Arduino 超声波雷达（HC-SR04）程序详解

本实验目标：通过超声波雷达对环境的探测实现一些功能

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：超声波雷达
控制器：UNO/ESP32
执行器：蜂鸣器/串口

接着是效果分析：

超声波雷达会进行正前方物品的测距，通过测距距离进行一些功能显示

明确目标系统与效果，开始正式学习---

程序架构说明

以下三个程序需分别烧录至Arduino开发板运行，实现不同超声波测距功能：

程序一：近接报警器

功能描述

当检测到障碍物距离小于30cm时点亮LED，否则熄灭。

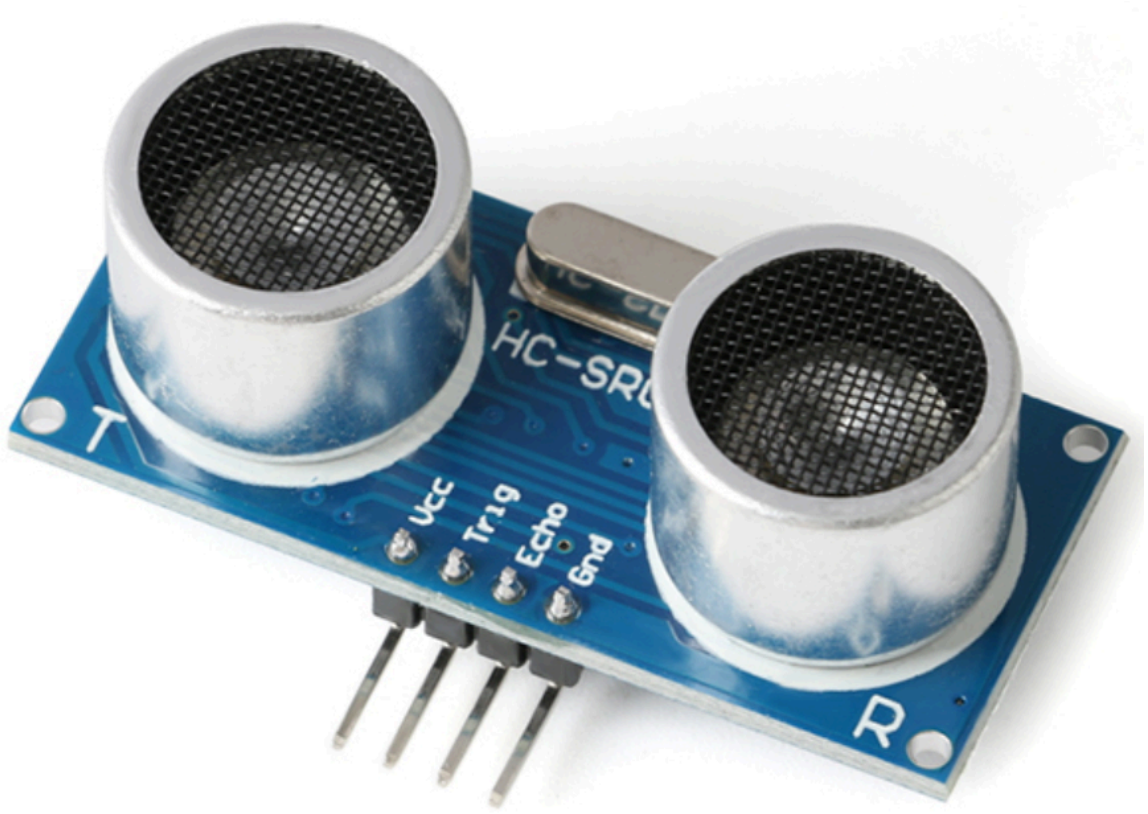
完整代码解析

```
// 超声波测距函数
float checkdistance_7_6() {
  digitalWrite(7, LOW);         // 确保Trig引脚初始低电平
  delayMicroseconds(2);         // 稳定电平（2μs）
  digitalWrite(7, HIGH);        // 发送10μs高电平脉冲
  delayMicroseconds(10);        // 保持高电平时间
  digitalWrite(7, LOW);         // 结束触发信号
  float distance = pulseIn(6, HIGH) / 58.00; // 计算距离
  delay(10);                    // 防止信号干扰
  return distance;
}

void setup(){
  pinMode(7, OUTPUT); // Trig引脚输出模式
  pinMode(6, INPUT);  // Echo引脚输入模式
  pinMode(8, OUTPUT); // LED控制引脚
}

void loop(){
  float currentDistance = checkdistance_7_6();
  if (currentDistance < 30) {  // 距离阈值判断
    digitalWrite(8, HIGH);     // 点亮LED
  } else {
    digitalWrite(8, LOW);      // 熄灭LED
  }
}
```

关键技术解析



1. **超声波测距原理**

```
工作流程：
1. Trig引脚发送10μs高电平脉冲
2. 模块自动发射8个40kHz超声波
3. 接收回波后Echo引脚输出高电平
4. 高电平持续时间与距离成正比
计算公式：距离(cm) = 高电平时间(μs)/58
```

2. **pulseIn()函数**
   - 功能：测量引脚高/低电平持续时间
   - 参数：`pulseIn(pin, state, timeout)`
   - 返回值：微秒级时间值（默认超时1秒）

程序二：串口距离显示器

功能描述

通过串口实时输出超声波测距数据。

关键代码解析

```
void setup(){
  Serial.begin(9600);           // 初始化串口通信
  // 引脚模式配置同上
}

void loop(){
  Serial.println(checkdistance_7_6()); // 发送距离数据
  delay(100);                    // 控制数据刷新率
}
```

串口通信参数

参数	值	说明
波特率	9600	数据每秒传输9600比特
数据位	8	每个字节8位
停止位	1	数据包结束标志
校验位	无	未启用校验机制

程序三：倒车频率雷达

功能描述

障碍物越近，LED闪烁频率越高（最大响应距离50cm）。

优化代码解析

```
void loop(){
  float distance = checkdistance_7_6();
  if (distance < 50) {
    int interval = abs(distance * 15); // 防止负值
    digitalWrite(8, HIGH);
    delay(interval);       // 亮灯时间与距离成正比
    digitalWrite(8, LOW);
    delay(100);            // 固定灭灯时间
  } else {
    digitalWrite(8, LOW); // 超出范围熄灭
  }
}
```

距离-频率关系

```
距离(d) → 时间间隔(t)
t = d * 15 + 100 (ms)
示例：
当d=10cm → t=10*15+100=250ms → 频率=2Hz
当d=40cm → t=40*15+100=700ms → 频率≈1.17Hz
```

硬件系统设计

电路连接规范

```
超声波模块：
  VCC → 5V
  Trig → D7
  Echo → D6
  GND → GND

LED电路：
  D8 → 220Ω电阻 → LED正极
  LED负极 → GND
```

元件参数

元件	规格	备注
HC-SR04模块	工作电压5V	测量范围2cm-400cm
电阻	220Ω 1/4W	限流保护LED
Arduino开发板	UNO R3	需5V PWM引脚支持

关键问题解析

Q1：测量数据不准确

- **校准方法**：

```
// 在标准距离下修正除数
float actualDistance = 50.0; // 实际测量距离
float measuredTime = pulseIn(6, HIGH);
float calibrationFactor = measuredTime / actualDistance;
```

Q2：信号干扰处理

```
// 添加中值滤波
float stableDistance() {
  float samples[5];
  for(int i=0; i<5; i++){
    samples[i] = checkdistance_7_6();
    delay(20);
  }
  // 排序取中间值
  sortArray(samples, 5);
  return samples[2];
}
```

Q3：多设备协同

```
// 同时控制蜂鸣器
void loop(){
  float d = checkdistance_7_6();
  if(d < 30){
    tone(9, 1000 + (30-d)*100); // 频率随距离变化
    digitalWrite(8, HIGH);
  } else {
    noTone(9);
    digitalWrite(8, LOW);
  }
}
```

系统优化建议

1. 安全保护机制

```
void checkdistance_7_6() {
  // 添加超时判断
  unsigned long timeout = micros();
  while(digitalRead(6)==LOW){
    if(micros()-timeout > 30000) return 999; // 30ms超时
  }
  // ...原测量逻辑
}
```

2. 可视化界面

```
// 发送结构化数据
Serial.print("DIST:");
Serial.print(distance);
Serial.print("CM");
Serial.print("|STAT:");
Serial.println(distance<30?"DANGER":"SAFE");
```

3. 低功耗模式

```
void loop(){
  if(checkdistance_7_6() < 100){ // 激活检测范围
    // 执行正常操作
  } else {//在范围外时
    delay(1000); // 进入节能模式
  }
}
```

核心概念总结

1. **数字信号时序控制**
   - 精确控制Trig引脚10μs触发脉冲
   - 通过`delayMicroseconds()`实现微秒级延时
2. **模拟信号采集**
   - 使用`pulseIn()`捕获Echo高电平时间
   - 时间分辨率达1μs（理论精度0.017cm）
3. **距离映射算法**
   - 基础公式：距离 = 时间 / 58
   - 声速补偿：343m/s（25℃空气）
4. **安全阈值设定**
   - 根据应用场景设定不同响应阈值
   - 工业检测：10-50cm
   - 倒车雷达：50-150cm

有问题找柯萌•ᴥ•
邮箱mingyoufhh@outlook.com