id=pets&offset=6> Acesso em: 06 de Dezembro de 2019.

[3] Nas farmácias, venda de remédio subiu 42% em cinco anos. Disponível em <https://infograficos.estadao.com.br/focas/tanto-remedio-para-que/checkup-1.php> Acesso em: 06 de Dezembro de 2019.

[4] Datasheet MSP430G2553. Disponível em <https://www.ti.com/lit/ds/symlink/msp430g2553.pdf> Acesso em 06 de Dezembro de 2019.

APÊNDICE A - CÓDIGO EM C PROJETO FINAL

```
#include <msp430.h>


//Definição das variáveis
int k;
int time_ms;
int distance;
long sensor;

void delay(volatile unsigned long i)
{
    do (i--);
    while (i != 0);
}

int main(void)
{
  BCSCTL1 = CALBC1_1MHZ;
  DCOCTL = CALDCO_1MHZ;
// Clock de 1Mhz
  WDTCTL = WDTPW + WDTHOLD;
// Desliga o WatchdogTimer

  CCTL0 = CCIE;
// Habilita a interrupção pelo CCR0
  CCR0 = 1000;
// 1ms em 1mhz
  TACTL = TASSEL_2 + MC_1;
// SMCLK - UpMode

  P1IFG  = 0x00;
// Limpa todas as flags de interrupçao

  P1DIR |= 0x01;
// P1.0 como saída do LED amarelo
  P1OUT &= ~0x01;
// Desliga o LED

  P2DIR |= 0x01;
// P2.0 como saida do LED verde
  P2OUT &= ~0x01;
// Desliga o LED

  _BIS_SR(GIE);
// Habilita a interrupcao global
```

```
  while (1)
  {
        P1IE &= ~0x01;
// Desabilita a interrupcao
        P1DIR |= 0x02;
// Pino do TRIGGER (P1.1) como saida
        P1OUT |= 0x02;
// Gera o pulso
        __delay_cycles(10);
// Delay de 10us
        P1OUT &= ~0x02;
// Para o pulso
        P1DIR &= ~0x04;
// Pino do ECHO (P1.2) como entrada
        P1IFG = 0x00;
// Limpa a flag, por precaução
        P1IE |= 0x04;
// Habilita a interrupção pelo pino do ECHO
        P1IES &= ~0x04;
// Define a borda de subida no pino ECHO

        __delay_cycles(30000);
// Delay de 30ms (depois desse tempo, o ECHO acaba
se nenhum objeto for detectado)
        distance = sensor / 58;
// Converte o valor de ECHO para cm

        P1DIR |= BIT6;
// P1.6/TA0.1 É USADO PARA O PWM, QUE FUNCIONA COMO
A SAÍDA
        P1OUT = 0;
// LIMPA AS SAIDAS P1
        P1SEL |= BIT6;
// P1.6 SELECIONA TA0.1

        P1DIR &= ~BIT3;
 // Botão
        P1REN |= BIT3;
        P1OUT |= BIT3;

 // O clock é de 1MHZ, sabendo que 1000ms = 1Hz =>
CCRO = 20000
// Temos que: 20000(1000000 / 1000 * 20) é igual a
um período de 20ms

        TA0CCR0 = 20000-1;
// PERIODO DO PWM TA0.1
        TA1CCR0 = 20000-1;
// PERIODO DO PWM TA1.1

        // SETANDO 2000 -> 0deg (Pós SERVO)
        TA0CCR1 = 2000;
// CCR1 PWM duty cycle
        TA1CCR1 = 2000;
// CCR1 PWM duty cycle

        TA0CCTL1 = OUTMOD_7;
// CCR1 reset/set
        TA0CTL   = TASSEL_2 + MC_1;
// SMCLK, up mode
        TA1CCTL1 = OUTMOD_7;
// CCR1 reset/set
        TA1CTL   = TASSEL_2 + MC_1;
// SMCLK, up mode

    if (distance > 30 && distance != 0 ) {
      P1OUT |= 0x01;
// Se a distância calculada for menor que 30 cm e
diferente de 0, o LED vermelho acende
      P2OUT &= ~0x01;
// led verde é apagado
    }
    else {
      P1OUT &= ~0x01;
// Caso contrario, o led vermelho permanece
desligado
      P2OUT |= 0x01;
// LED verde acende
    }

    if((P1IN&BIT3)==0)
        {
            for(k=0;k<4;k++){
// Repete 4 vezes (apresentação)
                aciona_motor();
                __delay_cycles(10000000);
// Abre a escotilha a cada 10s
            }
        }
  }
}

void aciona_motor(){
        delay(30000);
            TA0CCR1 = 1000;
            TA1CCR1 = 2000;

        delay(30000);
            TA0CCR1 = 2000;
            TA1CCR1 = 1000;
     }

#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void)
{
  if (P1IFG & 0x04)             // Verifica se tem
alguma interrupção pendente
  {
    if (!(P1IES & 0x04))        // Verifica se tem
uma borda de subida
    {
      TACTL |= TACLR;           // Se tem, limpa o
timer A
      time_ms = 0;
      P1IES |= 0x04;            // Borda de descida
    }
    else
    {
      sensor = (long)time_ms * 1000 + (long)TAR;
// Calcula o comprimento de ECHO
    }
    P1IFG &= ~0x04;
// Limpa a flag~0
  }
}
```

```
#pragma vector=TIMER0_A0_VECTOR
__interrupt void Timer_A (void)
{
  time_ms++;
}
```

APÊNDICE B - CÓDIGO EM ASSEMBLY SISTEMA DE ALARME

```
;-------------------------------------------------------
;  MSP430  Assembler  Code  Template  for  use  with  TI
Code Composer Studio
;
;
;-------------------------------------------------------
                .cdecls C,LIST,"msp430.h"         ;
Include device header file

;-------------------------------------------------------
            .def    RESET                   ; Export
program entry-point to
                                            ; make
it known to linker.
;-------------------------------------------------------
              .text                         ;
Assemble into program memory.
              .retain                       ;
Override ELF conditional linking
                                            ; and
retain current section.
             .retainrefs                    ; And
retain any sections that have
                                            ;
references to current section.
BUZZER      .set    BIT3
LED_R       .set    BIT1
LED_G       .set    BIT4
DelayLoops  .set    27000
A           .set    2
B           .set    6
SPACE       .set    2
Signal      .byte   A,B,A, B

;-------------------------------------------------------

RESET           mov.w    #__STACK_END,SP          ;
Initialize stackpointer
StopWDT      mov.w    #WDTPW|WDTHOLD,&WDTCTL   ; Stop
watchdog timer

; Main loop here
;-------------------------------------------------------

main:
    bis.b   #BUZZER, &P2OUT
    bis.b   #BUZZER, &P2DIR
    clr.w   R5
    bis.b   #LED_R, &P3OUT
    bis.b   #LED_R, &P3DIR
    jmp     AlertTest

AlertLoop:
    bic.b   #BUZZER,&P2OUT
    mov.b   Signal(R5),R12
    call    #LoopTest
    bis.b   #BUZZER,&P2OUT
    bis.b   #LED_R, &P3OUT
    mov.w   #SPACE,R12
    call    #LoopTest
    inc.w   R5

AlertTest:
    cmp.b   #0,Signal(R5)
    jne     AlertLoop
    bic.b   #LED_R, &P3DIR
    bic.b   #BUZZER,&P2OUT

    bis.b   #LED_G, &P7OUT
    bis.b   #LED_G, &P7DIR

OuterLoop:
    mov.w   #DelayLoops,R4

DelayLoop:
    dec.w   R4
    jnz     DelayLoop
    dec.w   R12
```