Aula 2 - estrutura hlcd e escrita em blocos

Revisão da aula 1

- Dependências do hardware são a SPI, os pinos
- para generalizar a SPI, foi feito um ponteiro que apontasse pra estrutura de dados da SPI selecionada, para que esse ponteiro passe os dados da SPI pras funções
- A ideia agora é fazer a mesma coisa com os pinos IO

Pinos IO

- Pinos
  - **Clock** e **MO** são informados pela SPI, então não precisa se preocupar com eles
  - **CS** é usado quando há dois ou mais dispositivos no barramento, mas ele vai ser consderado aqui
  - **DS** e **RST** são variáveis
- Para generalizar, é preciso do valor do pino e da porta que ele está!
  - ```
    GPIO_TypeDef    *Port_DC;
    uint16_t         Pin_DC;
    GPIO_TypeDef    *Port_CS;
    uint16_t         Pin_CS;
    ```
  - Para facilitar o uso, vamo por tudo isso numa estrutura de dados

Estrutura LCD

- foi feita uma estrutura para manipular o LCD, permitindo configurar os pinos de forma geral.
  - ```
    typedef struct
    {
        SPI_HandleTypeDef *hspi;        // Handler pra SPI usada

        GPIO_TypeDef    *CS_Port;       // Porta do CS
        uint16_t        CS_Pin;         // Pino do CS

        GPIO_TypeDef    *DC_Port;       // Porta do DC
        uint16_t        DC_Pin;         // Pino do DC
    }LCD_HandleTypeDef;
    ```
- Dessa forma, a estrutura é preenchida na main pelo usuário de acordo com sua necessidade. A estrutura então é passada para a LCD init que vai executar o resto
  - Preenchimento da estrutura:

```
// --- Configurações do hardware do display ---
    hlcd.hspi = &hspi2;

    hlcd.CS_Port = NK_CS_GPIO_Port;
    hlcd.CS_Pin = NK_CS_Pin;

    hlcd.DC_Port = NK_DC_GPIO_Port;
    hlcd.DC_Pin = NK_DC_Pin;
```

Enviar um bloco de dados

- A ideia é enviar um bloco todo de uma vez só ao invés de mandar o CS ficar subindo e descendo a cada transmissão de byte.
- Há também a vantagem de ser simples de implementar transmissões por DMA ou por interrupções.

Funções modificadas

Essas funções devem ser adcionadas ao LCD.h

Depois de testadas, as antigas devem ser apagadas

- `void LCD5110_LCD_write(uint8_t *data, uint16_t tam, uint8_t mode)`
- `void LCD5110_drawchar(char c, uint8_t *dat)`
- `void LCD5110_new_write_char(unsigned char c)`
- `void LCD5110_new_write_string(char *s)`

Nota

essas funções foram corrigidas no início da aula 3!