基于 Arduino 的红外解码器

163107110024 Jiaming Luo

摘要：本实验通过自学开源电子平台 Arduino UNO，尤其是其中的红外模块，学会了相关红外遥控电子器件的物理原理和通讯模式以及红外模块 IRremote 库和即使断电也能保存数据的 EEPROM（Electrically Erasbale Programmable Read-Only Memory）库的程序语法，设计了两个红外遥控实验。第一个是比较基础地利用红外模块控制 Arduino UNO 模块上的板载 LED 灯，之后涉及制作了一个较高难度的红外解码器。该款解码器可以实现接受，保存和再发射任何未知的红外信号的功能，可以在日常生活中遥控电器设备起着一定的辅助作用。

一，引言 [1,2]

人眼能看到的光如果按波长从长到短排列，依次是红、橙、黄、绿、青、蓝、紫。其中红光的波长范围为$0.62\backsim0.76\mu$m；紫光的波长范围为$0.38\backsim0.46\mu$m。而在自然界中，有一些非可见光的波长比紫光波长短，它们常常被称为紫外线。而红外线就是称呼那些波长比红光波长长的光。在光谱中波长自0.76至400微米的一段都可以称为红外线，所有高于绝对零度（-273.15℃）的物质都可以自发产生红外线，所以在现代物理学中，又可以称它为热射线。

红外有如下优点：成本低廉、功耗不高，连接方便、简单易用，点对点直线数据传输，保密性强且无辐射危害。所以在生活中应用非常的广泛。医用红外线主要使用近红外线与远红外线两类，在日常通信中，往往使用近红外线来作为传输信号的主要通信方式。该方式主要是用来取代点对点的线缆连接，和蓝牙、WiFi 等一样，都属于一种无线数据传输技术。由于不需要实体连线，简单易用且实现成本较低，因而广泛应用于小型移动设备互换数据和电器设备的控制中，例如笔记本电脑，移动手机之间或与电脑之间进行数据交换，电视机、空调器的遥控等。红外线遥控主要是利用波长为$0.76\backsim1.5\mu$m之间的近红外线来传送控制信号的。

不过硬币总有正反面，红外也有一些缺点。比如因为其点对点的传输连接，无法灵活地组成网络。再者，由于它的波长较短，对障碍物的衍射能力差，所以只适合于短距离无线通讯的场合。而且在发送数据时，输出的功率一定时，用于信号传输的功率小，接收到的数据的信噪比小，很容易数据误读。 而且红外使用调幅进行传输，所以抗外界干扰能力不强。

二，红外通信原理简介 [1,2]

红外通信是利用近红外波段的红外线作为传递信息的媒体，即通信信道。发送端将基带二进制信号调制为一系列的脉冲串信号，通过红外发射管发射红外信号。接收端将接收到的光脉转换成电信号，再经过放大、滤波等处理后送给解调电路进行解调，还原为二进制数字信号后输出。常用的有通过脉冲宽度来实现信号调制的脉宽调制（PWM）和通过脉冲串之间的时间间隔来实现信号调制的脉时调制（PPM）两种方法。 简而言之，红外通信的实质就是对二进制数字信号进行调制与解调，以便利用红外信道进行传输；红外通信接口就是针对红

外信道的调制解调器。

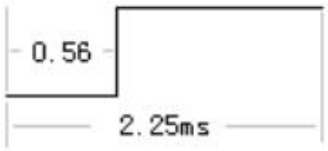
二进制数字信号如何进行调制与解调，利用何种红外信道进行传输，这就被称为红外协议。因为在后续解码器的设计中需要甄别不同的协议，所以下文将详细介绍几种常见的红外通信协议：

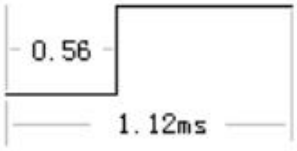
Ⅰ. NEC 协议：
基本特征：
（1）8 位地址位，8 位命令位
（2）为了可靠性地址位和命令位被传输两次
（3）脉冲位置调制
（4）载波频率 38kHz
（5）每一位的时间为1.125ms 或 2.25ms

逻辑 0 和 1 的定义：

逻辑 1 的是由 560μs 的高电平和 1.69ms 的低电平组成的脉冲表示。

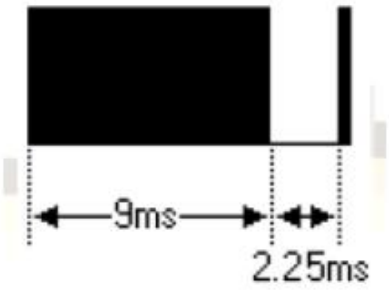
逻辑 0 的是由 560μs 的高电平和 565μs 的低电平组成的脉冲表示。





重复脉冲：

重复码的格式是由 9ms 的高电平和 2.25ms 的低电平及一个 560μs 的高电平组成。



发送格式如下：
NEC 协议中，首先是 9ms 的高电平脉冲，其后是 4.5ms 的低电平，接下来就是 8bit 的地址码（从低有效位开始发），而后是 8bit 的地址码的反码（主要是用于校验是否出错）。然后是 8bit 的命令码（也是从低有效位开始发），而后也是 8bit 的命令码的反码。

Ⅱ. Sony 协议：

有 12，15，20 位三种模式，本文主要介绍比较常见的 Sony 协议 12 位模式。

基本特征：

（1）5 位地址位，7 位命令位

（2）地址位和命令位被传输一次

（3）脉冲宽度编码

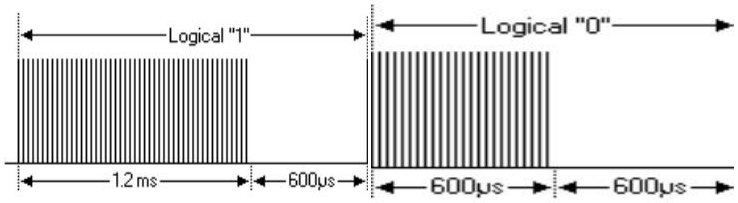
（4）载波频率 40kHz

（5）每一位的时间为 1.2ms 或 1.8ms

逻辑 0 和 1 的定义：

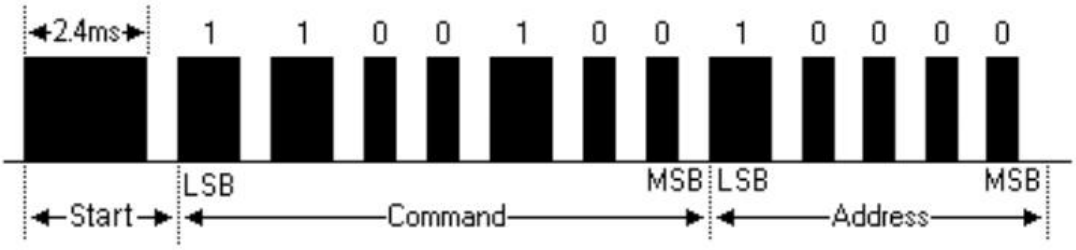
逻辑 1 的是由 1.2ms 的高电平和 600μs 的低电平组成的脉冲表示。

逻辑 0 的是由 600μs 的高电平和 600μs 的低电平组成的脉冲表示。



发送格式如下：

Sony 协议中，首先是 2.4ms 的高电平脉冲，其后是 0.6ms 的低电平，接下来就是 7bit 的命令码。然后是 5bit 的地址码。长按键时，数据每隔 45ms 重复发送一次。



Ⅲ. RC5 协议：

RC5 协议由 Philips 公司推出。它采用载波频率固定为 36kHz 的 ASK 调制和曼彻斯特编码。

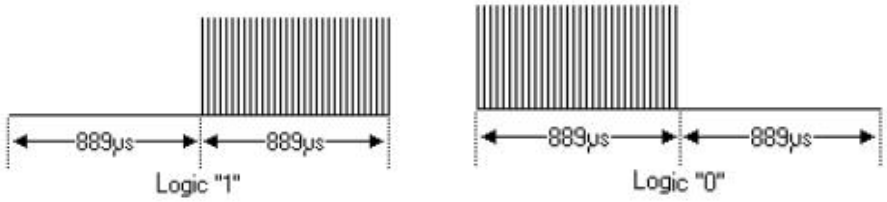
基本特征：

（1）4 位地址位，4 位命令位
（2）地址位和命令位被传输一次
（3）载波频率 36kHz
（4）每一位的时间为 1.05ms 或 2.1ms

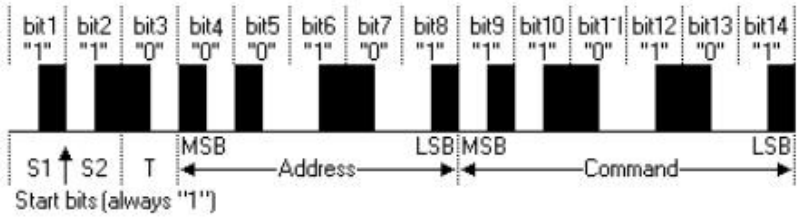
逻辑 0 和 1 的定义如：

逻辑 1 的是由 889μs 的低电平和 889μs 的高电平组成的脉冲表示。

逻辑 0 的是由 889μs 的高电平和 889μs 的低电平组成的脉冲表示。



发送格式如下：
RC5 协议中，首先是“110”的信号，接下来就是 5bit 的地址码。然后是 7bit 的命令码。

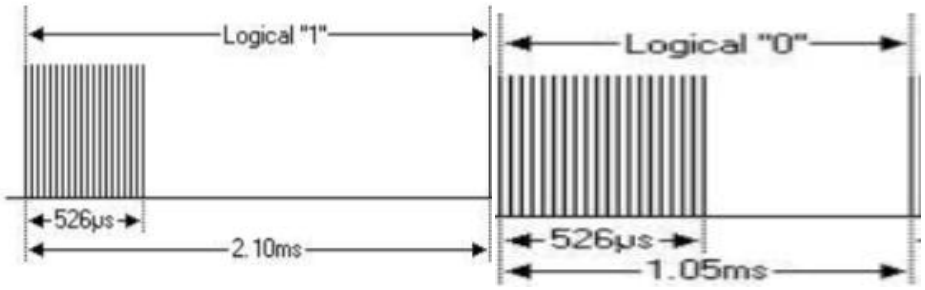


Ⅳ. JVC 协议：
基本特征：
（1）8 位地址位，8 位命令位
（2）地址位和命令位被传输一次
（3）脉冲位置调制
（4）载波频率 38kHz
（5）每一位的时间为 1.05ms 或 2.1ms

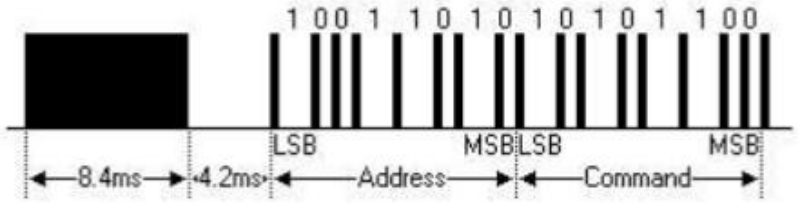
逻辑 0 和 1 的定义：

逻辑 1 的是由 526μs 的高电平和 1.574ms 的低电平组成的脉冲表示。

逻辑 0 的是由 526μs 的高电平和 0.524ms 的低电平组成的脉冲表示。



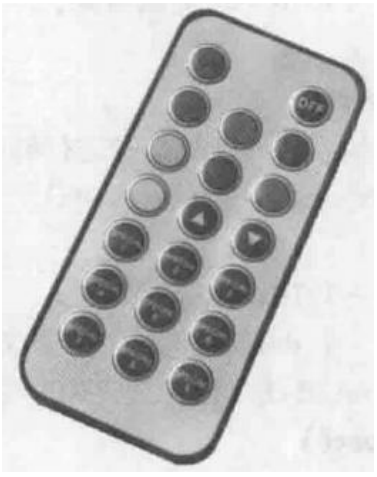
发送格式如下：
JVC 协议中，首先是 8.4ms 的高电平脉冲，其后是 4.2ms 的低电平，接下来就是 8bit 的地址码。然后是 8bit 的地址码。



三， IRremote 模块介绍

红外模块在硬件部分主要有红外遥控器，一体化红外接收头，红外发光二极管。

红外遥控器：
红外遥控器发出的信号是一连串的二进制脉冲码。为了使其在无线传输过程中免受其他红外信号的干扰,通常都是先将其调制在特定的载波频率上,然后再经红外发射二极管发射出去。

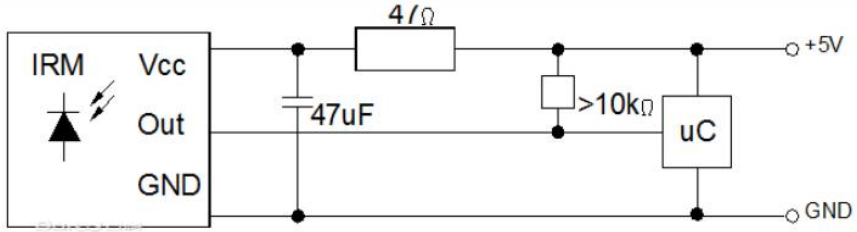


红外遥控器上每个按键都有独一无二的编码，按下按键之后，遥控器就会发送对应编码的红外波。

一体化红外接收头 [3]：
红外线接收装置要滤除干扰的杂波,接收特定频率的信号并将其还原成二进制脉冲码,也就是

解调。内置接收管将红外发射管发射出来的光信号转换为微弱的电信号，此信号经由 IC 内部放大器进行放大，然后通过自动增益控制、带通滤波、解调变、波形整形后还原为遥控器发射出的原始编码，经由接收头的信号输出脚输入到电器上的编码识别电路。

红外信号收发系统的典型电路如下图所示，红外接收电路通常被厂家集成在一个元件中，成为一体化红外接收头。 内部电路包括红外监测二极管，放大器，限幅器，带通滤波器，积分电路，比较器等。红外监测二极管监测到红外信号，然后把信号送到放大器和限幅器，限幅器把脉冲幅度控制在一定的水平，而不论红外发射器和接收器的距离远近。交流信号进入带通滤波器，带通滤波器可以通过 30kHz 到 60kHz 的负载波，通过解调电路和积分电路进入比较器，比较器输出高低电平，还原出发射端的信号波形。注意输出的高低电平和发射端是反相的，这样的目的是为了提高接收的灵敏度。



红外接收头的种类很多，引脚定义也不相同，一般都有三个引脚，包括供电脚，接地和信号输出脚。

红外发光二极管：

虽然红外发光二极管的外形和使用方法与普通发光二极管相似，但是它可以发出肉眼看不见的红外光。与红外一体接收管搭配组合使用，就可以进行最基本的红外通信了。

如果想使用红外遥控功能，不仅需要红外硬件三件套，还需要用到第三方的红外遥控库 IRremote 和即使断电也能保存数据的 EEPROM（Electrically Erasbale Programmable Read-Only Memory）库。

红外库主要负责接收和解码红外消息 IRrecv 和发射红外信号的 IRsend,其成员函数有：

指定红外一体化接收头的连接引脚：

IRrecvobject(recvpin)

初始化红外解码：

IRrecv.enableIRIn ()

对接收到的红外信号进行解码:

IRrecv.decode(&results)
接受下一个编码：
IRrecv.resume()
以 NEC 协议发送特定值：
IRsend.sendNEC(data,nbits)
发送原始红外编码信号, buf 表示存储原始编码的数组，len 表示数组长度，hz 表示红外发射频率：
IRsned.sendRaw(buf,len,hz)

EEPROM 库主要使用三种语法。
第一个是将数值 value 写入 EEPROM 的地址 address 中的语法：
EEPROM.write(address, value);
第二个是读出并利用 EEPROM 的地址 address，返回值便是读出的数据的语法：
EEPROM.read(address);
第三个则是清除 EEPROM 的内容，其实就是把每个 EEPROM 中的每个字节写入 0：

```
for(int i = 0; I < 512; i++);
   EEPROM.write( i, 0)
```

四，实验 1： 红外模块遥控板载 LED

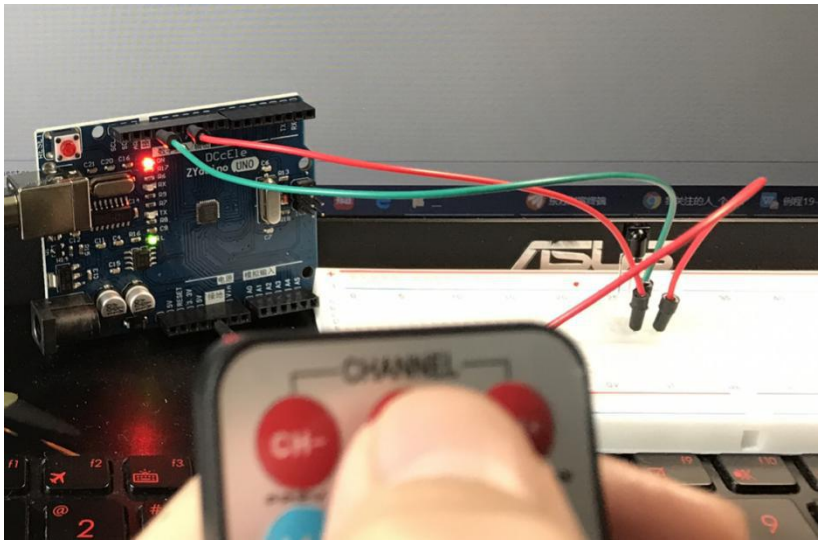
通过前面对 Arduino UNO 红外模块的自我学习和了解，化书本知识为基础实践，来完成一个利用红外遥控器控制 Uno 板载 LED 灯亮灭的基本实验。

红外遥控板载 LED 编码思路：

程序设计想要达成的目标是按下程序中指定的按键，控制 arduino 板载 LED 灯亮灭。

连接电路时，板载 LED 自动灯亮起。按下“CH-”时，收到信号“FFA25D”，则将板载 LED 灯周围的电压变低，从而使得其熄灭，且。按下“CH”时，收到信号“FF629D”板载 LED 灯周围的电压变高，从而使得其亮起。按下遥控器其他按键时板载 LED 灯明灭无变化。

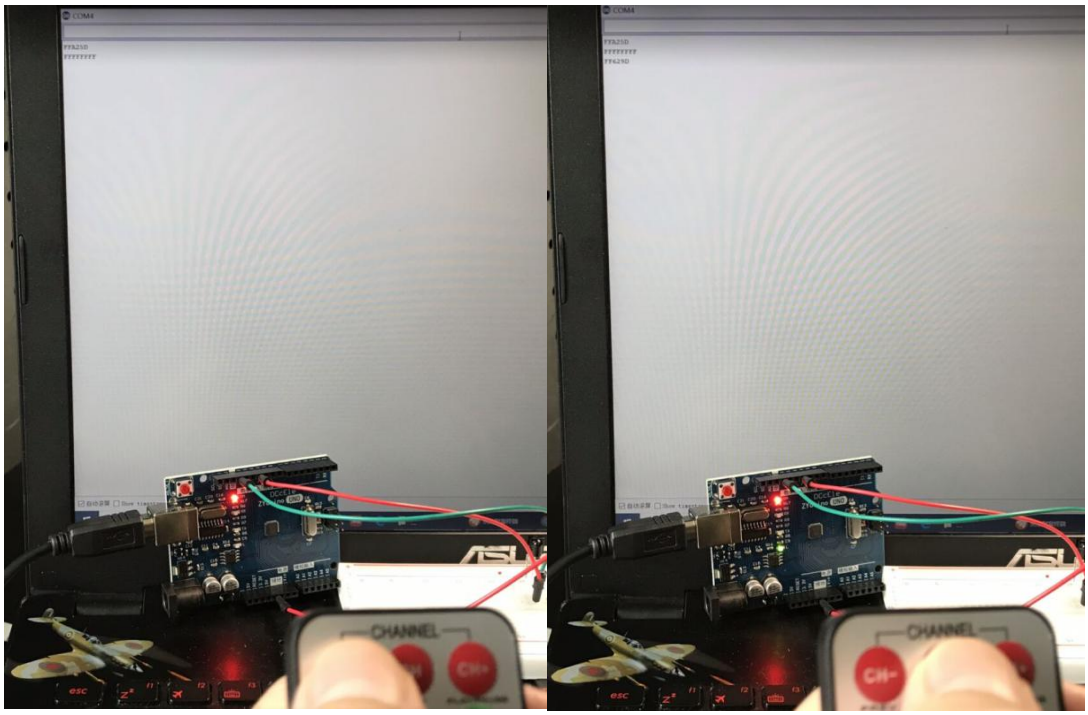
红外遥控板载 LED 实物图：



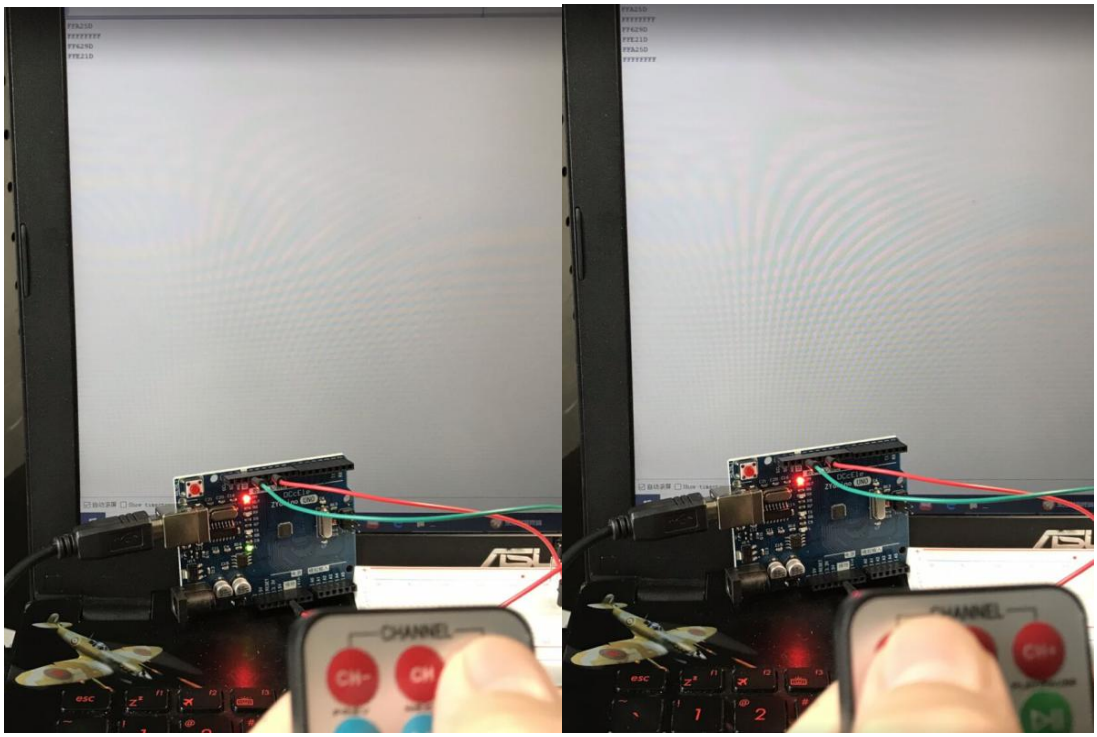
所需实验器材：一体化红外接收头，Arduino UNO，红外遥控器，导线若干。

红外遥控板载 LED 测试图：

1. 首先正常连接电路，插上电源后板载 LED 灯亮起。对着红外接收头按下按键“CH-”后，板载 LED 灯熄灭，且串口显示器显示收到信号“FFA25D”。再对着红外接收头按下按键“CH”后，板载 LED 灯亮起，且串口显示器收到信号“FF629D”。



2. 对着红外接收头按下按键“CH+”后，板载 LED 灯没有任何明灭变化，且串口显示器显示收到信号“FFE21D”。对着红外接收头按下按键“CH-”后，板载 LED 灯熄灭，且串口显示器显示收到信号“FFA25D”，说明实验基本成功。



五，实验 2： 制作 Arduino 红外解码器

Arduino 红外解码器设计目的：

基于 Arduino Uno 接收外部发射的红外遥控信号，在 EEPROM 中记录储存下这一信号，通过按键开关，红外 LED 会将这一段红外遥控信号再发射出来，仿佛解码重放了这一段红外遥控信号。通过这个小 Project， 用户可以用 Arduino Uno 来控制家中的电器，例如空调，电视机等，且因为储存在 EEPROM 的缘故，即使断电了也不会影响已储存的信号的发射。

Arduino 红外解码器编码思路：

总结一下红外解码器的设计目的，其实就算接收信号，储存信号，发射信号三部分。

在写主体程序之前，先完成一些准备工作：创建存放红外信号编码类型 EEPROM 地址，编码长度 EEPROM 地址，RC5/RC6 类型的 EEPROM 地址，信号数值的 EEPROM 地址。定义红外接收器和按键开关的引脚，并检查用户是否按下了按键开关。

先启动红外接收，识别接收到的红外信号的编码类型，如果不是 raw 型，那么存放红外信号数值和编码长度。如果收到的信号是无法识别的协议，则存储为 raw 型数据。

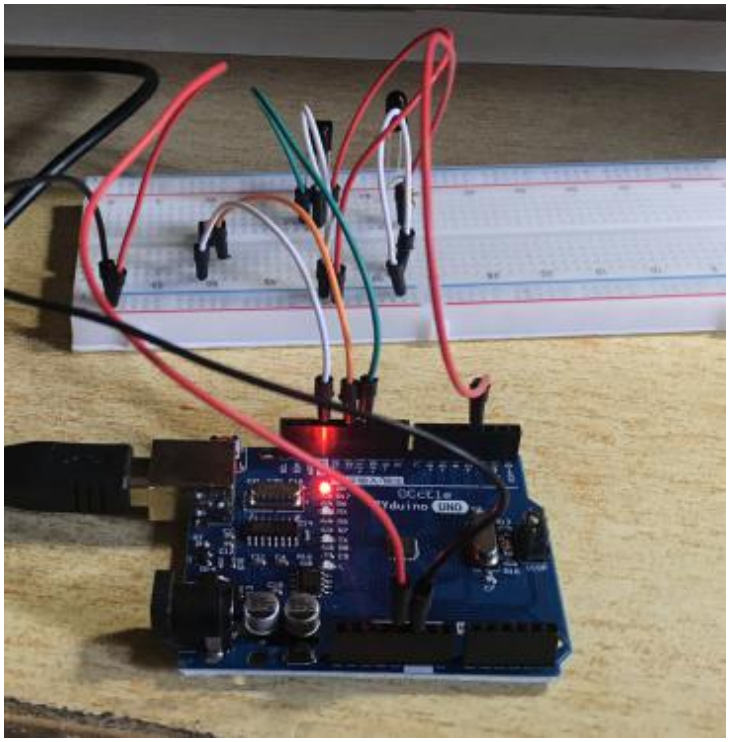
然后再按照 NEC 协议类型，RC5 协议类型，RC6 协议类型，PANASONIC 协议类型，JVC 协议

类型进行分类。在串口监视器中输出信号数值，并将收到的信号信息储存于 EEPROM。

按下按键开关后，根据之前不同协议的分类进行筛选，这是因为不同协议发送的格式不同，然后再发送接收到的红外信号，与此同时串口监视器输出信号数值，以方便和接收的数值进行比对。

Arduino 红外解码器器件实物图：

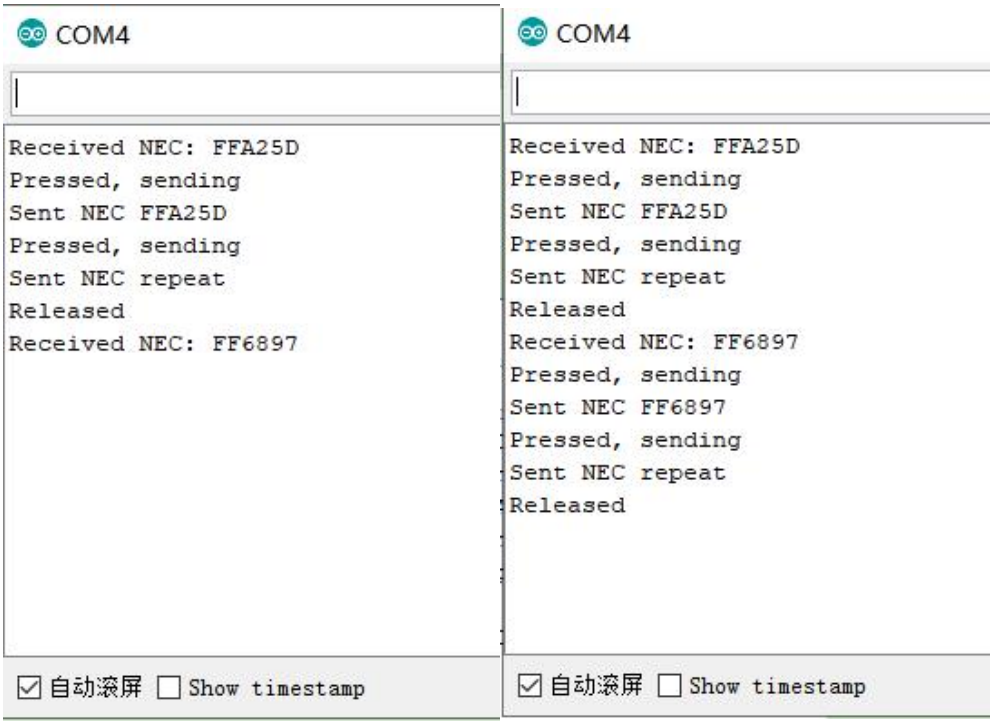
所需实验器材：红外发射 LED，一体化红外接收头，Arduino UNO， 红外遥控器，按键开关，电阻，导线若干。



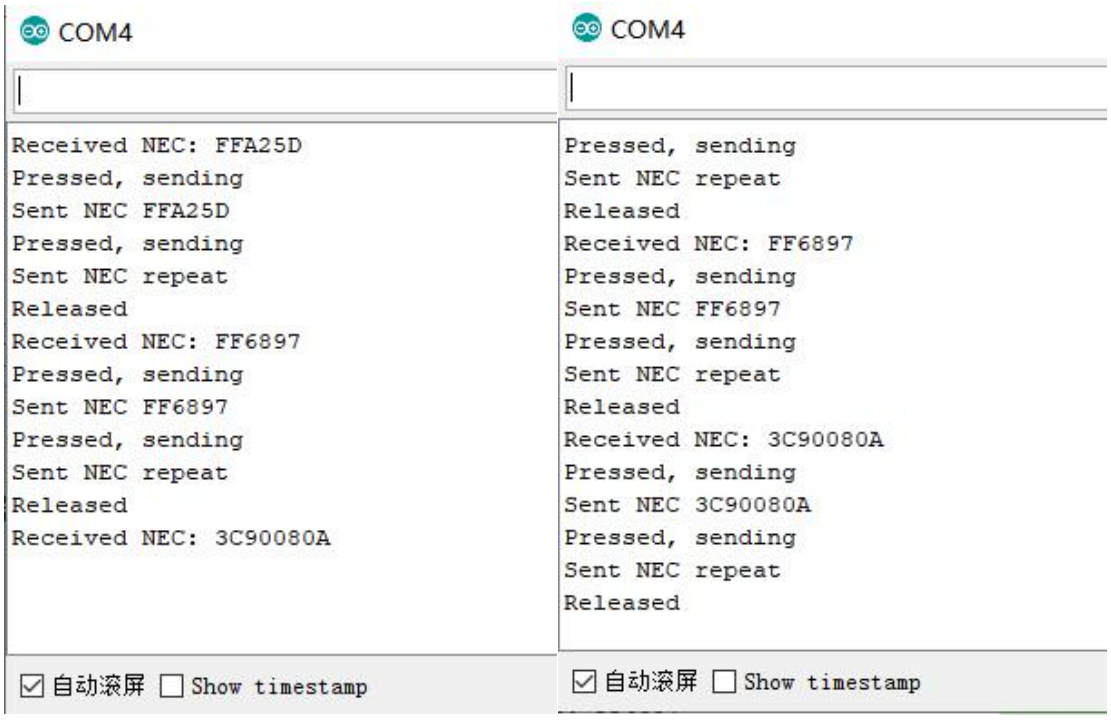
Arduino 红外解码器测试图：

1. 首先对准红外解码器按下红外遥控器的“CH-”按钮，在串口监视器中显示“FFA25D”，说明红外解码器收到了“CH-”的信号。之后按下按键开关，串口监视器收到了依托 NEC 协议的“CH-”红外信号，说明了红外解码器保存了“CH-”信号，识别了“CH-”信号和发送了“CH-”信号。

2. 对准红外解码器按下红外遥控器的“0”按钮，在串口监视器中显示“FF6897”，说明红外解码器收到了“0”的信号。之后按下按键开关，串口监视器收到了依托 NEC 协议的“0”红外信号，说明了红外解码器保存了“0”信号，识别了“0”信号和发送了“0”信号。通过上面两次实验，初步判定红外解码器设计成功。



3. 最后，脱离红外模块附带的遥控器，使用寝室内的空调遥控器。对准红外解码器按下空调遥控器的电源开关按钮，在串口监视器中显示“3C90080A”，说明红外解码器收到了空调开关的信号。之后按下按键开关，串口监视器收到了依托 NEC 协议的“3C90080A”红外信号，而且寝室空调正常打开，说明了红外解码器保存了空调遥控器信号，识别了空调遥控器信号和发送了空调遥控器信号，实验成功。



实验结论

本次实验学习了如何使用开源平台 Arduino UNO，尤其是仔细研究了 Arduino UNO 的红外模块，基于此模块，设计了两个循序渐进的红外遥控实验。第一个实验是利用红外模块去遥控 Arduino UNO 上的板载 LED 灯，操作和设计都源于基础的红外遥控知识。第二个实验，则是综合了红外通信和遥控两大主题，设计制作了一个功能比较丰富的红外解码器。该期间可以实现接受，保存和再发射任何未知的红外信号的功能，可以替代生活中的电视遥控器，空调遥控器等。通过这次实验，不仅提升了自学新科技的能力，如编程能力和电路的动手能力，也为枯燥的宅家生活增添了许多乐趣。

参考文献

1. Arduino 程序设计基础（第 2 版）

2. https://baike.baidu.com/item/%E7%BA%A2%E5%A4%96/1903975

3. https://baike.baidu.com/item/%E7%BA%A2%E5%A4%96%E6%8E%A5%E6%94%B6%E5%A4%B4/9569261?fr=aladdin

附录：

红外遥控 LED 代码：

```
#include<IRremote.h>

int RECV_PIN = 11;
int LED_PIN = 13;
int a=0;

IRrecv irrecv(RECV_PIN);
decode_results results;

void setup() {
  Serial.begin(9600);
  irrecv.enableIRIn();
  pinMode(LED_PIN,OUTPUT);
  digitalWrite(LED_PIN,HIGH);
}

void loop() {
  if(irrecv.decode(&results))
  {
    if(results.value == 0xFFA25D & a == 0)
    {
      digitalWrite(LED_PIN,HIGH);
      a = 1;
    } else if (results.value == 0xFF629D & a == 1)
    {
      digitalWrite(LED_PIN,LOW);
      a = 0;
    }
    irrecv.resume();
  }
  delay(100);
}
```

红外解码器代码：

```
#define codeTypeEAddr         0
#define codeLenEAddr          1
#define toggleEAddr           2
#define codeValueEAddr        3
#define RECV_PIN 11
#define STATUS_PIN LED_BUILTIN
#define BUTTON_PIN 12
#include <EEPROM.h>
#include <IRremote.h>

IRrecv irrecv(RECV_PIN);
IRsend irsend;
decode_results results;

void setup() {
   Serial.begin(9600);
   irrecv.enableIRIn();
   pinMode(STATUS_PIN, OUTPUT);
   pinMode(BUTTON_PIN, INPUT_PULLUP);
   delay(30);
   loadEepromValues();
}

unsigned int rawCodes[RAWBUF];
unsigned long codeValue;
int codeLen;
int codeType;
int toggle;

void storeCode(decode_results *results) {
   codeType = results->decode_type;
   int count = results->rawlen;
   if (codeType == UNKNOWN) {
      Serial.println("Received unknown code, saving as raw");
      codeLen = results->rawlen - 1;
      for (int i = 1; i <= codeLen; i++) {
         if (i % 2) {
            rawCodes[i - 1] = results->rawbuf[i]*USECPERTICK - MARK_EXCESS;
            Serial.print(" m");
         }
         else {
            rawCodes[i - 1] = results->rawbuf[i]*USECPERTICK + MARK_EXCESS;
            Serial.print(" s");
```

```
            }
            Serial.print(rawCodes[i - 1], DEC);
        }
        Serial.println("");
    }
    else {
        if (codeType == NEC) {
            Serial.print("Received NEC: ");
            if (results->value == REPEAT) {
                Serial.println("repeat; ignoring.");
                return;
            }
        }
        else if (codeType == JVC) {
            Serial.print("Received JVC: ");
        }
        else if (codeType == PANASONIC) {
            Serial.print("Received PANASONIC: ");
        }
        else if (codeType == RC5) {
            Serial.print("Received RC5: ");
        }
        else if (codeType == RC6) {
            Serial.print("Received RC6: ");
        }
        else if (codeType == SONY) {
            Serial.print("Received SONY: ");
        }
        else {
            Serial.print("Unknown codeType ");
            Serial.print(codeType, DEC);
            Serial.println("");
        }
        Serial.println(results->value, HEX);
        codeValue = results->value;
        codeLen = results->bits;
    }
    writeEepromVal();
}
```

发送部分版权原因不予显示 ：）

```
int lastButtonState;
```

```
void loop() {
   int buttonState = !digitalRead(BUTTON_PIN);
   if (lastButtonState == HIGH && buttonState == LOW) {
      Serial.println("Released");
      irrecv.enableIRIn();
   }

   if (buttonState) {
      Serial.println("Pressed, sending");
      digitalWrite(STATUS_PIN, HIGH);
      sendCode(lastButtonState == buttonState);
      digitalWrite(STATUS_PIN, LOW);
      delay(50);
   } else if (irrecv.decode(&results)) {
      digitalWrite(STATUS_PIN, HIGH);
      storeCode(&results);
      irrecv.resume();
      digitalWrite(STATUS_PIN, LOW);
   }
   lastButtonState = buttonState;
}

void loadEepromValues(){
   codeType = EEPROM.read(codeTypeEAddr);
   delay(20);
   codeLen = EEPROM.read(codeLenEAddr);
   delay(20);
   toggle = EEPROM.read(toggleEAddr);
   delay(20);
   toggle = EEPROM.read(toggleEAddr);
   delay(20);
   EEPROM.get(codeValueEAddr, codeValue);
}

void writeEepromVal(){
     EEPROM.write(codeTypeEAddr, codeType);
     delay(20);
     EEPROM.write(toggleEAddr, toggleEAddr);
     delay(20);
     EEPROM.write(codeLenEAddr, codeLen);
     delay(20);
     EEPROM.put(codeValueEAddr, codeValue);
     delay(20);
}
```