开发环境：eclipse

环境的搭建以及工程的配置可参考安信可官网<https://www.ai-thinker.com/home>

SDK：ESP8266\_RTOS-2.0.0 基于freertos操作系统

软件代码实现功能介绍：

1 给wifi配置网络

先将wifi配置成ap模式，让该wifi形成局域网服务功能，通过手机或笔记本的web网页访问该wifi，输入当前可用路由器的用户名和密码给该wifi。然后wifi设置成station模式，通过该用户名和密码连接上路由器。

2 获取公司的mqtt服务器的用户名和密码等信息

创建socket套接字，通过URL解析出域名和端口，并填入sockaddr\_in结构体中，之后connect和send，注意发送时，使用C语言拼接时产生的格式问题，从recv返回的字符串中获取用户名和密码等信息。

3 连接mqtt服务器

填入连接信息，也可添加mqtt的遗愿设置，然后发起请求。

连接成功后，要设置好订阅和发布的主题以及内容，为了加密信息传输，使用tls连接。

4 数据的传输格式

采用messagepack，因为它比json更小。从官网下载源码，通过xtensa-lx106-elf-gcc工具链编译成库，加入到工程中，通过头文件调用函数，拼接成协议中的数据格式，进行传输。

帧长等于接受与发送的长度

数据帧操作流程：设备----WIFI---APP

设备上报：设备到WiFi，WiFi应答正常回发原帧给设备并上传APP；应答错误回发错误帧

APP下发：APP到WiFi，WiFi应答正常回发原帧给APP并下发设备；应答错误回发错误帧；

若引发设备状态改变，还需运行“设备上报”操作。

在初始化中注册回调函数

wifi\_set\_event\_handler\_cb(wifi\_event\_handler\_cb);

此回调函数会定时调用，当获得稳定的ip连接时，停止调用（或者理解成当模块状态发生改变时，会进入回调函数）。注意不能用户代码里while(1){}，否则回调函数失败。包括创建任务xTaskCreate()的初始化函数中也不能有while(1){}，在任务函数里只能有一个for(;;){}，在for(;;){}不能有while(1)，不然优先级低的任务将不能执行。

void wifi\_event\_handler\_cb(System\_Event\_t \*evt)

模块连接AP过程表述：

自定义配置WiFi的SSID、PASSWORD。当SSID或PASSWORD错误时，可不断进入事件处理函数，匹配宏*EVENT\_STAMODE\_DISCONNECTED，*模块不断重新扫描，试图连接。当SSID或PASSWORD正确时，可先匹配宏EVENT\_STAMODE\_CONNECTED正在连接，在匹配宏*EVENT\_STAMODE\_GOT\_IP*以连接并获取ip地址。

ESP8266-RTOS-SDK-2.0.0初始打印输出

ets Jan 8 2013,rst cause:2, boot mode:(3,6)

load 0x40100000, len 30328, room 16

tail 8

chksum 0x37

load 0x3ffe8000, len 2008, room 0

tail 8

chksum 0x48

load 0x3ffe87e0, len 632, room 0

tail 8

chksum 0x1f

csum 0x1f

OS SDK ver: 2.0.0(e271380) compiled @ Mar 30 2018 18:54:06

phy ver: 1055\_1, pp ver: 10.7

rf cal sector: 1019

tcpip\_task\_hdl : 40107a00, prio:10,stack:512

idle\_task\_hdl : 40107ab0,prio:0, stack:384

tim\_task\_hdl : 40107bf8, prio:2,stack:512

之后进入**void** **user\_init**(**void**){}初始化配置入口。

（默认打印UART0,波特率74880，此时可以使用UART\_SetPrintPort(*UART1*)改变打印的串口。

os\_printf()和printf()的区别：os\_printf()配置 ld/eagle.app.v6.common.ld目录下.irom.text的参数，减小打印字符串所占用的 dRAM 空间，间接增大了可用的 heap 空间。）

自定义配置：

打印输出配置到uart1，波特率74880；与MCU通讯使用uart0，波特率9600，配置函数uart\_init\_new()，使用中断服务函数处理接受与发送。

设备与模块捂手：

48 16 FE 01 21 01 71 7B 6B AD 3F 9A FB 25 76 28 C3 5B BD 06 E2 DD

48 07 FE 01 01 00 4F

模块回应：

48 16 FE 01 21 01 71 7B 6B AD 3F 9A FB 25 76 28 C3 5B BD 06 E2 DD

48 0B FE 01 01 01 01 01 05 00 5B

串口函数运行机制，上电初次运行输出empty。使用串口调试助手“七个字符以下，输出tout;

8个到10个输出full，每次满十个，再进一次中断处理函数处理一次。此时需要将几次中断处理的数据连接到一起，接收结束还会加两个字节\t\n。使用一个全局变量，rev\_length++，然后另起一个任务，每次进串口中断启动一个50ms的软件定时器任务，50ms后清除rev\_length的长度即可。（使用定时器定时清除长度是比较好的方案）

目前MQTTPublish()发布函数只能跑在mqtt\_client\_thread()线程中，具体为什么不能单独拿出来使用，原因有待查证？处理方法：当MQTT连接成功后，使MQTTPublish()进入死循环。在单独写一个函数，封装MQTTPublish()的功能，然后在while(1)不断检测Publish()是否满足条件，满足条件就发布。直接调用publish()，感觉比设置标志位要好。

xTaskCreate创建一个任务mqtt\_client\_thread()，进行mqtt的初始化，订阅与发布。创建成功后订阅接受依然有效，而发布函数目前不能使用，原因有待查证。

**char** mqttBuf[256] = { 0 }; //考虑改成队列

**int** mqttLen = 0;

**void** **Publish**(**char**\* str, **int** len) {

memcpy(mqttBuf, str, len);

mqttLen = len;

}

客户端CA证书：在example/openssl\_client下，先运行gencrt.sh在运行genheader.sh，最后运行gen\_misc.sh。会生成证书ssl\_client\_crt.h

客户端公私钥生成：在tools下，运行makefile.sh会生成cer.h和private