# Prática 6

$I^2C$

**Inter-Integrated Circuit**

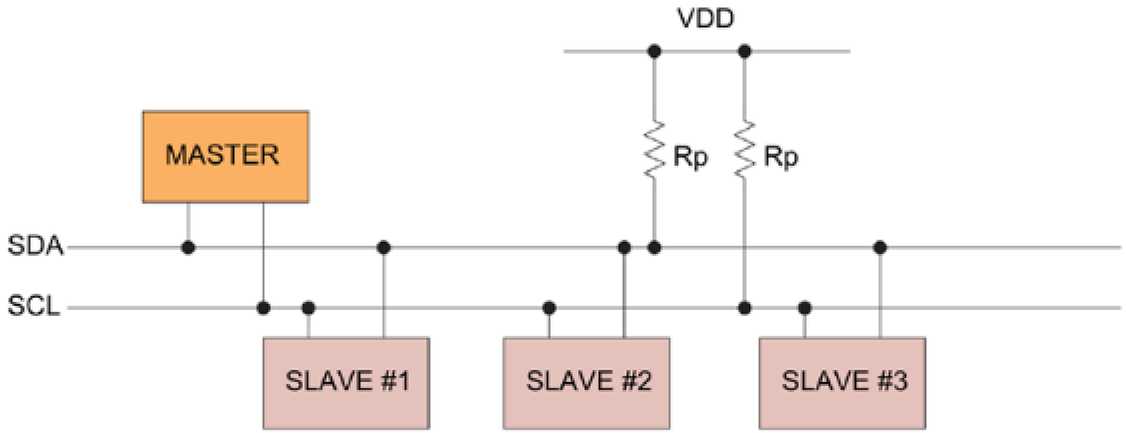
# I$^2$C

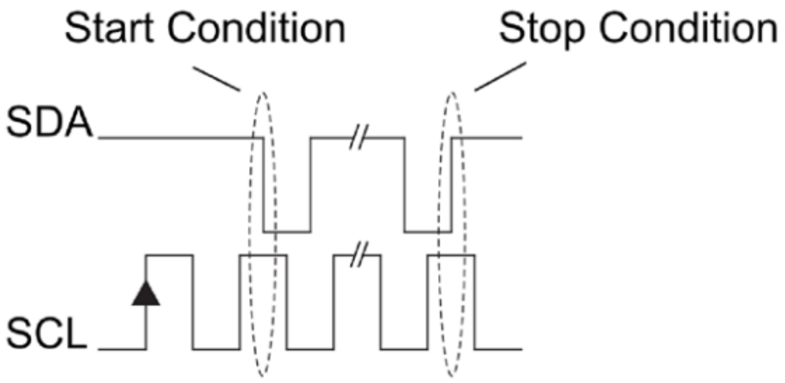
I2C is a serial, synchronous, half-duplex communication protocol that allows co-existence of multiple masters and slaves on the same bus. The I$^2$C bus consists of two lines: serial data line (SDA) and serial clock (SCL). Both lines require pull-up resistors.

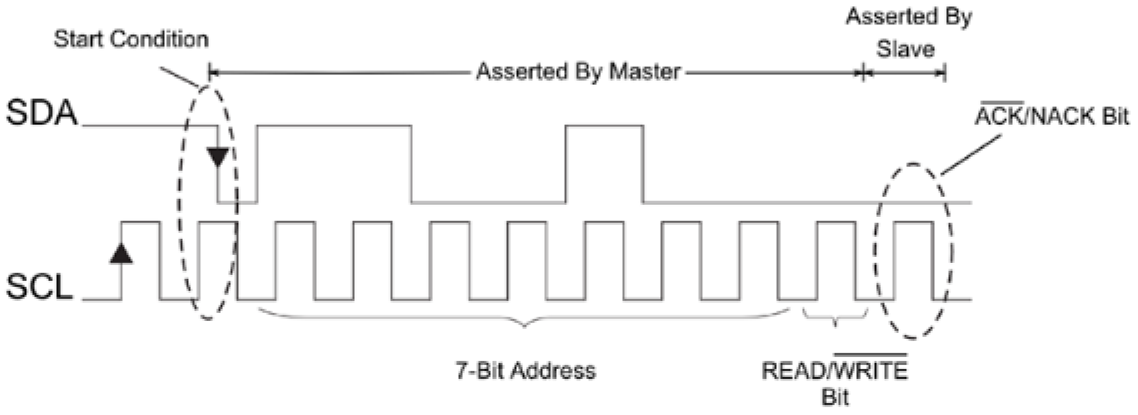
With such advantages as simplicity and low manufacturing cost, I2C is mostly used for communication of low-speed peripheral devices over short distances (within one foot).

ESP32 has 2 I$^2$C controller (also referred to as port), responsible for handling communications on the I$^2$C bus. A single I2C controller can operate as master or slave.

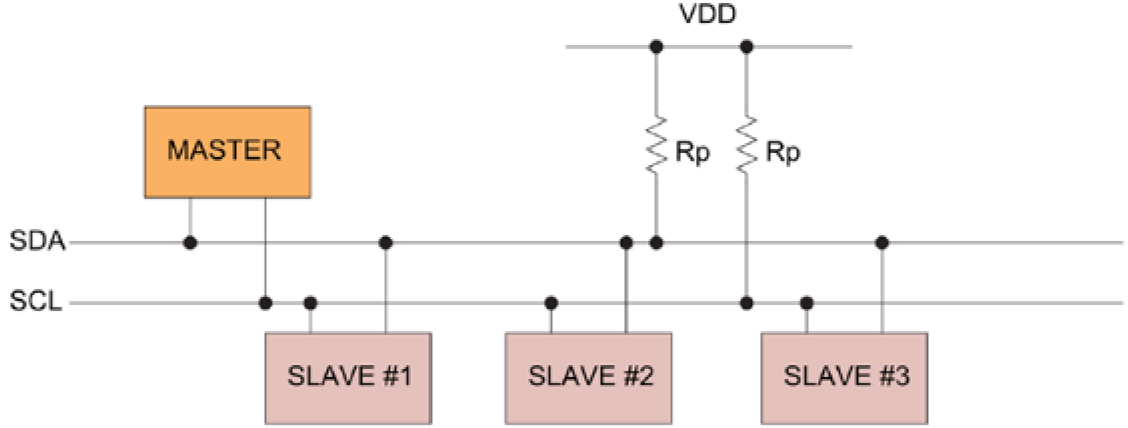
# I$^2$C

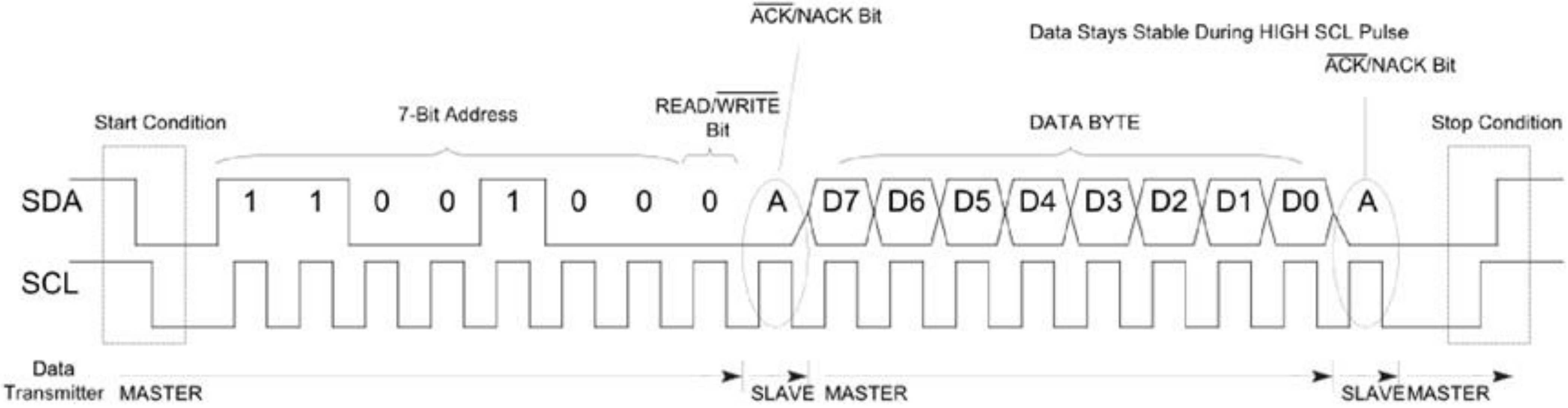

VDD
MASTER
Rp
Rp
SDA
SCL
SLAVE #1
SLAVE #2
SLAVE #3


Start Condition
Stop Condition
SDA
SCL

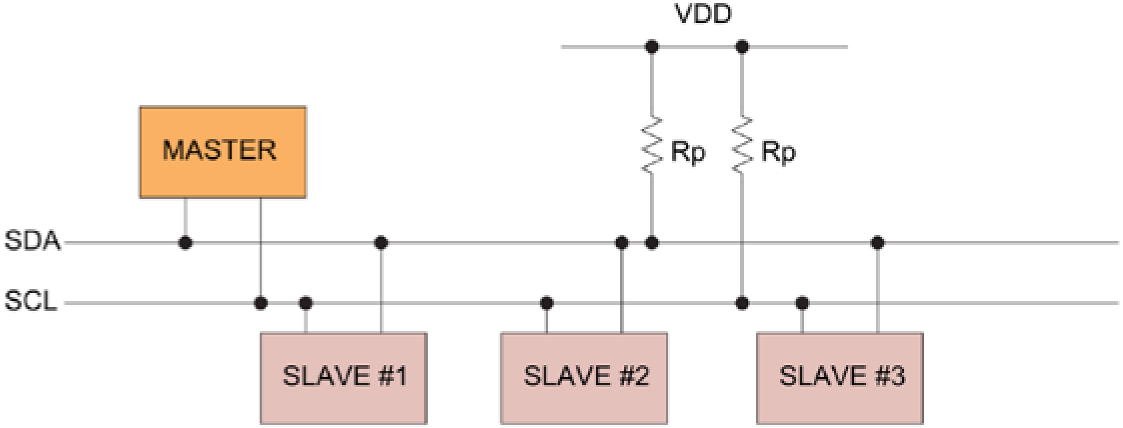

Start Condition
Asserted By Master
Asserted By Slave
SDA
$\overline{\text{ACK}}$/NACK Bit
SCL
7-Bit Address
READ/$\overline{\text{WRITE}}$ Bit

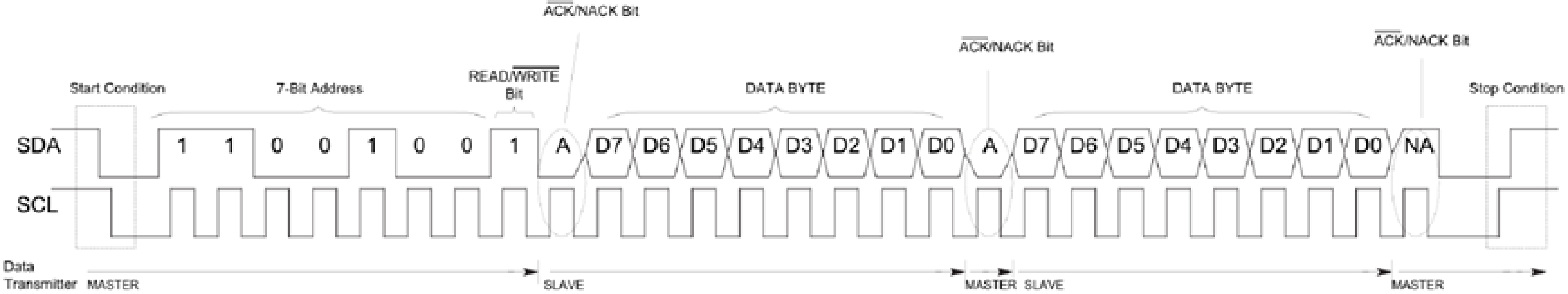
# I$^2$C - Write


VDD
Rp
Rp
MASTER
SDA
SCL
SLAVE #1
SLAVE #2
SLAVE #3


ACK/NACK Bit
Data Stays Stable During HIGH SCL Pulse
ACK/NACK Bit
Start Condition
7-Bit Address
READ/WRITE Bit
DATA BYTE
Stop Condition
SDA
1
1
0
0
1
0
0
0
A
D7
D6
D5
D4
D3
D2
D1
D0
A
SCL
Data Transmitter
MASTER
SLAVE
MASTER
SLAVE
MASTER

# I$^2$C - Read


VDD
Rp
Rp
MASTER
SDA
SCL
SLAVE #1
SLAVE #2
SLAVE #3


ACK/NACK Bit
ACK/NACK Bit
ACK/NACK Bit
Start Condition
7-Bit Address
READ/WRITE Bit
DATA BYTE
DATA BYTE
Stop Condition
SDA
1 1 0 0 1 0 0 1 A D7 D6 D5 D4 D3 D2 D1 D0 A D7 D6 D5 D4 D3 D2 D1 D0 NA
SCL
Data Transmitter
MASTER
SLAVE
MASTER
SLAVE
MASTER

# LCD

- Initialize I2C bus
- Install panel IO
- Install SSD1306 panel driver
- Initialize LVGL

# Initialize I2C bus

```
ESP_LOGI(TAG, "Initialize I2C bus");
    i2c_master_bus_handle_t i2c_bus = NULL;
    i2c_master_bus_config_t bus_config = {
        .clk_source = I2C_CLK_SRC_DEFAULT,
        .glitch_ignore_cnt = 7,
        .i2c_port = I2C_BUS_PORT,
        .sda_io_num = EXAMPLE_PIN_NUM_SDA,
        .scl_io_num = EXAMPLE_PIN_NUM_SCL,
        .flags.enable_internal_pullup = true,
    };
    ESP_ERROR_CHECK(i2c_new_master_bus(&bus_config, &i2c_bus));
```

# Install panel IO

```
ESP_LOGI(TAG, "Install panel IO");
    esp_lcd_panel_io_handle_t io_handle = NULL;
    esp_lcd_panel_io_i2c_config_t io_config = {
        .dev_addr = EXAMPLE_I2C_HW_ADDR,
        .scl_speed_hz = EXAMPLE_LCD_PIXEL_CLOCK_HZ,
        .control_phase_bytes = 1,               // According to SSD1306 datasheet
        .lcd_cmd_bits = EXAMPLE_LCD_CMD_BITS,   // According to SSD1306 datasheet
        .lcd_param_bits = EXAMPLE_LCD_CMD_BITS, // According to SSD1306 datasheet
        .dc_bit_offset = 6,                     // According to SSD1306 datasheet
    };
    ESP_ERROR_CHECK(esp_lcd_new_panel_io_i2c(i2c_bus, &io_config, &io_handle));
```

# Install SSD1306 panel driver

```
ESP_LOGI(TAG, "Install SSD1306 panel driver");
    esp_lcd_panel_handle_t panel_handle = NULL;
    esp_lcd_panel_dev_config_t panel_config = {
        .bits_per_pixel = 1,
        .reset_gpio_num = EXAMPLE_PIN_NUM_RST,
    };
    esp_lcd_panel_ssd1306_config_t ssd1306_config = {
        .height = EXAMPLE_LCD_V_RES,
    };
    panel_config.vendor_config = &ssd1306_config;
    ESP_ERROR_CHECK(esp_lcd_new_panel_ssd1306(io_handle, &panel_config, &panel_handle));
    ESP_ERROR_CHECK(esp_lcd_panel_reset(panel_handle));
    ESP_ERROR_CHECK(esp_lcd_panel_init(panel_handle));
    ESP_ERROR_CHECK(esp_lcd_panel_disp_on_off(panel_handle, true));
```

# Initialize LVGL

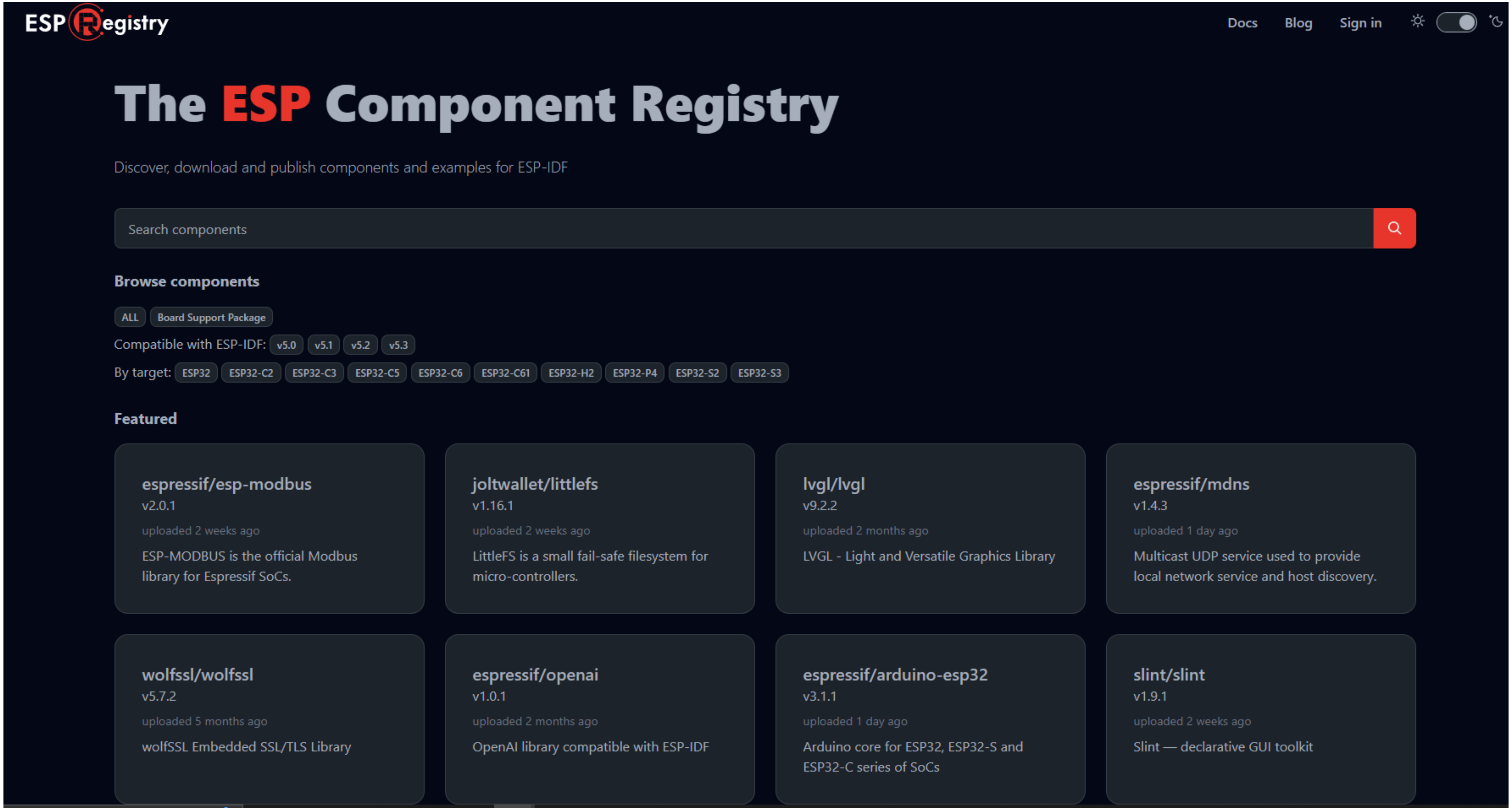
```
ESP_LOGI(TAG, "Initialize LVGL");
    const lvgl_port_cfg_t lvgl_cfg = ESP_LVGL_PORT_INIT_CONFIG();     lvgl_port_init(&lvgl_cfg);
    const lvgl_port_display_cfg_t disp_cfg = {
        .io_handle = io_handle,       .panel_handle = panel_handle,
        .buffer_size = EXAMPLE_LCD_H_RES * EXAMPLE_LCD_V_RES,
        .double_buffer = true,
        .hres = EXAMPLE_LCD_H_RES,       .vres = EXAMPLE_LCD_V_RES,
        .monochrome = true,
        .rotation = {
            .swap_xy = false,
            .mirror_x = false,
            .mirror_y = false,
        }
    };
    lv_disp_t *disp = lvgl_port_add_disp(&disp_cfg);
```

# Utilizando LVGL

```
lv_obj_t *scr = lv_disp_get_scr_act(disp);
lv_obj_t *label = lv_label_create(scr);
lv_label_set_long_mode(label, LV_LABEL_LONG_SCROLL_CIRCULAR);
lv_label_set_text(label, "Hello Espressif, Hello LVGL.");
lv_obj_set_width(label, disp->driver->hor_res);
lv_obj_align(label, LV_ALIGN_TOP_MID, 0, 0);
```

// Pode-se criar quantos labels forem necessários.

# https://components.espressif.com/



## The ESP Component Registry

Discover, download and publish components and examples for ESP-IDF

Search components

### Browse components

ALL | Board Support Package

Compatible with ESP-IDF: v5.0 | v5.1 | v5.2 | v5.3

By target: ESP32 | ESP32-C2 | ESP32-C3 | ESP32-C5 | ESP32-C6 | ESP32-C61 | ESP32-H2 | ESP32-P4 | ESP32-S2 | ESP32-S3

### Featured

**espressif/esp-modbus**
v2.0.1
uploaded 2 weeks ago
ESP-MODBUS is the official Modbus library for Espressif SoCs.

**joltwallet/littlefs**
v1.16.1
uploaded 2 weeks ago
LittleFS is a small fail-safe filesystem for micro-controllers.

**lvgl/lvgl**
v9.2.2
uploaded 2 months ago
LVGL - Light and Versatile Graphics Library

**espressif/mdns**
v1.4.3
uploaded 1 day ago
Multicast UDP service used to provide local network service and host discovery.

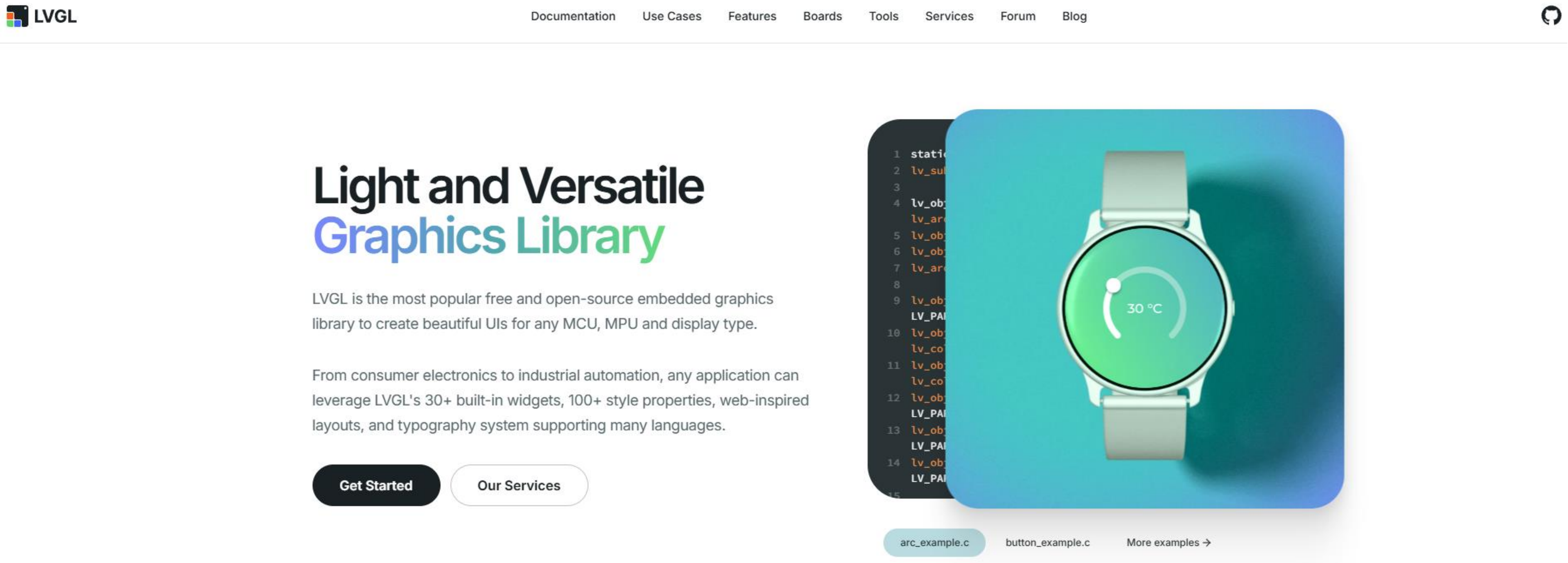
**wolfssl/wolfssl**
v5.7.2
uploaded 5 months ago
wolfSSL Embedded SSL/TLS Library

**espressif/openai**
v1.0.1
uploaded 2 months ago
OpenAI library compatible with ESP-IDF

**espressif/arduino-esp32**
v3.1.1
uploaded 1 day ago
Arduino core for ESP32, ESP32-S and ESP32-C series of SoCs

**slint/slint**
v1.9.1
uploaded 2 weeks ago
Slint — declarative GUI toolkit

# LVGL - https://lvgl.io/



## Light and Versatile Graphics Library

LVGL is the most popular free and open-source embedded graphics library to create beautiful UIs for any MCU, MPU and display type.

From consumer electronics to industrial automation, any application can leverage LVGL's 30+ built-in widgets, 100+ style properties, web-inspired layouts, and typography system supporting many languages.

```
stati
lv_sul

lv_obj
lv_arc
lv_obj
lv_obj
lv_arc

lv_obj
LV_PAI
lv_obj
lv_co
lv_obj
lv_co
lv_obj
LV_PAI
lv_obj
LV_PAI
lv_obj
LV_PAI
```

# Referências

- https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf