# GW3323
# Printer 开发板
# 使用说明书

**版本：V2.0**

# 目 录


**1 编译环境说明 ..... 2**

1.1 CODEBLOCK(IDE): (CODEBLOCKS-17.12)代码编辑器, 编译链接时会调用到 TOOLCHAIN 中提供的工具. 最终生成烧写用的 DCF 文件 ..... 2

1.2 把 APP.CBP 拖入 PROJECT 中, 即可打开工程进行编译了 ..... 3

1.3 TOOLCHAIN: (RV32-TOOLCHAIN-SETUP_VXXX)包含 RISC-V 编译器, BIN 文件转换工具等. ..... 4

1.4 DOWNLOADER 是烧录工具软件兼顾串口打印的功能，可以供开发人员调试，CP210X_WINDOWS_DRIVERS 是 XLINK 烧录器的驱动程序。 ..... 4

**2 工程的介绍 ..... 6**

**3 下载说明 ..... 12**

**4 例程展示 ..... 15**

**5 空中升级 ..... 23**

**6 常用设置 ..... 26**

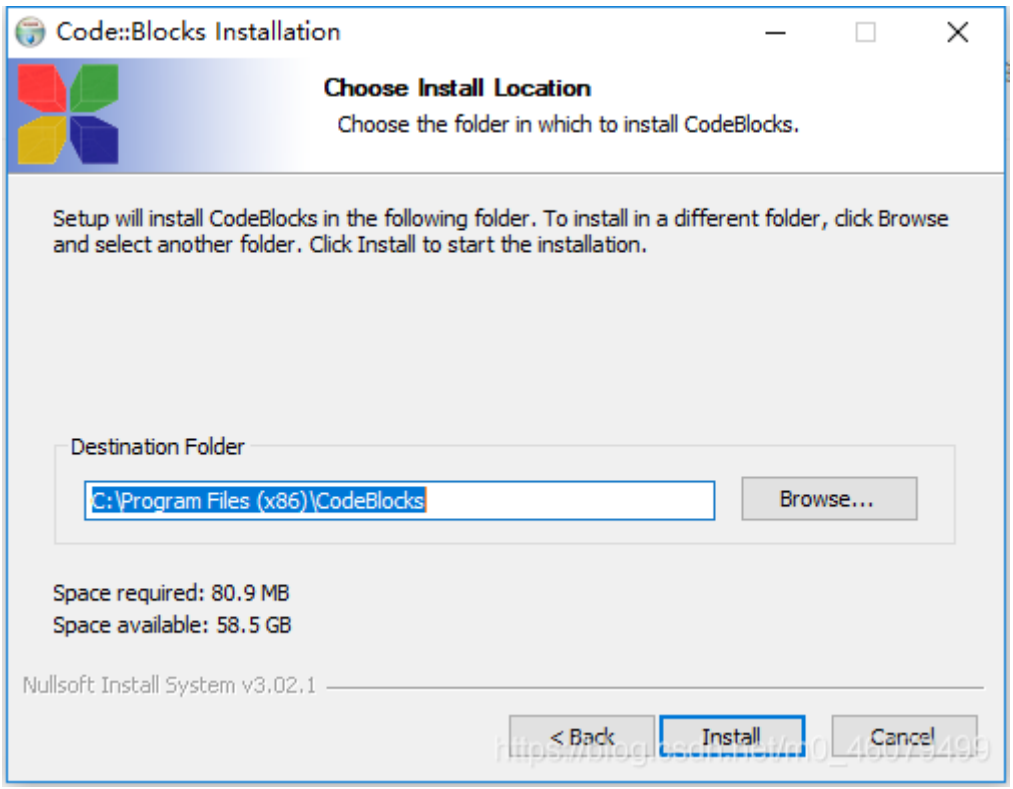
**7 版本历史 ..... 28**

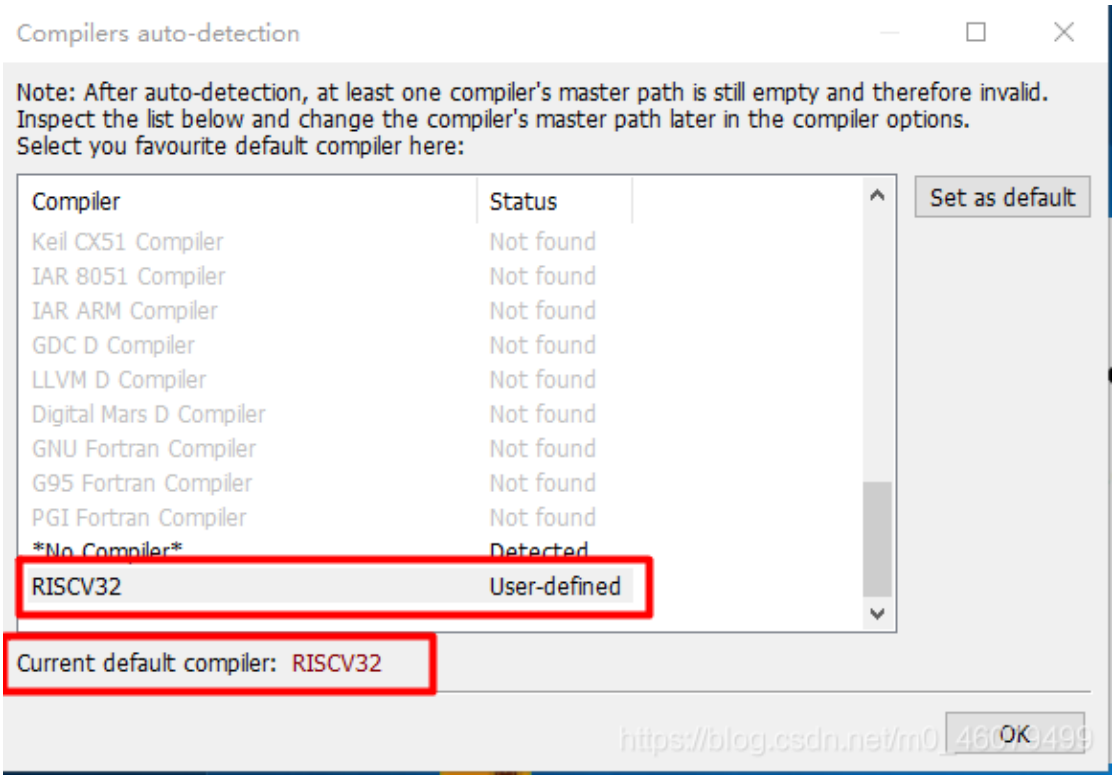
# 1 编译环境说明

GW3323 此款芯片的开发环境是“Codeblocks.exe”,安装包见“codeblocks-20.03mingw-setup.exe”. **先安装 CodeBlocks, 再安装 RV32-Toolchain** (安装 ToolChain 时，会向 CodeBlocks 注册配置相关编译环境)

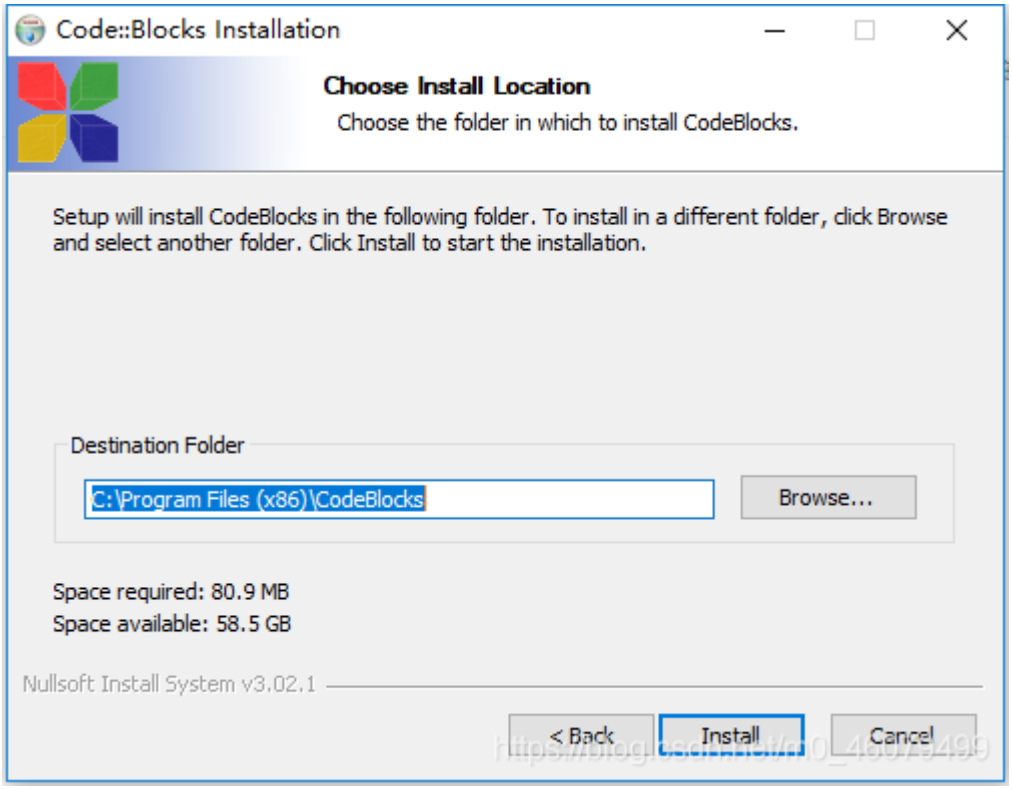
在开发中，我们一般是使用**串口打印**或者 GPIO 口和逻辑分析仪进行调试，**暂不支持断点调试和仿真**。

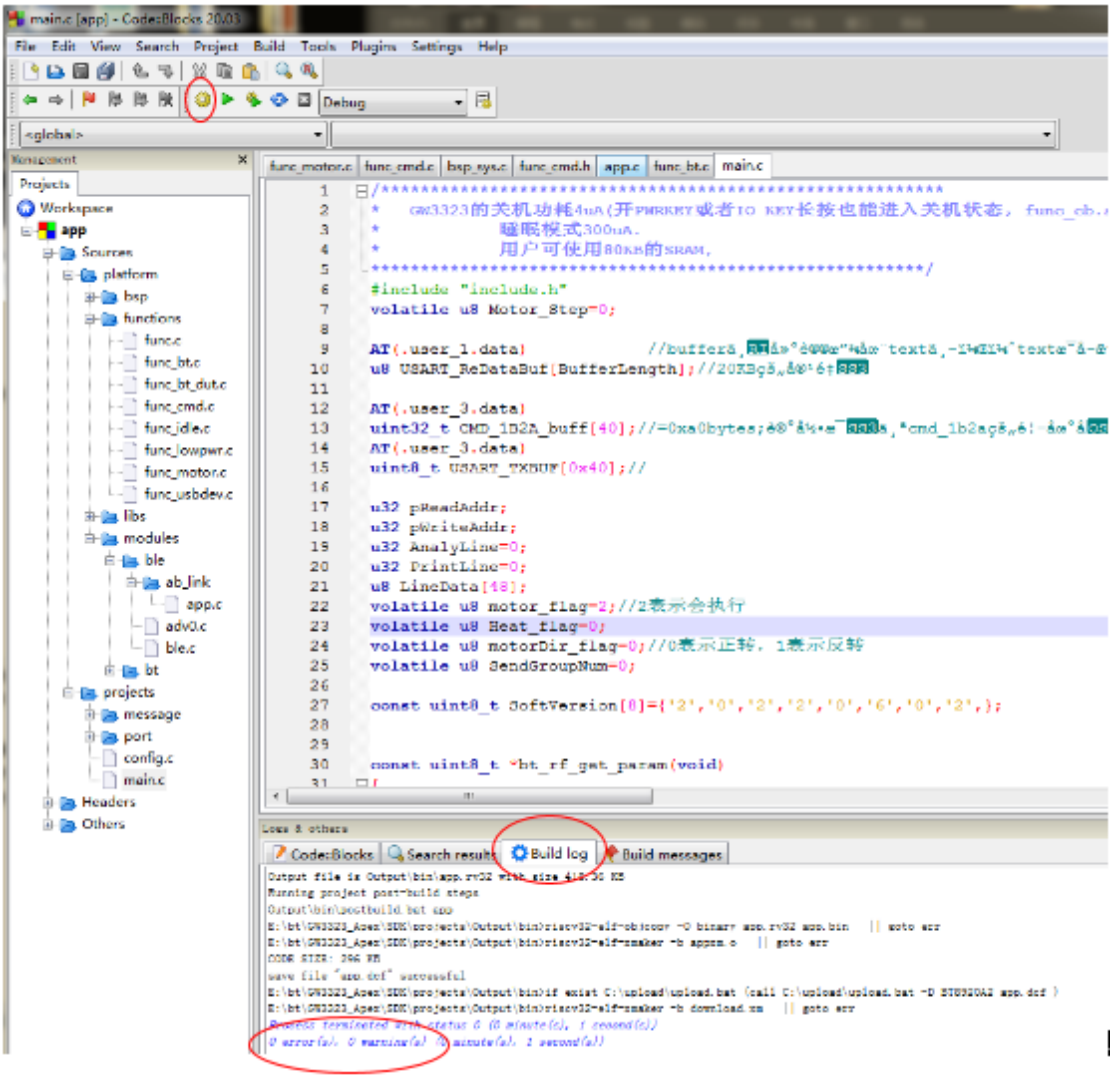
## 1.1 CodeBlock(IDE): (codeblocks-17.12)代码编辑器，编译链接时会调用到 ToolChain 中提供的工具. 最终生成烧写用的 dcf 文件.






1.2 把 **app.cbp** 拖入 **project** 中，即可打开工程进行编译了



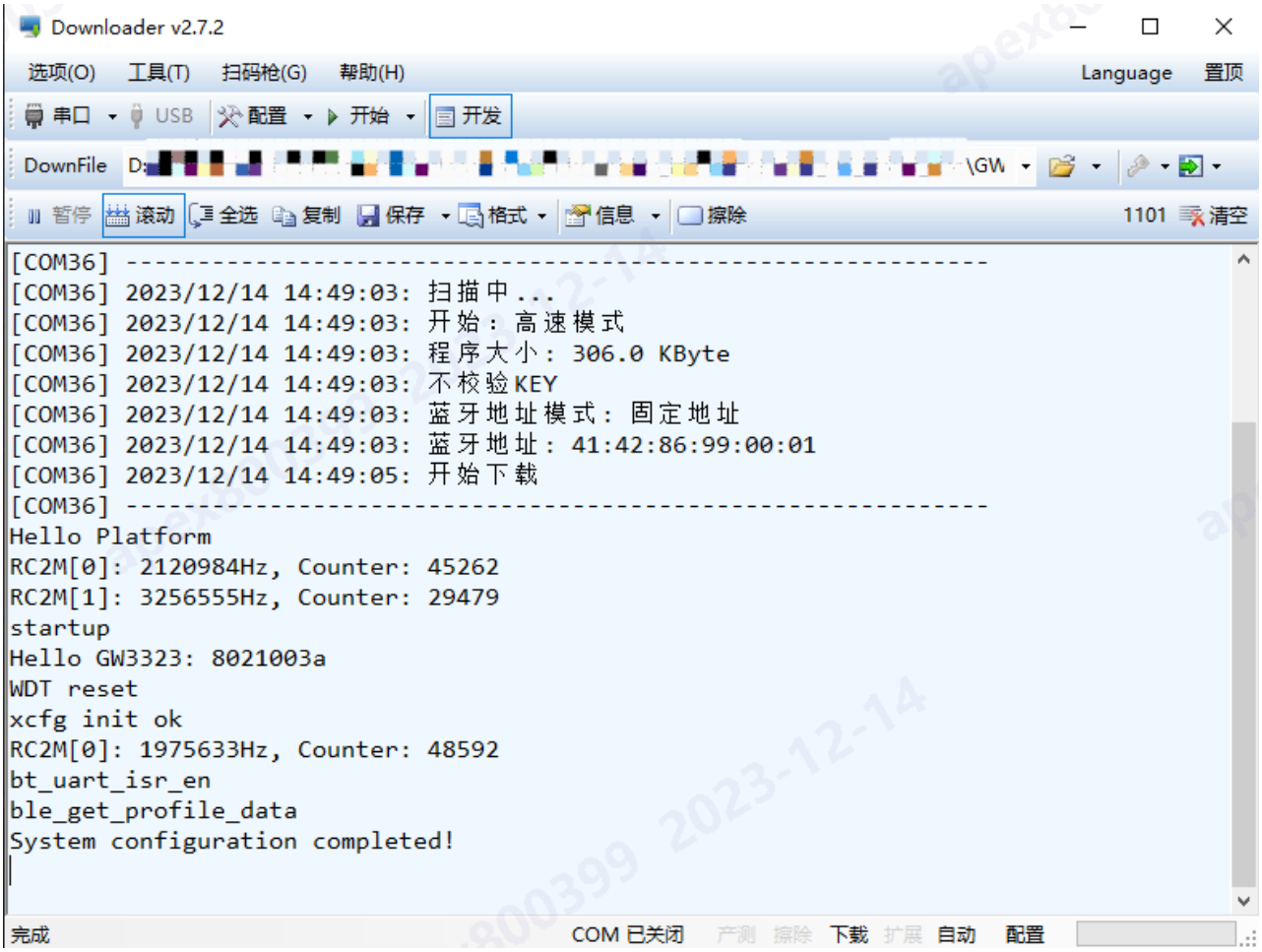


1.3 **ToolChain: (RV32-Toolchain-Setup_vxxx)包含 RISC-V 编译器, Bin 文件转换工具等.**

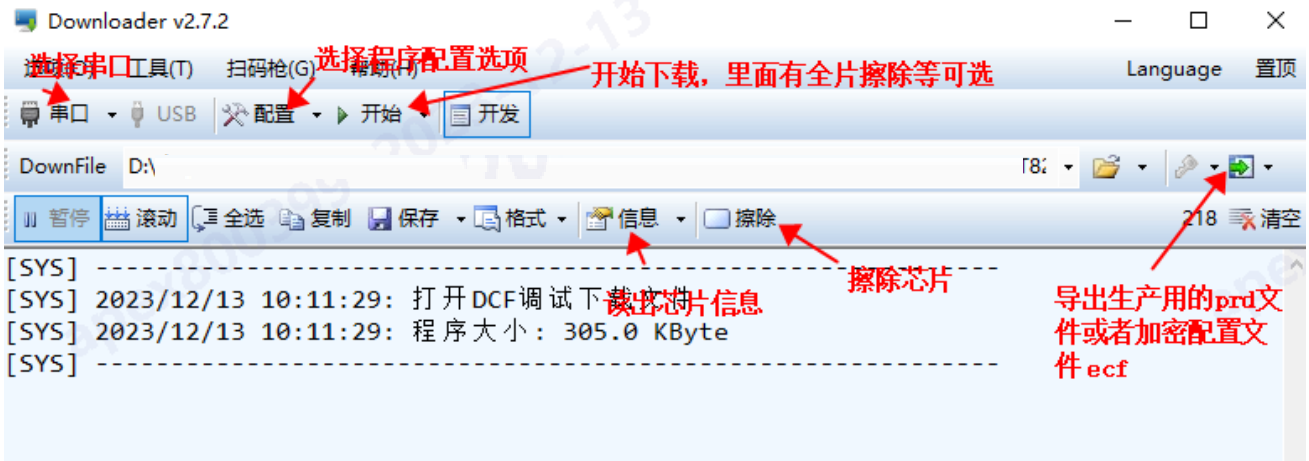
| | | | |
|---|---|---|---|
| RV32-Toolchain-Setup.exe | 2021/12/30 20:48 | 应用程序 | 24,977 KB |

1.4 **downloader 是烧录工具软件兼顾串口打印的功能，可以供开发人员调试，CP210x_Windows_Drivers 是 Xlink 烧录器的驱动程序。**

| | | | |
|---|---|---|---|
| CP210x_Windows_Drivers.rar | 2022/3/11 17:08 | WinRAR 压缩文件 | 3,656 KB |
| Downloader_v2.7.2.zip | 2022/3/11 16:38 | WinRAR ZIP 压缩... | 2,488 KB |

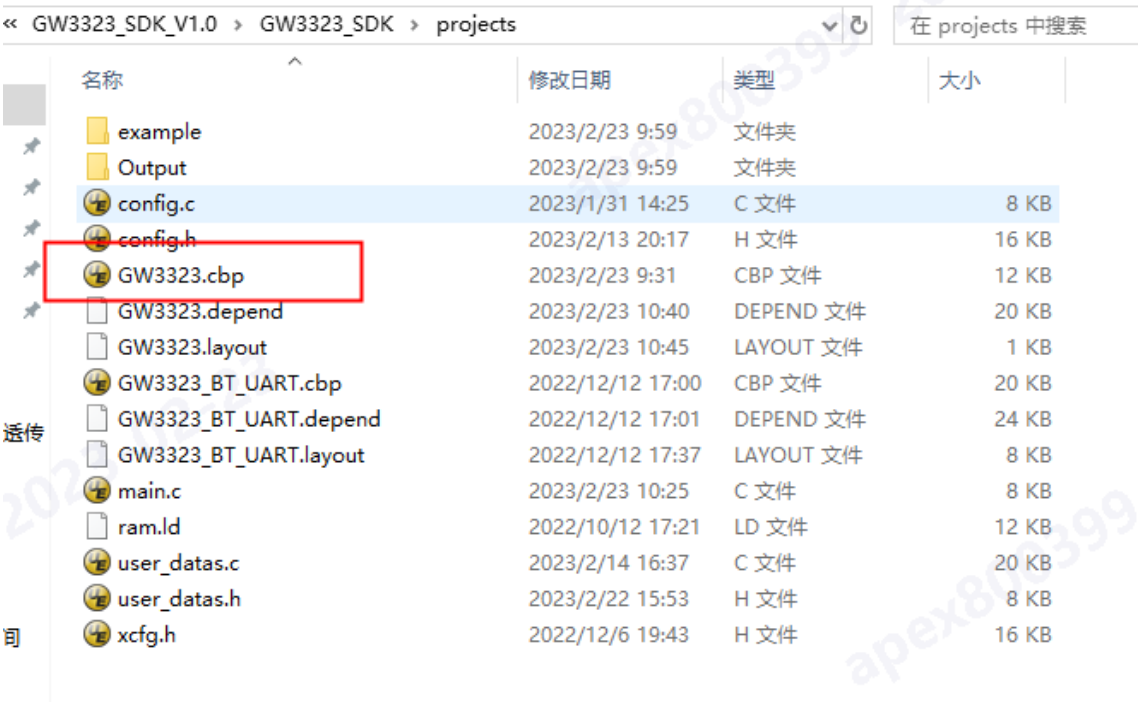



开发人员一般选中下图中的"开发"，可以方便下载后查看打印信息.

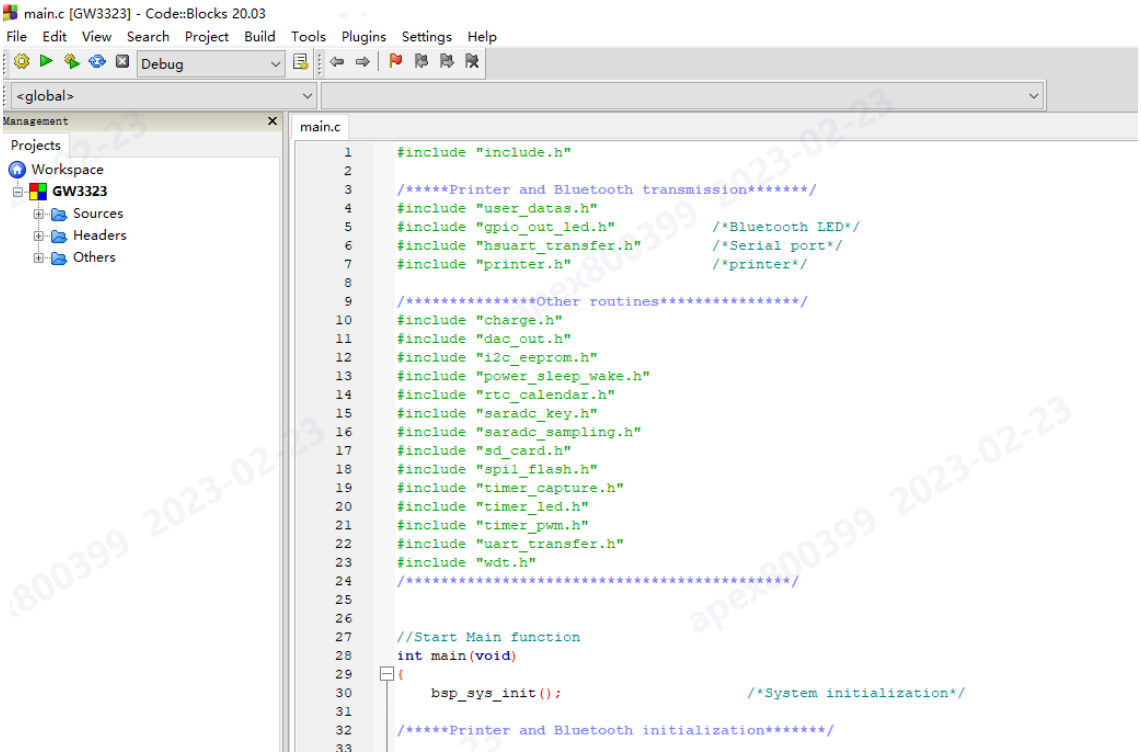


# 2 工程的介绍

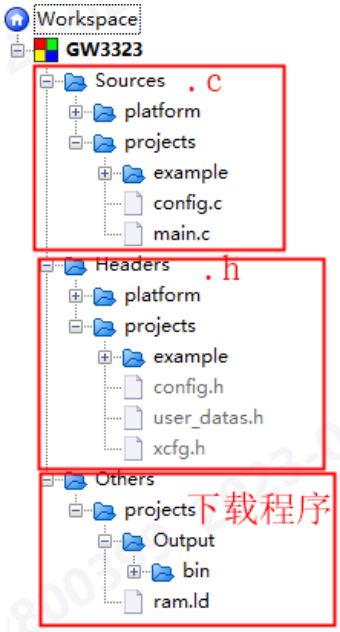
将“GW3323_SDK_V1.0\GW3323_SDK\projects\GW3323.cbp”文件用“codeBlocks”开发工具打开。



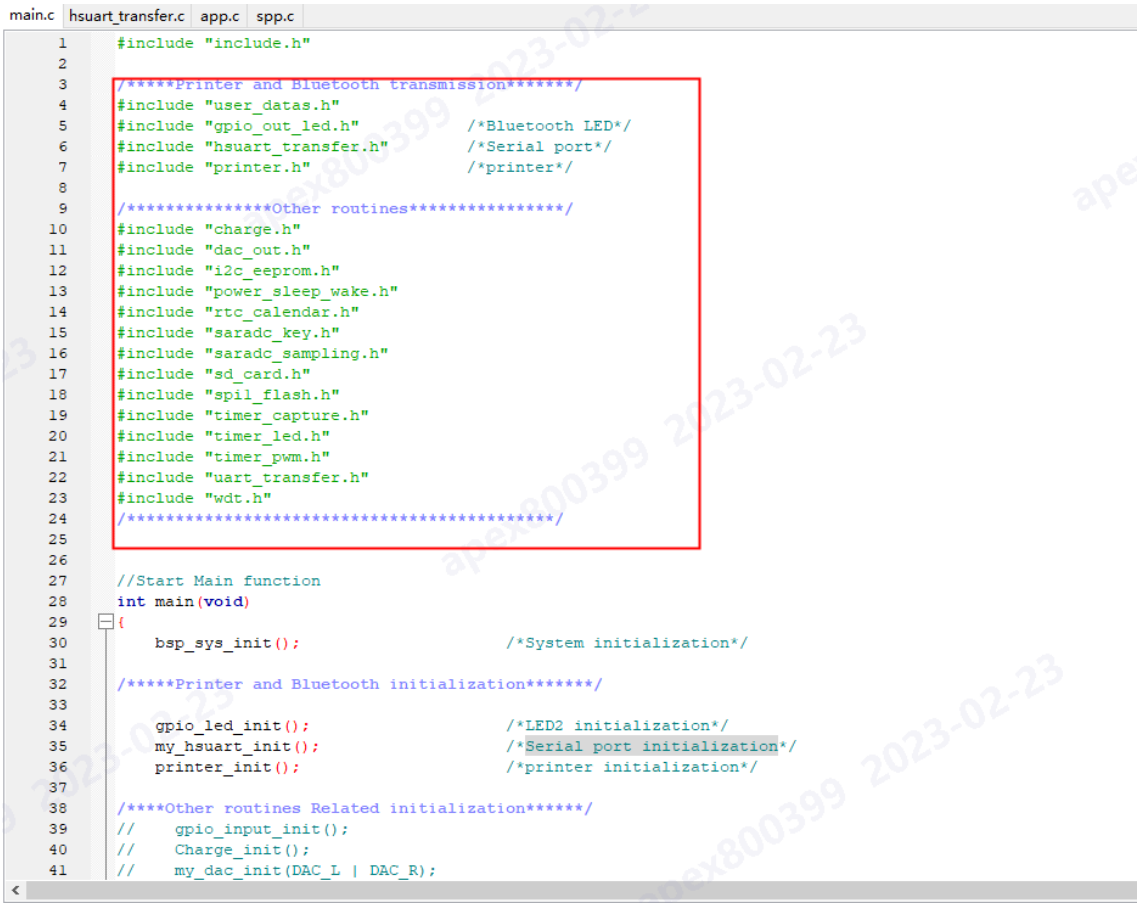

打开后出现如下界面：



左侧三个文件夹分别为放着工程文件的 xxx.c 和 xxx.h 文件、Others 文件夹下是输出下载文件




点击 main.c 可看到所有的示例程序




GW3323_SDK_V1.0 工程文件默认打开蓝牙透传功能和打印机打印固定字符功能，如需使用其它例程功能，打开对应初始化及示例函数即可。

例：

如需使用 IIC 功能，即可取消对应的 IIC_AT24C01_init()初始化函数和 IIC_AT24C01_example()函数的注释，并注释掉其它不用的初始化函数。

main.c | hsuart_transfer.c | app.c | spp.c

```
28  int main(void)
29  {
30      bsp_sys_init();                         /*System initialization*/
31
32  /*****Printer and Bluetooth initialization*******/
33
34  //    gpio_led_init();                      /*LED2 initialization*/
35  //    my_hsuart_init();                     /*Serial port initialization*/
36  //    printer_init();                       /*printer initialization*/
37
38  /****Other routines Related initialization******/
39  //    gpio_input_init();
40  //    Charge_init();
41  //    my_dac_init(DAC_L | DAC_R);
42      IIC_AT24C01_init();
43  //    power_sleep_wake_init();
44  //    rtc_calendar_init();
45  //    timer3_cap_init(66666);
46  //    timer3_init(500000 - 1);          // 500ms, f=1MHz
47  //    timer3_pwm_init(1000 - 1);        // Fsrc=1MHz, Fpwm=1KHz
48  //    uart2_init(115200);
49  //    Adc_key_init();
50  //    Adc_sampling_init();
51  //    sd_disk_init();
52  //    spi1_init(SPIFALSH_BAUD);
53  /*******************************************/
54      while (1)
55      {
56  //        LED_Bluetooth();                  /*LED2 is always on after Bluetooth
57  //        hsuart_transfer_example();        /*Wait for reception interruption -
58  //        printer_example();                /*Print fixed font "Geehy"*/
59
60  /****Other routines Execute functionn******/
61  //    gpio_input_key_example();
62  //    Charge_example();
63  //    my_dac_out_example();
64  //    gpio_out_led_example();
65      IIC_AT24C01_example();
66  //    power_sleep_wake_example();
67  //    rtc_calendar_example();
68  //    timer3_capture_example();
```

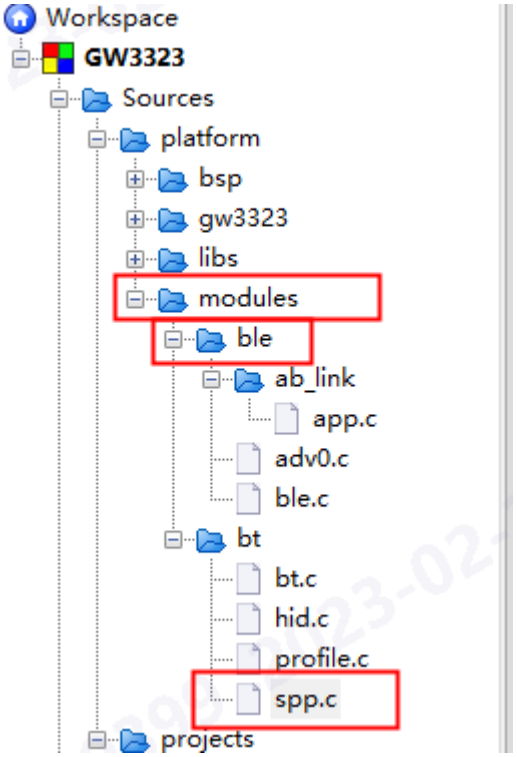
在 bsp_sys.c 里存放系统初始化的一些函数

```
24  /*******************************************/
25
26
27  //Start Main function
28  int main(void)
29  {
30      bsp_sys_init();                         /*System initialization*/
31
32  /*****Printer and Bluetooth initialization*******/
33
34      gpio_led_init();                        /*LED2 initialization*/
35      my_hsuart_init();                       /*Serial port initialization*/
36      printer_init();                         /*printer initialization*/
37
38  /****Other routines Related initialization******/
```

main.c | hsuart_transfer.c | app.c | spp.c | bsp_sys.c

```
590     }
591 }
592
593 u8 ack_Name[31]={"BT"};
594 AT(.text.bsp.sys.init)
595 void bsp_sys_init(void)
596 {
597     /*Power-on transmission test serial port*/
598     u32 lvdcon = LVDCON;
599     printf("Hello GW3323: %x\n", lvdcon);
600     if(lvdcon & BIT(18))        printf("WKO reset\n");
601     else if(lvdcon & BIT(17))   printf("VUSB reset\n");
602     else if(lvdcon & BIT(16))   printf("WDT reset\n");
603     else if(lvdcon & 0xf00)     printf("SW reset\n");
604
605     /// config
606     if (!xcfg_init(&xcfg_cb, sizeof(xcfg_cb))) {          //Get configuration
607         printf("xcfg init error\n");
608     }
609     else
610         printf("xcfg init ok\n");
611
612     // io init
613     bsp_io_init();
614
615     // var init
616     bsp_var_init();
617
618     // power init
619     pmu_init((BUCK_CURR_LIMIT_DIS << 7) | BUCK_MODE_EN);
620     adpll_init(0);
621     // clock init
622     set_sys_clk(SYS_CLK_SEL);
623
624     // peripheral init
625     rtc_init();
626     param_init(sys_cb.rtc_first_pwron);
627
```

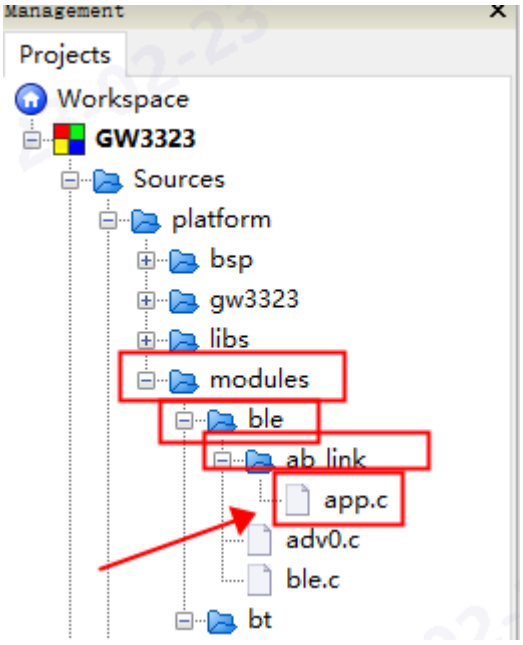
关于 SPP 蓝牙的控制在 spp.c 中：



main.c | hsuart_transfer.c | app.c | spp.c | bsp_sys.c

```
66      if(g_ucBleConnect != 0)
67      {
68          gpio_reset_bits(GPIOA_REG, GPIO_PIN_7);
69      }
70      printf("--->spp_disconnect_callback\n");
71  #if BT_SPP_FOT_EN
72      fot_spp_disconnect_callback();
73  #endif // BT_SPP_FOT_EN
74  }
75
76  extern uint8_t  print_pic;              /*  Judge whether Bluetooth sends 1d34    --Print command*/
77
78  u8 BtFlowControl_start=0;
79  u8 SPP_CreditCount=0;
80
81  extern volatile u8 g_ucAT_flag;
82  void spp_rx_callback(uint8_t *packet, uint16_t size)
83  {
84  #if BT_SPP_FOT_EN
85      if(fot_app_connect_auth(packet, size))
86      {
87          fot_recv_proc(packet,size);
88          return;
89      }
90  #endif // BT_SPP_FOT_EN
91
92      /*Classic Bluetooth - receiving Bluetooth data and sending it to serial port*/
93      printf("spp_rx_callback :%d,%d\n",SPP_CreditCount,size);          /*Test serial port*/
94      hsuart_dma_start(HSUT_TRANSMIT, (uint32_t)packet, size);          /*Send the received data through the tx pin of the seri
95      spp_set_rx_new_credit(1);                                         /*Close flow control*/
96
97      if(size == 2)                                                     /*Judge the print command*/
98      {
99          if( packet[0] == 0x1d && packet[1] == 0x34)
100             print_pic = 1;
101     }
102
103 }
104
```

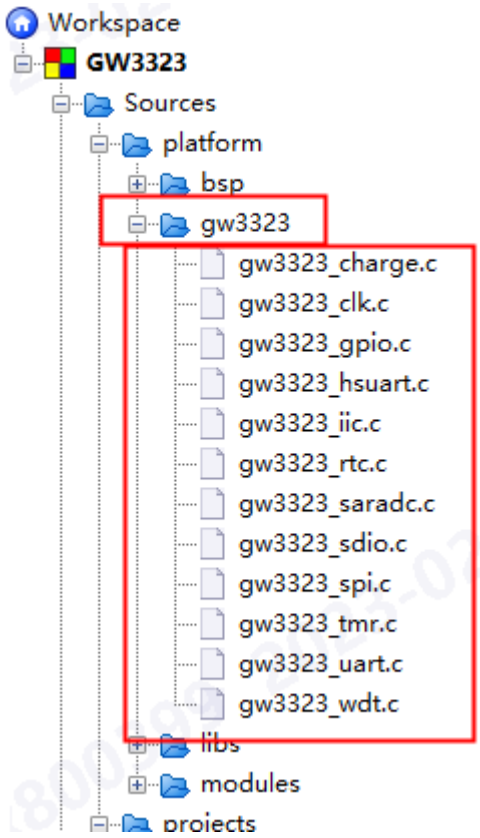
关于 BLE 蓝牙的控制在 app.c 中:



main.c | hsuart_transfer.c | app.c | spp.c | bsp_sys.c

```
160     //          {
161     //              buf[0] = 0x11;
162     //              ble_tx_notify(gatts_Datas_Characteristic_base.att_index, buf, 1);
163     //              BtFlowControl_start = 0;
164     ////               printf(" BTFlowsControl_Start \n");
165     //          }
166     //      }
167     }
168
169     /*Low-power Bluetooth*/
170     extern volatile u8 g_ucAT_flag;
171     extern uint8_t  print_pic;                                          /*  Judge the print c
172     static uint8_t gatt_callback_app(u8 *ptr, u16 len)
173     {
174         g_ucAT_flag = 0;
175         hsuart_dma_start(HSUT_TRANSMIT, (uint32_t)ptr, len);            /*Send the received
176         printf("[%x]%x %x...%x\n",len,ptr[0],ptr[1],ptr[len-1]);        /*Test serial port*
177
178         if(len == 2)                                                    /*Judge the print
179         {
180             if( ptr[0] == 0x1d && ptr[1] == 0x34)
181                 print_pic = 1;
182         }
183
184         return false;
185     }
186     /**********/
187
```

gw3323 工程文件中放置外设的库函数，自己写函数时需调用这些函数。

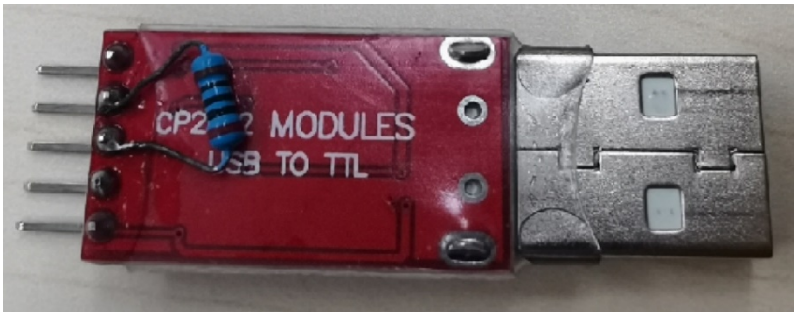


# 3 下载说明

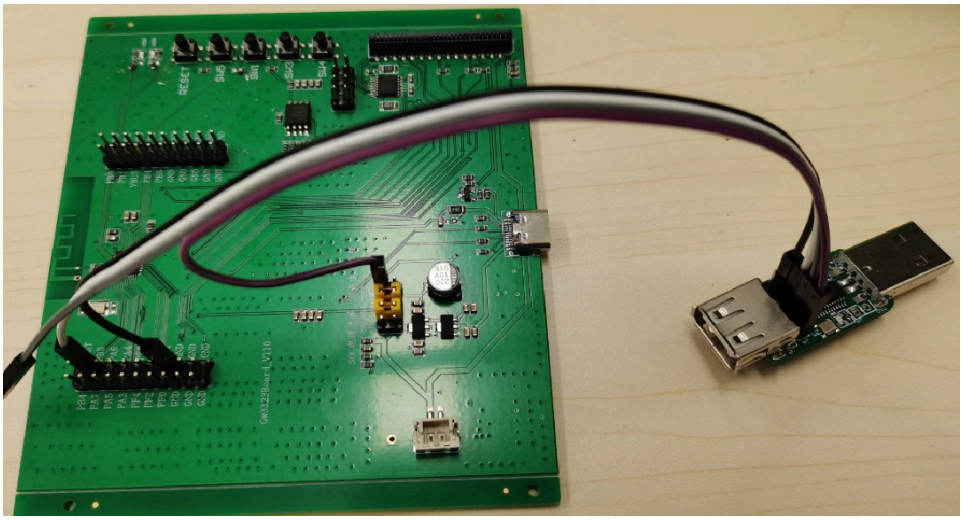
1、 用 CP2102 的串口模块，或我司自己的串口模块连接开发板。

极海的 XLink 模块: RX---->接 GW3323 的 PB3;

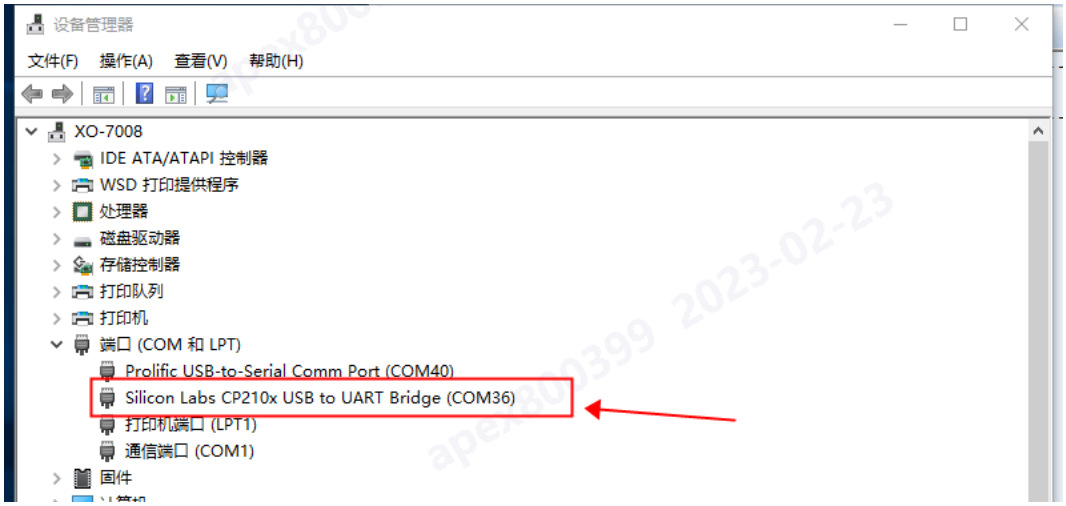
CP2102 串口模块： TX -- 串 200R -- RX （ 拉 RX 这根线到主控）--->接 GW3323 的 PB3;




2、 串口模块的（GND,RX,3.3V）3 根线接到开发板（GND,PB3,3.3V）上，另一端接电脑的 USB 口。

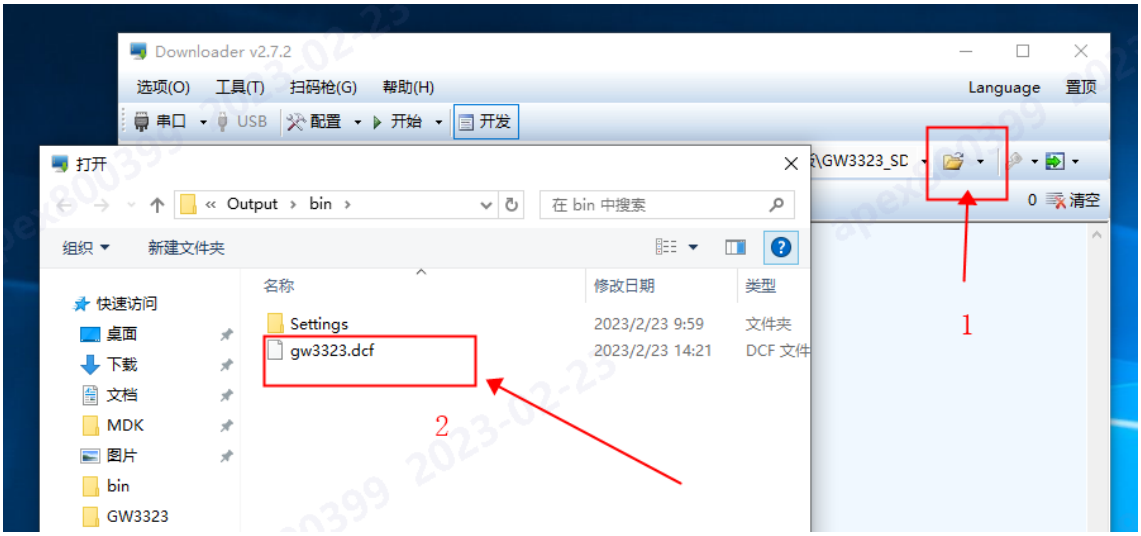


3、 连接串口后，“串口图标变黑”，表明串口已接上，否则查看串口硬件问题或者“CP210x_Windows_Drivers”驱动安装的问题。

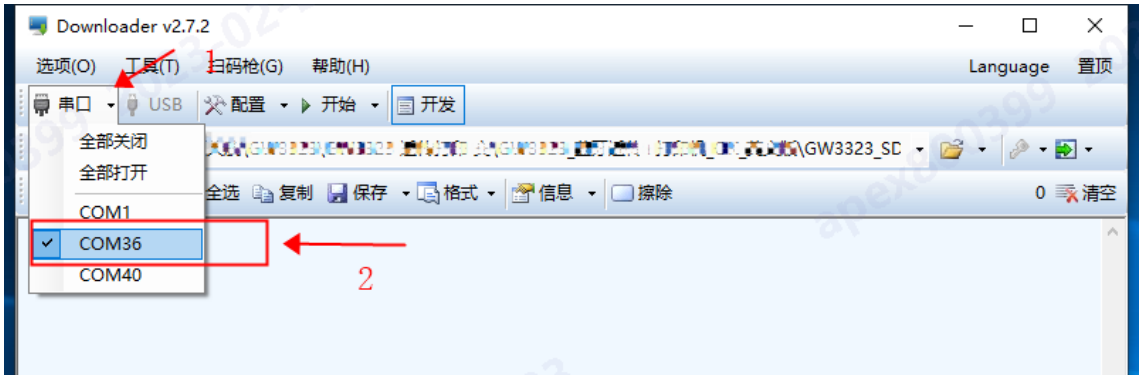


4、 然后再选择工程烧录文件.dcf;再选择开始烧录。如下图：

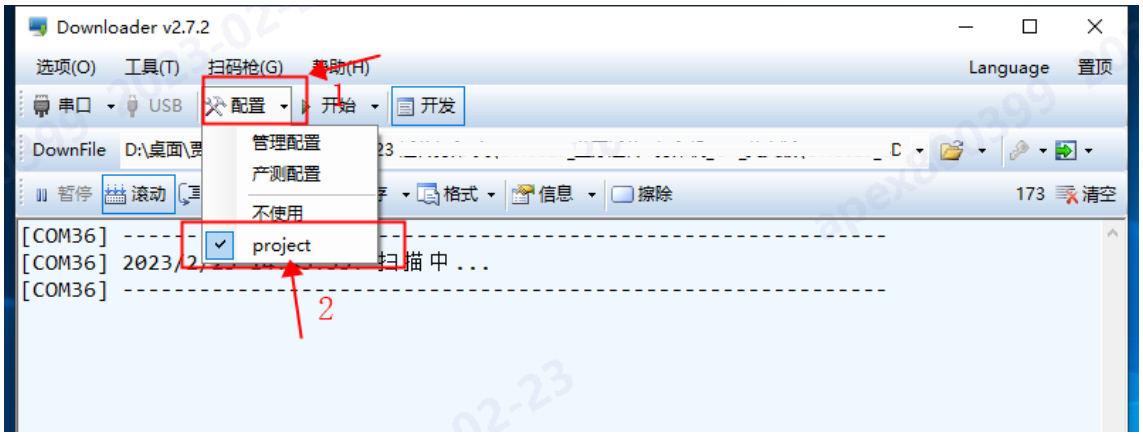
(1)例程的烧录文件在 GW3323_SDK_V1.0\GW3323_SDK\projects\Output\bin\gw3323.dcf



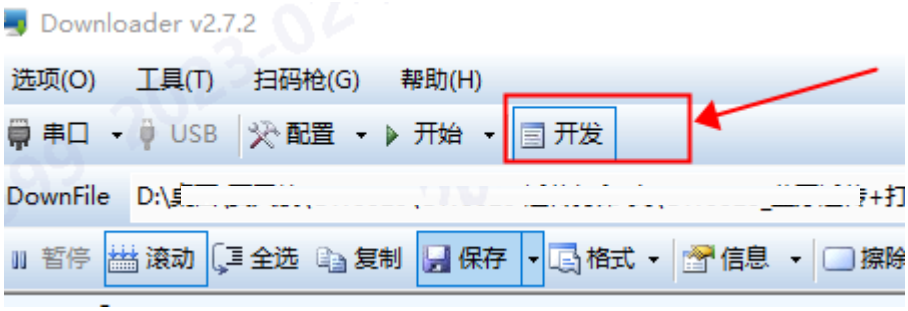

(2)选择 CP2102 串口在电脑上的串口号



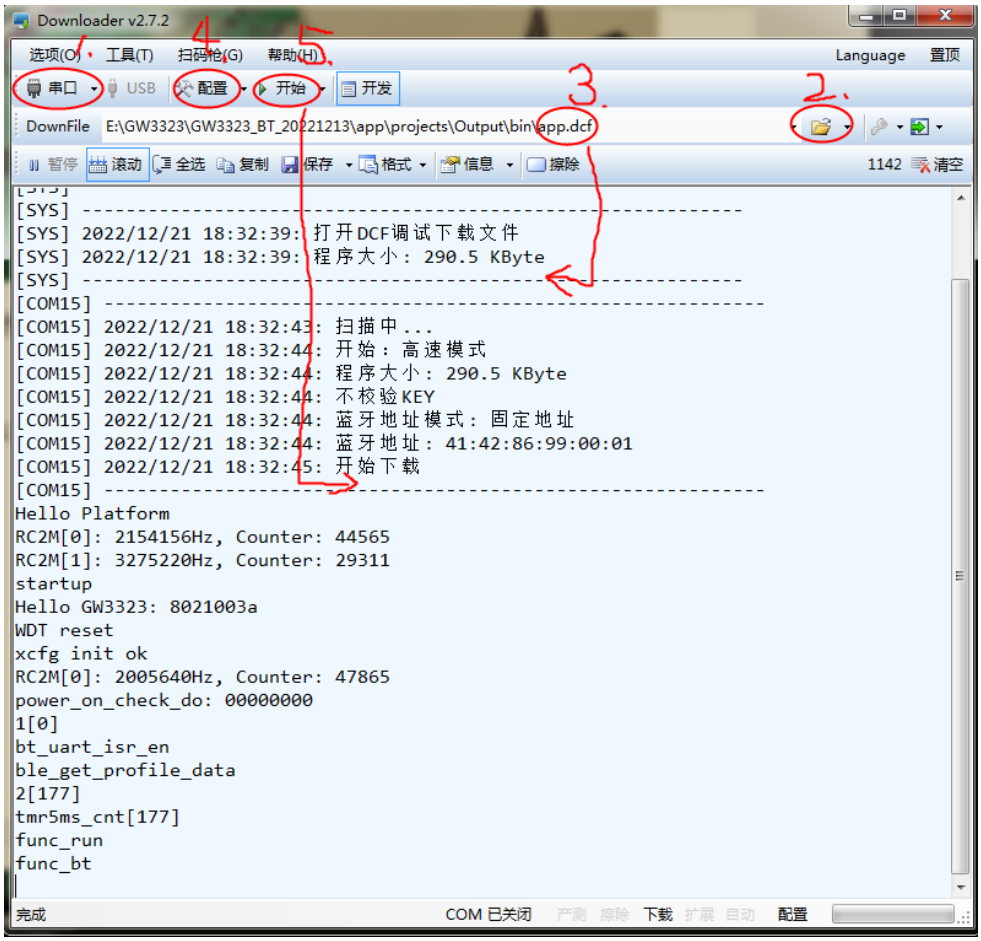

(3)点击配置 project




(4)点击开发按钮




(5)点击开始即可下载程序



# 4 例程展示

1、默认例程（GW3323_SDK_V1.0）是“打印机测试与蓝牙透传”，示例程序包括：

打印机输出固定字符：

按压开发版的 SW5 按键，能控制打印机输出“Geety”字符

也可连接蓝牙通过 16 进制指令->0x1d34 控制打印机输出固定字符“Geety”

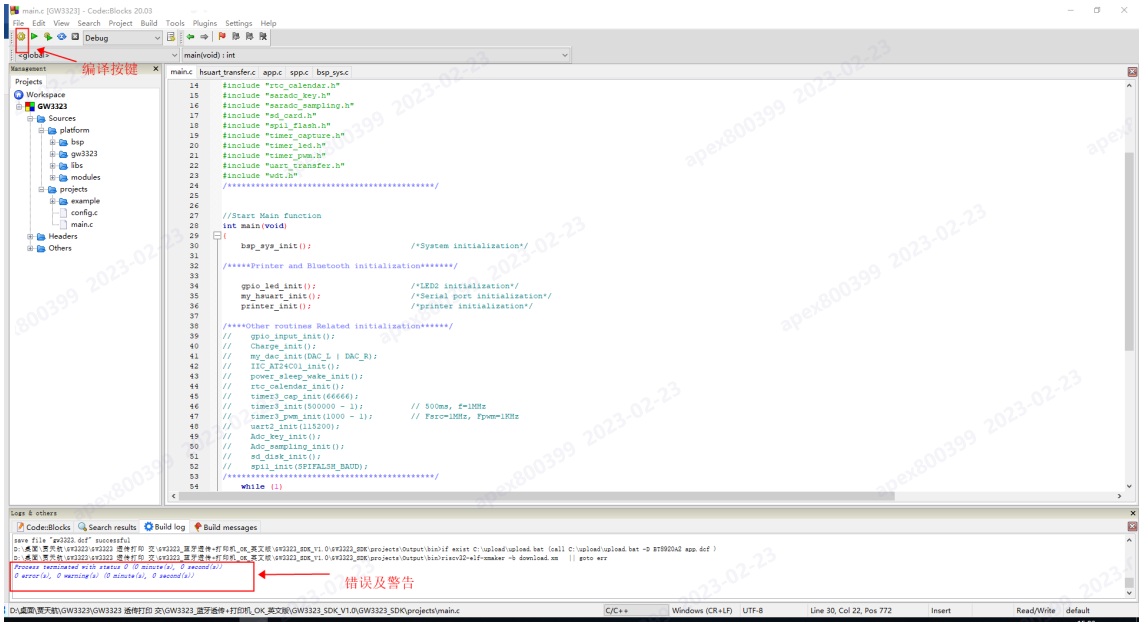
蓝牙透传:

蓝牙可与串口通讯，实现手机发送数据可通过高速串口（PB1 和 PB2）发送到电脑或其它开发板

PB1 为开发板的 RX 引脚

PB2 为开发板的 TX 引脚

2、更改例程后要进行编译，“0 错误 0 警告”后才能正常烧录。




3、蓝牙透传主要用到 spp.c（经典蓝牙）和 app.c（低功耗蓝牙）的两个文件，蓝牙接收到数据将自动进入中断，我们只需把接收到的数据发送给串口即可实现蓝牙透传。

4、spp.c 中，函数支持经典蓝牙协议，可在手机上下载“spp 蓝牙串口”APP 与开发板进行连接，在程序中我们将接收到的数据放在 packet 数组中，通过高速串口 PB1 和 PB2 发送到电脑端。

5、app.c 中，函数支持低功耗蓝牙协议，可在手机上下载“BLE 调试助手”APP 与开发板进行连接，在程序中我们将接收到的数据放在 ptr 数组中，通过高速串口 PB1 和 PB2 发送到电脑端。

main.c | hsuart_transfer.c | app.c | *spp.c

```
66      if(g_ucBleConnect == 0)
67      {
68          gpio_reset_bits(GPIOA_REG, GPIO_PIN_7);
69      }
70      printf("--->spp_disconnect_callback\n");
71  #if BT_SPP_FOT_EN
72      fot_spp_disconnect_callback();
73  #endif // BT_SPP_FOT_EN
74  }
75
76  extern uint8_t  print_pic;              /*  Judge whether Bluetooth sends 1d34    --Print command*/
77
78  u8 BtFlowControl_start=0;
79  u8 SPP_CreditCount=0;
80
81  extern volatile u8 g_ucAT_flag;
82  void spp_rx_callback(uint8_t *packet, uint16_t size)
83  {
84  #if BT_SPP_FOT_EN
85      if(fot_app_connect_auth(packet, size))
86      {
87          fot_recv_proc(packet,size);
88          return;
89      }
90  #endif // BT_SPP_FOT_EN
91
92      /*Classic Bluetooth - receiving Bluetooth data and sending it to serial port*/
93      printf("spp_rx_callback :%d,%d\n",SPP_CreditCount,size);        /*Test serial port*/
94      hsuart_dma_start(HSUT_TRANSMIT, (uint32_t)packet, size);        /*Send the received data through the tx pin of the serial port*/
95      spp_set_rx_new_credit(1);                                       /*Close flow control*/
96
97      if(size == 2)                                                   /*Judge the print command*/
98      {
99          if( packet[0] == 0x1d && packet[1] == 0x34)
100             print_pic = 1;
101     }
102
103 }
```

main.c | hsuart_transfer.c | app.c | spp.c

```
160     //          {
161     //              buf[0] = 0x11;
162     //              ble_tx_notify(gatts_Datas_Characteristic_base.att_index, buf, 1);
163     //              BtFlowControl_start = 0;
164     ////               printf(" BTFlowsControl_Start \n");
165     //          }
166     //      }
167     }
168
169 /*Low-power Bluetooth*/
170 extern volatile u8 g_ucAT_flag;
171 extern uint8_t  print_pic;                                          /*  Judge the print c
172 static uint8_t gatt_callback_app(u8 *ptr, u16 len)
173 {
174     g_ucAT_flag = 0;
175     hsuart_dma_start(HSUT_TRANSMIT, (uint32_t)ptr, len);             /*Send the received
176     printf("[%x]%x %x...%x\n",len,ptr[0],ptr[1],ptr[len-1]);         /*Test serial port*
177
178     if(len == 2)                                                     /*Judge the print
179     {
180         if( ptr[0] == 0x1d && ptr[1] == 0x34)
181             print_pic = 1;
182     }
183
184     return false;
185 }
186 /**********/
```


6、这样我们只需要在主函数中初始化串口和循环等待串口接收函数即可实现蓝牙的透传功能。波特率

设置为 115200，也可根据需要进行更改。

```
//Start Main function
int main(void)
{
    bsp_sys_init();                 /*System initialization*/

/*****Printer and Bluetooth initialization******/

    gpio_led_init();                /*LED2 initialization*/
    my_hsuart_init();               /*Serial port initialization*/
    printer_init();                 /*printer initialization*/

/****Other routines Related initialization******/
//    gpio_input_init();
//    Charge_init();
//    my_dac_init(DAC_L | DAC_R);
//    IIC_AT24C01_init();
//    power_sleep_wake_init();
//    rtc_calendar_init();
//    timer3_cap_init(66666);
//    timer3_init(500000 - 1);          // 500ms, f=1MHz
//    timer3_pwm_init(1000 - 1);        // Fsrc=1MHz, Fpwm=1KHz
//    uart2_init(115200);
//    Adc_key_init();
//    Adc_sampling_init();
//    sd_disk_init();
//    spi1_init(SPIFALSH_BAUD);
/*****************************************/
    while (1)
    {
        LED_Bluetooth();            /*LED2 is always on after Bluetooth connection*/
        hsuart_transfer_example();  /*Wait for reception interruption - receive Bluetooth data*/
        printer_example();          /*Print fixed font "Geehy"*/

/****Other routines Execute functionn******/
//    gpio_input_key_example();
//    Charge_example();
//    my_dac_out_example();
//    gpio_out_led_example();
//    IIC_AT24C01_example();
//    power_sleep_wake_example();
```

main.c | hsuart_transfer.c

```
    clock_gate0_cmd(CLOCK_GATE0_HSUART0, ENABLE);

    gpio_init_structure.gpio_pin = GPIO_PIN_1;
    gpio_init_structure.gpio_dir = GPIO_DIR_INPUT;
    gpio_init_structure.gpio_fen = GPIO_FEN_PER;
    gpio_init_structure.gpio_mode = GPIO_MODE_DIGITAL;
    gpio_init_structure.gpio_pupd = GPIO_PUPD_UP;
    gpio_init(GPIOB_REG, &gpio_init_structure);

    gpio_init_structure.gpio_pin = GPIO_PIN_2;
    gpio_init_structure.gpio_dir = GPIO_DIR_OUTPUT;
    gpio_init_structure.gpio_drv = GPIO_DRV_8MA;
    gpio_init(GPIOB_REG, &gpio_init_structure);

    gpio_func_mapping(HSUTTXMAP_PB2);
    gpio_func_mapping(HSUTRXMAP_PB1);

    hsuart_init_struct.baud = 115200;
//    hsuart_init_struct.baud = 460800;          /*Baud rate selection*/
//    hsuart_init_struct.baud = 230400;
//    hsuart_init_struct.baud = 256000;

    hsuart_init_struct.tx_mode = HSUT_TX_DMA_MODE;
    hsuart_init_struct.rx_mode = HSUT_RX_DMA_MODE;
    hsuart_init_struct.tx_stop_bit = HSUT_STOP_BIT_1BIT;
    hsuart_init_struct.tx_word_len = HSUT_TX_LENGTH_8b;
    hsuart_init_struct.rx_word_len = HSUT_RX_LENGTH_8b;
```

```
    /**********/

/*Receiver function*/
    if ((RECEIVE_STA & 0x80) != RESET)
    {
        len = RECEIVE_STA & 0x7f;
        p = RECEIVE_BUF;
        printf("receive success[%d]: ", len);       /*Test serial port PB3*/
        printf(RECEIVE_BUF, len);

        /*Send data to Bluetooth*/
        bt_spp_tx((uint32_t)RECEIVE_BUF, len);
        ble_send_packet((uint32_t)RECEIVE_BUF, len);

        RECEIVE_STA = 0;
        hsuart_dma_start(HSUT_RECEIVE, (uint32_t)RECEIVE_BUF, 100);
    }
}
```

7、打印机输出固定字符程序中我们初始化打印机的各个引脚，并在打印机例程中等待标志位，如按键按下与蓝牙接收到固定数据，如相关标志位置 1，我们让打印机打印“Geehy”字样。

```
/*****Printer and Bluetooth initialization*******/

    gpio_led_init();                    /*LED2 initialization*/
    my_hsuart_init();                   /*Serial port initialization*/
    printer_init();                     /*printer initialization*/

/****Other routines Related initialization******/
//    gpio_input_init();
//    Charge_init();
//    my_dac_init(DAC_L | DAC_R);
//    IIC_AT24C01_init();
//    power_sleep_wake_init();
//    rtc_calendar_init();
//    timer3_cap_init(66666);
//    timer3_init(500000 - 1);          // 500ms, f=1MHz
//    timer3_pwm_init(1000 - 1);        // Fsrc=1MHz, Fpwm=1KHz
//    uart2_init(115200);
//    Adc_key_init();
//    Adc_sampling_init();
//    sd_disk_init();
//    spi1_init(SPIFALSH_BAUD);
/*******************************************/
    while (1)
    {
        LED_Bluetooth();                /*LED2 is always on after Bluetooth connection*/
        hsuart_transfer_example();      /*Wait for reception interruption - receive Bluetoot
        printer_example();              /*Print fixed font "Geehy"*/
```

```
153
154     void printer_init(void)
155     {
156         gpio_init_typedef gpio_init_structure;
157         spi_init_typedef spi_init_structure;
158         uint8_t ii;
159
160         gpio_init_structure.gpio_pin = GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3;
161         gpio_init_structure.gpio_dir = GPIO_DIR_OUTPUT;
162         gpio_init_structure.gpio_fen = GPIO_FEN_GPIO;
163         gpio_init_structure.gpio_mode = GPIO_MODE_DIGITAL;
164         gpio_init_structure.gpio_drv = GPIO_DRV_8MA;
165
166         gpio_init(GPIOF_REG, &gpio_init_structure);
167
168         gpio_init_structure.gpio_pin = GPIO_PIN_5;
169         gpio_init_structure.gpio_dir = GPIO_DIR_INPUT;
170         gpio_init_structure.gpio_fen = GPIO_FEN_GPIO;
171         gpio_init_structure.gpio_mode = GPIO_MODE_DIGITAL;
172         gpio_init_structure.gpio_pupd = GPIO_PUPD_UP;
173
174         gpio_init(GPIOB_REG, &gpio_init_structure);
175
176         gpio_init_structure.gpio_pin = GPIO_PIN_0 | GPIO_PIN_4 | GPIO_PIN_5;
177         gpio_init_structure.gpio_dir = GPIO_DIR_OUTPUT;
178         gpio_init_structure.gpio_fen = GPIO_FEN_GPIO;
179         gpio_init_structure.gpio_mode = GPIO_MODE_DIGITAL;
180         gpio_init_structure.gpio_drv = GPIO_DRV_8MA;
181
182         gpio_init(GPIOE_REG, &gpio_init_structure);
183
184         clock_gate1_cmd(CLOCK_GATE1_SPI1, ENABLE);
185
186         gpio_init_structure.gpio_pin = GPIO_PIN_3 | GPIO_PIN_4;          //CLK
187         gpio_init_structure.gpio_dir = GPIO_DIR_OUTPUT;
188         gpio_init_structure.gpio_fen = GPIO_FEN_PER;
189         gpio_init_structure.gpio_mode = GPIO_MODE_DIGITAL;
190         gpio_init_structure.gpio_pupd = GPIO_PUPD_UP;
191         gpio_init(GPIOA_REG, &gpio_init_structure);
192         gpio_func_mapping(SPI1_MAP_G1);
```

main.c printer.c

```
255         Run_OK = 0;
256         tmr_cmd(TMR3, ENABLE);
257         while(Run_OK == 0){
258     //          printf( "Run_OK %d\n",Run_OK);
259         }
260
261     }
262
263     /*Judge the key and Bluetooth reception flag bit, and print fixed font*/
264     uint32_t picpt_offset;
265     uint8_t  print_pic = 0;
266     void printer_example(void)
267     {
268         if (gpio_read_bit(GPIOB_REG, GPIO_PIN_5) == RESET && print_pic != 1) {
269             print_pic = 1;
270         }
271
272         if(print_pic)
273         {
274             step = 0;
275             picpt_offset = 0;
276             while(step < 960)
277             {
278             PrinterLatch_RESET();
279             printhead_senddatas(pic_pt+picpt_offset,48);
```

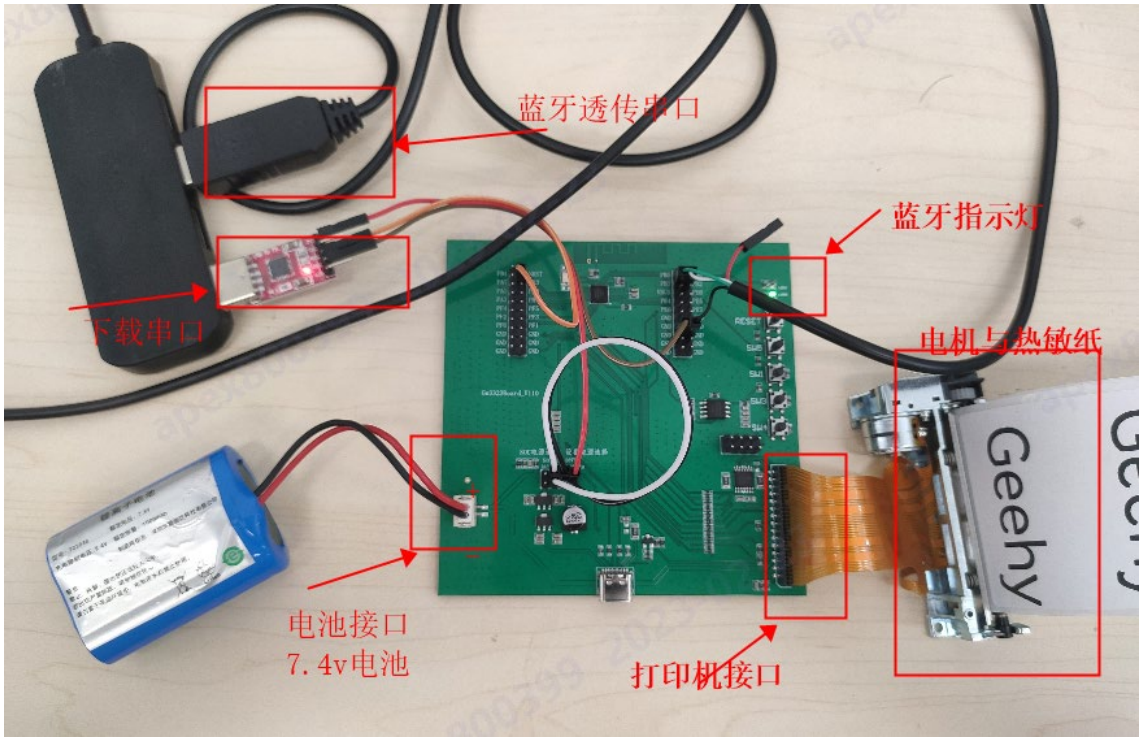
8、程序中还有蓝牙指示灯函数，通过判断蓝牙是否连接来展示不同的状态，当蓝牙尚未连接时，LED2 指示灯以 500ms 的频率闪烁，当蓝牙连接到开发板后，LED2 常亮提示连接成功。

```
31
32  /*****Printer and Bluetooth initialization*******/
33
34      gpio_led_init();                    /*LED2 initialization*/
35      my_hsuart_init();                   /*Serial port initialization*/
36      printer_init();                     /*printer initialization*/
37
38  /****Other routines Related initialization******/
39  //    gpio_input_init();
40  //    Charge_init();
41  //    my_dac_init(DAC_L | DAC_R);
42  //    IIC_AT24C01_init();
43  //    power_sleep_wake_init();
44  //    rtc_calendar_init();
45  //    timer3_cap_init(66666);
46  //    timer3_init(500000 - 1);          // 500ms, f=1MHz
47  //    timer3_pwm_init(1000 - 1);        // Fsrc=1MHz, Fpwm=1KHz
48  //    uart2_init(115200);
49  //    Adc_key_init();
50  //    Adc_sampling_init();
51  //    sd_disk_init();
52  //    spi1_init(SPIFALSH_BAUD);
53  /*****************************************/
54      while (1)
55      {
56          LED_Bluetooth();                /*LED2 is always on after Bluetooth connectior
57          hsuart_transfer_example();      /*Wait for reception interruption - receive Bl
58          printer_example();              /*Print fixed font "Geehy"*/
59
```

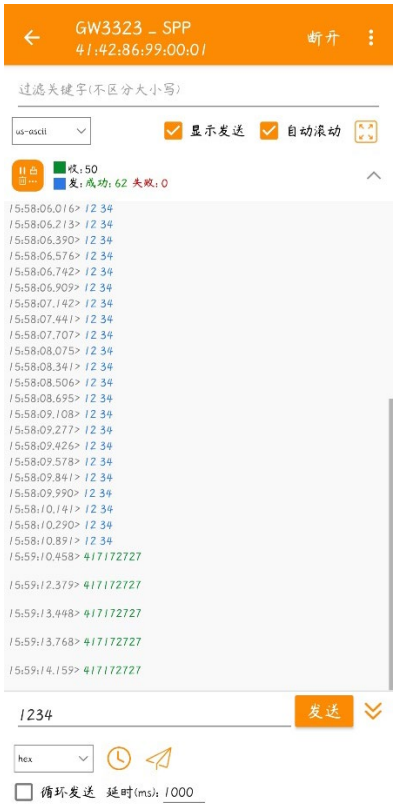
main.c gpio_out_led.c

```c
#include "gw3323_gpio.h"
#include "gpio_out_led.h"

void gpio_led_init(void)    /*Pb0 initialization*/
{
    gpio_init_typedef gpio_init_structure;

    gpio_init_structure.gpio_pin = GPIO_PIN_0;
    gpio_init_structure.gpio_dir = GPIO_DIR_OUTPUT;
    gpio_init_structure.gpio_fen = GPIO_FEN_GPIO;
    gpio_init_structure.gpio_mode = GPIO_MODE_DIGITAL;
    gpio_init_structure.gpio_drv = GPIO_DRV_8MA;

    gpio_init(GPIOB_REG, &gpio_init_structure);
}

/*LED flashing*/
void LED_Twinkle()
{
    gpio_set_bits(GPIOB_REG,GPIO_PIN_0);
    delay_ms(500);
    gpio_reset_bits(GPIOB_REG,GPIO_PIN_0);
    delay_ms(500);
}
        void delay_ms(uint n)

/*optional feature*/
void LED_Bluetooth()
{
    /*Determine whether to connect*/
    if((bt_get_status() == BT_STA_CONNECTED)||(ble_get_status() == LE_STA_CONNECTION))
    {
        gpio_set_bits(GPIOB_REG,GPIO_PIN_0);
    }
    else
    {
        LED_Twinkle();
    }
}
```

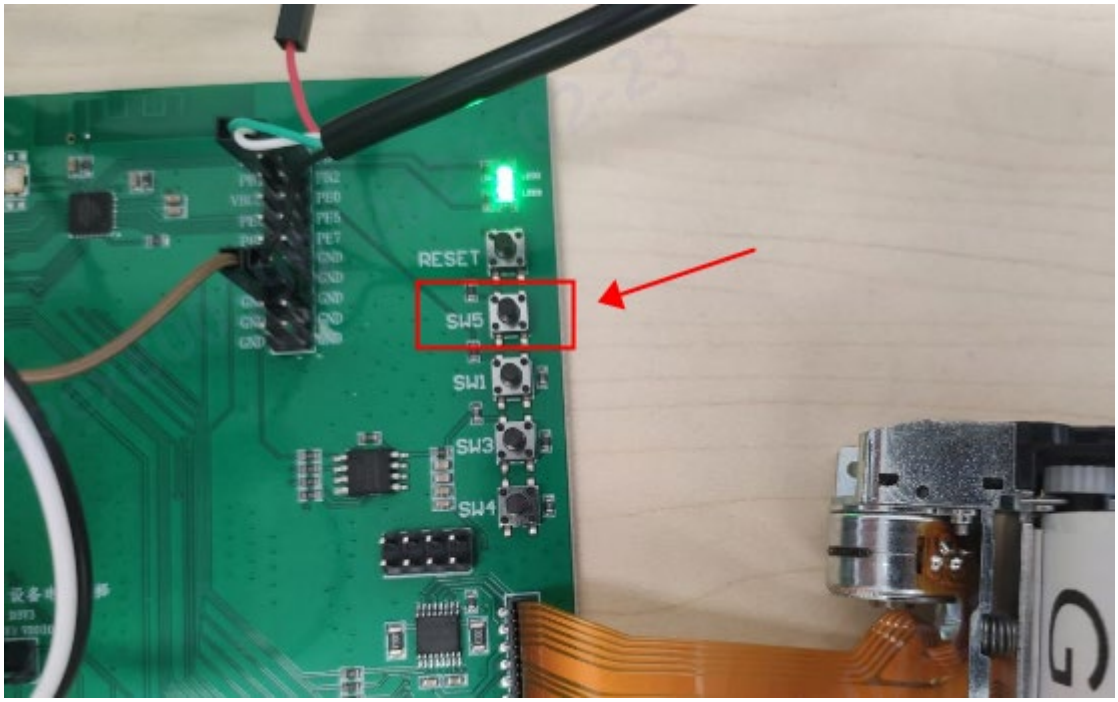
9、整体连接图



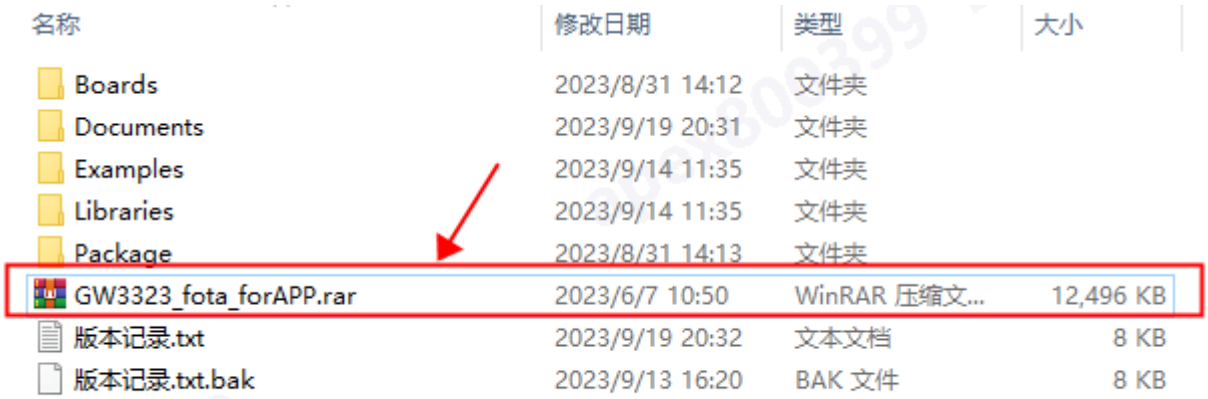
10、 程序下载完成后即可在电脑上打开串口助手（以 SSCOM 软件为例），设置比特率 115200，选择端口号，勾选 HEX 显示。使用手机软件（以 SPP 为例）连接开发板蓝牙，选择 hex 发送格式，输入数据即可在电脑端看到。

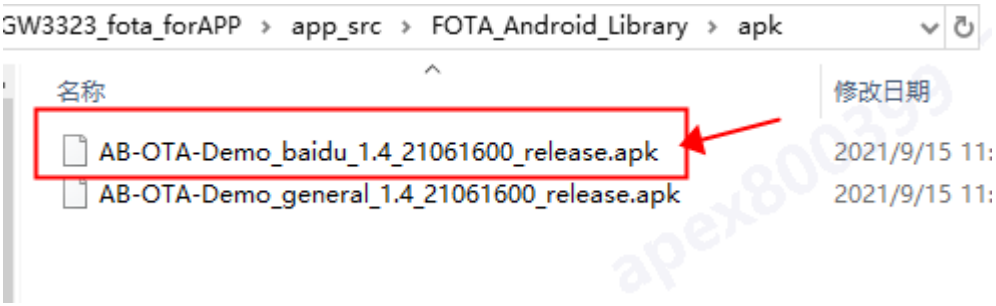
11、 我们也可发送 hex 格式的 1d34 数据，即可看到打印机打印出“Geehy”字符，也可通过 SW5 按键控制打印机输出“Geehy”字符。



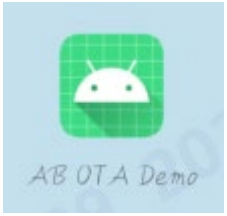

# 5 空中升级



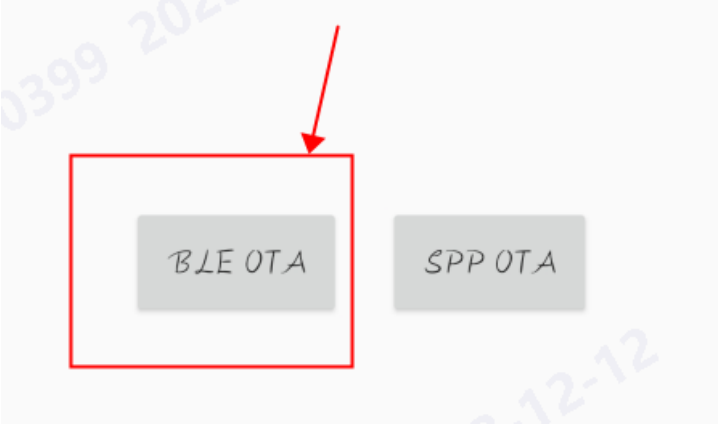

1、下载如图所示升级 APP 安装包到电脑，解压后，进入如下目录 app_src\FOTA_Android_Library\apk




2、将.apk 文件发送到手机，手机下载后安装软件，软件安装好后，如下图所示




3、打开软件后，选择 BLE OTA 或 SPP OTA 即可进行 GW3323 的空中软件升级，以 BLE OTA 为例，具体步骤如下图所示。



选择自己的蓝牙名称


出现版本号，点击选择文件

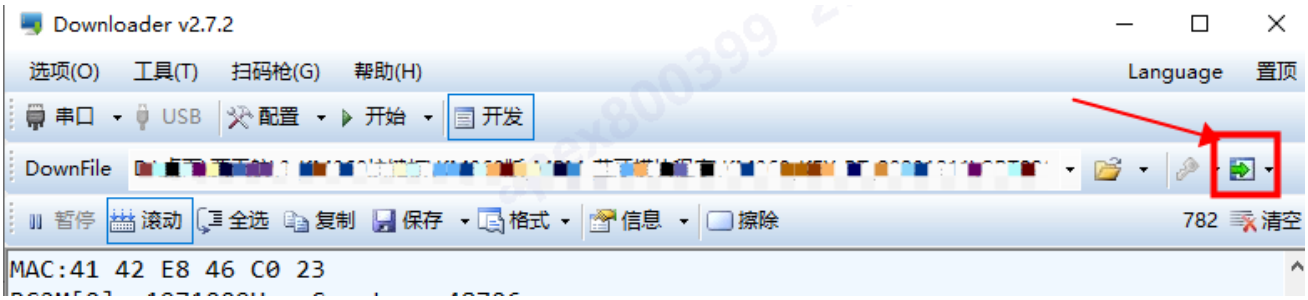

选择自己制作的.fot 文件，点击 SELECT 按键

点击开始升级，等待进度条结束后，重启设备即可

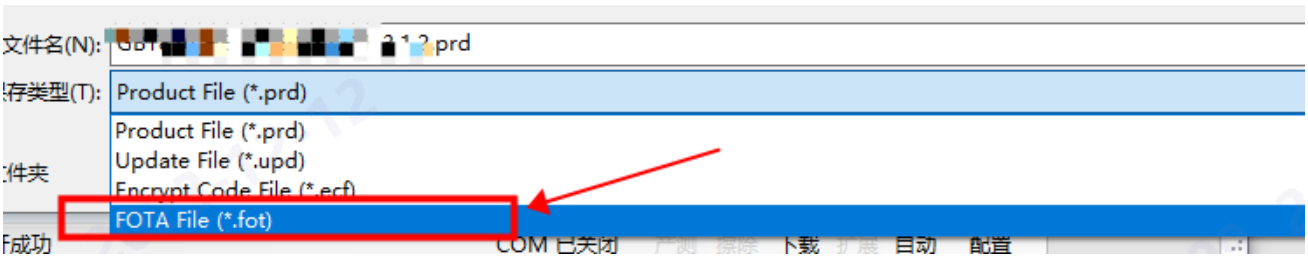


4、.fot 文件的制作

在下载工程界面点击如下图所示的图标

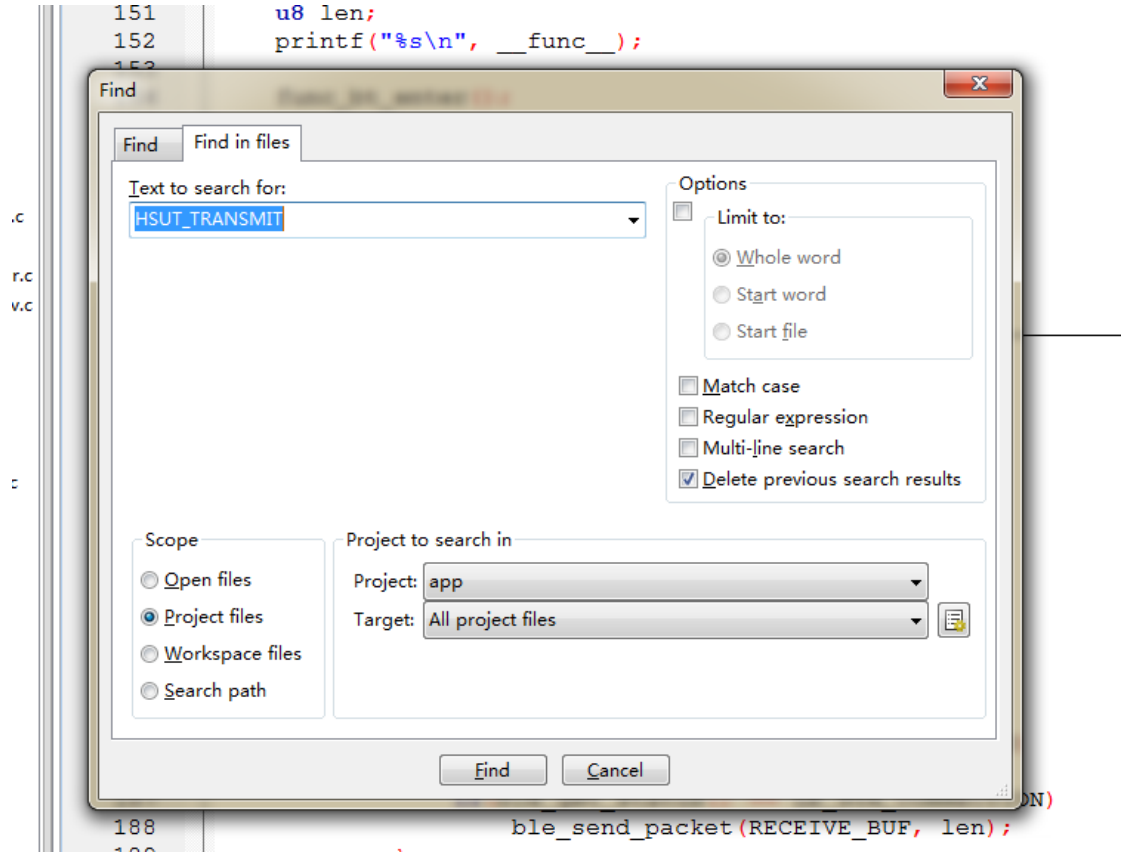



选择文件格式，设置文件名后，保存即可

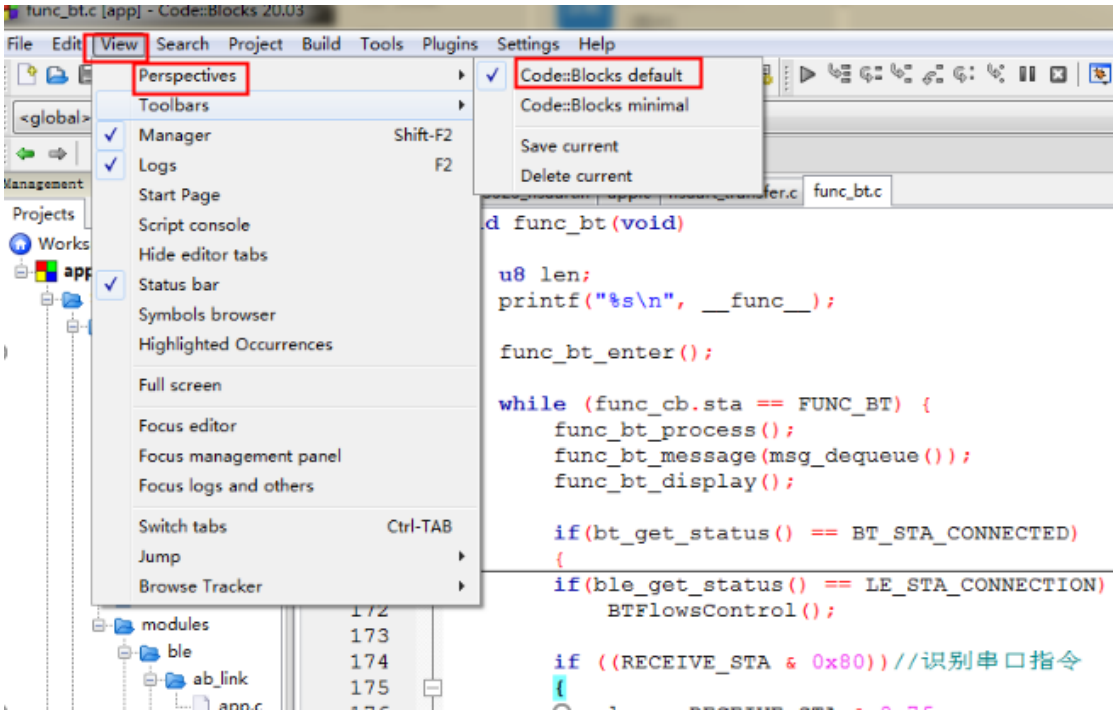


# 6 常用设置

1、查找所有文件(ctrl+F)




5、恢复窗口视图



6、注释掉不用不用函数

Ctrl+Shift+C

```
31
32  /*****Printer and Bluetooth initialization*******/
33
34      gpio_led_init();                    /*LED2 initialization*/
35      my_hsuart_init();                   /*Serial port initialization*/
36  //    printer_init();                    /*printer initialization*/
37
38  /****Other routines Related initialization******/
39  //    gpio_input_init();
40  //    Charge_init();
41  //    my_dac_init(DAC_L | DAC_R);
42  //    IIC_AT24C01_init();
43  //    power_sleep_wake_init();
44  //    rtc_calendar_init();
45  //    timer3_cap_init(66666);
46  //    timer3_init(500000 - 1);          // 500ms, f=1MHz
```

7、打开注释文件

Ctrl+Shift+X

```
22  #include "uart_transfer.h"
23  #include "wdt.h"
24  /*******************************************/
25
26
27  //Start Main function
28  int main(void)
29  {
30      bsp_sys_init();                     /*System initialization*/
31
32  /*****Printer and Bluetooth initialization*******/
33
34      gpio_led_init();                    /*LED2 initialization*/
35      my_hsuart_init();                   /*Serial port initialization*/
36      printer_init();                     /*printer initialization*/
37      void printer_init(void)
38  /****Other routines Related initialization******/
39  //    gpio_input_init();
40  //    Charge_init();
41  //    my_dac_init(DAC_L | DAC_R);
42  //    IIC_AT24C01_init();
```

# 7 版本历史

表格 1 文件版本历史

| 日期 | 版本 | 变更历史 |
| --- | --- | --- |
| 2022.11 | 1.0 | 新建 |
| 2022.12 | 2.0 | 增加“例程展示”和“常用设置” |

# 声明

本手册由珠海极海半导体有限公司（以下简称“极海”）制订并发布，所列内容均受商标、著作权、软件著作权相关法律法规保护，极海保留随时更正、修改本手册的权利。使用极海产品前请仔细阅读本手册，一旦使用产品则表明您（以下称“用户”）已知悉并接受本手册的所有内容。用户必须按照相关法律法规和本手册的要求使用极海产品。

1、权利所有

本手册仅应当被用于与极海所提供的对应型号的芯片产品、软件产品搭配使用，未经极海许可，任何单位或个人均不得以任何理由或方式对本手册的全部或部分内容进行复制、抄录、修改、编辑或传播。

本手册中所列带有“®”或“™”的“极海”或“Geehy”字样或图形均为极海的商标，其他在极海产品上显示的产品或服务名称均为其各自所有者的财产。

2、无知识产权许可

极海拥有本手册所涉及的全部权利、所有权及知识产权。

极海不应因销售、分发极海产品及本手册而被视为将任何知识产权的许可或权利明示或默示地授予用户。

如果本手册中涉及任何第三方的产品、服务或知识产权，不应被视为极海授权用户使用前述第三方产品、服务或知识产权，除非在极海销售订单或销售合同中另有约定。

3、版本更新

用户在下单购买极海产品时可获取相应产品的最新版的手册。

如果本手册中所述的内容与极海产品不一致的，应以极海销售订单或销售合同中的约定为准。

4、信息可靠性

本手册相关数据经极海实验室或合作的第三方测试机构批量测试获得，但本手册相关数据难免会出现校正笔误或因测试环境差异所导致的误差，因此用户应当理解，极海对本手册中可能出现的该等错误无需承担任何责任。本手册相关数据仅用于指导用户作为性能参数参照，不构成极海对任何产品性能方面的保证。

用户应根据自身需求选择合适的极海产品，并对极海产品的应用适用性进行有效验证和测试，以确认极海产品满足用户自身的需求、相应标准、安全或其它可靠性要求；若因用户未充分对极海产品进行有效验证和测试而致使用户损失的，极海不承担任何责任。

5、合规要求

用户在使用本手册及所搭配的极海产品时，应遵守当地所适用的所有法律法规。用户应了解产品可能受

到产品供应商、极海、极海经销商及用户所在地等各国有关出口、再出口或其它法律的限制，用户（代表其本身、子公司及关联企业）应同意并保证遵守所有关于取得极海产品及／或技术与直接产品的出口和再出口适用法律与法规。

6、免责声明

本手册由极海“按原样”（as is）提供，在适用法律所允许的范围内，极海不提供任何形式的明示或暗示担保，包括但不限于对产品适销性和特定用途适用性的担保。

对于用户后续在针对极海产品进行设计、使用的过程中所引起的任何纠纷，极海概不承担责任。

7、责任限制

在任何情况下，除非适用法律要求或书面同意，否则极海和/或以“按原样”形式提供本手册的任何第三方均不承担损害赔偿责任，包括任何一般、特殊因使用或无法使用本手册相关信息而产生的直接、间接或附带损害（包括但不限于数据丢失或数据不准确，或用户或第三方遭受的损失）。

8、适用范围

本手册的信息用以取代本手册所有早期版本所提供的信息。