

Computação embarcada 2023-1

LAB 7 – LCD – LVGL

CÓDIGO BASE — [SAME70-examples/Screens/RTOS-TFT-LCD-ILI9341-LVGL](#)



1. Plugue o LCD max Touch no **EXT2** seguindo as instruções em: [SAME70-examples/Screens/2.8-TFT-LCD-ILI9341-Hardware/](#)

➔ View ➔ Terminal Window

Configure o terminal para a porta que (COM) correta (verificar no windiows) e para operar com um BaudRate de 115200.

INCORPORANDO EM NOVO PROJETO

<https://docs.lvgl.io/latest/en/html/porting/index.html>

Porting lvgl

O [site do LVGL](#) descreve o passo a passo de como incorporar a biblioteca em um novo projeto/ microcontrolador, além de criar um arquivo de configuração **lv_conf.h** com as propriedades do LCD e com configurações de como a biblioteca irá trabalhar, temos que implementar duas funções:

- ✦ **my_flush_cb**: Função chamada pelo lvgl sempre que necessita atualizar a tela
 - ✦ Depende do driver ili9341
- ✦ **my_input_read**: Função chamada pelo lvgl sempre que quer ler uma informação de touch
 - ✦ Depende do driver do touch resistivo

my_flush_cb

Lab 3 - PIO - IRQ

Lab 4 - RTOS

Lab 5 - RTOS HC-SR04

Lab 6 - RTOS IMU

Lab 7 - RTOS - LCD - LVGL

Começando

Começando

Início

LCD

LVGL

FRAME BUFFER

No LVGL devemos declarar esse frame buffer como global ou estático:

```
/*Static or global buffer(s). The second buffer is optional*/  
static lv_color_t buf_1[LV_HOR_RES_MAX * LV_VER_RES_MAX];
```

O LVGL irá então manter uma cópia do que será exibido no LCD nesta região continua de memória e de tempos em tempos atualizar o LCD chamando a função ***my_flush_cb***.

CÓDIGO JÁ COM INICIALIZAÇÃO

```
/* **** */
int main(void) {
    /* board and sys init */
    board_init();
    sysclk_init();
    configure_console();

    /* Lcd, touch and lvgl init*/
    configure_lcd();
    configure_touch();
    configure_lvgl();

    /* Create task to control oled */
    if (xTaskCreate(task_lcd, "LCD", TASK_LCD_STACK_SIZE, NULL, TASK_LCD_STACK_PRIORITY, NULL) != pdPASS) {
        printf("Failed to create lcd task\r\n");
    }

    /* Start the scheduler. */
}
```

```
void configure_lvgl(void) {
    lv_init();
    lv_disp_draw_buf_init(&disp_buf, buf_1, NULL, LV_HOR_RES_MAX * LV_VER_RES_MAX);

    lv_disp_drv_init(&disp_drv);          /*Basic initialization*/
    disp_drv.draw_buf = &disp_buf;        /*Set an initialized buffer*/
    disp_drv.flush_cb = my_flush_cb;      /*Set a flush callback to draw to the display*/
    disp_drv.hor_res = LV_HOR_RES_MAX;    /*Set the horizontal resolution in pixels*/
    disp_drv.ver_res = LV_VER_RES_MAX;     /*Set the vertical resolution in pixels*/

    lv_disp_t * disp;
    disp = lv_disp_drv_register(&disp_drv); /*Register the driver and save the created display objects*/




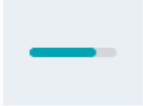
    /* Init input on LVGL */
    lv_indev_drv_init(&indev_drv);
    indev_drv.type = LV_INDEV_TYPE_POINTER;
    indev_drv.read_cb = my_input_read;
    lv_indev_t * my_indev = lv_indev_drv_register(&indev_drv);
}
```

CRIANDO UMA TELA

Note que na **task_lcd** chamamos a função **lv_ex_btn_1()** esta função cria os widgets no LCD, os widgets estão listados na página:

✦ <https://docs.lvgl.io/master/widgets/index.html>

O exemplo fornecido cria dois tipos de botões diferentes um do push button e outro do tipo toggle, notem que tudo é realizado pela API do lvgl. Podemos associar um handler (**event_handler**) ao botão, este handler sera chamado (pela **lv_task_handler**) sempre que acontecer um evento neste widget (botão apertado, botão liberado, ...).

Widget	Example
Button (lv_btn)	
LED (lv_led)	
Roller (lv_roller)	
Bar (lv_bar)	

INICIANDO AS ATIVIDADES DE CRIAR NOSSA TELA

<https://docs.lvgl.io/8/>



1 – NOVA FUNÇÃO DENTRO DA TASK

```
+void lv_thermostato(void) {  
+    lv_obj_t * labelBtn1;  
  
+    lv_obj_t * btn1 = lv_btn_create(lv_scr_act());  
+    lv_obj_add_event_cb(btn1, event_handler, LV_EVENT_ALL, NULL);  
+    lv_obj_align(btn1, LV_ALIGN_CENTER, 0, -40);  
  
+    labelBtn1 = lv_label_create(btn1);  
+    lv_label_set_text(labelBtn1, "Teste");  
+    lv_obj_center(labelBtn1);  
+  
+}
```

```
static void task_lcd(void *pvParameters) {  
  
-    lv_ex_btn_1();  
+    lv_thermostato();  
  
    ....  
    ....  
}
```


2 – MODIFIQUE O BACKGROUND – em *config/lv_conf.h*

1. Modifique o define ***LV_THEME_DEFAULT_DARK*** de ***0*** para ***1***
2. Teste no uC

Agora deve estar mostrando um fundo preto.

3 – FONTES – em *config/lv_conf.h*

```
289  /*Montserrat fonts with ASCII range and some symbols using b
290   *https://fonts.google.com/specimen/Montserrat*/
291  #define LV_FONT_MONTSEERRAT_8   0
292  #define LV_FONT_MONTSEERRAT_10  0
293  #define LV_FONT_MONTSEERRAT_12  0
294  #define LV_FONT_MONTSEERRAT_14  1
295  #define LV_FONT_MONTSEERRAT_16  0
296  #define LV_FONT_MONTSEERRAT_18  0
297  #define LV_FONT_MONTSEERRAT_20  0
298  #define LV_FONT_MONTSEERRAT_22  0
299  #define LV_FONT_MONTSEERRAT_24  0
300  ...
```

3 – FONTES – em *config/lv_conf.h*

Modifique o arquivo ***lv_conf.h*** para:

1. Incluir a fonte tamanho 24 no projeto

```
#define LV_FONT_MONTERRAT_12    0
#define LV_FONT_MONTERRAT_14    0
...
+#define LV_FONT_MONTERRAT_24    1
```

2. Torne a fonte tamanho 24 padrão

```
/*Always set a default font*/
+#define LV_FONT_DEFAULT &lv_font_montserrat_24
```


























































4 – HANDLERS

- ♦ **LV_EVENT_PRESSED** An object has been pressed
- ♦ **LV_EVENT_PRESSING** An object is being pressed (called continuously while pressing)
- ♦ **LV_EVENT_PRESS_LOST** An object is still being pressed but slid cursor/finger off of the object
- ♦ **LV_EVENT_SHORT_CLICKED** An object was pressed for a short period of time, then released. Not called if scrolled.
- ♦ **LV_EVENT_LONG_PRESSED** An object has been pressed for at least the long_press_time specified in the input device driver. Not called if scrolled.
- ♦ A lista completa pode ser acessada na página de Events:
<https://docs.lvgl.io/master/overview/event.html>

```
lv_obj_t * btn1 = lv_btn_create(lv_scr_act());  
lv_obj_add_event_cb(btn1, event_handler, LV_EVENT_ALL, NULL);  
lv_obj_align(btn1, LV_ALIGN_CENTER, 0, -40);
```

5 – LABELS

```
label = lv_label_create(btn1);  
lv_label_set_text(label, "Teste");  
lv_obj_center(label);
```

	LV_SYMBOL_AUDIO		LV_SYMBOL_WARNING
	LV_SYMBOL_VIDEO		LV_SYMBOL_SHUFFLE
	LV_SYMBOL_LIST		LV_SYMBOL_UP
	LV_SYMBOL_OK		LV_SYMBOL_DOWN
	LV_SYMBOL_CLOSE		LV_SYMBOL_LOOP
	LV_SYMBOL_POWER		LV_SYMBOL_DIRECTORY
	LV_SYMBOL_SETTINGS		LV_SYMBOL_UPLOAD
	LV_SYMBOL_TRASH		LV_SYMBOL_CALL
	LV_SYMBOL_HOME		LV_SYMBOL_CUT
	LV_SYMBOL_DOWNLOAD		LV_SYMBOL_COPY
	LV_SYMBOL_DRIVE		LV_SYMBOL_SAVE
	LV_SYMBOL_REFRESH		LV_SYMBOL_CHARGE
	LV_SYMBOL_MUTE		LV_SYMBOL_PASTE
	LV_SYMBOL_VOLUME_MID		LV_SYMBOL_BELL
	LV_SYMBOL_VOLUME_MAX		LV_SYMBOL_KEYBOARD
	LV_SYMBOL_IMAGE		LV_SYMBOL_GPS
	LV_SYMBOL_EDIT		LV_SYMBOL_FILE
	LV_SYMBOL_PREV		LV_SYMBOL_WIFI
	LV_SYMBOL_PLAY		LV_SYMBOL_BATTERY_FULL
	LV_SYMBOL_PAUSE		LV_SYMBOL_BATTERY_3
	LV_SYMBOL_STOP		LV_SYMBOL_BATTERY_2
	LV_SYMBOL_NEXT		LV_SYMBOL_BATTERY_1
	LV_SYMBOL_EJECT		LV_SYMBOL_BATTERY_EMPTY
	LV_SYMBOL_LEFT		LV_SYMBOL_USB
	LV_SYMBOL_RIGHT		LV_SYMBOL_BLUETOOTH
	LV_SYMBOL_PLUS		LV_SYMBOL_BACKSPACE
	LV_SYMBOL_MINUS		LV_SYMBOL_SD_CARD
	LV_SYMBOL_EYE_OPEN		LV_SYMBOL_NEW_LINE
	LV_SYMBOL_EYE_CLOSE		

6 – POSIÇÃO

```
lv_obj_align_to(btn2, btn1, LV_ALIGN_RIGHT, 0, 0);
```

```
+-----+ +-----+  
|  btn1  ||  btn2  |  
|         ||         |  
+-----+ +-----+
```

```
lv_obj_align_to(btn2, btn1, LV_ALIGN_BOTTOM_RIGHT, 0,  
0);
```

```
+-----+  
|  btn1  |  
|         |  
+-----+ +-----+  
                |  btn2  |  
                |         |  
                +-----+
```

7 – ESTILOS

```
void lv_thermostato(void) {  
    static lv_style_t style;  
    lv_style_init(&style);  
    lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_PURP  
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_  
    lv_style_set_border_width(&style, 5);
```

```
    lv_obj_t * btn1 = lv_btn_create(lv_scr_act());  
    lv_obj_add_event_cb(btn1, event_handler, LV_EVENT_ALL, NULL);  
    lv_obj_align(btn1, LV_ALIGN_BOTTOM_LEFT, 0, 0);  
+   lv_obj_add_style(btn1, &style, 0);
```

8 – CRIE BOTÕES

Agora você é capaz de recriar os demais botões da interface, para cada botão crie uma função de callback (similar ao ***event_handler***).

Implemente:

- ♦ ***M(btnMenu/ menu_handler)***: Menu
- ♦ ***Clock(btnClk/ clk_handler)***: Relógio
- ♦ ***^(btnUp/ up_handler)***: Aumentar (temperatura/ alarme):
- ♦ ***v(btnDown/ down_handler)***: Baixar (temperatura/ alarme)

Lembre de testar na placa! Vai precisar de ajustes.



PARTE II – FONTE E RELÓGIO SENDO ATUALIZADO PELO RTC

Sobre o curso

SAME70-Examples

Vídeos

⚡ Regras de firmware

▸ Util

▼ Labs

- Lab 1 - PIO
- Lab 2 - PIO - Driver
- Lab 3 - PIO - IRQ
- Lab 4 - RTOS

Lab 5 - RTOS HC-SR04

Lab 6 - RTOS IMU

▼ Lab 7 - RTOS - LCD - LVGL
Começando

Lab - Parte 1

▼ **Lab - Parte 2**

Lab - Parte 2

DS » Lab 7 - RTOS - LCD - LVGL » Lab - Parte 2

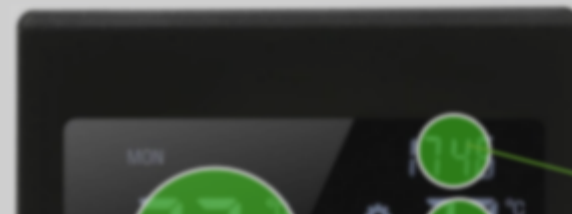
Lab - Parte 2

Pastas: /Lab7-RTOS-LCD-LVGL

Data LIMITE para entrega: 26/04/2023 - 23h59

Info

Não podemos esquecer nossa referencia!



PARTE II – RESULTADO ESPERADO



RUBRICA

C

Concluiu o lab e fez com que o relógio fosse atualizado pelo RTC (igual a imagem)

EXTRAS → A+



Implementar o dígito da Temperatura: 23.4

O botão de settings deve possibilitar o usuário configurar a hora certa

Incluir demais labels e logos

Gerar um logo para o relógio e usar no lugar do de settings

Implementar o botão de Power que desliga a tela

Colocar um potenciômetro que altera o valor da temperatura atual.