Sobre o Sensor

O sensor SpO2 MAX30102 apresenta 2 funções principais, sendo estas a medição dos batimentos cardíacos e a medição indireta (não invasiva) da quantidade de oxigênio no sangue, ambos exibidos no display.

Sobre a biblioteca

Esta biblioteca utiliza de outras bibliotecas necessárias para as medições dos dados pelo sensor e para a exibição das mesmas no display. Para o correto funcionamento desta biblioteca são utilizadas as seguintes bibliotecas:

- **time.h:** utilizada para a implementação de *delays* no funcionamento do módulo.
- **Wire.h:** biblioteca responsável por conter as funções necessárias para gerenciar a comunicação entre os dispositivos através do protocolo I2C.
- **MAX30105.h:** biblioteca responsável por configurar o sensor MAX30102.
    - **OBS:** Esta biblioteca é funcional tanto para o MAX30105 quanto para o MAX30102.
- **spo2_algorithm.h:** biblioteca responsável por calcular os batimentos cardíacos e a oxigenação do sangue.
- **lcd.h:** biblioteca responsável por configurar o lcd para que o mesmo mostre os dados recebidos do módulo max30102.

Configurações

Clock

O *clock* é configurado em 16Mhz por padrão na biblioteca “MAX30102_MODULE.h”. Caso seu projeto utilize outra frequência de *clock*, utiliza-se a função **setClock** para alterá-la.

```
#define F_CPU 16000000UL

void MAX30102_Module::setClock(unsigned long clock){
    #define F_CPU clock;
}
```

Sensor

O sensor está configurado para funcionar integrado ao módulo, não necessitando de nenhuma configuração manual. Para realizar a configuração, utiliza-se a função **initialize_module(void)** da biblioteca “MAX30102_MODULE.h”, após isto, o módulo já estará pronto para iniciar as leituras.

```
MAX30105 particleSensor;

// Initialize sensor
bool MAX30102_Module::initialize_module(){

    lcd.LCD_Initializer(); //Inicializa o lcd
    lcd.LCD_String("Inicializando...");
    _delay_ms(1000);

    byte ledBrightness = 60; //Options: 0=Off to 255=50mA
    byte sampleAverage = 4; //Options: 1, 2, 4, 8, 16, 32
    byte ledMode = 2; //Options: 1 = Red only, 2 = Red + IR, 3 = Red + IR + Green
    byte sampleRate = 100; //Options: 50, 100, 200, 400, 800, 1000, 1600, 3200
    int pulseWidth = 411; //Options: 69, 118, 215, 411
    int adcRange = 4096; //Options: 2048, 4096, 8192, 16384

    //A linha a seguir, aplica as configurações
    particleSensor.setup(ledBrightness, sampleAverage, ledMode,
                         sampleRate, pulseWidth, adcRange);

    lcd.LCD_Clear();
    lcd.LCD_String("Inicializado!");
    _delay_ms(400);
    return true;
}
```

Display LM016L 16x2

Todas as bibliotecas necessárias para uso desse display estão incluídas, a configuração é realizada pela função interna **LCD_Initializer()** da biblioteca “lcd.h”, que é executada dentro da função **initialize_module(void)**.

```
void LCD::LCD_Initializer(void)
{
  LCD_Dir_00 = 0xFF;
  _delay_ms(20);

  LCD_Commandgiver(0x02);
  LCD_Commandgiver(0x28);
  LCD_Commandgiver(0x0c);
  LCD_Commandgiver(0x06);
  LCD_Commandgiver(0x01);
  _delay_ms(2);
}
```

Funções

- **void setClock(unsigned long Clock) -** Aplica uma nova configuração de Clock. Atenção: A configuração default é de 16MHz.

  código dessa função é mostrado na seção anterior

- **bool initialize_module() -** Configurar e inicializar o display e o sensor. Retorna **True** se tiver inicializado com sucesso ou **False**, caso contrário.

  código dessa função é mostrado na seção anterior

- **bool hasNewData() -** Verifica se existem novos dados no armazenamento do sensor. Retorna **True** se houver novos dados, ou **False** se não houver.

```cpp
bool MAX30102_Module::hasNewData(){
    if (particleSensor.available() == true){
        particleSensor.check();
        return true;
    }
    return false;
}
```

- **void readData() -** Verifica se existem dados no sensor, quando houver, realiza a leitura destes dados no armazenamento do sensor e exibe no display.

```cpp
void MAX30102_Module::readData(){
    lcd.LCD_Clear();
    while (hasNewData() == false){ //Aguardando por novos dados
        //particleSensor.check(); //Checa os dados no armazenamento do sensor;
        lcd.LCD_String(("Encoste o dedo..."));
        delay(100);
    }

    if (particleSensor.available() == true) {
        for (byte i = 0 ; i < 100 ; i++){

            redBuffer[i] = particleSensor.getRed();
            irBuffer[i] = particleSensor.getIR();
            particleSensor.nextSample(); //We're finished with this sample so move to next sample
            lcd.LCD_Clear();
            lcd.LCD_String(("red="));
            lcd.LCD_String((char *)redBuffer[i]);
            lcd.LCD_String((", ir="));
            lcd.LCD_String((char *)irBuffer[i]);
            _delay_ms(50);
        }
    }
}
```