一、 模块介绍

NRF905 无线模块特点：

(1) 433Mhz 开放 ISM 频段免许可证使用

(2) 最高工作速率 50kbps，高效 GFSK 调制，抗干扰能力强，特别适合工业控制场合

(3) 125 频道，满足多点通信和跳频通信需要

(4) 内置硬件 CRC 检错和点对多点通信地址控制

(5) 低功耗 1.9 - 3.6V 工作，待机模式下状态仅为 2.5uA

(6) 收发模式切换时间 < 650us

(7) 模块可软件设地址，只有收到本机地址时才会输出数据（提供中断指示)，可直接接各种单片机使用，软件编程非常方便

(8) TX Mode: 在+10dBm 情况下，电流为 30mA; RX Mode: 12.2mA

(9) 标准 DIP 间距接口，便于嵌入式应用

(10)RFModule-Quick-DEV 快速开发系统，含开发板

二、接口电路管脚说明

管脚	名称	管脚功能	说明
1	VCC	电源	电源+3.3~3.6V DC
2	TX_EN	数字输入	TX_EN=1 TX 模式 TX_EN=0 RX 模式
3	TRX_CE	数字输入	使能芯片发射或接收
4	PWR_UP	数字输入	芯片上电
5	uCLK	时钟输出	本模块该脚废弃不用，向后兼容
6	CD	数字输出	载波检测
7	AM	数字输出	地址匹配
8	DR	数字输出	接收或发射数据完成
9	MISO	SPI	接口 SPI 输出
10	MOSI	SPI	接口 SPI 输入
11	SCK	SPI	时钟 SPI 时钟
12	CSN	SPI	使能 SPI 使能
13	GND	地	接地
14	GND	地	接地

说明：

(1) VCC 脚接电压范围为 3.3V~3.6V 之间，不能在这个区间之外，超过 3.6V 将会烧毁模块。推荐电压 3.3V 左右。

(2) 除电源 VCC 和接地端，其余脚都可以直接和普通的 5V 单片机 IO 口直接相连，无需电平转换。当然对 3V 左右的单片机更加适用了。

(3) 硬件上面没有 SPI 的单片机也可以控制本模块，用普通单片机 IO 口模拟 SPI 不需要单片机 SPI 模块介入，只需添加代码模拟 SPI 时序即可。

(4) 13 脚、14 脚为接地脚,需要和母板的逻辑地连接起来。

(5) 排针间距为 100mil,标准 DIP 插针。

(6) 与 51 系列单片机 P0 口连接时候，需要加 10K 的上拉电阻,与其余口连接不需要。

(7) 其他系列的单片机，如果是 5V 的，请参考该系列单片机 IO 口输出电流大小，如果超过 10mA，需要串联电阻分压，否则容易烧毁模块！如果是 3.3V 的，可以直接和 RF905 模块的 IO 口线连接。

三、模块引脚和电气参数说明

RF905 模块使用 Nordic 公司的 nRF905 芯片开发而成。 RF905 单片无线收发器工作在 433/868/915MHZ 的 ISM 频段由一个完全集成的频率调制器一个带解调器的接收器一个功率放大器一个晶体震荡器和一个调节器组成 ShockBurst 工作模式的特点是自动产生前导码 和 CRC 可以很容易通过 SPI 接口进行编程配置电流消耗很低在发射功率为＋10dBm 时发射电流为 30mA 接收电流为 12.5mA. 进入 POWERDOWN 模式可以很容易实现节电。

RF905SE 模块性能参考数据

参数	数值	单位
最低工作电压	3.0	V
最大发射功率	10	dBm
最大数据传输率曼切斯特编码	50	kbps
输出功率为-10 dBm 时工作电流	9	mA
接收模式时工作电流	12.5	mA
温度范围	-40 to+85	摄氏度
典型灵敏度	-100	dBm
POWERDOWN 模式时工作电流	2.5	uA

RF905SE 模块工作电压与最大发射增益参考数据

工作电压(模块 VCC 供电电压)	模块最大发射增益(dBm)
+3.3V	+7.3dBm
+3.6V	+10dBm

四、工作方式

RF905 一共有四种工作模式，其中有两种活动 RX/TX 模式和两种节电模式。

- 活动模式

ShockBurst RX ShockBurst TX

- 节电模式

1) 掉电 和 SPI 编程

2) STANDBY 和 SPI 编程

nRF905 工作模式由 TRX_CE、TX_EN、PWR_UP 的设置来设定。

PWR_UP	TRX_CE	TX_EN	工作模式
0	X	X	掉电和 SPI 编程
1	0	X	Standby 和 SPI 编程
1	1	0	ShockBurst RX
1	1	1	ShockBurst TX

4.1 ShockBurst 模式

ShockBurstTM 收发模式下，使用片内的先入先出堆栈区，数据低速从微控制

器送入，但高速发射，这样可以尽量节能，因此，使用低速的微控制器也能得到很高的射频数据发射速率。与射频协议相关的所有高速信号处理都在片内进行，这种做法有三大好处：尽量节能；低的系统费用(低速微处理器也能进行高速射频发射)；数据在空中停留时间短，抗干扰性高。ShockBurstTM 技术同时也减小了整个系统的平均工作电流。 在 ShockBurstTM 收发模式下， RF905 自动处理字头和 CRC 校验码。在接收数据时，自动把字头和 CRC 校验码移去。在发送数据时，自动加上字头和 CRC 校验码，当发送过程完成后，DR 引脚通知微处理器数据发射完毕。

4.1.1 ShockBurst TX 发送流程

典型的 RF905 发送流程分以下几步：

A. 当微控制器有数据要发送时，通过 SPI 接口，按时序把接收机的地址和要发送的数据送传给 RF905，SPI 接口的速率在通信协议和器件配置时确定；

B. 微控制器置高 TRX_CE 和 TX_EN，激发 RF905 的 ShockBurstTM 发送模式；

C. RF905 的 ShockBurstTM 发送：

(1) 射频寄存器自动开启；

(2) 数据打包(加字头和 CRC 校验码)；

(3) 发送数据包；

(4) 当数据发送完成，数据准备好引脚被置高；

D. AUTO_RETRAN 被置高， RF905 不断重发，直到 TRX_CE 被置低；

E. 当 TRX_CE 被置低，RF905 发送过程完成，自动进入空闲模式。 注意：ShockBurstTM 工作模式保证，一旦发送数据的过程开始，无论 TRX_EN 和 TX_EN 引脚是高或低，发送过程都会被处理完。只有在前一个数据包被发送完毕，RF905 才能接受下一个发送数据包。

4.1.2 ShockBurst RX 接收流程

接收流程

A. 当 TRX_CE 为高、TX_EN 为低时，RF905 进入 ShockBurstTM 接收模式；

B. 650us 后，RF905 不断监测，等待接收数据；

C. 当 RF905 检测到同一频段的载波时，载波检测引脚被置高；

D. 当接收到一个相匹配的地址，AM 引脚被置高；

E. 当一个正确的数据包接收完毕， RF905 自动移去字头、地址和 CRC 校验位，然后把 DR 引脚置高

F. 微控制器把 TRX_CE 置低，nRF905 进入空闲模式；

G. 微控制器通过 SPI 口，以一定的速率把数据移到微控制器内；

H. 当所有的数据接收完毕，nRF905 把 DR 引脚和 AM 引脚置低；

I. nRF905 此时可以进入 ShockBurstTM 接收模式、ShockBurstTM 发送模式或关机模式。

当正在接收一个数据包时，TRX_CE 或 TX_EN 引脚的状态发生改变，RF905 立即把其工作模式改变，数据包则丢失。当微处理器接到 AM 引脚的信号之后， 其就知道 RF905 正在接收数据包，其可以决定是让 RF905 继续接收该数据包还是进入另一个工作模式。

4.1.3 节能模式

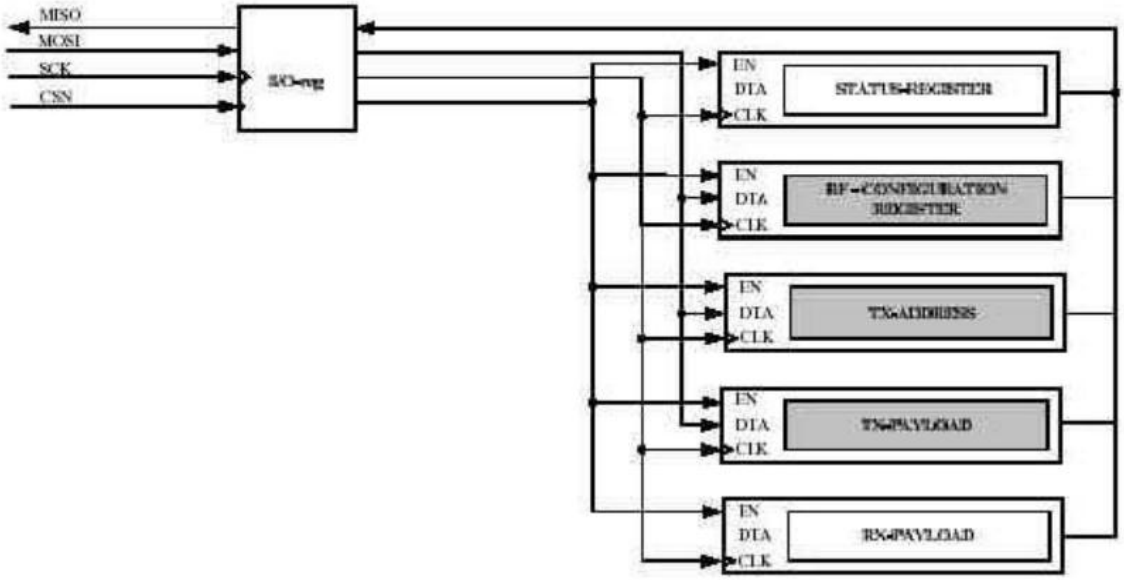
RF905 的节能模式包括关机模式和节能模式。在关机模式，RF905 的工作电流最小，一般为 2.5uA。进入关机模式后，RF905 保持配置字中的内容，但不会接收或发送任何数据。 空闲模式有利于减小工作电流，其从空闲模式到发送模式或接收模式的启动时间也比较短。在空闲模式下，RF905 内部的部分晶体振荡器处于工作状态。

五、配置 RF905 模块

所有配置字都是通过 SPI 接口送给 RF905。SIP 接口的工作方式可通过 SPI 指令进行设置。当 RF905 处于空闲模式或关机模式时，SPI 接口可以保持在工作状态。

5.1 SPI 接口寄存器配置

SPI 接口由状态寄存器、射频配置寄存器、发送地址寄存器发送数据寄存器和接收数据寄存器 5 个寄存器组成。状态寄存器包含数据准备好引脚状态信息和地址匹配引脚状态信息；射频配置寄存器包含收发器配置信息，如频率和输出功能等；发送地址寄存器包含接收机的地址和数据的字节数；发送数据寄存器包含待发送的数据包的信息，如字节数等；接收数据寄存器包含要接收的数据的字节数等信息。SPI 接口由 5 个内部寄存器组成执行寄存器的回读模式来确认寄存器的内容。



状态寄存器 Status-Register

寄存器包含数据就绪 DR 和地址匹配 AM 状态

RF 配置寄存器 RF-Configuration Register

寄存器包含收发器的频率,输出功率等配置信息

发送地址 TX-Address

寄存器包含目标器件地址字节长度由配置寄存器设置

发送有效数据 TX-Payload

寄存器包含发送的有效 ShockBurst 数据包数据字节长度由配置寄存器设置

接收有效数据 TX-Payload

寄存器包含接收到的有效 ShockBurst 数据包数据字节长度由配置寄存器设置在寄存器中的有效数据由数据准备就绪 DR 指示

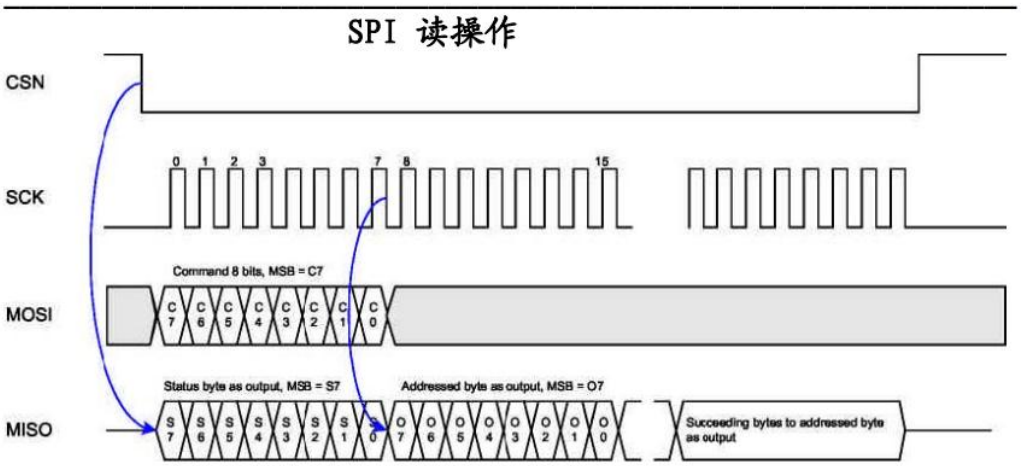
5.2 SPI 指令设置

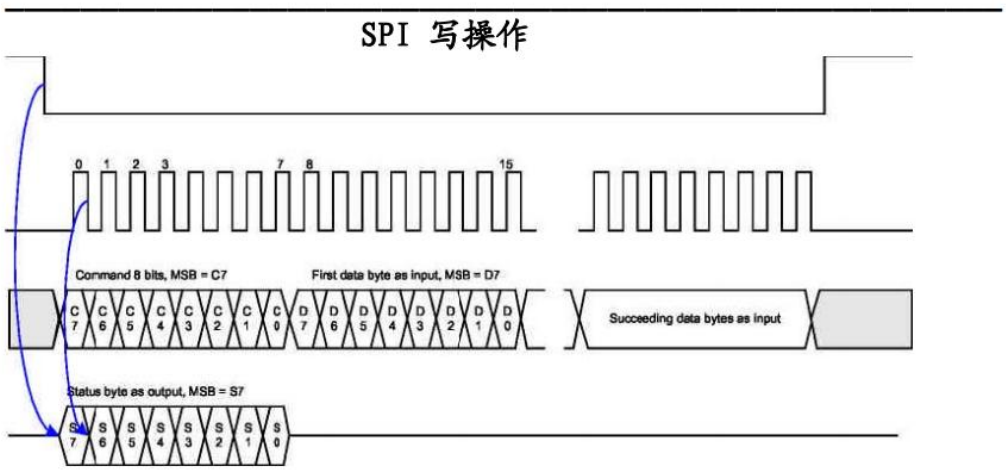
当 CSN 为低时, SPI 接口开始等待一条指令。任何一条新指令均由 CSN 的由高到低的转换开始。用于 SPI 接口的有用命令见下表：

SPI 串行接口指令设置

		SPI 串行接口指令
指令名称	指令格式	操作
W_CONFIG(WC)	0000AAAA	写配置寄存器 AAAA 指出写操作的开始字节字节数量取决于 AAAA 指出的开始地址
R_CONFIG(RC)	0001AAAA	读配置寄存器 AAAA 指出读操作的开始字节字节数量取决于 AAAA 指出的开始地址
W_TX_PAYLOA D(WTP)	00100000	写 TX 有效数据 1-32 字节写操作全部从字节 0 开始
R_TX_PAYLOA D(RTP)	00100001	读 TX 有效数据 1-32 字节读操作全部从字节 0 开始
W_TX_ADDRES S(WTA)	00100010	写 TX 地址 1-4 字节写操作全部从字节 0 开始
R_TX_ADDRES S(RTA)	00100011	读 TX 地址 1-4 字节读操作全部从字节 0 开始
R_RX_PAYLOA D(RRP)	00100100	读 RX 有效数据 1-32 字节读操作全部从字节 0 开始
CHANNEL_CON FIG(CC)	1000pphc cccccccc	快速设置配置寄存器中 CH_NO HFREQ_PLL 和 PA_PWR 的专用命令 CH_NO=ccccccccc HFREQ_PLL=h PA_PWR=pp

5.3 SPI 时序


SPI 读操作
CSN
SCK
MOSI
MISO
Command 8 bits, MSB = C7
Status byte as output, MSB = S7
Addressed byte as output, MSB = O7
Succeeding bytes to addressed byte as output


SPI 写操作
Command 8 bits, MSB = C7
First data byte as input, MSB = D7
Succeeding data bytes as input
Status byte as output, MSB = S7

GS-03 是一款基于单片机的无线开发板，最大限度地使用 SMT 贴片工艺技术，合理的器件布局与高质量的器件选用，使开发板更显人性化，并且获得最好的稳定性。板载 nRF24L01+和 nRF905 无线模块接口，配合丰富的外接接口及完整的开发例程，为用户开发无线产品带来便利。

开发板标准配置：

名称	数量
GS-3 单片机开发板	2 块
11.0592 晶振	2 个
串口连接线	1 条
STC89C52RC 单片机	2 片
9V 电源适配器	2 个
nRF905 无线模块	2 个
DS18B20	1 个
杜邦线	5 条

开发板硬件资源：

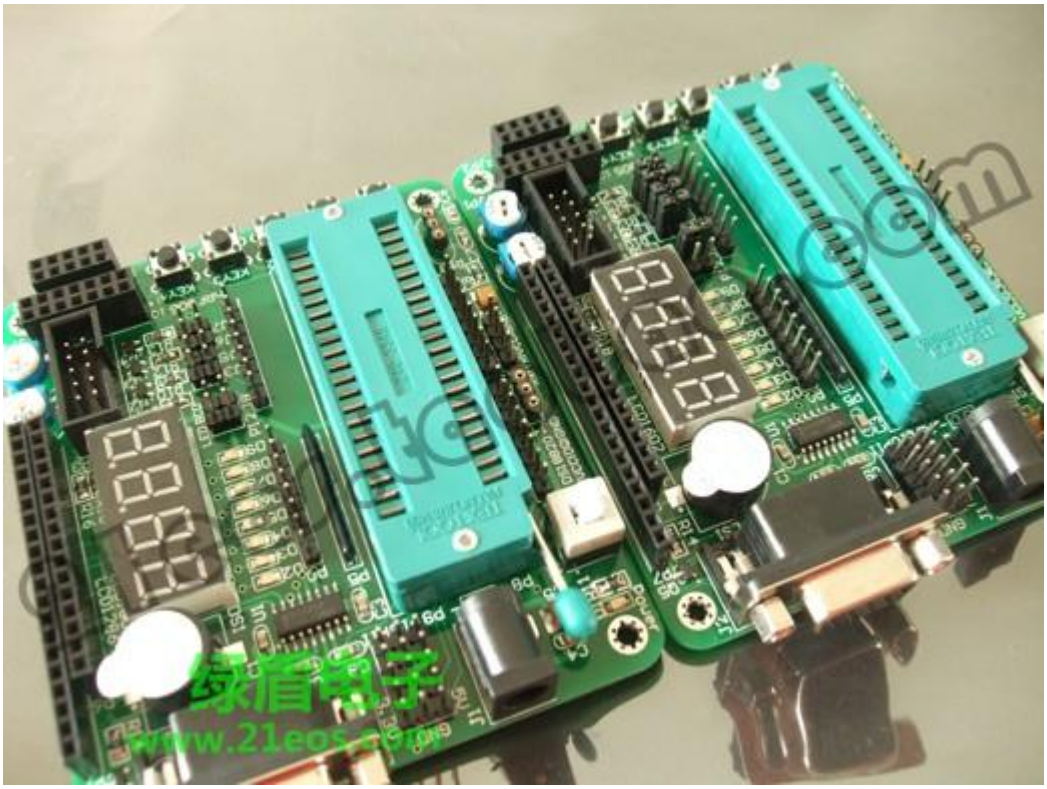
1. 3.3V 和 5V 稳压电路

2. 1A 自恢复保险丝电路
3. 8 个 LED 指示灯
4. 4 位共阳数码管
5. 5V 有源蜂鸣器
6. MAX202 串口电路
7. AT89S5X 系列单片机的 ISP 下载接口
8. 单片机复位按键电路
9. 4 个用户自定义按键
10. DS18B20 接口电路
11. nRF24L01+无线模块接口电路
12. nRF905 无线模块接口电路
13. LCD12864 液晶屏电路，可调显示灰度
14. LCD1602 液晶电路，可调显示灰度

开发板特色：

1. 5V、3.3V 电压输出接口
2. 活动单片机插座，方便更换单片机
3. 使用圆孔晶振接口，方便更换晶振
4. 支持 STC 和 AT 单片机
5. 胆电容滤波电路，是开发板稳定性的强力后盾

一对无线开发板：

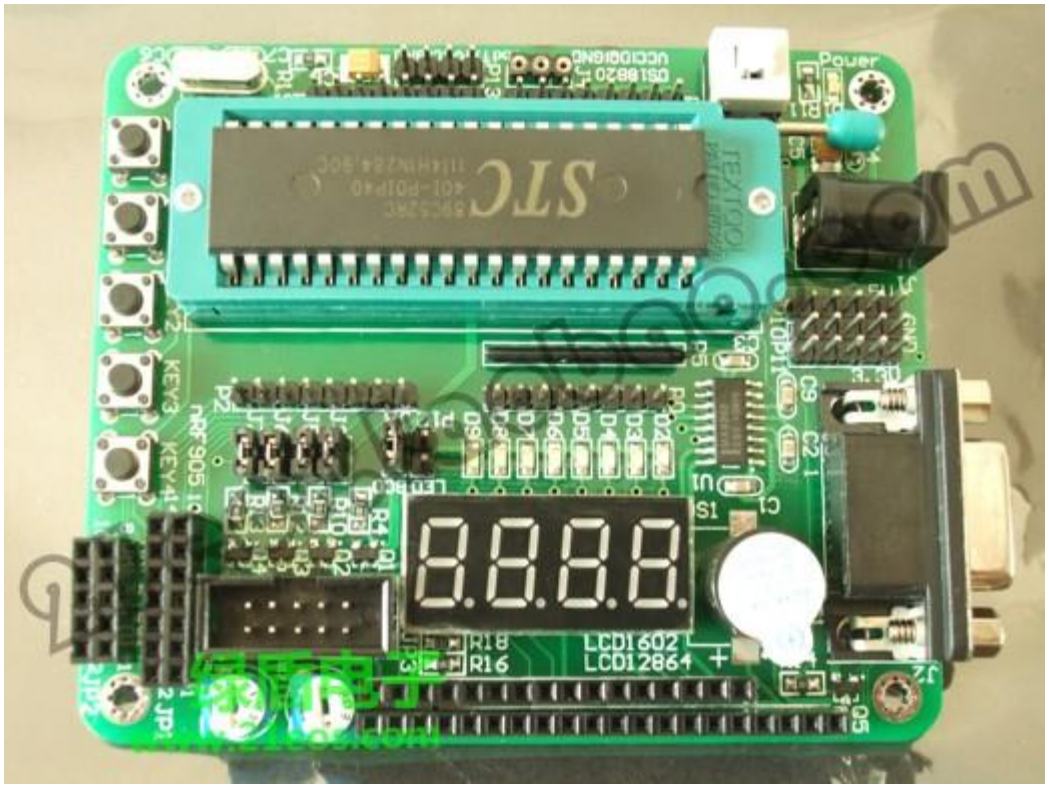


单独一个开发板：

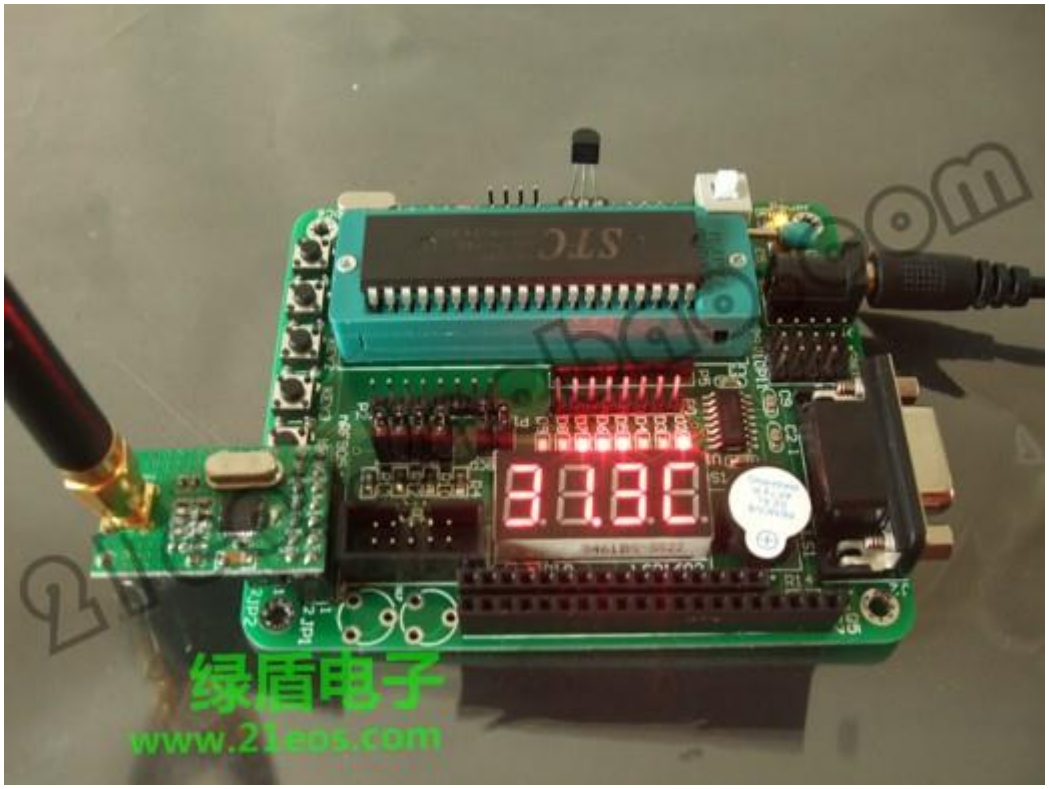
开发板背面：


GS-03 V2.0
21eos.taobao.com

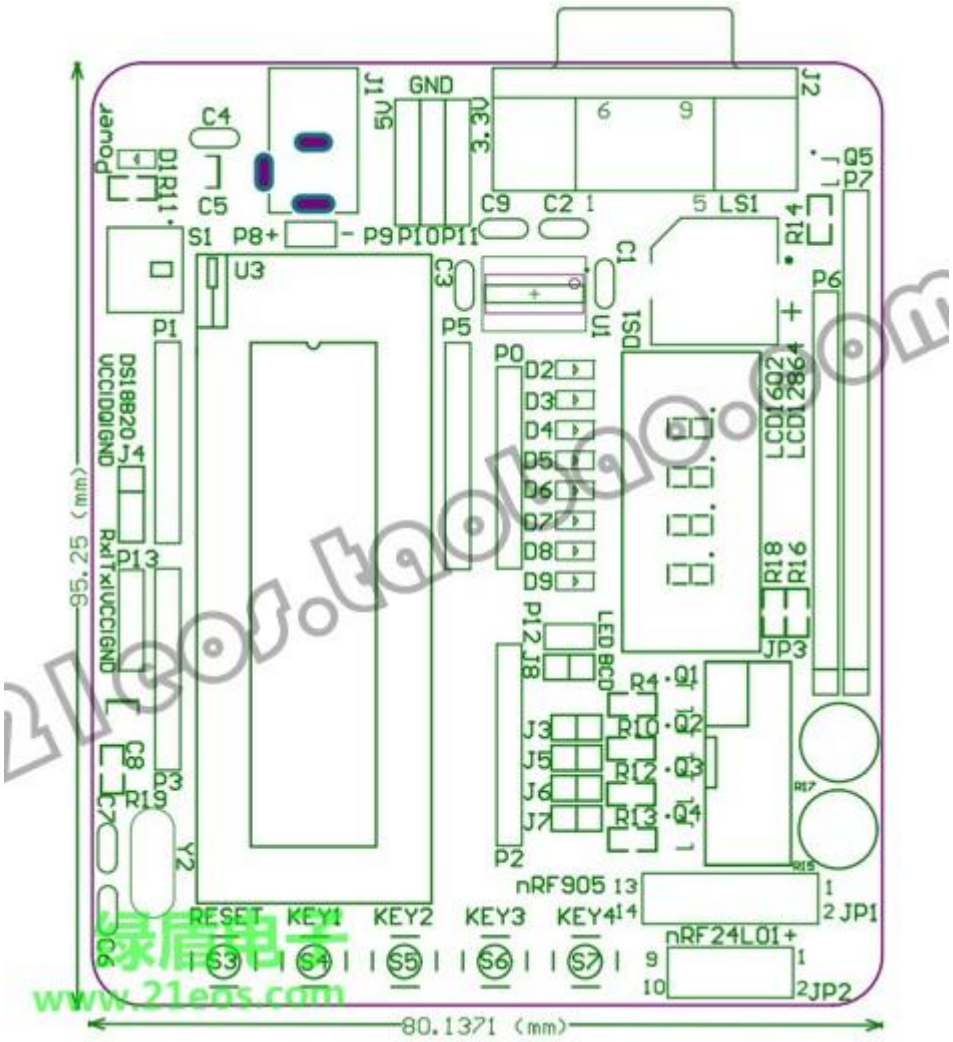
开发板正面：


STC
LCD1602
LCD12864
R18
R16
Power

开发板采集温度：

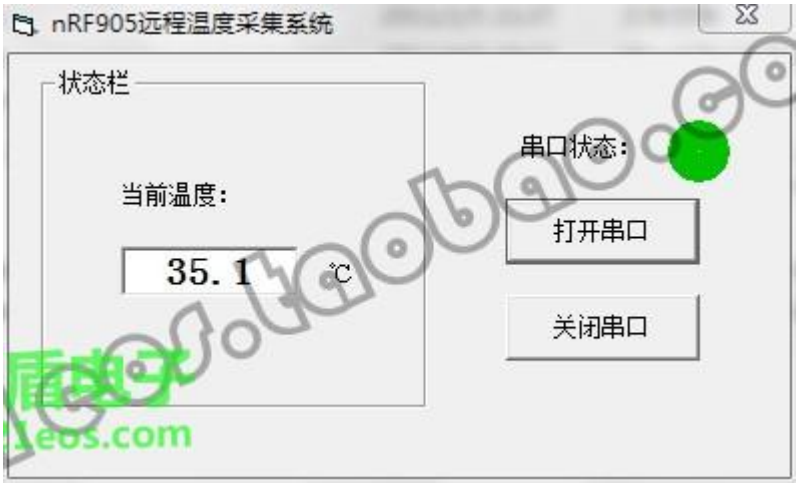


开发板尺寸 95mm*80mm：



无线继电器控制 VB 上位机：

无线温度检测（精简版）VB 上位机：



无线温度检测绘制曲线 VB 上位机：


温度采集数据
086 42.4 ℃
087 42.4 ℃
088 41.9 ℃
089 41.9 ℃
090 41.7 ℃
091 41.7 ℃
092 41.4 ℃
093 41.4 ℃
094 41.2 ℃
095 41.2 ℃
096 41.0 ℃
097 41.0 ℃
清 除
连接
已连接在COM1
连接测试
温度曲线
当前温度值值：41.0 ℃
最低温度值：34.9 ℃
最高温度值：69.3 ℃
连续读取
停止读取
退 出

51 单片机驱动 nRF905 例程

```
/****************************************
//DS18B20 温度传感器//
****************************************/

/*************************************************/
//包含头文件
#include "inc/nrf905.h"

bdata unsigned    char DATA_BUF;
#define DATA7 ((DATA_BUF&BYTE_BIT7) != 0)
#define DATA0      ((DATA_BUF&BYTE_BIT0) != 0)
sbit  flag  =DATA_BUF^7;
sbit  flag1=DATA_BUF^0;

unsigned char nRF905_TxRxBuf[nRF905_TxRxBuf_Len]=
{
      0x00,0x03,0x1,0x02,
};

code nRF905_TxAddress[4]={0xcc,0xcc,0xcc,0x00};

//-----------------------------------------NRF905 寄存器配置-----------------------------------------
unsigned char idata RFConf[11]=
{
      0x00,                                   //配置命令//
      0x4c,                                   //CH_NO,配置频段在 423MHZ
      0x0C,                                   //输出功率为 10db,不重发，节电为正常模式
```

```
    0x44,                                   //地址宽度设置，为 4 字节
    0x04,0x04,                              //接收发送有效数据长度为 32 字节
    0xCC,0xCC,0xCC,0xCC,                    //接收地址
    0x58,                                   //CRC 充许，8 位 CRC 校验，外部时钟信号不使
能，16M 晶振
};

//-----------------------------------------延时-----------------------------------------------------
void delay(uchar n)
{
    uint i;
    while(n--)
    for(i=0;i<80;i++);
}

//-----------------------------------------SPI 写函数-----------------------------------------------
void SpiWrite(unsigned char send)
{
    unsigned char i;
    DATA_BUF=send;
    for (i=0;i<8;i++)
    {
        if (DATA7)//总是发送最高位
        {
            MOSI=1;
        }
        else
        {
            MOSI=0;
        }
```

```
            SCK=1;
            DATA_BUF=DATA_BUF<<1;
            SCK=0;
        }
}

//-----------------------------------------------SPI 读 1 字节函数-----------------------------------------------
unsigned char SpiRead(void)
{
        unsigned char j;

        for (j=0;j<8;j++)
        {
            DATA_BUF=DATA_BUF<<1;
            SCK=1;
            if (MISO) //读取最高位，保存至最末尾，通过左移位完成整个字节
            {
                DATA_BUF|=BYTE_BIT0;
            }
            else
            {
                DATA_BUF&=~BYTE_BIT0;
            }
            SCK=0;
         }
         return DATA_BUF;
}

//-------------------------------------------初始化 nRF905 状态-------------------------------------------------
void nRF905_Init(void)
```

```
{
    CSN=1;                          // Spi    disable
    SCK=0;                          // Spi clock line init low
    DR=1;                           // Init DR for input
    AM=1;                           // Init AM for input
    CD=1;                           // Init CD for input
    PWR=1;                      // nRF905 power on
    TRX_CE=0;                       // Set nRF905 in standby mode
    TXEN=0;                     // set radio in Rx mode
}
//-----------------------------------------NRF905 初始化寄存器----------------------------------------
void nRF905_Config(void)
{
    uchar i;

    CSN=0;                          // Spi enable for write a spi command
    //SpiWrite(WC);                 // Write config command 写放配置命令
    for (i=0;i<11;i++)    // Write configration words   写放配置字
    {
        SpiWrite(RFConf[i]);
    }
    CSN=1;                      // Disable Spi
}

//-----------------------------------------设置发送模式----------------------------------------------
void nRF905_SetTxMode(void)
{
    TRX_CE=0;
    TXEN=1;
    delay(1);                       // delay1 for mode change(>=650us)
```

```
}

//-----------------------------------------------设置接收状态-----------------------------------------------
void nRF905_SetRxMode(void)
{
    TXEN=0;
    TRX_CE=1;
    delay(1);                          // delay for mode change(>=650us)
    //delay(20);//GGG
}

//-------------------------------------------------------------------------------------------------------
unsigned char nRF905_CheckCD(void)      //Pin->检查是否已存在 同频率载波
{
    if (CD==1)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}

//-----------------------------------------------判断是否接收数据-----------------------------------------
unsigned char nRF905_CheckDR(void)      //检查是否有新数据传入 Data Ready
{
    DR=1;
    //通过对端口写 1，可以使端口为输入状态，这 51 的 特性。不熟悉者可以参阅 51 相关书籍作证(将 DR 端口设置为输入状态。)
```

```
    if (DR==1)
    {
        DR=0;
        return 1;
    }
    else
    {
        return 0;
    }
}

//---------------------------------------打包待发送的数据-----------------------------------------------------
void nRF905_nRF905_TxPacket(void)
{
    uchar i;
    //nRF905_Config();
    CSN=0;
    SpiWrite(WTP);                  // Write payload command
    for (i=0;i<4;i++)
    {
        SpiWrite(nRF905_TxRxBuf[i]);    // Write 32 bytes nRF905_Tx data
    }// Spi enable for write a spi command
    CSN=1;
    delay(1);                       // Spi disable
    CSN=0;                          // Spi enable for write a spi command
    SpiWrite(WTA);                  // Write address command
    for (i=0;i<4;i++)               // Write 4 bytes address
    {
        SpiWrite(nRF905_TxAddress[i]);
    }
```

```
    CSN=1;                          // Spi disable
    TRX_CE=1;                       // Set TRX_CE high,start nRF905_Tx data transmission
    delay(1);                   // while (DR!=1);
    TRX_CE=0;                       // Set TRX_CE low
}

//----------------------------------------------数据发送----------------------------------------------
void nRF905_Tx(void)
{
    nRF905_SetTxMode();// Set nRF905 in nRF905_Tx mode
    nRF905_nRF905_TxPacket();// Send data by nRF905
}

//----------------------------------------------读取接收缓冲区数据----------------------------------
void nRF905_RxPacket(void)                          //读数据
{
    uchar i;
    delay(1);
    //TRX_CE=0;                     // Set nRF905 in standby mode
    TRX_CE=0;
    CSN=0;                          // Spi enable for write a spi command
    delay(1);
    SpiWrite(RRP);
    for (i = 0 ;i < 4 ;i++)
    {
        nRF905_TxRxBuf[i]=SpiRead();        // Read data and save to buffer
    }
    CSN=1;
    TRX_CE=1;
}
```

```
//-----------------------------------------------数据接收-----------------------------------------------
void    nRF905_Rx(void)
{
      nRF905_SetRxMode();                 // Set nRF905 in Rx mode
      delay(10);
      if(nRF905_CheckDR())
            nRF905_RxPacket();
}
```