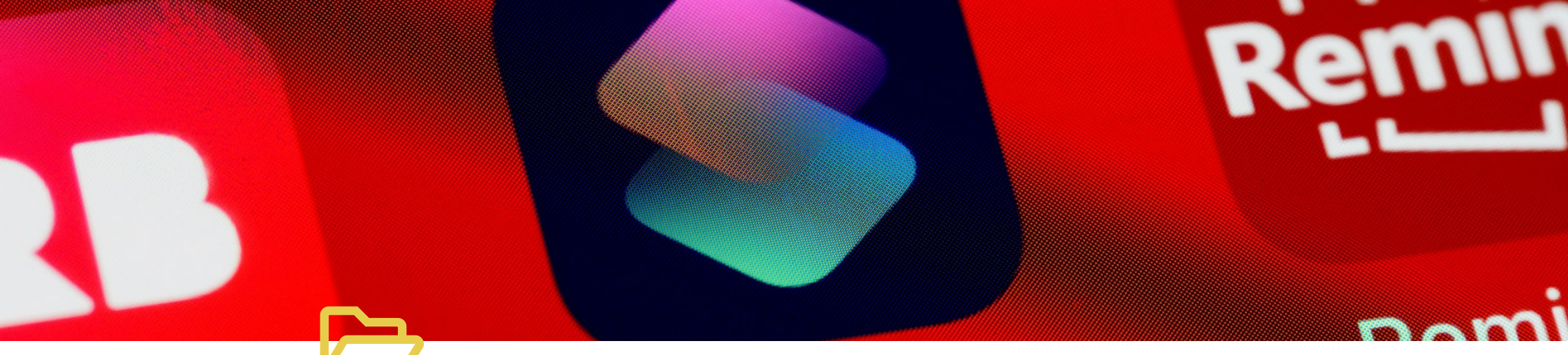


Blinker版

材料

硬件

- esp8266 nodeMCU 开发板
- 舵机

软件

- Arduino IDE
- Blinker APP

一句话原理

> 将 ESP8266 NodeMCU 看成一个可以联网的单片机，可以通过 Blinker 的按钮来控制单片机。

Blinker 配置

Blinker配置教程

代码

1）网络

```
#define BLINKER_WIFI
#define BLINKER_MIOT_LIGHT

#include <Blinker.h>

char auth[] = "5b76962ec546";  //点灯Key279babef0a2c
char ssid[] = "TX";      //wifi名称
char pswd[] = "20020603wz";     //wifi密码

// 新建组件对象
BlinkerButton Button1("test");

int counter = 0;

void miotPowerState(const String & state)//电源类操作（用户自定义电源类操作的回调函数）
{
    BLINKER_LOG("need set power state: ", state);
    /******************在此开始开启Blinker的操作*********************/

    /******************在此结束开启Blinker的操作*********************/

    /******************在此开始关闭Blinker的操作*********************/

    /******************在此结束关闭Blinker的操作*********************/
}

// 按下按键即会执行该函数
```

```c
void button1_callback(const String & state)
{
    BLINKER_LOG("get button state: ", state);
     /*******************在此开始开关灯的操作*********************/

    /*******************在此结束开关灯的操作*********************/

}

// 如果未绑定的组件被触发，则会执行其中内容
void dataRead(const String & data)
{
    BLINKER_LOG("Blinker readString: ", data);
    counter++;

}

void setup()
{
    // 初始化串口
    Serial.begin(115200);
    BLINKER_DEBUG.stream(Serial);
    /*******************在此开始开初始化的操作*********************/

    /*******************在此结束初始化的操作*********************/
    BlinkerMIOT.attachPowerState(miotPowerState);

    // 初始化blinker
    Blinker.begin(auth, ssid, pswd);
    Blinker.attachData(dataRead);

    Button1.attach(button1_callback);
}

void loop() {
    Blinker.run();
}
```

C

2）控制

在此用 servo.h 库

```c
#include <Servo.h>

Servo myservo;//定义舵机


void miotPowerState(const String & state)//电源类操作（用户自定义电源类操作的回调函数）
{
    BLINKER_LOG("need set power state: ", state);
    /*******************在此开始开启Blinker的操作*********************/
    if (state == BLINKER_CMD_ON) {

    myservo.write(170);//收到“on”的指令后舵机旋转170度（待修改）
    BlinkerMIOT.powerState("on");
    BlinkerMIOT.print();//反馈状态
    delay(1000);//延时1秒
    myservo.write(0);//舵机归零，回到垂直状态

    }
    /*******************在此结束开启Blinker的操作*********************/

    /*******************在此开始关闭Blinker的操作*********************/
    else if (state == BLINKER_CMD_OFF) {

     myservo.write(170);            //舵机偏转170°
     BlinkerMIOT.powerState("off");
     BlinkerMIOT.print();
     delay(1000);
     myservo.write(0);
    }
    /*******************在此结束关闭Blinker的操作*********************/
```

```
}

// 按下按键即会执行该函数
void button1_callback(const String & state)
{
    BLINKER_LOG("get button state: ", state);
   /*******************在此开始开关灯的操作*********************/
      if (state=="on")
    {
        myservo.write(170);//收到“on”的指令后舵机旋转 170
        delay(1000);//延时
        myservo.write(0);//舵机归零垂直

    } else if(state=="press"||state=="tap")
    {
       myservo.write(170);//长按开关按键后舵机旋转 170
       delay(1000);//延时
       myservo.write(0);//舵机归零垂直
    }
    /*******************在此结束开关灯的操作*********************/

}


void setup()
{
    /*******************在此开始开初始化的操作*********************/
    // 初始化舵机
    myservo.attach(2);//舵机的 IO 口，nodemcu 的 D4 口
    myservo.write(0);//上电时舵机归零垂直
    /*******************在此结束初始化的操作*********************/

}
```

C

3）总体代码

```
#define BLINKER_WIFI
#define BLINKER_MIOT_LIGHT

#include <Blinker.h>
#include <Servo.h>

Servo myservo;//定义舵机

char auth[] = "5b76962ec546";  //点灯 Key279babef0a2c
char ssid[] = "TX";      //wifi 名称
char pswd[] = "20020603wz";     //wifi 密码

// 新建组件对象
BlinkerButton Button1("test");

int counter = 0;

void miotPowerState(const String & state)//电源类操作（用户自定义电源类操作的回调函数）
{
    BLINKER_LOG("need set power state: ", state);
    /*******************在此开始开启 Blinker 的操作*********************/
    if (state == BLINKER_CMD_ON) {

    myservo.write(170);//收到“on”的指令后舵机旋转 170 度（待修改）
    BlinkerMIOT.powerState("on");
    BlinkerMIOT.print();//反馈状态
    delay(1000);//延时 1 秒
    myservo.write(0);//舵机归零，回到垂直状态

    }
    /*******************在此结束开启 Blinker 的操作*********************/

    /*******************在此开始关闭 Blinker 的操作*********************/
    else if (state == BLINKER_CMD_OFF) {
```

```
      myservo.write(170);             //舵机偏转170°
      BlinkerMIOT.powerState("off");
      BlinkerMIOT.print();
      delay(1000);
      myservo.write(0);
     }
     /*******************在此结束关闭Blinker的操作*********************/
}

// 按下按键即会执行该函数
void button1_callback(const String & state)
{
    BLINKER_LOG("get button state: ", state);
   /*******************在此开始开关灯的操作*********************/
      if (state=="on")
     {
         myservo.write(170);//收到“on”的指令后舵机旋转170
         delay(1000);//延时
         myservo.write(0);//舵机归零垂直

     } else if(state=="press"||state=="tap")
     {
        myservo.write(170);//长按开关按键后舵机旋转170
        delay(1000);//延时
        myservo.write(0);//舵机归零垂直
     }
     /*******************在此结束开关灯的操作*********************/
}

// 如果未绑定的组件被触发，则会执行其中内容
void dataRead(const String & data)
{
    BLINKER_LOG("Blinker readString: ", data);
    counter++;

}

void setup()
{
    // 初始化串口
    Serial.begin(115200);
    BLINKER_DEBUG.stream(Serial);
    /*******************在此开始开初始化的操作*********************/
    // 初始化舵机
    myservo.attach(2);//舵机的IO口，nodemcu的D4口
    myservo.write(0);//上电时舵机归零垂直
    /*******************在此结束初始化的操作*********************/

    // 初始化blinker
    Blinker.begin(auth, ssid, pswd);
    Blinker.attachData(dataRead);

    Button1.attach(button1_callback);
}

void loop() {
    Blinker.run();
}
```

C

效果展示

1）upload

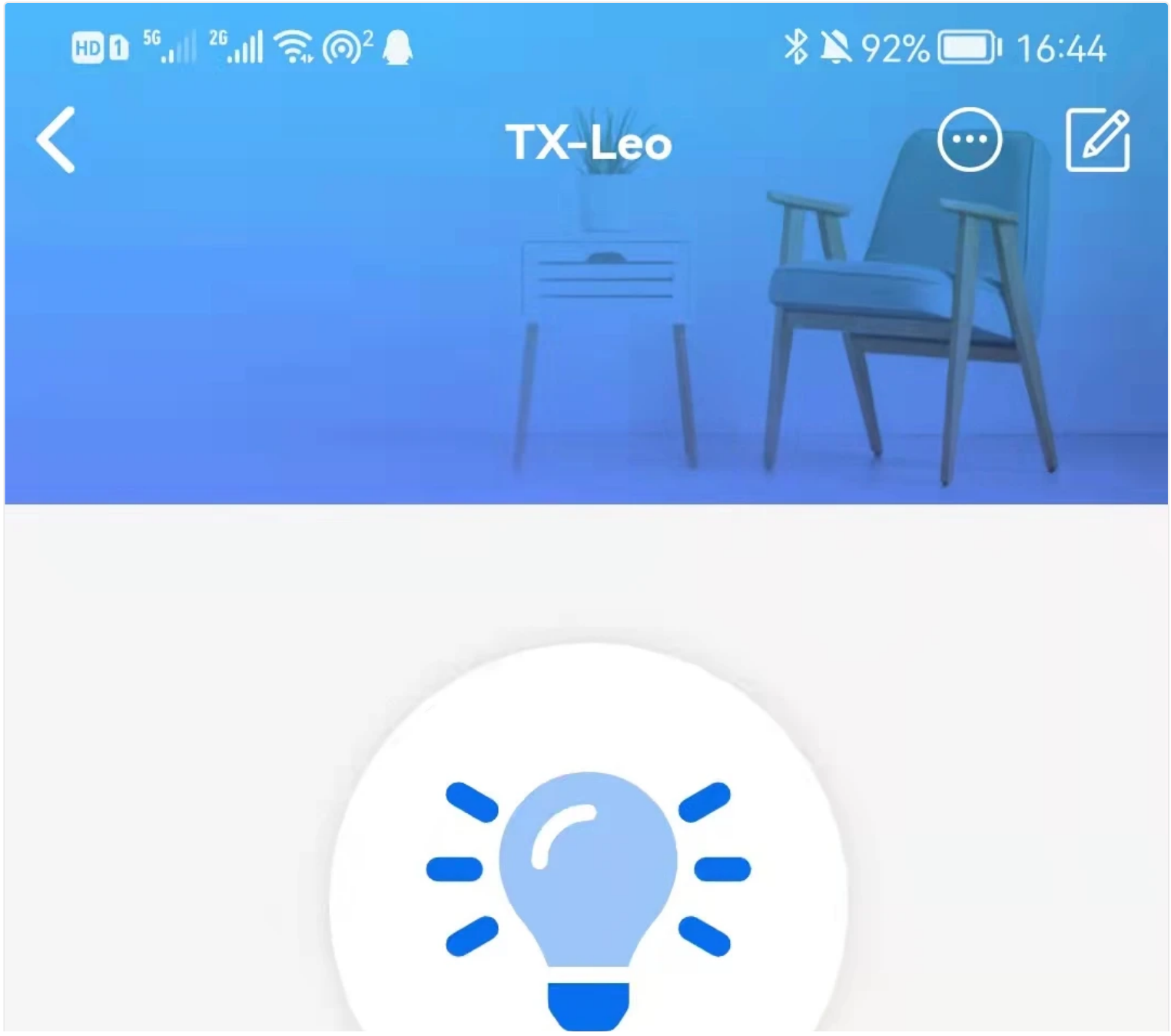
```
Done Saving.
Executable segment sizes:
ICACHE : 32768           - flash instruction cache
IROM   : 499556          - code in flash         (default or ICACHE_FLASH_ATTR)
IRAM   : 29737   / 32768 - code in IRAM          (IRAM_ATTR, ISRs...)
DATA   : 1740  )         - initialized variables (global, static) in RAM/HEAP
RODATA : 3848  ) / 81920 - constants             (global, static) in RAM/HEAP
BSS    : 30904 )         - zeroed variables      (global, static) in RAM/HEAP
Sketch uses 534881 bytes (51%) of program storage space. Maximum is 1044464 bytes.
Global variables use 36492 bytes (44%) of dynamic memory, leaving 45428 bytes for local variables. Maximum is 81920 bytes.
esptool.py v3.0
Serial port COM5
Connecting....
Chip is ESP8266EX
Features: WiFi
Crystal is 26MHz
MAC: 30:83:98:a2:cd:d6
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 539040 bytes to 384963...
```

```
Compressed 539040 bytes to 384963...
Writing at 0x00000000... (4 %)
Writing at 0x00004000... (8 %)
Writing at 0x00008000... (12 %)
Writing at 0x0000c000... (16 %)
Writing at 0x00010000... (20 %)
Writing at 0x00014000... (25 %)
Writing at 0x00018000... (29 %)
Writing at 0x0001c000... (33 %)
Writing at 0x00020000... (37 %)
Writing at 0x00024000... (41 %)
Writing at 0x00028000... (45 %)
Writing at 0x0002c000... (50 %)
Writing at 0x00030000... (54 %)
Writing at 0x00034000... (58 %)
Writing at 0x00038000... (62 %)
Writing at 0x0003c000... (66 %)
Writing at 0x00040000... (70 %)
Writing at 0x00044000... (75 %)
Writing at 0x00048000... (79 %)
Writing at 0x0004c000... (83 %)
Writing at 0x00050000... (87 %)
Writing at 0x00054000... (91 %)
Writing at 0x00058000... (95 %)
Writing at 0x0005c000... (100 %)
Wrote 539040 bytes (384963 compressed) at 0x00000000 in 34.0 seconds (effective 126.8 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

2）serial monitor

```
COM5

                  __        __                 __
16:38:35.759 -> /\ \      /\ \    __        /\ \              v0.3.80210803
16:38:35.759 -> \ \ \___  \ \ \  /\_\    ___\ \ \/'\      __   _ __
16:38:35.759 ->  \ \ '__`\\ \ \ \/\ \ /' _ `\ \ , <    /'__`\/\`'__\
16:38:35.759 ->   \ \ \L\ \\ \ \_\ \ \/\ \/\ \ \ \\`\ /\  __/\ \ \./
16:38:35.807 ->    \ \_,__/ \ \__\\ \_\ \_\ \_\ \_\ \_\ \____\\ \_\
16:38:35.807 ->     \/___/   \/__/ \/_/\/_/\/_/\/_/\/_/\/____/ \/_/
16:38:35.807 ->     To better use blinker with your IoT project!
16:38:35.807 ->     Download latest blinker library here!
16:38:35.807 ->     => https://github.com/blinker-iot/blinker-library
16:38:35.807 ->
16:38:35.807 -> [107] Connecting to TX
16:38:35.807 -> [112] ESP8266_MQTT initialized...
16:38:35.807 -> [112]
16:38:35.807 -> ============================================================
16:38:35.807 -> ================== Blinker Timer loaded! ==================
16:38:35.807 ->      EEPROM address 1536-2431 is used for Blinker Timer!
16:38:35.807 -> ========= PLEASE AVOID USING THESE EEPROM ADDRESS! ========
16:38:35.852 -> ============================================================
16:38:35.852 ->
16:38:43.640 -> [7933] WiFi Connected.
16:38:43.640 -> [7933] IP Address:
16:38:43.640 -> [7933] 192.168.43.220
16:38:43.918 -> [8213] Freeheap: 36144
16:38:43.918 -> [8215] mDNS responder started
16:38:43.918 -> [8216] webSocket_MQTT server started
16:38:43.918 -> [8216] ws://5DB0A67A43ZKEBA1NN4WFQI4.local:81
16:38:45.455 -> [9741] ========================================================
16:38:45.455 -> [9741] =========== Blinker Auto Control mode init! ===========
16:38:45.455 -> [9741]      EEPROM address 0-1279 is used for Auto Control!
16:38:45.455 -> [9746] ======= PLEASE AVOID USING THESE EEPROM ADDRESS! ======
16:38:45.455 -> [9752] ========================================================
16:38:45.455 -> [9760] Connecting to MQTT...
16:38:45.455 -> [9760] reconnect_time: 0
16:38:45.688 -> [9983] MQTT Connected!
16:38:45.688 -> [9983] Freeheap: 37248
16:39:08.014 -> [32305] get button state: on
16:39:33.416 -> [57742] get button state: on
```

3）Blinker


92%
16:44
TX-Leo