ZDP14x0 UI开发设计演示

GZLG Technology Corp.,Ltd

芯片与智能物联解决方案供应商

目录

AWTK下一代开源GUI引擎

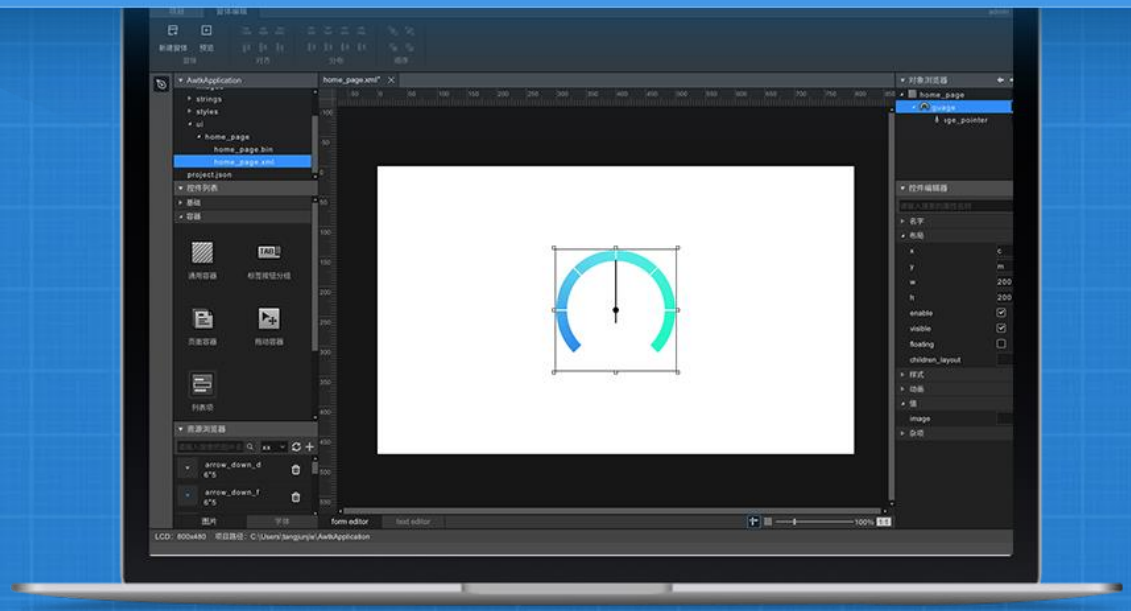
AWTK全称为Toolkit AnyWhere，是ZLG倾心打造的一套基于C语言开发的GUI框架。旨在为用户提供一个功能强大、高效可靠、简单易用、可轻松做出炫酷效果的GUI引擎，支持跨平台同步开发，一次编程，到处编译，跨平台使用。

- 支持 纯C语言
- 支持 多种操作系统
- 支持 硬件加速
- 完善的 动画系统
- 丰富的 GUI控件
- 支持 位图和矢量字体
- 内置 中英文输入法
- 支持 组态方式开发界面

AWTK Designer

AWTK Designer是专门用来制作AWTK应用程序UI界面的实用型工具，只要通过拖曳和点击就可以完成复杂的界面设计，并且能够随时预览效果图。

工程管理 | 页面管理 | 资源管理 | 样式管理 | 控件列表 | 界面布局 | 属性编辑 | 打包发布



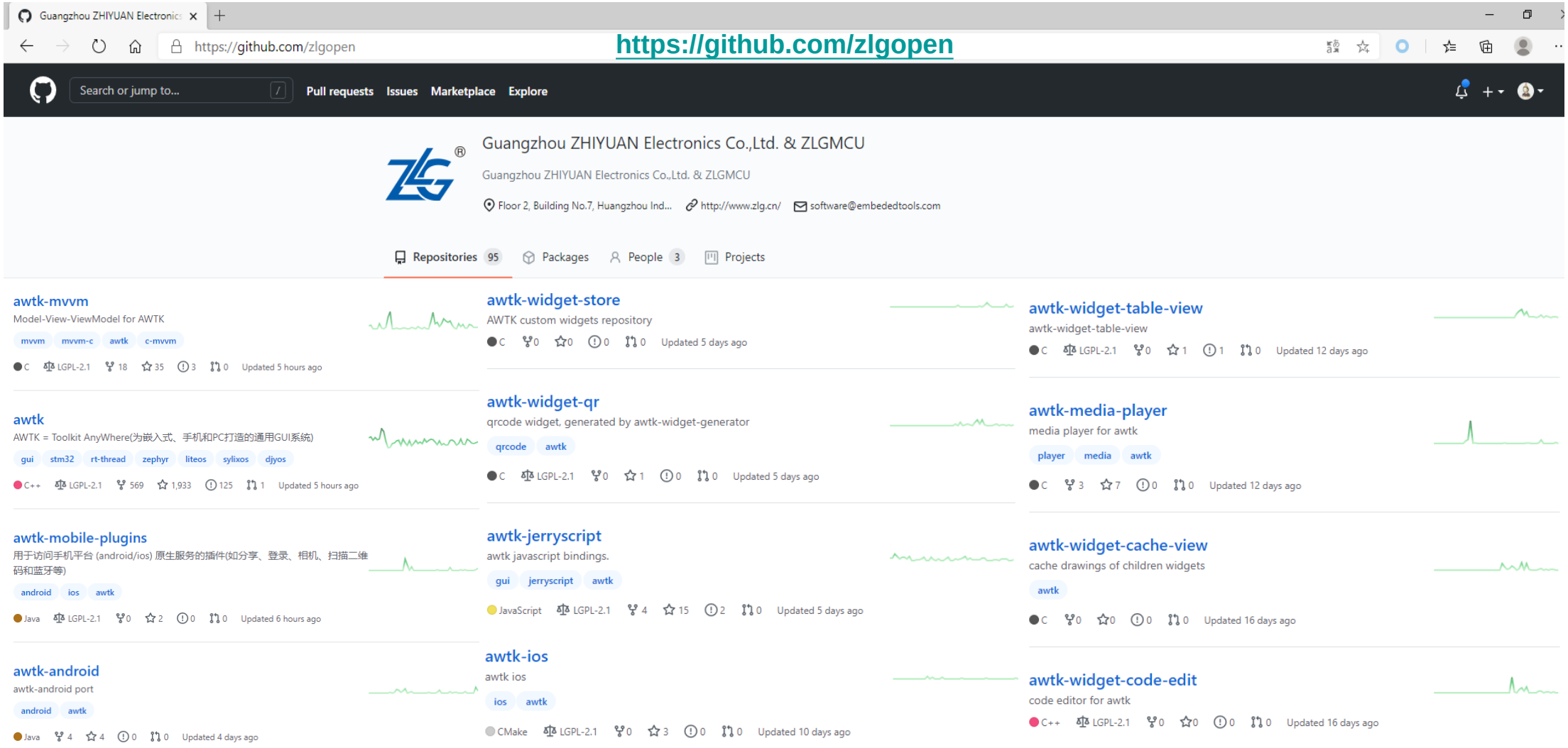
得益于AWTK强大的功能和跨平台特性，AWTK Designer本身也是基于AWTK构建的。

所见即所得，更多功能触手可及

界面编辑区界面

- 不再需要手写 XML
- 拖拽方式设计界面，所见即所得
- 快速预览，一键打包资源

开放源码，免费商用 (LGPL)



AWTK与其他GUI比较

相比目前市面上常见的GUI开发框架，如：emWin、Qt、TouchGFX等，AWTK有如下独特的优势：

	语言	大小	效率	炫酷	跨平台	开源	免费
AWTK	C、Js	小	高	是	全平台	是	是
emWin SEGGER	C	小	高	否	嵌入式平台	否	部分平台
Qt	C++	大	低	一般	中高端平台	是	是
TouchGFX	C++	较小	较高	是	嵌入式平台	否	部分平台

AWTK支持控件


基础
OK
按钮
多选按钮
单选按钮
TAB
标签按钮
AB
单行编辑
1
微调编辑
下拉列表
下拉列表项
静态文本
图片
滑动条
70%
进度条
滚动条
Mobile滚动条
指示器
弧形指示器
srf
输入法
输入法候选字


图文
GIF
GIF图片
SVG
SVG图片
图片动画
0.00
图片值
富文本
表盘
表盘指针
123
环形进度条
模拟时钟
8:30
数字时钟
多行编辑器
行号
滚动文本
选择器
开关
7:00
8:00
文本选择器
滑动菜单

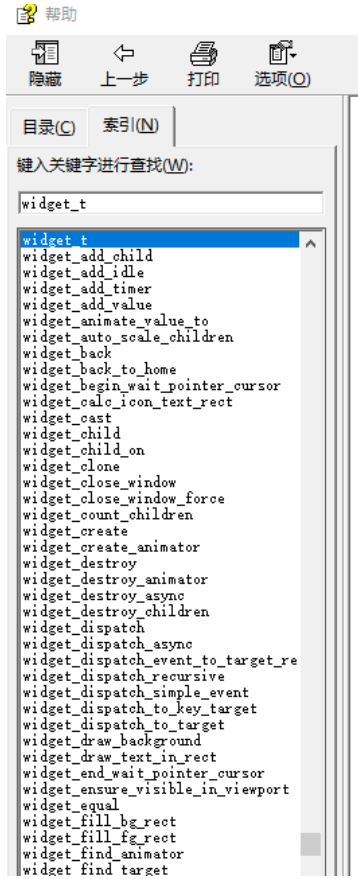

容器
通用容器
TAB
标签按钮分组
页面容器
拖动容器
列表项
剪裁容器
网格容器
视图
标签视图
滚动视图
滑动视图
列表视图
水平列表视图

基本控件使用介绍

进入https://awtk.zlg.cn/api/awtk下载并打开 AWTK_API.chm，点击索引，在输入框内输入“对应的控件名称”+“_t”即可查找该控件专用函数，以button为例，搜索button_t，刨去button_case，一共有四个函数可用：

基本控件使用介绍

很多情况下，控件专用函数并不能满足需求，可在AWTK_API索引中搜索“widget_t”，这些函数为通用函数，只要控件支持，就能使用。



函数

函数名称	说明
widget_add_child	加入一个子控件。
widget_add_idle	创建idle。
widget_add_timer	创建定时器。
widget_add_value	增加控件的值。
widget_animate_value_to	设置控件的值(以动画形式变化到指定的值)。
widget_auto_scale_children	根据缩放子控件的位置和大小。
widget_back	请求返回到前一个窗口。
widget_back_to_home	请求返回到home窗口。
widget_begin_wait_pointer_cursor	开始等待鼠标指针。
widget_calc_icon_text_rect	计算icon text的位置。
widget_cast	转换为widget对象(供脚本语言使用)。
widget_child	查找指定名称的子控件(同widget_lookup(widget, name, FALSE))。
widget_child_on	为指定名称的子控件注册指定事件的处理函数。

常用函数

函数名称	说明	函数名称	说明
widget_add_idle(timer)	创建定时器。	widget_foreach	遍历当前控件及子控件。
widget_get_prop_(xxxx)	获取控件指定属性的值。	widget_lookup	查找指定名称的子控件(返回第一个)。
widget_on	注册指定事件的处理函数。	widget_set_enable	设置控件的可用性。
widget_set_prop_(xxxx)	设置控件指定属性的值。	widget_set_style_(xxxx)	设置widget私有样式。
widget_set_text_utf8	设置控件的文本。	widget_set_visible	设置控件的可见性。
widget_start_animator	播放动画。	widget_use_style	启用指定的style。

常用函数

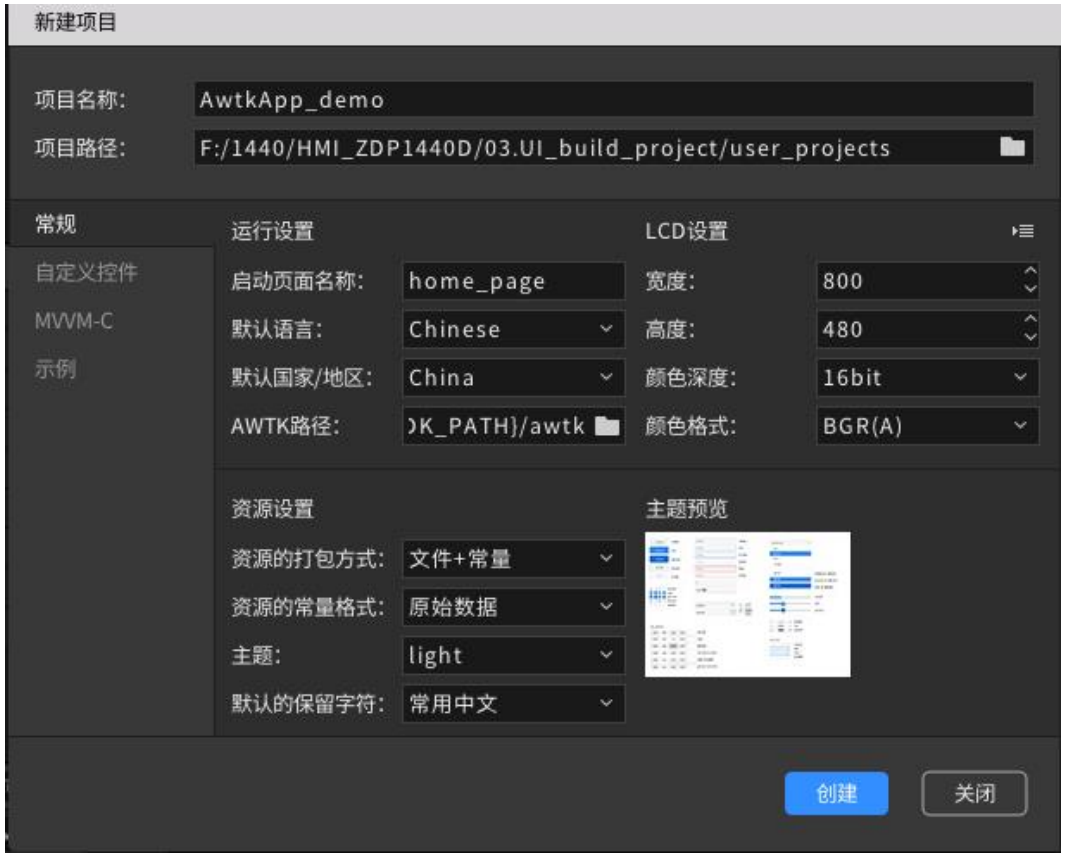
widget_set_prop_(xxxx)	说明
widget_set_prop_bool	设置布尔格式的属性。
widget_set_prop_int	设置整数格式的属性。
widget_set_prop_str	设置字符串格式的属性。

widget_set_style_(xxxx)	说明
widget_set_style_color	设置颜色类型的style。
widget_set_style_int	设置整数类型的style。
widget_set_style_str	设置字符串类型的style。

widget_get_prop_(xxxx)	说明
widget_get_prop_bool	获取布尔格式的属性。
widget_get_prop_int	获取整数格式的属性。
widget_get_prop_str	获取字符串格式的属性。

新建UI工程

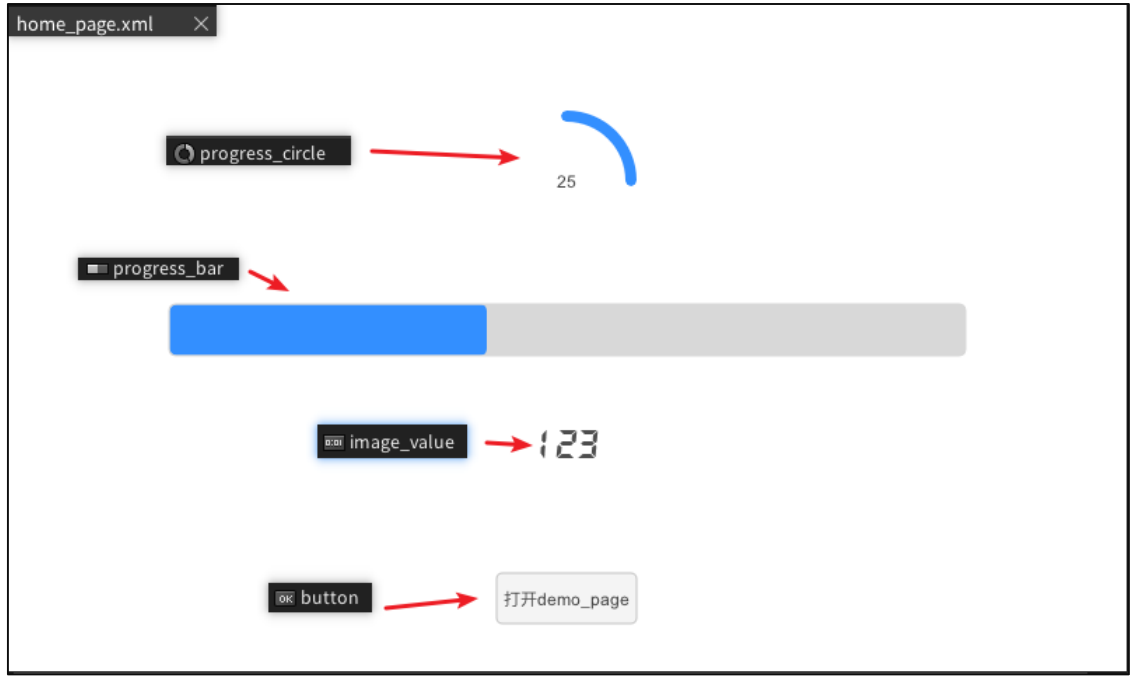
根据屏幕大小，在user_projects中创建一个UI。

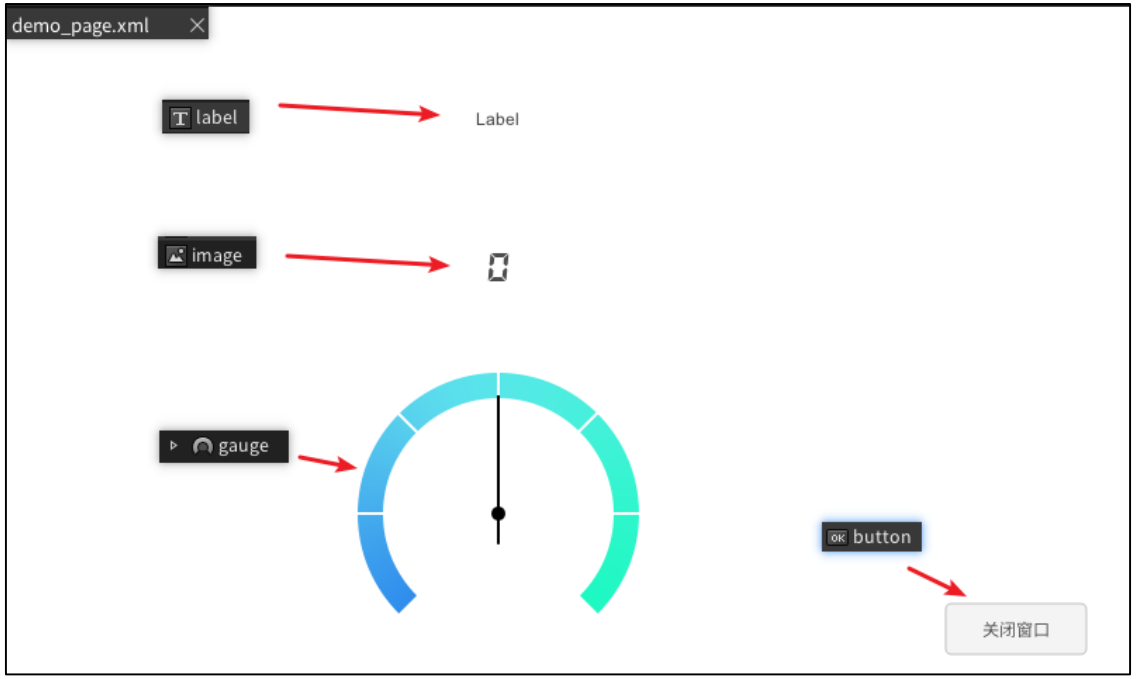


拖拽生成页面

创建好后，默认存在一个home_page页面。再新建一个demo_page，拖拽一些控件至两个页面上，home_page上放置一个按钮用来打开demo_page， demo_page上也增加一个按钮用来关闭自身。其余控件的刷新和显示通过通讯协议来完成交互。

绘制完页面后点击打包。





添加虚拟串口和解析器

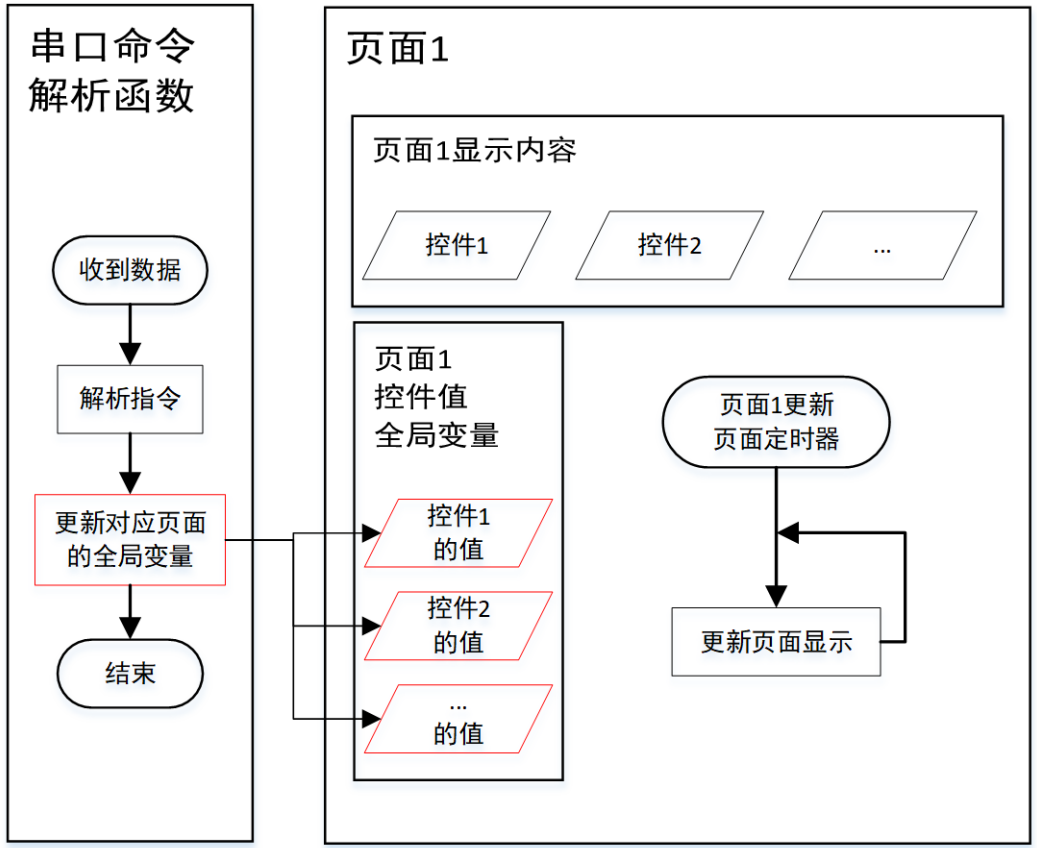
打开上位机工具，点击刷新，选择新建的UI工程，点击“添加PC端虚拟串口&命令解析器文件”；

默认仅使能PC端虚拟串口，如果需要用到命令解析器，需要勾选“使能命令解析器”。

建议更新页面方式

建议使用右侧图片中的方式更新页面控件。页面中的全局变量可以理解为当前设置的值。每个页面中的元素可以定义一个结构体，修改时会更加清晰。

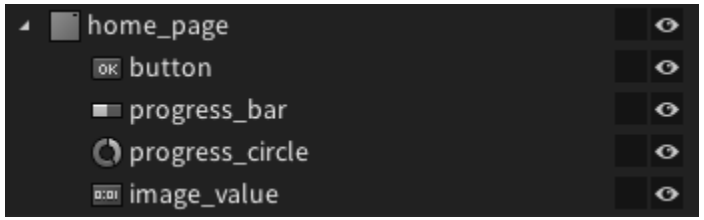
- 避免创建多个EVT_USER_RX_DATA事件的回调函数；
- 页面定时器使用widget_add_idle/widget_add_timer创建，关闭页面时可以自动销毁对应定时器；



添加更新页面代码

使用VS Code打开UI文件夹，根据上一节提到的内容，先为home_page创建一个包含页面元素数据的结构体。

在src文件夹中，创建一个struct_define.h文件，用以定义这些结构体；home_page里可以为进度条、环形进度条以及图片值这三个控件创建一个变量，保存各自的值。把这三个变量统一放在homepage_set_t结构体中。



struct_define.h ...\AwtkApp_demo\... | navigator.h

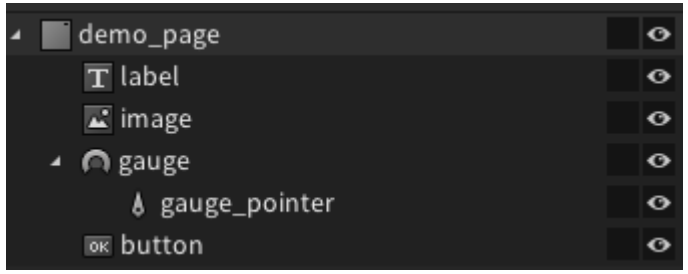
user_projects > AwtkApp_demo > src > struct_define.h > ...

```c
#ifndef __STRUCT_DEFINE__
#define __STRUCT_DEFINE__
#include "awtk.h"


/* home_page中显示的内容 */
typedef struct _homepage_data_t {
    uint8_t     progress_bar_data;
    uint8_t     progress_circle_data;
    uint16_t    image_value_data;
}homepage_data_t;


#endif  //__STRUCT_DEFINE__

```

添加更新页面代码

同理，demo_page中可以为静态文本、图片和表盘创建对应的变量，放在同一个结构体中；
为方便演示，静态文本和图片所显示的内容均来自字符串数组，label_data以及image_data为字符串数组的下标。

```
static const char *label_str[] = {"label", "一", "二", "三", "四", "五"};
static const char *image_str[] = {"num0", "num1", "num2", "num3", "num4", "num5"};
```



struct_define.h 1

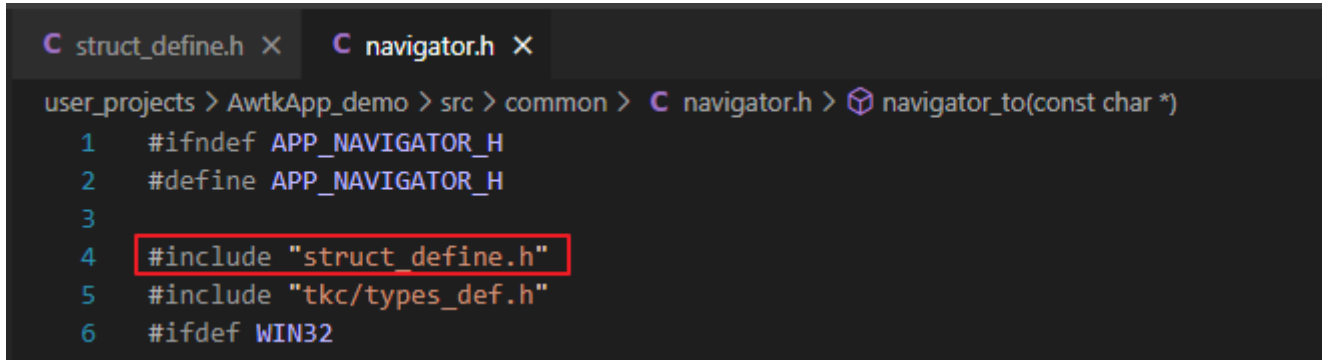
src > struct_define.h > ...

```
/* demo_page中显示的内容 */
typedef struct _demopage_data_t {
    uint8_t     label_data;
    uint8_t     image_data;
    int16_t     gauge_pointer_data;
}demopage_data_t;
```

添加更新页面代码

在common/navigator.h中包含新增的struct_define.h。

common/navigator.h在UI项目的每一个C文件中都会包含，虽然struct_define.h中定义的结构体在多个文件中被使用，但仅需在navigator.h中包含一次。



添加更新页面代码

为每个页面定义全局结构体，用来存放设置的值。建议在定义时，将两个结构体都初始化为UI的初始值。
全局变量可在struct_define.h中统一声明。

home_page.c 2

c > pages > C home_page.c > refresh_homepage(const

```
4   homepage_data_t __g_homepage_set = {
5       .progress_bar_data     = 40,
6       .progress_circle_data  = 25,
7       .image_value_data      = 123,
8   };
```

demo_page.c 2

c > pages > C demo_page.c > refresh_demopage(con

```
4   demopage_data_t __g_demopage_set = {
5       .label_data = 0,
6       .image_data = 0,
7       .gauge_pointer_data = 0,
8   };
```

struct_define.h 1

c > C struct_define.h > ...

```
19
20      extern homepage_data_t __g_homepage_set;
21      extern demopage_data_t __g_demopage_set;
22
```

添加更新页面代码

可以利用visit_init_child函数获取每一个控件的指针，保存在静态全局指针中。在home_page_init中使用widget_add_idle(win, refresh_homepage);创建一个idle，在其回调函数中更新控件显示。

home_page.c 3

ser_projects > AwtkApp_demo > src > pages > C home_page.c > refresh_homepage(const idle_i

```c
static widget_t* progress_bar    = NULL;
static widget_t* progress_circle = NULL;
static widget_t* image_value     = NULL;
/**
 * 初始化窗口的子控件
 */
static ret_t visit_init_child(void* ctx, const void* iter) {
  widget_t* win = WIDGET(ctx);
  widget_t* widget = WIDGET(iter);
  const char* name = widget->name;

  // 初始化指定名称的控件（设置属性或注册事件），请保证控件名称在窗口上唯一
  if (name != NULL && *name != '\0') {
    if (tk_str_eq(name, "button")) {
      widget_on(widget, EVT_CLICK, on_button_click, win);
    } else if (tk_str_eq(name, "progress_bar")) {
      progress_bar = widget;
    } else if (tk_str_eq(name, "progress_circle")) {
      progress_circle = widget;
    } else if (tk_str_eq(name, "image_value")) {
      image_value = widget;
    }
  }
  return RET_OK;
}
```

home_page.c 2

src > pages > C home_page.c > refresh_homepage(const idle_info_t *)

```c
static ret_t refresh_homepage(const idle_info_t* d)
{
  progress_bar_set_value(progress_bar, __g_homepage_set.progress_bar_data);
  progress_circle_set_value(progress_circle, __g_homepage_set.progress_circle_data);
  image_value_set_value(image_value, __g_homepage_set.image_value_data);

  return RET_REPEAT;
}
```

添加更新页面代码

demo_page同理。

```c
static widget_t* label         = NULL;
static widget_t* image         = NULL;
static widget_t* gauge_pointer = NULL;
/**
 * 初始化窗口的子控件
 */
static ret_t visit_init_child(void* ctx, const void* iter) {
  widget_t* win = WIDGET(ctx);
  widget_t* widget = WIDGET(iter);
  const char* name = widget->name;

  // 初始化指定名称的控件（设置属性或注册事件），请保证控件名称在窗口上唯一
  if (name != NULL && *name != '\0') {
    if (tk_str_eq(name, "button")) {
      widget_on(widget, EVT_CLICK, on_button_click, win);
    } else if (tk_str_eq(name, "label")) {
      label = widget;
    }  else if (tk_str_eq(name, "image")) {
      image = widget;
    }  else if (tk_str_eq(name, "gauge_pointer")) {
      gauge_pointer = widget;
    }
  }
  return RET_OK;
}
```

demo_page.c 2

src > pages > C demo_page.c > refresh_demopage(const idle_info_t *)

```c
static ret_t refresh_demopage(const idle_info_t* d)
{
  static const char *label_str[] = {"label", "一", "二", "三", "四", "五"};
  static const char *image_str[] = {"num0", "num1", "num2", "num3", "num4", "num5"};

  widget_set_text_utf8(label, label_str[__g_demopage_set.label_data]);
  image_base_set_image(image, image_str[__g_demopage_set.image_data]);
  gauge_pointer_set_angle(gauge_pointer, __g_demopage_set.gauge_pointer_data);
  return RET_REPEAT;
}
```

添加协议解析

在cmd_parsing_example.c中使用ZDP_REGISTER_CMD注册命令：	在对应命令的回调函数中实现更改控件值对应的全局变量:	
• home_add_proba	progress_bar的值+5	指令
• home_add_proci	progress_circle的值+5	
• home_add_imgva	image_value的值+5	
• home_dec_proba	progress_bar的值-5	
• home_dec_proci	progress_circle的值-5	
• home_dec_imgva	image_value的值-5	
• demo_set_label	设置demo_page中label的值	指令+数据
• demo_set_image	设置demo_page中image的值	
• demo_set_gauge	设置demo_page中gauge的值	

添加协议解析

使用内置协议解析器注册两条指令，用于修改home_page下进度条的值为例；

```
static void home_add_pro_ba()
{
    if (__g_homepage_set.progress_bar_data < 100) {
        __g_homepage_set.progress_bar_data += 5;
    }
}
static void home_dec_pro_ba()
{
    if (__g_homepage_set.progress_bar_data >= 5) {
        __g_homepage_set.progress_bar_data -= 5;
    }
}
ZDP_REGISTER_CMD("home_add_proba", home_add_pro_ba);
ZDP_REGISTER_CMD("home_dec_proba", home_dec_pro_ba);
```

在更改控件值的回调函数中，可以加上限定值范围的判断，以免设置的值超出控件显示范围：

添加协议解析

虽然协议解析器限定了命令的长度，但命令所带的数据可以随命令一起发送：

XX XX XX XX…(命令长度PARSE_CMD_LEN)	XX（数据）

可以使用cmd_parsing_example.c中提供的接口get_cmd_data获取数据的地址及长度，数据长度不作限制；

```
static void demo_set_label()
{
    char* cmd_data;
    int data_len;
    get_cmd_data(&cmd_data, &data_len);
    if (data_len == 1) {
        if ((cmd_data[0] >= 0) && cmd_data[0] <= 5) {
            __g_demopage_set.label_data = cmd_data[0];
        }
    }
}
ZDP_REGISTER_CMD("demo_set_label", demo_set_label);
```

PC端调试

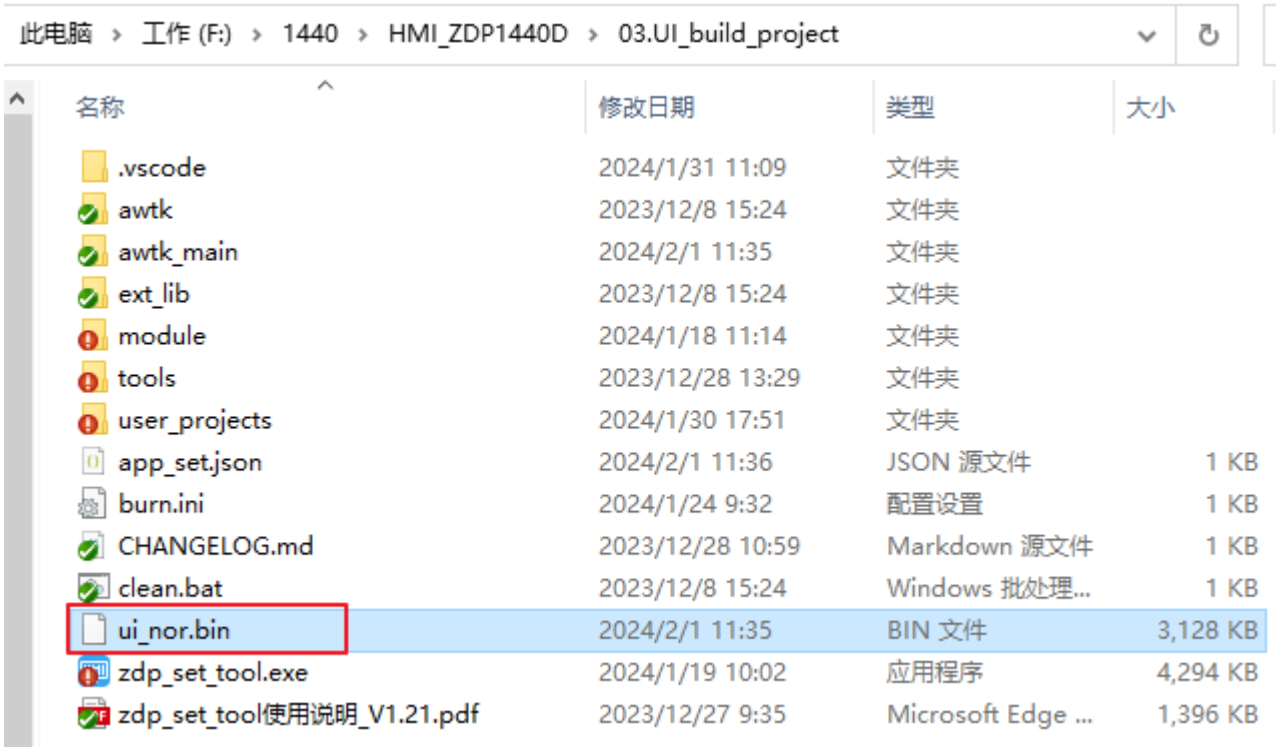
对应命令的回调函数实现完成后，可以在designer中编译&模拟运行，测试验证UI。

交叉编译

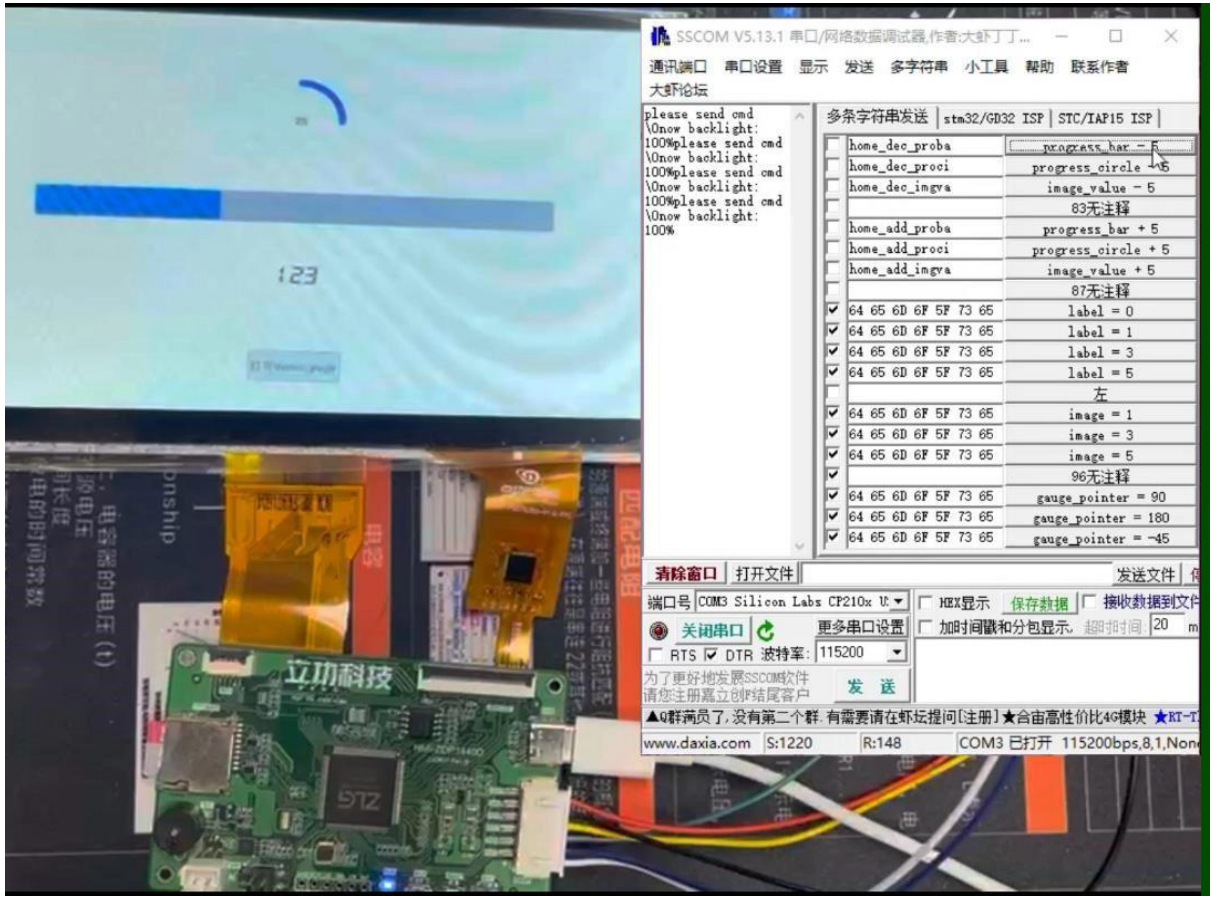
选中AwtkApp_demo，点击生成固件，等待打包完成。

验证固件

将生成ui_nor.bin的拷贝到TF卡中。随后把TF卡插入ZDP14x0板子上，按住KEY按钮，再按一下RST按钮，等待屏幕显示升级进度条时松开KEY按钮。当屏幕显示升级完成时将自动复位，届时将运行TF卡中的UI固件。



验证固件


SSCOM V5.13.1 串口/网络数据调试器,作者:大虾丁丁...
通讯端口 串口设置 显示 发送 多字符串 小工具 帮助 联系作者
大虾论坛
please send cmd
\Onow backlight:
100%please send cmd
\Onow backlight:
100%please send cmd
\Onow backlight:
100%please send cmd
\Onow backlight:
100%
多条字符串发送 | stm32/GD32 ISP | STC/IAP15 ISP
home_dec_proba progress_bar - 5
home_dec_proci progress_circle - 5
home_dec_imgva image_value - 5
83无注释
home_add_proba progress_bar + 5
home_add_proci progress_circle + 5
home_add_imgva image_value + 5
87无注释
64 65 6D 6F 5F 73 65 label = 0
64 65 6D 6F 5F 73 65 label = 1
64 65 6D 6F 5F 73 65 label = 3
64 65 6D 6F 5F 73 65 label = 5
左
64 65 6D 6F 5F 73 65 image = 1
64 65 6D 6F 5F 73 65 image = 3
64 65 6D 6F 5F 73 65 image = 5
96无注释
64 65 6D 6F 5F 73 65 gauge_pointer = 90
64 65 6D 6F 5F 73 65 gauge_pointer = 180
64 65 6D 6F 5F 73 65 gauge_pointer = -45
清除窗口 打开文件 发送文件
端口号 COM3 Silicon Labs CP210x U
关闭串口 更多串口设置
RTS DTR 波特率: 115200
为了更好地发展SSCOM软件
请您注册嘉立创P结尾客户
发 送
HEX显示 保存数据 接收数据到文件
加时间戳和分包显示, 超时时间: 20 m
www.daxia.com S:1220 R:148 COM3 已打开 115200bps,8,1,None
立功科技

FAQ

Q1：UI 很简单，素材也很少，但是打包出来的固件特别大?

A1：使用 AWTK Designer 创建 UI 项目后，需将资源的打包方式更改为：“仅文件”；同时检 查字体文件是否有裁剪。

Q2： UI 运行过程中，出现图片位置显示白色，或 AWTK 线程停止运行?

A2-1：有可能是由于内存不足导致的，内存分配失败时，参考 https://awtk.zlg.cn/docs/awtk_docs/HowTo/out_of_memory.html ，建议注册 EVT_LOW_MEMORY 事件和 EVT_OUT_OF_MEMORY 事件的回调函数，（若通过上 位机添加过虚拟串口和命令解析器，则已默认直接添加了该回调）若没有在回调函数中 做出对应的处理，可能产生的结果将不可控：背景图片或控件显示白色，甚至 AWTK 线 程停止运行；

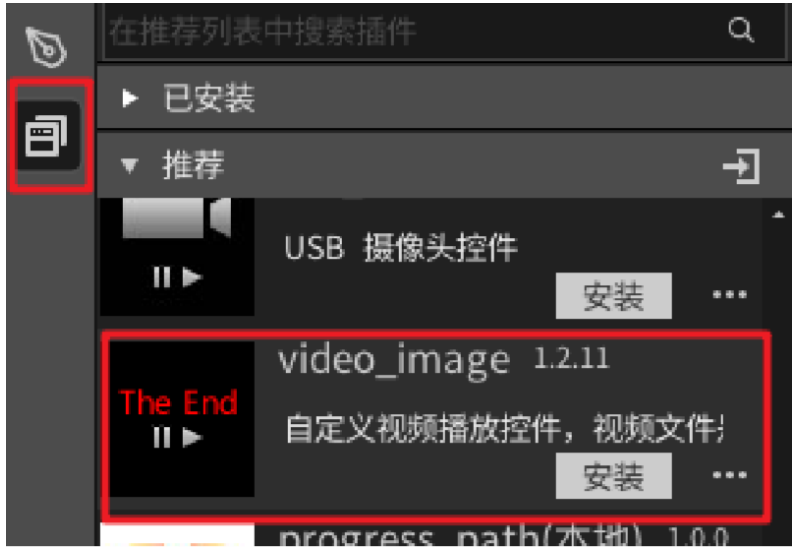
A2-2. 避免使用窗口动画，实现窗口动画需要额外使用两个屏幕 framebuffer；

A2-3.使用 Dialog 窗口时，避免使用对话框高亮，会额外使用至少一个屏幕 framebuffer。 AVDD_3V是音频模拟电源输出，外部需要加去耦电容。该路电源是需要程序开启后才有电源输出。

FAQ

Q3： UI 中使用了 GIF 图片，PC 端模拟显示正常，但是在 HMI 设备上显示不出来?

A3：在 AWTK 中，显示 GIF 图片的过程是先将 GIF 解析成一帧帧的位图，再将它们拼 接起来加载到内存中，比如一个分辨率为 480 * 480 的 GIF 图片，其中包含 120 帧图像， 需要将其解析为 16 位色的 bitmap 显示到 LCD 上，那么就至少需要 480 * 480 * 120 * 2 大小的内存，约为 52.7 M。针对这一问题，建议使用 AWTK 提供的自定义控件 video_image 来显示 GIF 图片，详见图 3.2。



FAQ

Q4： 在 HMI 设备中播放序列帧图片卡顿?

A4：在 HMI 设备中，播放 jpg/png 序列帧建议使用 video-image 自定义控件，因为软件解析 jpg/png 图片的速度是很慢的，特别在播放高分辨率的序列帧的时候特别明显。video_image 控件采用帧间差异的图像算法，把序列帧压缩为一个自定义的视频文件，这种方法的原理本 质上就是空间换时间的策略，虽然整体视频文件的大小会比 JPG 文件（PNG 文件）组成的 序列帧要大，但是会比位图文件小很多。

Q5：在 HMI 设备中旋转图片卡顿?

A5：芯片硬件加速仅支持90°、180°和270°的加速旋转，其余角度需要用软件旋转 。图片旋转对性能影响非常大，建议预先制作好 0~90°旋转的图片，91~359°的图片可以由 0~90°的图片水平/垂直翻转得到。

Q6：切换窗口的时候需不需要关闭之前的窗口？关闭和不关闭有什么影响?

A6：窗口 1 切换窗口 2 时可选择两种方式：关闭窗口 1 和不关闭窗口 1。关闭窗口 1 可节省 部分内存，但切回窗口 1 时，会重新打开窗口 1，页面控件会显示 xml 文件中配置的内容， 也就是说在之前运行过程中更改了窗口 1 中控件的属性会丢失，需重新更改。

Dream come true with professionalism and dedica

专业·专注成就梦想

测试
方案

干货
文章

精彩
活动

行业
热点

THANKS!

广州立功科技股份有限公司
GZLG Technology Corp., Ltd.

广州市天河区思成路43号ZLG致远电子大厦
Hotline：400-888-2705
www.zlgmcu.com