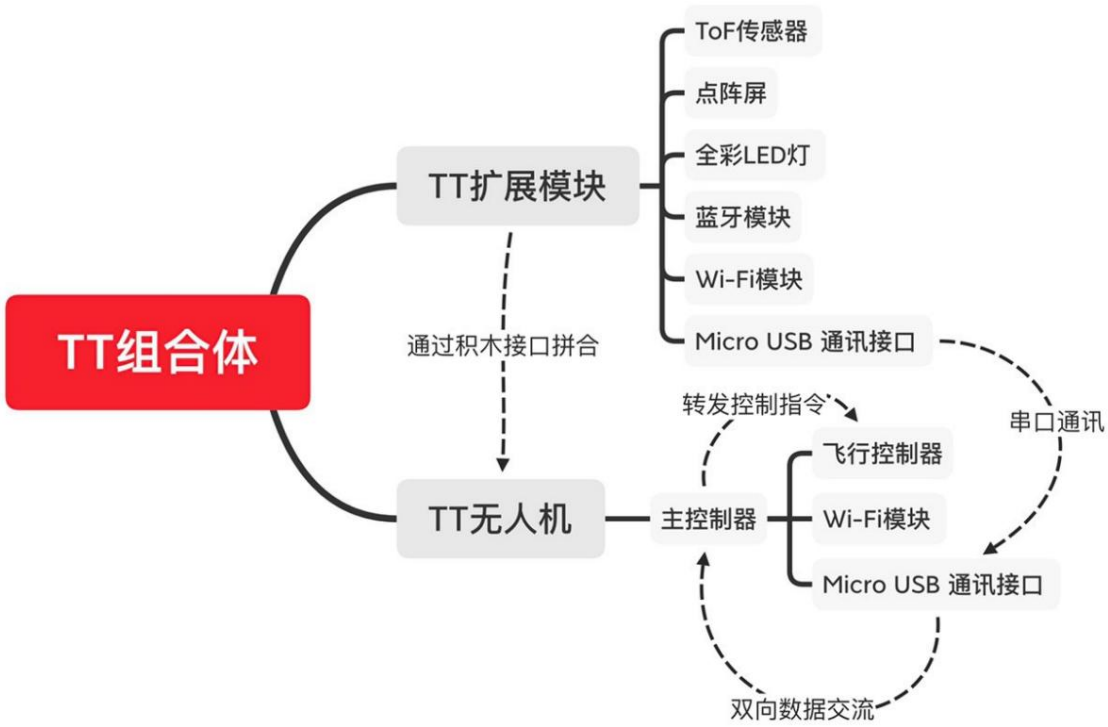
ROBOMASTER

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。

Necessary knowledge before starting development - combined control mode of TT expansion module and TT drone



If you need to achieve control of TT drones, you always need to look at TT drones separately from TT expansion modules.

We first look at the TT expansion module from the perspective of a TT drone.

The TT UAV has its own flight controller, which controls the four hollow cup motors of the aircraft by obtaining sensor information such as the gyroscope of the aircraft itself to control the physical movement of the aircraft. Control instructions from different sources (such as mobile phones, handles) are further sent to the flight controller through the main controller of the TT UAV to complete specific flight instructions, and the return of aircraft information (such as power, current altitude, etc.) It is also done by the main controller of the TT drone.

The control of the UAV by the TT expansion module is the same as that of a mobile phone and a handle, and it is realized by transmitting control instructions to the main controller of the TT UAV . **And this specific communication process is realized through serial communication through the onboard Micro USB interface of the drone, and this interface will also supply power to the TT expansion module.**

When we change the perspective, from the perspective of the TT expansion module, it is equivalent to the external brain attached to the TT drone. It not only expands the perception capability of TT drone itself (forward ToF sensor), but also enriches the functionality of TT drone. At the same time, you can also write your own program to the expansion module to make the TT combination complete complex flight tasks that are difficult to complete only by the TT drone itself. By communicating with the TT drone to obtain the sensor information of the drone itself, combined with the sensor information of the expansion module itself, you can realize a wealth of flight control functions, such as realizing forward obstacle avoidance for the TT combination. More exciting ways to play are waiting for your exploration!

Building a Development Environment (Windows)

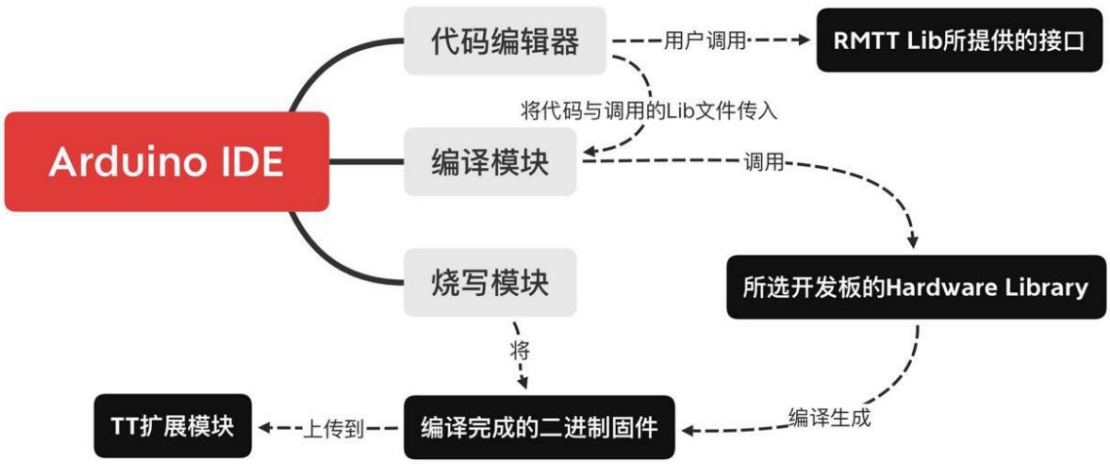
In this guide, we have chosen the Arduino IDE as the user-facing programming environment. As a derivative of the Arduino open source project , **the Arduino IDE has various excellent features such as portability, ease of use, high integration, and ease of expansion. Even novices who know little about programming can quickly get** started and use it to construct their own wonderful projects.

Before using the Arduino IDE to start **the development of the Tello Talent expansion module, we need to make some necessary configurations on the development environment** to ensure that the Arduino IDE can load the necessary hardware toolkit and the corresponding underlying library for compiling the Tello Talent expansion module document.

Installation of Arduino IDE

1. If you have never installed Arduino IDE, please use the Arduino integrated version we provide for direct development. 2. If you have already installed Arduino IDE, please make sure the version is not lower than 1.8.12, and follow the steps below to add RMTT expansion module support holding bag.

Download link: https://www.dji.com/cn/robomaster-tt/downloads



After the Arudino IDE is installed, we need to configure the development environment. The above figure is the internal principle process of developing TT expansion module through Arduino IDE. We can see that there are two parts independent of Arduino IDE, namely Hardware **Library (hardware library) and RMTT Lib (functional interface library of Tello Talent expansion module).** These two independent parts need **to be installed independently after we** complete the installation of Arduino IDE, so that Arduino IDE can carry out integrated development of the Tello Talent expansion module.

Installation of the Arduino Hardware library

Please find and enter the following directory in your system disk (C drive):

> C:\Users\your username\Documents\Arduino

Please note that in some versions of Windows, the first five characters of your actual user name will be truncated at "your user name", such as

> If your actual user name is Username, then the directory is C:\Users\Usern\Documents\Arduino

You can avoid file path problems caused by naming by entering the "User" folder from the C drive and finding the folder corresponding to your account name

question.

In the package we provide, you can see the "hardware" folder. Copy it to the Arduino directory you entered, and merge (Merge) it into a folder with the same name under this directory. At this point, the installation of the Hardware library is complete, and then please complete the installation in the next section.

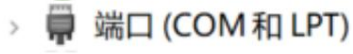
Installation of Arduino RMTT Library

Open the Arduino IDE at this point, and find the "RMTT_Libs.7z" file in the file package we provided in the project-load library-add .ZIP library on the top menu bar . Click OK and wait for a while, after which the RMTT Library should have been correctly imported into the Arduino IDE.

Build your project in the Arduino IDE

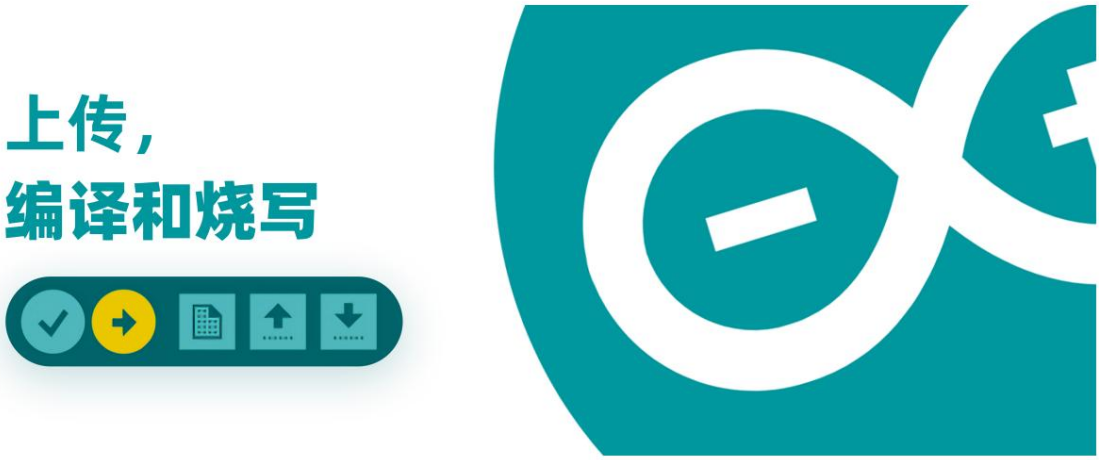
Before starting your first project, remember to select "RMTT Module" in the menu bar of Arduino IDE - Tools - Development Board, and then select the port corresponding to your TT expansion module in the "Port" below.

If you don't know which port your TT module corresponds to, please open Windows Device Manager, in



Find a device similar to the one pictured below:

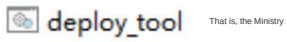
The brackets behind the device name is the port number you need, just select it in the Arduinio IDE.



In the toolbar under the menu bar of the Arduino IDE, the two buttons on the left are "compile only" and "compile+upload" **buttons** from left to right . **Usually we only need to click the second button, and the Arduino** IDE will automatically upload (program) the compiled binary file to the TT expansion module **after the compilation is completed .**

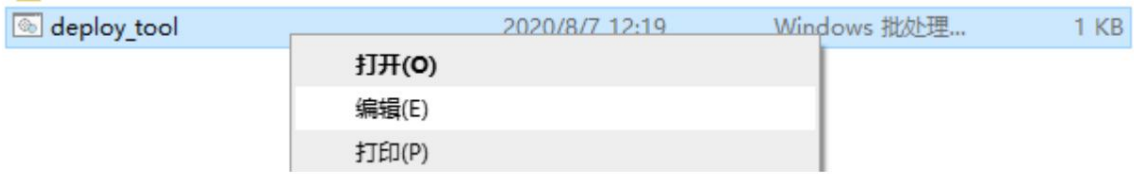
Usually, Arduino IDE takes a long time (three to five minutes, depending on computer performance) to compile the contents of the RMTT core library when compiling a new project, and then the compilation process will call the compilation cache for secondary compilation. If you think compilation is too slow, there is a solution mentioned in the next section.

External compilation and upload mode without Arduino IDE (external deployment)

Inside the TT package you can see the "External Deployment Tools" folder, go into that folder and you will see 

That is, the Ministry signing tool.

first use



You need to select "Edit" in the right-click menu, and change the environment path ("<arduino_loc>" **and** "<lib_loc>") in the first line of the deploy_tool.bat file to your Arduino program path and the installation of the Hardware library respectively **path (mentioned in the corresponding section above).**

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

After the changes are completed, the contents of the entire deploy_tool.bat file should be similar to the following:

```
@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
```

How to use after the environment path is configured

Press SHIFT + right key in the blank area of the "External Deployment Tools" folder, and select "Open Powershell window here" or "Open command prompt (CMD) window here".

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

Enter the above command in the Powershell (or CMD) window that appears.

Please remember to replace the three brackets ("<>") with the corresponding content before entering the command, which are "the address of the ino file that needs to be compiled", "the COM slogan that needs to be uploaded", "the name of your ino file "

For example:

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

In this way, the external deployment tool can automatically use the previously compiled and cached core library content to speed up the compilation process.

Get to know the Tello Talent extension module

This module is an expansion module of Tello Talent. It is fixed with TT through the building block interface, and the module is powered and data exchanged through the onboard Micro USB interface.

This module is equipped with full-color LED lights, Tof sensors and 8*8 red and blue LED dot matrix screens. Through our example, you will learn how to program and control this expansion module in Arduino, and use it to create your own projects.

Top full color LED

Module composition

This module consists of three high-brightness LEDs, red, green and blue, and an external diffuser. We can control the color of the external perception by controlling the brightness of LEDs of different colors and supplementing it with an external diffuser.

module interface

```
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); };
```

Common function name	pass parameters	parameter range	Parameter interpretation	function
Heat	/	/	/	**Necessary** - Initialize LEDs
SetRed	val	0-255	val LED brightness of this channel	Set the brightness of the red LED bead
SetBlue	val	0-255	val LED brightness of this channel	Set the brightness of the blue LED bead
SetGreen	val	0-255	val LED brightness of this channel	Set the brightness of the green lamp bead
SetRGB	R, G, B	Both are 0-255	R Red LED brightness G Green LED brightness B Blue LED brightness	Set the brightness of three channel lamp beads at the same time

Control Example - Breathing Light

The example of this module is to use RMTT_RGB to control the full-color LED light to make it appear breathing light effect.

```
#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; // instantiate RMTT_RGB object

bool breath_toggle = false; // Brightness change direction
uint8_t breath_rate = 0; int period =            // Percentage of brightness brightness percentage
10;

void setup()
        { Wire.begin(27, 26); // Initialize the I2C bus
        Wire.setClock(400000);

        period = 1000 / BREATH_Hz; // Calculate the duration of a single breathing
        cycle in ms
        tt_rgb.Init(); tt_rgb.SetRGB(0, 0, 0);
}

void loop() {

        /* Brightness increase or decrease, pay attention to the direction
        of breath_toggle to avoid overflow */ breath_rate = breath_toggle ? breath_rate - 1 : breath_rate + 1;


        if ((breath_rate == 100) || (breath_rate == 0))
                breath_toggle = !breath_toggle; // Change the brightness increase or decrease direction

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

        delay(period / 100); // delay one hundredth of a breathing cycle
}
```

Red and blue 8*8 LED matrix

Module composition

This module consists of 64 red lamp beads and 64 blue lamp beads, a total of 128 lamp beads. Through the packaged interface, you can control the color and brightness of each light bead in the LED matrix.

module interface

```
class RMTT_Matrix{ public:

                static void On(); static
                void Off();
```

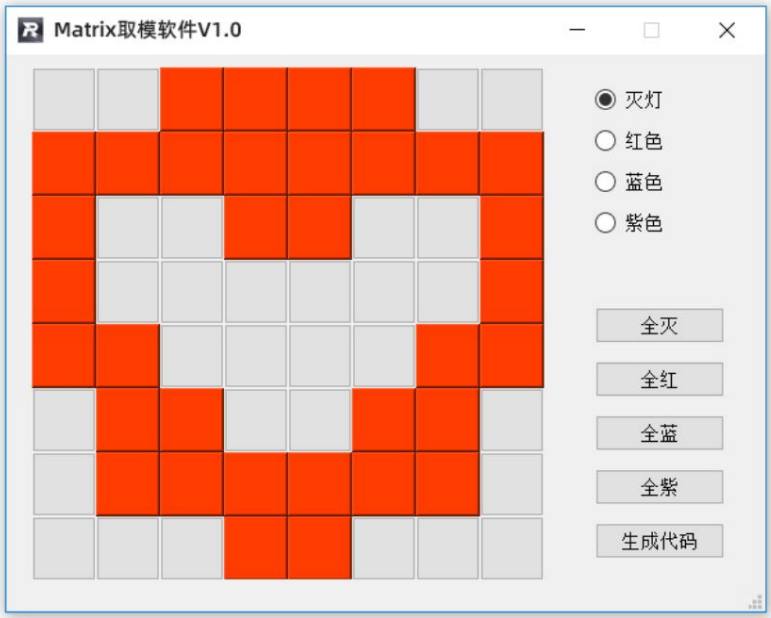
```
            static void Init(uint8_t gcc); static void
            SetGCC(uint8_t gcc); static void
            SetLEDStatus(uint8_t cs, uint8_t sw, IS31FL3733_LED_STATE
state);
            static void SetLEDPWM(uint8_t cs, uint8_t sw, uint8_t value); static void
            SetAllPWM(uint8_t *val);

            /*Below are some
            internal functions*/ 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();
};
```

Commonly used function name	parameter range		Parameter interpretation	function
On	/	/	/	open dot matrix
Off	/	/	/	turn off dot matrix
Heat	gcc	0-127	gcc global current control	The dot matrix must be initialized first
SetGCC	gcc	0-127	gcc global current control	Set global current (i.e. control brightness)
SetLEDStatus	cs, sw, state	0-8 / 0-16 / {0,1}	cs horizontal row number sw vertical column number state LED status	Switches that control all LEDs
SetLEDPWM	cs, sw, value	0-8 / 0-16 / 0-255	cs horizontal row number sw vertical column number value LED color value	Control the color value of all LEDs
SetAllPWM	*val	One-dimensional array of size 128	The detailed structure of the array is shown below	set bitmap

Dot matrix modulo and how to use it

In the process of controlling the dot matrix LED screen, we need to make the screen display the content we need. In embedded development, unless you have a packaged display library, you usually need to write a program yourself to control the brightness of each pixel and each LED from the bottom layer, in order to achieve the effect of displaying the content we want.

This means that even if we need to display simple content such as numbers and text, we need to control the brightness of each LED to achieve the corresponding effect. Fortunately, you don't need to implement this control process yourself, just use the modulo software to draw the content to be displayed in a simple graphical way, and then you can get the automatically generated dot matrix code from the modulo software.



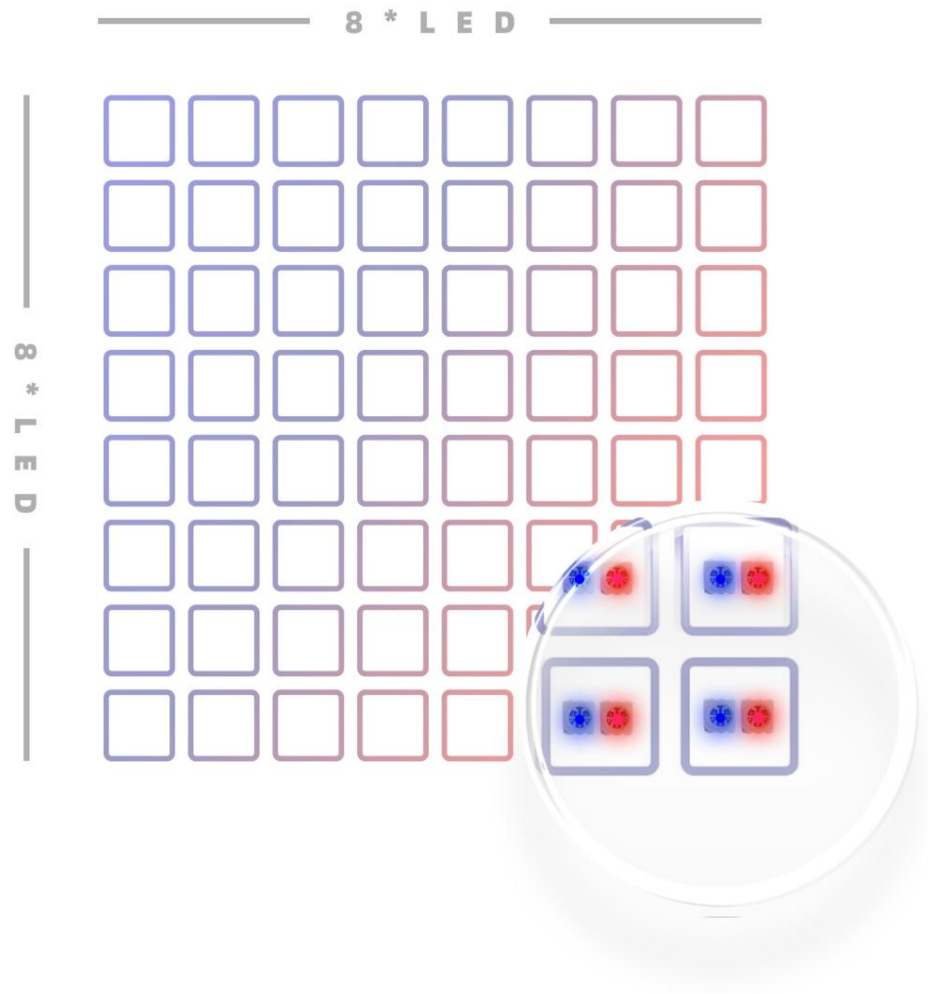
Open matrix.exe under the folder "Lattice Modulo", and then draw the graphics you need. click 生成代码 ,Right now

The corresponding lattice array can be obtained. Copy it into your project and it's ready to use.

```
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,    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,    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,
0,
        0,    0,    0,    0, 0, 0,       0, 255, 0, 255, 0,       0,    0, 0, 0,

0, };
```

The code generated by the modulo software is shown in the figure. Looking confused? In fact, its structure is not that complicated.

Let's take a look at the hardware composition of the dot matrix LED screen.



The dot matrix screen on the Talent expansion module consists of 64 red LEDs and 64 blue LEDs, a total of 128 LEDs. As shown in the figure, two SMD LEDs, one red and one blue, are installed in each grid (pixel) (although they are packaged in the same SMD component).

Based on the hardware, we need to use a one-dimensional array with a size of 64 (number of pixels) * 2 (number of colors) = 128 to represent the brightness and darkness of **each LED in each grid.**

From the picture above, you can know that the display of each pixel requires the color values of the blue and red channels for control.

For easy understanding, we can divide the data into 64 groups, each with two color values. These data are arranged from top to bottom and left to right in a one-dimensional array with a size of 128 , respectively representing the LED color value data of 64 pixels.

We can summarize a set of rules:

2 * n represents the blue light in the nth pixel **"n belongs to [0, 7]"**

> 2 * n + 1 represents the red light in the nth pixel

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

After generating the matrix data you need, call the SetAllPWM function and pass in the matrix data generated by the modulo software to light up the dot matrix screen. (Please remember to initialize the dot matrix screen module before use)

Control Example - Digital Display

```
#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, 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, 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, 0, 255,            0, 255,    0,    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, 0, 255,            0, 255,    0,    0,    0,    0, 0, 0,
0,
        0,    0,    0,    0, 0, 0, 255,            0, 255,    0,    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, 0,

0, };

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

    tt_matrix.Init(127); // Initialize the module and set the global current
    control value tt_matrix.SetLEDStatus(RMTT_MATRIX_CS, RMTT_MATRIX_SW,
RMTT_MATRIX_LED_ON);// Turn on all LEDs }


void loop()
    { tt_matrix.SetAllPWM((uint8_t*)matrix_b3); // Set the number to blue 3 delay(1000);

    tt_matrix.SetAllPWM((uint8_t*)matrix_b2); // Set the number to blue 2 delay(1000);

    tt_matrix.SetAllPWM((uint8_t*)matrix_b1); // Set the number to blue 1 delay(1000);

}
```

If all goes well, you should see the blue numbers 3, 2, 1 alternately displayed on the dot matrix screen.

ToF (Time of Flight) ranging sensor

Module composition

This module consists of infrared transmitter and receiver.

We use the **formula**

$$d = \frac{ct}{2}$$ (where is the speed of light and is the time difference) c t

The one-way travel distance d of light can be calculated, which is the distance we need to measure.

module interface

```
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);


};
```

Common function name	pass parameter (return value)	Range definition		function
Heat	/	/	/	must be initialized first ToF sensor
SetTimeout	timeout	none	Set a single measurement timeout	The single measurement timeout must be set first
ReadRangeSingleMillimeters	resultÿuint16_tÿ	0-8000	single measurement result	Get a single measurement result (in mm)
TimeoutOccurred	resultÿboolÿ	true / false	Whether to measure timeout	Determine whether a single measurement has timed out

Control example - display distance on dot matrix screen

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

#define RANGE_MAX (0.64) // The full detection range is set to 0.64m, so that each grid represents 1cm

RMTT_RGB tt_rgb;
RMTT_Matrix tt_matrix;
RMTT_TOF tt_sensor;

uint8_t tof_display[128] = {0}; // dot matrix content array float
range = 0;

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

```
        tt_sensor.SetTimeout(500); // Set a single measurement timeout

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



        }

        tt_matrix.Init(127); // Initialize dot matrix
        tt_matrix.SetLEDStatus(RMTT_MATRIX_CS, RMTT_MATRIX_SW, RMTT_MATRIX_LED_ON); // Turn
on dot matrix
LED }

void loop() {

        range = tt_sensor.ReadRangeSingleMillimeters(); range =
        range / 1000; // convert data unit to meter

        Serial.println(range);

        if (tt_sensor.TimeoutOccurred()) { // Judge whether the current measurement has not timed out (whether it is a valid measurement)
            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); // Only control the red brightness value
to complete the gradient effect
                    tof_display[i * 2] = 255; // within the range blue is always
                the
                largest } else { /* all LEDs
                    outside the range are off */
                    tof_display[i * 2 + 1] = 0; tof_display[i * 2] = 0;
                }
            }

            tt_matrix.SetAllPWM(tof_display); // Send to dot matrix screen to
            display delay(15);


        }
}
```

This example completes a ToF rangefinder based on the red and blue LED dot matrix control knowledge learned in the previous section. The measurement results are displayed on the **dot matrix** screen in the number of grids , and each grid represents 1cm.

Communicate with UAV - TT_Protocol module

module interface

```
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
        Forword(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();

};
```

Common function name	pass parameter (return value)	range definition function		
SDKOn	/	/	/	Initialize the aircraft plaintext communication SDK
TakeOff	/	/	/	take off
Land	/	/	/	landing

The TT drone supplies power and communicates with the TT expansion module through the Micro USB interface of the fuselage, and the communication part is completed through serial communication.

The specific serial port sending steps are cumbersome, we have encapsulated it into a class named RMTT_Protocol for your convenience.

Before using the RMTT_Protocol module, please remember to initialize the object you created (call the SDKOn function).

*For specific command sending and parameter ranges, please refer to Tello Talent's clear text SDK documentation.

It should be noted that if you need to control the drone in your program, you need to ensure that the drone has been initialized. Here we list two solutions:

> 1. Send the command command to the drone regularly, and when it returns "ETT OK", the aircraft has been initialized and can take off. 2. After sending the command command to the drone in a loop, send the battery? command at the same time. When the battery data returns, the aircraft has been initialized and can take off.

You can read serial port data by calling Serial1 (note, not Serial), and its usage is the same as ordinary Serial.

Here we list a simple example of judging whether the UAV initialization is successful or not. Wait for the initialization of the drone to be completed, the top light of the TT expansion module will change from red to blue, press the button on the TT expansion module, the TT drone will take off after 5 seconds, and then land automatically.

Note: The initialization of the TT drone may take a few minutes at most, please wait patiently.

```
#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(); /* Block the
    thread and keep reading the data returned by the serial port of the drone, after which Tello can ensure that it has been initialized */
```

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

}


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

    // Wait for the button
    to be pressed 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();

}
```

advanced development

TT expansion modules with rich hardware functions may still not be able to fully unleash your creativity.

To this end, we have implemented two additional basic project templates, and you can build more colorful applications based on the templates we provide .

You only need to build your program on the samples we provide, and you can continue your creative construction while retaining the function of controlling the drone with the GameSir game controller or the Tello EDU APP on the mobile phone.

有手柄？

玩点花样。

只有手机？

也没问题。

Engineering Template- GameSir GameSir Chick Handle Control

此处**仅列出部分**核心代码
完整代码**请使用库中 Example**

Machine Translated by Google

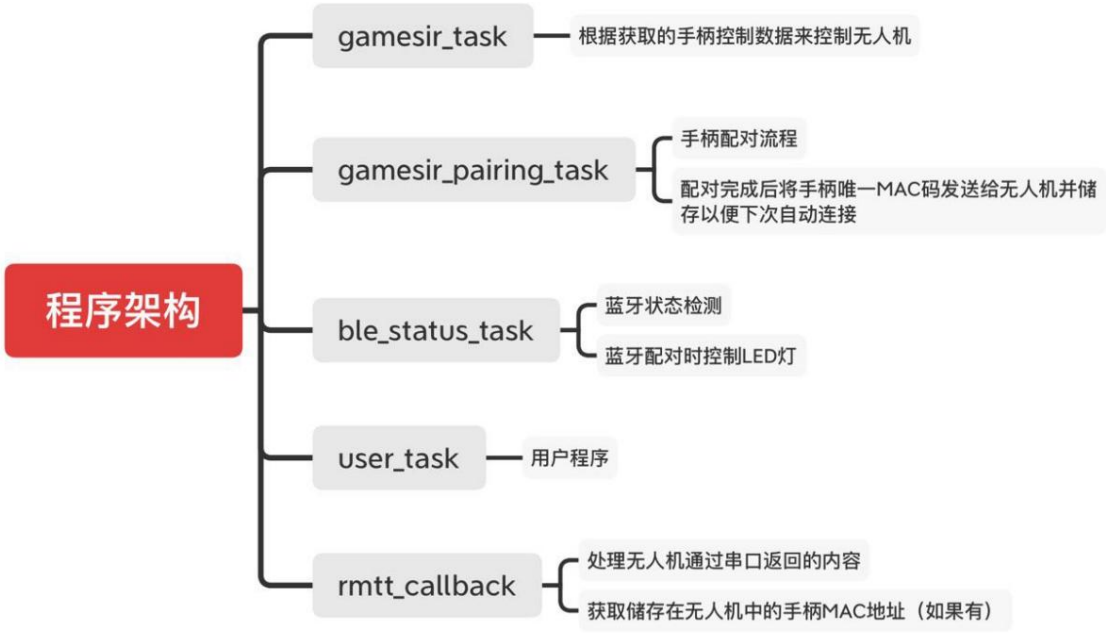
```
void user_task(void *arg) { while (1)
        { /* Put your
                code here ! */

                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); /* Refresh rc cleanup
                time */ 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);
        }
}
```

Example clarification



Here is the complete code for this sample project. This code realizes the serial communication with the aircraft and the Bluetooth pairing, connection and communication functions with the GameSir controller.

Through the program architecture flowchart, we can understand the implementation logic of this project sample.

If you just want to use it for further development, you only need to edit your code in the loop body of the user_task function, which will not affect the functions already implemented in the sample itself.

```
void user_task(void *arg) {

    while (1) {

        /* Put your code here ! */ delay(10);

    }
}
```

When pairing the handle with the TT expansion module, just press and hold the button in the TT expansion module for two seconds, and the TT expansion module will enter pairing mode. After waiting for the TT expansion module to automatically pair with the handle, you can use the GameSir handle to control the TT drone through the operation guide in the picture.

Engineering Templates - Interact with Tello EDU APP

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(argv2)) {

            if (rbpstr2buff(buff, argv2, 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 if (!strcmp(argv[3], "heart")) {

            if (mled_font2buff(buff, 0x104, argv[2][0], MLED_BRIGHT) != 0) {

                goto end;
            }

        } else
        {
            goto end;
        }

        matrix_effect_set_graph(buff);

    } else if ((!strcmp(argv[1], "sl")) && (argc == 3)) {

        int gcc = 127; if
        (sscanf(argv[2], "%d", &gcc)) {

            if ((gcc >= 0) && (gcc <= 255)) {

                tt_matrix.SetGCC(gcc);

            } else
            {
                goto end;
            }

        } else
        {
            goto end;
        }

    } else if (!move_param_is_valid(argv[1], argv[2])) {

        uint8_t mv_t = 0;

        if (sscanf(argv[3], "%d", &mv_t)) {

            if (!((mv_t >= 0) && (mv_t <= 10))) {

                goto end;
            }

        } else
        {
            goto end;
        }

        if (argc == 5) {

            matrix_effect_move_str(argv[4], strlen(argv[4]), argv[2][0], argv[1][0], (11 - mv_t) *
                                                    MOVE_UNIT);

        } else if ((argc == 4) && strlen(argv2)) {

            matrix_effect_move_str(argv2, strlen(argv2), argv[2][0], argv[1][0],
                                                    (11 - mv_t) * MOVE_UNIT);
```

```
                } 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],
    "bl")) {

        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 (rmtt_int_is_valid()) {

                int val = get_rmtt_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);
        }
}
```

Example clarification

Here is the complete code for this sample project. This code implements the serial communication with the drone and the function of calling samples for the ToF sensor, dot matrix screen and LED lights on the top of the machine.

Through the program architecture flowchart, we can understand the implementation logic of this project sample.

If you just want to use it for further development, you only need to edit your code in the loop body of the user_task function, which will not affect the functions already implemented in the sample itself.

```
void user_task(void *arg) {

        while (1) {

                /* Put your code here ! */ delay(10);

        }
}
```

You can then edit your own awesome programs while retaining the ability to control them with the Tello EDU APP.