

**嵌入式系统实验报告**

**基于STM32的网络编程**

|  |  |
| --- | --- |
| 专 业 名 称 | 控制工程6班 |
| 学 生 姓 名 |  |
| 学 号 |  |

目录

[一、 硬件简介 3](#_Toc70942676)

[1、STM32F1 3](#_Toc70942677)

[2、ENC28J60 3](#_Toc70942678)

[3、图片 3](#_Toc70942679)

[二、 环境搭建 4](#_Toc70942680)

[1、LWIP带FreeRTOS操作系统移植 4](#_Toc70942681)

[2、MQTT移植 6](#_Toc70942682)

[三、 主要程序 7](#_Toc70942683)

[四、 程序运行结果 11](#_Toc70942684)

[五、 实验总结 14](#_Toc70942685)

# 基于STM32的网络编程

**摘 要**：本次网络编程实验使用STM32搭载FreeRTOS操作系统与LWIP协议实现接入阿里云物联网平台，最终的效果为通过阿里云在线调试实现控制板载LED的亮灭。本设计用到的硬件是正点原子STM32F1精英版、芯片型号STM32F103ZET6，没有板载网卡、采用外接网卡ENC28J60模块。

关键词：STM32 阿里云 FreeRTOS

引言

Cortex-M3采用ARM V7构架，不仅支持Thumb-2指令集，而且拥有很多新特性。较之ARM7 TDMI，Cortex-M3拥有更强劲的性能、更高的代码密度、位带操作、可嵌套中断、低成本、低功耗等众多优势。

国内Cortex-M3市场，ST（意法半导体）公司的STM32无疑是最大赢家，作为Cortex-M3内核最先尝蟹的两个公司（另一个是Luminary（流明））之一，ST无论是在市场占有率，还是在技术支持方面，都是远超其他对手。在Cortex-M3芯片的选择上，STM32无疑是大家的首选。

RTOS全称是Real Time Operating System，中文名就是实时操作系统。FreeROTS就是一个免费的RTOS类系统。这里要注意，RTOS不是指某一个确定的系统，而是指一类系统。比如UCOS，FreeRTOS，RTX，RT-Thread等这些都是RTOS类操作系统。

LWIP是瑞典计算机科学院(SICS)的Adam Dunkels等开发的一个小型开源的TCP/IP协议栈。LWIP是轻量级IP协议，有无操作系统的支持都可以运行，LWIP实现的重点是在保持TCP协议主要功能的基础上减少对RAM 的占用，它只需十几KB的RAM和40K左右的ROM就可以运行，这使LWIP协议栈适合在低端的嵌入式系统中使用。

本设计通过在STM32上移植FreeRTOS和LWIP协议栈，实现接入互联网并接入阿里云物联网平台，实现MQTT通讯。

# 硬件简介

## 1、STM32F1

STM32F103ZETT6是STM32F103里面配置非常强大的了，它拥有的资源包括：64KB SRAM、512KB FLASH、2个基本定时器、4个通用定时器、2个高级定时器、2个DMA控制器（共12个通道）、3个SPI、2个IIC、5个串口、1个USB、1个CAN、3个12位ADC、1个12位DAC、1个SDIO接口、1个FSMC接口以及112个通用IO口。该芯片的配置十分强悍，并且还带外部总线（FSMC）可以用来外扩SRAM和连接LCD等，通过FSMC驱动LCD，可以显著提高LCD的刷屏速度，是STM32F1家族常用型号里面，最高配置的芯片了。

## 2、ENC28J60

ENC28J60是带有行业标准串行外设接口（Serial Peripheral Interface，SPI）的独立以太网控制器。它可作为任何配备有SPI 的控制器的以太网接口。ENC28J60符合IEEE 802.3的全部规范，采用了一系列包过滤机制以对传入数据包进行限制。它还提供了一个内部DMA模块，以实现快速数据吞吐和硬件支持的IP校验和计算。与主控制器的通信通过两个中断引脚和SPI 实现，数据传输速率高达10 Mb/s。两个专用的引脚用于连接LED，进行网络活动状态指示。

## 3、图片



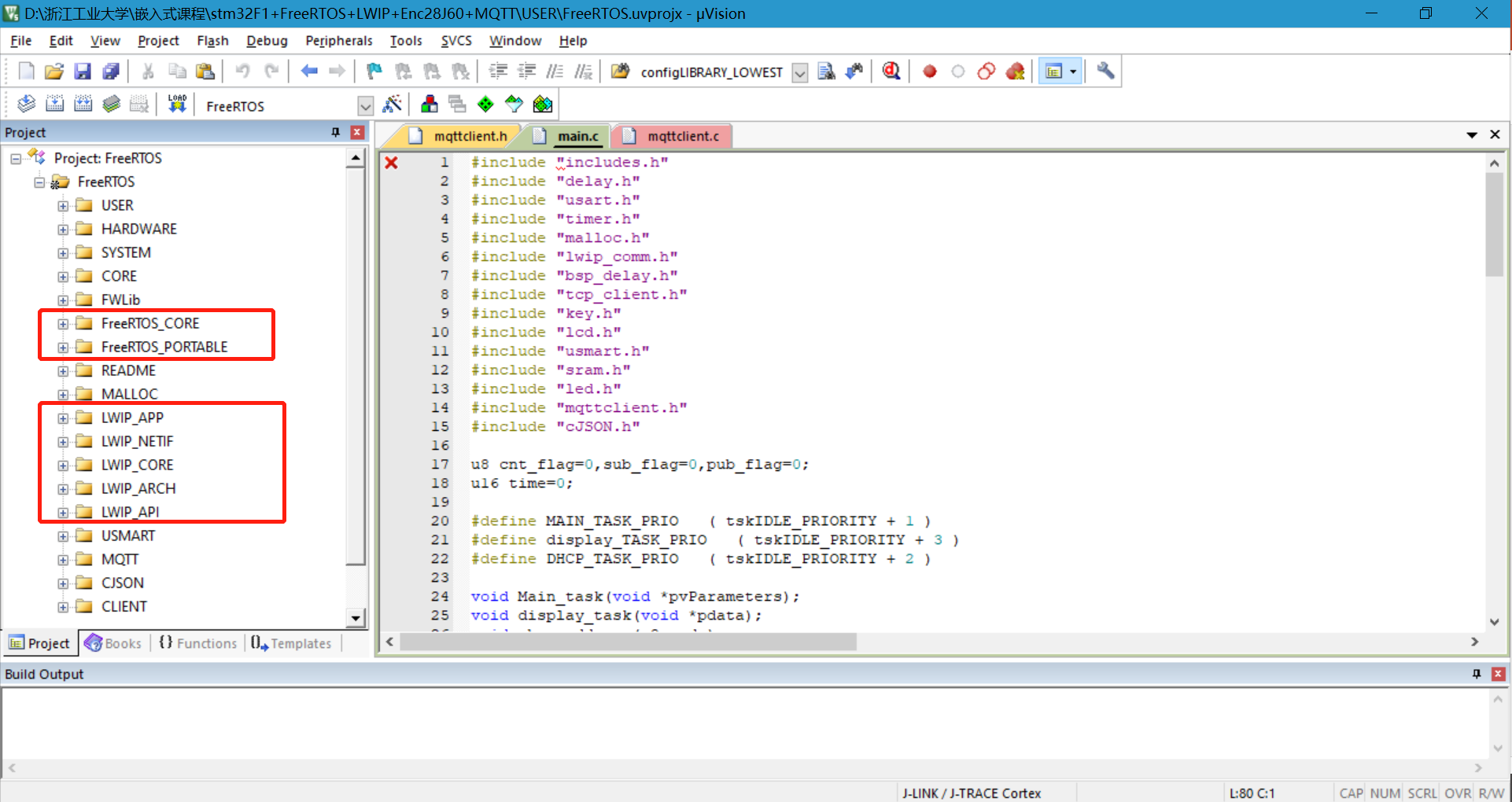
# 环境搭建

## 1、LWIP带FreeRTOS操作系统移植

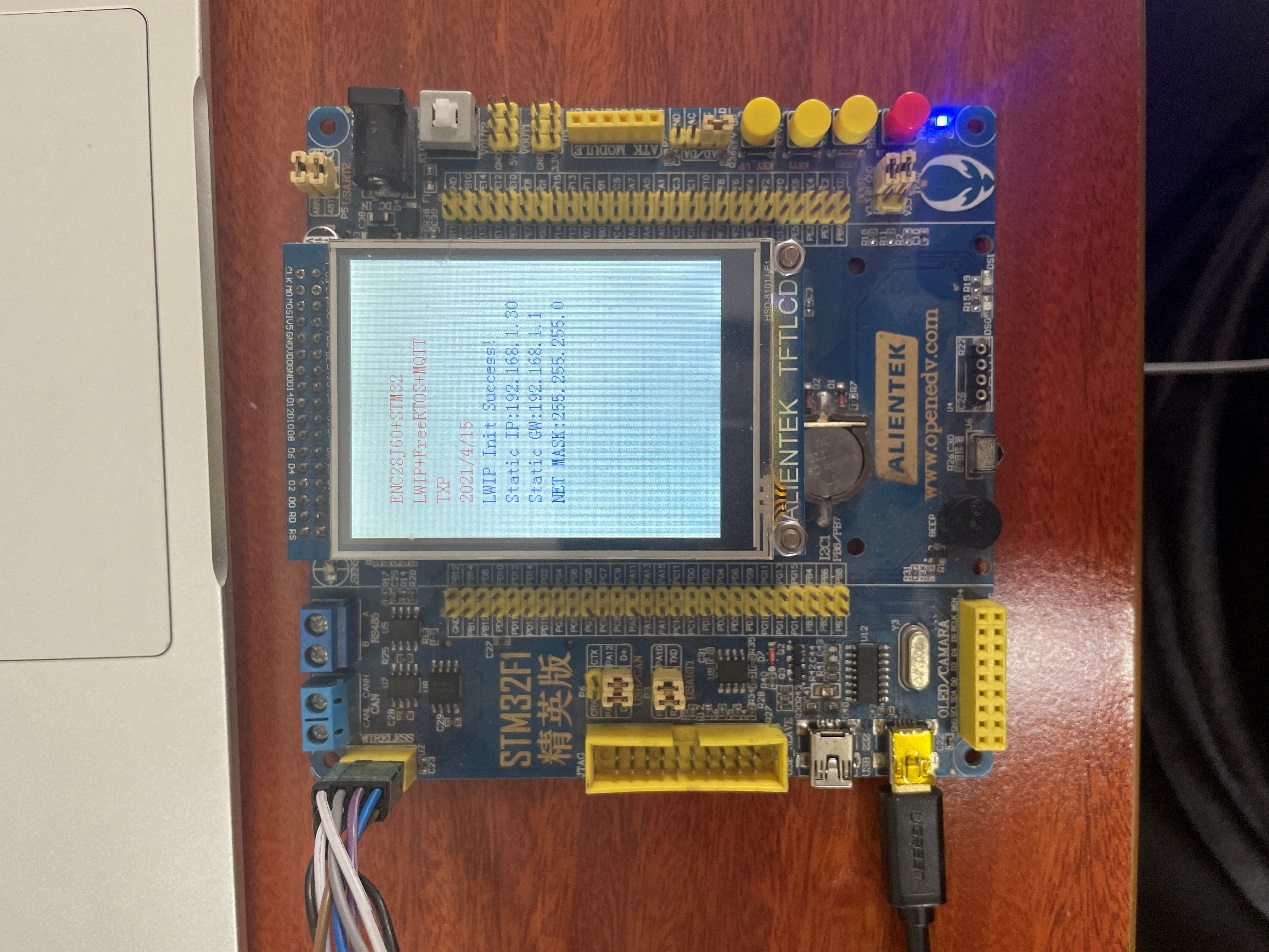
由于STM32F1在cubemx没有eth选项，所以无法通过cubemx快速配置本实验示例。

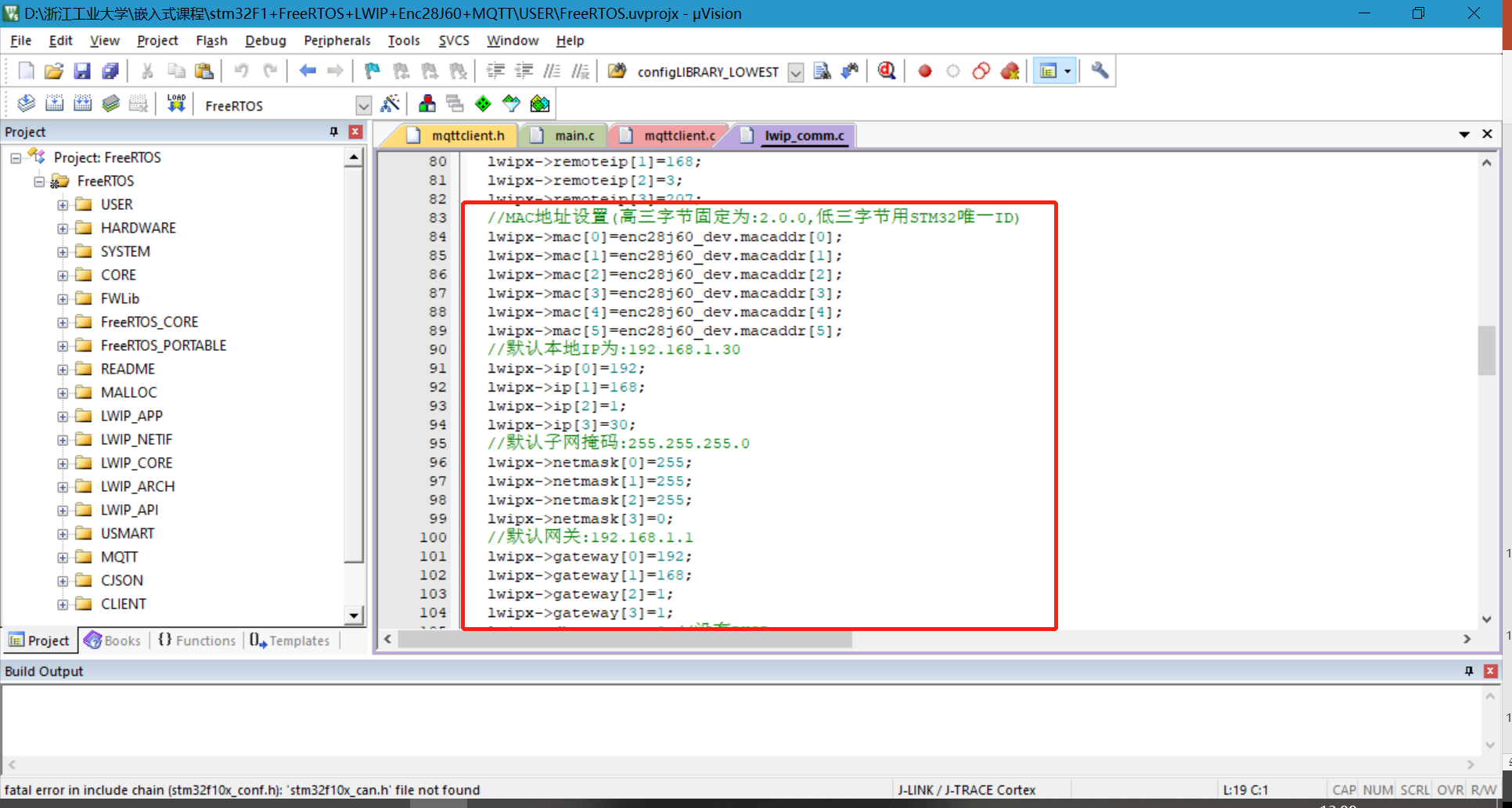
移植方法：正点原子在F1精英版的资料实验例程中有《LWIP带UCOS操作系统移植》和《FreeRTOS实验2-1 FreeRTOS移植实验》。参考这两个例程，搭建本次实验环境：LWIP带FreeRTOS操作系统。

移植后目录：



移植成功图片：



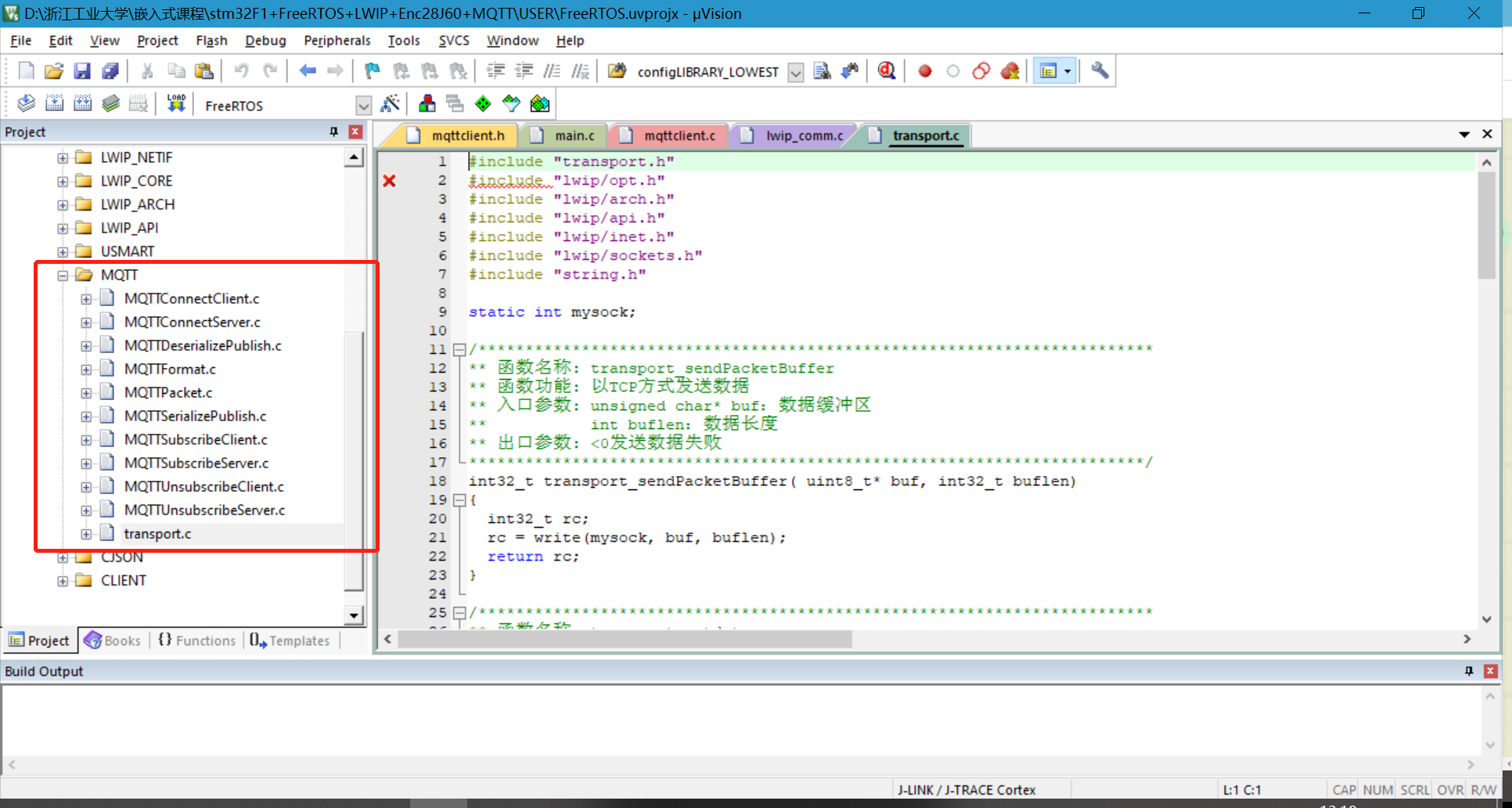


在没有接入路由器的情况下使用默认的静态IP。

## 2、MQTT移植

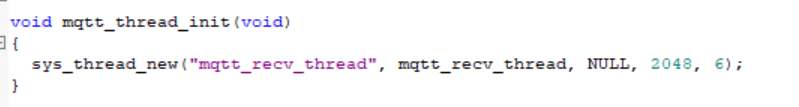
MQTT 是一个轻量的发布订阅模式消息传输协议，专门针对低带宽和不稳定网络环境的物联网应用设计。MQTT 基于发布/订阅范式，工作在 [TCP/IP协议族](https://baike.baidu.com/item/TCP%2FIP%E5%8D%8F%E8%AE%AE%E6%97%8F)上，MQTT 协议轻量、简单、开放并易于实现，这些特点使它适用范围非常广泛。

下载paho库，将MQTTPacket\src目录下的文件添加到工程，MQTTPacket\samples下的transport.c、transport.h添加到工程。



# 主要程序

1在main函数中新建一个任务，运行此函数进入mqtt\_recv\_thread进程。



2开始进行MQTT连接

void mqtt\_recv\_thread(void \*pvParameters)

{

uint32\_t curtick;

uint8\_t no\_mqtt\_msg\_exchange = 1;

uint8\_t buf[MSG\_MAX\_LEN];

int32\_t buflen = sizeof(buf);

int32\_t type;

fd\_set readfd;

struct timeval tv; //等待时间

tv.tv\_sec = 0;

tv.tv\_usec = 10;

MQTT\_START:

//开始连接

Client\_Connect();

//获取当前滴答，作为心跳包起始时间

curtick = xTaskGetTickCount();

while(1)

{

//表明无数据交换

no\_mqtt\_msg\_exchange = 1;

FD\_ZERO(&readfd);

FD\_SET(MQTT\_Socket,&readfd);

//等待可读事件

select(MQTT\_Socket+1,&readfd,NULL,NULL,&tv);

//判断MQTT服务器是否有数据

if(FD\_ISSET(MQTT\_Socket,&readfd) != 0)

{

//读取数据包--注意这里参数为0，不阻塞

type = ReadPacketTimeout(MQTT\_Socket,buf,buflen,0);

if(type != -1)

{

mqtt\_pktype\_ctl(type,buf,buflen);

//表明有数据交换

no\_mqtt\_msg\_exchange = 0;

//获取当前滴答，作为心跳包起始时间

curtick = xTaskGetTickCount();

}

}

//这里主要目的是定时向服务器发送PING保活命令

if((xTaskGetTickCount() - curtick) >(KEEPLIVE\_TIME/2\*1000))

{

curtick = xTaskGetTickCount();

//判断是否有数据交换

if(no\_mqtt\_msg\_exchange == 0)

{

//如果有数据交换，这次就不需要发送PING消息

continue;

}

if(MQTT\_PingReq(MQTT\_Socket) < 0)

{

//重连服务器

printf("发送保持活性ping失败....\r\n");

goto CLOSE;

}

//心跳成功

printf("发送保持活性ping作为心跳成功....\r\n");

//表明有数据交换

no\_mqtt\_msg\_exchange = 0;

}

}

CLOSE:

//关闭链接

transport\_close();

//重新链接服务器

goto MQTT\_START;

}

3处理收到的消息（调用Cjson库进行json解析）

void UserMsgCtl(MQTT\_USER\_MSG \*msg)

{

//这里处理数据只是打印，用户可以在这里添加自己的处理方式

printf("\*\*\*\*\*收到订阅的消息！\*\*\*\*\*\*\r\n");

//返回后处理消息

switch(msg->msgqos)

{

case 0:

printf("MQTT>>消息质量：QoS0\r\n");

break;

case 1:

printf("MQTT>>消息质量：QoS1\r\n");

break;

case 2:

printf("MQTT>>消息质量：QoS2\r\n");

break;

default:

printf("MQTT>>错误的消息质量\r\n");

break;

}

printf("MQTT>>消息主题：%s\r\n",msg->topic);

printf("MQTT>>消息内容：%s\r\n",msg->msg);

printf("MQTT>>消息长度：%d\r\n",msg->msglenth);

cJSON \* root = NULL;

cJSON \* item = NULL;

cJSON \* led0item = NULL;

root = cJSON\_Parse(msg->msg);

item = cJSON\_GetObjectItem(root, "params");

led0item = cJSON\_GetObjectItem(item, "LED0");

int led0=led0item->valueint;

cJSON\_Delete(root);

free(root);

LED0=led0; //打开或关闭led

msg->valid = 0;

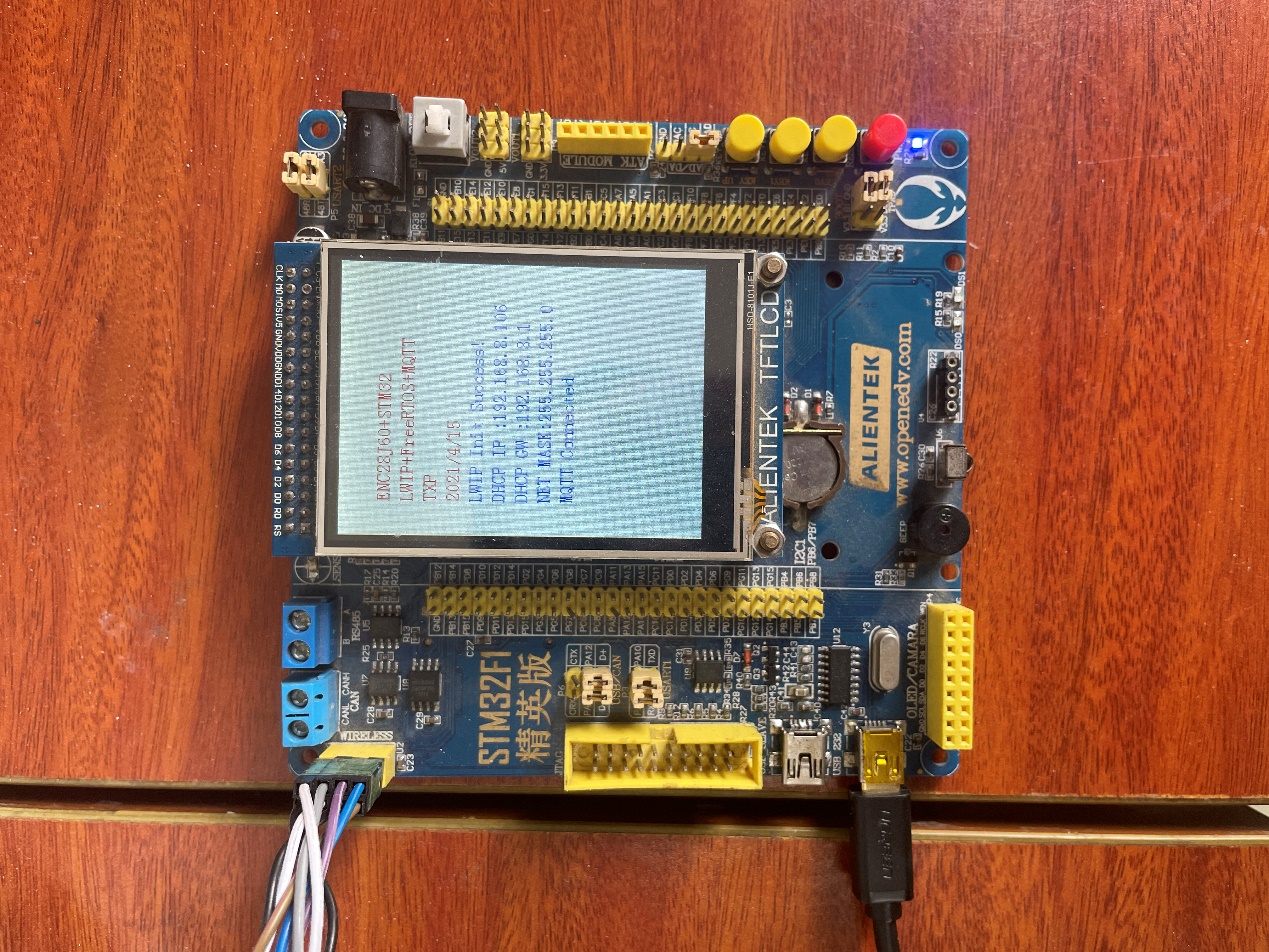
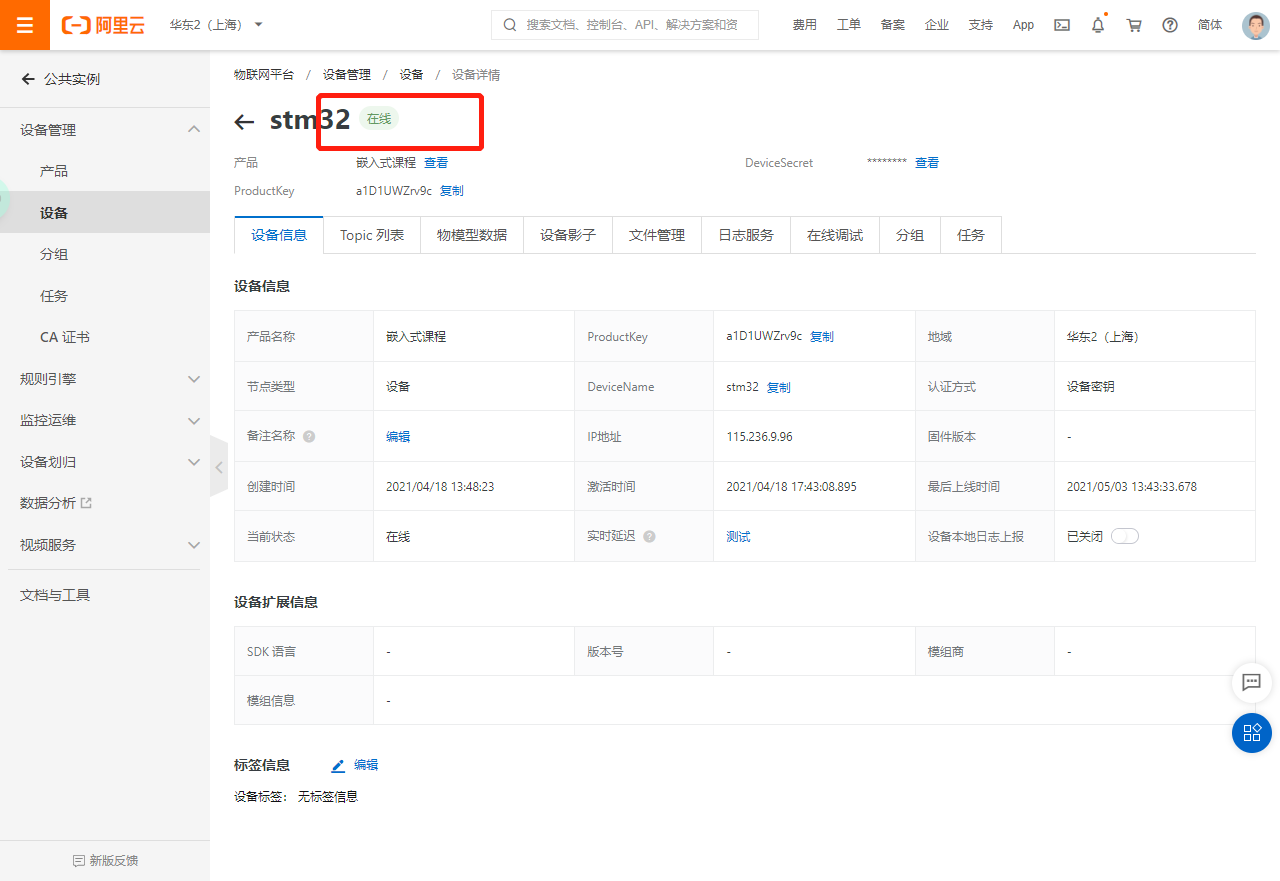
}

4在mqttclient.h中配置阿里云信息

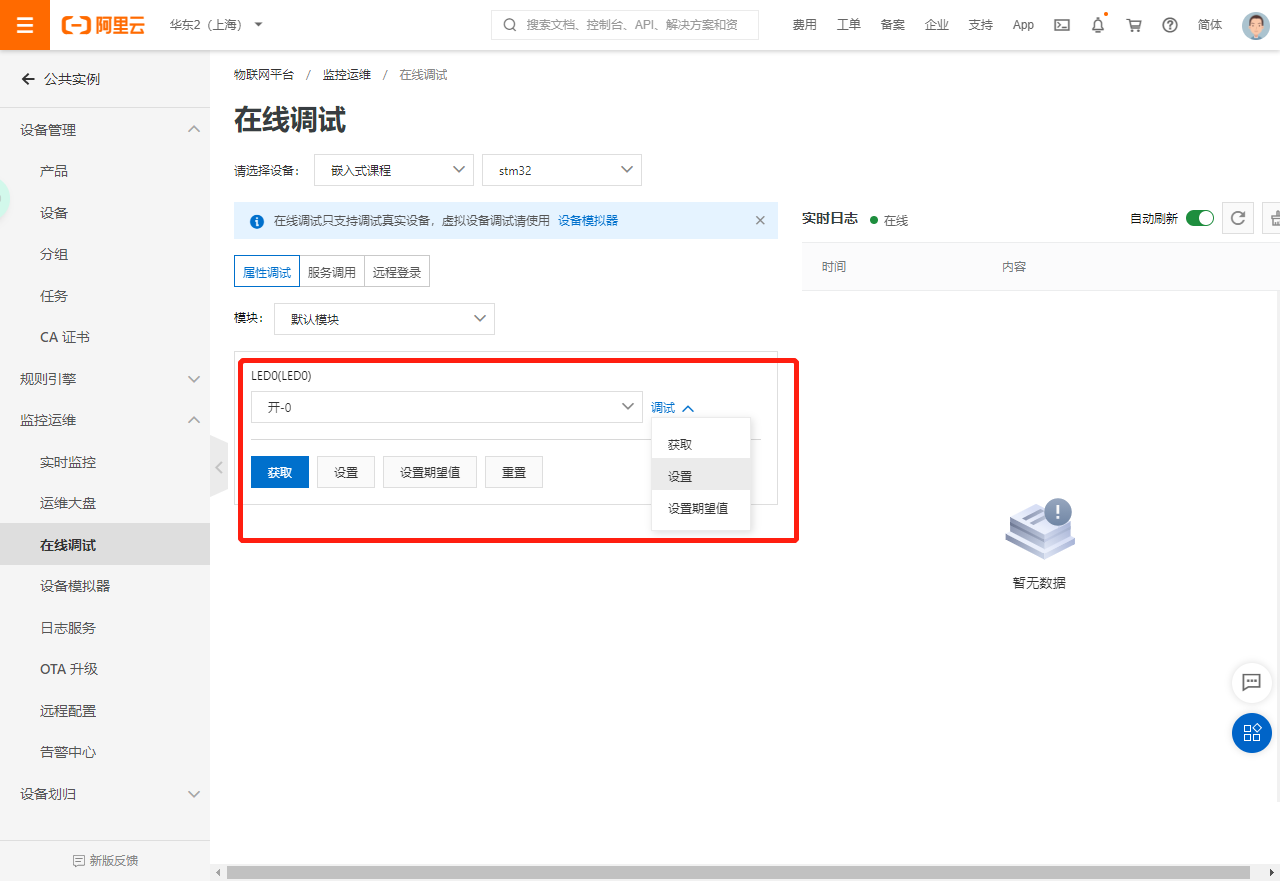


# 程序运行结果

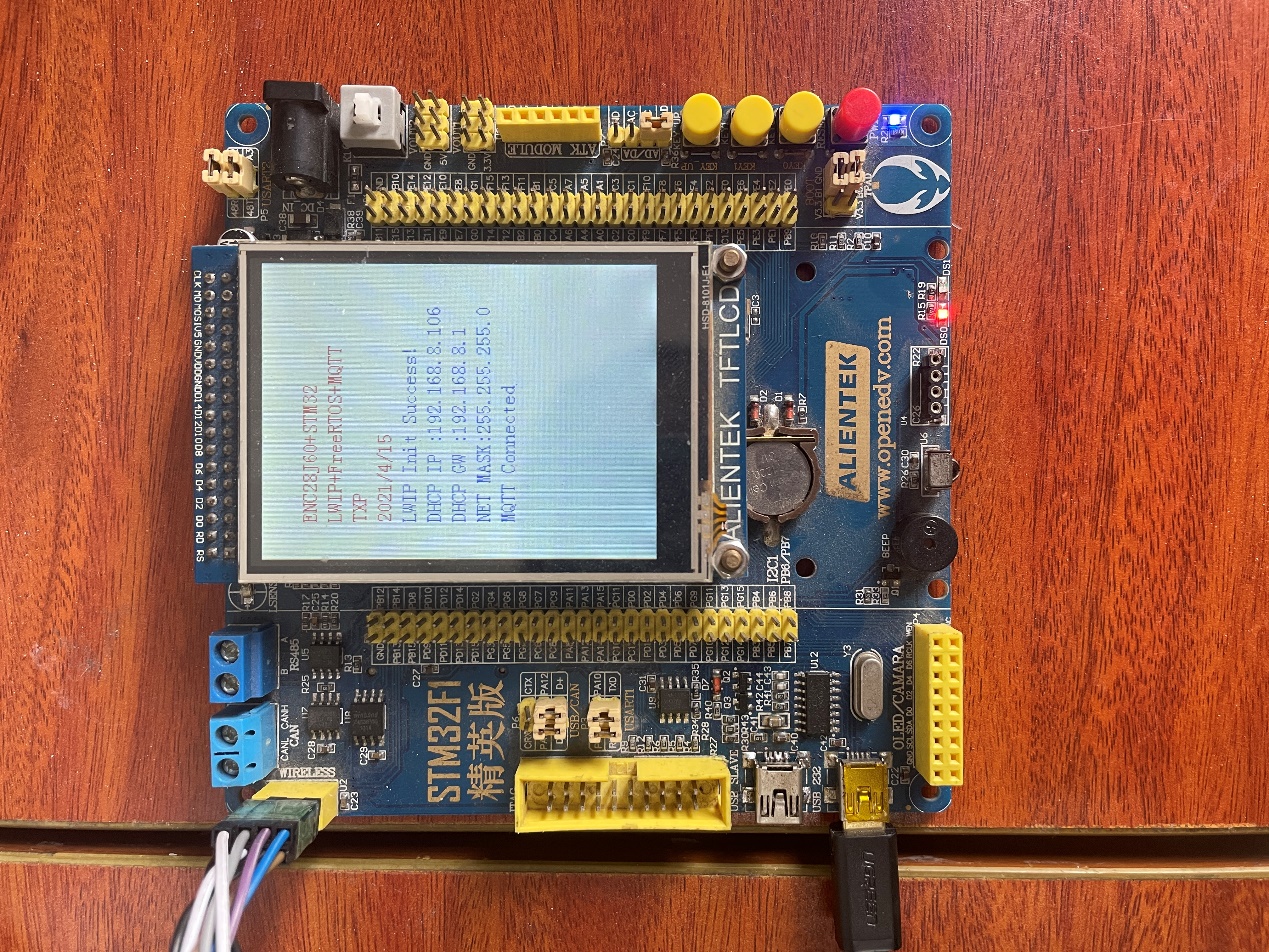
将开发板接入路由器

在阿里云物联网平台可以看到stm32开发板已经上线。接下来通过在线调试控制LED



设置LED开观察开发板



可以看到LED灯已经点亮。

# 实验总结

F4系列的板子可以通过cubemx快速配置FreeRTOS系统和LWIP协议，由于使用F103的开发板，在环境配置上花了不少时间。主要是参考正点原子的例程。在这个过程中对FreeRTOS和LWIP都有了更深入的了解。

在移植完成MQTT后，按照paho库里的例程修改，接入阿里云的过程，对MQTT协议也有了一定的理解。

总的来说，本次实验让自己收获了很多。