智能硬件综合实践

**设计总结报告**

题目：简单危险感知报警通知系统

**设计者：**

**向萱岭 201531060552**

**郑博 201531060548**

**廖梦羽 201531060549**

**冉驰昊 201531060557**

**任启强 201531060559**

**孔维坤 201531060562**

**石华 201531060570**

**班 级：物联网工程1501**

**指导教师：梅大成**

**时间：2017年7月7日**

目录

第一章 课题简介 **3**

1.1背景介绍3

第二章 设计方案 **3**

2.1设计思路3

2.2 nodemcu平台介绍3

2.3 硬件设计原理4

2.4 软件流程图6

2.5 重要算法——mqtt通信7

2.5.1 简介7

2.5.2 mqtt相关原型函数9

2.5.3 mqtt通信Android端部分相关10

第三章 操作说明 **15**

3.1 操作说明15

第四章 调试信息和实验结果 **15**

第五章 实验总结 **15**

# 第一章 课题简介

* 1. 背景介绍

实验室、家庭等小空间存在着各种各样的安全隐患，如可燃气体泄漏、火灾等。而因为这些隐患爆发的时候并没有人察觉发现，就有可能由一开始比较小的伤害在没有制止的情况下会逐渐扩大。一直以来很多的大型火灾等意外都是因为没有及时发现并解决，导致比较多的财力、物力的损失。因此能够检测危险并及时通知的小系统就显得格外重要。这次我们组所选择的课题便是能够在一定程度上解决此类问题的简单危险感知报警通知系统。

# 第二章 设计方案

2.1 设计思路

本次选题的设计思路主要为，系统首先通过传感器检测危险信号（烟雾，火焰等传感器，我们因为设备等原因，用触摸传感器代替其他信号输入）。ESP8266（本次课题所选择的主要平台）判断信号为危险后点亮led（报警灯）。之后利用wifi模块所连接的网络，采用mqtt协议将“危险提醒”以topic的方式借助iot.eclipse.org所提供的公开broker进行发布，订阅该topic的pc或手机的mqtt客户端将会收到该“危险提醒”。若收到来自其他设备发布的“ok”消息，led灯变为闪烁状态，并停止持续（每隔10s）发布危险信息。

2.2 nodemcu平台介绍

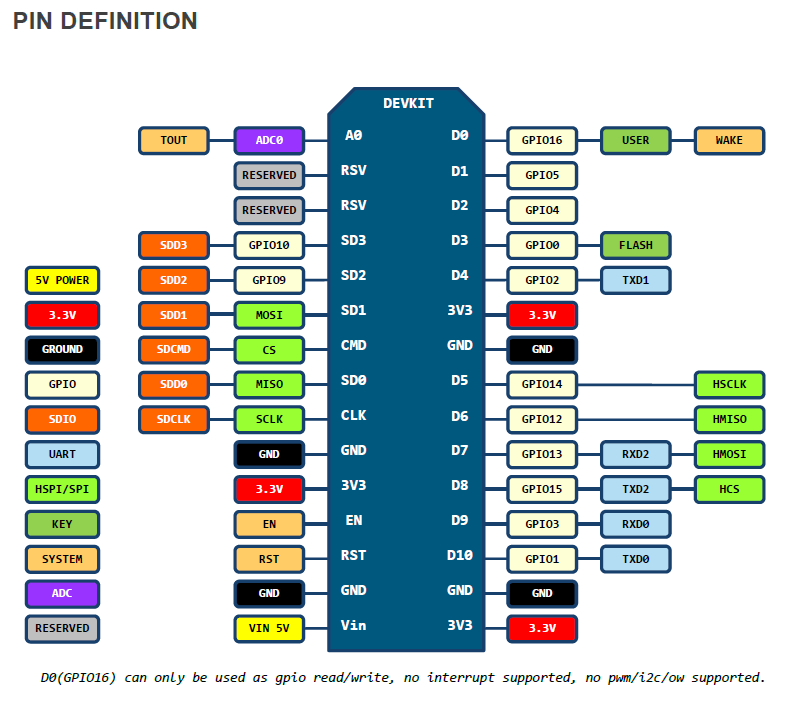
ESP8266是上海Espressif生产的一种小型WIFI芯片，该芯片旨在给嵌入式系提供接入网络的功能。NodeMCU是一个集成了ESP8266的开源开发板，并且封装了ESP8266的硬件操作命令，使用户可以更容易和硬件交互。

NodeMCU项目的发起人是黄锐，发起这个项目的时候他还是一名在读研究生。当时黄锐想把正在制作的脑电波传感器加上WiFi功能，便开始选型WiFi芯片。他在了解到ESP8266的内核LX106后，发现这个芯片性能颇为强大，比起畅销的MTK7681更胜一筹。基于ESP8266的脑电波传感器完成之后，黄锐决定再做点什么，来消除软件工程师和硬件工程师的隔阂。和开源社区爱好者zeroday商量后，他决定发起一个NodeMCU项目。NodeMCU的Firmware是由zeroday主导，并和黄锐共同完成的。

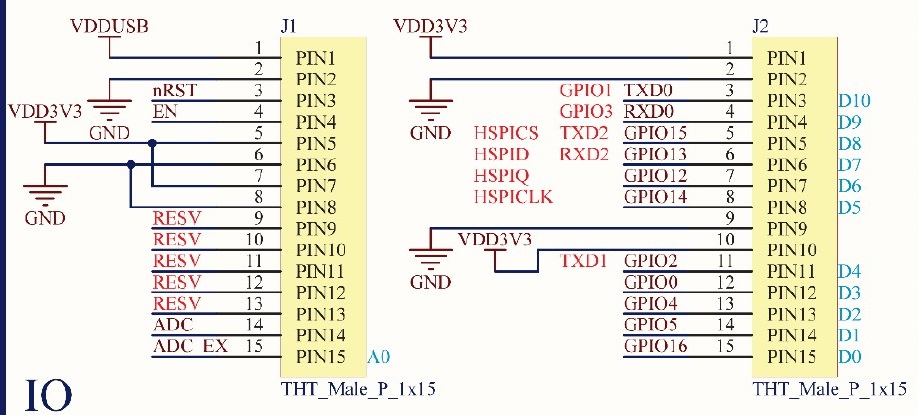
NodeMCU是目前最廉价的带WIFI(ESP8266)模块的开发板，如果有兴趣自己动手做智能家居，把家里的电器都联网，开发板的成本是一个重要的因素。其它的类似选择包括：SparkFun的Photon, ArduinoYun, Arduino + Zigbee, Arduino + ESP8266 ， Arduino + ethernet shield，Raspberry + WIFI 或者自带wifi的Raspberry 3. 然而这些开发板+通讯模块加在一起的价格至少是NodeMCU的5到10倍，所以NodeMCU具有明显的价格优势。

NodeMCU自带LUA的解释器，所以可以用LUA语言给其编写控制程序，LUA是一个轻量级的脚本语言，用C语言编写，编译后解释器大小仅仅100多K。LUA旨在针对内存有很大限制的嵌入式系统。NodeMCU还提供交互命令行环境，比Arduino的编程调试环境友好。 NodeMCU也支持Arduino的IDE对其编程。

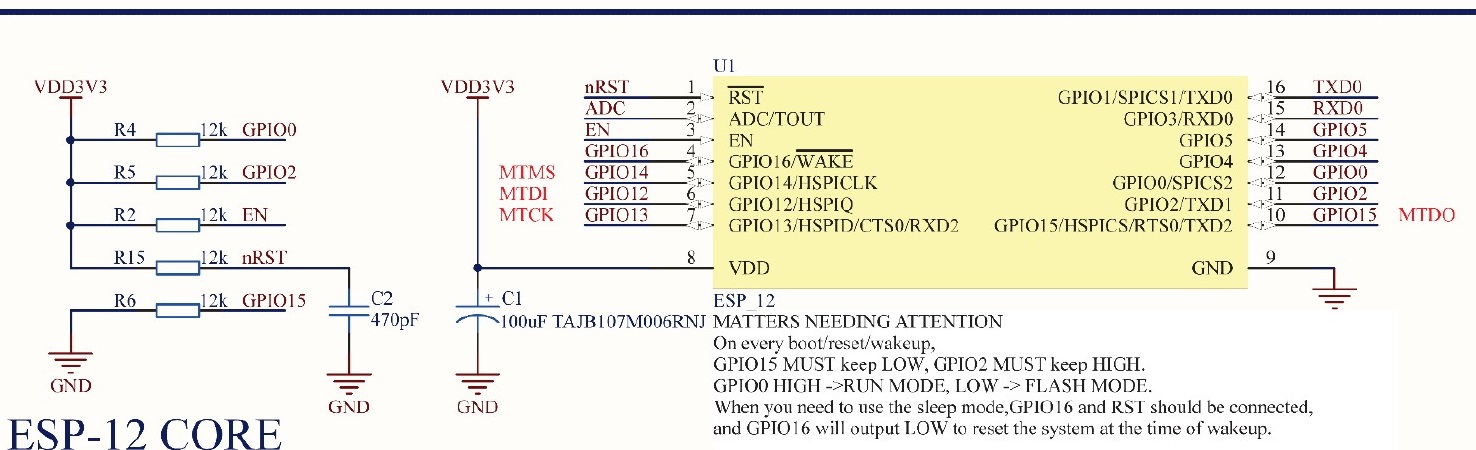
2.3 硬件设计原理



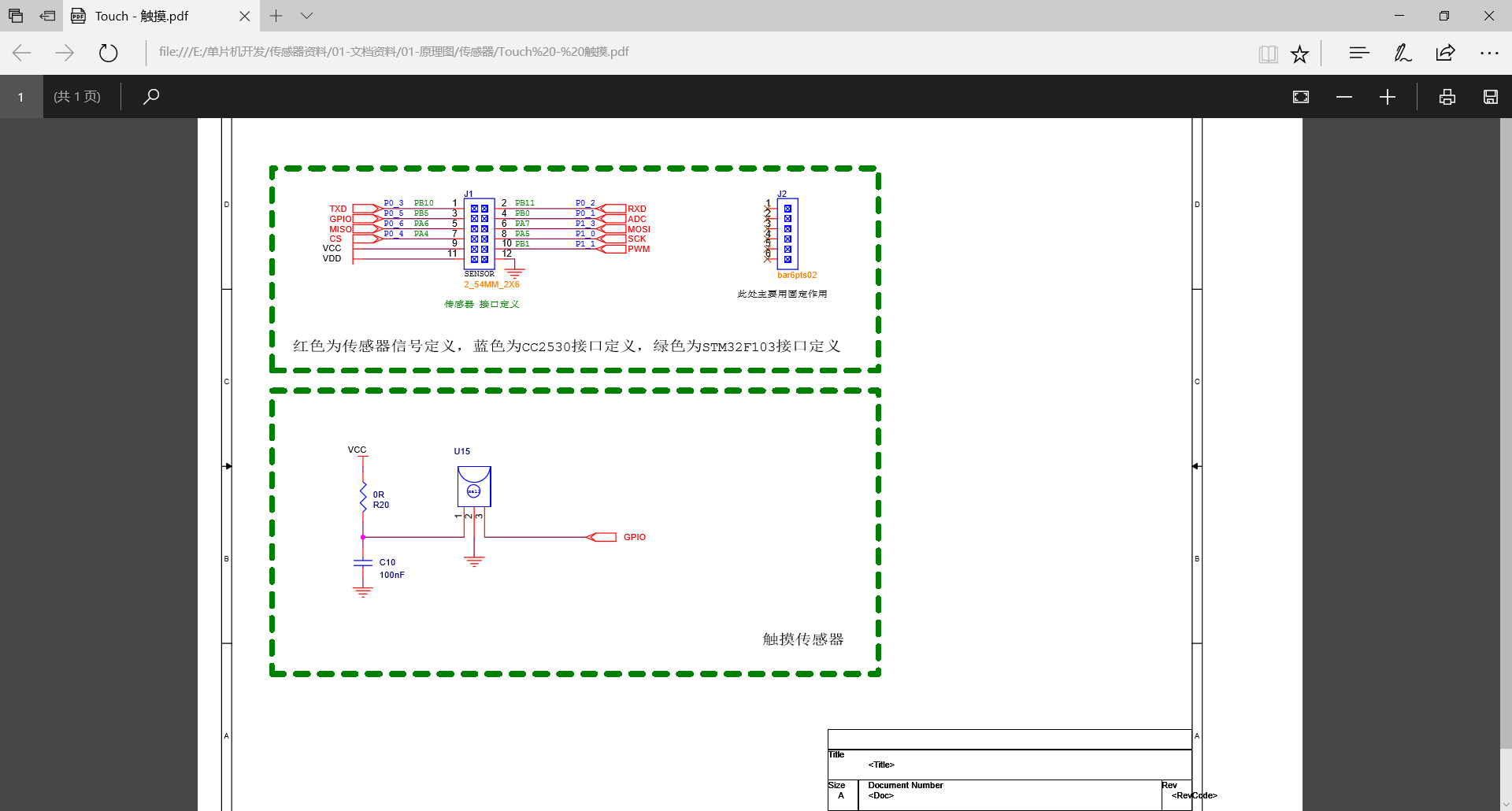
nodemcu v1.1（ESP8266）管脚说明



IO口电路图



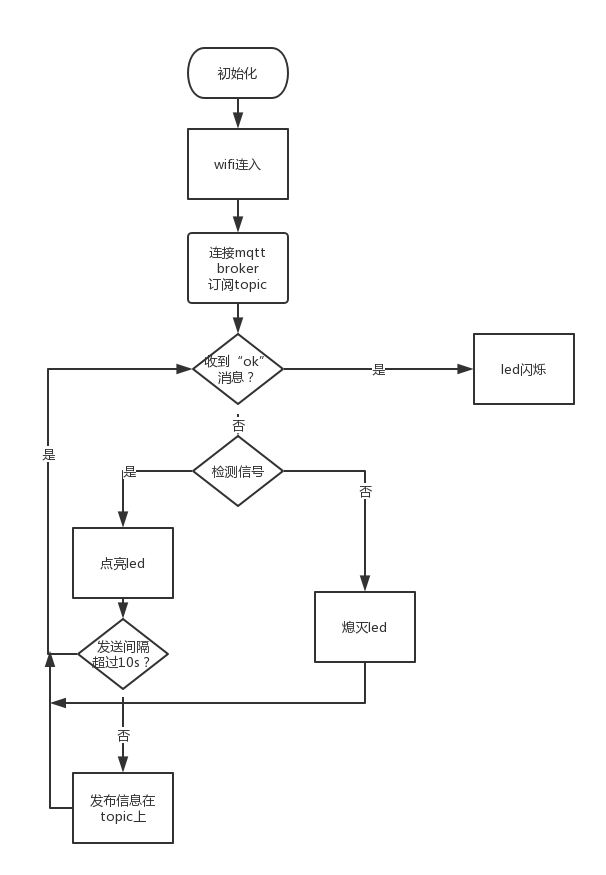
Wifi核心电路图



触摸传感器电路图

注：本课题将触摸传感器（代替火焰、可燃气体传感器等）的gpio管脚接在nodemcu的gpio05（d1）上，led灯在gpio02（d4）上。

2.4 软件流程图



2.5 重要算法——mqtt协议通信

2.5.1 简介

本课题重要的算法在于使用mqtt协议通信。Mqtt因为其开销小、一对多等特点，非常适合作为物联网的通信手段。并且利用wifi模块连接网络，借助开放broker（mqtt代理），能够实现其他设备无论距离，接入网络就能进行通信，以及简单的增加待通知设备。

早在1999年，IBM的Andy Stanford-Clark博士以及Arcom公司ArlenNipper博士发明了MQTT（Message Queuing Telemetry Transport，消息队列遥测传输）技术[1] 。据Andy Stanford-Clark博士称，MQTT将在今年和明年呈现爆炸式增长。

MQTT的话题是我俩谈论开源物联网平台Pachube时提到的。Stanford-Clark认为Pachube很酷，其不足之处是不具备真正的推送功能。你需要不断轮询才能得到即时数据。这正是MQTT能够实现的，他提到了使用推送通信系统的石油管道检测系统。

MQTT协议是为大量计算能力有限，且工作在低带宽、不可靠的网络的远程传感器和控制设备通讯而设计的协议，它具有以下主要的几项特性：

1、使用发布/订阅消息模式，提供一对多的消息发布，解除应用程序耦合；

2、对负载内容屏蔽的消息传输；

3、使用 TCP/IP 提供网络连接；

4、有三种消息发布服务质量：

“至多一次”，消息发布完全依赖底层 TCP/IP 网络。会发生消息丢失或重复。这一级别可用于如下情况，环境传感器数据，丢失一次读记录无所谓，因为不久后还会有第二次发送。

“至少一次”，确保消息到达，但消息重复可能会发生。

“只有一次”，确保消息到达一次。这一级别可用于如下情况，在计费系统中，消息重复或丢失会导致不正确的结果。

5、小型传输，开销很小（固定长度的头部是 2 字节），协议交换最小化，以降低网络流量；

6、使用 Last Will 和 Testament 特性通知有关各方客户端异常中断的机制；

2.5.2 mqtt相关原型函数

mqtt.Client()

####描述 创建一个mqtt client.

####语法 mqtt.Client(clientid, keepalive, user, pass)

####参数 clientid: mqtt客户端id.

keepalive: 保持连接的时间，单位：秒.

user: 用户名，字符串.

pass: 密码，字符串.

####返回值 mqtt客户端.

mqtt:connect()

####描述 连接到mqtt服务器.

####语法 mqtt:connect( host, port, secure, function(client) )

####参数 host: 主机域名或者ip地址，字符串类型.

port: 服务器端口号.

secure: 0 或者 1, 默认值为0.

function(client): 连接成功的回调函数.

####返回值 nil.

mqtt:publish()

####描述 发布一个消息.

####语法 mqtt:publish( topic, payload, qos, retain, function(client) )

####参数 topic: 需要发布消息的标题, 字符串类型.

message: 需要发布的消息, 字符串类型.

qos: qos值, 默认值为0.

retain: 保留标志, 默认值为0.

function(client): 发送成功的回调函数，如果接收到PUBACK回调函数解除.

####返回值 nil.

mqtt:subscribe()

####描述 订阅一个或者多个标题的消息.

####语法 mqtt:subscribe(topic, qos, function(client, topic, message))

####参数 topic: 需要订阅消息的标题.

qos: 订阅消息的qos值, 默认值为0

function(client, topic, message): 接收消息的回调函数，接收后即解除.

####返回值 nil.

mqtt:on()

####描述 注册mqtt事件的回调函数.

####语法 mqtt:on(event, function(client, [topic], [message]))

####参数 event: 字符串，取值为: "connect", "message", "offline"

function cb(client, [topic], [message]): 事件触发的回调函数. 第一个参数是mqtt client.

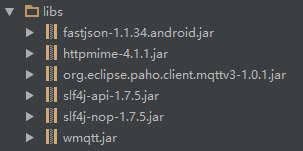
如果事件是"message", 第二个和第三个参数分别是标题和消息内容，字符串类型.

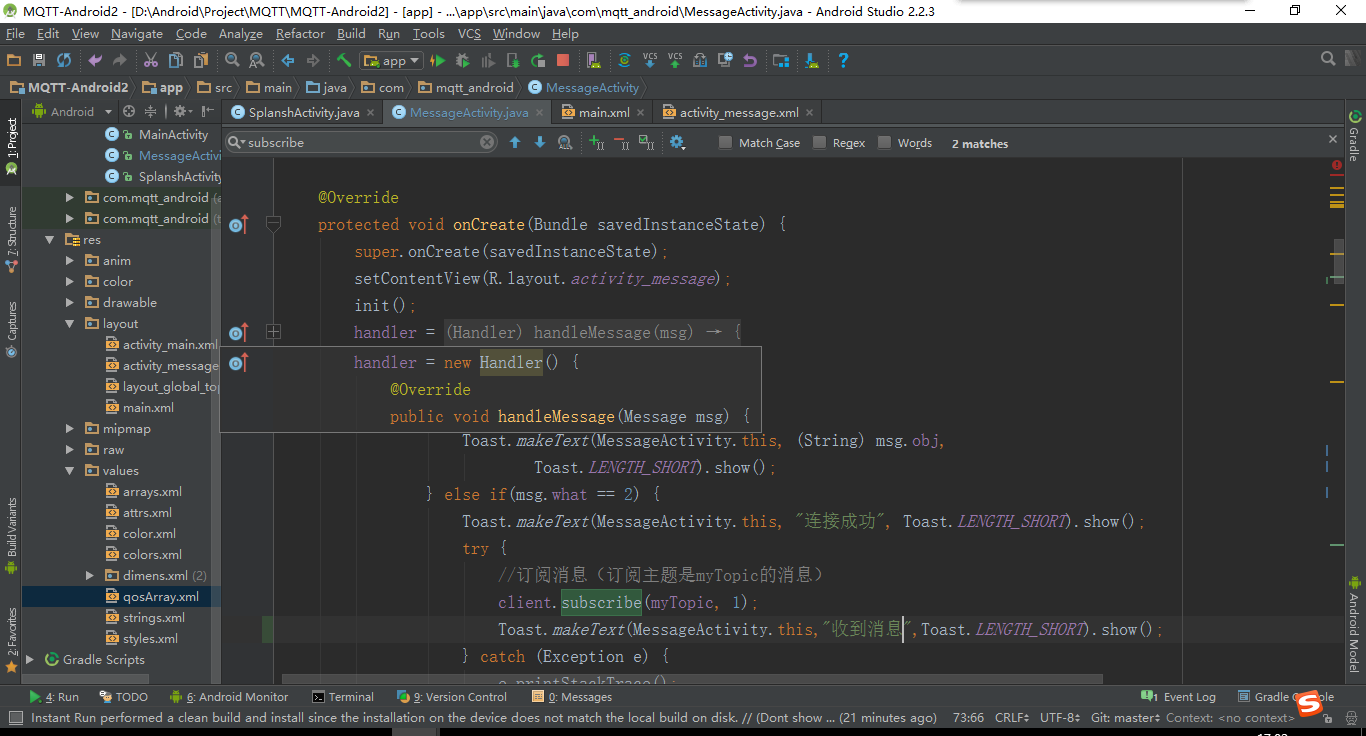
####返回值 nil.

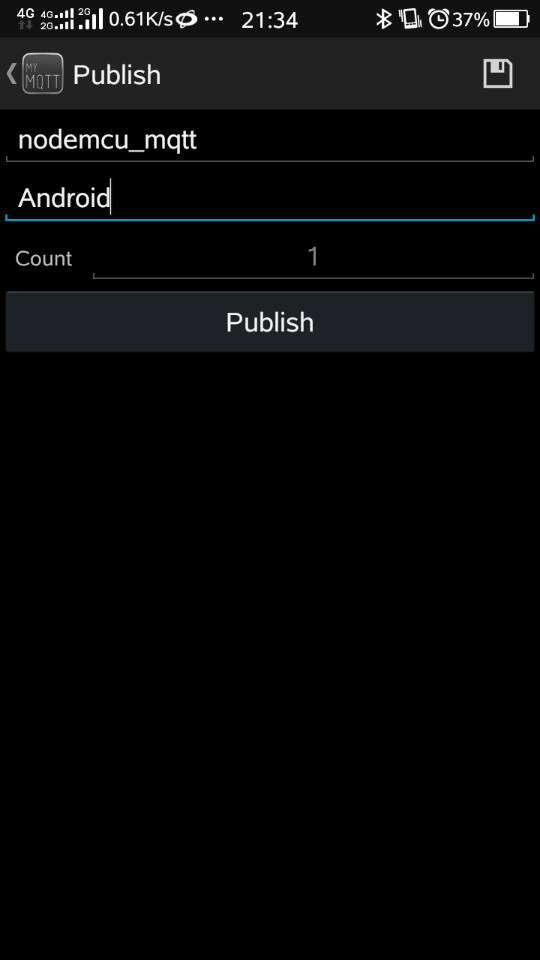
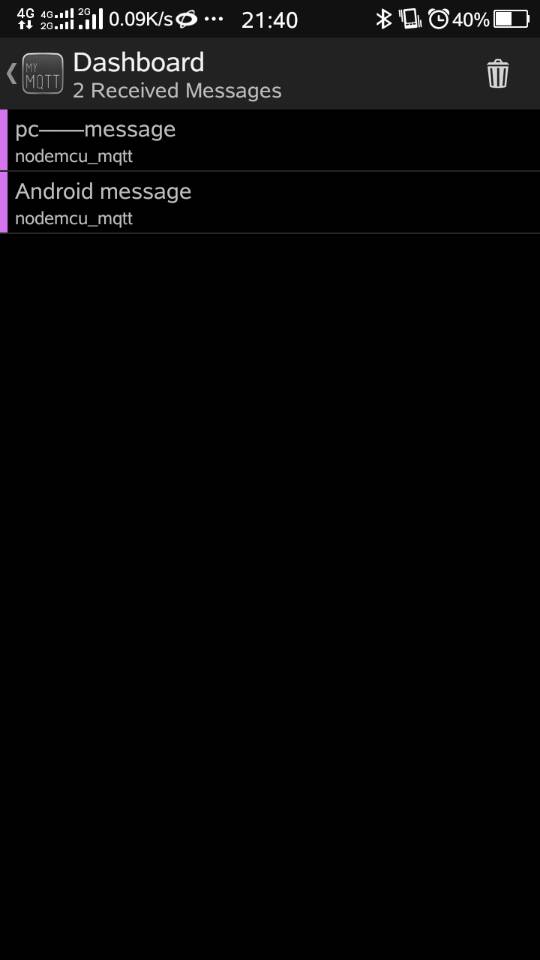
2.5.3 mqtt通信Android端部分相关

**通过引用第三方的包** org.eclipse.paho.client.mqttv3**可以快速使用MQTT。**

**引用的包：**

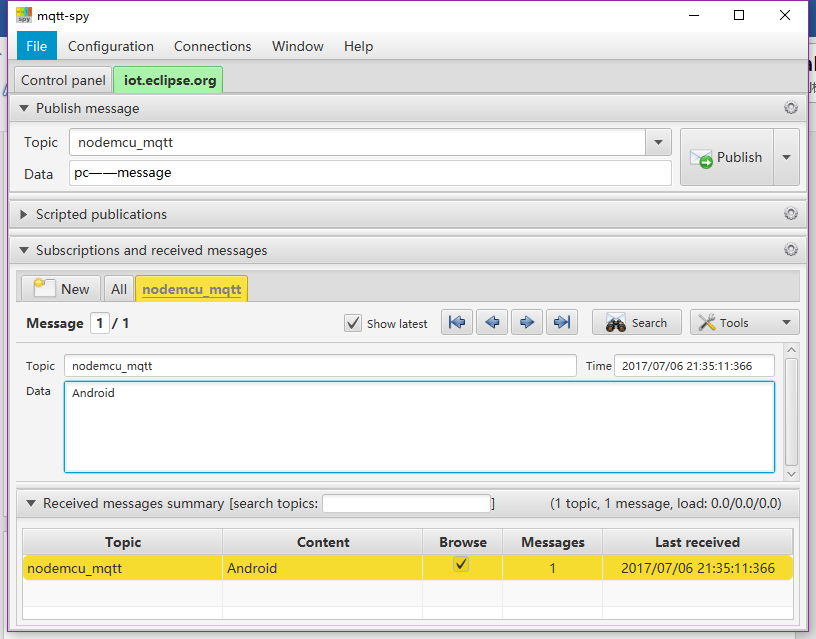




**Android接受消息，传感器监测到危险可以随时查看。**

**Android端发送消息，设备可以根据发送的指令，进行相应的处理。以达到控制的目的。**



**代码：**

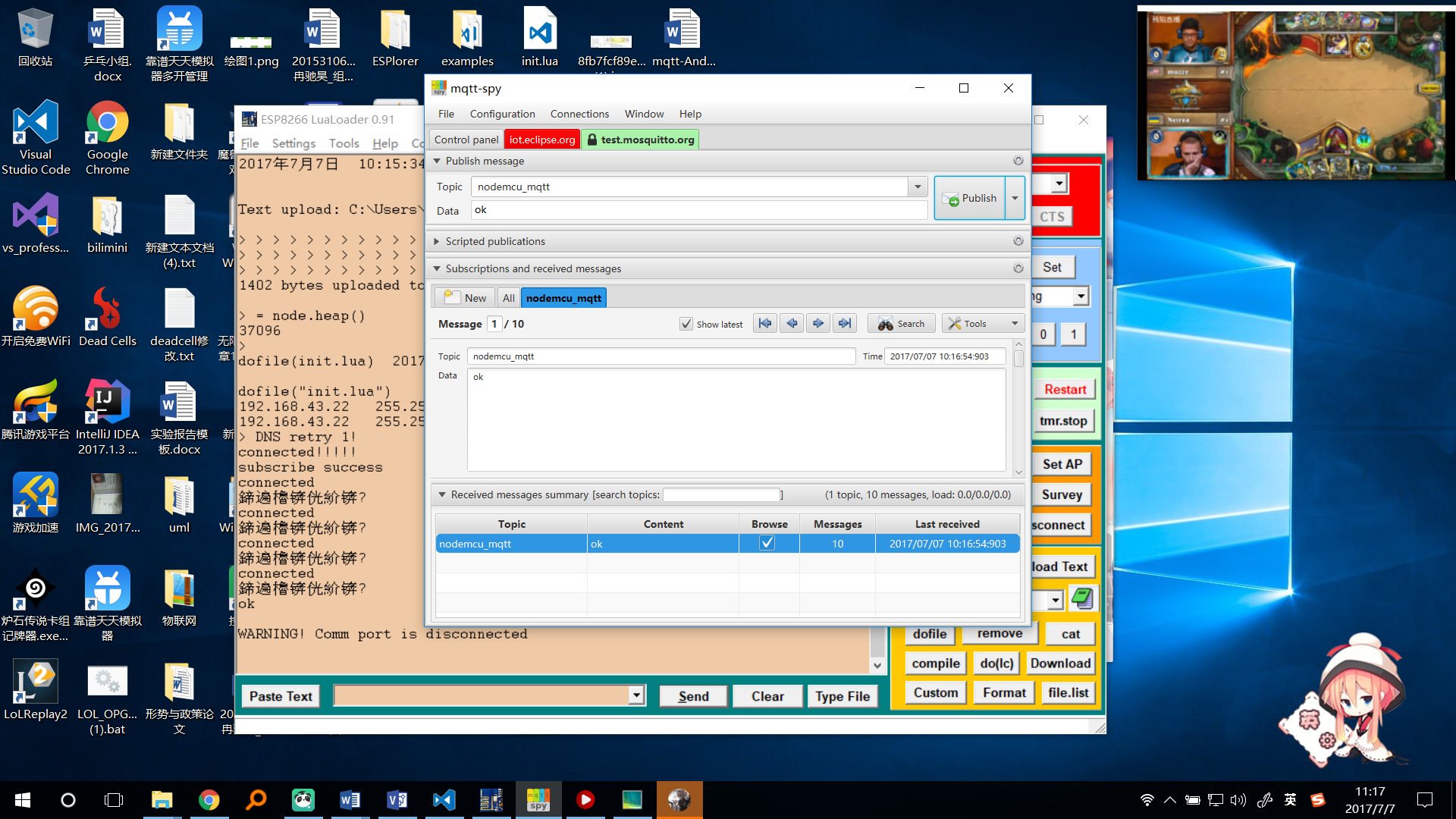
@Override  
protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_message*);  
 init();  
 handler = new Handler() {  
 @Override  
 public void handleMessage(Message msg) {  
 super.handleMessage(msg);  
 if(msg.what == 1) {  
 //接收订阅消息  
 Toast.*makeText*(MessageActivity.this, (String) msg.obj,  
 Toast.*LENGTH\_SHORT*).show();  
 } else if(msg.what == 2) {  
 Toast.*makeText*(MessageActivity.this, "连接成功", Toast.*LENGTH\_SHORT*).show();  
 try {  
 //订阅消息（订阅主题是myTopic的消息）  
 client.subscribe(myTopic, 1);  
 Toast.*makeText*(MessageActivity.this,"收到消息",Toast.*LENGTH\_SHORT*).show();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 } else if(msg.what == 3) {  
 Toast.*makeText*(MessageActivity.this, "连接失败，系统正在重连", Toast.*LENGTH\_SHORT*).show();  
 }else if(msg.what == 4){  
 Log.*i*("lymain","发布信息：QOS="+qos+"; topic="+sendTopic.getText().toString()+"; 内容："+sendMessage.getText().toString());  
 try{  
 //发布信息 参数说明（主题，内容，qos，是否保留）  
 client.publish(sendTopic.getText().toString(),sendMessage.getText().toString().getBytes(),qos,false);  
 }catch (Exception e){  
  
 }  
  
 }  
 }  
 };  
 startReconnect();  
  
 btnSendMessage.setOnClickListener(this);  
}

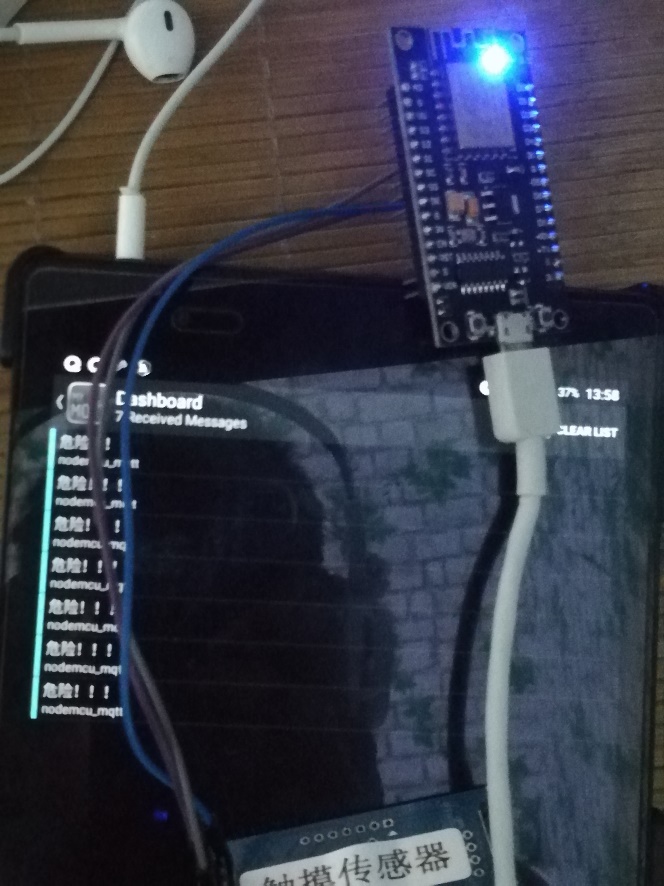
# 第三章 操作说明

2.1 操作说明

系统连上电源后会自动连接wifi和公开broker，同时开始检测传感信号，检测到传感信号后会每10s对所有客户端进行通知，并常亮灯；而在收到“ok”的回复信息后，将会停止通知并且灯开始闪烁。

# 第四章 调试信息和实验结果





# 第五章 实验总结

通过本次硬件实践，我们小组对设计一个综合的项目有了真实的体会，让之前课堂上的理论知识得到实际的操作，对物联网项目设计的总体流程和思路有了更为客观的认识。

在实践中我们遇到了很多问题，从不知从何下手，到做出一个真正的东西，之间的历程是我们小组每个人最大的收获。从小组讨论选题，到开发所遇到的问题 ，我们每个人都绞尽脑汁，尝试各种办法，最后成功完成了项目。从中也知道了，在项目开发中，团队合作的重要性，单靠一个人是不能完成的，只有每个人都竭尽全力，才能使得项目成功的进行下去。

我们也认识到，课堂上的理论知识和实际操作之间必须通过这样的实践才能真正连接起来。只有真正的自己动手，才能体会到其中的原理。所以以后一定会多动手实践，把理论和实践相结合，真正提高自己的能力！

附录：代码

cout = 0

f1 = 0

f = 0

function wifi\_link() --连接wifi

gpio.mode(4,gpio.OUTPUT)

print(wifi.sta.getip())

--nil

wifi.setmode(wifi.STATION)

wifi.sta.config("huan","hahahahahah")

print(wifi.sta.getip())

end

function mqtt\_re() --连接mqtt公开broker及订阅topic

m1 = mqtt.Client("Node\_MM", 120) -- 创建客户端

m1:on('message',function(client,topic,data) --接收到消息后判断消息内容

  print(data)

  if(data == "ok")

  then

    f1 = 1

  end

  end

)

m1:connect("test.mosquitto.org", --建立连接

function(client)

print("connected!!!!!")

m1:subscribe("nodemcu\_mqtt",0,

  function(conn)

  print("subscribe success")

  end

)

end,

function(client, reason)

print("fail reason" .. reason)

end

)

end

function mqtt\_pin(s) --连接mqtt公开broker及发布消息

m = mqtt.Client("Node\_NN", 90) --同上

m:connect("test.mosquitto.org",

function(client)

print("connected")

m:publish("nodemcu\_mqtt", s , 0, 0) --发布消息

end,

function(client, reason)

print("fail reason" .. reason)

end

)

m:close()

end

function pin\_state() --主要判断状态执行函数

if((gpio.read(1)==0) and (f1==0))--检测到信号

then

if(f==0)

then

f = 1;

mqtt\_pin("危险！！！")

end

gpio.write(4,gpio.LOW)--led亮

else

gpio.write(4,gpio.HIGH)--led灭

end

cout = cout+1

if(cout == 100)--计数，有信号时每10s发布一次消息

then

cout = 0

f = 0

end

  if(f1==1)--灯光闪亮

  then

    gpio.write(4,gpio.LOW)

    for i=1000,1,-1 do

    end

    gpio.write(4,gpio.HIGH)

    end

end

wifi\_link()

mqtt\_re()

tmr.alarm(0, 100, tmr.ALARM\_AUTO, pin\_state) --计数器自动循环延时函数