实验七 无线数据传输 实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | **班级** | **学号** | **指导老师** |
| 上官鸿 | 计算1614 | 201621121110 | 刘晋明 |

1. **实验目的**
2. 掌握SPI总显得使用方式
3. 掌握NRF24L01的工作原理和使用方式
4. 掌握无线传输数据的基本原理
5. **实验内容**

学习NRF24L01的工作原理，利用总线技术实现两台单片机之间的AD转换数据的无线传输

1. **实验步骤**
2. 编写NRF24L01的读写程序

nrf24l01.h :

#ifndef \_\_NRF24L01\_H\_

#define \_\_NRF24L01\_H\_

#include "reg52.h"

// Define interface to nRF24L01

// Define SPI pins

sbit CE **=** P1 **^** 0**;** // Chip Enable pin signal (output)

sbit CSN **=** P3 **^** 7**;** // Slave Select pin, (output to CSN, nRF24L01)

sbit IRQ **=** P3 **^** 3**;** // Interrupt signal, from nRF24L01 (input)

sbit MISO **=** P2 **^** 0**;** // Master In, Slave Out pin (input)

sbit MOSI **=** P2 **^** 2**;** // Serial Clock pin, (output)c

sbit SCK **=** P3 **^** 6**;** // Master Out, Slave In pin (output)

#ifndef uchar

#define uchar unsigned char

#endif

#ifndef uint

#define uint unsigned int

#endif

#define TX\_ADR\_WIDTH 5 // 5 uints TX address width

#define RX\_ADR\_WIDTH 5 // 5 uints RX address width

#define TX\_PLOAD\_WIDTH 5 // 5 uints TX payload

#define RX\_PLOAD\_WIDTH 5 // 5 uints TX payload

// SPI(nRF24L01) commands

#define READ\_REG 0x00 // Define read command to register

#define WRITE\_REG 0x20 // Define write command to register

#define RD\_RX\_PLOAD 0x61 // Define RX payload register address

#define WR\_TX\_PLOAD 0xA0 // Define TX payload register address

#define FLUSH\_TX 0xE1 // Define flush TX register command

#define FLUSH\_RX 0xE2 // Define flush RX register command

#define REUSE\_TX\_PL 0xE3 // Define reuse TX payload register command

#define NOP 0xFF // Define No Operation, might be used to read status register

// SPI(nRF24L01) registers(addresses)

#define CONFIG 0x00 // 'Config' register address

#define EN\_AA 0x01 // 'Enable Auto Acknowledgment' register address

#define EN\_RXADDR 0x02 // 'Enabled RX addresses' register address

#define SETUP\_AW 0x03 // 'Setup address width' register address

#define SETUP\_RETR 0x04 // 'Setup Auto. Retrans' register address

#define RF\_CH 0x05 // 'RF channel' register address

#define RF\_SETUP 0x06 // 'RF setup' register address

#define STATUS 0x07 // 'Status' register address

#define OBSERVE\_TX 0x08 // 'Observe TX' register address

#define CD 0x09 // 'Carrier Detect' register address

#define RX\_ADDR\_P0 0x0A // 'RX address pipe0' register address

#define RX\_ADDR\_P1 0x0B // 'RX address pipe1' register address

#define RX\_ADDR\_P2 0x0C // 'RX address pipe2' register address

#define RX\_ADDR\_P3 0x0D // 'RX address pipe3' register address

#define RX\_ADDR\_P4 0x0E // 'RX address pipe4' register address

#define RX\_ADDR\_P5 0x0F // 'RX address pipe5' register address

#define TX\_ADDR 0x10 // 'TX address' register address

#define RX\_PW\_P0 0x11 // 'RX payload width, pipe0' register address

#define RX\_PW\_P1 0x12 // 'RX payload width, pipe1' register address

#define RX\_PW\_P2 0x13 // 'RX payload width, pipe2' register address

#define RX\_PW\_P3 0x14 // 'RX payload width, pipe3' register address

#define RX\_PW\_P4 0x15 // 'RX payload width, pipe4' register address

#define RX\_PW\_P5 0x16 // 'RX payload width, pipe5' register address

#define FIFO\_STATUS 0x17 // 'FIFO Status Register' register address

void Delay1ms**(**uint y**);**

uchar SPI\_RW**(**uchar byte**);**

uchar SPI\_RW\_Reg**(**uchar reg**,** uchar value**);**

uchar SPI\_Read**(**uchar reg**);**

uchar SPI\_Read\_Buf**(**uchar reg**,** uchar **\***pBuf**,** uchar bytes**);**

uchar SPI\_Write\_Buf**(**uchar reg**,** uchar **\***pBuf**,** uchar bytes**);**

void NRF24L01\_Config**(**void**);**

uchar NRF24L01\_RxPacket**(**uchar **\***rx\_buf**);**

void NRF24L01\_TxPacket**(**uchar **\*** tx\_buf**);**

void RX\_Mode**(**void**);**

void TX\_Mode**(**void**);**

#endif

nrf24l01.c :

#include "nrf24l01.h"

uchar TX\_ADDRESS**[**TX\_ADR\_WIDTH**]** **=** **{**0xE7**,** 0xE7**,** 0xE7**,** 0xE7**,** 0xE7**};** //本地地址

uchar RX\_ADDRESS**[**RX\_ADR\_WIDTH**]** **=** **{**0xE7**,** 0xE7**,** 0xE7**,** 0xE7**,** 0xE7**};** //接收地址

sbit led7 **=** P1**^**7**;**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名 : Delay1ms

\* 函数功能 : 延时函数

\* 输 入 : 无

\* 输 出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay1ms**(**uint y**)** **{**

uint x**;**

**for** **(;** y **>** 0**;** y**--)** **{**

**for** **(**x **=** 110**;** x **>** 0**;** x**--);**

**}**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function: SPI\_RW();

Description:

Writes one byte to nRF24L01, and return the byte read

from nRF24L01 during write, according to SPI protocol

假设下面的8位寄存器装的是待发送的数据10101010，上升沿发送、下降沿接收、高位先发送。

那么第一个上升沿来的时候 数据将会是sdo=1；

寄存器中的10101010左移一位，后面补入送来的一位未知数x，成了0101010x。

下降沿到来的时候，sdi上的电平将锁存到寄存器中去，那么这时寄存器=0101010sdi，

这样在 8个时钟脉冲以后，两个寄存器的内容互相交换一次。这样就完成里一个spi时序

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

uchar SPI\_RW**(**uchar byte**)** **{**

uchar bit\_ctr**;**

**for** **(**bit\_ctr **=** 0**;** bit\_ctr **<** 8**;** bit\_ctr**++)** **{** // output 8-bit // 输出 8 位

MOSI **=** **(**byte **&** 0x80**);** // output 'byte', MSB to MOSI // 从MOSI脚输出,从高位到低位.当uchar 最高位为1时逻辑运算为1,输出1.运算为0时输出0

byte **=** **(**byte **<<** 1**);** // shift next bit into MSB.. // 左移1位.将低位向高位转移.

SCK **=** 1**;** // Set SCK high.. // 将SCK脚拉高,

byte **|=** MISO**;** //byte=byte|MISO capture current MISO bit // 从MISO 中读出状态位存入uchar中.

SCK **=** 0**;** // ..then set SCK low again // 将SCK脚拉低.发送一位完成.

**}**

**return** **(**byte**);** // return read byte // 返回状态位

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function: SPI\_RW\_Reg();

Description:

Writes value 'value' to register 'reg'

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

uchar SPI\_RW\_Reg**(**uchar reg**,** uchar value**)** **{**

uchar status**;**

CSN **=** 0**;** // CSN low, init SPI transaction // CS为低,开始SPI传输

status **=** SPI\_RW**(**reg**);** // select register // 传送命令与地址

SPI\_RW**(**value**);** // ..and write value to it.. // 写1 uchar 值

CSN **=** 1**;** // CSN high again // 完成SPI传输

**return** **(**status**);** // 返回 status // return nRF24L01 status byte

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function: SPI\_Read();

Description:

Read one byte from nRF24L01 register, 'reg'

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

uchar SPI\_Read**(**uchar reg**)** **{**

uchar reg\_val**;**

CSN **=** 0**;** // CSN low, initialize SPI communication... // CS为低,开始SPI传输

SPI\_RW**(**reg**);** // Select register to read from.. // 设置读地址

reg\_val **=** SPI\_RW**(**0**);** // ..then read registervalue // 读数据

CSN **=** 1**;** // CSN high, terminate SPI communication // 完成SPI传输

**return** **(**reg\_val**);** // return register value // 返回数据

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function: SPI\_Read\_Buf();

Description:

Reads 'bytes' of bytes from register 'reg'

Typically used to read RX payload, Rx/Tx address

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uchar SPI\_Read\_Buf**(**uchar reg**,** uchar **\*** pBuf**,** uchar bytes**)** **{**

uchar status**,** byte\_ctr**;**

CSN **=** 0**;** // CS为低,开始SPI传输// Set CSN low, init SPI tranaction

status **=** SPI\_RW**(**reg**);** // 传输读接收数据的命令. // Select register to write to and read status byte

**for** **(**byte\_ctr **=** 0**;** byte\_ctr **<** bytes**;** byte\_ctr**++)**

pBuf**[**byte\_ctr**]** **=** SPI\_RW**(**0**);** // 读 1 uchar数据 // Perform SPI\_RW to read byte from nRF24L01

CSN **=** 1**;** // 数据传送完成. // Set CSN high again

**return** **(**status**);** // return nRF24L01 status byte

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function: SPI\_Write\_Buf();

Description:

Writes contents of buffer '\*pBuf' to nRF24L01

Typically used to write TX payload, Rx/Tx address

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uchar SPI\_Write\_Buf**(**uchar reg**,** uchar **\*** pBuf**,** uchar bytes**)** **{**

uchar status**,** byte\_ctr**;**

CSN **=** 0**;** // CS为低,开始SPI传输 // Set CSN low, init SPI tranaction

status **=** SPI\_RW**(**reg**);** // 传输写发送数据的命令. // Select register to write to and read status byte

**for** **(**byte\_ctr **=** 0**;** byte\_ctr **<** bytes**;** byte\_ctr**++)**// 传送发送数据 // then write all byte in buffer(\*pBuf)

SPI\_RW**(** **\*** pBuf**++);**

CSN **=** 1**;** // 数据传送完成. // Set CSN high again

**return** **(**status**);** // return nRF24L01 status byte

**}**

//配置函数

void NRF24L01\_Config**(**void**)** **{**

//initial io

CE **=** 0**;** //芯片使能

CSN **=** 1**;** //SPI禁能

SCK **=** 0**;** //SPI时钟线拉高

CE **=** 0**;**

SPI\_RW\_Reg**(**WRITE\_REG **+** CONFIG**,** 0X0F**);** //上电模式，使能CRC(2个字节)和RX\_DR使能

SPI\_RW\_Reg**(**WRITE\_REG **+** EN\_AA**,** 0X3f**);** //频道0自动，ack应答允许

SPI\_RW\_Reg**(**WRITE\_REG **+** EN\_RXADDR**,** 0X3f**);** //允许pipe0

SPI\_RW\_Reg**(**WRITE\_REG **+** RF\_CH**,** 0**);** // 设置信道工作为2.4GHZ，收发必须一致

SPI\_RW\_Reg**(**WRITE\_REG **+** RF\_SETUP**,** 0x07**);** //设置发射速率为1MHZ，发射功率为最大值0dB

SPI\_RW\_Reg**(**WRITE\_REG **+** CONFIG**,** 0x0e**);** // IRQ收发完成中断响应，16位CRC，主发送

SPI\_RW\_Reg**(**WRITE\_REG **+** RX\_PW\_P0**,** RX\_PLOAD\_WIDTH**);** //设置接收数据长度，本次设置为5字节

SPI\_Write\_Buf**(**WRITE\_REG **+** TX\_ADDR**,** TX\_ADDRESS**,** TX\_ADR\_WIDTH**);** // 写本地地址

SPI\_Write\_Buf**(**WRITE\_REG **+** RX\_ADDR\_P0**,** RX\_ADDRESS**,** RX\_ADR\_WIDTH**);** // 写接收端地址0

SPI\_RW\_Reg**(**WRITE\_REG **+** CONFIG**,** 0x0e**);** // IRQ收发完成中断响应，16位CRC，主发送

CE **=** 1**;** /////////////

**}**

/////接收数据

uchar NRF24L01\_RxPacket**(**uchar **\*** rx\_buf**)** **{**

uchar revale **=** 0**,** sta**;**

SPI\_RW\_Reg**(**WRITE\_REG **+** CONFIG**,** 0X0F**);**

CE **=** 1**;**

Delay1ms**(**10**);**

sta **=** SPI\_Read**(**STATUS**);**

**if** **(**sta **&** 0x40**)** **{**//RX\_DR = 1

CE **=** 0**;**

SPI\_Read\_Buf**(**RD\_RX\_PLOAD**,** rx\_buf**,** RX\_PLOAD\_WIDTH**);**

revale **=** 1**;**

**}**

SPI\_RW\_Reg**(**WRITE\_REG **+** STATUS**,** sta**);**

**return** revale**;**

**}**

void NRF24L01\_TxPacket**(**uchar **\*** tx\_buf**)** **{**

CE **=** 0**;**

SPI\_Write\_Buf**(**WR\_TX\_PLOAD**,** tx\_buf**,** TX\_PLOAD\_WIDTH**);** // Writes data to TX payload

CE **=** 1**;**

Delay1ms**(**10**);**

**}**

//发送数据

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function: RX\_Mode();

Description:

This function initializes one nRF24L01 device to

RX Mode, set RX address, writes RX payload width,

select RF channel, datarate & LNA HCURR.

After init, CE is toggled high, which means that

this device is now ready to receive a datapacket.

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

void RX\_Mode**(**void**)** **{**

CE **=** 0**;**

SPI\_Write\_Buf**(**WRITE\_REG **+** RX\_ADDR\_P0**,** RX\_ADDRESS**,** RX\_ADR\_WIDTH**);** // Use the same address on the RX device as the TX device

SPI\_RW\_Reg**(**WRITE\_REG **+** EN\_AA**,** 0x01**);** // Enable Auto.Ack:Pipe0

SPI\_RW\_Reg**(**WRITE\_REG **+** EN\_RXADDR**,** 0x01**);** // Enable Pipe0

SPI\_RW\_Reg**(**WRITE\_REG **+** RF\_CH**,** 40**);** // Select RF channel 40

SPI\_RW\_Reg**(**WRITE\_REG **+** RX\_PW\_P0**,** RX\_PLOAD\_WIDTH**);** // Select same RX payload width as TX Payload width

SPI\_RW\_Reg**(**WRITE\_REG **+** RF\_SETUP**,** 0x07**);** // TX\_PWR:0dBm, Datarate:2Mbps, LNA:HCURR

SPI\_RW\_Reg**(**WRITE\_REG **+** CONFIG**,** 0x0f**);** // Set PWR\_UP bit, enable CRC(2 bytes) & Prim:RX. RX\_DR enabled..

CE **=** 1**;** // Set CE pin high to enable RX device

Delay1ms**(**10**);**

//CE = 0;

// This device is now ready to receive one packet of 16 bytes payload from a TX device sending to address

// '3443101001', with auto acknowledgment, retransmit count of 10, RF channel 40 and datarate = 2Mbps.

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function: TX\_Mode();

Description:

This function initializes one nRF24L01 device to

TX mode, set TX address, set RX address for auto.ack,

fill TX payload, select RF channel, datarate & TX pwr.

PWR\_UP is set, CRC(2 bytes) is enabled, & PRIM:TX.

ToDo: One high pulse(>10us) on CE will now send this

packet and expext an acknowledgment from the RX device.

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

void TX\_Mode**(**void**)** **{**

CE **=** 0**;**

SPI\_Write\_Buf**(**WRITE\_REG **+** TX\_ADDR**,** TX\_ADDRESS**,** TX\_ADR\_WIDTH**);** // Writes TX\_Address to nRF24L01

SPI\_Write\_Buf**(**WRITE\_REG **+** RX\_ADDR\_P0**,** RX\_ADDRESS**,** RX\_ADR\_WIDTH**);** // RX\_Addr0 same as TX\_Adr for Auto.Ack

SPI\_RW\_Reg**(**WRITE\_REG **+** EN\_AA**,** 0x01**);** // Enable Auto.Ack:Pipe0

SPI\_RW\_Reg**(**WRITE\_REG **+** EN\_RXADDR**,** 0x01**);** // Enable Pipe0

SPI\_RW\_Reg**(**WRITE\_REG **+** SETUP\_RETR**,** 0x1a**);** // 500us + 86us, 10 retrans...

SPI\_RW\_Reg**(**WRITE\_REG **+** RF\_CH**,** 40**);** // Select RF channel 40

SPI\_RW\_Reg**(**WRITE\_REG **+** RF\_SETUP**,** 0x07**);** // TX\_PWR:0dBm, Datarate:2Mbps, LNA:HCURR

SPI\_RW\_Reg**(**WRITE\_REG **+** CONFIG**,** 0x0e**);** // Set PWR\_UP bit, enable CRC(2 bytes) & Prim:TX. MAX\_RT & TX\_DS enabled..

CE **=** 1**;**

//Delay1ms(10);

// CE = 0;

**}**

1. 编写主控制程序，实现双向通信。

main.c :

#include "reg52.h" //此文件中定义了单片机的一些特殊功能寄存器

#include "nrf24l01.h"

#include "lcd.h"

#include "XPT2046.h"

**typedef** unsigned char u8**;**

**typedef** unsigned int u16**;**

sbit beep **=** P1**^**5**;**

uchar DATA**[**5**]** **=** **{**0**},** rx\_buf**[**5**]** **=** **{**0**};**

u16 cnt **=** 0**,** temp **=** 0**;**

int readData**()**

**{**

u16 t**,** change **=** 0**;**

**if(**cnt **==** 25**)**

**{**

cnt **=** 0**;**

t **=** Read\_AD\_Data**(**0x94**);** // 电位器

**if(**t **!=** temp**)** **{**

change **=** 1**;**

temp **=** t**;**

DATA**[**0**]=** temp**/**1000**;**//千位

DATA**[**1**]=** temp**%**1000**/**100**;**//百位

DATA**[**2**]=** temp**%**1000**%**100**/**10**;**//十位

DATA**[**3**]=** temp**%**1000**%**100**%**10**;**//个位

**}**

**}**

cnt **++;**

**return** change**;**

**}**

uchar NRF24L01\_Check**(**void**)**

**{**

uchar bu**[**5**]={**0XA5**,**0XA5**,**0XA5**,**0XA5**,**0XA5**};**

uchar bu1**[**5**];**

uchar i**;**

SPI\_Write\_Buf**(**WRITE\_REG**+**TX\_ADDR**,**bu**,**5**);** //写入5个字节的地址.

SPI\_Read\_Buf**(**TX\_ADDR**,**bu1**,**5**);** //读出写入的地址

**for(**i**=**0**;**i**<**5**;**i**++)if(**bu1**[**i**]!=**0XA5**)break;**

**if(**i**!=**5**)return** 1**;** //NRF24L01不在位

**return** 0**;** //NRF24L01在位

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名 : main

\* 函数功能 : 主函数

\* 输 入 : 无

\* 输 出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main**()**

**{**

LcdInit**();**

NRF24L01\_Config**();**

**while(**NRF24L01\_Check**())** //检查不到24l01则报警

**{**

beep**=**0**;**

Delay1ms**(**20**);**

beep**=**1**;**

Delay1ms**(**20**);**

**}**

**while** **(**1**)**

**{**

RX\_Mode**();**

**if(**NRF24L01\_RxPacket**(**rx\_buf**))** **{**

LcdWriteCom**(**0x80**);** //设置数据指针起点

LcdWriteData**(**rx\_buf**[**0**]** **%** 10 **+** '0'**);**

LcdWriteData**(**rx\_buf**[**1**]** **%** 10 **+** '0'**);**

LcdWriteData**(**rx\_buf**[**2**]** **%** 10 **+** '0'**);**

LcdWriteData**(**rx\_buf**[**3**]** **%** 10 **+** '0'**);**

**}**

**if(**readData**())** **{**

TX\_Mode**();**

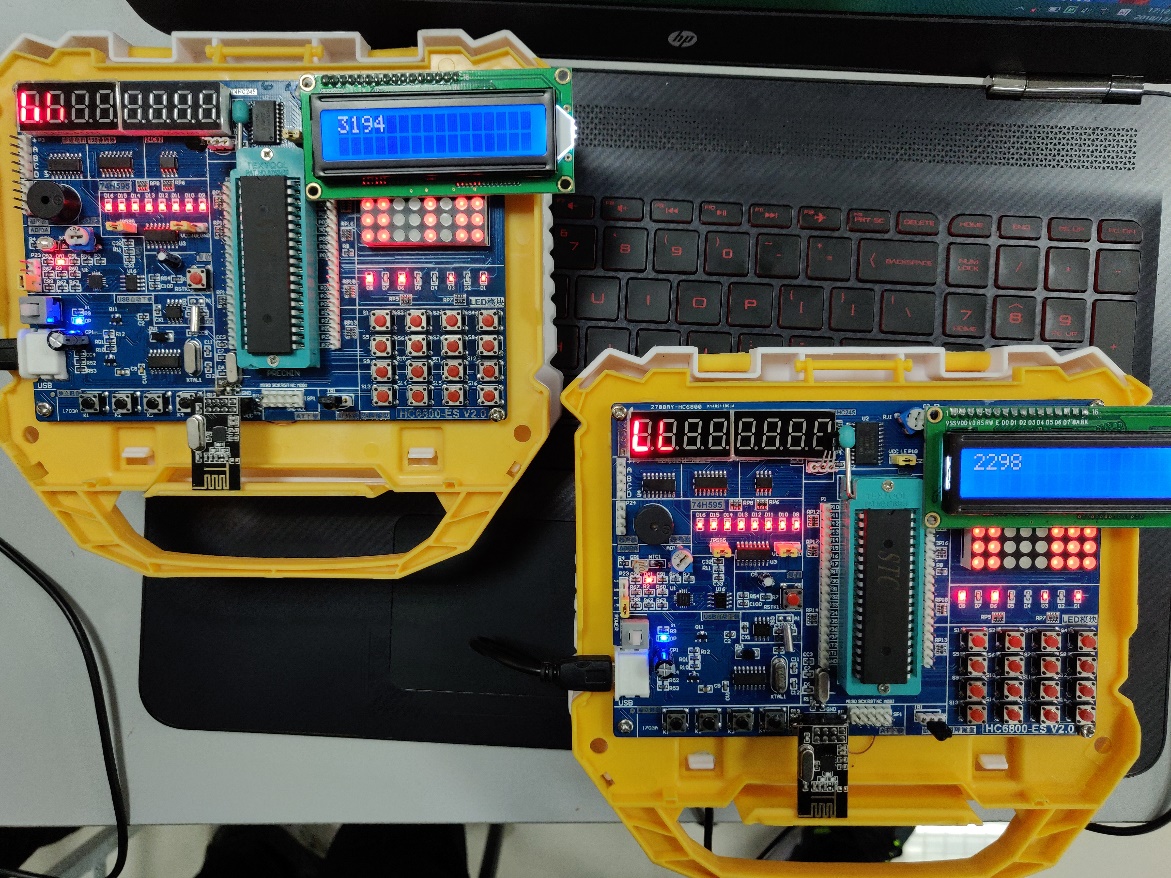
NRF24L01\_TxPacket**(**DATA**);**

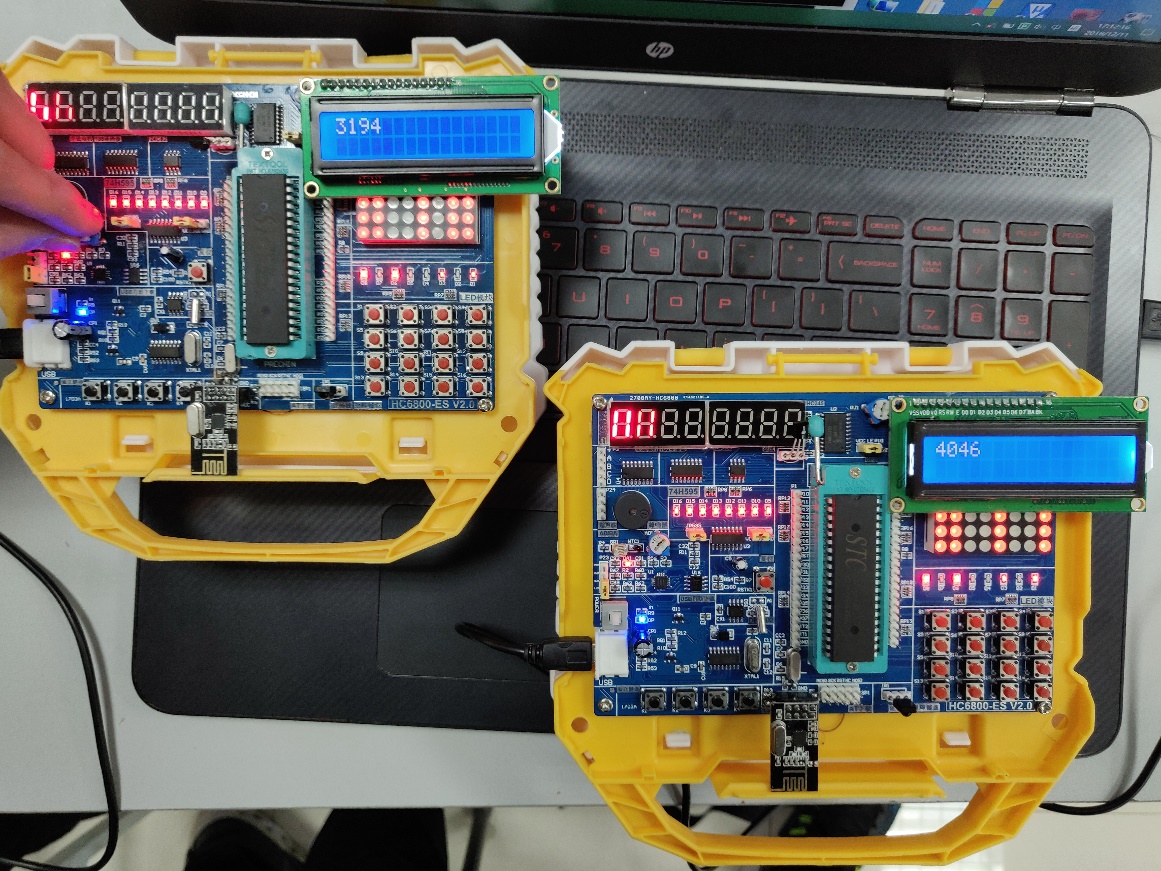
**}**

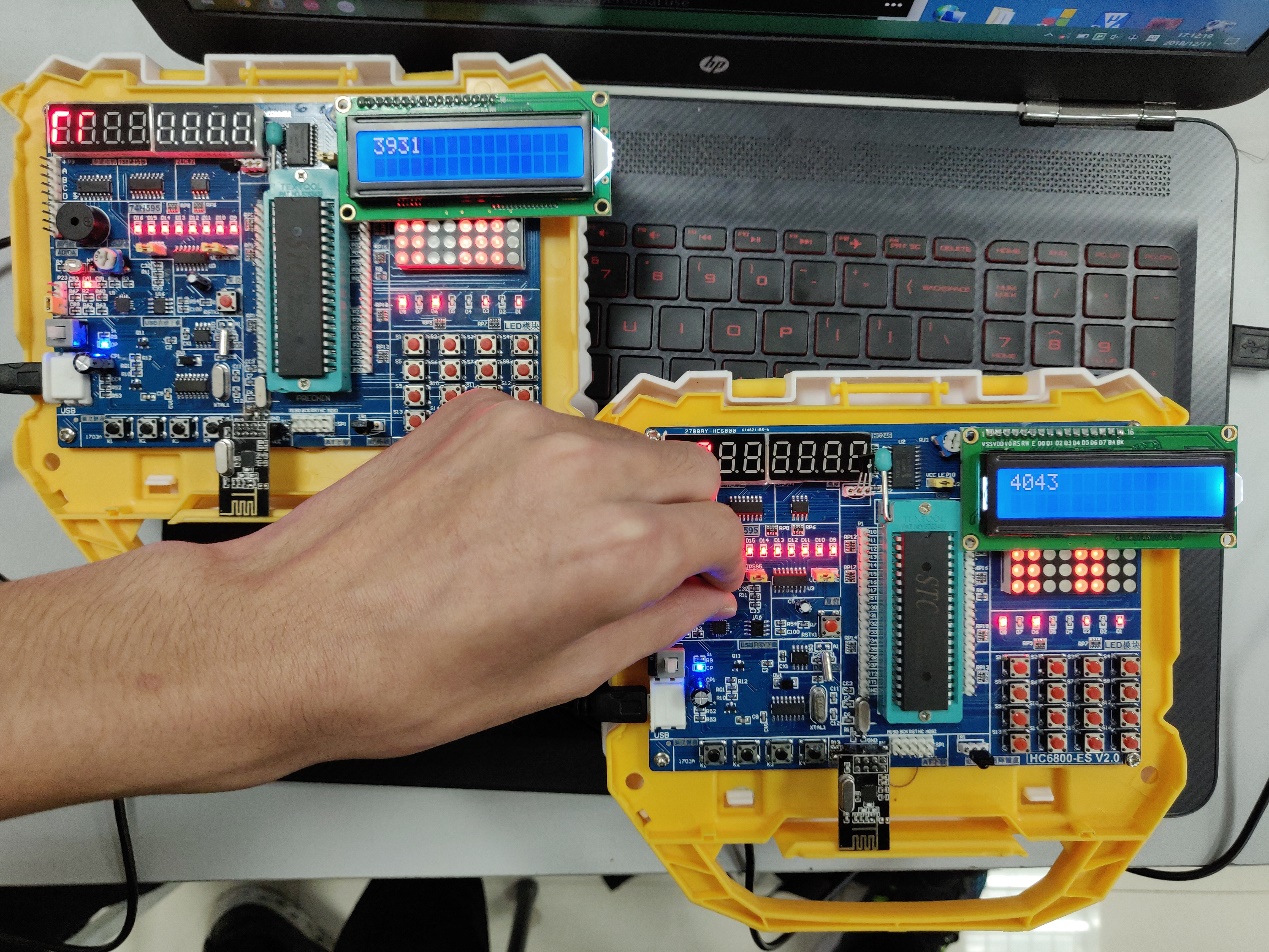
**}**

**}**

1. 调试







1. **实验总结**
2. 掌握了SPI总线的使用方式
3. 掌握了NRF24L01的工作原理和使用方式
4. 掌握了无线传输的基本原理