Desenvolvimento de um Oxímetro de Pulso Usando Microcontrolador MSP430

Davi de Alencar Mendes, João Paulo Sanches Guimarães

***Resumo*—No presente trabalho, propomos e demonstramos o desenvolvimento de um Oxímetro de Pulso usando o Microcontrolador MSP430FR2433 da *Texas Instruments*. A Saturação de Oxigênio SpO_2 e a Frequência Cardíaca são parâmetros chave para o monitoramento da saúde de pacientes. O sistema proposto consiste em um sensor de SpO_2, MSP430FR2433 e um *display*. A saturação do oxigênio é calculada a partir da razão entre duas intensidades de luz, já a frequência cardíaca é calculada a partir da diferença de tempo entre dois picos da intensidade do sinal infravermelho. Os parâmetros medidos são processados no microcontrolador e mostrados no *display*.**

***Index Terms*—Oxímetro de Pulso, SpO_2, Frequência Cardíaca, MSP430.**

I. Introdução

NAS últimas décadas observa-se uma crescente preocupação com assuntos relacionados a saúde. Em um contexto normal de monitoramento da saúde de pacientes, tem-se grandes restrições em mobilidade e usabilidade de tal maneira que soluções portáteis se tornam necessárias para diversos tipos de pacientes. O gás oxigênio é parte integrante dos processos biológicos que ocorrem no corpo humano. O transporte desse importante gás ocorre através das hemoglobinas nas células vermelhas do sangue. Informações críticas podem ser adquiridas por meio da medição da quantidade de oxigênio presente no sangue na forma de um índice percentual do total da capacidade máxima. O oxímetro de pulso é um instrumento que realiza tal medida [1].

O oxímetro de pulso inclui dois diodos emissores de luz (*LEDs*), um no espectro vermelho visível (660nm) e outro com espectro infravermelho (940nm) [2]. Mudanças na intensidade da luz transmitida pelos tecidos causadas pela pressão arterial sanguínea são detectadas como um sinal de voltagem pelo fotopletismógrafo (sensor SpO_2). No oxímetro apresentado será utilizado um sensor que adota o método de reflectância em sua operação, ou seja, há um emissor de luz ao lado de um fotodetector que mede a resposta após a emissão de luz. A Figura 1 mostra que há uma absorção constante de luz sempre presente devido aos diferentes tecidos presentes, sangue venoso e sangue arterial.

Davi de Alencar Mendes é estudante de Graduação em Engenharia Eletrônica pela Universidade de Brasília - UnB, Brasília, Brasil. Email institucional: dmendes@aluno.unb.br - Matrícula: 16/0026415

João Paulo Sanches Guimarães é estudante de Graduação em Engenharia Eletrônica pela Universidade de Brasília - UnB, Brasília, Brasil. Email institucional: sanches.joao@aluno.unb.br - Matrícula: 16/0031923

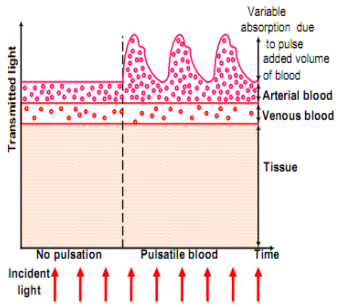

Figura 1. Luz transmitida em fluxo sanguíneo no tempo

Entretanto, a cada batimento cardíaco há um deslocamento de sangue arterial que provoca o aumento do volume de sangue que transita pelo espaço de medição do oxímetro, causando maior absorção de luz durante esse período caraterístico [1]. Se o sinal adquirido pelo fotodetector for analisado como um sinal de onda é possível observar que há picos a cada batimento e vales entre os batimentos. Com a luz absorvida em um vale (sinal que deve incluir todas as absorções constantes) for subtraída de uma amostra de pico é obtido o resultado do volume de sangue arterial trazido a cada batimento. Com esses sinais constantes e sua variação calcula-se um valor intermediário chamado R, a Razão Normalizada. Usando R, podemos calcular SpO2 usando a fórmula [1]:

$$SpO2 = 110 - 25 * R$$

O MSP430 incorpora uma CPU RISC de 16-bits, periféricos e um sistema de *clock* flexível que se interconecta usando uma arquitetura Von-Neumann com barramento comum de memória e dados. Com suporte para periféricos digitais e analógicos, o MSP430 oferece soluções para aplicações que usam diferentes sinais.

II. Desenvolvimento

A. Panorama do Protótipo Funcional

Para o ponto de controle 2 é esperado que seja obtido um protótipo funcional que atenda os principais requisitos do projeto. Para o Oxímetro de pulso é esperado que os seguintes requisitos funcionais sejam desenvolvidos:

- [RF01] O sistema deverá exibir em um *display* os dados obtidos.
- [RF02] O sistema deverá alertar o usuário com avisos sonoros.
- [RF03] O sistema deverá processar os dados obtidos do sensor.
- [RF04] O sistema deverá se comunicar com o sensor utilizando o protocolo I^2C (*Inter-Integrated Circuit*).

Visando alcançar estes requisitos, foi utilizada a plataforma de desenvolvimento Energia. Esta plataforma se assemelha à plataforma de desenvolvimento do Arduino, baseada em C++, proporcionando um desenvolvimento de alto nível em comparação ao C puro. Além disso, há bibliotecas prontas para alguns periféricos, as quais facilitaram a utilização de destes.

Como forma de Revisão bibliográfica foram pesquisadas bibliotecas de Arduino [3],[4] que implementam a comunicação e o processamento dos dados Obtidos do sensor MAX30100. Além disso, o *datasheet* do sensor[2] foi analisado para averiguar o bom funcionamento das bibliotecas que foram utilizadas para o desenvolvimento do protótipo funcional.

B. O Protótipo Funcional

Para organizar o desenvolvimento de um protótipo funcional que satisfaça os requisitos propostos para este projeto, este foi divido nos seguintes itens:

1) Descrição de Hardware: Para o seu funcionamento, o protótipo utilizará a seguinte lista de materiais:

- MSP430FR2433;
- Sensor MAX30100;
- *Display* LCD 16x2 - JHD 162A;
- *Buzzer* (buzina);

O microcontrolador MSP430FR2433 processará todos os dados e também integrará os periféricos envolvidos no sistema.

O Sensor MAX30100 é um componente fundamental ao protótipo, uma vez que este é responsável por obter e armazenar os dados obtidos pela fotopletismografia, medição feita a partir dos LEDS vermelho e infravermelho. Além desses dados, o sensor armazena em seus registradores a temperatura, parâmetro de extrema relevância para a calibração do sensor.

O *display* LCD será utilizado para transmitir a informação processada pela MSP para o usuário ou para a equipe médica.

O *buzzer* será utilizado como um alerta em situações de emergência que possam apresentar risco para o paciente, devido à baixa oxigenação sanguínea ou à elevada frequência cardíaca.

O arranjo desses componentes foi esboçado por meio do diagrama de blocos representado na Figura 2.

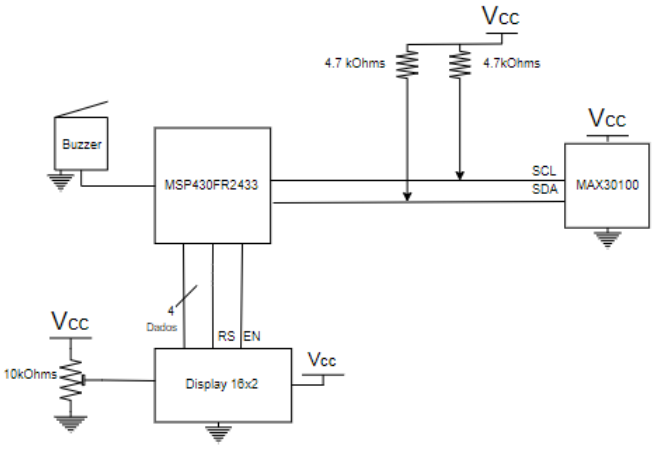

Figura 2. Diagrama de Blocos do Protótipo

Mais detalhadamente, as ligações entre o o MSP e o Sensor MAX30100 foram efetuadas de acordo com o esquemático a seguir:

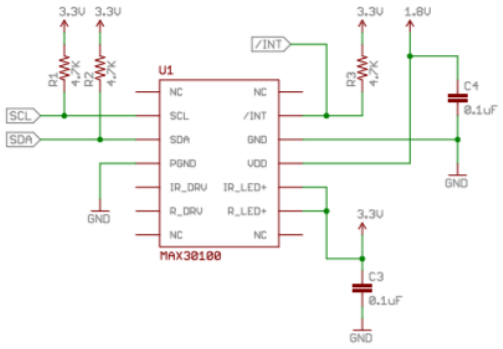

Figura 3. Esquemático de conexões do Sensor MAX3010

Para a versão atual do protótipo funcional vale notar que os pinos: *INT* - interrupção, *IR_DV* e *R_DV* - cátodo dos LEDs não estão sendo utilizados, pois não há necessidade de utilizar nenhuma dessas funções. É interessante notar que foram utilizados resistores de *pull-up* externos nos pinos *SDA* e *SCL* para estabelecer a comunicação I^2C (Inter-Integrated Circuit) com o sensor.

2) Descrição de Software: Para o Ponto de Controle 2, é permitido utilizar o Energia e bibliotecas que já implementam funções em alto nível. Para a implementação do Display foi utilizada a biblioteca *LiquidCrystal.h* que já implementa a escrita no display 16x2 por meio da função *print()*. Para a implementação do *Buzzer* não é necessário utilizar bibliotecas próprias. A implementação do Sensor MAX30100 por ser mais complexa contou com uma etapa de teste utilizando um *Arduino* e as bibliotecas pesquisadas anteriormente[3][4]. Finalmente, a implementação da comunicação I^2C foi realizada através da biblioteca *Wire.h* para simplificar a utilização desse protocolo.

Os códigos utilizados encontram-se no apêndice deste documento. Os resultados obtidos a partir dos testes executados revelaram problemas de compatibilidade de software entre as bibliotecas pesquisas e o ambiente de desenvolvimento Energia. Para uma melhor discussão de tais problemas os resultados encontrados durante o desenvolvimento do protótipo funcional compõe a próxima seção do presente trabalho.

III. Resultados

A implementação do requisito funcional [RF01] - O sistema deverá exibir em um *display* os dados obtidos foi realizada através do uso de uma biblioteca conforme foi descrito na descrição de software do presente trabalho. Com a implementação do protótipo funcional foi possível atender o requisito funcional [RF02] - O sistema deverá alertar o usuário com avisos sonoros. Esses requisitos compõe a [NE03] que envolve informar o usuário por meio do *display* e avisos sonoros.

Inicialmente, foram testadas as bibliotecas do Sensor MAX30100 [3],[4] em um Arduino para comprovar suas

funcionalidades. Os testes demonstraram que as bibliotecas juntamente com o *hardware* utilizado apresentavam bom funcionamento, sendo capazes de adquirir e processar os dados para exibir os valores de SpO_2 e frequência cardíaca. Após os testes, foram realizadas tentativas para utilizar as bibliotecas no ambiente Energia que resultaram em problemas de compatibilidade de software. Um dos problemas detectados foi a ausência da função *Wire.setClock()*, função que configura a velocidade do sinal de *clock* do pino *SCL - Serial Clock Line* na biblioteca *Wire.h* do Energia. Sem essa função não é possível alterar a velocidade do sinal de *clock* de 100kHz (padrão) para 400kHz que é a velocidade de operação do MAX30100. Dessa maneira, não é possível estabelecer uma comunicação com o Sensor, uma vez que o protocolo de comunicação que este opera (I^2C) é altamente dependente do sinal de *clock*.

Para testar o funcionamento do protocolo de comunicação foi realizado um teste que realiza a leitura em um registrador específico que guarda o *Part ID* do sensor. O valor lido no registrador é comparado com o valor esperado para o *Part ID* do sensor e assim é possível avaliar se o protocolo está funcionando corretamente. O resultado experimental revelou um problema ao ler os valores do registador. O código para o teste realizado se encontra no apêndice deste trabalho. Com esse resultado, não foi possível cumprir o requisito funcional [RF04] que envolve o protocolo de comunicação. Consequentemente, o requisito funcional [RF03] que envolve o processamento dos dados obtidos não pode ser implementado já que depende do funcionamento do protocolo I^2C.

IV. Conclusão

Ao contemplar o material exposto neste documento, é possível notar que as dificuldades encontradas na implementação do requisito funcional [RF04] que envolve a comunicação com o sensor prejudicaram o desenvolvimento do projeto para o ponto de controle 2. No entanto, é possível adequar o cronograma atual para atender as necessidades [NE01] e [NE02] que envolvem aferir o nível de SpO_2 e frequência cardíaca.

Quanto ao desenvolvimento geral do projeto, é possível notar que não houve progresso significativo no objetivo [O1] - Monitorar a SpO_2 presente no sangue de pacientes que sofram de distúrbios pulmonares. E para o objetivo [O2] - Alertar o usuário ou a equipe médica quando o nível de saturação de oxigênio sanguíneo for inferior a 95% já é possível mensurar algum progresso, uma vez que o *display* e os avisos sonoros já estão operacionais. No entanto, o objetivo [O1] é o mais relevante para o sucesso do projeto como uma solução viável para o problema que se propõe a solucionar.

Referências

[1] Alexander, Christian M., Lynn E. Teller, and Jeffrey B. Gross. "Principles of pulse oximetry: theoretical and practical considerations."Anesthesia & Analgesia 68.3 (1989): 368-376.

[2] "MAX30100 Pulse Oximeter and Heart-Rate Sensor IC for Wearable Health - Maxim." Pulse Oximeter and Heart-Rate Sensor IC for Wearable Health, MAXIM Integrated, www.maximintegrated.com/en/products/sensors-and-sensor-interface/MAX30100.html.

[3] Intersecans, OXullo. Arduino-MAX30100. n.d. https://github.com/oxullo/Arduino-MAX30100

[4] Strogonovs, Raivis. Implementing Pulse Oximeter Using MAX30100. 2018, https://morf.lv/files//max30100.pdf. Accessed 1 May 2018.

APÊNDICE

A. Código para Testes do Display e do Buzzer

```
/* inclui biblioteca para operar o display */
#include <LiquidCrystal.h>

/* inicia a biblioteca com a pinagem adequada ao hardware */
const int rs = P2_0, en = P2_1, d4 = P2_2, d5 = P2_6, d6 = P2_4, d7 = P2_5;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

float seno;
int frequencia;

void setup() {
  /* set up do numero de colunas e linhas do display */
  lcd.begin(16, 2);
  /* Escreve uma mensagem no display. */
  lcd.print("SpO2:    Bpm:   ");
  pinMode(P2_7, OUTPUT);
}

void loop() {
  /* Desliga o display: */
  lcd.noDisplay();
  tone(P2_7, 0); // para o som
  delay(500);
   /* Liga o display: */
  lcd.display();
  for (int x = 0; x < 180; x++){ /* altera a frequencia emitida pela buzina */
    seno = sin(x * 3.1416/180);
    frequencia = 150 + int(seno * 50);
    tone(P2_7, frequencia);
  }
  delay(500);
}
 }
```

B. Código para Testes do Sensor MAX30100

```
//Objetivo: Testar o protocolo I2C com o Sensor MAX30100 ao ler o PART_ID no registrador 0xff
#include <Wire.h>
#include "MAX30100_Registers.h"

#define EXPECTED_PART_ID            0x11

void setup()
{
    Serial.begin(115200);
    Serial.print("Inicializando Sensor... \n");
    if(begin() == false)
      Serial.print("PART_ID NAO foi encontrado\n");
    else
      Serial.print("PART_ID lido com sucesso!\n");
}

void loop()
{
  Serial.print("LOOP. \n");
  for(;;);
}

//Objetivo: Realizar leitura do PART_ID do sensor.
//Retorno: Valor logico indicando sucesso (true) ou falha na leitura (false).
bool begin()
{
  Wire.begin();
  Wire.setModule(0);

  if (getPartId() != EXPECTED_PART_ID)
  {
    return false;
  }
  return true;
}

//Objetivo: Ler um valor em um registrador usando o protocolo I2C.
//Entrada: Endereco para leitura.
//Retorno: Valor lido no registrador
uint8_t readRegister(uint8_t address)
{
    Wire.beginTransmission(MAX30100_I2C_ADDRESS);
    Serial.print("Iniciando Transmissao ... \n");
    Wire.write(address);
    Serial.print("Write no address... \n");
    Wire.endTransmission(false);
    Serial.print("END Transmissao...\n");
    Wire.requestFrom(MAX30100_I2C_ADDRESS, 1);
    Serial.print("End Request... \n");
    return Wire.read();
}

uint8_t getPartId()
{
    return readRegister(0xff);
}
```