实验 6：External Interrupt

一、实验器材

硬件：ARM-STM32 开发板，St-Link。

软件：Win10, STM32CubeIDE，HAL library

二、实验要求

1. Use EXTI to control the LED: Press KEY0 to blink LED0 three times, press KEY1 to blink LED1 three times, and transmit the corresponding message by UART.

2. Receive UART data in non-blocking mode: when the UART receives the text interrupt, transmits the corresponding message by UART.

3. About the requirement of receiving UART in non-blocking mode in assignment Lab6, your program should send response data when receives the text "interrupt" and ignore the case of receiving other text.

三、实验过程

1. 根据 lab 课内容，安装 STM32CubeIDE 和 HAL library，配置参数。

2. 实现两个功能：

   ① 用 External Interrupt 去控制 LED 灯的亮灭，要求按 KEY0 让 LED0 闪三下，按 KEY1 让 LED1 闪三下，并输出信息（“Key0 pressed”/“Key1 pressed”）。

   ② 当接收到“interrupt”的时候，给出回应（“It is a interrupt!”）；当收到其他 text 的时候，不予回应。

3. Build，生成“.hex”文件，用“FlyMcu”下载到板子上。

4. 测试，测试所有情况，观察实验板是否和预期显示一致。

四、实验结果

软件代码（只需用户实现功能的主要代码部分）：

HAL_UART_RxCpltCallback:

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart) {
    if (huart->Instance == USART1) {
        static unsigned char uRx_Data[1024] = { 0 };
        static unsigned char uLength = 0;
        if (rxBuffer[0] == '\n') {
            // Judge whether the text is interrupt
            if (uRx_Data[0] == 'i' && uRx_Data[1] == 'n' && uRx_Data[2] == 't'
                    && uRx_Data[3] == 'e' && uRx_Data[4] == 'r'
                    && uRx_Data[5] == 'r' && uRx_Data[6] == 'u'
                    && uRx_Data[7] == 'p' && uRx_Data[8] == 't') {
                // String copy to uRx_Data
                strcpy(uRx_Data, "It is a interrupt!\r\n");
                // send the message
                HAL_UART_Transmit(&huart1, uRx_Data, strlen(uRx_Data), 0xffff);
                uLength = 0;
            }
            // set the length zero
            uLength = 0;
        } else {
            uRx_Data[uLength] = rxBuffer[0];
            uLength++;
        }
    }
}
```

HAL_GPIO_EXIT_CALLback:

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) {
    HAL_Delay(100);
    switch (GPIO_Pin) {
    case KEY0_Pin:
        if (HAL_GPIO_ReadPin(KEY0_GPIO_Port, KEY0_Pin) == GPIO_PIN_RESET) {
            // send the message
            HAL_UART_Transmit(&huart1, (uint8_t*) "Key0 pressed\r\n", 14,
                    0xffff);
            // repeat for 3 times
            for (int i = 0; i < 3; i++) {
                HAL_GPIO_TogglePin(LED0_GPIO_Port, LED0_Pin);
                HAL_Delay(100);
                HAL_GPIO_TogglePin(LED0_GPIO_Port, LED0_Pin);
                HAL_Delay(100);
            }
        }
        break;
    case KEY1_Pin:
        if (HAL_GPIO_ReadPin(KEY1_GPIO_Port, KEY1_Pin) == GPIO_PIN_RESET) {
            // send the message
            HAL_UART_Transmit(&huart1, (uint8_t*) "Key1 pressed\r\n", 14,
                    0xffff);
                    // repeat for 3 times
            for (int i = 0; i < 3; i++) {
                HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);
                HAL_Delay(100);
                HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);
                HAL_Delay(100);
            }
        }
        break;
    default:
        break;
    }
}
```

USART1_IRQHandler:

```
uint8_t rxBuffer[20];
/* USER CODE END PV */

/* Private function prototypes -----------------------------------
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */
void USART1_IRQHandler(void) {
    /* USER CODE BEGIN USART1_IRQn 0 */
    /* USER CODE END USART1_IRQn 0 */
    HAL_UART_IRQHandler(&huart1);
    /* USER CODE BEGIN USART1_IRQn 1 */
    HAL_UART_Receive_IT(&huart1, (uint8_t *) rxBuffer, 1);
    /* USER CODE END USART1_IRQn 1 */
}
```

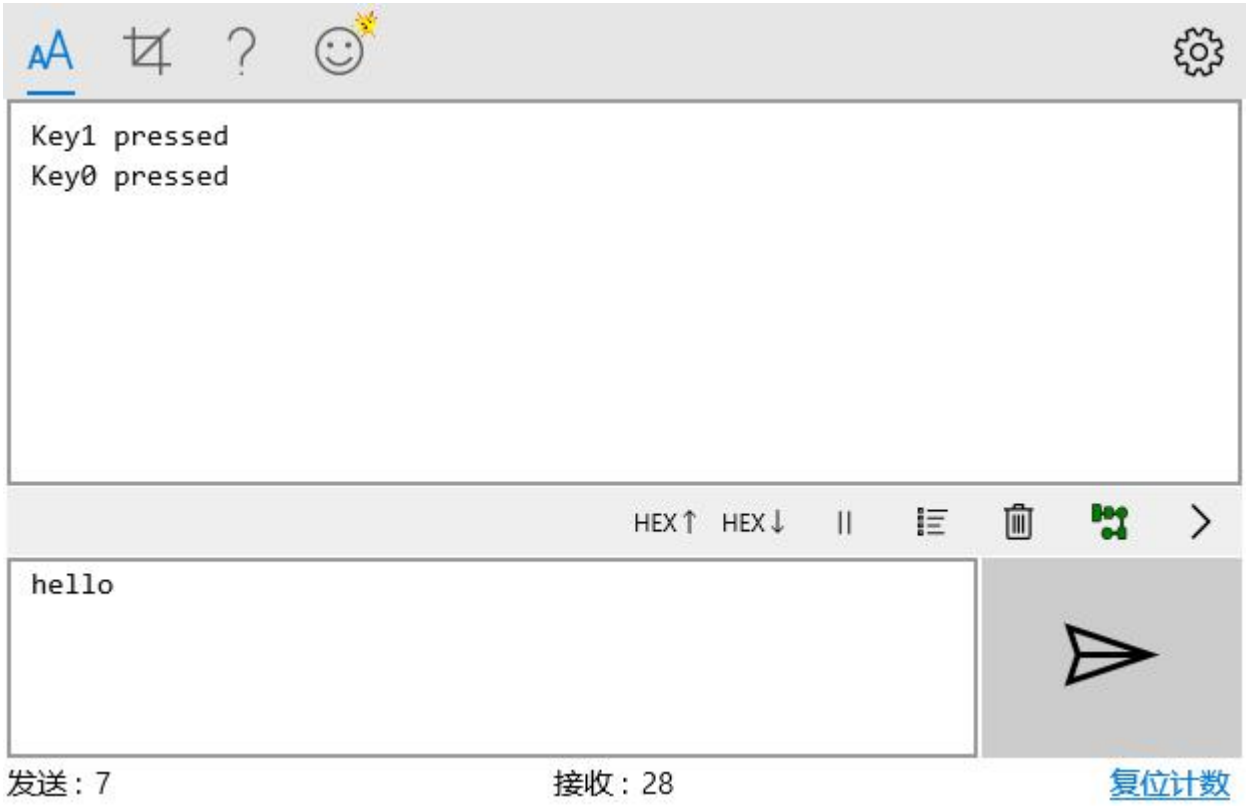
实际验证附图（含开发板状态拍照、仿真截图等）：

功能 1(LED)：

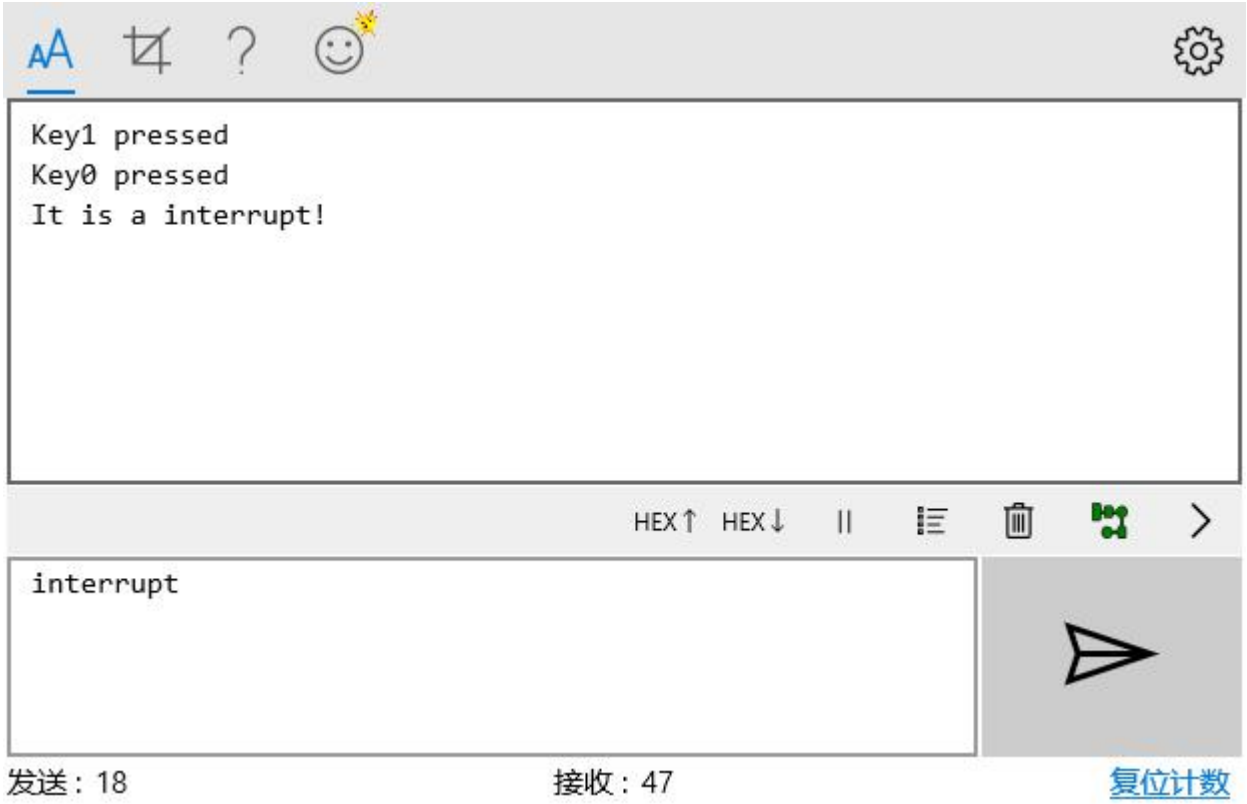
```
Key1 pressed
Key0 pressed
```

按对应的按键，对应的灯会闪三下，并且发出一条信息“Key0 pressed”/“Key1 pressed”

功能 2：



发送非“interrupt”的消息会被忽略，不予回复。



发送“interrupt”会给予回应“It is a interrupt!”。

五、实验总结

简述在实验过程中出现的问题，解决的过程和结果，及其他需要说明的情况。

1. 用异步收发传输器（Universal Asynchronous Receiver/Transmitter)，UART。它将要传输的资料在串行通信与并行通信之间加以转换。作为把并行输入信号转成串行输出信号的芯片，UART 通常被集成于其他通讯接口的连结上。

2. 比较字符串的时候，需要 include <string.h>，使用里面的 strcmp 方法，如果返回值为 0 说明两个字符串相同。

3. 赋值字符数组的时候，需要 include <string.h>，使用里面的 strcpy 方法。

4. 每次 Transmit 之后，记得将 ulength 设为 0，清空数组。

5. 串口波特率需要设置为 115200。

6. 串口调试工具内，如果代码里面返回值有中文，需要将编码格式改为 utf-8。

7. 串口调试工具内，发送的内容后需要加一个回车。

8. 每次上板前记得重新 build 一次。

9. 注意填写GPIO mode的时候,分清楚External Interrupt Mode with Rising edge trigger detection和External Interrupt Mode with Falling edge trigger detection，课件由于长度问题无法展示完全。

10. 注意设置EXTI的优先级。

11. Debug后Programming需要在Post-build steps的Command下加上arm-none-eabi-objcopy "${ProjName}.elf" -O ihex"${ProjName}.hex"

12. 用“FlyMcu”下载到板子和串口调试工具只能有一个连着，不能同时连着，会报串口占用的错误。