《 智能系统 》实验报告

实验题目	数据采集与通信

一、实验目的

为实现十字路口红绿灯智能控制，本次实验的目的是：

（1）了解传感器与下位机

（2）设计并实现传感器连接与设计采集

（3）设计并实现上位机与下位机通信的数据包与解析

（4）设计并实现下位机与上位机通信

二、实验项目内容

1、传感器-下位机-上位机的连接

（1）了解所用传感器的原理；

（2）设计传感器与下位机连接方案，给出方案说明（文字与图表），给出实物连接图表；

（3）了解下位机与上位机通信协议，设计下位机与上位机连接方案，给出实物连接图表。

2、下位机数据采集

（1）设计传感器数据采集方案；

（2）设计并实现数据采集程序（函数）。

3、数据编码与传输

（1）下位机到上位机传输数据包设计；

（2）在下位机设计并实现数据包编码与传输程序（函数）；

（3）上位机到下位机传输数据包设计；

（4）在上位机设计并实现数据包编码与传输程序（函数）。

4、数据解析与输出

（1）设计并实现上位机接收数据包的解析与展示程序；

（2）设计并实现下位机接收数据包的解析程序；

（3）设计信号灯显示方案；

（4）设计并实现下位机控制信号灯显示程序。

三、实验过程或算法（代码）

1、传感器-下位机-上位机的连接

报告创建时间：

（1）了解所用传感器的原理

本次实验主要涉及两类传感器模块，一类是交通灯模块，一类是红外避障模块。

交通灯模块：

如实物图所示，交通灯模块拥有红、黄、绿三种颜色的灯，将模块的GND引脚与Arduino开发板上的GND引脚相连，模块的R、Y、G引脚与Arduino开发板上的数字供电引脚相连，即可在Arduino程序中控制对应引脚输出电平的高低控制灯的开关。

红外避障模块：

红外避障模块其具有一对红外线发射与接收管，发射管发射出一定频率的红外线，当检测方向遇到障碍物（反射面）时，红外线反射回来被接收管接收，经过比较器电路处理之后，绿色指示灯会亮起，同时信号输出接口（OUT）输出数字信号（一个低电平信号）。

（2）传感器与下位机连接方案

从（1）中传感器的原理，了解到交通灯模块与红外避障模块均需要从下位机取电，其中交通灯模块需要数字供电信号，以便在Arduino程序中控制灯的开关。红外避障模块还需将障碍物检测的结果通过OUT引脚传输至下位机。结合Arduino官方提供的Arduino UNO R3引脚定义，得出如下连接方案：

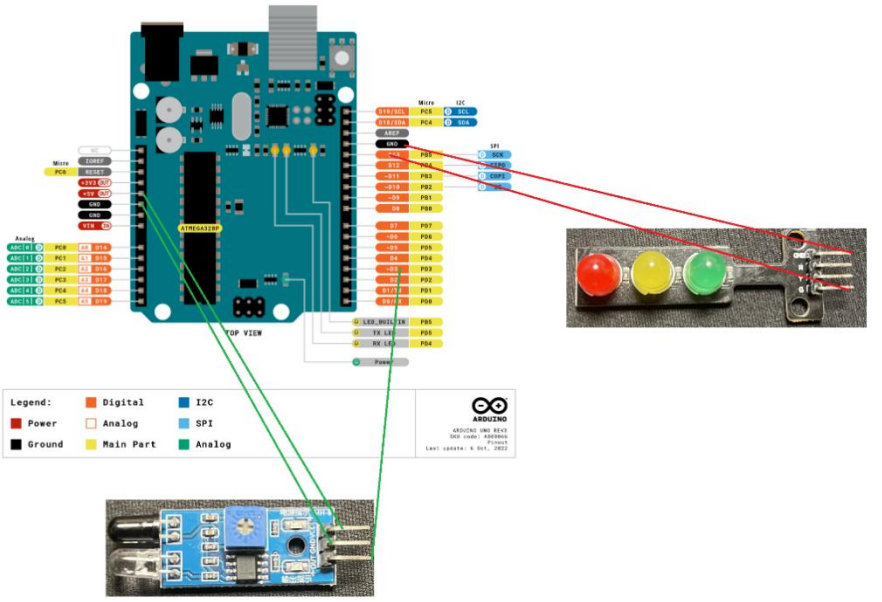
交通灯模块（仅使用绿灯）：

交通灯模块引脚	Arduino UNO R3引脚
GND	GND
R	不接入
Y	不接入
G	13号数字引脚

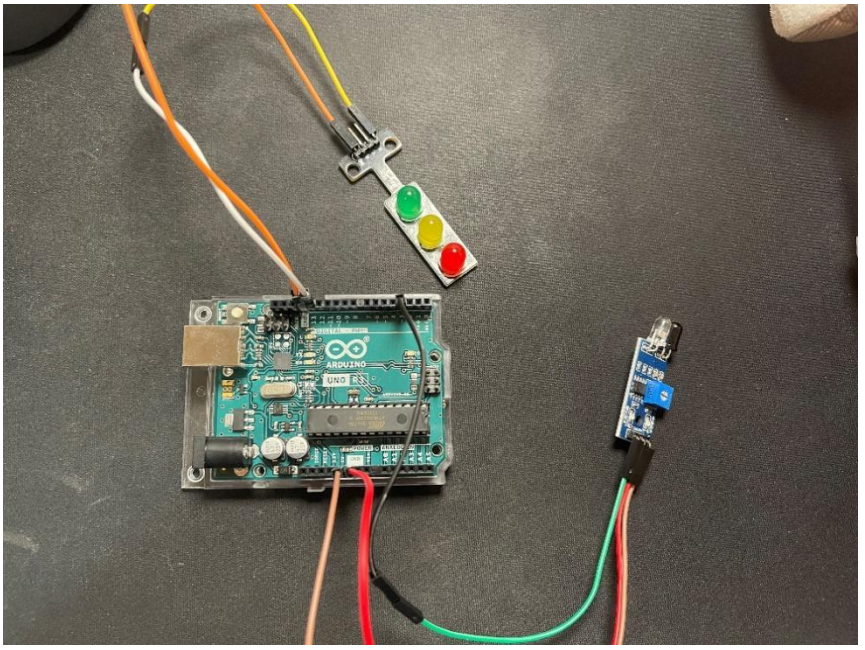
红外避障模块：

红外避障模块引脚	Arduino UNO R3引脚
VCC	5V供电引脚
GND	GND
OUT	3号数字引脚

结合官方引脚定义说明：



实物连接图：

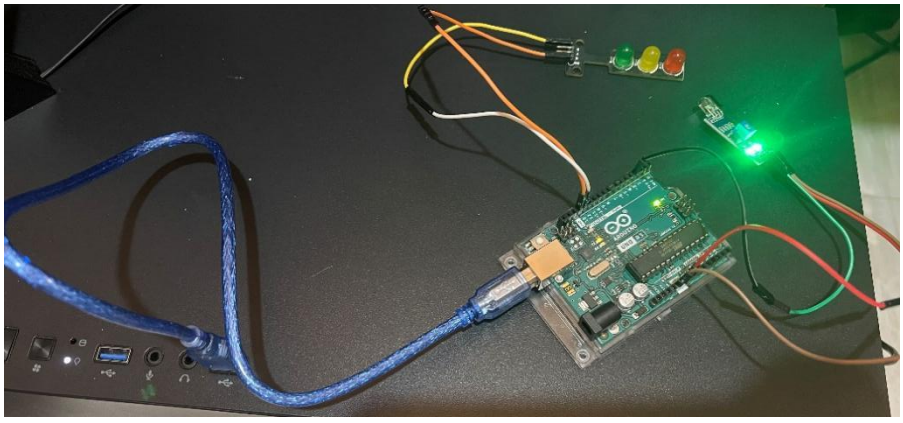


（3）下位机与上位机连接方案

系统整体连接图：

如图所示，下位机与上位机之间通过串口进行通信。下位机需要输出给上位机红外避障模块的检测结果，0表示无障碍，1表示有障碍，下位机输出至上位机可看作0、1串。上位机需要向下位机输出控制信号，控制交通灯是否点亮，0表示关闭交通灯，1表示点亮交通灯，上位机输出至下位机的控制信号也可看做0、1串。

实物连接图：



2、下位机数据采集

（1）传感器数据采集方案

下位机通过读取3号数字引脚的电平高低状态采集红外避障模块的结果，通过改变13号数字引脚的电平高低状态控制交通等模块的开关。

（2）数据采集程序

定义引脚常量：

```
int led_pin = 13;
int detect_pin = 3;
```

在setup函数中定义数字引脚的输入输出属性：

```
void setup() {
  // put your setup code here, to run once:
  pinMode(led_pin, OUTPUT);
  pinMode(detect_pin, INPUT);
}
```

在loop函数中编写避障结果读取逻辑与交通灯控制逻辑：

```
void loop() {
  // put your main code here, to run repeatedly:
  int detect_result = digitalRead(detect_pin);
  if (detect_result == HIGH)
    digitalWrite(led_pin, HIGH);
  else
    digitalWrite(led_pin, LOW);
}
```

需要注意的是，此处代码表述的逻辑为：输入电平为高时有障

碍，交通灯亮起，输入电平为低时无障碍，交通灯关闭。输入电平定义与说明文档不一致，这可能是硬件型号改变导致的。

3、数据编码与传输

（1）下位机到上位机传输数据包设计

如前 1.4 中所述，下位机通过向上位机传输 0、1(ASCII 编码)串来传达红外避障模块的结果——0 代表无障碍，1 代表有障碍。上位机通过向下位机传输 0、1(ASCII 编码)串来表示当前交通灯的开关控制——0 代表关闭，1 代表开启。

（2）在下位机设计并实现数据包编码与传输程序

在 setup 函数新增开启串口通信的代码：

```
void setup() {
  // put your setup code here, to run once:
  pinMode(led_pin, OUTPUT);
  pinMode(detect_pin, INPUT);
  Serial.begin(9600);
}
```

在 loop 函数中新增上位机通信代码：

```
void loop() {
  // put your main code here, to run repeatedly:
  int detect_result = digitalRead(detect_pin);
  if (detect_result == HIGH)
    Serial.print("1");
  else
    Serial.print("0");

  char read_result = Serial.read();
  if (read_result == '1')
    digitalWrite(led_pin, HIGH);
  else
    digitalWrite(led_pin, LOW);
}
```

（3）上位机到下位机传输数据包设计

如前 1.4 中所述，下位机通过向上位机传输 0、1(ASCII 编码)串来传达红外避障模块的结果——0 代表无障碍，1 代表有障碍，上位机可以读取结果，并打印结果给用户展示。上位机通过向下位机传输 0、1(ASCII 编码)串来表示当前交通灯的开关控制——0 代表关闭，1 代表开启。

（4）在上位机设计并实现数据包编码与传输程序

导入Python包+定义状态变量：

```
import serial
import threading

CURR_DETECT_TEXT = "当前避障模块是否有障碍物：无"
CURR_LED_TEXT = "当前LED状态：灭"
CURR_LED_STATE = '0'
```

读取串口数据，并更新状态：

```
def read_serial_data(ser):
    global CURR_DETECT_TEXT
    while True:
        data = ser.read().decode()  # 读取单个字节的数据
        CURR_DETECT_TEXT = f"当前避障模块是否有障碍物：{'有' if data == '1' else '无'}"
```

向串口输出：

```
def write_serial_data(ser):
    global CURR_LED_STATE
    while True:
        ser.write(CURR_LED_STATE.encode(encoding='ascii'))
```

打印当前系统状态：

```
def print_curr_text():
    global CURR_DETECT_TEXT,CURR_LED_TEXT
    while True:
        print(f"\r{CURR_DETECT_TEXT},{CURR_LED_TEXT}", end='', flush=True)
```

打开串口通信，并启动后台读取、写入、打印线程：

```
# 打开串口
ser = serial.Serial('COM3', 9600)  # 串口名称和波特率
# 创建后台线程读取数据,写入数据,并打印当前系统状态
read_thread = threading.Thread(target=read_serial_data, args=(ser,))
read_thread.start()
write_thread = threading.Thread(target=write_serial_data, args=(ser,))
write_thread.start()
print_thread = threading.Thread(target=print_curr_text)
print_thread.start()
```

在主线程中获得用户输入，并更新状态：

```
# 主线程执行其他操作
while True:
    user_input = input()
    CURR_LED_STATE = user_input
    CURR_LED_TEXT = f"当前LED状态：{'亮' if user_input == '1' else '灭'}"
```

4、数据解析与输出

（1）设计并实现上位机接收数据包的解析与展示程序

如第 3 节中展示，将不同部分的代码拼接起来，即可得到上位机整体的接收数据包的解析与展示程序：

control.py > ...

```python
import serial
import threading

CURR_DETECT_TEXT = "当前避障模块是否有障碍物：无"
CURR_LED_TEXT = "当前LED状态：灭"
CURR_LED_STATE = '0'

def read_serial_data(ser):
    global CURR_DETECT_TEXT
    while True:
        data = ser.read().decode()  # 读取单个字节的数据
        CURR_DETECT_TEXT = f"当前避障模块是否有障碍物：{'有' if data == '1' else '无'}"

def write_serial_data(ser):
    global CURR_LED_STATE
    while True:
        ser.write(CURR_LED_STATE.encode(encoding='ascii'))

def print_curr_text():
    global CURR_DETECT_TEXT,CURR_LED_TEXT
    while True:
        print(f"\r{CURR_DETECT_TEXT},{CURR_LED_TEXT}", end='', flush=True)
# 打开串口
ser = serial.Serial('COM3', 9600)  # 串口名称和波特率
# 创建后台线程读取数据,写入数据,并打印当前系统状态
read_thread = threading.Thread(target=read_serial_data, args=(ser,))
read_thread.start()
write_thread = threading.Thread(target=write_serial_data, args=(ser,))
write_thread.start()
print_thread = threading.Thread(target=print_curr_text)
print_thread.start()

# 主线程执行其他操作
CURR_LED_STATE = "0"
while True:
    user_input = input()
    CURR_LED_STATE = user_input
    CURR_LED_TEXT = f"当前LED状态：{'亮' if user_input == '1' else '灭'}"

```

（2）设计并实现下位机接收数据包的解析程序、（4）设计并实现下位机控制信号灯显示程序

最终实验 1b 的下位机程序代码如下图（loop 中包含了包解析与信号灯显示）：

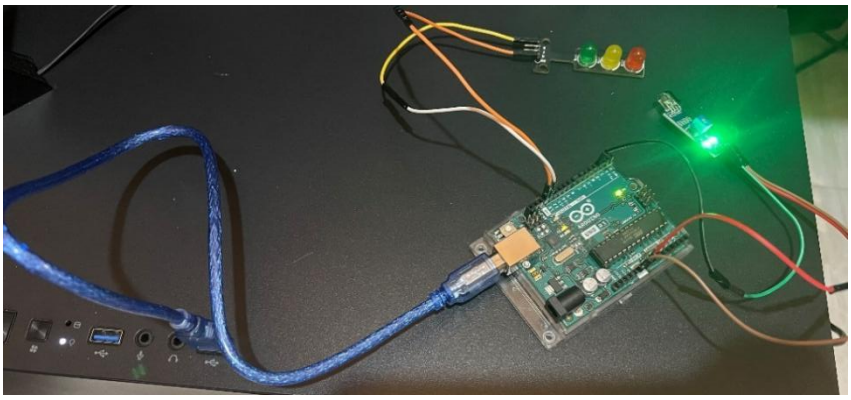
```arduino
int led_pin = 13;
int detect_pin = 3;

void setup() {
  // put your setup code here, to run once:
  pinMode(led_pin, OUTPUT);
  pinMode(detect_pin, INPUT);
  Serial.begin(9600);
  digitalWrite(led_pin, LOW);
}

void loop() {
  // put your main code here, to run repeatedly:
  int detect_result = digitalRead(detect_pin);
  if (detect_result == HIGH)
    Serial.print("1");
  else
    Serial.print("0");

  char read_result = Serial.read();
  if (read_result == '1')
    digitalWrite(led_pin, HIGH);
  else
    digitalWrite(led_pin, LOW);
```

（3）设计信号灯显示方案

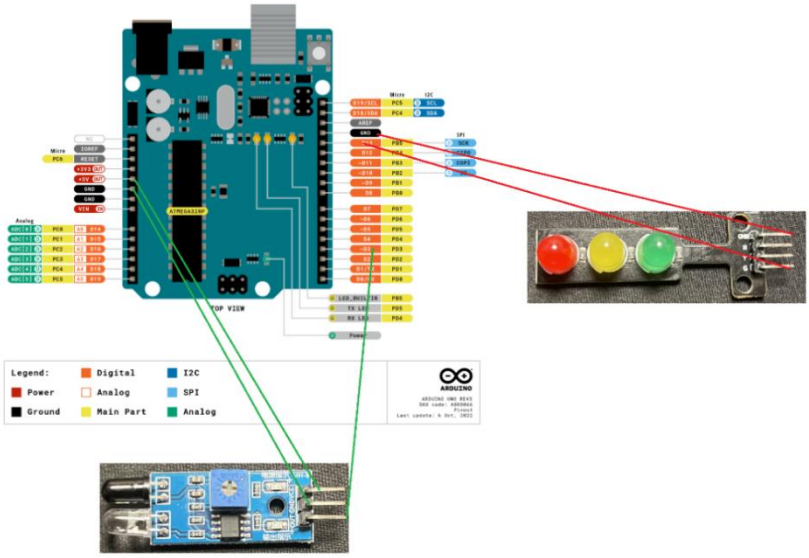
当用户在上位机中输入1时，则控制信号灯亮，否则信号灯关闭。

四、实验结果及分析

实物连接图：

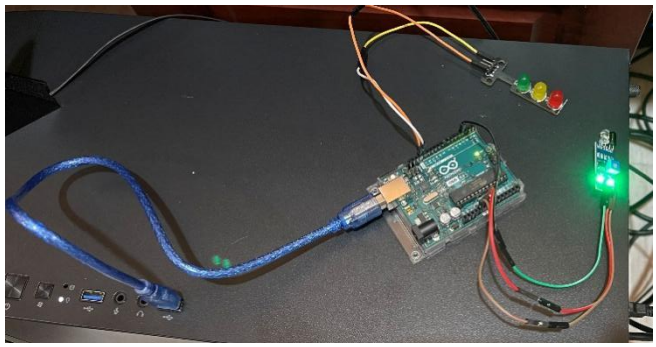


说明（参照**三、实验过程或算法**中**传感器-下位机-上位机的连接**部分的说明）：



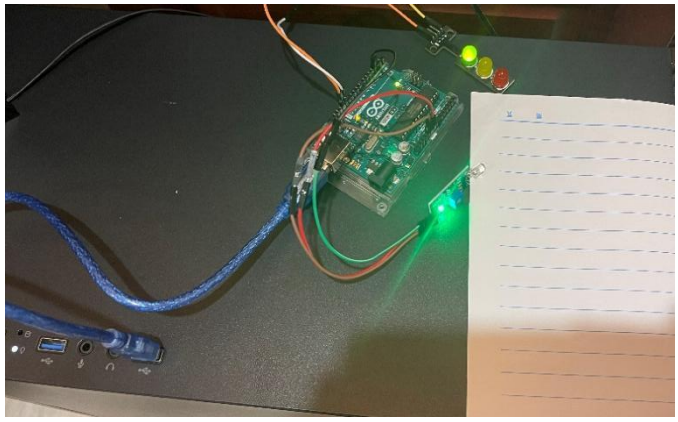
实验 1a 的结果：

- 没有障碍物：



避障模块没有检测到障碍物，依照 loop 函数逻辑，交通灯关闭。

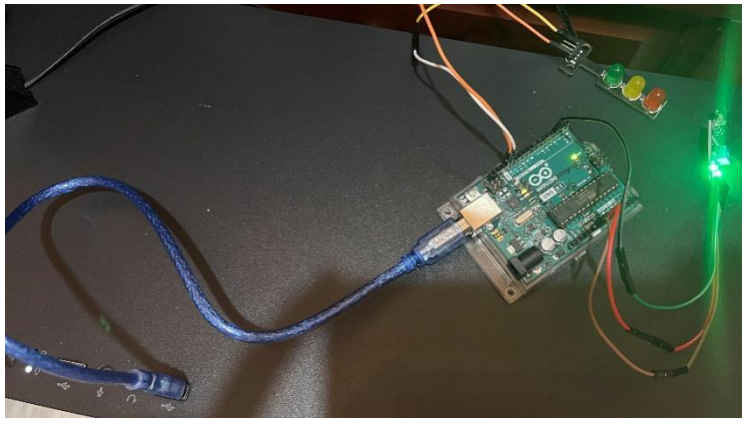
- 有障碍物：



在避障模块红外线发射与接收管之间放入纸张，避障模块检测到障碍，依照 loop 函数逻辑，交通灯亮。

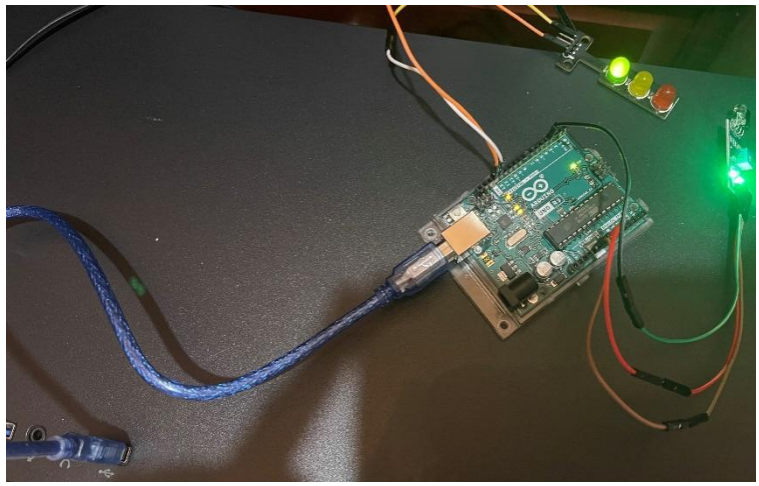
实验 1b 的结果：

- 初始启动状态（无障碍物，LED 灯关）：



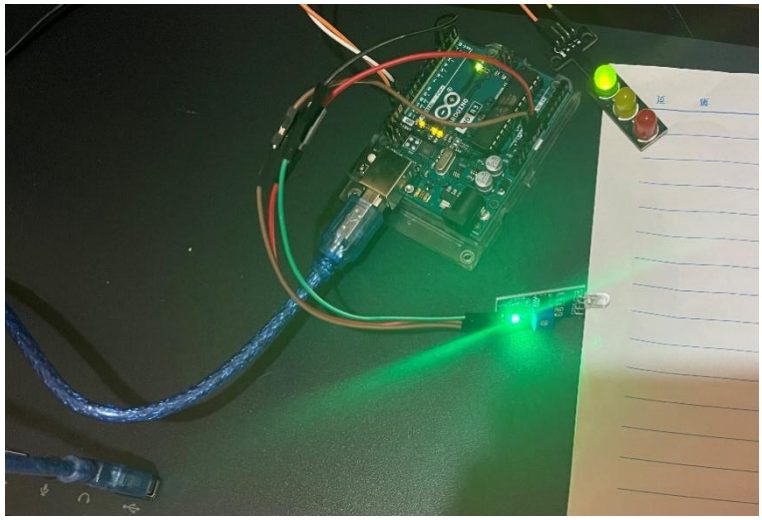
- 输入 1，启动 LED 灯后：

```
当前避障模块是否有障碍物：无,当前LED状态：灭1
当前避障模块是否有障碍物：无,当前LED状态：亮
```



- 在 LED 亮时，用纸挡住红外避障模块：

```
当前避障模块是否有障碍物：无,当前LED状态：灭1
当前避障模块是否有障碍物：有,当前LED状态：亮
```



- 在 LED 灯关时，用纸挡住红外避障模块：

```
当前避障模块是否有障碍物：无,当前LED状态：灭1
当前避障模块是否有障碍物：有,当前LED状态：亮0
当前避障模块是否有障碍物：有,当前LED状态：灭
```