《 智能系统 》实验报告

年级、专业、班级	2021级计算机科学与技术 卓越1班、卓越2班、1班	姓名	李宽宇、朱齐高、 禹天尹、张志维
实验题目	数据采集与通信		
实验时间	2024.3.31	实验地点	DS1401
实验成绩		实验性质	□验证性 □设计性 □综合性

教师评价：

□算法/实验过程正确　　□源程序/实验内容提交

□程序结构/实验步骤合理　　□实验结果正确

□语法、语义正确　　□报告规范

其他：

评价教师签名：

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）。

报告创建时间：

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；

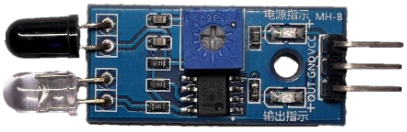
（4）设计并实现下位机控制信号灯显示程序。

三、实验过程或算法（代码）

1、传感器-下位机-上位机的连接

（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，一类是红外避障模块，另一类是交通灯模块。



红外避障传感器模块对环境光线适应能力强，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），可通过电位器旋钮调节检测距离，有效距离范围 2～30cm，工作电压为 3.3V-5V。

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。

（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

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。红外避障模块除了需要供电，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：

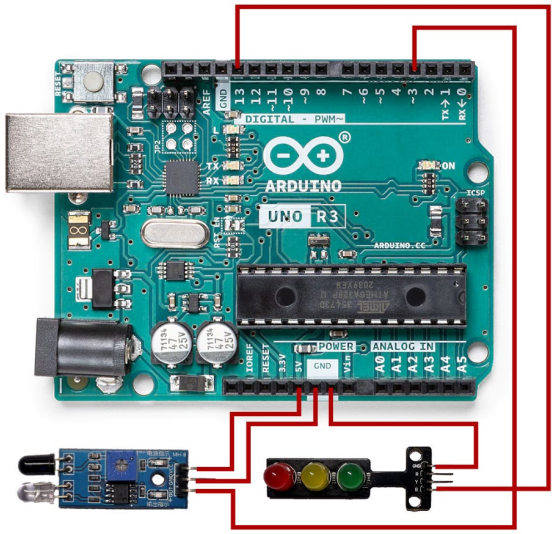
交通灯模块：

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

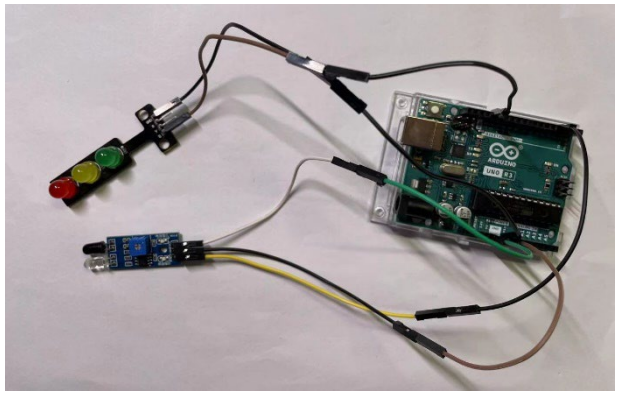
红外避障模块：

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

引脚连接图：



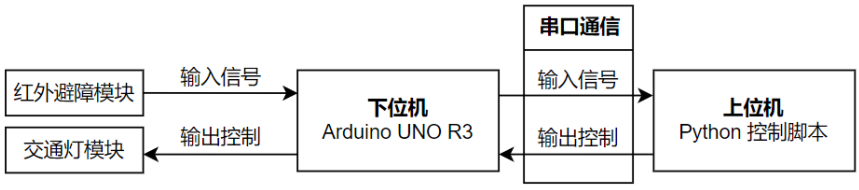
实物连接图：



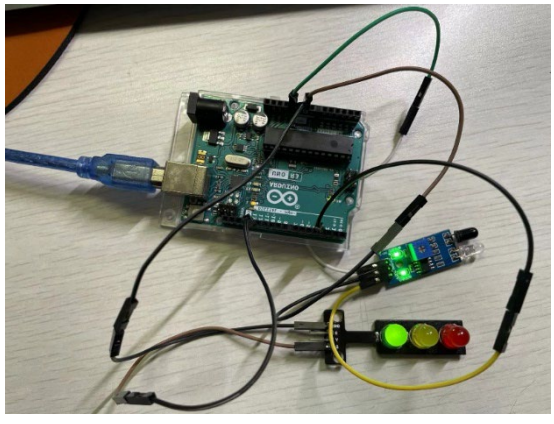
（3）了解下位机与上位机通信协议，设计下位机与上位机连接方案，给出实物连接图表

通信协议：

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作 0、1 串。



实物连接图：



2、下位机数据采集

（1）设计传感器数据采集方案

避障传感器只有两种状态——有障碍、无障碍，故设计方案如下：

①数据产生：前方无障碍物，此时避障传感器为输出接口为高电平；将避障模块的红外探头对准障碍物，距离为 3-30cm 厘米以内，此时避障传感器为输出接口为低电平。

②数据收集：将红外避障传感器的输入接口接到 Arduino UNO R3 板的 3 号接口。

③数据控制：将 LED 的输出接口接到 Arduino UNO R3 板的 13 号接口，红外避障传感器检测到有障碍物后，使 LED 输出接口为高电平灯亮，并且为了使得亮灯时间明显，延迟 1s。当没有检测到障碍物时，使 LED 输出接口为低电平灯灭。

（2）设计并实现数据采集程序（函数）

①定义避障传感器接口为 3，LED 接口为 13，避障传感器采集的数据值为 val

```
int buttonpin = 3; //定义避障传感器接口
int Led = 13;      //定义LED 接口
int val;           //定义数字变量val
```

②定义避障传感器为输入接口、LED 为输出接口，并初始化 LED 为低电平灯灭

```
void setup()
{
    pinMode(buttonpin, INPUT); //定义避障传感器为输入接口
    pinMode(Led, OUTPUT);      //定义LED 为输出接口
    digitalWrite(Led, LOW);    //初始LED灭
}
```

③循环读取避障传感器输入引脚的电平，当有障碍物时，避障传感器为低电平，设定 LED 输出接口为高电平灯亮，并延迟 1s；当无障碍物时，避障传感器为高电平，设定 LED 输出接口为低电平灯灭

```
void loop()
{
    val = digitalRead(buttonpin); //将数字接口3的值读取赋给val
    if (val == LOW)               //当避障传感器检测低电平时，有障碍物，LED 亮
    {
        digitalWrite(Led, HIGH);
        delay(1000);              //延迟1s，亮灯时间明显
    }
    else // 避障传感器检测为高电平，无障碍物，灯灭
    {
        digitalWrite(Led, LOW);
    }
}
```

3、数据编码与传输

（1）下位机到上位机传输数据包设计

①开启串口，实现下位机到上位机的串口通信

②规定数据包，当红外避障传感器信号发生变化时，如果是低电平，说明有障碍物，则向上位机串行传输字符串“YES”；如果是高电平，说明无障碍物，则向上位机串行传输字符串“NO”。

（2）在下位机设计并实现数据包编码与传输程序（函数）

①定义避障传感器接口为 3，LED 接口为 13，避障传感器采集的数据值为 val,前一次传感器的信号值为 prv

```
int buttonpin = 3;  //定义避障传感器接口
int Led = 13;       //定义LED 接口
int val;            //定义避障传感器接口的数字变量val
int prv = -1;       //定义前一次传感器收到的val
```

②定义避障传感器为输入接口、LED 为输出接口，并初始化 LED 为低电平灯灭，规定串口通信，波特率设置为 9600

```
void setup() {
  pinMode(buttonpin, INPUT);  //定义避障传感器为输出接口
  pinMode(Led, OUTPUT);       //定义LED 为输出接口
  Serial.begin(9600);         //连接上位机，波特率为9600
  digitalWrite(Led, LOW);     //初始LED为低电平灯灭
}
```

③接收传感器信号然后发送给上位机，循环读取避障传感器输入引脚的电平，并于前一次传感器的信号判断是否发生变化，当有障碍物时，避障传感器为低电平，向上位机串行传输字符串“YES”；当无障碍物时，避障传感器为高电平，向上位机串行传输字符串“NO”

```
void loop() {
  // 接收传感器信号然后发送给上位机
  val = digitalRead(buttonpin);  //将数字接口3的值读取赋给val
  if(val != prv) {  // 只有当val发生变化的时候才向上位机发数据
    if (val == LOW) {  //当避障传感器检测低电平时，有障碍物，传输YES
      Serial.print("YES\n");
    } else {  //当避障传感器检测高电平时，无障碍物，传输NO
      Serial.print("NO\n");
    }
  }
  prv = val;  //前一次传感器收到的val
}
```

（3）上位机到下位机传输数据包设计

①开启串口，实现上位机到下位机的串口通信

②规定数据包，用户在上位机通过按键键入字符“1”，并串行传输该字符给下位机模拟前方有障碍；或者用户在上位机通过按键键入字符“0”，并串行传输该字符给下位机模拟前方无障碍。

（4）在上位机设计并实现数据包编码与传输程序（函数）

①import 显示窗口所需要的库和串口通信所需要的库

```
# 显示窗口所需要的库
from prompt_toolkit import Application
from prompt_toolkit.buffer import Buffer
from prompt_toolkit.layout.containers import HSplit, Window
from prompt_toolkit.layout.controls import BufferControl
from prompt_toolkit.layout.layout import Layout
from prompt_toolkit.key_binding import KeyBindings

# 串口通信所需要的库
import serial
import threading
from queue import Queue
```

②创建串口，包括创建缓冲区，设置串口参数信息，串口号、波特率等

```
# 创建三个缓冲区
buffer_display = Buffer()   # 用于显示上方提示信息
buffer_output = Buffer()    # 用于显示从串口读入的数据
buffer_input = Buffer()     # 用于从用户读入要向串口写入的数据

# 串口参数设置
serialPort = "COM3"  # 串口
baudRate = 9600     # 波特率
buffer_display.text = "输入`exit`退出程序，输入`1`或`0`控制交通灯亮灭\n"
buffer_display.text += "参数设置：串口=%s ，波特率=%d" % (serialPort, baudRate)
```

③连接串口；创建输出队列，存储要向串口写入的数据；创建终止标志变量，用于 Control-C 退出时终止线程

```
# 连接串口
try:
    ser = serial.Serial(serialPort, baudRate, timeout=0.5)
except Exception:
    raise Exception("无法打开串口！请确保串口号正确，且设备已连接！")

# 创建输出队列，存储要向串口写入的数据
output_queue = Queue()

# 创建终止标志变量，用于Control-C退出时终止线程
stop_event = threading.Event()
```

④创建写串口线程，从输出队列中读取数据，并将数据写入串口

```python
# 写串口线程
def write_serial(event):
    while not event.is_set():
        try:
            # 从输出队列中读取数据
            # [Note] 如果队列为空，会阻塞，直到有数据写入队列。因此要设置一个超时时间
            data = output_queue.get(timeout=0.5)
            # 将数据写入串口
            if data:
                ser.write(data)
        except Exception:
            pass
    print("Write Serial Exit!")
```

⑤设置显示信息，首先创建一个显示容器和布局，并绑定快捷键，包括退出快捷键（Ctrl＋C）和回车快捷键（Enter），并设置通信内容，将输入的数据写入缓冲区，最后创建终端应用

```python
# 创建一个显示容器
root_container = HSplit([
    Window(height=2, content=BufferControl(buffer=buffer_display,
                                           focusable=False)),  # 上方提示信息
    Window(height=1, char='-'),  # 分割线
    Window(height=5, content=BufferControl(buffer=buffer_output,
                                           focusable=False)),  # 上方显示
    Window(height=1, char='-'),  # 分割线
    Window(height=1, content=BufferControl(buffer=buffer_input,
                                           focusable=True)),  # 下方指令输入
])
# 创建一个布局
layout = Layout(root_container)

# 绑定快捷键
kb = KeyBindings()

# 退出快捷键（Ctrl + C）
@kb.add('c-c')
def exit_(event):
    event.app.exit()  # 退出应用

# 回车快捷键（Enter）
@kb.add('enter')
def enter_(event):
    text = buffer_input.text.strip()
    if text == 'exit':
        exit_(event)
    # 将输入的数据写入缓冲区
    elif text == '1':
        output_queue.put(b"1")
        buffer_output.text += f"Sent: {text}\n"
    elif text == '0':
        output_queue.put(b"0")
        buffer_output.text += f"Sent: {text}\n"
    else:
        buffer_output.text += f"Error: Invalid input: {text}\n"
    # 清空输入缓冲区
    buffer_input.text = ""

# 创建终端应用
app = Application(layout=layout, key_bindings=kb, full_screen=True)
```

⑥启动线程，用上位机控制下位机 LED，并在退出后终止线程和关闭串口

```
if __name__ == '__main__':
    # 启动线程
    t2 = threading.Thread(target=write_serial, args=(stop_event,))    # 上位机控制下位机LED
    t2.start()
    app.run()  # 阻塞
    # 退出后终止线程
    stop_event.set()
    # 关闭串口
    ser.close()
```

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

①import 显示窗口所需要的库和串口通信所需要的库

```
# 显示窗口所需要的库
from prompt_toolkit import Application
from prompt_toolkit.buffer import Buffer
from prompt_toolkit.layout.containers import HSplit, Window
from prompt_toolkit.layout.controls import BufferControl
from prompt_toolkit.layout.layout import Layout
from prompt_toolkit.key_binding import KeyBindings

# 串口通信所需要的库
import serial
import threading
from queue import Queue
```

②创建串口，包括创建缓冲区，设置串口参数信息，串口号、波特率等

```
# 创建三个缓冲区
buffer_display = Buffer()    # 用于显示上方提示信息
buffer_output = Buffer()     # 用于显示从串口读入的数据
buffer_input = Buffer()      # 用于从用户读入要向串口写入的数据

# 串口参数设置
serialPort = "COM3"  # 串口
baudRate = 9600      # 波特率
buffer_display.text += "参数设置：串口=%s ，波特率=%d" % (serialPort, baudRate)
```

③连接串口并创建终止标志变量，用于 Control-C 退出时终止线程

```
# 连接串口
try:
    ser = serial.Serial(serialPort, baudRate, timeout=0.5)
except Exception:
    raise Exception("无法打开串口！请确保串口号正确，且设备已连接！")

# 创建终止标志变量，用于Control-C退出时终止线程
stop_event = threading.Event()
```

④创建读串口线程，从读取串口数据，并将读取的数据写入显示缓冲区

```
# 读串口线程
def read_serial(event):
    while not event.is_set():
        try:
            # 读取串口数据
            data = ser.readline().decode('utf-8').rstrip()
            # 将读取的数据写入显示缓冲区
            if data:
                buffer_output.text += f"Received: {data}\n"
            if buffer_output.text.count('\n') > 5:  # 限制显示行数
                buffer_output.text = buffer_output.text.split('\n', 1)[1]
        except Exception:
            pass
    print("Read Serial Exit!")
```

⑤设置显示信息，首先创建一个显示容器和布局，并绑定退出快捷键（Ctrl + C），最后创建终端应用

```
# 创建一个显示容器
root_container = HSplit([
    Window(height=2, content=BufferControl(buffer=buffer_display,
                                           focusable=False)),  # 上方提示信息
    Window(height=1, char='-'),  # 分割线
    Window(height=5, content=BufferControl(buffer=buffer_output,
                                           focusable=False)),  # 上方显示
    Window(height=1, char='-'),  # 分割线
    Window(height=1, content=BufferControl(buffer=buffer_input,
                                           focusable=True)),  # 下方指令输入
])
# 创建一个布局
layout = Layout(root_container)

# 绑定快捷键
kb = KeyBindings()


# 退出快捷键（Ctrl + C）
@kb.add('c-c')
def exit_(event):
    event.app.exit()  # 退出应用

# 创建终端应用
app = Application(layout=layout, key_bindings=kb, full_screen=True)
```

⑥启动线程，用显示下位机向上位机传输的数据，并在退出后终止线

程和关闭串口

```
if __name__ == '__main__':
    # 启动线程
    t1 = threading.Thread(target=read_serial, args=(stop_event,))     #下位机向上位机传输的信号
    t1.start()
    app.run()  # 阻塞
    # 退出后终止线程
    stop_event.set()
    # 关闭串口
    ser.close()
```

（2）设计并实现下位机接收数据包的解析程序

①定义上位机的输入字符信号为 chr

```
char chr;              //定义上位机的输入字符信号
```

②在下位机循环读取上位机向下位机串口通信的输入控制信号

```
void loop() {
  // 接收上位机控制信号
  chr = Serial.read();
}
```

（3）设计信号灯显示方案

下位机接收上位机串口通信发送的控制信号，当上位机信号为 1，则设定 LED 为输出接口为高电平灯亮，并延迟 1 秒，使亮灯时间明显；当上位机信号为 0，则设定 LED 为输出接口为低电平灯灭。

（4）设计并实现下位机控制信号灯显示程序

①定义避障传感器接口为 3，LED 接口为 13，上位机的输入字符信号为 chr

```
int Led = 13;          //定义LED 接口
int buttonpin = 3;     //定义避障传感器接口
char chr;              //定义上位机的输入字符信号
```

②定义避障传感器为输入接口、LED 为输出接口，并初始化 LED 为低电平灯灭，规定串口通信，波特率设置为 9600

```
void setup() {
  pinMode(Led, OUTPUT);          //定义LED 为输出接口
  pinMode(buttonpin, INPUT);     //定义避障传感器为输出接口
  Serial.begin(9600);            //连接上位机，波特率为9600
  digitalWrite(Led, LOW);
}
```

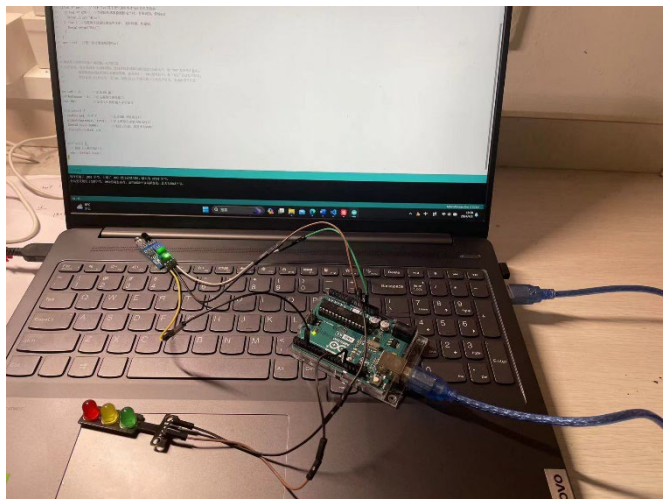
③接收上位机发送给下位机的控制信号，循环读取上位机的输入控制信号，当上位机信号为 1，则设定 LED 为输出接口为高电平灯亮，并延迟 1 秒，使亮灯时间明显，当上位机信号为 0，则设定 LED 为输出接口为低电平灯灭

```
void loop() {
  // 接收上位机信号控制小灯开关
  chr = Serial.read();
  if (chr == '1') {  //上位机信号为1，则设定LED为输出接口为高电平灯亮
    digitalWrite(Led, HIGH);
    delay(1000);  //延迟1s，亮灯时间明显
  } else {  //上位机信号为0，则设定LED为输出接口为低电平灯灭
    digitalWrite(Led, LOW);
  }
}
```

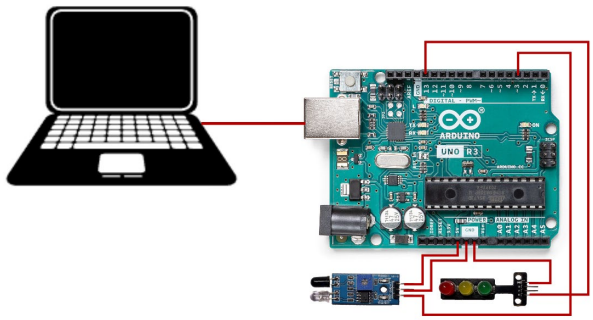
四、实验结果及分析

1、给出实物连接图并做说明

（1）实物连接图



（2）连接示意图



（3）连接说明

交通灯模块需要数字供电信号，以便在Arduino程序或者python程序中控制灯的开关，故将交通灯模块的GND引脚连接到Arduino板上的GND，将交通灯模块的G引脚连接到Arduino板上的13号接口。

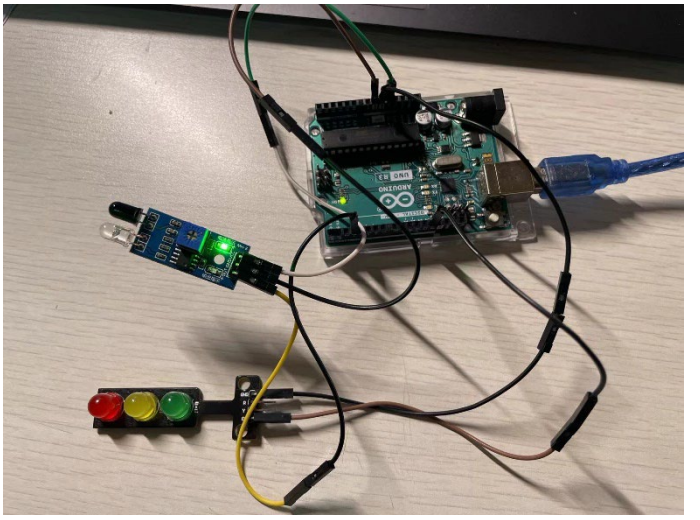
红外避障模块除了需要供电，还需将障碍物检测的结果通过OUT引脚传输至下位机，故将红外避障模块VCC引脚连接到Arduino板上5v接口，将红外避障模块GND引脚连接到Arduino板上GND接口，将红外避障模块OUT引脚连接到Arduino板上3号接口。

需要将Arduino程序烧录到Arduino板上，故将Arduino板连接到电脑USB口上。

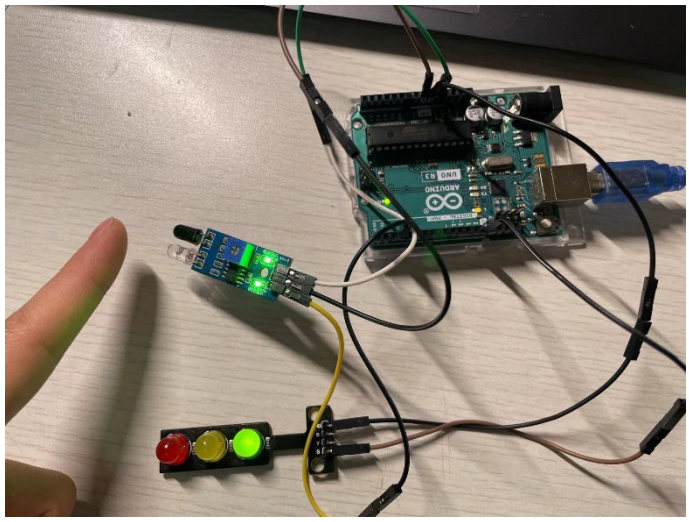
2、给出各种测试情况及结果并做说明分析

（1）红外避障传感器使用及小灯点亮实验

•当避障传感器模块检测到前方无障碍物时，电路板上绿色指示灯熄灭，同时 OUT 端口持续输出高电平信号, LED 绿灯为低电平灯灭



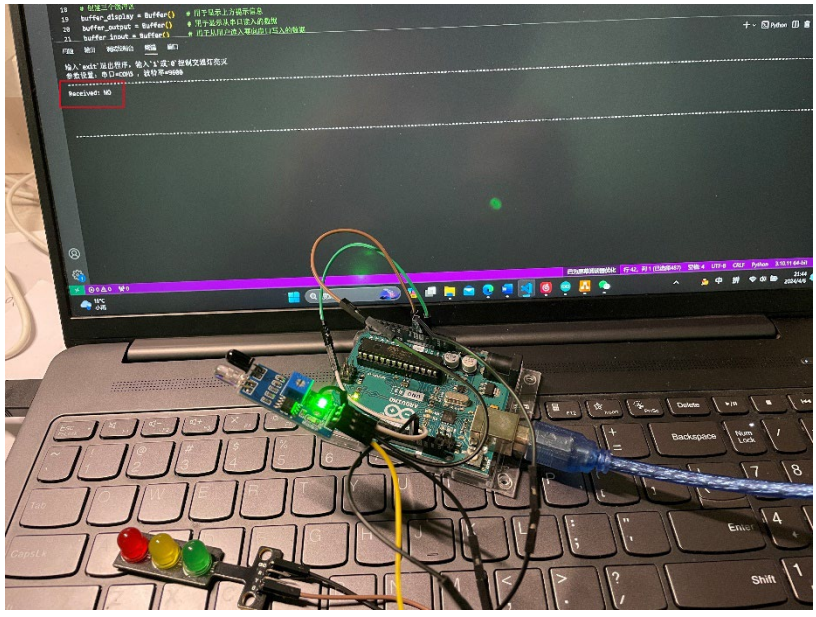
•当避障传感器模块检测到前方障碍物信号时，电路板上绿色指示灯点亮，同时 OUT 端口持续输出低电平信号, LED 绿灯为高电平灯亮



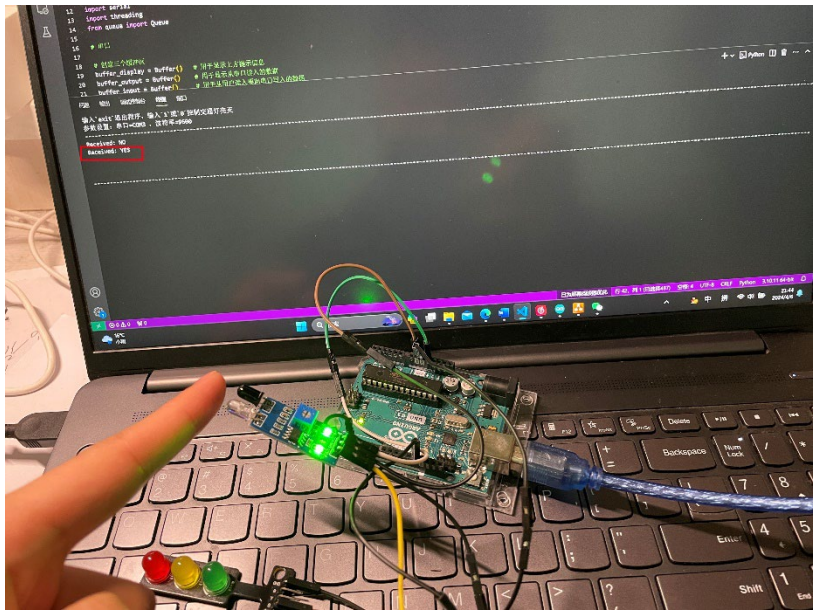
（2）Python 上位机与下位机通信实验

①下位机向上位机传输信号

•当避障传感器模块检测到前方无障碍物时，电路板上绿色指示灯熄灭，同时 OUT 端口持续输出高电平信号，将“NO”发送给上位机

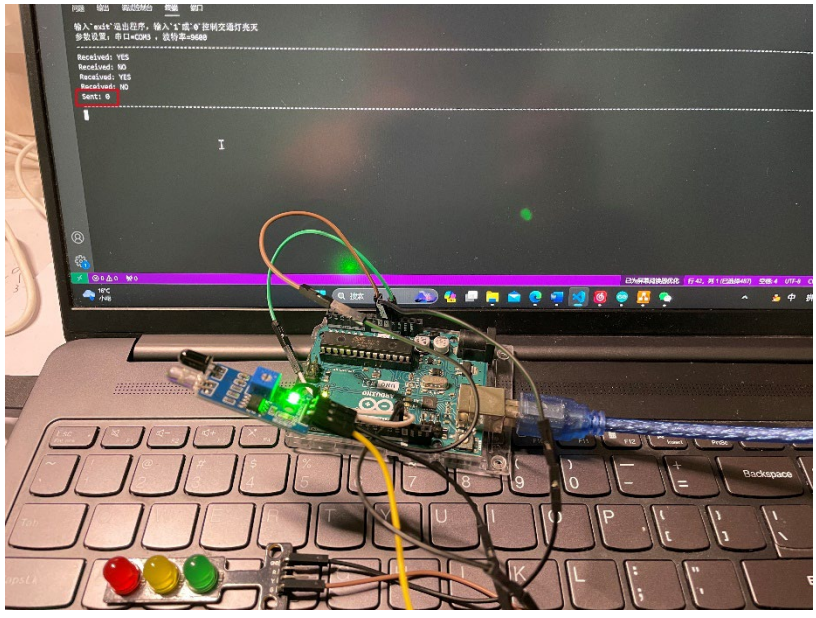


•当避障传感器模块检测到前方障碍物信号时，电路板上绿色指示灯点亮，同时 OUT 端口持续输出低电平信号，将“YES”发送给上位机

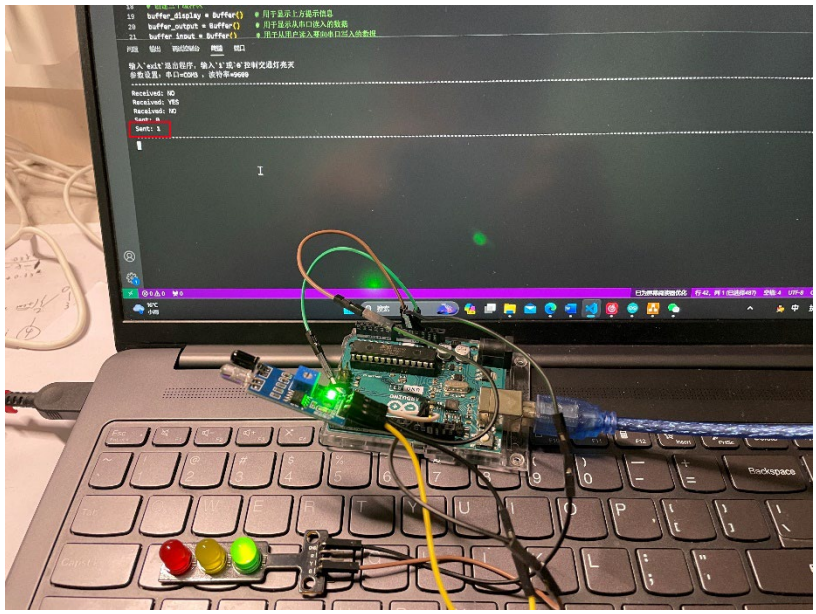


②上位机向下位机传输控制信号

- 上位机信号为 0，则 LED 为输出接口为低电平灯灭



- 上位机信号为 1，则 LED 为输出接口为高电平灯亮


buffer_display = Buffer()
buffer_output = Buffer()
buffer_input = Buffer()
输入'exit'退出程序，输入'1'或'0'控制交通灯亮灭
Received: NO
Received: YES
Received: NO
Sent: 1
Backspace