

ROBOMASTER



Tello Talent 扩展模块开发指南

Programming Guide of Tello Talent

欢迎使用 **Tello Talent 扩展模块** 进行开发

本指南将带领您
从上手实践中快速全面了解
TT 扩展模块的使用及注意事项。

开始开发前的必要知识 - TT扩展模块与TT无人机的组合体控制模式 搭建开发环境 (Windows)

Arduino IDE的安装

Arduino Hardware库的安装

Arduino RMTT Library的安装

在Arduino IDE中搭建您的项目

脱离Arduino IDE的外部编译及上传模式 (外部部署)

初次使用

环境路径配置完毕后的使用方法

初识Tello Talent 扩展模块

机顶全彩LED

模块组成

模块接口

控制样例 - 呼吸灯

红蓝8*8 LED矩阵

模块组成

模块接口

点阵取模及其使用方式

控制样例 - 数字显示

ToF (Time of Flight) 测距传感器

模块组成

控制样例 - 在点阵屏上显示距离

与无人机通讯 - TT_Protocol模块

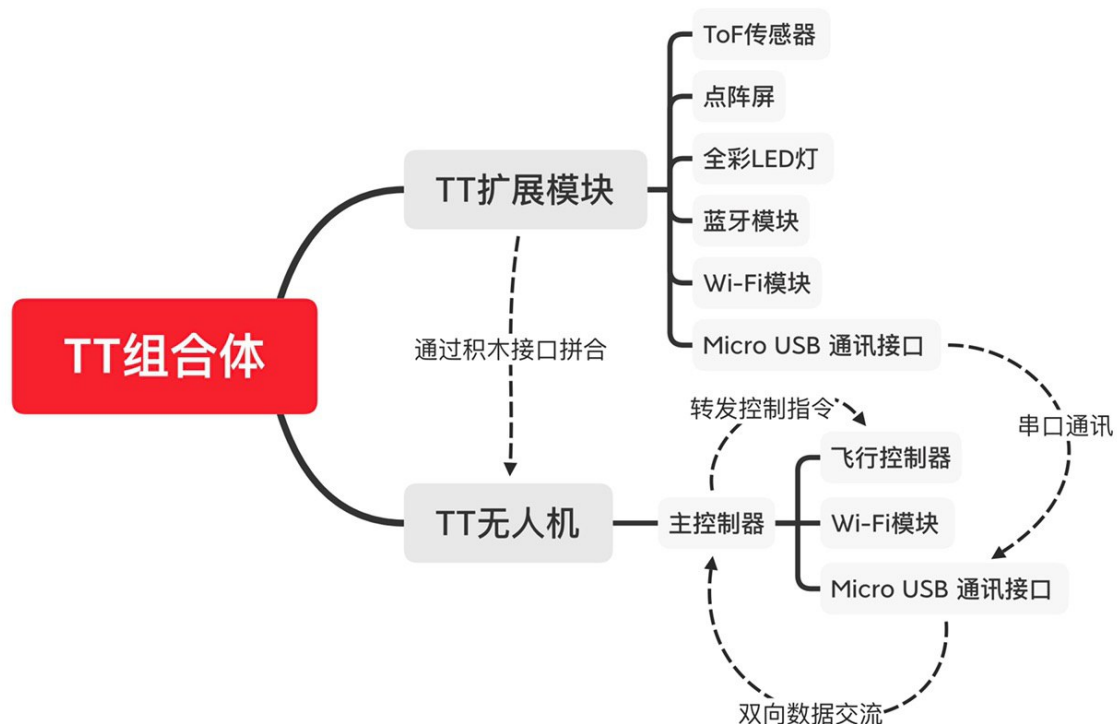
模块接口

进阶开发

工程模板 - GameSir盖世小鸡手柄操控

工程模板 - 与Tello EDU APP互动

开始开发前的必要知识 - TT扩展模块与TT无人机的组合体控制模式



如果您需要实现对TT无人机的控制，则您始终需要将TT无人机与TT扩展模块分开看待。

我们首先从TT无人机的视角来看待TT扩展模块。

TT无人机有自己的飞行控制器，它通过获取飞机自身的陀螺仪等传感器信息来控制飞机的四个空心杯电机以控制飞机的物理运动。来自不同来源（比如手机、手柄）的**控制指令通过TT无人机的主控制器进一步下发到飞行控制器**以完成具体的飞行指令，而飞机信息（比如电量、当前高度等）的返回也是由TT无人机的主控制器完成。

TT扩展模块对无人机的控制也如手机、手柄一样，是通过向TT无人机的主控制器传递控制指令来实现的。而这个**具体的通信过程经由无人机的机载Micro USB接口通过串口通讯来实现**，同时这个接口也会对**TT扩展模块进行供电**。

当我们转换视角，**从TT扩展模块的角度来看**，它便相当于附着于TT无人机之上的**外部大脑**。它不仅拓展了TT无人机本身的感知能力（前向ToF传感器），还丰富了TT无人机的功能性。同时，您也可以**通过向扩展模块烧写自己的程序来使TT组合体完成仅仅依靠TT无人机本身难以完成的复杂飞行任务**。通过与TT无人机通讯获取无人机本身的传感器信息，结合扩展模块本身的传感器信息，您可以实现丰富的飞行控制功能，比如为TT组合体实现前向避障等功能。更多精彩玩法，亟待您的探索！

搭建开发环境（Windows）



```
void setup() {  
  // put your setup code here, to run once:  
  Serial.begin(115200);  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
  digitalWrite(LED_BUILTIN, HIGH);  
  delay(1000);  
  digitalWrite(LED_BUILTIN, LOW);  
  delay(1000);  
}  
  
xTaskCreateUniversal(control_task, "control_task", 23, 18);  
//sdk_cmd_init();
```

使用

Arduino IDE

进行开发。



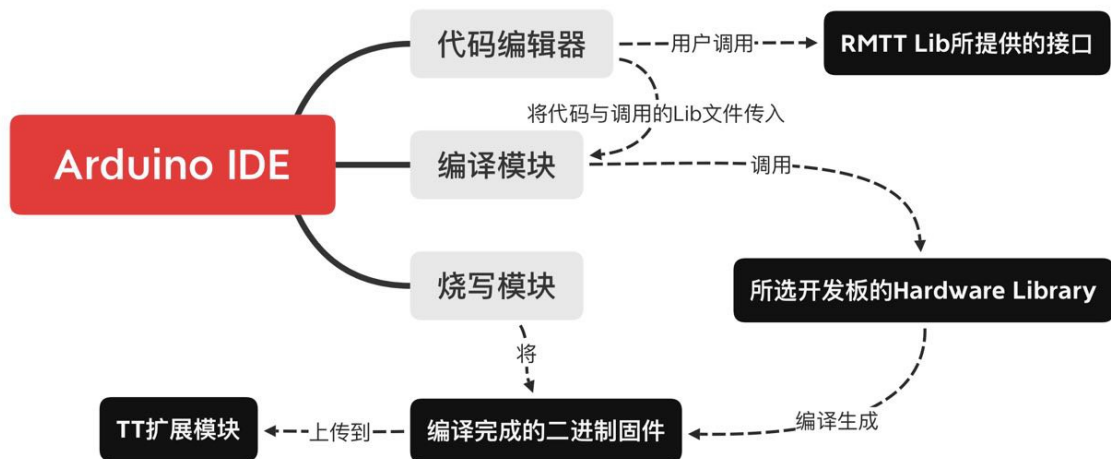
在本指南中，我们选择了Arduino IDE作为面向用户的编程环境。Arduino IDE作为Arduino开源项目的衍生产品，拥有**轻便、易于使用、集成度高以及便于扩展**的种种优秀特性，即使对编程几无了解的新手也可**快速上手**并且使用它构造自己的精彩项目。

在使用Arduino IDE开始对Tello Talent扩展模块的开发之前，**我们需要先对开发环境进行一些必要的配置**，以确保Arduino IDE可以加载面向Tello Talent扩展模块进行编译所必须的硬件工具包以及对应底层库文件。

Arduino IDE的安装

1. 如果您从未安装过Arduino IDE，请使用我们提供的Arduino集成版直接进行开发。
2. 如果您已经安装了Arduino IDE，请确保版本不低于1.8.12，并按照以下步骤添加RMTT扩展模块支持包。

下载链接：<https://www.dji.com/cn/robomaster-tt/downloads>



在Arudino IDE安装完成之后，我们需要进行开发环境的配置。上图是通过Arduino IDE对TT扩展模块进行开发的内部原理流程。我们可以看到，**独立于Arduino IDE之外的有两个部分**，分别是**Hardware Library（硬件库）**以及**RMTT Lib（Tello Talent拓展模块的功能接口库）**。这两个独立部分需要我们在**完成对Arduino IDE的安装之后进行独立的安装**，以使Arduino IDE可以对Tello Talent拓展模块进行集成化开发。

Arduino Hardware库的安装

请您在您的系统盘中（C盘）**找到并进入**如下目录：

C:\Users\您的用户名\Documents\Arduino

请注意，在某些版本的Windows系统中，“您的用户名”处将会被截取您实际用户名的前五个字符，比如

若您的实际用户名为Username，此时该目录为C:\Users\Usern\Documents\Arduino

您可以通过从C盘进入“用户”文件夹并找到您的账户名对应文件夹的方式来避免命名带来的文件路径问题。

在我们提供的文件包中，您可以看到"hardware"文件夹。将其复制到您进入的Arduino目录中，并合并(Merge)到该目录下同名文件夹内。至此，Hardware库的安装就完成了，接着请完成下一小节的安装。

Arduino RMTT Library的安装



添加 RMTT Library。



此时打开Arduino IDE，在顶部菜单栏的项目-加载库-添加.ZIP库中找到我们提供的文件包中的“RMTT_Libs.7z”文件。点击确定，稍等一会儿，此后RMTT Library应该已经被正确导入Arduino IDE。

在Arduino IDE中搭建您的项目



对开发板进行选择 同时选择对应端口。

在开始您的第一个项目之前，记得在Arduino IDE的菜单栏-工具-开发板中选择“RMTT Module”，之后在下方的“端口”处选择您的TT扩展模块对应的端口。

如果您不知道您的TT模块对应哪一端口，请打开Windows设备管理器，在

> 端口 (COM 和 LPT) 中找到类似于下图的设备：

该设备名称后方的括号内便是您所需要的端口号，在Arduino IDE中选择即可。


上传， 编译和烧写



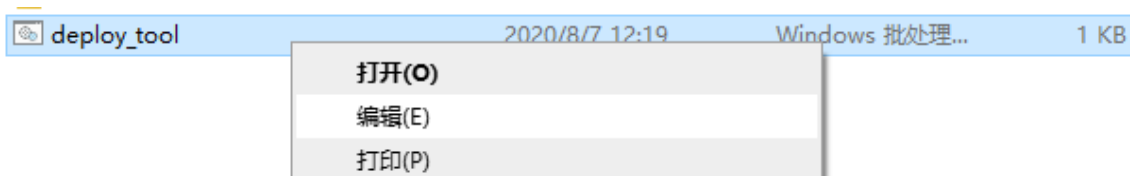
在Arduino IDE的菜单栏下的工具栏中，左侧两个按钮从左到右分别是“仅编译”以及“编译+上传”按钮。通常我们仅需要点击第二个按钮，Arduino IDE会在完成编译工作之后自动将编译之后的二进制文件上传（烧写）到TT扩展模块中。

通常，Arduino IDE在编译新项目时需要较长时间（三到五分钟，视计算机性能而定）来编译RMTT核心库的内容，而此后的编译过程才会调用编译缓存进行二次编译。如果您认为编译速度太为缓慢，下一小节将会提到一个解决方案。

脱离Arduino IDE的外部编译及上传模式（外部部署）

在TT文件包中您可以看到“外部部署工具”文件夹，进入该文件夹，您会看到  `deploy_tool` 也即部署工具。

初次使用



您需要在右键菜单中选择“编辑”，将`deploy_tool.bat`文件中第一行的环境路径（“<arduino_loc>”以及“<lib_loc>”）分别改为您的Arduino程序路径以及Hardware库的安装路径（在上文对应章节提到）。

```
@echo off
call tool_core\build.bat <arduino_loc> <lib_loc> %1
call tool_core\flash_firmonly.bat %2 %3
```

更改完成之后，整个`deploy_tool.bat`文件内容应当类似于如下所示：

```
@echo off
call tool_core\build.bat D:\arduino-1.8.12 C:\Users\username\Documents\Arduino %1
call tool_core\flash_firmonly.bat %2 %3
```

环境路径配置完毕后的使用方法



在“外部部署工具”文件夹空白处按下SHIFT+右键，选择“在此处打开Powershell窗口”或是“在此处打开命令提示符（CMD）窗口”。

```
.\deploy_tool.bat <your_ino_file_location> <your_board_COM_NO>  
<your_project_name>
```

在出现的Powershell（或者CMD）窗口中输入以上命令。

请谨记，在输入命令前将三个括号("<>")整体替换成对应内容，分别是“需要编译的ino文件地址”、“需要上传的COM口号”、“你的ino文件名称”

例如：

```
.\deploy_tool.bat D:\MyProj\first.ino COM6 first
```

如此，外部部署工具便可以自动使用之前已经编译缓存好的核心库内容，以此来加速编译进程。

初识Tello Talent 扩展模块

体积小？
功能不少。
ROBOMASTER



全彩内置 LED
Tof 测距传感器
8*8 红蓝点阵屏

本模块是Tello Talent的扩展模块，通过积木接口与TT固定，并通过机载Micro USB接口对本模块进行供电与数据交换。

本模块搭载了全彩LED灯、ToF传感器以及8*8红蓝LED点阵屏。通过我们的例子，您将了解如何在Arduino中编程控制本扩展模块，并且使用它打造自己的项目。

机顶全彩LED

机顶全彩LED模块

RGB LED Module



模块组成

本模块由红绿蓝三颗高亮度LED以及外部的柔光罩组成，我们可以通过控制不同颜色LED的亮度辅以外部柔光罩来达到控制外部观感颜色的目的。

模块接口

```
class RMTT_RGB{
public:
    static void Init();
    static void SetRed(uint32_t val, uint32_t valueMax = 255);
    static void SetBlue(uint32_t val, uint32_t valueMax = 255);
    static void SetGreen(uint32_t val, uint32_t valueMax = 255);
    static void SetRGB(uint32_t R, uint32_t G, uint32_t B, uint32_t valueMax
= 255);
};
```

常用函数名	传参	参数范围	参数释义	函数作用
Init	/	/	/	必要 - 初始化LED
SetRed	val	0-255	val 该通道灯珠亮度	设置红色灯珠亮度
SetBlue	val	0-255	val 该通道灯珠亮度	设置蓝色灯珠亮度
SetGreen	val	0-255	val 该通道灯珠亮度	设置绿色灯珠亮度
SetRGB	R, G, B	均为0- 255	R 红色灯珠亮度 G 绿色灯珠亮度 B 蓝色灯珠亮度	同时设置三个通道灯珠亮度

控制样例 - 呼吸灯

本模块的样例是使用RMTT_RGB控制全彩LED灯使其呈现呼吸灯效。

```

#include <RMTT_Libs.h>
#include <Wire.h>

#define BREATH_Hz (0.5)

#define BREATH_RED (0)
#define BREATH_BLUE (255)
#define BREATH_GREEN (255)

RMTT_RGB tt_rgb; // 实例化RMTT_RGB对象

bool breath_toggle = false; // 亮度改变方向
uint8_t breath_rate = 0; // Percentage of brightness 亮度百分比
int period = 10;

void setup() {
    wire.begin(27, 26); // 初始化I2C总线
    wire.setClock(400000);

    period = 1000 / BREATH_Hz; // 计算单个呼吸周期时长 单位ms
    tt_rgb.Init();
    tt_rgb.SetRGB(0, 0, 0);
}

void loop() {

    /* 亮度增或减, 注意breath_toggle的方向避免溢出 */
    breath_rate = breath_toggle ? breath_rate - 1 : breath_rate + 1;

    if ((breath_rate == 100) || (breath_rate == 0))
        breath_toggle = !breath_toggle; // 改变亮度增减方向

    tt_rgb.SetRGB(BREATH_RED * breath_rate / 100.0,
                  BREATH_GREEN * breath_rate / 100.0,
                  BREATH_BLUE * breath_rate / 100.0);

    delay(period / 100); // 延时一个呼吸周期的百分之一
}

```

红蓝8*8 LED矩阵

模块组成

本模块由64颗红色灯珠以及64颗蓝色灯珠组成，一共128颗灯珠。通过封装好的接口，您可以实现对LED矩阵每颗灯珠的色彩、亮度等控制。

模块接口

```

class RMTT_Matrix{
public:
    static void On();
    static void Off();
}

```

```

static void Init(uint8_t gcc);
static void SetGCC(uint8_t gcc);
static void SetLEDStatus(uint8_t cs, uint8_t sw, IS31FL3733_LED_STATE
state);
static void SetLEDPWM(uint8_t cs, uint8_t sw, uint8_t value);
static void SetAllPWM(uint8_t *val);

/*下方是一些内部函数*/
static uint8_t ReadCommonReg(uint8_t reg_addr);
static void WriteCommonReg(uint8_t reg_addr, uint8_t reg_value);
static void SetLEDMode(uint8_t cs, uint8_t sw, IS31FL3733_LED_MODE
mode);
static void ConfigABM(IS31FL3733_ABM_NUM n, IS31FL3733_ABM *config);
static void StartABM();
};

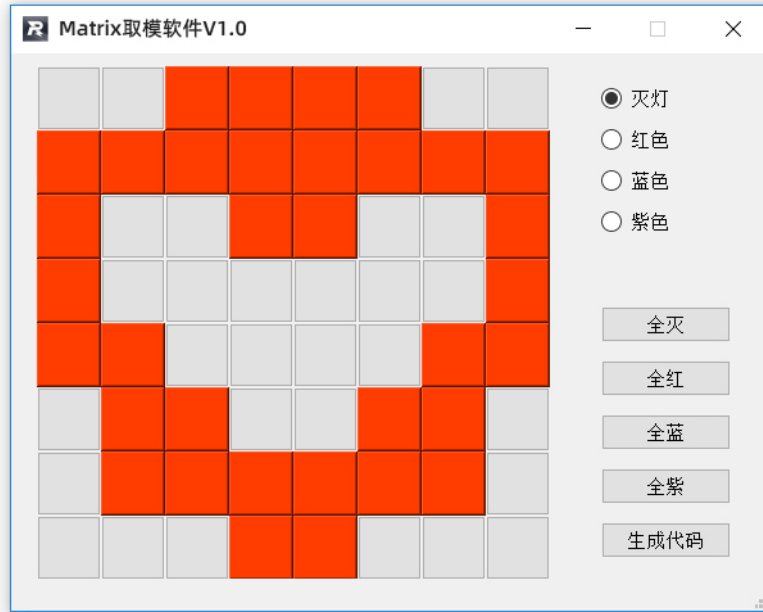
```

常用函数名	传参	参数范围	参数释义	函数作用
On	/	/	/	打开点阵
Off	/	/	/	关闭点阵
Init	gcc	0-127	gcc 全局电流控制	必须先初始化点阵
SetGCC	gcc	0-127	gcc 全局电流控制	设置全局电流（即控制亮度）
SetLEDStatus	cs, sw, state	0-8 / 0-16 / {0,1}	cs 横排数 sw 竖列数 state LED状态	控制所有LED的开关
SetLEDPWM	cs, sw, value	0-8 / 0-16 / 0-255	cs 横排数 sw 竖列数 value LED色值	控制所有LED的色值
SetAllPWM	*val	大小为128的一维数组	数组详细结构在下方展示	设置点阵图形

点阵取模及其使用方式

在点阵LED屏控制过程中，我们需要让该屏幕显示我们需要的内容。在嵌入式开发中，除非已有封装好的显示库，否则您通常需要自己编写程序，从最底层控制每个像素每颗LED的亮暗，才能达到显示我们想要的内容的效果。

这意味着，即使我们需要显示数字、文字等简单内容，也需要控制每颗LED的亮暗来达到对应效果。幸运的是，您无需亲自实现这个控制过程，只需使用取模软件，将需要显示的内容以简单的图形化方式绘制，即可从取模软件中获得自动生成的点阵数组代码。

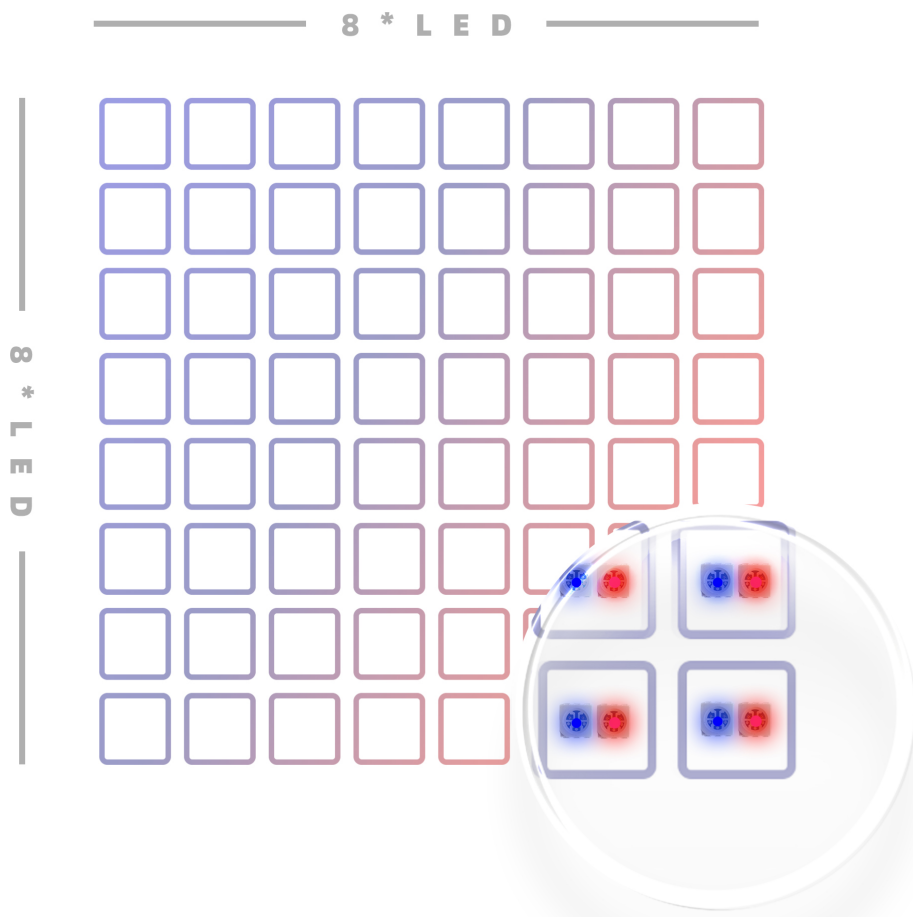


打开“点阵取模”文件夹下的matrix.exe，然后随心绘制您所需要的图形。点击 **生成代码**，即可得到对应的点阵数组。将其复制到您的项目中即可使用。

```
uint8_t matrix[] = {
    0, 0, 0, 0, 0, 255, 0, 255, 0, 255, 0, 255, 0, 0, 0,
    0,
    0, 255, 0, 255, 0, 255, 0, 255, 0, 255, 0, 255, 0, 255, 0,
    255,
    0, 255, 0, 0, 0, 0, 0, 255, 0, 255, 0, 0, 0, 0, 0,
    255,
    0, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    255,
    0, 255, 0, 255, 0, 0, 0, 0, 0, 0, 0, 0, 255, 0,
    255,
    0, 0, 0, 255, 0, 255, 0, 0, 0, 0, 0, 255, 0, 255, 0,
    0,
    0, 0, 0, 255, 0, 255, 0, 255, 0, 255, 0, 255, 0, 255, 0,
    0,
    0, 0, 0, 0, 0, 0, 0, 255, 0, 255, 0, 0, 0, 0, 0,
    0,
    };
```

取模软件生成的代码如图。看起来让人一头雾水？其实它的结构并没有那么复杂。

让我们先来看看点阵LED屏的硬件组成形式。



Talent扩展模块上的点阵屏由64颗红色灯珠以及64颗蓝色灯珠组成，一共128颗灯珠。如图中演示，每个格子（像素）中安装有一红一蓝两颗贴片LED（虽然他们被封装在同一个贴片元件内）。

```
uint8_t matrix[] = {
```

0, 255,
蓝色 LED 色值 红色 LED 色值

像素 01

0, 128,
蓝色 LED 色值 红色 LED 色值

像素 02

基于硬件，我们需要使用一个大小为64(像素数)*2(色彩数)=128的一维数组来表示每个格子中每颗LED的亮暗情况。

从上图中您可以知道，每个像素的显示需要蓝红两个通道的色值以进行控制。

为了方便理解，我们可以将数据分为64组，每组两个色值。这些数据在大小为128的一维数组中**由上至下由左至右排列**，分别代表64个像素的LED色值数据。

我们可以总结出一套规律：

2 * n 代表第n个像素中的蓝灯 「n属于[0, 7]」

$2 * n + 1$ 代表第n个像素中的红灯

```
tt_matrix.SetAllPWM((uint8_t*)matrix);
```

在生成了您所需的矩阵数据之后，调用SetAllPWM函数，并且将取模软件生成的矩阵数据传入，即可点亮点阵屏。（使用前请记得初始化点阵屏模块）

控制样例 - 数字显示

```
#include <RMTT_Libs.h>
#include <wire.h>

RMTT_Matrix tt_matrix;

uint8_t matrix_b3[] = {
    0, 0, 0, 0, 0, 0, 255, 0, 255, 0, 0, 0, 0, 0, 0,
    0,
    0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0,
    0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0,
    0,
    0, 0, 0, 0, 0, 0, 255, 0, 255, 0, 0, 0, 0, 0, 0,
    0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0,
    0,
    0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0,
    0,
    0, 0, 0, 0, 0, 0, 255, 0, 255, 0, 0, 0, 0, 0, 0,
    0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0,
};

uint8_t matrix_b2[] = {
    0, 0, 0, 0, 0, 0, 255, 0, 255, 0, 0, 0, 0, 0, 0,
    0,
    0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0,
    0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0,
    0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0,
    0,
    0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0, 0, 0,
    0,
    0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0,
    0, 0, 0, 0, 255, 0, 255, 0, 255, 0, 255, 0, 0, 0, 0,
    0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0,
};

uint8_t matrix_b1[] = {
    0, 0, 0, 0, 0, 0, 255, 0, 255, 0, 0, 0, 0, 0, 0,
    0,
};
```

```

    0, 0, 0, 0, 255, 0, 255, 0, 255, 0, 0, 0, 0, 0,
0,
    0, 0, 0, 0, 0, 0, 255, 0, 255, 0, 0, 0, 0, 0,
0,
    0, 0, 0, 0, 0, 0, 255, 0, 255, 0, 0, 0, 0, 0,
0,
    0, 0, 0, 0, 0, 0, 255, 0, 255, 0, 0, 0, 0, 0,
0,
    0, 0, 255, 0, 255, 0, 255, 0, 255, 0, 255, 0, 255, 0,
0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
};

void setup() {
  wire.begin(27, 26);
  wire.setClock(400000);

  tt_matrix.Init(127); // 初始化模块并设置全局电流控制值
  tt_matrix.SetLEDStatus(RMTT_MATRIX_CS, RMTT_MATRIX_SW,
RMTT_MATRIX_LED_ON); // 打开所有LED
}

void loop() {
  tt_matrix.SetAllPWM((uint8_t*)matrix_b3); // 设置数字为蓝色的3
  delay(1000);
  tt_matrix.SetAllPWM((uint8_t*)matrix_b2); // 设置数字为蓝色的2
  delay(1000);
  tt_matrix.SetAllPWM((uint8_t*)matrix_b1); // 设置数字为蓝色的1
  delay(1000);
}

```

如果一切顺利，您应当能看到点阵屏上轮流显示蓝色数字3、2、1。

ToF (Time of Flight) 测距传感器

模块组成



本模块由红外发射器和接收器组成。

通过比对红外光出射和返回的时间差，也即光的飞行时间(Time of Flight, ToF)，我们使用公式

$$d = \frac{ct}{2} \quad (\text{其中 } c \text{ 为光速, } t \text{ 为时间差})$$

即可计算得知光的单程行走路程d, 也即我们所需要测量的距离。

模块接口

```
class RMTT_TOF{
public:

    bool Init(bool io_2v8 = true);

    uint16_t ReadRangeSingleMillimeters(void);

    inline void SetTimeout(uint16_t timeout) { io_timeout = timeout; }
    bool TimeoutOccurred(void);

};
```

常用函数名	传参 (返回值)	范围	释义	函数作用
Init	/	/	/	必须先初始化 ToF传感器
SetTimeout	timeout	无	设置单次 测量超时 时间	必须先设置单 次测量超时 时间
ReadRangeSingleMillimeters	result (uint16_t)	0- 8000	单次测量 结果	获取单次测量 结果(单位 mm)
TimeoutOccurred	result (bool)	true / false	是否测量 超时	判断单次测量 是否超时

控制样例 - 在点阵屏上显示距离

```
#include <RMTT_Libs.h>
#include <Wire.h>

#define RANGE_MAX (0.64) // 满探测距离设置为0.64m, 使每格表示1cm

RMTT_RGB tt_rgb;
RMTT_Matrix tt_matrix;
RMTT_TOF tt_sensor;

uint8_t tof_display[128] = {0}; // 点阵内容数组
float range = 0;

void setup() {
    Serial.begin(115200);
    Wire.begin(27, 26);
    Wire.setClock(400000);
```

```

tt_sensor.SetTimeout(500); // 设置单次测量超时时间

if (!tt_sensor.Init()) {
    Serial.println("Failed to detect and initialize sensor!");
    while (1) {
    }
}

tt_matrix.Init(127); // 初始化点阵
tt_matrix.SetLEDStatus(RMTT_MATRIX_CS, RMTT_MATRIX_SW, RMTT_MATRIX_LED_ON);
// 开启点阵LED
}

void loop() {

    range = tt_sensor.ReadRangeSingleMillimeters();
    range = range / 1000; // 将数据单位转换至米

    Serial.println(range);

    if (tt_sensor.TimeoutOccurred()) { // 判断当次测量是否未超时（是否为有效测量）
        Serial.print("TIMEOUT");
    } else {

        for (int i = 0; i < 64; i++) {
            if (range / RANGE_MAX > (float)i / (float)63) {
                tof_display[i * 2 + 1] =
                    (int)((float)i / (float)63 * (float)255); // 仅控制红色亮度值
                // 以完成渐变色效果
                tof_display[i * 2] = 255; // 范围内蓝色始终最大
            } else {
                /* 范围外所有LED熄灭 */
                tof_display[i * 2 + 1] = 0;
                tof_display[i * 2] = 0;
            }
        }

        tt_matrix.SetAllPWM(tof_display); // 发送至点阵屏以显示
        delay(15);
    }
}
}

```



本样例基于上一小节所学的红蓝LED点阵控制知识完成了一个ToF测距仪，测量结果以格子数量显示在点阵屏上，每格代表1cm。

与无人机通讯 - TT_Protocol模块

模块接口

```
class RMTT_Protocol{
public:

    void SDKOn();
    void SDKOff();

    void TakeOff();
    void Land();
    void Emergency();
    void Up(int16_t x);
    void Down(int16_t x);
    void Left(int16_t x);
    void Right(int16_t x);
    void Forward(int16_t x);
    void Back(int16_t x);
    void CW(uint16_t x);
    void CCW(uint16_t x);
    void Flip(char x);
    void Go(int16_t x, int16_t y, int16_t z, uint16_t speed);
    void Go(int16_t x, int16_t y, int16_t z, uint16_t speed, char *mid);
    void Stop();
    void Curve(int16_t x1, int16_t y1, int16_t z1, int16_t x2, int16_t y2,
int16_t z2, uint16_t speed);
    void Curve(int16_t x1, int16_t y1, int16_t z1, int16_t x2, int16_t y2,
int16_t z2, uint16_t speed, char *mid);
    void Jump(int16_t x, int16_t y, int16_t z, uint16_t speed, int16_t yaw, char
*mid, char *mid2);

    void SetSpeed(int16_t x);
    void SetRC(int16_t a, int16_t b, int16_t c, int16_t d);
    void SetMon();
    void SetMoff();
    void SetMdirection(uint8_t x);

    void ReadSpeed();
    void ReadBattery();
    void ReadTime();
    void ReadSN();
    void ReadSDKVersion();

};
```

常用函数名	传参（返回值）	范围	释义	函数作用
SDKOn	/	/	/	初始化飞机明文通讯SDK
TakeOff	/	/	/	起飞
Land	/	/	/	降落

TT无人机通过机身的Micro USB接口与TT扩展模块供电与通讯，其中通讯部分通过串口通讯完成。

具体的串口发送步骤比较繁琐，我们已经将其封装成名为RMTT_Protocol的类，以便您使用。

在使用RMTT_Protocol模块之前，请谨记将您创建的对象初始化（调用SDKOn函数）。

*具体的命令发送以及参数范围请参照Tello Talent的明文SDK文档。

需要注意的是，如果您需要在您的程序中控制无人机，您需要确保无人机已经初始化完毕，在此我们列出两个解决方案：

1. 定时向无人机发送command指令，当其返回"ETT OK"时则飞机已经初始化完成可以起飞。
2. 循环向无人机发送command指令后同时发送battery?指令，当有电量数据返回时则飞机已经初始化完成可以起飞。

您对串口数据的读取可以通过调用Serial1（注意，不是Serial）来完成，其使用方法与普通Serial无异。

在此我们列出一个对无人机初始化成功与否的简单判断实例。等待无人机初始化完毕，TT扩展模块的顶灯将由红色变为蓝色，按压TT扩展模块上的按钮，TT无人机将在5秒后起飞，并随后自动降落。

注意：TT无人机的初始化最长可能需要几分钟的时间，请耐心等待。

```
#include <RMTT_Libs.h>
#include <wire.h>

RMTT_Protocol tt_sdk;

void waitTelloReady()
{
  while (1)
  {
    if (Serial1.available())
    {
      String ret = Serial1.readString();
      if (!strncmp(ret.c_str(), "ETT ok", 6))
      {
        Serial.println(ret.c_str());
        return;
      }
    }
    delay(500);
    tt_sdk.SDKOn();
  }
}

void setup() {
  wire.begin(27, 26);
  Serial1.begin(1000000, SERIAL_8N1, 23, 18);

  RMTT_RGB::Init();
  RMTT_RGB::SetRGB(255, 0, 0);

  tt_sdk.SDKOn();
  waitTelloReady();
  /* 阻塞线程并一直读取无人机串口返回数据，此后Tello可以确保已经初始化完毕 */
```

```
RMTT_RGB::SetRGB(0, 255, 255);

}

void loop() {
  pinMode(34, INPUT_PULLUP);

  // 等待按键被按下
  while (digitalRead(34) == 1);

  // LED Blink
  RMTT_RGB::SetRGB(0, 255, 0);
  delay(1000);
  RMTT_RGB::SetRGB(0, 0, 0);
  delay(1000);
  RMTT_RGB::SetRGB(0, 255, 0);
  delay(1000);
  RMTT_RGB::SetRGB(0, 0, 0);
  delay(1000);
  RMTT_RGB::SetRGB(0, 255, 0);

  tt_sdk.TakeOff();
  delay(6000);
  tt_sdk.Land();
}
```

进阶开发

有着丰富硬件功能的TT扩展模块依然可能无法完全释放您的创造力。

为此，我们实现了两个额外**的基础工程模板**，您可以在**我们提供的模板的基础上构建更加丰富多彩的应用程序**。

您只需在我们提供的样例上构建您的程序，便可以在**保留使用GameSir盖世小鸡游戏手柄或者手机端的Tello EDU APP操控无人机的功能的前提下**继续您的创意构造。

有手柄？

玩点花样。



只有手机？

也没问题。

工程模板 - GameSir盖世小鸡手柄操控



此处**仅列出部分**核心代码
完整代码**请使用库中 Example**



```

void user_task(void *arg) {
    while (1) {
        /* Put your code here ! */

        delay(10);
    }
}

void loop() {
    /* Serial receive process from drone */
    while (CommonSerial.available()) {
        int ret = cmd_process(
            CommonSerial.read()); // Send received data to the command parser
        if (ret != 0) {
            CommonSerial.printf("command error: %d\r\n", ret);
        }
    }
    /* -----!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!----- */
    /* DO NOT ADD ANY CODE HERE FOR NOT BLOCKING THE RECEIVE FROM THE SERIAL */
    /* YOU CAN ADD YOUR USER CODE TO THE 'user_task' FUNCTION ABOVE */
    /* -----!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!----- */
}

/*****Drone control part*****/
int rmtt_int = 0;
bool rmtt_bool = false;

bool int_is_valid = false;
bool bool_is_valid = false;

bool mac_is_valid = false;
uint8_t rmtt_mac[6] = {0};

/**
 * Drone connection handling, measure the connection state by serial data
 * feedback from the drone
 * - Update connection state by parsing data feeded back from inner serial1
 * - Fetch the MAC of the drone
 *
 * @param arg Parameter about task control
 */
int rmtt_callback(int argc, char* argv[], char argv2[]) {
    if (!strcmp(argv[1], "ok")) {
        bool_is_valid = true;
        rmtt_bool = true;
    } else if (!strcmp(argv[1], "error")) {
        bool_is_valid = true;
        rmtt_bool = false;
    } else if (!strcmp(argv[1], "mac")) {
        if ((sscanf(argv[2], "%02x%02x%02x%02x%02x%02x", &rmtt_mac[0],
            &rmtt_mac[1], &rmtt_mac[2], &rmtt_mac[3], &rmtt_mac[4],
            &rmtt_mac[5]) == 6) &&
            (argc == 3)) {
            mac_is_valid = true;
            Serial.println("rmtt_callback(): mac get ok");
        } else {
            Serial.println("rmtt_callback(): mac get error");
        }
    }
}

```

```

    }
} else if (sscanf(argv[1], "%d", &rmtt_int) && (argc == 2)) {
    int_is_valid = true;
}
return 0;
}

/**
 * Gamesir joystick pairing process handling
 * - Detect the press event of the pairing button
 * - Pair the joystick with the TT Plugin Module
 *
 * @param arg Parameter about task control
 */

void gamesir_pairing_task(void* arg) {
    int __key_cnt = 0;
    pinMode(34, INPUT_PULLUP);
    for (;;) {
        if (digitalRead(34) == 0)
            __key_cnt++;
        else
            __key_cnt = 0;

        if (__key_cnt >= 20 && !p_tt_gamesir->GetConnectedStatus()) {
            pair_mode = true;
            p_tt_gamesir->SetMACFilterEnable(false);
        }

        delay(100);
    }
}

#define TAKEOFF_TIMEOUT 200

int takeoff_status = 0;

int now_time = 0;
int last_clean_time = 0;

/**
 * Gamesir joystick control handling
 * - Receive command from the joystick
 * - Control the drone by received command
 *
 * @param arg Parameter about task control
 */

void gamesir_task(void *arg) {
    uint8_t command_init = 0;
    uint8_t mac_init = 0;

    for (;;) {
        if (mac_init == 0) {
            /* Try to fetch the MAC of any already paired joysticks from the
             * drone */
            CommonSerial.print("[TELLO] getmac?");
            delay(100);

```

```

Serial.println("gamesir_task(): mac is ok?");
if (rmtt_joystick_mac_is_valid()) {
    Serial.println("gamesir_task(): ble mac init");
    p_tt_gamesir->Init(get_rmtt_joystick_mac());
    mac_init = 1;
}
} else if ((command_init == 0) &&
           (p_tt_gamesir->GetConnectedStatus())) {
    tt_sdk.SDKOn();
    delay(100);
    if (rmtt_bool_is_valid()) {
        command_init = 1;
    }
} else if (p_tt_gamesir->DataIsValid()) {
    PlainData data = p_tt_gamesir->GetData();

    int lx = ((float)data.left_x_3d - 512) / 512.0 * 100;
    int ly = ((float)data.left_y_3d - 512) / 512.0 * 100;
    int rx = ((float)data.right_x_3d - 512) / 512.0 * 100;
    int ry = ((float)data.right_y_3d - 512) / 512.0 * 100;

    if ((data.bnt3 == 0x01) && (data.L2)) {
        tt_sdk.Flip('f');
    } else if ((data.bnt3 == 0x03) && (data.L2)) {
        tt_sdk.Flip('r');
    } else if ((data.bnt3 == 0x05) && (data.L2)) {
        tt_sdk.Flip('b');
    } else if ((data.bnt3 == 0x07) && (data.L2)) {
        tt_sdk.Flip('l');
    } else if ((data.Y) && (data.R2)) {
        if (takeoff_status == 0) {
            tt_sdk.TakeOff();
            takeoff_status = 1;
        } else {
            tt_sdk.Land();
            takeoff_status = 0;
        }
    } else {
#ifdef BLE_JAPAN_CTRL
        tt_sdk.SetRC(lx, -ly, -ry, rx);
#else
        tt_sdk.SetRC(rx, -ry, -ly, lx);
#endif
    }
}

/* Regularly send data packet to ensure the drone
floating steadily after the controller was offline*/

if ((now_time - last_clean_time > 300) &&
    (p_tt_gamesir->GetDataOffline()) && command_init) {
    tt_sdk.SetRC(0, 0, 0, 0);
    /* 刷新rc清除时间 */
    last_clean_time = millis();
} else {
}

now_time = millis();

```

```

        delay(10);
    }
}

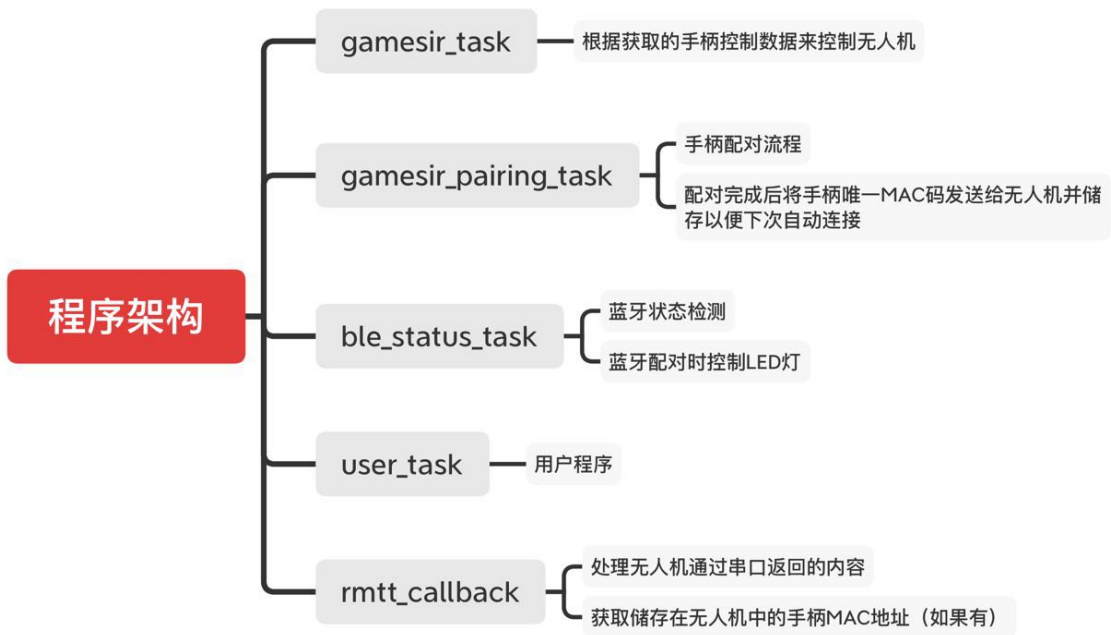
/**
 * Gamesir joystick Bluetooth(LE) connection handling
 *
 * @param arg Parameter about task control
 */
void ble_status_task(void *arg) {
    static int __led_cnt = 0;
    static uint8_t toggle = 0;
    uint8_t ble_mac[6] = {0};

    while (1) {
        if (get_led_effect_mode() == LED_EFFECT_FACTORY_MODE) {
            if (p_tt_gamesir->GetConnectedStatus()) {
                if (pair_mode == true) {
                    memcpy(ble_mac, p_tt_gamesir->GetMAC(), 6);
                    p_tt_gamesir->SetMACFilterEnable(true);
                    /* Write the MAC of the joystick to the drone */
                    CommonSerial.printf(
                        "[TELLO] setmac %02x%02x%02x%02x%02x%02x", ble_mac[0],
                        ble_mac[1], ble_mac[2], ble_mac[3], ble_mac[4],
                        ble_mac[5]);
                    delay(50);
                    if (rmtt_bool_is_valid()) {
                        Serial.println(
                            "ble_status_task(): pairing is successful");
                        if (get_rmtt_bool()) {
                            pair_mode = false;
                        }
                    }
                }
                RMTT_RGB::SetBlue(255);
            } else if (pair_mode == true) {
                if (__led_cnt % 4 == 0) {
                    toggle = ~toggle;
                }

                if (!toggle) {
                    RMTT_RGB::SetBlue(0);
                } else {
                    RMTT_RGB::SetBlue(255);
                }
            } else {
                RMTT_RGB::SetBlue(0);
            }
        }

        __led_cnt++;
        delay(100);
    }
}

```



这里是这个工程样例的完整代码。这份代码实现了与飞机的串口通讯以及与GameSir手柄的蓝牙配对、连接、通讯的功能。

通过程序架构流程图，我们可以了解这个工程样例的实现逻辑。

如果您只是想要使用它进行进一步开发，您只需要在user_task函数内的循环体中编辑您的代码即可，这样不会对样例本身已经实现的功能造成影响。

```
void user_task(void *arg)
{
    while (1)
    {
        /* Put your code here ! */
        delay(10);
    }
}
```

左右摇杆 ⤴

以美国手规则控制无人机姿态、位置

左扳机 + 十字键 ⬆

前后左右四向翻滚

右扳机 + Y 键 Ⓚ

起飞 / 降落



在将手柄与TT扩展模块配对时，只需对TT扩展模块中的按钮长按两秒，TT扩展模块便会进入配对模式。等待TT扩展模块自动与手柄配对完成之后，通过图中的操作指南，您便可以使用GameSir手柄操纵TT无人机了。

工程模板 - 与Tello EDU APP互动



此处**仅列出部分**核心代码
完整代码**请使用**库中 **Example**



```
int tof_callback(int argc, char *argv[], char argv2[])
{
    CommonSerial.printf("tof %d", tof_range);
}

void user_task(void *arg)
{
    while (1)
    {
        /* Put your code here ! */
        delay(10);
    }
}

void loop()
{
    /* serial receive process from drone */
    while (CommonSerial.available())
    {
        int ret = cmd_process(CommonSerial.read());
        if (ret != 0)
        {
            CommonSerial.printf("command error: %d\r\n", ret);
        }
    }
    /* -----!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!----- */
    /* DO NOT ADD ANY CODE HERE FOR NOT BLOCKING THE RECEIVE FROM THE SERIAL */
    /* YOU CAN ADD YOUR USER CODE TO THE 'user_task' FUNCTION ABOVE */
    /* -----!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!----- */
}

/*****Matrix control part*****/

/**
 * Handling the commands about the Matrix
 *
 * @param argc Control argument
 * @param argv[] value argument 1
 * @param argv2[] value argument2
 */
int matrix_callback(int argc, char *argv[], char argv2[])
```



```

{
    if ((!strcmp(argv[1], "g")) || (!strcmp(argv[1], "sg")))
    {
        uint8_t buff[128] = {0};
        if (argc == 3)
        {
            if (rbpstr2buff(buff, argv[2], MLED_BRIGHT) != 0)
            {
                goto end;
            }
        }
        else if ((argc == 2) && strlen(argv[2]))
        {
            if (rbpstr2buff(buff, argv[2], MLED_BRIGHT) != 0)
            {
                goto end;
            }
        }
        else
        {
            goto end;
        }

        matrix_effect_set_graph(buff);

        if (!strcmp(argv[1], "sg"))
        {
            File file = SPIFFS.open("/matrix_graph.txt", FILE_WRITE);
            File file2 = SPIFFS.open("/graph_enable.txt", FILE_WRITE);
            if (file)
            {
                file.write(buff, sizeof(buff));
            }
            file.close();
            file2.close();
        }
    }
    else if ((!strcmp(argv[1], "sc")) && (argc == 2))
    {
        if (SPIFFS.exists("/graph_enable.txt"))
        {
            SPIFFS.remove("/graph_enable.txt");
        }
        uint8_t buff[128] = {0};
        matrix_effect_set_graph(buff);
    }
    else if ((!strcmp(argv[1], "s")) && (argc == 4) &&
             (strlen(argv[2]) == 1))
    {
        uint8_t buff[128] = {0};
        if (strlen(argv[3]) == 1)
        {
            if (mled_font2buff(buff, argv[3][0], argv[2][0], MLED_BRIGHT) !=
                0)
            {
                goto end;
            }
        }
    }
}

```



```

    }
    else
    {
        goto end;
    }
}
else
{
    goto end;
}
CommonSerial.print("matrix ok");
return 0;
end:
CommonSerial.print("matrix error");
return 0;
}

/*****Drone control part*****/
int rmtt_int = 0;
bool rmtt_bool = false;

bool int_is_valid = false;
bool bool_is_valid = false;

/**
 * Drone connection handling, measure the connection state by serial data
 * feedback from the drone
 * - Update connection state by parsing data feeded back from inner serial1
 *
 * @param argc Control argument
 * @param argv[] Value argument 1
 * @param argv2[] Value argument2
 */
int rmtt_callback(int argc, char *argv[], char argv2[])
{
    if (!strcmp(argv[1], "ok"))
    {
        bool_is_valid = true;
        rmtt_bool = true;
    }
    else if (!strcmp(argv[1], "error"))
    {
        bool_is_valid = true;
        rmtt_bool = false;
    }
    else if (sscanf(argv[1], "%d", &rmtt_int) && (argc == 2))
    {
        int_is_valid = true;
    }
    return 0;
}

/**
 * Led command handling, control the color of the LED by input commands
 *
 * @param argc Control argument
 * @param argv[] Value argument 1
 * @param argv2[] Value argument2

```

```

*/
int led_callback(int argc, char *argv[], char argv2[])
{
    uint32_t r1, b1, g1, r2, b2, g2;
    if (!strcmp(argv[1], "b1"))
    {
        uint8_t blink_time = 1;
        if ((argc == 9) && sscanf(argv[2], "%d", &blink_time) &&
            sscanf(argv[3], "%d", &r1) && sscanf(argv[4], "%d", &g1) &&
            sscanf(argv[5], "%d", &b1) && sscanf(argv[6], "%d", &r2) &&
            sscanf(argv[7], "%d", &g2) && sscanf(argv[8], "%d", &b2))
        {
            if ((blink_time >= 1) && (blink_time <= 10))
            {
                blink_time = (11 - blink_time) * BLINK_UNIT;
                led_effect_blink(r1, g1, b1, r2, g2, b2, blink_time);
                CommonSerial.print("led ok");
            }
            else
            {
                goto end;
            }
        }
        else
        {
            goto end;
        }
    }
    else if (!strcmp(argv[1], "br"))
    {
        int breath_time = 6;
        if ((argc == 6) && sscanf(argv[2], "%d", &breath_time) &&
            sscanf(argv[3], "%d", &r1) && sscanf(argv[4], "%d", &g1) &&
            sscanf(argv[5], "%d", &b1))
        {
            if ((breath_time >= 1) && (breath_time <= 10))
            {
                breath_time = (11 - breath_time) * BREATH_UNIT;
                led_effect_breath(r1, g1, b1, breath_time);
                CommonSerial.print("led ok");
            }
            else
            {
                goto end;
            }
        }
        else
        {
            goto end;
        }
    }
    else if (argc == 4)
    {
        if (sscanf(argv[1], "%d", &r1) && sscanf(argv[2], "%d", &g1) &&
            sscanf(argv[3], "%d", &b1))
        {
            led_effect_set_rgb(r1, g1, b1);
            CommonSerial.print("led ok");
        }
    }
}

```

```

    }
    else
    {
        goto end;
    }
}
else
{
    goto end;
}

return 0;
end:
CommonSerial.print("led error");
return 0;
}

/**
 * ToF measuring command handling
 *
 * @param arg Parameter about task control
 */
void tof_battery_read_task(void *arg)
{
    int range_cm = 0;
    int battery_cnt = 0;
    for (;;)
    {
        tof_range = tt_tof.ReadRangeSingleMillimeters();

        if (!(battery_cnt % 10))
        {
            tt_sdk.ReadBattery();
        }

        if (tof_range >= 40 || tof_range <= 800)
        {
            range_cm = tof_range / 200.0;
            for (int i = 0; i < 8; i += 2)
            {
                if (2 * range_cm > i)
                {
                    tt_graph_buff[16 * 7 + i] = 0;
                    tt_graph_buff[16 * 7 + i + 1] = 255;
                }
                else
                {
                    tt_graph_buff[16 * 7 + i] = 0;
                    tt_graph_buff[16 * 7 + i + 1] = 0;
                }
            }
        }
        else if (tof_range > 800)
        {
            for (int i = 0; i < 8; i++)
            {
                tt_graph_buff[16 * 7 + i] = 0;
            }
        }
    }
}

```

```

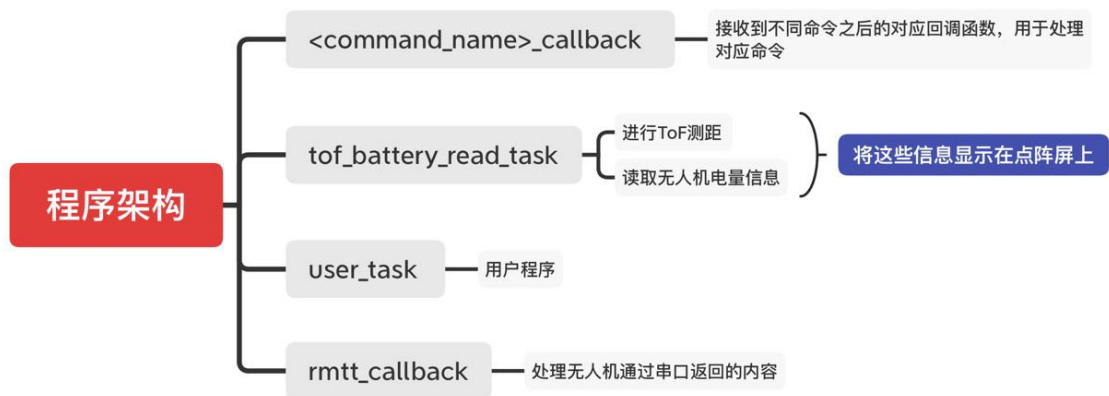
}

if (rmmt_int_is_valid())
{
    int val = get_rmmt_int() / 25.0;
    for (int i = 8; i < 16; i += 2)
    {
        if (2 * val > i - 8)
        {
            tt_graph_buff[16 * 7 + i] = 255;
            tt_graph_buff[16 * 7 + i + 1] = 255;
        }
        else
        {
            tt_graph_buff[16 * 7 + i] = 0;
            tt_graph_buff[16 * 7 + i + 1] = 0;
        }
    }
}

if (get_matrix_effect_mode() == MATRIX_EFFECT_FACTORY_MODE)
{
    RMTT_Matrix::SetAllPWM((uint8_t *)tt_graph_buff);
}
battery_cnt++;
delay(100);
}
}

```

样例释疑

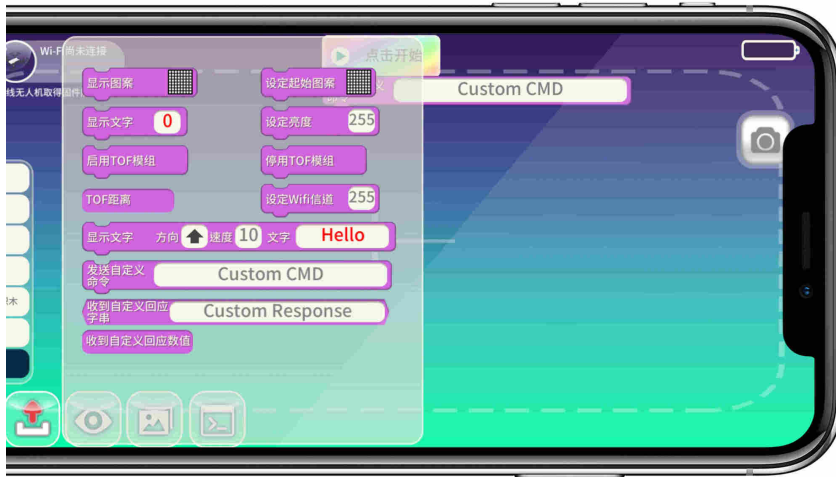


这里是这个工程样例的完整代码。这份代码实现了与无人机的串口通讯以及对ToF传感器、点阵屏及机顶LED灯的调用样例的功能。

通过程序架构流程图，我们可以了解这个工程样例的实现逻辑。

如果您只是想要使用它进行进一步开发，您只需要在user_task函数内的循环体中编辑您的代码即可，这样不会对样例本身已经实现的功能造成影响。

```
void user_task(void *arg)
{
  while (1)
  {
    /* Put your code here ! */
    delay(10);
  }
}
```



使用 Tello EDU 控制

接下来您便可以在保留使用Tello EDU APP进行控制的能力的基础上编辑您自己的精彩程序了。