Arduino 红外遥控模式

本实验目标：学习使用红外遥控与红外接收，并依此实现远程控制

目标分析：为了完成这个实验，我们首先对于它进行架构的分析

传感器：红外遥控/红外接收
控制器：UNO/ESP32
执行器：一位数码管，LED灯

接着是效果分析：

通过红外遥控，可实现远程控制元件

明确目标系统与效果，开始正式学习---

程序架构说明

以下两个程序需分别烧录到Arduino开发板运行：

程序一：红外信号分析器

功能描述

实时捕获并解析红外遥控信号，通过串口显示协议类型和原始数据。

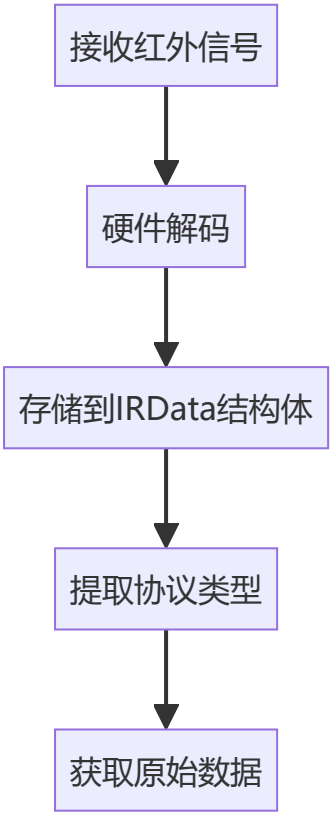
完整代码解析

```
#include <IRremote.h> // 引入红外遥控库

// 协议类型名称数组（26种协议）
const String IR_PROTOCOL_TYPE[] = {
  "UNKNOWN", "PULSE_DISTANCE", ..., "WHYNTER"
};

IRrecv irrecv_2(2); // 创建红外接收对象（接D2）

void setup(){
  Serial.begin(9600);       // 初始化串口通信
  irrecv_2.enableIRIn();    // 启动红外接收
}

void loop(){
  if (irrecv_2.decode()) {  // 检测是否接收到数据
    // 获取解码数据指针
    struct IRData *pIrData = &irrecv_2.decodedIRData;

    // 提取原始数据（32位十六进制）
    long ir_item = pIrData->decodedRawData;

    // 获取协议类型名称
    String irProtocol = IR_PROTOCOL_TYPE[pIrData->protocol];

    // 串口输出协议类型和数值
    Serial.print("IR TYPE:" + irProtocol + "\tVALUE:");
```

```
    Serial.println(ir_item, HEX); // HEX表示十六进制

    irrecv_2.resume(); // 准备接收下一个信号
  }
}
```

关键技术解析

1. **红外信号解码流程**



2. **IRData结构体成员**

成员变量	说明
protocol	协议类型索引（对应数组）
decodedRawData	原始32位数据
address	设备地址码
command	指令码

程序二：红外遥控控制器

功能描述

通过特定红外指令控制三个LED状态（D6/D7/D8引脚），实现：

- 0xBA45FF00：切换D7 LED
- 0xB946FF00：切换D6 LED
- 0xB847FF00：切换D8 LED

- 其他指令：关闭所有LED

关键代码解析

```
void setup(){
  // 新增引脚配置
  pinMode(7, OUTPUT); // LED1控制引脚
  pinMode(6, OUTPUT); // LED2控制引脚
  pinMode(8, OUTPUT); // 蜂鸣器控制引脚
}

void loop(){
  if (irrecv_2.decode()) {
    // ...数据解析部分与程序一相同

    // 指令判断与执行
    switch (ir_item) {
     case 0xBA45FF00: // 按钮1编码
      digitalWrite(7, !digitalRead(7)); // 翻转D7状态
      break;
     case 0xB946FF00: // 按钮2编码
      digitalWrite(6, !digitalRead(6)); // 翻转D6状态
      break;
     case 0xB847FF00: // 按钮3编码
      digitalWrite(8, !digitalRead(8)); // 翻转D8状态
      break;
     default:          // 其他指令
      // 关闭所有设备
      digitalWrite(6, LOW);
      digitalWrite(7, LOW);
      digitalWrite(8, LOW);
      break;
    }

    irrecv_2.resume(); // 继续接收
  }
}
```

核心概念详解

1. **十六进制数值表示**
   - `0x` 前缀表示十六进制数
   - 示例：`0xBA45FF00` 对应二进制 `10111010010001011111111000000000`
2. **状态翻转逻辑**

   ```
   digitalWrite(pin, !digitalRead(pin));
   ```

   - `digitalRead()` 读取当前状态
   - `!` 运算符进行逻辑取反
   - `digitalWrite()` 写入新状态
3. **switch-case结构**

```
switch (判断值) {
  case 值1: 操作1; break;
  case 值2: 操作2; break;
  default: 默认操作;
}
```

- `break`关键字用于退出判断结构
- `default`处理未匹配的情况

硬件系统设计

电路连接规范

```
红外接收模块：
  VCC → 5V
  GND → GND
  OUT → D2

LED电路：
  D6/D7/D8 → 220Ω电阻 → LED正极
  LED负极 → GND

蜂鸣器电路：
  D8 → 1kΩ电阻 → 蜂鸣器+
  蜂鸣器- → GND
```

常见红外编码示例

按键	典型编码（NEC协议）
电源	0xBA45FF00
音量+	0xB946FF00
音量-	0xB847FF00

调试与优化

红外信号捕获流程

1. 运行程序一，打开串口监视器
2. 对准接收器按下遥控按键
3. 记录显示的十六进制编码
4. 将目标按键编码填入程序二的case语句

常见问题解决

现象	解决方案
无任何响应	检查接收模块是否接反
编码显示UNKNOWN	确认库文件支持该协议
LED状态异常	检查引脚接线和逻辑取反操作
响应延迟	增加 `irrecv_2.resume()` 频率
十六进制识别	在编码信号的前端带上0x字符

扩展应用

多设备控制

```
case 0xBB44FF00: // 组合控制指令
  digitalWrite(6, HIGH);
  tone(8, 1000, 500); // 蜂鸣器响0.5秒
  break;
```

信号转发

```
IRsend irsend;
case 0xA15EFF00:
  irsend.sendNEC(0xA15EFF00, 32); // 转发原始信号
  break;
```

状态记忆

```
bool ledState = LOW;
case 0xBA45FF00:
  ledState = !ledState;
  digitalWrite(7, ledState);
  break;
```

核心概念总结

1. **红外通信原理**
   - 载波频率：38kHz（常见）
   - 编码方式：PWM/PPM
   - 传输距离：典型3-5米
2. **数字信号处理**
   - 高低电平表示二进制数据
   - 脉宽编码携带信息

3. **协议差异**

协议	特点
NEC	32位数据，重复码机制
RC5	14位数据，双相位编码
SONY	12-20位数据，脉宽调制

红外遥控的使用非常广泛，通过这里的学习可以直接制作一个控制程序来实现家里的电视/空调智能控制，搭配物联网可以实现智能家居远程控制等。

有问题找柯萌ଘ(੭ˊ~>ˇ<)✧
邮箱mingyoufhh@outlook.com