



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

- 01 AWTK基本介绍
- 02 基本控件使用介绍
- 03 常用函数介绍
- 04 创建简单UI
- 05 PC端验证
- 06 下载到ZDP14x0

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

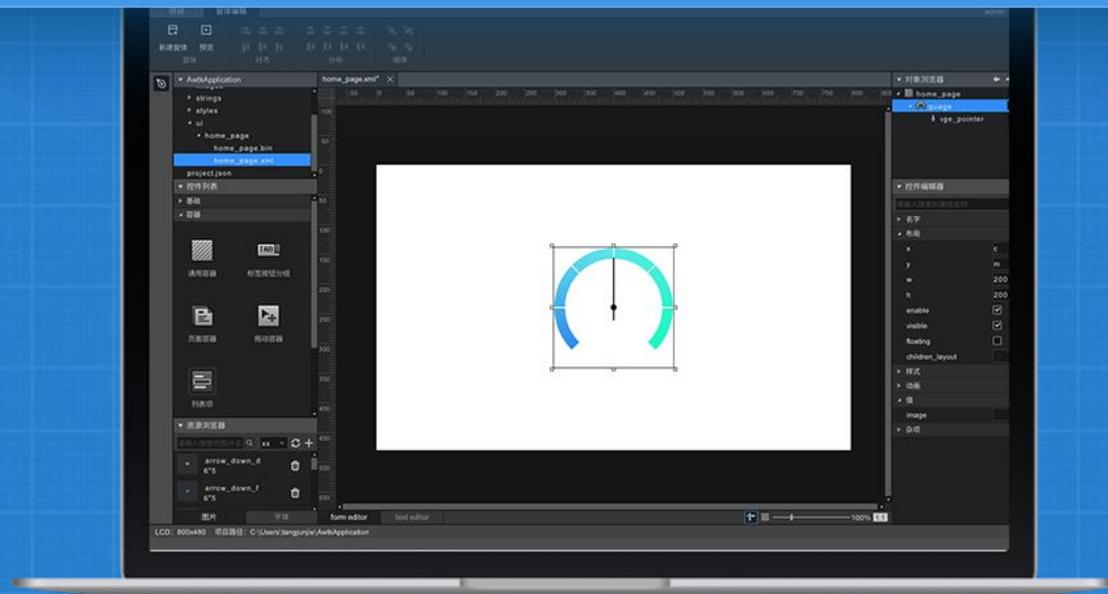
样式管理

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)

Guangzhou ZHIYUAN Electronics Co.,Ltd. & ZLGMCU
Guangzhou ZHIYUAN Electronics Co.,Ltd. & ZLGMCU
Floor 2, Building No.7, Huangzhou Ind... http://www.zlg.cn/ software@embedtools.com

Repositories 95 Packages 3 People 3 Projects

- awtk-mvvm**
Model-View-ViewModel for AWTK
mvvm mvvm-c awtk c-mvvm
C LGPL-2.1 18 35 3 0 0 Updated 5 hours ago
- awtk**
AWTK = Toolkit AnyWhere(为嵌入式、手机和PC打造的通用GUI系统)
gui stm32 rt-thread zephyr liteos sylxos djiyos
C++ LGPL-2.1 569 1,933 125 1 1 Updated 5 hours ago
- awtk-mobile-plugins**
用于访问手机平台 (android/ios) 原生服务的插件(如分享、登录、相机、扫描二维码和蓝牙等)
android ios awtk
Java LGPL-2.1 0 2 0 0 0 Updated 6 hours ago
- awtk-android**
awtk-android port
android awtk
Java 4 4 0 0 0 Updated 4 days ago
- awtk-jerrycript**
awtk javascript bindings.
gui jerrycript awtk
JavaScript LGPL-2.1 4 15 2 0 0 Updated 5 days ago
- awtk-ios**
awtk ios
ios awtk
CMake LGPL-2.1 0 3 0 0 0 Updated 10 days ago
- awtk-widget-store**
AWTK custom widgets repository
C 0 0 0 0 0 Updated 5 days ago
- awtk-widget-qr**
qrcode widget, generated by awtk-widget-generator
qrcode awtk
C LGPL-2.1 0 1 0 0 0 Updated 5 days ago
- awtk-jerryscript**
awtk javascript bindings.
gui jerrycript awtk
JavaScript LGPL-2.1 4 15 2 0 0 Updated 5 days ago
- awtk-ios**
awtk ios
ios awtk
CMake LGPL-2.1 0 3 0 0 0 Updated 10 days ago
- awtk-widget-table-view**
awtk-widget-table-view
C LGPL-2.1 0 1 1 0 0 Updated 12 days ago
- awtk-media-player**
media player for awtk
player media awtk
C 3 7 0 0 0 Updated 12 days ago
- awtk-widget-cache-view**
cache drawings of children widgets
awtk
C 0 0 0 0 0 Updated 16 days ago
- awtk-widget-code-edit**
code editor for awtk
C++ LGPL-2.1 0 0 0 0 0 Updated 16 days ago



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：

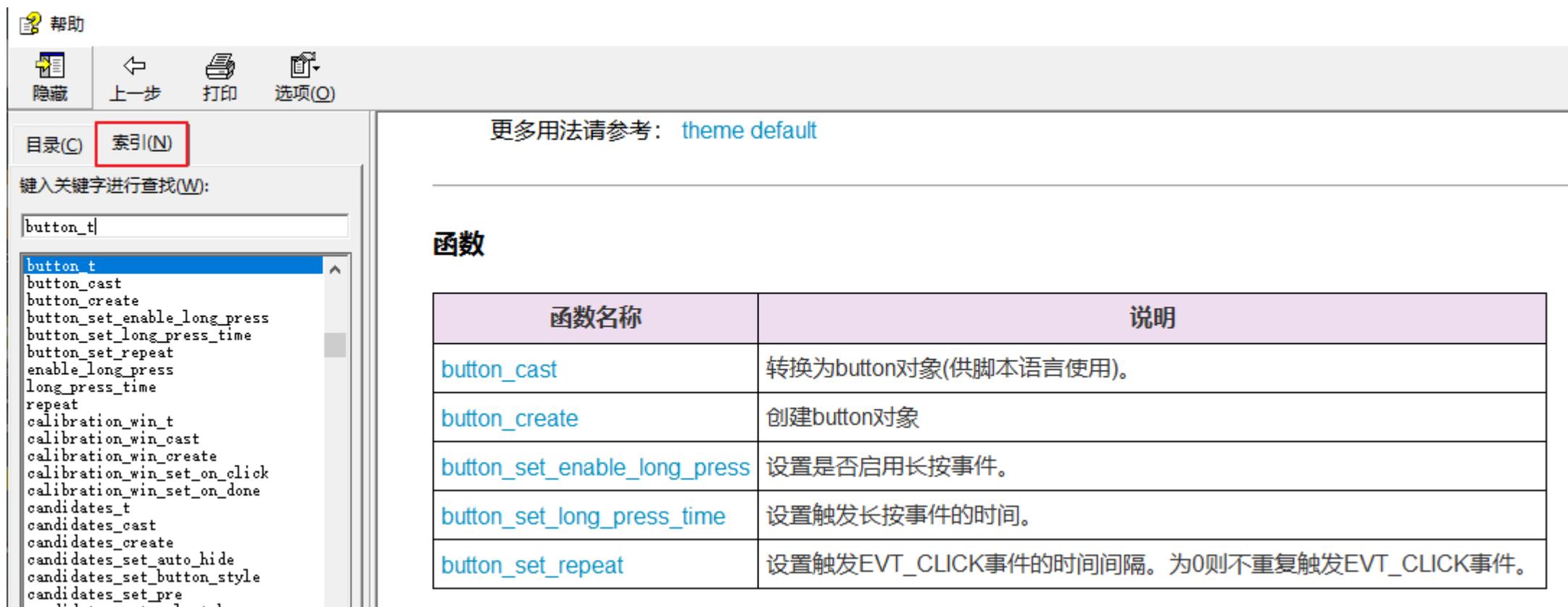
	语言	大小	效率	炫酷	跨平台	开源	免费
	C、Js ✓	小 ✓	高 ✓	是 ✓	全平台 ✓	是 ✓	是 ✓
	C	小 ✓	高 ✓	否	嵌入式平台	否	部分平台
	C++	大	低	一般	中高端平台	是 ✓	是 ✓
	C++	较小	较高	是 ✓	嵌入式平台	否	部分平台

AWTK支持控件



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：



The screenshot shows the AWTK API help file interface. The search bar contains 'button_t', and the results list includes 'button_t', 'button_cast', 'button_create', 'button_set_enable_long_press', 'button_set_long_press_time', 'button_set_repeat', 'enable_long_press', 'long_press_time', 'repeat', 'calibration_win_t', 'calibration_win_cast', 'calibration_win_create', 'calibration_win_set_on_click', 'calibration_win_set_on_done', 'candidates_t', 'candidates_cast', 'candidates_create', 'candidates_set_auto_hide', 'candidates_set_button_style', and 'candidates_set_pre'. The 'button_t' entry is selected. The main content area shows the heading '更多用法请参考: [theme default](#)' and a table of functions.

函数名称	说明
button_cast	转换为button对象(供脚本语言使用)。
button_create	创建button对象
button_set_enable_long_press	设置是否启用长按事件。
button_set_long_press_time	设置触发长按事件的时间。
button_set_repeat	设置触发EVT_CLICK事件的时间间隔。为0则不重复触发EVT_CLICK事件。

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。



The screenshot shows a software interface with a search bar containing 'widget_t'. Below the search bar is a list of functions. To the right of the list is a table with two columns: '函数名称' (Function Name) and '说明' (Description). The table lists various widget functions and their purposes.

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

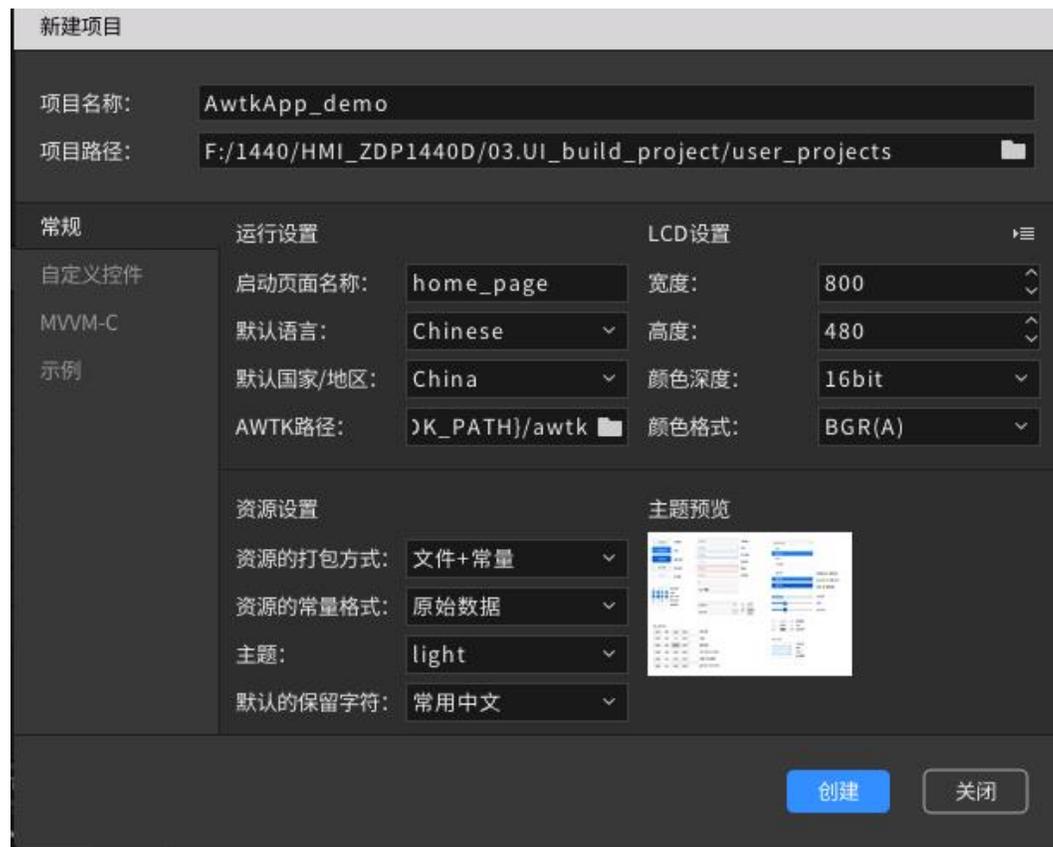
常用函数

<code>widget_set_prop_(xxxx)</code>	说明	<code>widget_set_style_(xxxx)</code>	说明
<code>widget_set_prop_bool</code>	设置布尔格式的属性。	<code>widget_set_style_color</code>	设置颜色类型的style。
<code>widget_set_prop_int</code>	设置整数格式的属性。	<code>widget_set_style_int</code>	设置整数类型的style。
<code>widget_set_prop_str</code>	设置字符串格式的属性。	<code>widget_set_style_str</code>	设置字符串类型的style。

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

新建UI工程

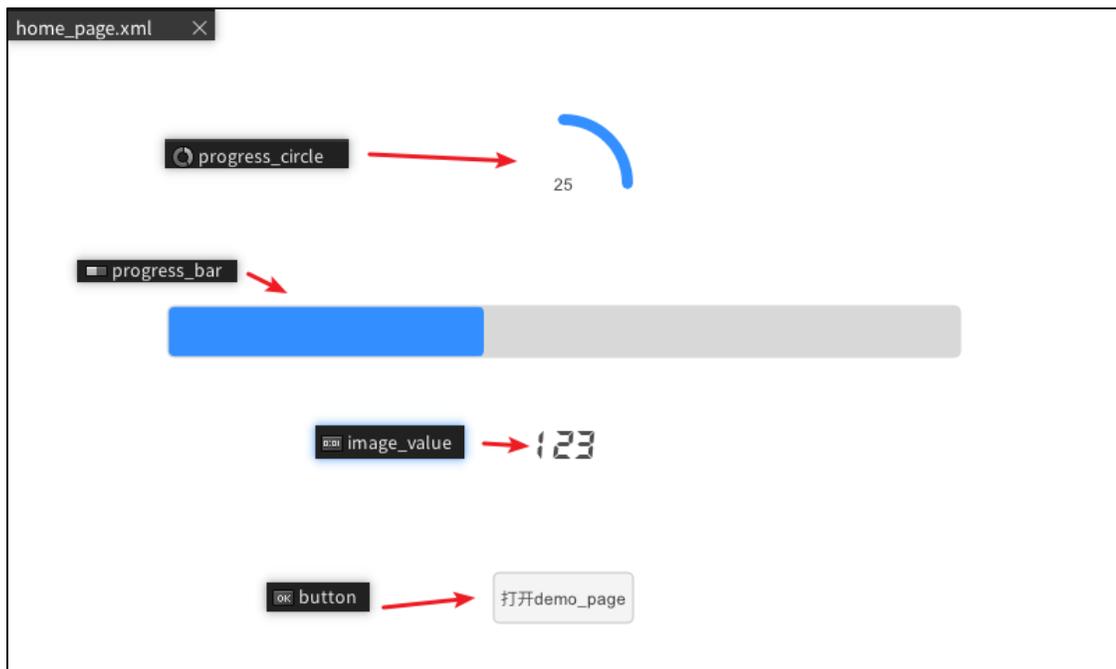
根据屏幕大小，在user_projects中创建一个UI。



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”；

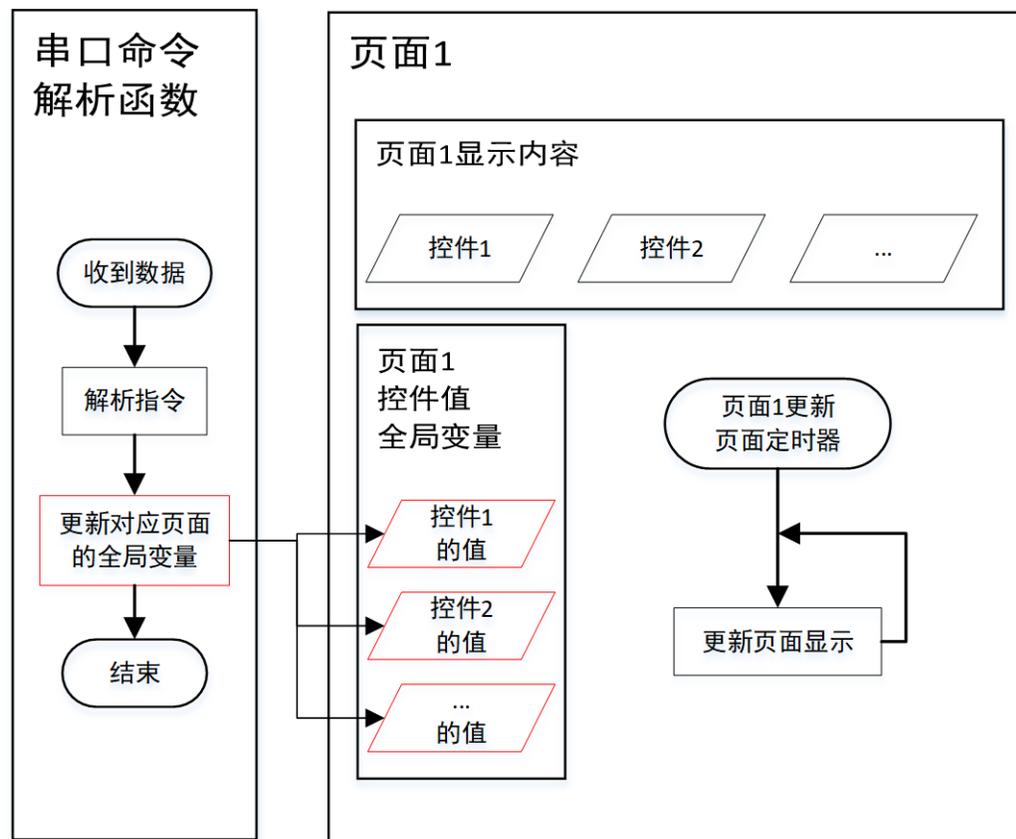
默认仅使能PC端虚拟串口，如果需要用到命令解析器，需要勾选“使能命令解析器”。



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。

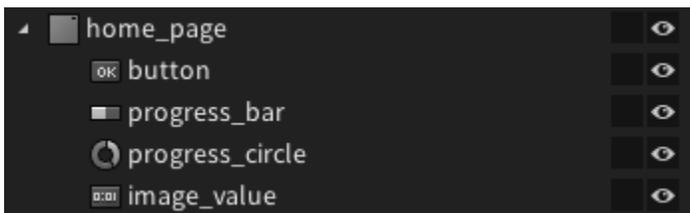
- 避免创建多个EVT_USER_RX_DATA事件的回调函数;
- 页面定时器使用widget_add_idle/widget_add_timer创建，关闭页面时可以自动销毁对应定时器;



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。



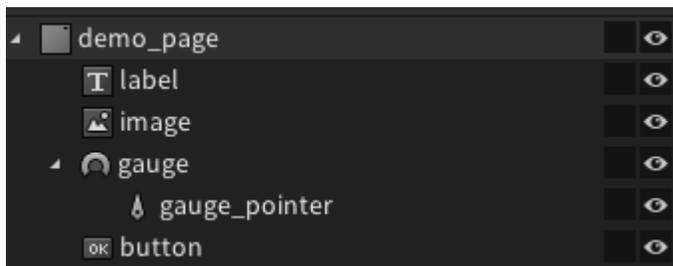
```
C struct_define.h ...\AwtkApp_demo\... X C navigator.h
user_projects > AwtkApp_demo > src > C struct_define.h > ...
1  #ifndef __STRUCT_DEFINE__
2  #define __STRUCT_DEFINE__
3  #include "awtk.h"
4
5
6  /* home_page中显示的内容 */
7  typedef struct _homepage_data_t {
8      uint8_t    progress_bar_data;
9      uint8_t    progress_circle_data;
10     uint16_t   image_value_data;
11 }homepage_data_t;
12
13
14 #endif // __STRUCT_DEFINE__
15
```

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"};
```



```
C struct_define.h 1 x
src > C struct_define.h > ...
13  /* demo_page中显示的内容 */
14  typedef struct _demopage_data_t {
15      uint8_t    label_data;
16      uint8_t    image_data;
17      int16_t    gauge_pointer_data;
18  }demopage_data_t;
19
```

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。

```
C struct_define.h × C navigator.h ×
user_projects > AwtkApp_demo > src > common > C navigator.h > navigator_to(const char *)
1  #ifndef APP_NAVIGATOR_H
2  #define APP_NAVIGATOR_H
3
4  #include "struct_define.h"
5  #include "tkc/types_def.h"
6  #ifdef WIN32
```

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 x
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 x
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 x
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 X
ser_projects > AwtkApp_demo > src > pages > C home_page.c > refresh_homepage(const idle_info_t*)
19 static widget_t* progress_bar = NULL;
20 static widget_t* progress_circle = NULL;
21 static widget_t* image_value = NULL;
22 /**
23  * 初始化窗口的子控件
24  */
25 static ret_t visit_init_child(void* ctx, const void* iter) {
26     widget_t* win = WIDGET(ctx);
27     widget_t* widget = WIDGET(iter);
28     const char* name = widget->name;
29
30     // 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
31     if (name != NULL && *name != '\0') {
32         if (tk_str_eq(name, "button")) {
33             widget_on(widget, EVT_CLICK, on_button_click, win);
34         } else if (tk_str_eq(name, "progress_bar")) {
35             progress_bar = widget;
36         } else if (tk_str_eq(name, "progress_circle")) {
37             progress_circle = widget;
38         } else if (tk_str_eq(name, "image_value")) {
39             image_value = widget;
40         }
41     }
42     return RET_OK;
43 }
```

```
C home_page.c 2 X
src > pages > C home_page.c > refresh_homepage(const idle_info_t*)
40
41 static ret_t refresh_homepage(const idle_info_t* d)
42 {
43     progress_bar_set_value(progress_bar, __g_homepage_set.progress_bar_data);
44     progress_circle_set_value(progress_circle, __g_homepage_set.progress_circle_data);
45     image_value_set_value(image_value, __g_homepage_set.image_value_data);
46
47     return RET_REPEAT;
48 }
```

添加更新页面代码

demo_page同理。

```
19 static widget_t* label      = NULL;
20 static widget_t* image      = NULL;
21 static widget_t* gauge_pointer = NULL;
22 /**
23  * 初始化窗口的子控件
24  */
25 static ret_t visit_init_child(void* ctx, const void* iter) {
26     widget_t* win = WIDGET(ctx);
27     widget_t* widget = WIDGET(iter);
28     const char* name = widget->name;
29
30     // 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
31     if (name != NULL && *name != '\0') {
32         if (tk_str_eq(name, "button")) {
33             widget_on(widget, EVT_CLICK, on_button_click, win);
34         } else if (tk_str_eq(name, "label")) {
35             label = widget;
36         } else if (tk_str_eq(name, "image")) {
37             image = widget;
38         } else if (tk_str_eq(name, "gauge_pointer")) {
39             gauge_pointer = widget;
40         }
41     }
42     return RET_OK;
43 }
```

```
C demo_page.c 2 x
src > pages > C demo_page.c > refresh_demopage(const idle_info_t *)
40 static ret_t refresh_demopage(const idle_info_t* d)
41 {
42     static const char *label_str[] = {"label", "一", "二", "三", "四", "五"};
43     static const char *image_str[] = {"num0", "num1", "num2", "num3", "num4", "num5"};
44
45     widget_set_text_utf8(label, label_str[__g_demopage_set.label_data]);
46     image_base_set_image(image, image_str[__g_demopage_set.image_data]);
47     gauge_pointer_set_angle(gauge_pointer, __g_demopage_set.gauge_pointer_data);
48     return RET_REPEAT;
49 }
```

添加协议解析

在cmd_parsing_example.c中使用
ZDP_REGISTER_CMD注册命令:

- home_add_proba
- home_add_proci
- home_add_imgva
- home_dec_proba
- home_dec_proci
- home_dec_imgva

- demo_set_label
- demo_set_image
- demo_set_gauge

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:

- progress_bar的值+5
- progress_circle的值+5
- image_value的值+5
- progress_bar的值-5
- progress_circle的值-5
- image_value的值-5

- 设置demo_page中label的值
- 设置demo_page中image的值
- 设置demo_page中gauge的值

指令

指令+数据

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：

XXXX XXXX...(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。



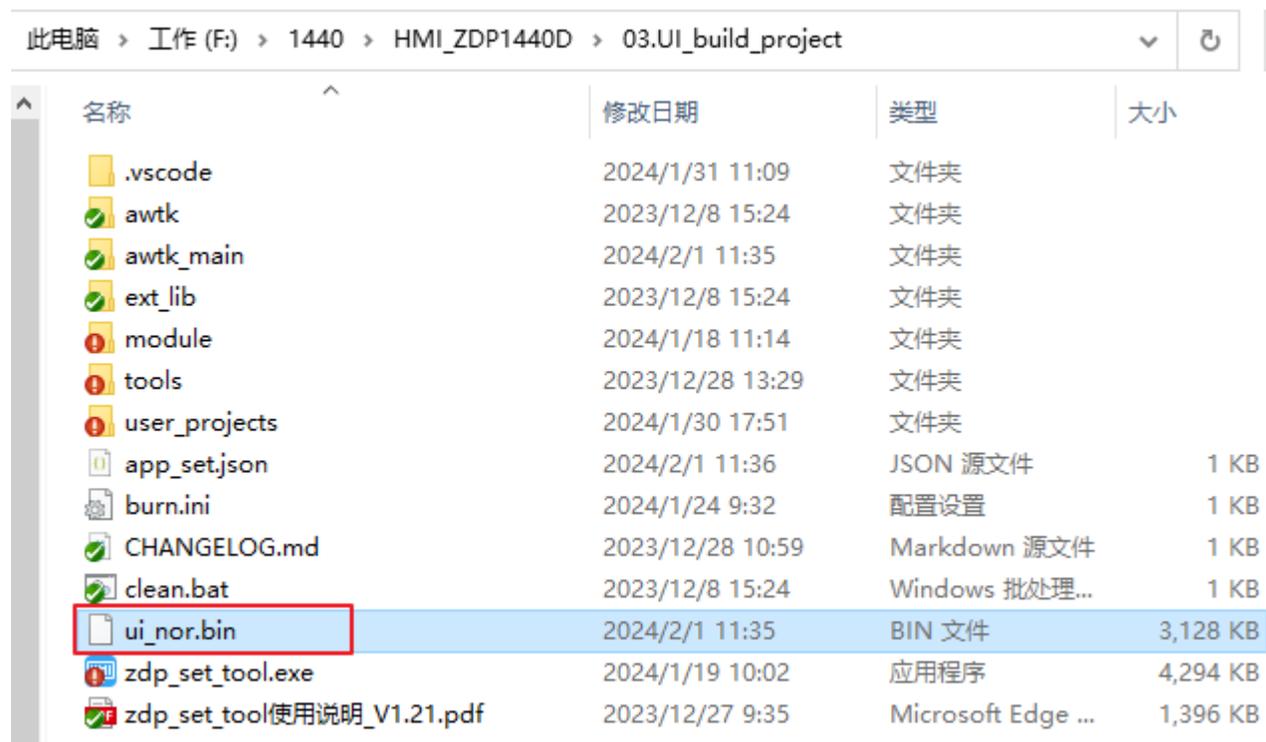
交叉编译

选中AwtkApp_demo, 点击生成固件, 等待打包完成。



验证固件

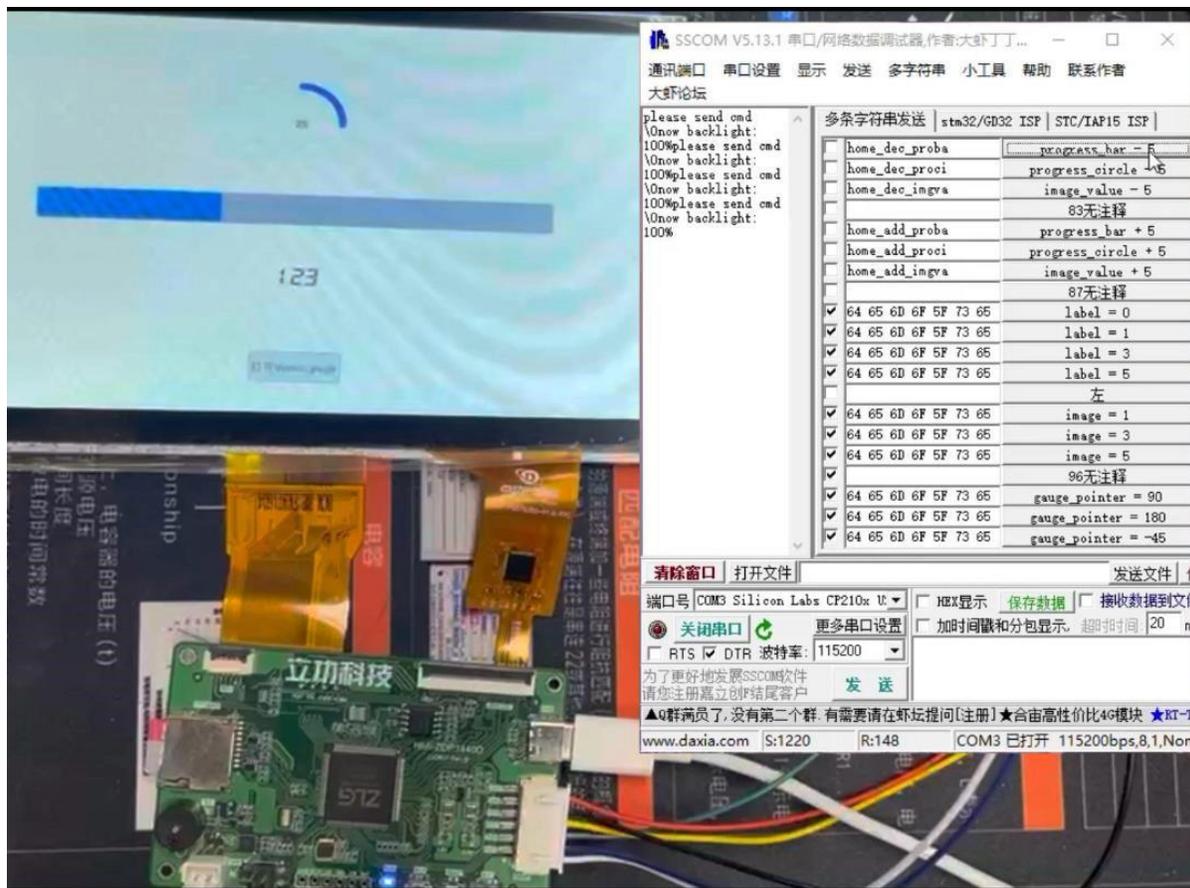
将生成ui_nor.bin的拷贝到TF卡中。随后把TF卡插入ZDP14x0板子上，按住KEY按钮，再按一下RST按钮，等待屏幕显示升级进度条时松开KEY按钮。当屏幕显示升级完成时将自动复位，届时将运行TF卡中的UI固件。



此电脑 > 工作 (F:) > 1440 > HMI_ZDP1440D > 03.UI_build_project

名称	修改日期	类型	大小
.vscode	2024/1/31 11:09	文件夹	
awtk	2023/12/8 15:24	文件夹	
awtk_main	2024/2/1 11:35	文件夹	
ext_lib	2023/12/8 15:24	文件夹	
module	2024/1/18 11:14	文件夹	
tools	2023/12/28 13:29	文件夹	
user_projects	2024/1/30 17:51	文件夹	
app_set.json	2024/2/1 11:36	JSON 源文件	1 KB
burn.ini	2024/1/24 9:32	配置设置	1 KB
CHANGELOG.md	2023/12/28 10:59	Markdown 源文件	1 KB
clean.bat	2023/12/8 15:24	Windows 批处理...	1 KB
ui_nor.bin	2024/2/1 11:35	BIN 文件	3,128 KB
zdp_set_tool.exe	2024/1/19 10:02	应用程序	4,294 KB
zdp_set_tool使用说明_V1.21.pdf	2023/12/27 9:35	Microsoft Edge ...	1,396 KB

验证固件



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。



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

GZLG

