内蒙古师范大学计算机科学技术学院

《模拟电子技术》课程设计报告

设计题目	智慧联网教室安防系统		
指导教师	张鹏举	**职称**	副教授
姓　　名	张晓阳		
学　　号	20171105231		
日　　期	2019年06月26日		

智慧联网教室安防系统

计算机科学技术学院　2017 级通信工程（物联网）班　张晓阳　20171105231

指导教师　张鹏举　副教授

摘要　本文提出了基于物联网领域的智能教室安防解决方案，采用互联网+硬件+云服务+可视化操作界面四位一体的综合架构，构建了一个顺应当下物联网大潮的智能教室安防系统，适用于居室实时监测，判断，报警，采取措施，远程控制，在智能安防领域发挥着巨大作用。

关键词　智能居室安防；物联网；云服务

1 设计任务及主要技术指标和要求

1.1 设计任务

设计一个能对可燃气体监测，做出判断，报警，远程监控的安全系统并且能通过互联网远程控制的继电器，继而让继电器控制其他电路的综合小规模实用型安防系统。

1.2 主要技术指标

⑴ 通过现有的 MQ-2 气体传感器将具有 DO 开关信号（TTL）输出进行判断。

⑵ 对 Raspberry Pi 3B+开发板的 GPIO 引脚与各传感器及用电器件的连接及读取，控制。

⑶ 对服务器后端的环境搭建与前端的数据实时交互。

⑷ 突破内网访问的局限性，将内网映射到公网，通过公网让所有用户可以远程访问并控制。

1.3 技术要求

⑴ 对传感器返回的数据做出准确判断

⑵ 保证数据的准确与实时性

⑶ 将内网映射到公网，并通过公网访问

2 电路总体说明（基本组成、工作原理）

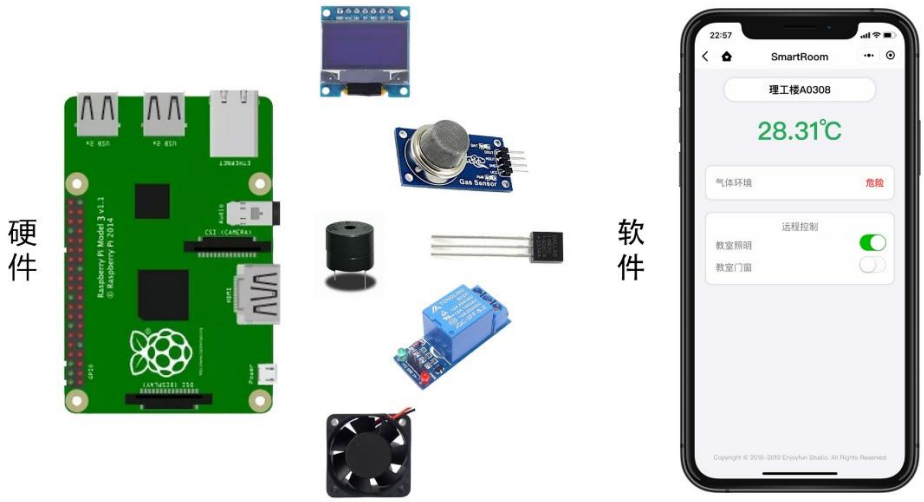


图 1 基本组成

2.1 基本组成

基本组成如图 1 所示

2.1.1 硬件部分

Raspberry Pi 3B+开发板，SSD1306 0.96 寸 OLED 显示屏，18B20 温度传感器，MQ-2 可燃气体传感器， 1 路继电器，有源蜂鸣器，蓝色 LED 灯珠 1 个，4.7KΩ 电阻 1 个，杜邦线若干，5V-2A 电源适配器

2.1.2 软件部分

⑴ 客户端

微信小程序

⑵ 运行系统

① 本地服务器： 基于 Linux 系统 Rispbian 版本

② 云服务器：基于 Linux 系统 CentOS 7 版本

⑶ 开发语言

① 前端： HTML5 + Javascript + CSS

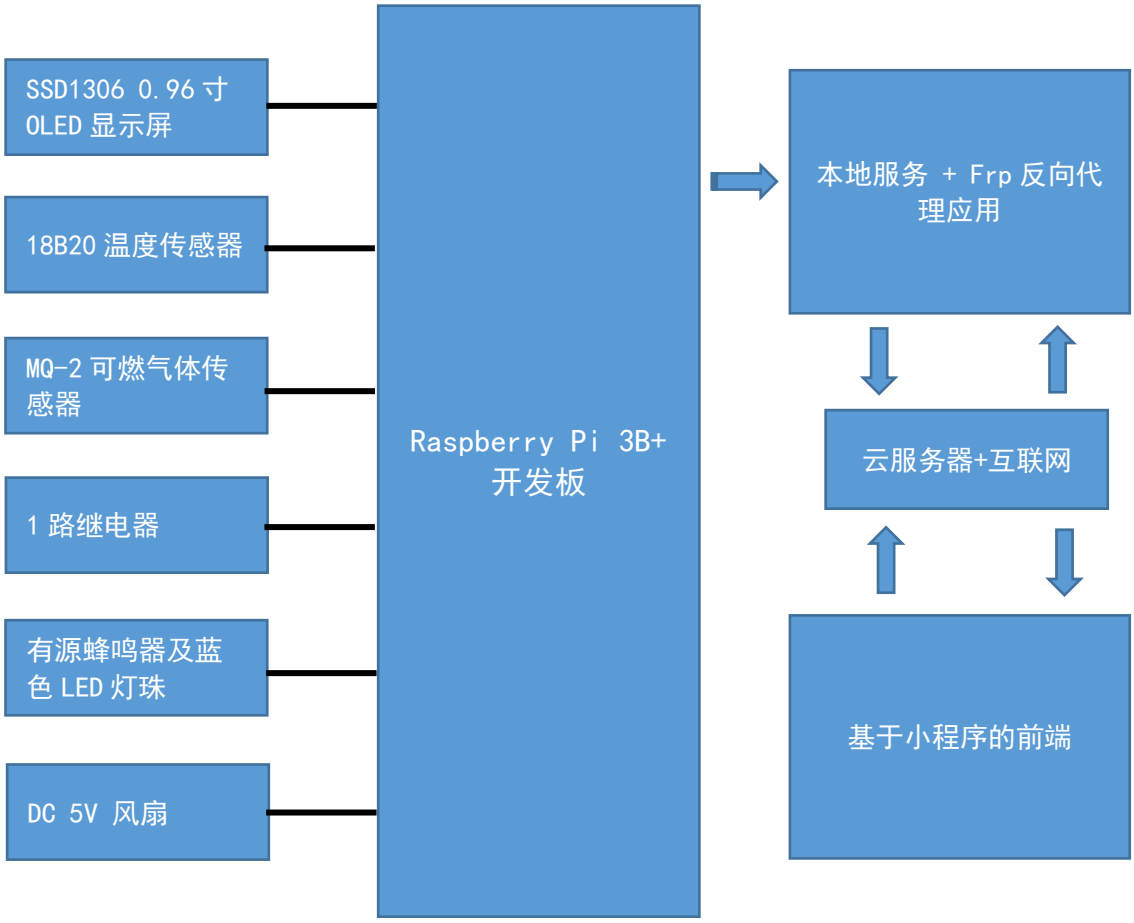
② 后端： Python 3，Frp 反向代理应用，Nignx1.8 Web 服务器，Tornado Web 服务框架，Supervisord 监听自启动应用

⑷ 开发环境

PyCharm（后端），微信 Web 开发者工具（前端）

2.2 工作原理

2.2.1 基本组成原理



2.2.2 传感器

1. 18B20 温度传感器

⑴ 简介

DS18B20 是美国 DALLAS 公司生产的单总线数字式温度传感器，由于具有结构简单，不需要外接电路，可用一根 I/O 数据线既供电又传输数据，可由用户设置温度报警界限等特点，近年来广泛用于粮库等需要测量和控制温度的地方。前些年，DS1820 应用较多，近期，DALLAS 公司又推出了 DS1820 的改进型产品 DS18B20，该产品具有比 DS1820 更好的性能，目前该产品已成为 DS1820 的替代品，在温控系统中得到广泛应用。

⑵ 主要特性

① 适应电压范围 3.0V～5.5V，在寄生电源方式下可由数据线供电。

② DS18B20 与微处理器之间仅需要一条口线即可双向通讯。

③支持多点组网功能，多个 DS18B20 可以并联在唯一的三线上，实现组网多点测温。

④ 不需要任何外围元件，全部传感元件及转换电路集成在外形如一只三极管的电路内。

⑤ 测温范围－55℃～＋125℃，在－IO℃～＋85℃时精度为±0.5℃。

⑥ 可编程的分辨率为 9 位～12 位，对应的可分辨温度分别为 0.5℃、0.25℃、0.125℃和 0.0625℃，可实现高精度测温。

⑦ 在 9 位分辨率时，最多 93.75ms 便可把温度转换为数字，12 位分辨率时最多 750ms 便可把温度值转换为数字。

⑧ 直接输出数字温度信号，以一线总线串行传送给 CPU，同时可传送 CRC 校验码，具有极强的抗干扰纠错能力。

⑨ 电源极性接反时，芯片不会因发热而烧毁，但不能正常工作。

DS18B20 遵循单总线协议，每次测温时必须有初始化、传送 ROM 命令、传送 RAM 命令、数据交换等 4 个过程。

⑶ 工作原理

DS18B20 的读写时序和测温原理与 DS1820 相同，只是得到的温度值的位数因分辨率不同而不同，且温度转换时的延时时间由 2s 减为 750ms。DS18B20 测温原理如图 1 所示。图中低温度系数晶振的振荡频率受温度影响很小，用于产生固定频率的脉冲信号送给计数器 1。高温度系数晶振随温度变化其振荡率明显改变，所产生的信号作为计数器 2 的脉冲输入。

计数器 1 和温度寄存器被预置在-55℃所对应的一个基数值。计数器 1 对低温度系数晶振产生的脉冲信号进行减法计数，当计数器 1 的预置值减到 0 时，温度寄存器的值将加 1，计数器 1 的预置将重新被装入，计数器 1 重新开始对低温度系数晶振产生的脉冲信号进行计数，如此循环直到计数器 2 计数到 0 时，停止温度寄存器值的累加，此时温度寄存器中的数值即为所测温度。斜率累加器用于补偿和修正测温过程中的非线性，其输出用于修正计数器 1 的预置值。

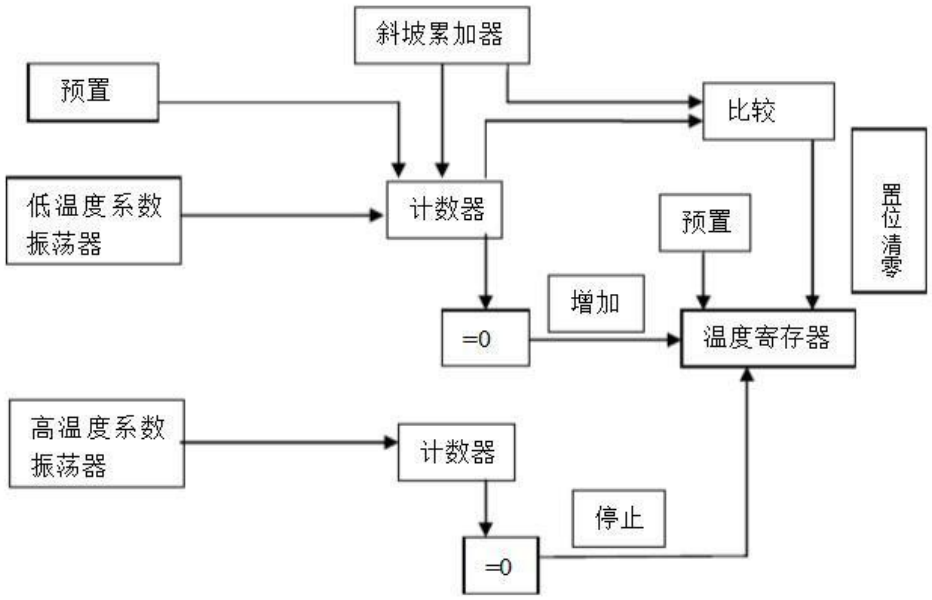


图 2 DS18B20 测温原理

⑷ 测温原理

DS18B20 的测温原理如图 2 所示，图中低温度系数晶振的振荡频率受温度的影响很小，用于产生固定频率的脉冲信号送给减法计数器 1，高温度系数晶振随温度变化其震荡频率明显改变，所产生的信号作为减法计数器 2 的脉冲输入，图中还隐含着计

数门，当计数门打开时，DS18B20 就对低温度系数振荡器产生的时钟脉冲后进行计数，进而完成温度测量。

计数门的开启时间由高温度系数振荡器来决定，每次测量前，首先将-55 ℃所对应的基数分别置入减法计数器 1 和温度寄存器中，减法计数器 1 和温度寄存器被预置在 -55 ℃ 所对应的一个基数值。

减法计数器 1 对低温度系数晶振产生的脉冲信号进行减法计数，当减法计数器 1 的预置值减到 0 时温度寄存器的值将加 1，减法计数器 1 的预置将重新被装入，减法计数器 1 重新开始对低温度系数晶振产生的脉冲信号进行计数，如此循环直到减法计数器 2 计数到 0 时，停止温度寄存器值的累加，此时温度寄存器中的数值即为所测温度。

图 2 中的斜率累加器用于补偿和修正测温过程中的非线性，其输出用于修正减法计数器的预置值，只要计数门仍未关闭就重复上述过程，直至温度寄存器值达到被测温度值，这就是 DS18B20 的测温原理。

2. MQ-2 烟雾传感器

⑴ 简介

MQ-2 气体传感器对液化气、丙烷、氢气的灵敏度高,对天然气和其它可燃蒸汽的检测也很理想。这种传感器可检测多种可燃性气体,是一款适合多种应用的低成本传感器。可以用于家庭和工厂的气体泄漏监测装置,适宜于液化气、丁烷、丙烷、甲烷、烟雾等的探测。

⑵ 工作原理

MQ-2 气体传感器所使用的气敏材料是在清洁空气中电导率较低的二氧化锡(SnO2)。当传感器所处环境中存在可燃气体时,传感器的电导率随空气中可燃气体浓度的增加而增大。使用简单的电路即可将电导率的变化转换为与该气体浓度相对应的输出信号。

⑶ 主要特性

① MQ-2 型传感器对天然气、液化石油气等烟雾有很高的灵敏度，尤其对烷类烟雾更为敏感，具有良好的抗干扰性，可准确排除有刺激性非可燃性烟雾的干扰信息。

② MQ-2 型传感器具有良好的重复性和长期的稳定性。初始稳定，响应时间短，长时间工作性能好。需要注意的是：在使用之前必须加热一段时间，否则其输出的电阻和电压不准确。

③ 其检测可燃气体与烟雾的范围是 100~10000ppm(ppm 为体积浓度。 1ppm=1 立方厘米/1 立方米)

④ 电路设计电压范围宽，24V 以下均可，加热电压 5±0.2V

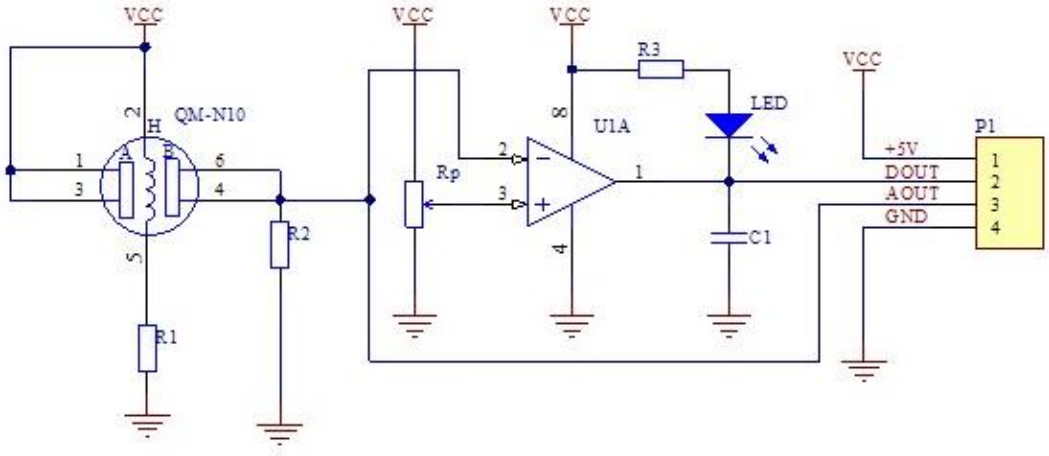


图 3 MQ-2 测控原理

⑷ 测控原理

如图 3 所示，Vc 为回路电压即电源电压，MQ-2 的 4 脚输出随烟雾浓度变化的直流信号，被加到比较器 U1A 的 2 脚，Rp 构成比较器的门槛电压。当烟雾浓度较高输出电压高于门槛电压时，比较器输出低电平（0v），此时 LED 亮报警；当浓度降低传感器的输出电压低于门槛电压时，比较器翻转输出高电平（Vcc），LED 熄灭。同时调节 Rp，可以调节比较器的门槛电压，从而调节报警输出的灵敏度。最后通过 DOUT 引脚输出电平信号（0 和 1），利用单片机编程判断并作出相应操作。

⑸ MQ-2 参数

工作参数如图 4 所示

产品型号			MQ-2
产品类型			半导体气敏元件
标准封装			胶木（黑胶木）
检测气体			可燃气体、烟雾
检测浓度			300-10000ppm(可燃气体)
标准电路条件	回路电压	Vc	≤24V DC
	加热电压	V_H	5.0V±0.2V ACorDC
	负载电阻	R_L	可调
标准测试条件下气敏元件特性	加载电阻	R_H	31Ω±3Ω（室温）
	加热功耗	P_H	≤900mW
	敏感体表面电阻	Rs	2KΩ-20KΩ(in 2000ppm C_3H_8)
	灵敏度	S	Rs(in air)/Rs(1000ppm异丁烷)≥5
	浓度斜率	α	≤0.6($R_{3000ppm}$/$R_{1000ppm}$ C_3H_8)
标准测试条件	温度、湿度		20℃±2℃；65%±5%RH
	标准测试电路		Vc:5.0V±0.1V；
			V_H: 5.0V±0.1V
	预热时间		不少于48小时

图 4 MQ-2 工作参数

2.2.3 本地应用程序

1.本地应用程序传感器监听服务

本地应用程序的传感器监听服务由 Python 编写，应用程序负责在本地监听可燃气体传感器状态，若 MQ-2 的 DOUT 引脚输出高电平为正常，反之将触发继电器和蜂鸣器置为高电平，使风扇和蜂鸣器进入工作状态，以此模拟报警和灭火情况。

本地应用核心程序（Sensor.py）代码如图 5 所示

```python
# 导入库文件
import RPi.GPIO as GPIO
import time

# 创建While循环，使之不停监听传感器
try:
    while True:
        # 定义GPIO引脚及其工作模式
        mq2 = 40
        beep = 38
        relay = 36
        GPIO.setmode(GPIO.BOARD)
        GPIO.setup(mq2, GPIO.IN)
        GPIO.setup(beep, GPIO.OUT)
        GPIO.output(beep, 0)
        GPIO.setup(relay, GPIO.OUT)

        # 若获取到的MQ-2输入为高电平，则将蜂鸣器置为低电平
        if GPIO.input(mq2) == 1:
            GPIO.output(beep, 0)
        # 若获取到的MQ-2输入为低电平，则将蜂鸣器置为高电平触发，将继电器置为高电平闭合
        else:
            GPIO.output(relay, 1)
            GPIO.output(beep, 1)
            time.sleep(0.19)
            GPIO.output(beep, 0)
        time.sleep(0.01)
except KeyboardInterrupt:
    pass
# 关闭GPIO引脚监听，避免资源浪费
GPIO.cleanup()
```

图 5 本地应用程序（Sensor.py）代码

2. 本地应用程序 Web 服务

本地应用程序的 Web 服务由 Python 编写，应用程序负责与小程序前端进行交互，如本地应用程序返回温度，烟雾，继电器状态信息到前端；前端发送 Request 请求来控制本地继电器进而控制用电器。

本地应用核心程序（Web.py）代码如图 6 所示

```python
# 导入库文件
import time
import tornado.ioloop
import tornado.web
from tornado.httpclient import AsyncHTTPClient, HTTPRequest
import json
import RPi.GPIO as GPIO
import subprocess


# 远程获取信息Handler
class InfoHandler(tornado.web.RequestHandler):
    async def get(self):

        # 定义GPIO引脚及其工作模式
        mq2 = 40
        GPIO.setmode(GPIO.BOARD)
        GPIO.setup(mq2, GPIO.IN)

        #  创建字典及数组
        list = {}
        result = []
        try:
            # 获取18B20温度传感器
            cmd = "cd /sys/bus/w1/devices/28-800000208f22 && cat w1_slave | awk 'NR==2{printf \"%s\
            # 将命令放入子进程
            Evtmp = subprocess.check_output(cmd, shell=True)
            # 温度值精确到小数点后2位
            evtmp = str(round(int(str(Evtmp, encoding='utf-8').replace("t=", "")) / 1000, 2) - 1)
            # 将转换值放入list字典
            list['evtmp'] = evtmp

            # 获取MQ-2可燃气体传感器数据
            if GPIO.input(mq2) == 1:
            # 若获取到的MQ-2输入为高电平，则给list字典smog状态赋值'working'
                list['smog'] = 'working'
            else:
            # 若获取到的MQ-2输入为低电平，则给list字典smog状态赋值'danger'
                list['smog'] = 'danger'
        except KeyboardInterrupt:
            pass
        # 关闭GPIO引脚监听，避免资源浪费
        GPIO.cleanup()

        # 将字典存入数组并转换成JSON类型
        result.append(list)
        results = json.loads(json.dumps(result, ensure_ascii=False))

        # 将数据写入到response响应中
        self.write({
            'success': True,
            'result': results
        })


# 远程获取继电器状态Handler
class RelayInfoHandler(tornado.web.RequestHandler):
    async def get(self):

        # 定义GPIO引脚及其工作模式
```

```
        relay = 36
        GPIO.setmode(GPIO.BOARD)
        GPIO.setup(relay, GPIO.IN)

        #  创建字典及数组
        list = {}
        result = []
        try:
            # 获取继电器当前状态
            if GPIO.input(relay) == 1:
            # 若获取到的MQ-2输入为高电平，则给list字典relay状态赋值'on'
                list['relay'] = 'on'
            else:
            # 若获取到的MQ-2输入为低电平，则给list字典relay状态赋值'off'
                list['relay'] = 'off'
        except KeyboardInterrupt:
            pass
        # 关闭GPIO引脚监听，避免资源浪费
        GPIO.cleanup()

        # 将字典存入数组并转换成JSON类型
        result.append(list)
        results = json.loads(json.dumps(result, ensure_ascii=False))

        # 将数据写入到响应中
        self.write({
            'success': True,
            'result': results
        })

# 远程控制继电器状态Handler
class ControlHandler(tornado.web.RequestHandler):
    async def get(self):

        relay_switch = self.get_argument("relay")

        # 定义GPIO引脚及其工作模式
        relay = 36
        GPIO.setmode(GPIO.BOARD)
        GPIO.setup(relay, GPIO.OUT)


        if relay_switch == 'on':
        # 若获取的URL中继电器状态为'on'，则将继电器输出引脚置为高电平'
            GPIO.output(relay, 1)
        else:
        # 若获取的URL中继电器状态为'off'，则将继电器输出引脚置为低电平'
            GPIO.output(relay, 0)

        # 关闭GPIO引脚监听，避免资源浪费
        GPIO.cleanup()

        # 将数据写入到响应中
        self.write({
        'success': True,
        })
# 建立make_app函数，为域名创建路径并映射到各Handler
def make_app():
    return tornado.web.Application([
        (r"/info", InfoHandler),
        (r"/relayinfo", RelayInfoHandler),
        (r"/control", ControlHandler),
    ])


# 监听来自8888端口的请求
if __name__ == "__main__":
    app = make_app()
    app.listen(8888)
    tornado.ioloop.IOLoop.current().start()
```

图 6 本地应用核心程序（Web.py）代码

3.本地环境辅助应用程序服务

本地的 Tornado Web 框架负责将 Python 命令转发 HTTP 请求；Frp 反向代理应用负责将本地网络映射到公网，Supervisord 监听自启动应用负责在无人值守的情况下，在开机或宕机后能够自启动本地应用程序，Nignx1.8 Web 应用负责处理来自前端的 HTTP 请求。

3 设计步骤及方法

3.1 设计步骤

构建设计想法 → 掌握传感器工作原理 → 绘制电路图 → 连接开发板与各传感器→ 开发前端 →配置本地服务器环境→ 开发后端→ 本地测试→映射公网→远程交互测试 → 优化监听服务

3.2 设计方法

阶段一：设计并绘出 Raspberry Pi 3B+开发板与各传感器的连线设计图，标出各硬件基本信息，测算硬件的相应工作电压，电流等条件。了解掌握传感器的工作原理。

阶段二：通过绘制完成的设计图进行连线并测试硬件。

阶段三：基于 HTML5 + Javascript + CSS 开发前端。

阶段四：配置本地服务器环境，基于 Nignx1.8 Web 应用服务，Supervisord 监听自启动应用服务搭建 HTTP 服务器并使其自动化运行。

阶段五：基于 Python，Tornado Web 服务框架开发后端。

阶段六：基于 Frp 反向代理应用负责将本地网络映射到公网。

阶段七：本地进行交互测试，调试，修改，通过公网交互达到设计目的。

阶段八：不断调整程序，优化传感器监听服务，实现全自动预警。

4 设计所用器材

设计所用器材清单如表 1 所示。

序号	编号	名称	型号	数量
1	RPI	Raspberry Pi 3B+开发板	3B+	1
2	OLED	SSD1306 0.96 寸 OLED 显示屏	SSD1306	1
4	18B20	18B20 温度传感器	18B20	1
5	MQ-2	MQ-2 可燃气体传感器	MQ-2	1
6	Relay	1 路继电器	DC 5V	1
7	Beep	有源蜂鸣器	1206	1
8	LED	蓝色 LED 灯	DC 3-5V	1
9	FAN	风扇	DC 5V	1
10	R	电阻	4.7KΩ	1

表 1 元器件清单

5 小结(通过课程设计收获和心得)

⑴ 作品展示

作品实物如图 7，图 8 所示

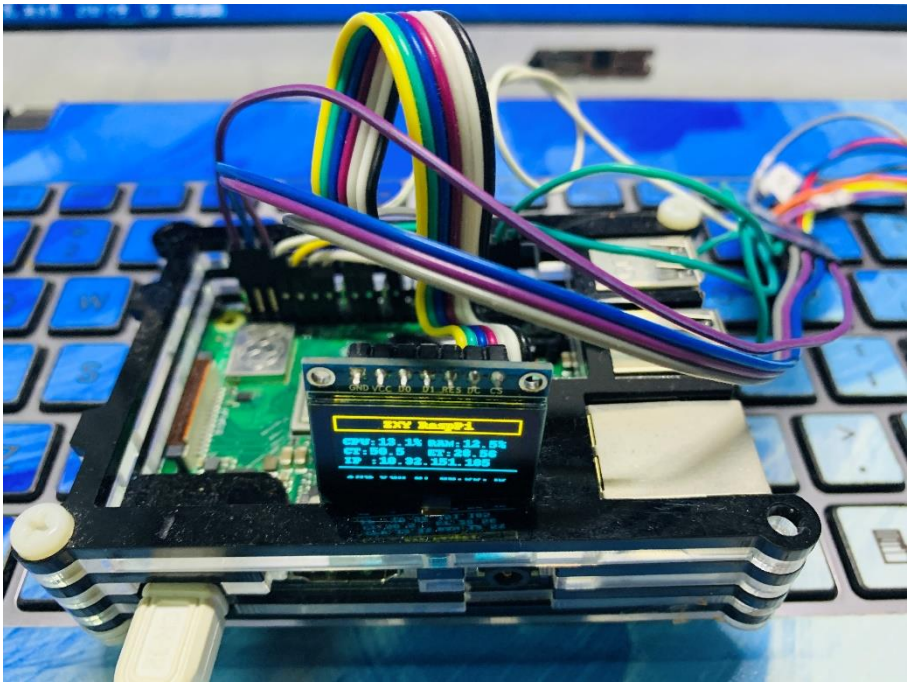


图 7 Raspberry Pi 3B+开发板

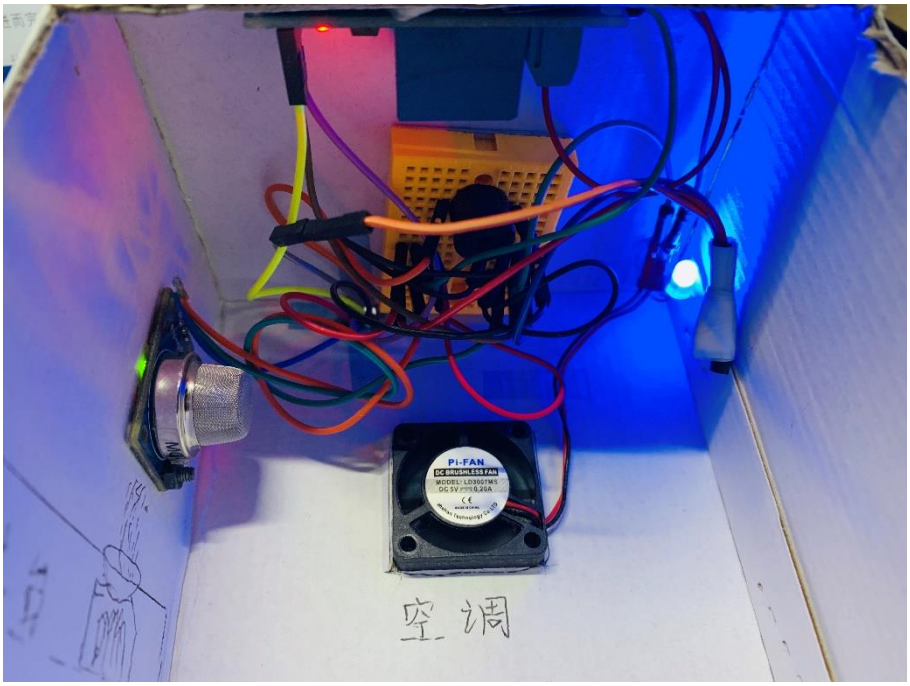


图 8 传感器等硬件

⑵ 设计收获与心得

通过此次课程设计，全面了解了传感器的工作原理及传感器与其它电路的交互方法，通过实践及开发经验，基于 HTML5 + Javascript + CSS 开发前端，基于 Python，Tornado Web 服务框架开发后端并配置云服务器工作环境，解决了传感器数据交互的延时问题，突破了内网访问的局限性，将内网映射到公网，通过公网让所有用户可以远程访问并控制。

通过此次课程设计，设计了一个能对可燃气体监测，做出判断，报警，远程监控的安全系统并且能通过互联网远程控制的继电器，继而让继电器控制其他电路的综合小规模实用型安防系统。系统的学习了各器件，编程，环境配置等知识，将其整合成一个具有实际意义及使用价值的产品。

参考文献

[1] 崔庆才 著.Python3 网络爬虫开发实战.人民邮电出版社.