

# Lab2 开关灯

---

- Lab2 开关灯
  - 实验目的
  - 实验器材
    - 硬件
    - 软件
  - 实验原理
    - USB-UART CP2102
    - 连接电路图
  - 实验步骤
    - 工程配置
    - 连线
    - 驱动安装
    - PuTTY监听端口设置
    - 编写程序代码
      - 使103在串口输出“Hello World”给PC;
      - 按下按钮时103上的TST LED点亮, 放开按钮熄灭;
      - 按下按钮切换TST LED的亮灭;
      - 串口输出TST LED的亮灭状态, 如ON/OFF。
  - 扩展内容
    - 分别用串口的轮询、中断和DMA三种方式操作103的串口的数据收发;
    - 用示波器或逻辑分析仪观察串口输出的信号, 分析其速率(提示: 输出0/1交替的字节);
    - 编写程序, 从PC发送ON/OFF字符串到103来控制TST LED的亮灭。
  - 实验心得

## 实验目的

- 掌握串口的使用: 配置103板上的串口发送调试信息, 在PC上安装驱动和串口软件接收信息;
- 掌握面包板的使用;
- 熟练掌握操纵MCU的GPIO做输入输出的方法: 配置103板上的GPIO端口来获得按钮按下与否的状态。
- 去抖: 写程序处理按钮的抖动。

## 实验器材

### 硬件

- STM32F103核心板1块;
- ST-Link 1个;
- 杜邦线(孔-孔) 4根;
- 杜邦面包线(孔-针) 3根;
- 面包线(针-针) 若干;
- 按钮1个;
- 面包板1块。

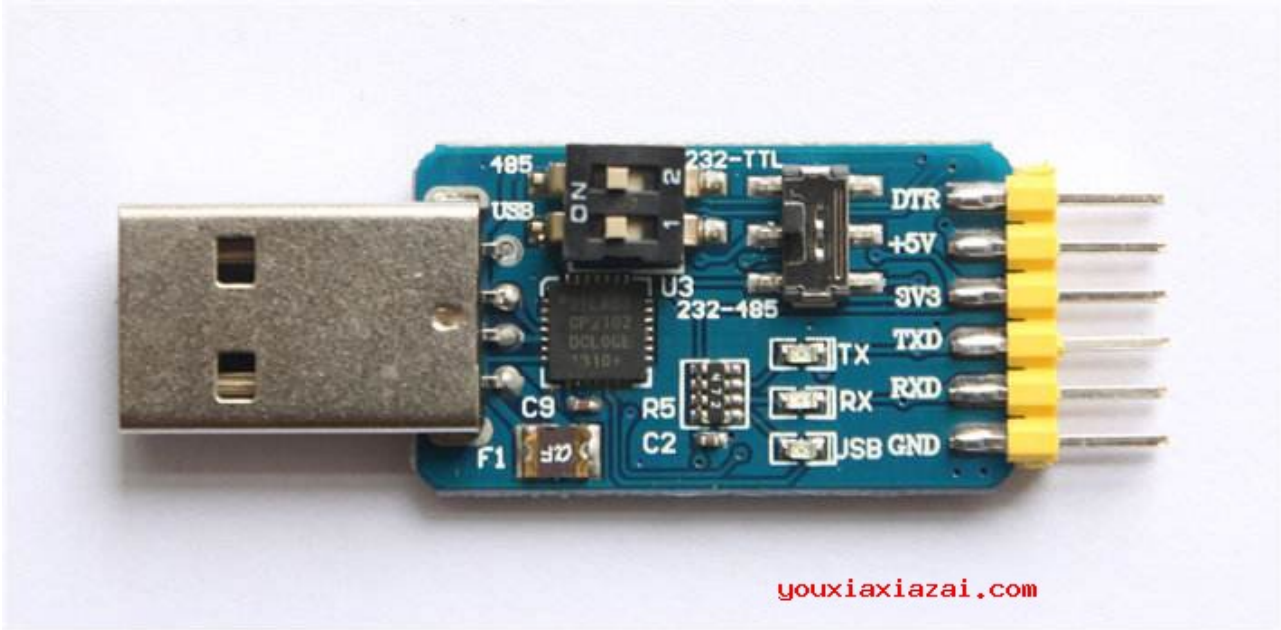
### 软件

- STM32CubeIDE;
- Putty
- Serial Port Utility(友善串口调试助手)

实验原理

USB-UART CP2102

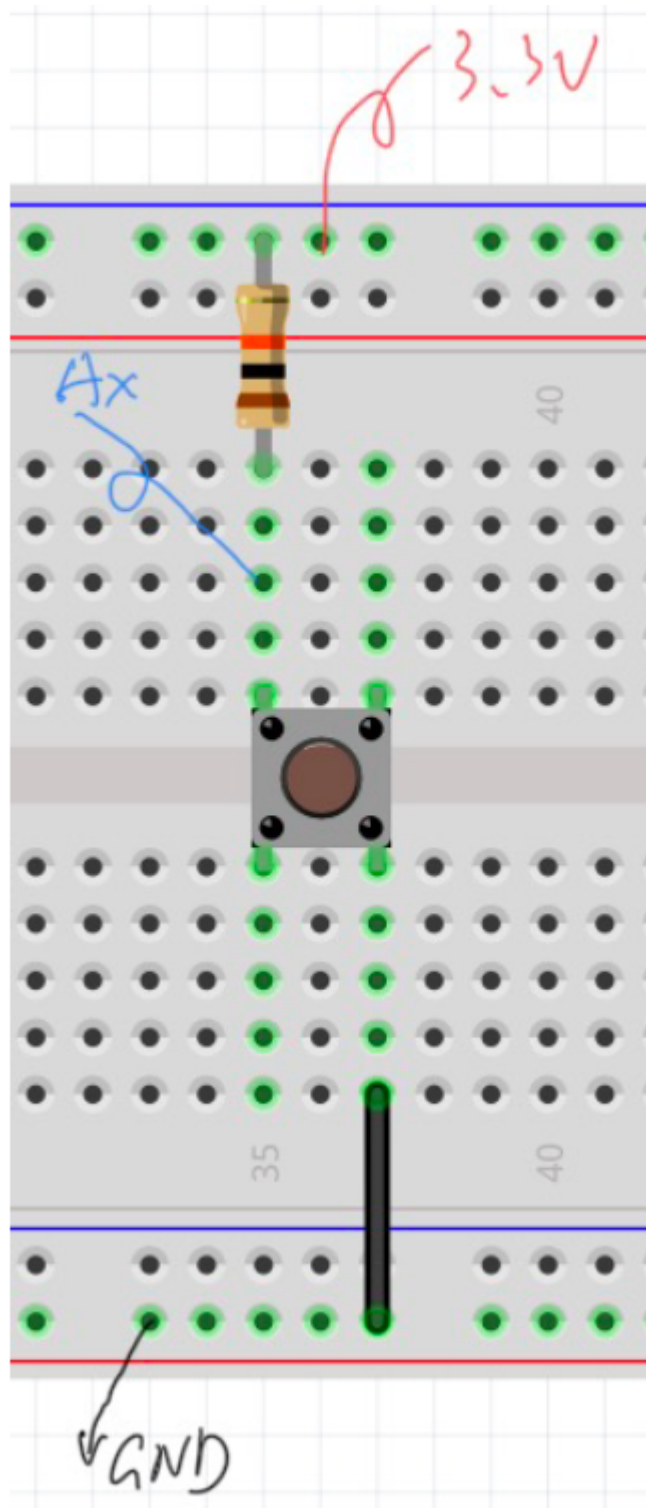
- 主芯片为CP2102，安装驱动后生成虚拟串口;
- USB取电，引出接口包括3.3V (<40mA) , 5V, GND, TX, RX, 信号脚电平为3.3V, 正逻辑;
- 板载状态指示灯、收发指示灯，正确安装驱动后状态指示灯会常亮，收发指示灯在通信的时候会闪烁，波特率越高亮度越低;
- 支持从300bps~1Mbps间的波特率;
- 通信格式支持：
  - 5,6,7,8位数据位;
  - 支持1,1.5,2停止位;
  - odd,even,mark,space,none校验。
- 支持操作系统： windows vista/xp/server 2003/200,Mac OS-X/OS-9,Linux;
- USB头为公头，可直接连接电脑USB口。



连接电路图

- CP2102连接

103	CP2102	颜色	意义
A9	RXD	黄色	103发送数据给PC
A10	TXD	绿色	PC发送数据给103
GND	GND	黑色	接地



- 按钮连接

## 实验步骤

### 工程配置

Configuration

Group By Peripherals

GPIO

RCC

SYS

USART

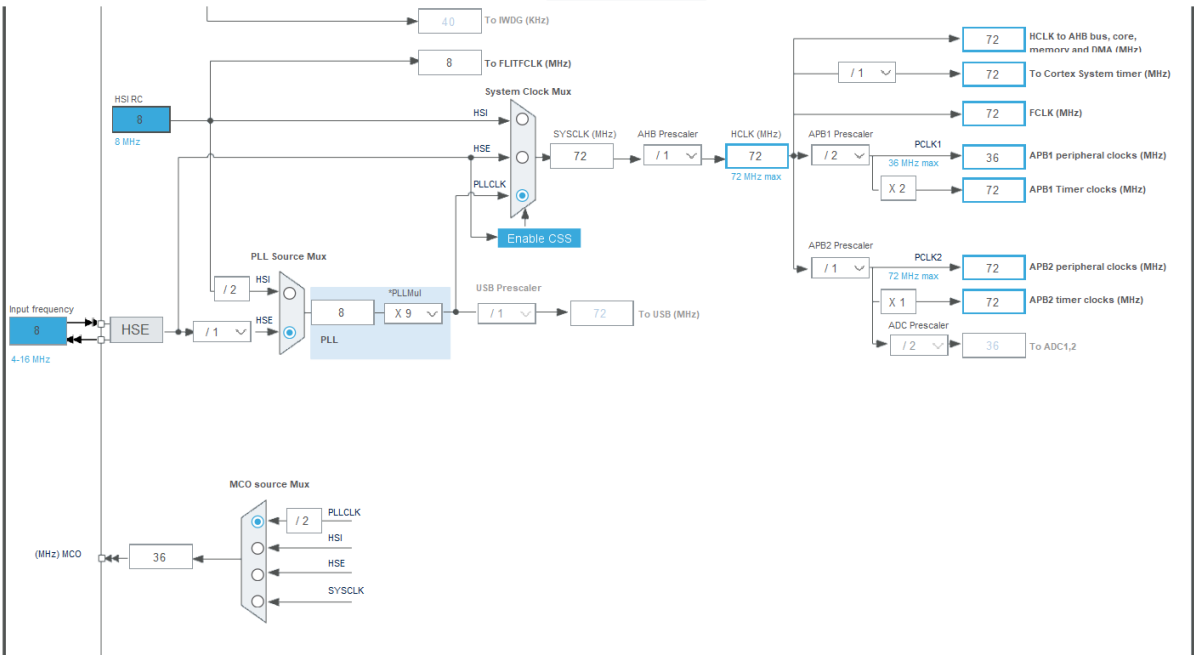
Search Signals

Search (Ctrl+F)

☐ Show only Modified Pins

Pin...	Signal ...	GPIO ...	GPIO ...	GPIO ...	Maxim...	User L...	Modifi...
PC13-...	n/a	Low	Output...	No pull...	Low	LED	<input checked="" type="checkbox"/>

- 配置GPIO
- 配置时钟



Mode

Debug

Serial Wire

☐ System Wake-Up

Timebase Source

SysTick

配置SYS

✓ DMA Settings

✓ GPIO Settings

✓ Parameter Settings

✓ User Constants

✓ NVIC Settings

Search Signals

Search (Ctrl+F)

Pin Name	Signal on Pin	GPIO output I...	GPIO mode	GPIO P
PA9	USART1_TX	n/a	Alternate Fun...	n/a
PA10	USART1_RX	n/a	Input mode	pull-dov

配置UART

✓ DMA Settings

✓ GPIO Settings

✓ Parameter Settings

✓ User Constants

✓ NVIC Settings

Configure the below parameters :

Search (Ctrl+F)

⏪

⏩

ⓘ

Basic Parameters

Baud Rate

115200 Bits/s

Word Length

8 Bits (including Parity)

Parity

None

Stop Bits

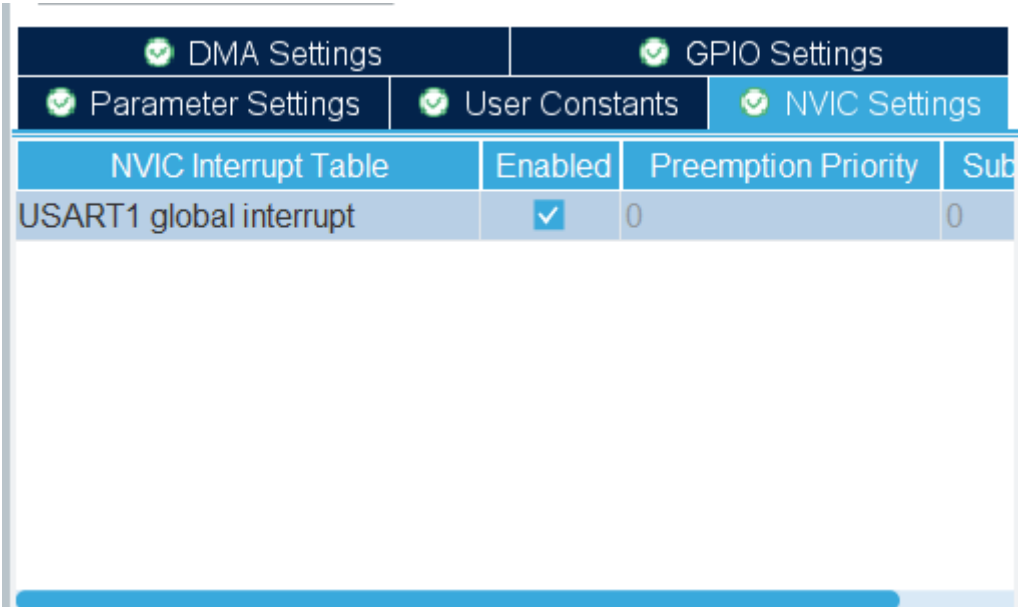
1

Advanced Parameters

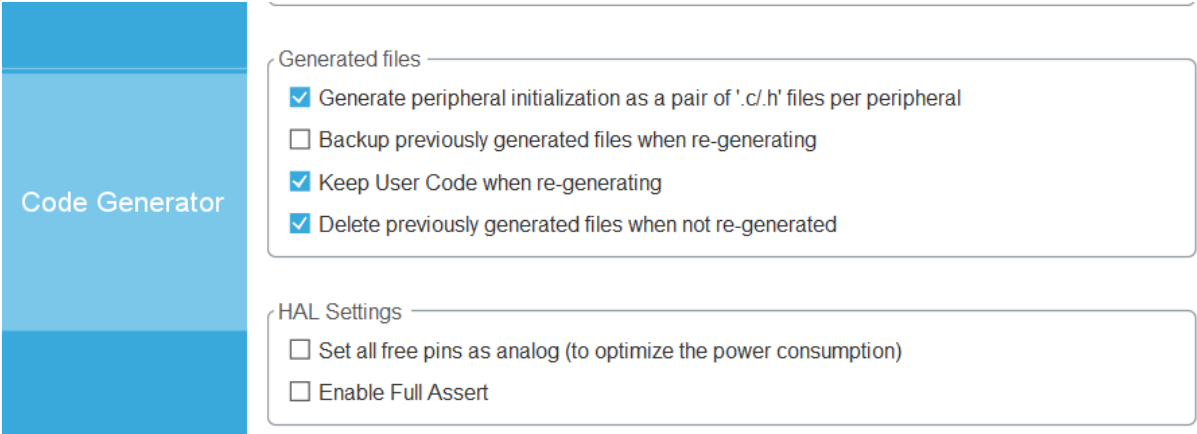
Data Direction

Receive and Transmit

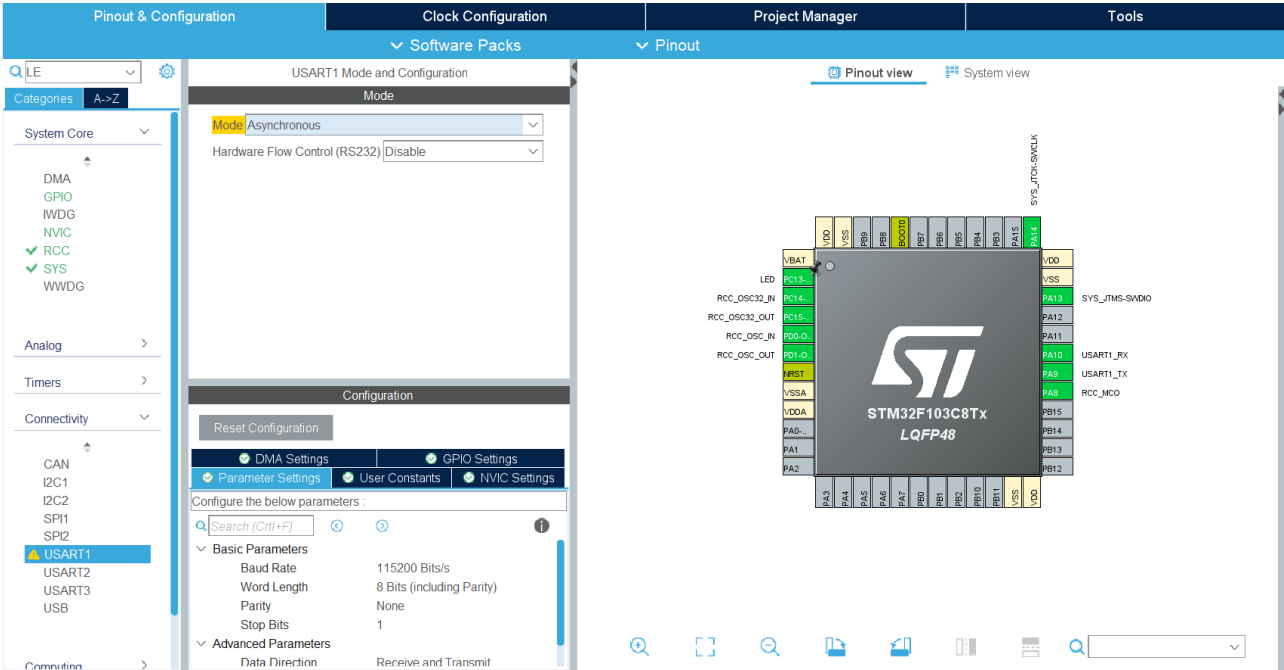
5 / 22



- 配置每个功能生成独立的代码文件



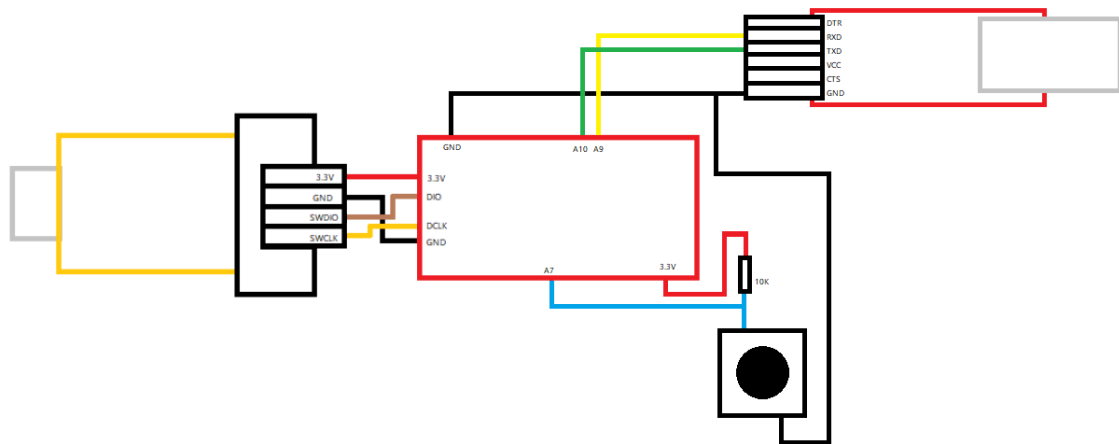
- 配置完成总体效果



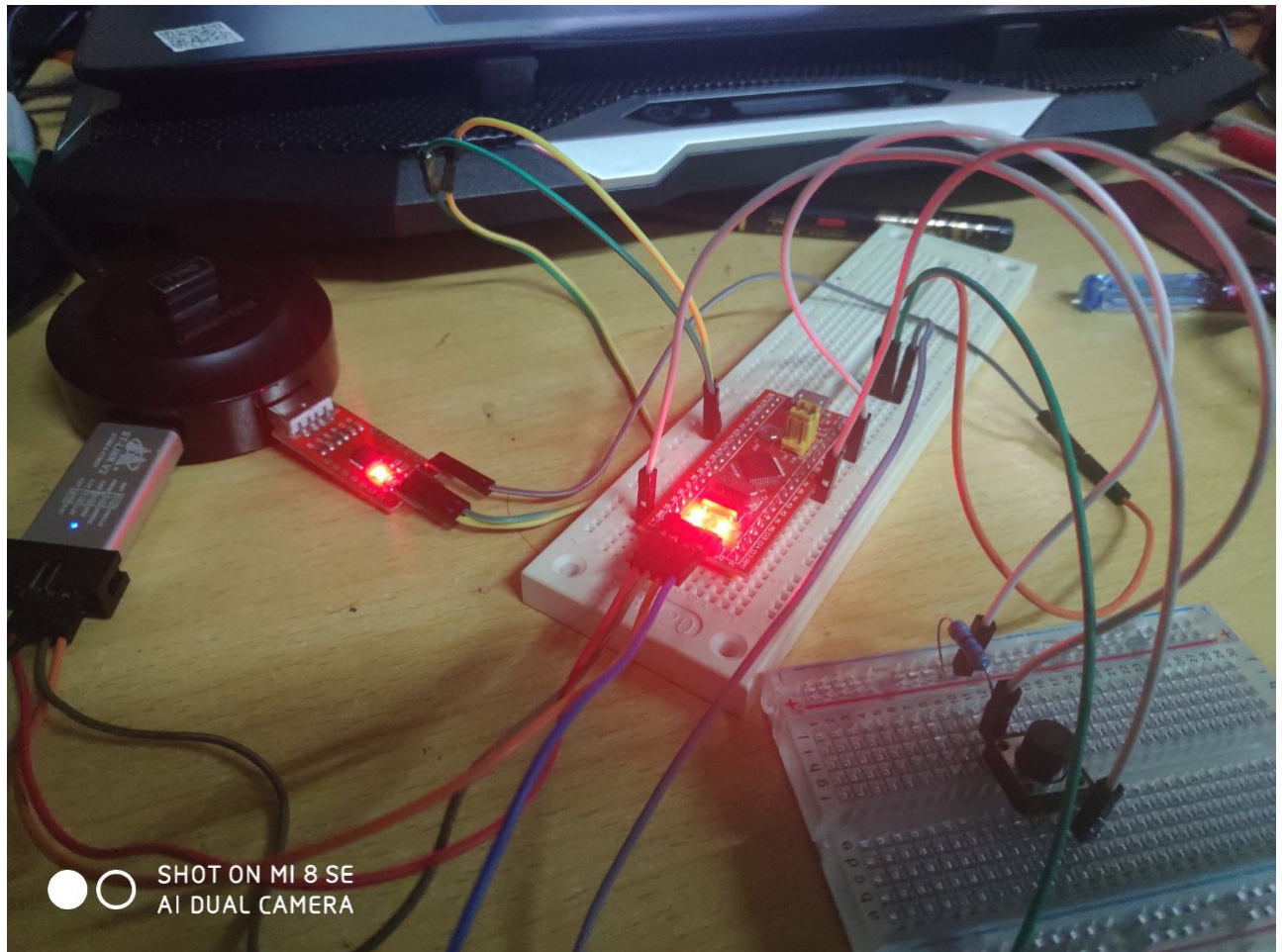
连线



- 连接示意图



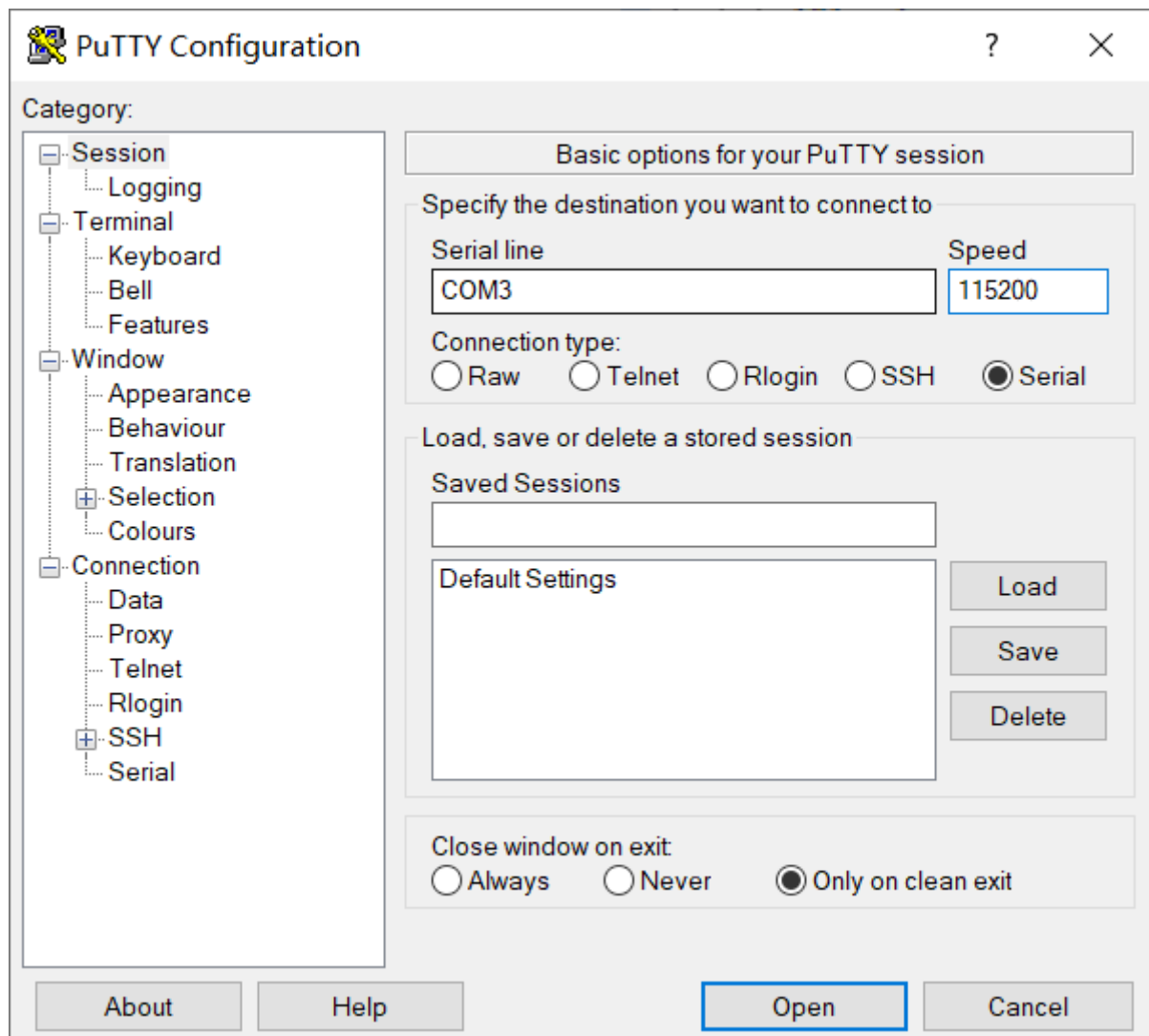
- 连接实物图



## 驱动安装

- 端口 (COM 和 LPT)
  - Silicon Labs CP210x USB to UART Bridge (COM3)

## PuTTY监听端口设置



## 编写程序代码

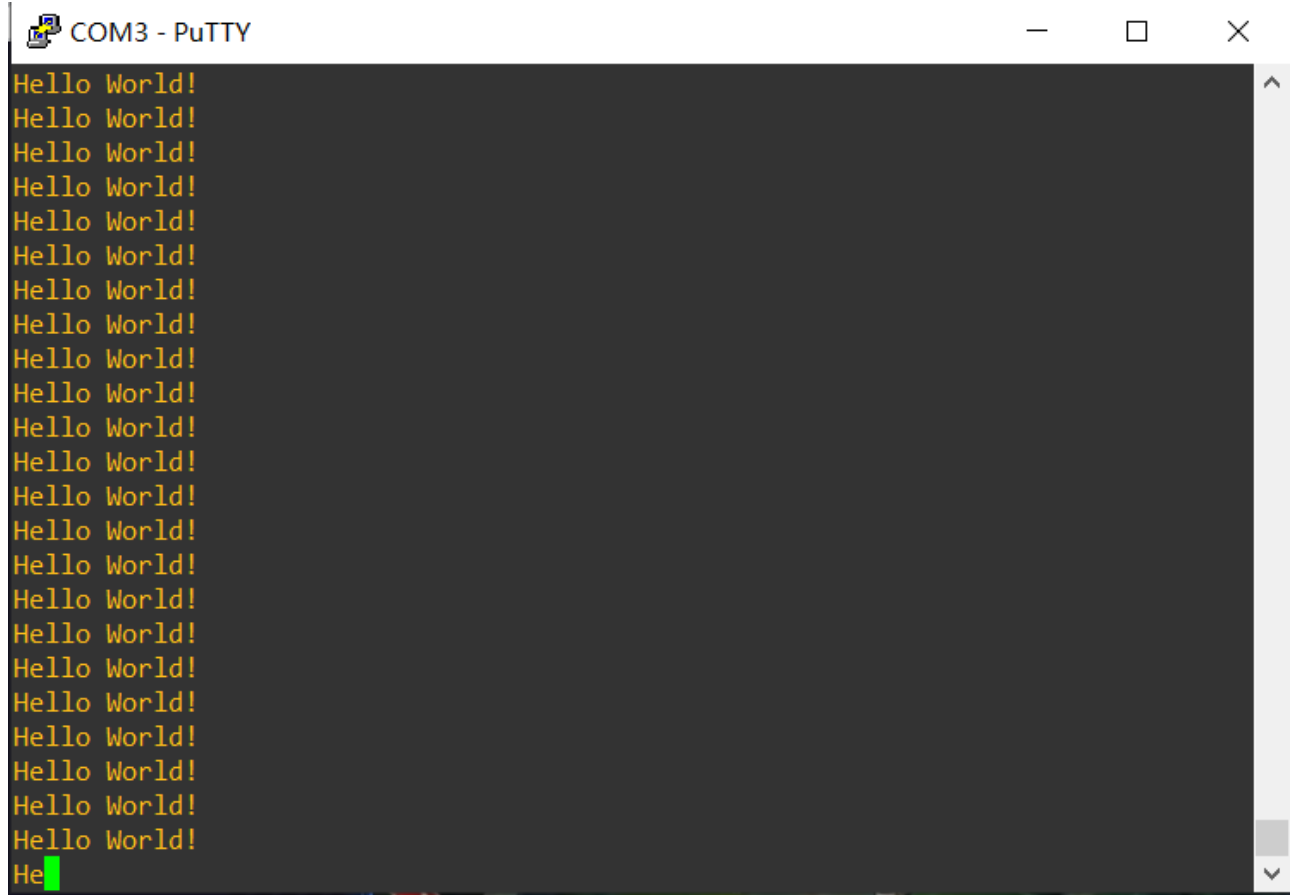
使103在串口输出“Hello World”给PC;

- 代码

```
/* 轮询，在main.c中main函数的while循环内插入 */  
HAL_UART_Transmit(&huart1, (uint8_t *)"Hello World!\r\n", 14, 1000);
```



- 结果



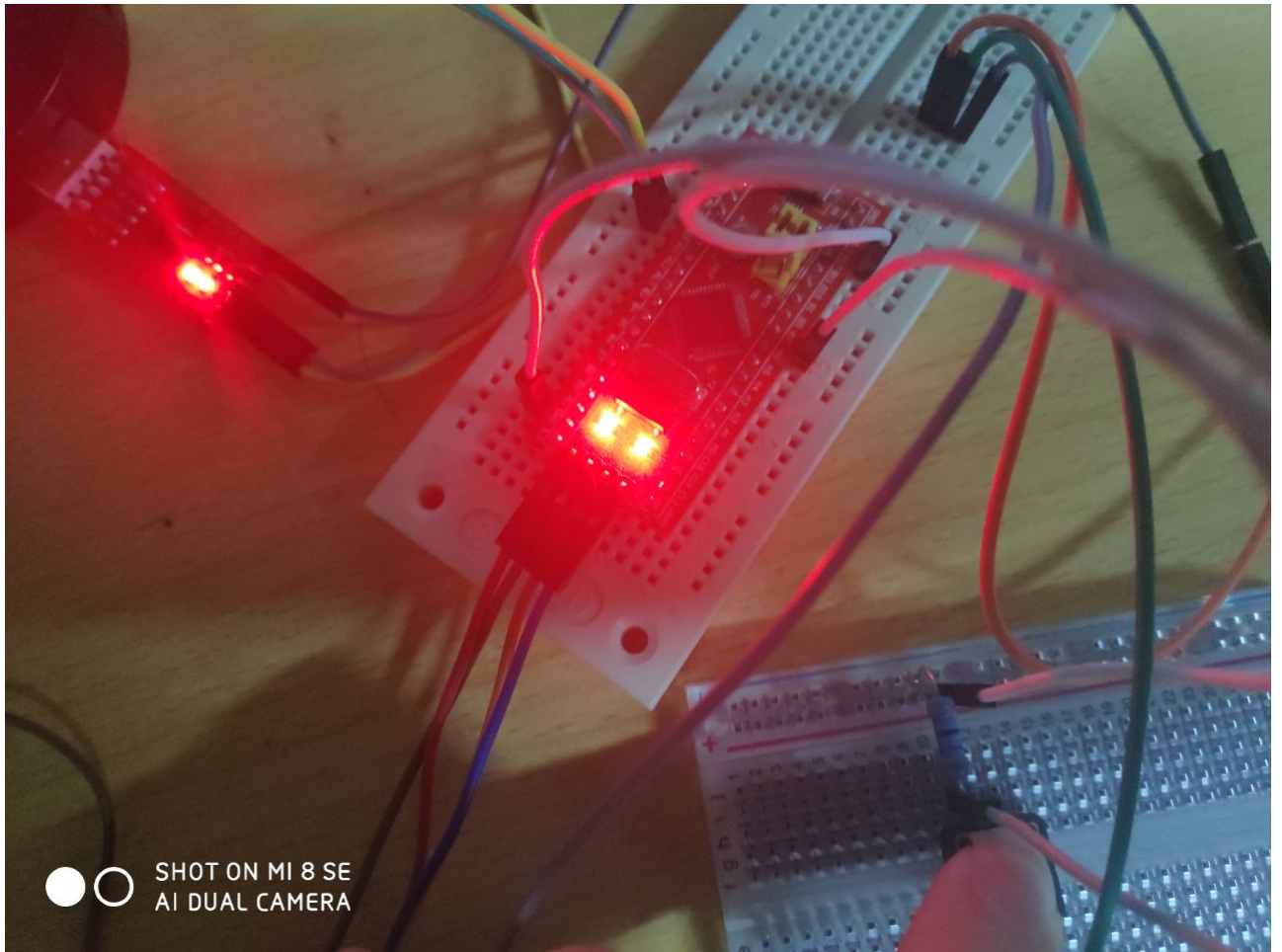
```
COM3 - PuTTY
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
He
```

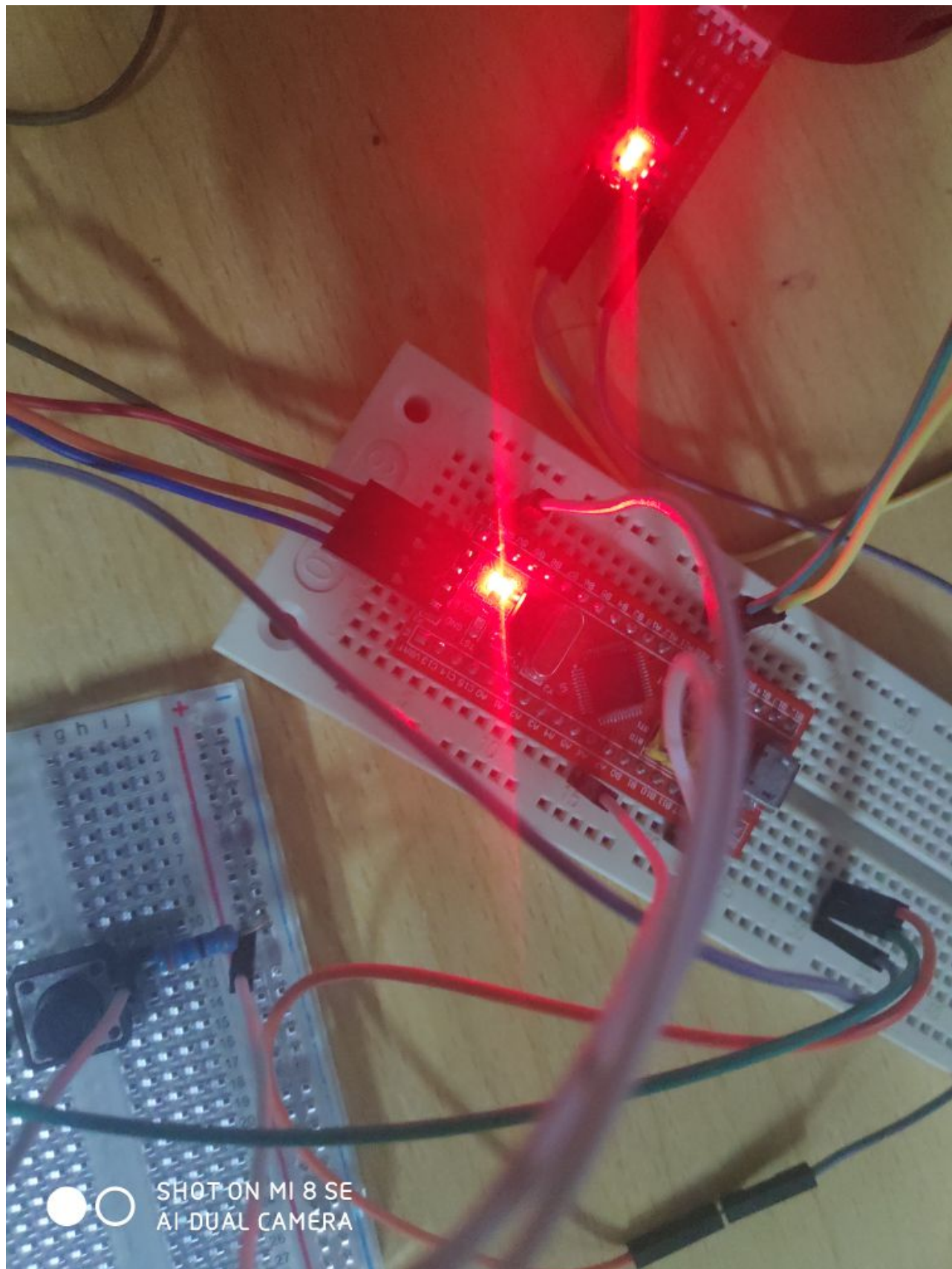
**按下按钮时103上的TST LED点亮，放开按钮熄灭;**

- 代码

```
/* 按下去点亮，松开熄灭 */
if(HAL_GPIO_ReadPin(BTN_GPIO_Port, BTN_Pin) == GPIO_PIN_RESET)
{
    HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_RESET);
}
else
{
    HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET);
}
```

- 结果(实验中的结果可详细见附件视频)





按下按钮切换TST LED的亮灭;

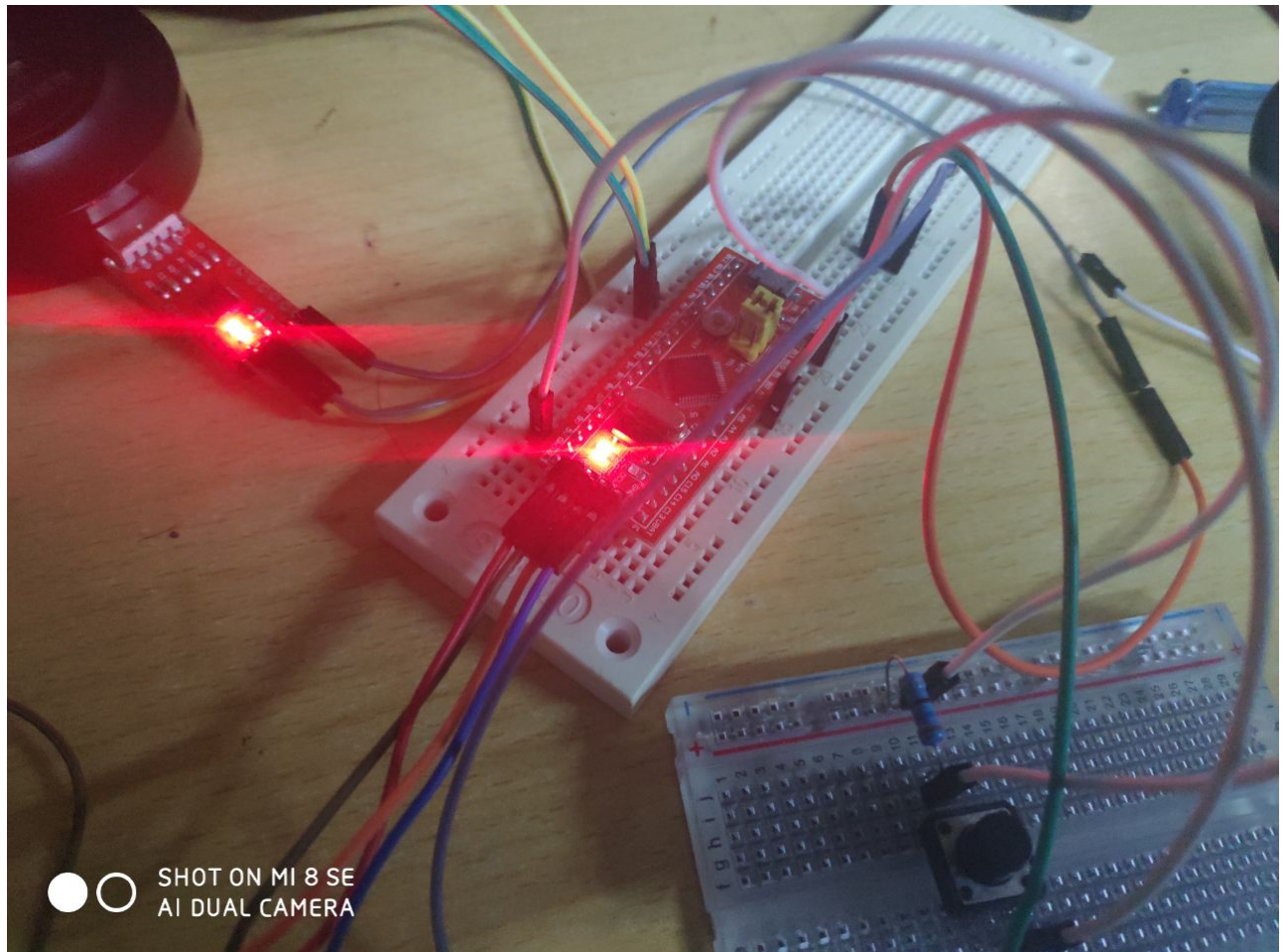
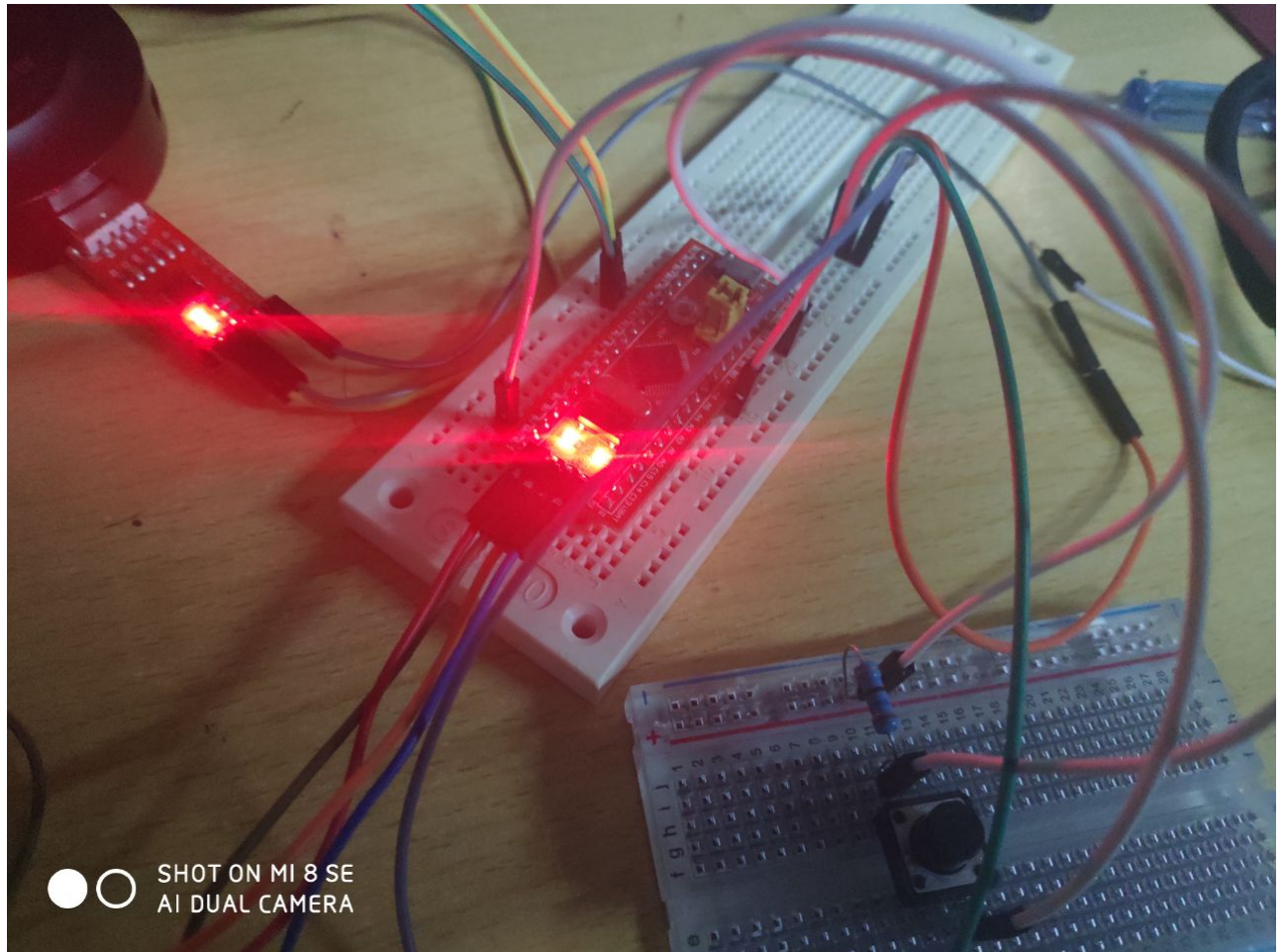
```
/* 按下去点亮，松开熄灭，main函数中 */  
if(BtnState == 1)  
{  
    HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);  
}
```

```
    HAL_Delay(1);
}
/* stm32f1xx_it.c中的SysTick_Handler()函数中加入自定义函数Key_Scan进行去抖动操作 */
void SysTick_Handler(void)
{
    /* USER CODE BEGIN SysTick_IRQn 0 */
    Key_Scan();
    /* USER CODE END SysTick_IRQn 0 */
    HAL_IncTick();

    HAL_SYSTICK_IRQHandler();
    /* USER CODE BEGIN SysTick_IRQn 1 */
    /* USER CODE END SysTick_IRQn 1 */
}
uint8_t btnCount;
uint8_t pushFlag;
extern uint8_t BtnState;
void Key_Scan(void)
{
    /*检测BTN是否按下 */
    if (HAL_GPIO_ReadPin(BTN_GPIO_Port, BTN_Pin) == GPIO_PIN_RESET)
    {
        btnCount++;          //BTN键按下，计数btnCount加1
        if (btnCount >= 64) //1MS中断一次，btnCount大于等于32，即按键已按下32ms
        {
            if (pushFlag == 0) //判断有没有重按键，1为有，0为没有
            {
                BtnState = 1; //设置按键标志
                btnCount = 0;
                pushFlag = 1; //设置重按键标志
            }
            else
                btnCount = 0;
        }
        else
            BtnState = 0;
    }
    else //无按键按下
    {
        btnCount = 0; //清零btnCount
        BtnState = 0; //清除按键标志
        pushFlag = 0; //清除重按键标志
    }
}
```



- 结果(实验中的结果可详细见附件视频)

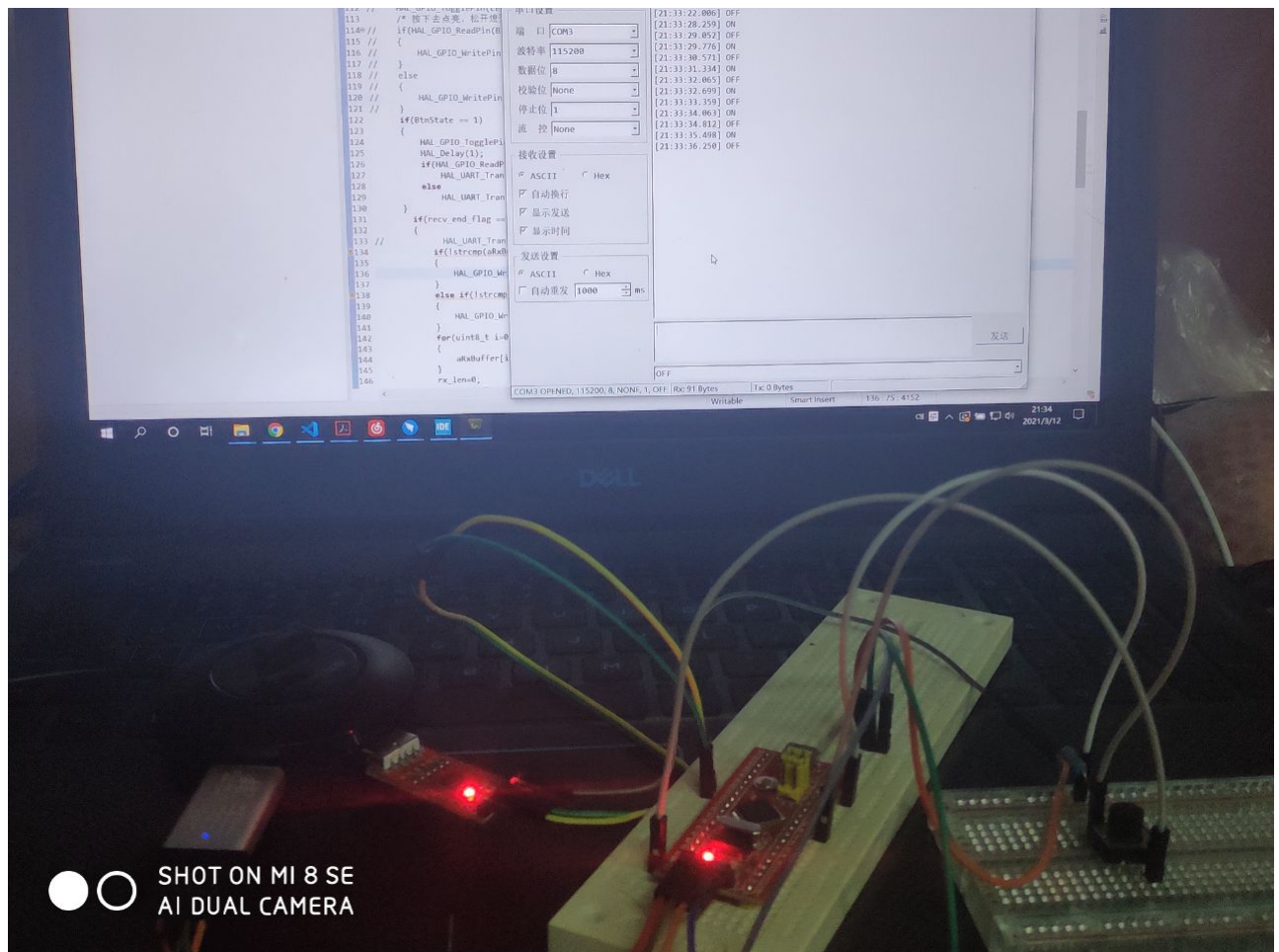


串口输出TST LED的亮灭状态，如ON/OFF。

- 代码

```
/* 在上面main函数的按钮功能部分加上串口通信功能 */  
if(BtnState == 1)  
{  
    HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);  
    if(HAL_GPIO_ReadPin(LED_GPIO_Port, LED_Pin) == GPIO_PIN_RESET)  
        HAL_UART_Transmit(&huart1, (uint8_t *)"ON\r\n", 7, 1000);  
    else  
        HAL_UART_Transmit(&huart1, (uint8_t *)"OFF\r\n", 7, 1000);  
    HAL_Delay(1);  
}
```

- 结果



## 扩展内容

分别用串口的轮询、中断和DMA三种方式操作103的串口的数据收发;

- 轮询方式见前文发送"Hello World!";
- 中断
  - 代码



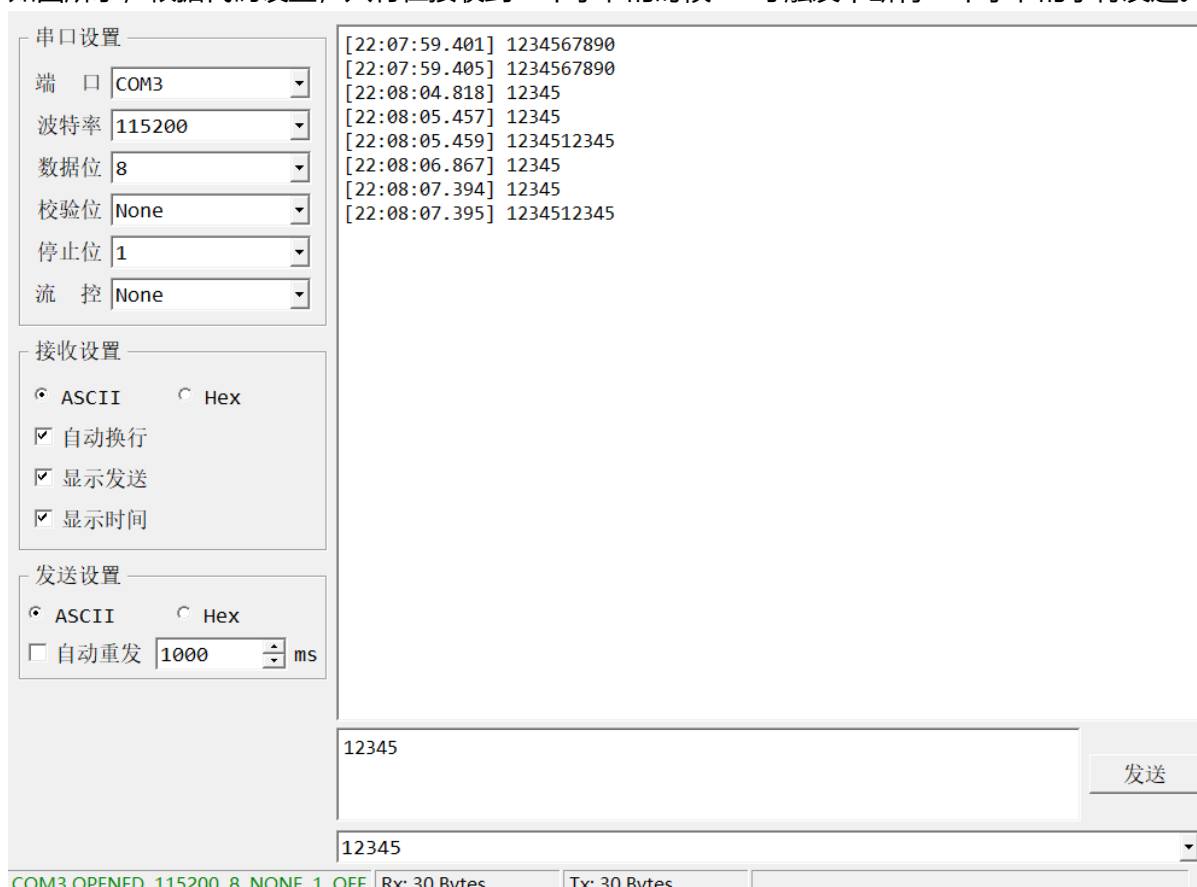
```

/* main.c定义全局变量 */
uint8_t aRxBuffer[20];
/* main.c中的main函数中while循环外插入 */
HAL_UART_Receive_IT(&huart1,aRxBuffer,10);
/* main.c中的void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)插入 */
HAL_UART_Transmit(&huart1,aRxBuffer,10,1000);
/* stm32f1xx_it.c中定义外部变量 */
extern uint8_t aRxBuffer[20];
/* stm32f1xx_it.c中的void USART1_IRQHandler(void)插入 */
HAL_UART_Receive_IT(&huart1,aRxBuffer,10);

```

- 结果

- 如图所示，根据代码设置，只有在接收到10个字节的时候103才触发中断将10个字节的字符发送。



- DMA

- 代码

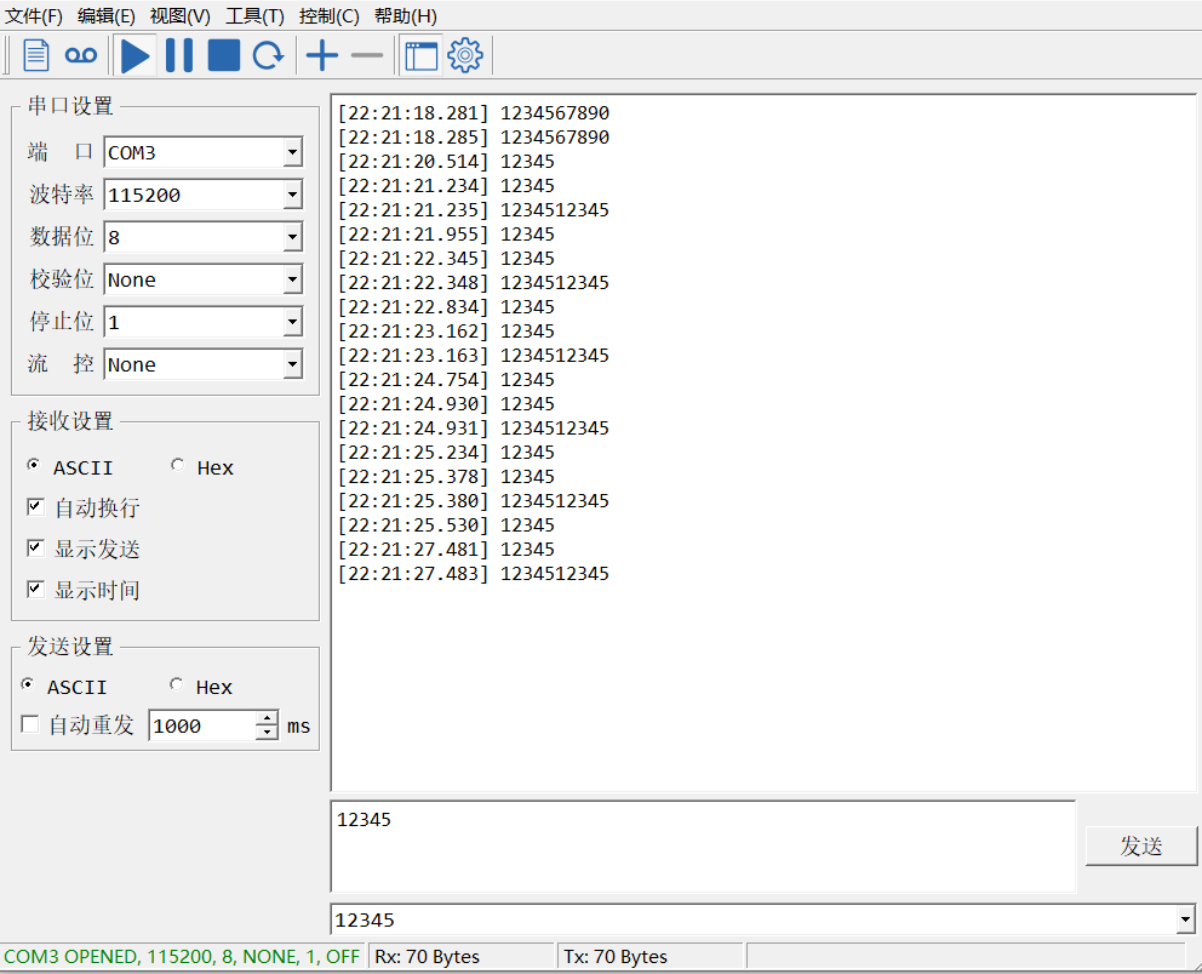
```

/* main.c定义全局变量 */
uint8_t aRxBuffer[20];
/* main.c中的main函数中while循环外插入 */
HAL_UART_Receive_DMA(&huart1,aRxBuffer,10);
/* main.c中的void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)插入 */
HAL_UART_Transmit_DMA(&huart1,aRxBuffer,10);
/* stm32f1xx_it.c中定义外部变量 */
extern uint8_t aRxBuffer[20];
/* stm32f1xx_it.c中的void USART1_IRQHandler(void)插入 */
HAL_UART_Receive_DMA(&huart1,aRxBuffer,10);

```



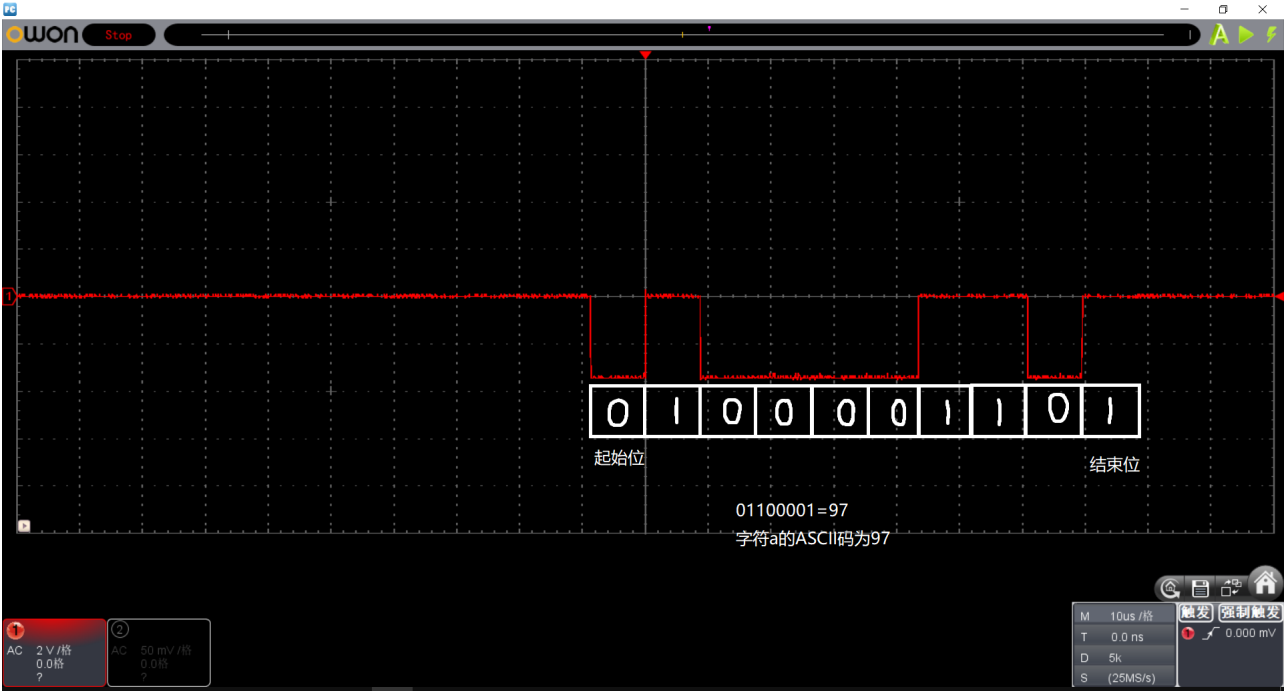
- 结果
  - 如图所示，根据代码设置，只有在接收到10个字节的时候103才触发中断将10个字节的字符发送。



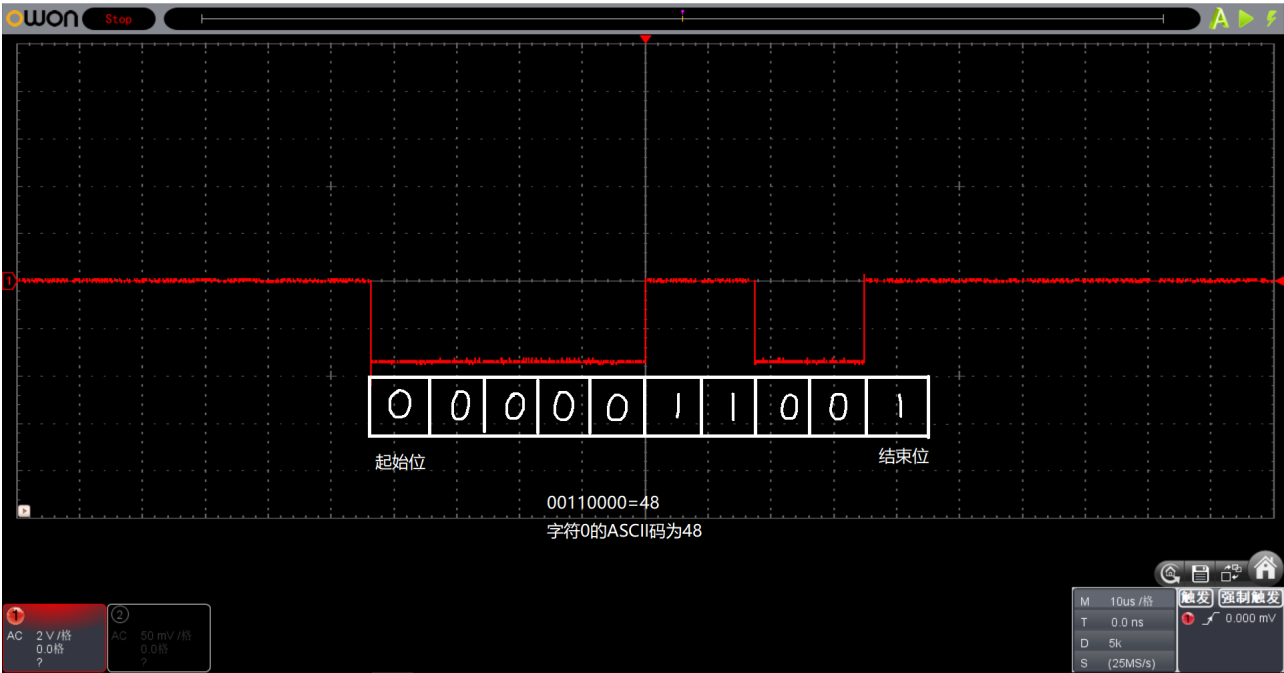
**用示波器或逻辑分析仪观察串口输出的信号，分析其速率(提示：输出0/1交替的字节);**

- 异步串行数据的一般格式是：起始位+数据位+停止位，其中起始位只占1位，数据位可以占5、6、7、8位，停止位可以占1、1.5、2位。
- 起始位是一个值为0的位，所以对于正逻辑的TTL电平，起始位是一位时间的低电平；停止位是值为1的位，所以对于正逻辑的TTL电平，停止位是高电平。线路空闲或者数据传输结束，对于正逻辑的TTL电平，线路总是1。对于负逻辑(如RS-232电平)则相反。

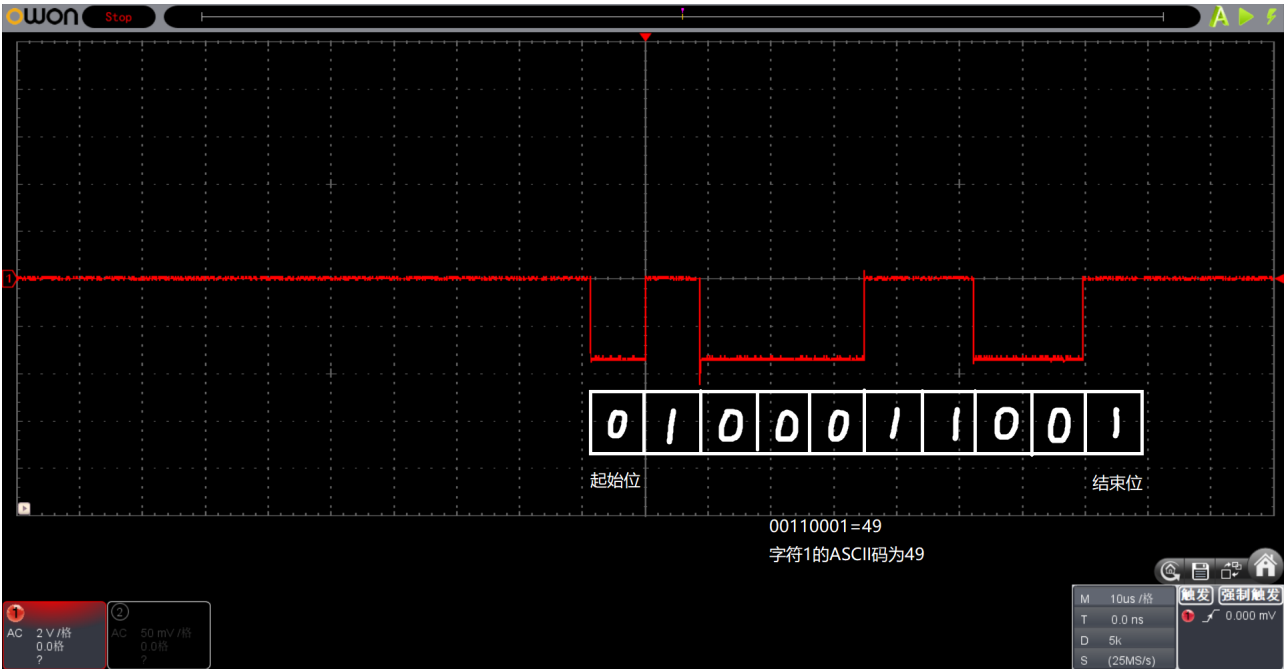
- 例如示波器接收到字符a的波形图如下：



- 示波器接收到字符0的波形图如下：



- 示波器接收到字符1的波形图如下：



- 0110110001

串口设置

端 口 COM3

波特率 115200

数据位 8

校验位 None

停止位 1

流 控 None

接收设置

☒ ASCII ☐ Hex

☒ 自动换行

☒ 显示发送

☒ 显示时间

发送设置

☒ ASCII ☐ Hex

☒ 自动重发 1000 ms

[20:50:19.749] 0110110001

[20:50:20.737] 0110110001

[20:50:20.739] 0110110001

[20:50:21.738] 0110110001

[20:50:21.741] 0110110001

[20:50:22.741] 0110110001

[20:50:22.743] 0110110001

[20:50:23.738] 0110110001

[20:50:23.740] 0110110001

[20:50:24.742] 0110110001

[20:50:24.745] 0110110001

[20:50:25.739] 0110110001

[20:50:25.741] 0110110001

[20:50:26.737] 0110110001

[20:50:26.739] 0110110001

[20:50:27.741] 0110110001

[20:50:27.743] 0110110001

[20:50:28.745] 0110110001

[20:50:28.750] 0110110001

[20:50:29.737] 0110110001

[20:50:29.740] 0110110001

[20:50:30.737] 0110110001

[20:50:30.739] 0110110001

[20:50:31.744] 0110110001

[20:50:31.747] 0110110001

[20:50:32.737] 0110110001

[20:50:32.739] 0110110001

[20:50:33.739] 0110110001

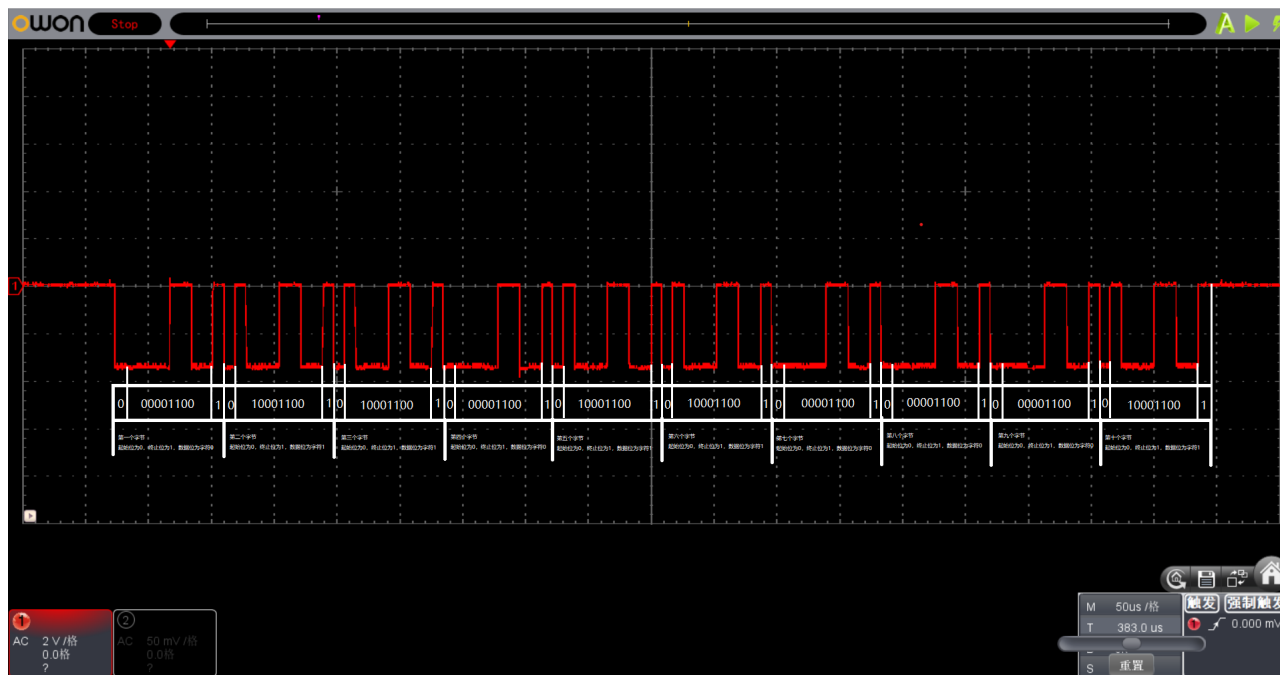
[20:50:33.741] 0110110001

0110110001

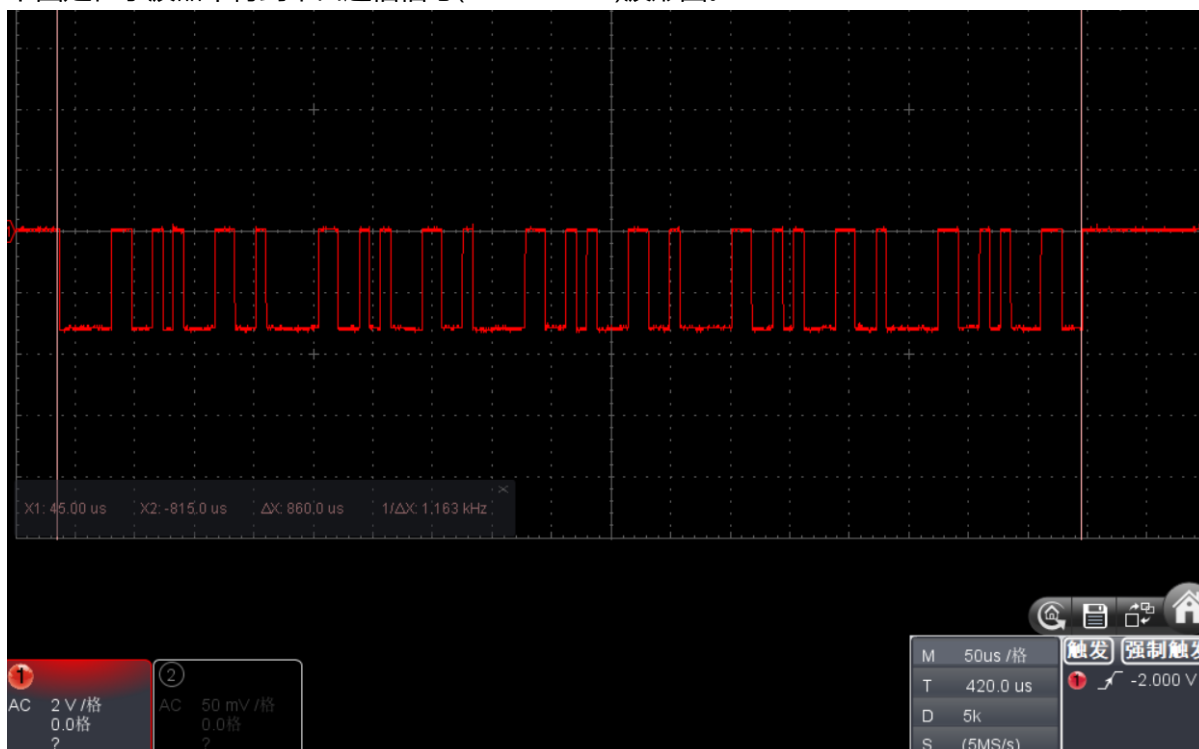
发送

0110110001

COM3 OPENED. 115200. 8. NONE. 1. OFF Rx: 34.560 Bytes Tx: 35.341 Bytes



- 分析
  - 对照发送的数据和示波器分析得到的波形图，发现串口通信采用的是正逻辑的TTL电平，即起始位是一位时间的低电平；停止位是一位时间的高电平。数据位要表示某一位用1是一位时间的高电平，表示某一位为0用一位时间的低电平。且传输字节的时候从高位开始传输。
- 根据波形图计算波特率
  - 下图是在示波器中得到串口通信信号(0101010101)波形图。



- 发送十个字节数据示波器测得发送十个字节大概需要 $869\mu s$ (图中的显示 $860\mu s$ 不准确, 因为最后一个字节的结束位没有包括进去, 十个字节, 每个字节10位(1位起始位+8位数据位+1位结束位), 一共100位, 图中两个轴之间只有99位, 所以实际四舍五入大概需要为 $860/99*100=869\mu s$ )。一个字节则大概需要 $87\mu s$ 。
  - 这样可计算出其波特率约为:  $10\text{bit}/(87 * 10^{-6})s = 114943\text{bit}/s$ , 与 $1152000\text{bit}/s$ 相对误差0.2%, 这个数值足够小, 基本可以忽略, 说明对示波器得到波形图的分析基本正确。

**编写程序，从PC发送ON/OFF字符串到103来控制TST LED的亮灭。**

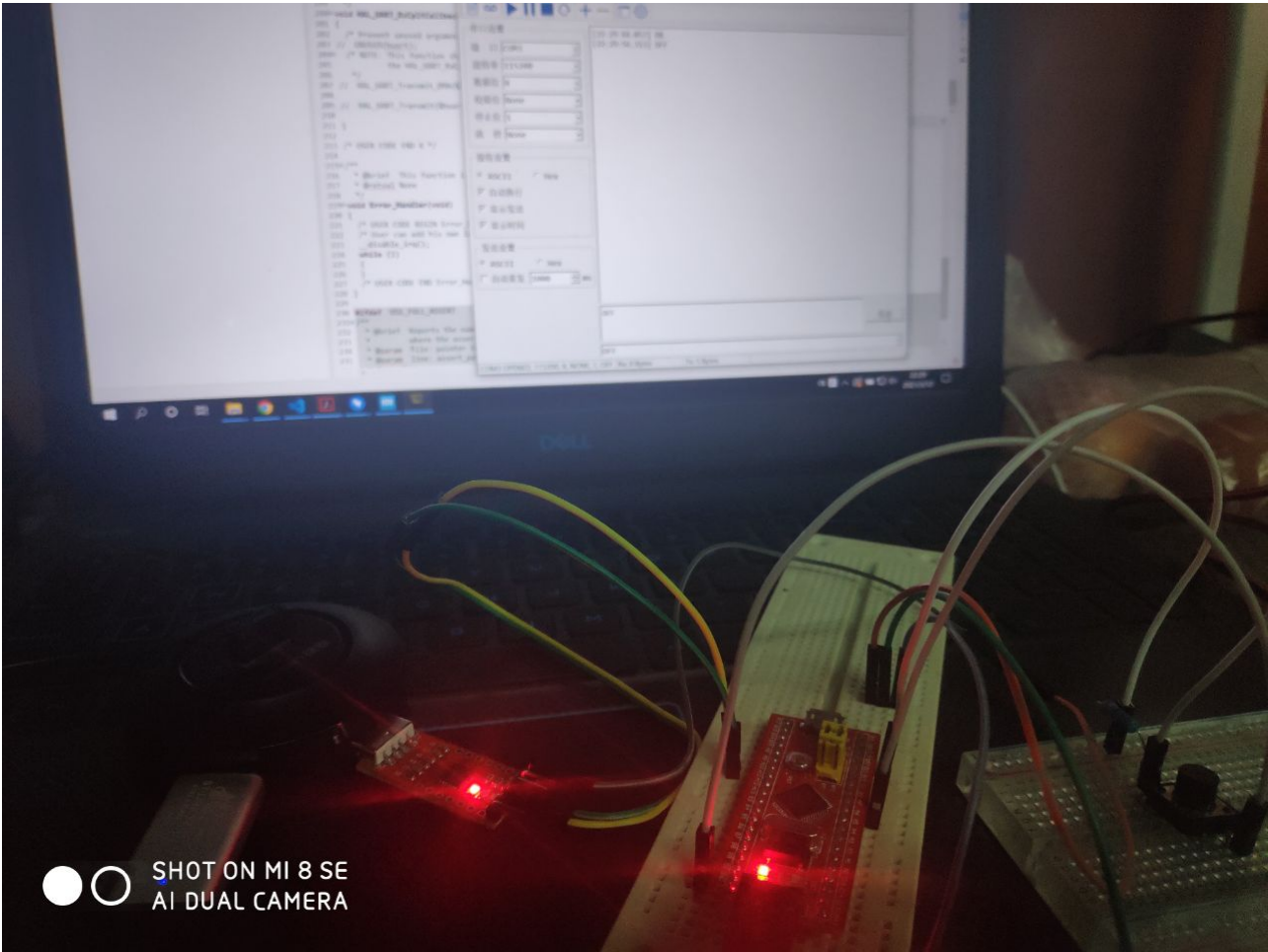
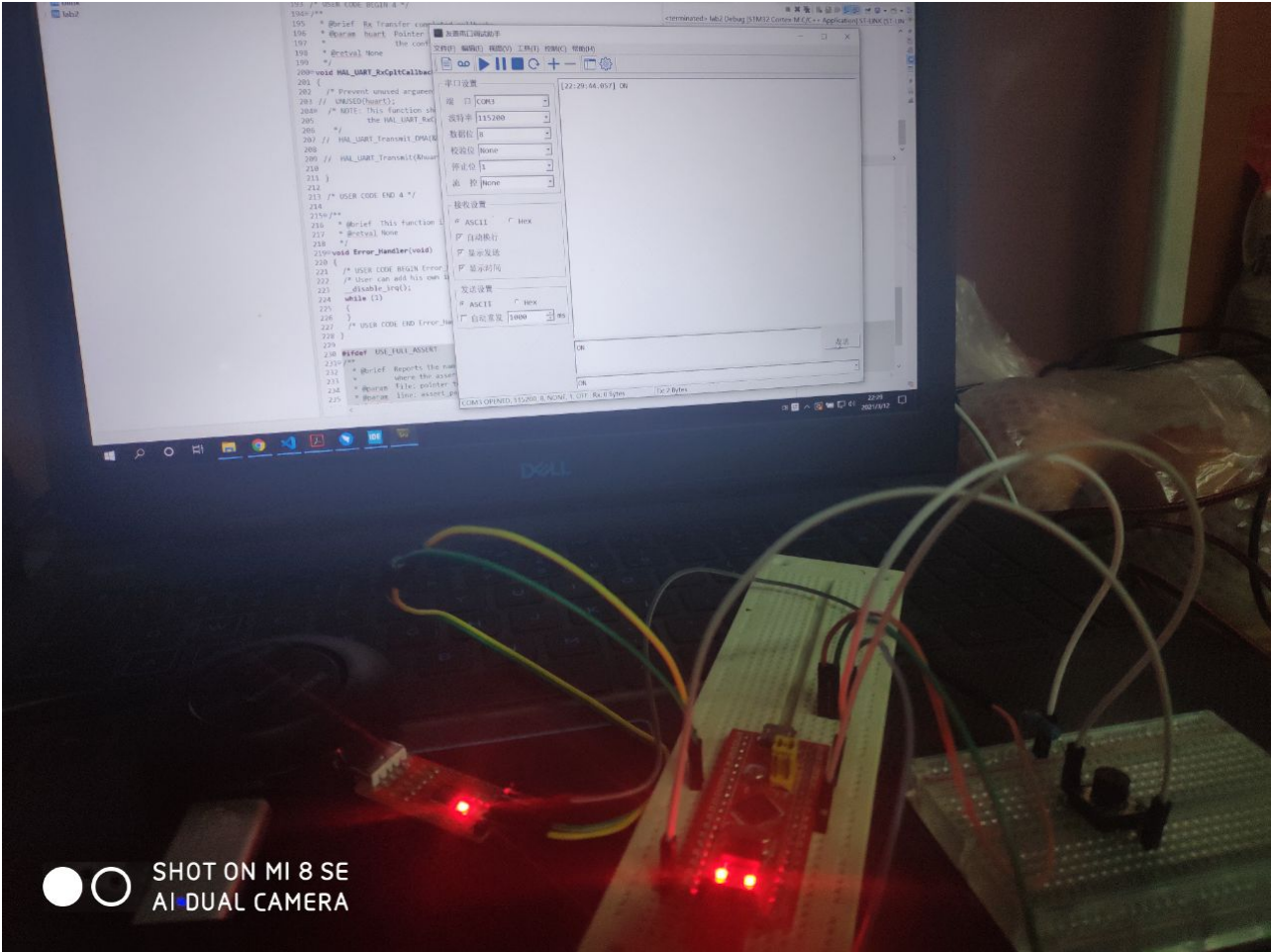
```

/* main.c定义全局变量 */
uint8_t aRxBuffer[20];
volatile uint8_t rx_len=0;
volatile uint8_t recv_end_flag=0;
char          BUFFER_SIZE=100;
100则最大长度为100
/* main.c中的main函数中while循环外插入 */
HAL_UART_Receive_DMA(&huart1,aRxBuffer,BUFFER_SIZE);
__HAL_UART_ENABLE_IT(&huart1, UART_IT_IDLE);
/* main.c中的main函数中while循环内插入 */
if(recv_end_flag ==1)
{
    if(!strcmp(aRxBuffer, "ON"))
    {
        HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_RESET);
    }
    else if(!strcmp(aRxBuffer, "OFF"))
    {
        HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET);
    }
    for(uint8_t i=0;i<rx_len;i++)
    {
        aRxBuffer[i]=0;
    }
    rx_len=0;
    recv_end_flag=0;
}
HAL_UART_Receive_DMA(&huart1,aRxBuffer,BUFFER_SIZE);
/* stm32f1xx_it.c中定义外部变量 */
extern uint8_t aRxBuffer[20];
extern volatile uint8_t rx_len;
extern volatile uint8_t recv_end_flag;
extern char          BUFFER_SIZE;
/* stm32f1xx_it.c中的void USART1_IRQHandler(void)插入 */
uint32_t tmp_flag = 0;
uint32_t temp;
tmp_flag = __HAL_UART_GET_FLAG(&huart1, UART_FLAG_IDLE);
if((tmp_flag != RESET))
{
    __HAL_UART_CLEAR_IDLEFLAG(&huart1);
    temp = huart1.Instance->SR;
    temp = huart1.Instance->DR;
    HAL_UART_DMAStop(&huart1);
    temp = hdma_usart1_rx.Instance->CNDTR;
    rx_len = BUFFER_SIZE - temp;
    recv_end_flag = 1;
}

```



- 结果(实验中的结果可详细见附件视频)



## 实验心得

- 本次实验的主题开关灯看似简单，实际内容还是较为丰富的。不仅学习了103板通过CP2102进行USB to UART的串口通信，还学习了使用开关按钮对LED开关进行控制。
- 个人在实验过程中遇到的困难主要有
  - 用按钮作为开关控制LED灯的亮灭时，需要进行去抖动;
  - 理解如何运用中断和DMA进行串口通信需要用到的中断处理函数和回调函数;
  - 用示波器接收到的串口通信信号需要进行分析;
  - 用PC发送ON和OFF控制LED灯亮灭的时候，需要在拓展1的基础上利用DMA完成收发不定长的字符。
- 通过查阅资料和仔细的观察分析，基本解决了以上的问题。更加熟悉了串口通信的收发操作。