QUECTEL

Build a smarter world

移远通信

嵌入式软件开发课程



软件部-蒙政威

01 LVGL介绍

Part One

Build a smarter world

概述

LVGL：

LVGL(轻巧而多功能的图形库)是一个免费的开放源代码图形库，它提供创建具有易于使用的图形元素，精美的视觉效果和低内存占用的嵌入式GUI所需的一切。



LVGL开源库链接：

什么是Lvgl？ — 百问网LVGL中文教程手册文档 1.0 文档

QuecPython LVGL 的核心特性

01 上手简单 · 低门槛开发

基于Python简洁语法，无需深入底层C语言开发，大幅降低嵌入式GUI开发的学习成本，让开发者快速上手实现界面逻辑。

02 组件丰富 · 即开即用

内置Label、Button、Img、List、Slider、Chart等海量UI控件，覆盖文本、交互、图表、列表等全场景界面需求。

03 视觉细腻 · 动态交互

支持硬件加速的平滑动画、反锯齿边缘渲染、透明度渐变及流畅列表滚动，打造媲美移动端的精致视觉体验。

04 灵活适配 · 多元输入

完美兼容电容/电阻触摸屏、物理按键、旋转编码器等多种输入设备，轻松适配不同交互场景的硬件需求。

05 所见即所得 · 极速开发

配套 GUI Guider 可视化拖拽工具，无需手写代码即可快速构建界面原型，自动生成工程代码，开发效率倍增。

06 资源完备 · 轻松上手

提供详尽的中英文在线/离线文档、丰富的实战示例代码及活跃的技术社区支持，快速解决开发疑难。

02 GUI Guide 工具的使用

Part Two

Build a smarter world

LVGL 工作流程概览

01 显示输出数据流

从图形渲染到屏幕点亮的全链路：
LVGL 核心绘制→ 显示驱动(disp_drv) **→** 刷新回调(flush_cb) **→LCD 物理屏幕**
将矢量图形与控件最终转换为像素点显示。

02 输入事件处理流

从硬件触摸到系统响应的交互闭环：
触摸面板(TP)→ 读取回调(read_cb) **→** 输入驱动(indev_drv) **→LVGL 事件分发**
实时捕捉用户操作并映射为界面交互事件。

03 核心组件与职责

- **LVGL核心库：**负责布局计算、样式渲染、动画与事件逻辑的“大脑”。
- **驱动层：**屏蔽硬件差异，作为连接上层GUI与底层硬件的桥梁。

架构价值总结：
LVGL 采用了典型的“应用层-中间件-驱动层-硬件层”分层架构。通过驱动回调机制（flush_cb/read_cb）实现了与具体显示、输入硬件的解耦，这使得 LVGL 具备了极高的跨平台移植性，开发者无需关注底层硬件细节，即可快速实现高效的嵌入式图形界面开发。

基础对象（Base Object）

01 核心定义：所有组件的“基因库”

`lv.obj` 是 LVGL 中**所有可视组件的基类**（Base Class），它实现了UI元素通用的基础属性与交互行为，是构建按钮、标签、列表等所有组件的底层基石。

🔑 通用属性的统一封装

它封装了所有组件的共性特征：包括坐标位置、尺寸大小、父子层级关系、样式风格（颜色/字体/边框）以及事件回调机制，让所有组件具备一致的基础交互能力。

🌳 组件继承体系的源头

LVGL 中所有具体组件（如Button按钮、Label标签、List列表）均直接或间接继承自 `lv.obj`，在复用其基础能力的同时，扩展各自的专属功能与显示形态。

🖥️ 角色一：独立的显示界面（Screen）

当 `lv.obj` 不指定任何父对象时，它将作为**根对象**成为一个独立的显示界面（屏幕），是承载当前页面所有交互元素的顶层容器。

📦 子元素的容器（Container）

当指定父对象后，它作为普通容器存在，用于分组管理多个子组件（如按钮组、输入框组合），实现界面元素的模块化布局、统一控制与样式管理。

核心机制一：坐标系统

01 尺寸定义（Size）

设定组件的基础几何边界，通过宽度(Width)和高度(Height)属性，决定组件在屏幕上的占用空间，是所有布局的基础。

```
obj.set_size(new_width, new_height)
```

02 坐标定位（Position）

以父容器的左上角为原点(0,0)，通过设置X轴和Y轴的像素偏移量，实现组件在画布上的绝对位置控制。

```
obj.set_pos(new_x, new_y)
```

03 智能对齐（Alignment）

提供相对父对象的高级定位能力，支持居中、边缘吸附及偏移对齐，无需手动计算坐标，简化动态布局。

```
obj.align(lv.ALIGN.CENTER, dx, dy)
```

💡 典型应用场景与代码示例

场景一：自动居中- 无需计算父容器尺寸，直接调用`obj.align(lv.ALIGN.CENTER, 0, 0)`即可让组件在父容器中完美居中，适配不同屏幕分辨率。

场景二：边缘偏移- 实现与父容器边缘的距离控制：`obj.align(lv.ALIGN.TOP_RIGHT, -20, 20)`，轻松完成距离右上角内缩20px的布局需求。

核心机制二：样式系统（Styles）

01 核心作用

用于定义组件的视觉外观，包含背景色、字体、边框、圆角等属性。实现样式与业务逻辑的解耦，让界面主题切换和美化变得灵活高效。

02 实例化样式

创建样式对象并完成初始化，为后续配置属性做准备：

```
style_btn = lv.style_t()
style_btn.init()
```

03 配置视觉属性

设置具体的样式参数，定制组件外观：

```
style_btn.set_bg_color(
lv.PALETTE.YELLOW)
style_btn.set_radius(5)
```

04 绑定到组件

将样式对象应用到具体的组件实例上，使样式生效：

```
# 0 代表默认状态
btn1.add_style(
style_btn, 0)
```

核心机制三：事件系统（Events）

事件系统是嵌入式GUI的交互神经中枢，负责实时捕获**用户操作（点击、滑动、按键）**与系统状态变化。通过「回调函数」机制实现界面操作与业务逻辑的解耦，当特定事件触发时自动执行预设的处理逻辑，是实现按钮响应、菜单切换、数据交互等动态功能的基础。

```
# 定义事件回调函数：响应按钮点击
def on_btn_click(evt):
 if evt.get_code() == lv.EVENT.CLICKED: # 判定点击事件类型
 print("Button Clicked! 执行业务逻辑...")

# 为组件绑定事件监听器
btn = lv.button(lv.scr_act()) # 创建按钮对象
btn.add_event_cb(on_btn_click, lv.EVENT.CLICKED, None) # 绑定回调
btn.set_pos(100, 150) # 设置显示位置
```

01. lv.EVENT.CLICKED
点击并释放后触发，最常用的交互事件。适用于按钮确认、菜单选项选择、弹窗确认等一次性触发场景。

02. lv.EVENT.PRESSED
按下瞬间立即触发（无需释放）。常用于实现长按检测、连续触发（如快速调节音量）或按下时的即时视觉反馈。

03. lv.EVENT.VALUE_CHANGED
组件数值或状态改变时触发。典型应用于滑块拖动、下拉列表选择、开关切换、进度条更新等动态状态场景。

03 常用组件开发

Part Three

Build a smarter world

标签（Label）组件基础

核心用途：作为 LVGL 中最基础的文本显示单元，Label 组件负责在图形界面上呈现静态或动态文本信息，是构建用户交互界面的基石。它支持丰富的排版属性，如对齐方式、自动换行、颜色定制等，满足多样化的文本展示需求。

```
# 1. 初始化 Label 并挂载到当前屏幕
demo_label = lv.label(lv.scr_act())
# 2. 设置文本内容与居中对齐属性
demo_label.set_text("Hello, QuecPython LVGL World!")
```

01 set_text(text)

用于更新或设置组件的文本内容。支持直接传入字符串常量，也可通过字符串拼接实现动态数据的实时展示，是 Label 最常用的基础方法。

02 set_long_mode(mode)

控制长文本的显示策略。当文本长度超出组件宽度时，可配置为自动换行、尾部裁剪、滚动循环等模式，确保界面排版整洁美观。

03 set_recolor(en)

开启或关闭文本内联颜色标记。启用后，可在字符串中通过［颜色值］标记语法为局部文字单独设置颜色，实现多彩文本效果。

按钮（Button）组件

01 组件用途与定位

作为嵌入式 GUI 中最核心的交互触发单元，Button 组件用于响应用户的点击、按压等操作，是实现“用户指令输入”与“系统功能响应”之间的关键桥梁，广泛应用于功能触发、确认提交、模式切换等场景。

02 关键设计特性

- **容器特性：**按钮本身仅为交互载体，需配合 Label 组件承载文字内容，二者为“父容器+子元素”的依附关系；
- **事件驱动：**支持捕获 CLICKED（点击）、LONG_PRESSED（长按）等事件，通过回调函数实现自定义业务逻辑。

Python 快速实现示例：

```
btn = lv.btn(screen) # 1. 创建按钮并挂载到屏幕根对象
btn.center() # 2. 设置按钮居中显示
label = lv.label(btn) # 3. 创建标签并作为按钮的子元素
label.set_text("点击执行") # 4. 为按钮设置显示文本
btn.add_event_cb(cb_click, lv.EVENT.CLICKED, None) # 5. 绑定点击事件回调
```

LVGL 图像（Image）组件

01 核心用途：作为LVGL的基础视觉组件，用于在嵌入式GUI中高效渲染图片资源（支持JPG/PNG/BMP等）及内置矢量符号，是构建界面视觉元素、实现图文展示与动态交互的核心载体。

02 基础实现（Python API）：

```
img = lv.img(screen) # 1. 创建图像对象，挂载至屏幕根节点
img.set_pos(80, 80) # 2. 设置组件坐标（X轴，Y轴）
img.set_src("icon.png") # 3. 加载本地图片 / 或传 lv.SYMBOL.OK 调用内置符号
```

03 变换 · 旋转控制

调用 set_angle(angle) 实现旋转，参数单位为 0.1°（如450即45°）。支持0-360°自由调整，可用于实现动态仪表盘、指针旋转等交互效果。

03 变换 · 缩放调节

调用 set_zoom(zoom) 调整尺寸，以256为原始比例基准。数值>256为放大（如512=2倍），数值<256为缩小（如128=0.5倍），适配不同分辨率屏幕显示。

Thank you

获取更多产品信息, 请访问: www.quectel.com.cn
全国热线: 400 960 7678

移远全球中心
上海市松江区泗泾镇外婆泾路8号
联系电话：021 5108 6236　销售支持：sales@quectel.com
技术支持：support@quectel.com　电子邮件：info@quectel.com

QUECTEL

移远微信公众号