科星 F107 开发板学习笔记应用篇之 wifi 模块控制 LED 灯的学习

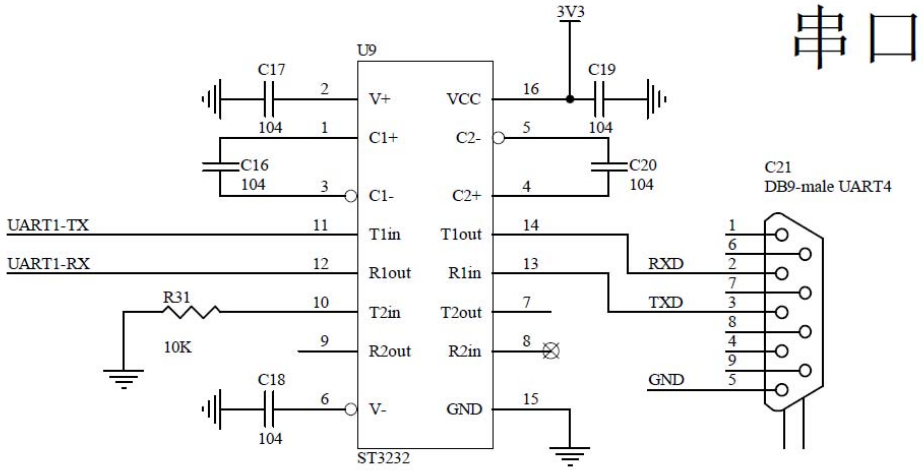
硬件准备：

1、 科星 F107 开发板
2、 ETW232D wifi 转串口模块
3、 公对母串口线（直连）
4、 装有 TCP 程序的电脑
5、 装有安卓 2.2 系统以上的智能手机（支持蓝牙）

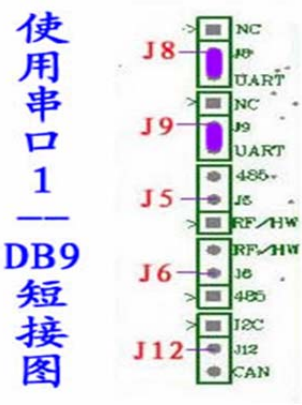
软件准备：

1、 WIFIDemo.apk（自主开发，可提供安卓源码）
2、 科星 F107 开发板上位机程序（TCP 调试助手等连接网络程序均可）
3、 开发板程序：“科星 F107 开发板应用篇之蓝牙&WIFI 控制 LED 灯”（IAR5.4）

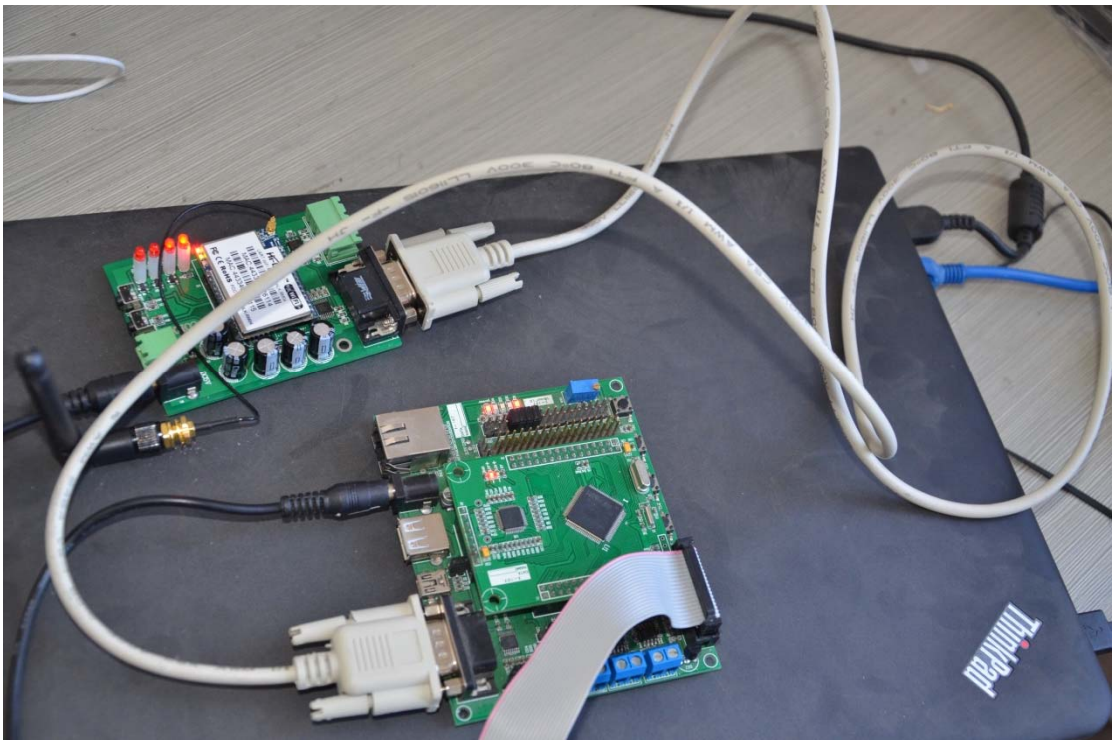
这里我们用到了开发板的串口 USART1，USART1 负责与 WIFI 模块连接，WIFI 与电脑或手机等 WIFI 设备连接。下面我们看一下开发板的原理图：



下面我们把科星 F107 串口（J8、J9、J5、J6）处的短路帽都盖好，如下图所示：



WIFI 模块与开发板连接图



下面我们开分析下程序，程序内容相对简单，主要就是两路串口的配置（USART2 是针对蓝牙模块的，WIFI 模块用不到的），程序如下：

```
SystemInit();
TIM2_Configuration();
/* NVIC configuration */
NVIC_Configuration();
/* Configure the GPIO ports */
GPIO_Configuration();
/* Configure the USART ports */
USART_Configuration();
```

这里讲一下串口 2 所用到的 GPIO 口的配置，

```
void GPIO_Configuration(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1,ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO,ENABLE);

    RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2, ENABLE);

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);

    GPIO_PinRemapConfig(GPIO_Remap_USART2, ENABLE);//重映像 USART2 的引脚

     /* Configure USARTx_Tx as alternate function push-pull */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_Init(GPIOD, &GPIO_InitStructure);

    /* Configure USARTx_Rx as input floating */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOD, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    /* Configure USARTx_Tx as alternate function push-pull */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    /* Configure USARTx_Rx as input floating */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
}
```

我们看一下寄存器手册的内容就清楚了，科星 F107 开发板上 USART2 使用的是 PD5 和 PD6 引脚作为 TX 和 RX 引脚，所以这里我们需要把 复用重映射和调试 I/O 配置寄存器的响

应值设置为 USART2 是重映像的，如下图所示：

下面我们来看主函数的操作：

```
while (1)
{
   if(r_flag==1)
   {
      if(RxBuffer1[1]==0x31)
          GPIO_SetBits(GPIOA, GPIO_Pin_4);
      else if(RxBuffer1[1]==0x30)
          GPIO_ResetBits(GPIOA, GPIO_Pin_4);
      if(RxBuffer1[2]==0x31)
          GPIO_SetBits(GPIOA, GPIO_Pin_5);
      else if(RxBuffer1[2]==0x30)
          GPIO_ResetBits(GPIOA, GPIO_Pin_5);
      if(RxBuffer1[3]==0x31)
          GPIO_SetBits(GPIOA, GPIO_Pin_7);
      else if(RxBuffer1[3]==0x30)
          GPIO_ResetBits(GPIOA, GPIO_Pin_7);
      if(RxBuffer1[4]==0x31)
          GPIO_SetBits(GPIOA, GPIO_Pin_6);
      else if(RxBuffer1[4]==0x30)
          GPIO_ResetBits(GPIOA, GPIO_Pin_6);


   USART_SendData(USART1, RxBuffer1[0]);//发送接收到的字符
   while(USART_GetFlagStatus(USART1, USART_FLAG_TXE) == RESET)//等得发送完成
   {
   }

   USART_SendData(USART1, RxBuffer1[1]);//发送接收到的字符
   while(USART_GetFlagStatus(USART1, USART_FLAG_TXE) == RESET)//等得发送完成
   {
   }
```

```
    USART_SendData(USART1, RxBuffer1[2]);//发送接收到的字符
    while(USART_GetFlagStatus(USART1, USART_FLAG_TXE) == RESET)//等得发送完成
    {
    }
    USART_SendData(USART1, RxBuffer1[3]);//发送接收到的字符
    while(USART_GetFlagStatus(USART1, USART_FLAG_TXE) == RESET)//等得发送完成
    {
    }
    USART_SendData(USART1, RxBuffer1[4]);//发送接收到的字符
    while(USART_GetFlagStatus(USART1, USART_FLAG_TXE) == RESET)//等得发送完成
    {
    }

    r_flag=0;
    }
}
```

前面是根据 串口接收到的数据进行对 LED 灯的亮灭操作，一共 5 个 16 进制数，第一位是没用的，一个前置字节，2~5 位分别为 0x31（1）或者 0x30（0），0x31 表示灯亮，0x30 表示灯灭。

这里两路串口的数据接收缓冲区均为 RxBuffer1[]，所以手机发送数据可以使开发板灯亮灭，电脑通过 USB 转串口线发送正确数据也可以控制开发板的亮灭的，程序还把外部发送给开发板的数据通过串口 1 打印到电脑上。

说明，这里外部可以发送 16 进制数据，“0x80 0x31 0x31 0x31 0x31”，也可以发送字符“21111”，第一个字符不做解析，可为任意值。

下面我们来看演示，如下图：