Sistemas Microprocessados - 2019.2
DETI/UFC - Cursos de Eng. de Computação e Eng. de Telecomunicações
Elaborada por Danilo Coutinho, Jorge Reis e Jardel Silveira

PRÁTICA OLed, SPI e Conversão AD

Dentre os métodos de comunicação serial mais conhecidos, destacam-se três:

- UART: Universal Asynchronous Receiver Transmitter;
- SPI: Serial Peripheral Interface;
- I2C: Inter Integrated Circuit.

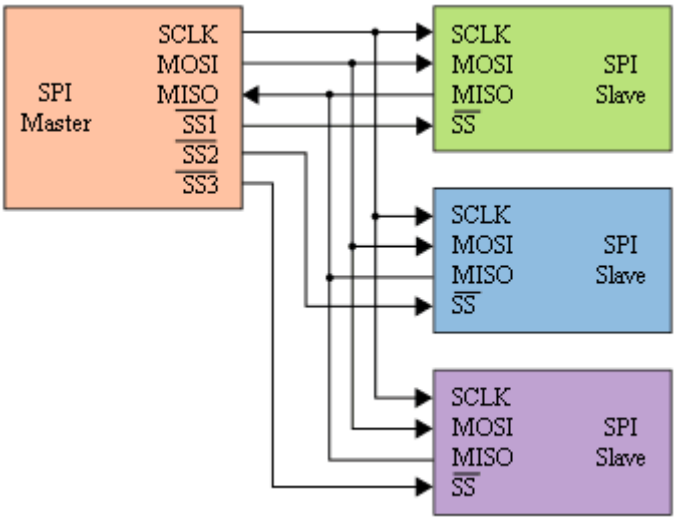
SPI é a abreviação do Inglês Serial Peripheral Interface, ou em bom português, Protocolo de Interface Periférica Serial. Este protocolo é uma tecnologia de comunicação serial síncrona de dados realizado com dispositivos periféricos, de forma rápida e em tempo real.

O SPI opera em **modo full duplex** , isto significa que os dados são transferidos em ambas as direções e ao mesmo tempo e isso faz com que sua velocidade de troca de dados seja bem mais rápida, superior a 10 Mhz em comparação com outros sistemas.

Na comunicação serial síncrona definimos o conceito de Mestre-Escravo. Normalmente o gerador do sinal de sincronismo é definido como o Mestre (Master) da comunicação. Para os dispositivos que utilizam do sinal de sincronismo gerado damos a definição de Escravo (Slave). A ligação mais comum desse tipo de comunicação é um Master e vários Slaves.

Os pinos básicos de comunicação entre dispositivos SPI e o esquema de ligação são os seguintes:

Pino	Nome Padrão	Significado	Nomes Alternativos
Do Master para o Slave	MOSI	Master Output Slave Input	SDO, DO, SO
Do Slave para o Master	MISO	Master Input Slave Output	SDI, DI, SI
Clock	SCLK	Serial Clock	SCK, CLK
Seleção de Slave	SS	Slave Select	CS, nSS, nCS



Mais informações sobre o protocolo SPI em:

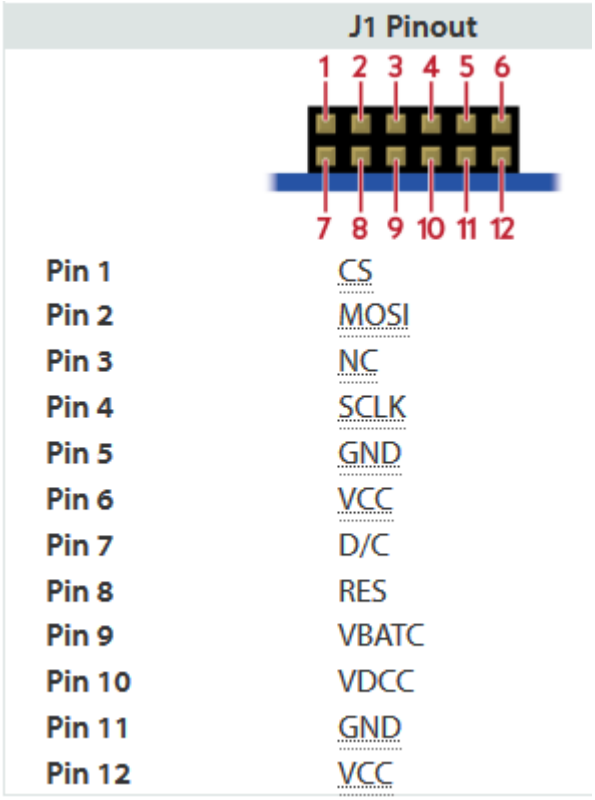
https://www.embarcados.com.br/spi-parte-1/

https://www.embarcados.com.br/comunicacao-spi-parte-2/

Prática

1. Nesta prática utilizaremos a Greenpill e PMOD OLED, que é uma tela de led 128x32 que utiliza comunicação SPI, como um Voltímetro.Para isso, utilizaremos também nosso conversor AD interno para fazer a medição. O AD retorna valores de 0 a 4036 e devemos parametrizar esses valores para a voltagem que se deseja medir. No caso, faremos 0=0 e 3,3=4095, os valores parametrizados foram colocados numa tabela e sua utilização será explicada abaixo.

1- Dado o pinout do OLED, conecte-o na bluepill como na tabela abaixo.



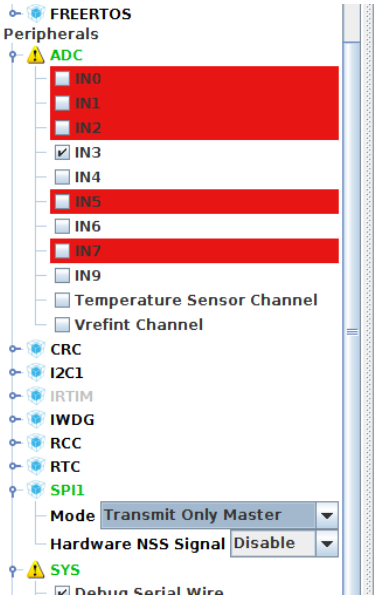
Pinout do PMOD OLED

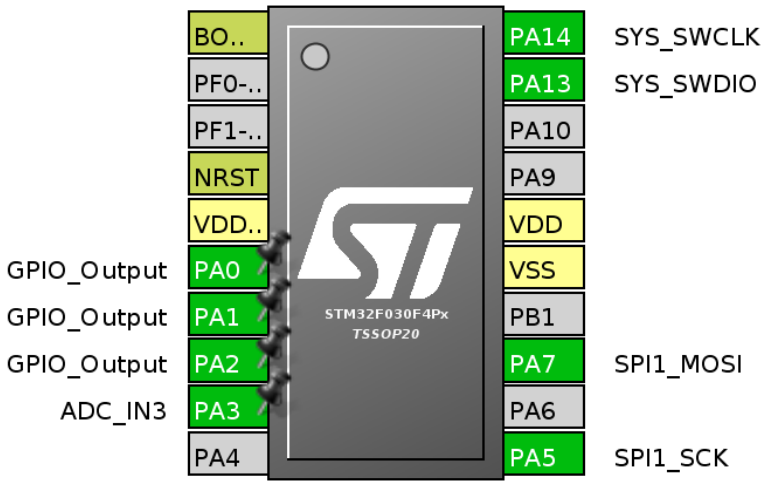
Tabela 1:Conexões entre OLed PMOD e Bluepill

OLed PMOD	Bluepill
1 (CS)	PA0
2 (MOSI)	PA7

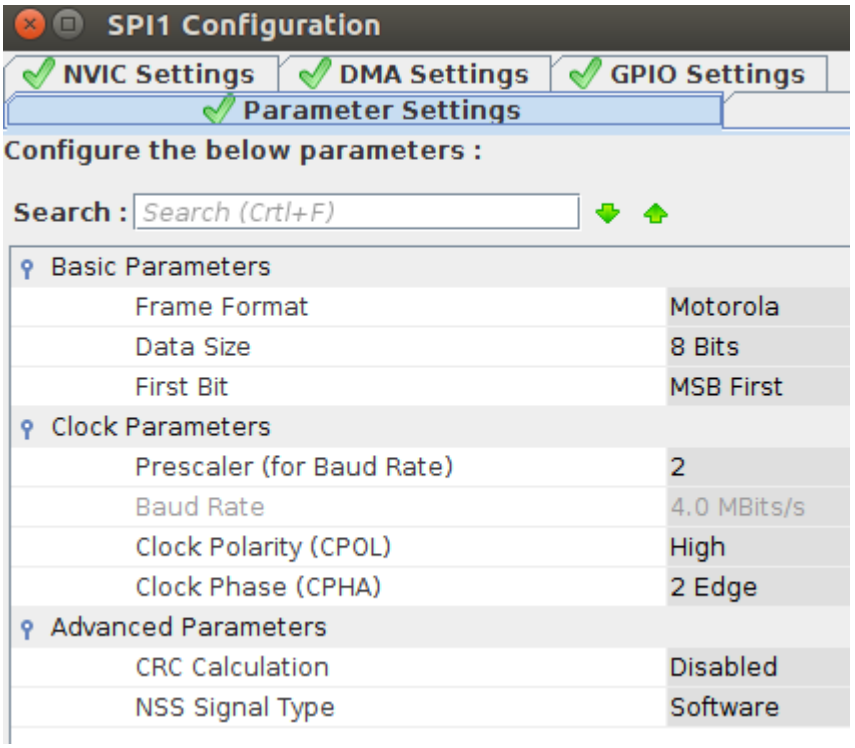
3 (NC)	NC
4 (SCLK)	PA5
5 (GND)	GND
6 (VCC - 3.3V)	VCC
7 (DATA/COMMAND)	PA1
8 RES	PA2
9 VBATC	GND
10 VDDC	GND
11 GND	NC
12 VCC	NC

2- Crie um novo projeto no STM32CUBE(lembrando que a Bluepill é a stm32F103C8T6) e configure os pinos como ilustrado nas figuras abaixo:





3- Na aba Configuration>Middlewares>Connectivity clique em SPI1 e configure como descrito na imagem abaixo.



- **Formato de quadro** - tipo de quadro de dados. Praticamente é sempre um formato Motorola.
- **Tamanho dos dados** - a largura do quadro de dados. O SPI permite a transmissão de quadros de 8 ou 16 bits. Nós deixamos o valor padrão de *8 bits* .
- **Primeiro bit** - Esta é a página a partir da qual a transmissão do byte começa. MSB *(bit mais significativo)* - do bit mais significativo, ou seja, do bit [7]. LSB *(bit menos significativo)* , bit [0]. Como sabemos da documentação, o driver de exibição de dados **deve ser enviado no modo MSB.**
- **Prescaler** - relógio divisor SPI relógio.
- **Baud Rate** - taxa de bits por segundo.
- **Polaridade do Relógio (CPOL)** - descrita no início do artigo.
- **Fase do Relógio (CPHA)** - descrita no início deste artigo.

Importe os arquivos .h para a pasta inc do projeto, e os .c para a pasta sources.

Funções que serão utilizadas:
void OledInit() - Inicializa o oLed
void OledClear() - Apaga todos os dados do oLed
HAL_ADC_Start(&hadc)- Inicializa o adc.
HAL_Delay(valor)- Insere um delay.
OledSetCursor(coluna,linha)- Coloca o cursor na posição desejada da tela. Cacada caractere ocupa uma coluna.
 OledPutString(“string”)- Escreve uma string na tela.
OledPutChar(“char”)- Escreve um char na tela.
HAL_ADC_GetValue(&hadc)- Lê o valor do AD

Dada a explicação faça o código que lerá o valor medido pelo AD interno da Bluepill e exibirá o valor no OLED.

Coloque um breakpoint em qualquer linha da biblioteca do LCD que faça uso da função de transmitir um dado (ou comando) pela SPI. Use 2 canais osciloscópio, um para clock e outro para os dados, para visualizar o dado transmitido pela SPI. O Dado está correto ? Qual a frequência de operação ?

Use a fonte de tensão da sua bancada para validar o voltímetro que você desenvolveu. Antes, tome o cuidado de limitar a corrente de curto da sua fonte a 30mA. OBSERVAÇÃO: NÃO ULTRAPASSE O VALOR DE 3.3 !!!!!

```
/* USER CODE BEGIN Includes */
#include "OledGrph.h"
#include "OledDriver.h"

/* USER CODE END Includes */
```

```
  /* USER CODE BEGIN 2 */

  OledInit(); // Inicializa o oLed
  OledClear(); //- Apaga todos os dados  do oLed
  //OledSetCursor(0,0); //- Coloca o cursor na posição desejada da tela. Cacada caractere ocupa uma coluna.
  OledPutString("string"); //Escreve uma string na tela.
  OledPutChar('c'); //- Escreve um char na tela.

  /* USER CODE END 2 */



#include <stdio.h>
#include <math.h>

// reverses a string 'str' of length 'len'
void reverse(char *str, int len)
{
        int i=0, j=len-1, temp;
        while (i<j)
        {
        temp = str[i];
        str[i] = str[j];
        str[j] = temp;
        i++; j--;
        }
}

 // Converts a given integer x to string str[].  d is the number
 // of digits required in output. If d is more than the number
 // of digits in x, then 0s are added at the beginning.
int intToStr(int x, char str[], int d)
```

```
{
        int i = 0;
        while (x)
        {
        str[i++] = (x%10) + '0';
        x = x/10;
        }

        // If number of digits required is more, then
        // add 0s at the beginning
        while (i < d)
        str[i++] = '0';

        reverse(str, i);
        str[i] = '\0';
        return i;
}

// Converts a floating point number to string.
void ftoa(float n, char *res)
{
        // Extract integer part
        int ipart = (int)n;

        // Extract floating part
        float fpart = n - (float)ipart;

        // convert integer part to string
        int i = intToStr(ipart, res,0);

        // check for display option after point
        //if (afterpoint != 0)
        //{
        res[i] = '.';  // add dot
```

```
        // Get the value of fraction part upto given no.
        // of points after dot. The third parameter is needed
        // to handle cases like 233.007
        fpart = fpart * 100.0 ; // pow(10, afterpoint);

        intToStr((int)fpart, res + i + 1,2);
        //}
}
```