21级物联网通信技术复习

通信串口

参照上一节课

物联网是新一代信息技术领域的重要组成部分。其英文名称是“The Internet of things”，顾名思义，“物联网就是物物相连的互联网”。

这有两层意思：

第一，物联网的核心和基础仍然是互联网，是在互联网基础上的延伸和扩展的网络；第二，其用户端延伸和扩展到了任何物品与物品之间，进行信息交换和通信。

1. 全面感知：全面感知就是通过各种类型的传感器实时感知被测物理对象的状态。
2. 可靠传递：可靠传递就是通过各种网络与互联网的融合，将物体的信息实时准确地传递出去。
3. 智能计算：智能处理就是利用云计算、模糊识别等各种智能计算技术，对海量的数据和信息分析和处理，以实现对物体智能化控制。

通信

通信的目的

传递消息中所包含的信息

1. 消息 是物质或精神状态的一种反映，例如语音、文字、音乐、数据、图片或活动图片等

2. 信息 是消息的有效信息内容

通信系统的一般模型

信息源、发送设备、信道、接收设备、受信者、噪声源。

- 发送端：信息源、发送设备；
- 接收端：接收设备、受信源；

通信系统分类

根据按信道中传输的 **信号分类** ，通信系统分为 **模拟通信系统** 和 **数字通信系统** 。模拟通信系统传送的是模拟信号，这代表消息的信号参量取值是连续的。

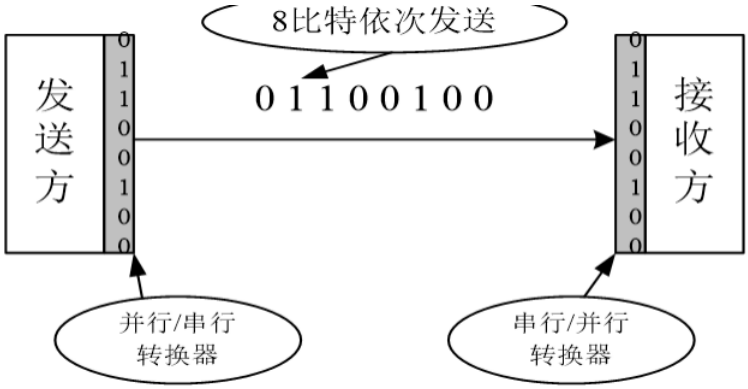
调制是指把信号转换成适合在信道中传输的形式的一种方产。广义调制分为 **基带调制** 和 **频带（载波）调制**

点对点之间的通信，按消息传送的方向，通信方式可分为 **单工通信** 、**半双工通信及全双工通信** 三种

串行通信

将数字元码序列以串行方式一个元码接着一个元码的在一条信道上传输。

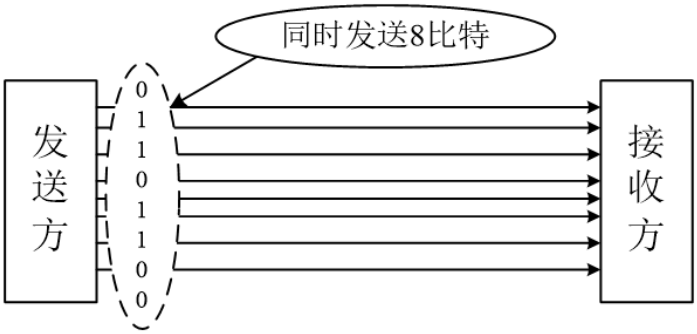
优点：只需一条通信信道，节省线路铺设费用； 缺点：速度慢，需要外加码组或字符同步措施。



随着技术的发展，以USB为代表的串行传输速度已经超过并行传输。

并行通信

将代表信息的数字信号元码序列以组的方式在两条或者两条以上在并行信道上同时传输。



优点：节省传输时间，速度快，不需要字符同步措施； 缺点：需要n条数据线，成本高。

标准，ASCI码使用7bit数据来表示所有的大写和小写字母、数字0到9、标点符号,以及在美式英语中使用的特殊控制字符。

串行通信是指使用一条数据线,将数据一位一位地依次传输,每一位数据占据一个固定的时间长度

在进行串口通信时,两个串口设备间需要**发送端(TX)**与**接收端(RX)**交叉相连,并共用电源地(GND)。

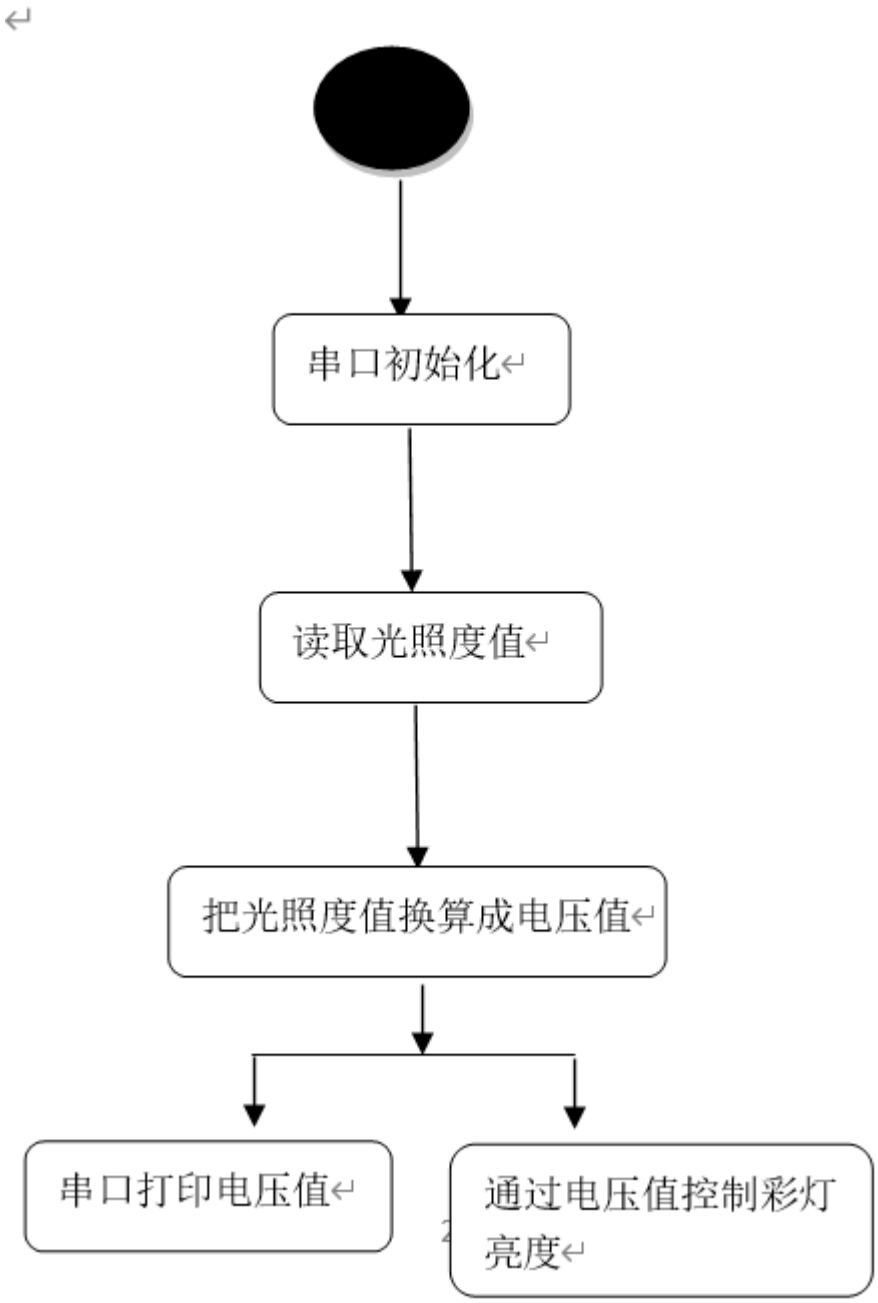
1. **串行通信的约定** 串行通信中的这种约定包含两个方面，
    - 一方面是通信的速率要保持一致，
    - 另一方面是字符的编码要一致。
2. **波特率** 通信速率是指单位时间内传输的信息量,可用比特率和波特率来表示。 **比特率** 是指每秒传输的二进制位数,用bps (bits)表示。 **波特率** 是指每秒传输的符号数,若每个符号所含的信息量为1比特,则波特率等于比特率。
3. 串口工作原理

在Arduino与其它器件通讯的过程中,数据传输实际上都是以数字信号(即电平高低变化)的形式进行的,串口通信也是如此.当使用Serial.print()函数输出数据时,Arduino的发送端会输出一连串的数字信号,称这些数字信号为数据帧. 例如,当时用Serial.print('A')语句发送数据时,实际发送的数据帧格式如下图所示:

Serial.begin()是Arduino开发平台串口初始化函数。 物联网设备跟云平台的通讯方法分为同步传输和异步传输

以光照度传感器为例，讲解串口通信模式。

```
const int sensorPin = 2;
const int ledPin = 13;
int lineFound = -1;
void setup() {
Serial.begin(115200);

pinMode(sensorPin, INPUT);
pinMode(ledPin, OUTPUT);
}
void loop() {
lineFound = digitalRead(sensorPin);
if(lineFound == HIGH) {
Serial.println("    ");
digitalWrite(ledPin, HIGH);
} else {
Serial.println("    ");
digitalWrite(ledPin, LOW);
}
delay(10);
}
```

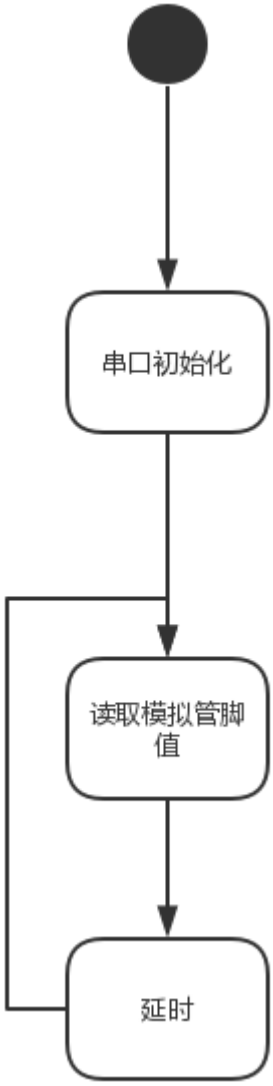


流程如下：

串口例程：

```
const int analogInPin = A0;
int  sensorValue = 0;
void setup(){
     Serial.begin(115200,8N1);
 }
void loop(){
     sensorValue = analogRead(analogInPin);
delay(250);
}
```

（1）串口通信模式有：1）波特率为115200 （2）校验方式为无校验 (3)数据位为8位、停止位为1位

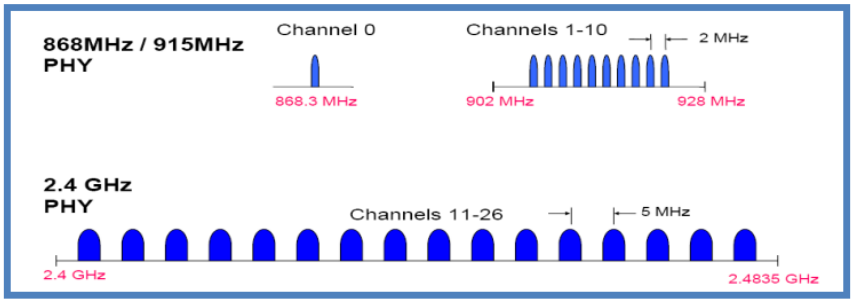


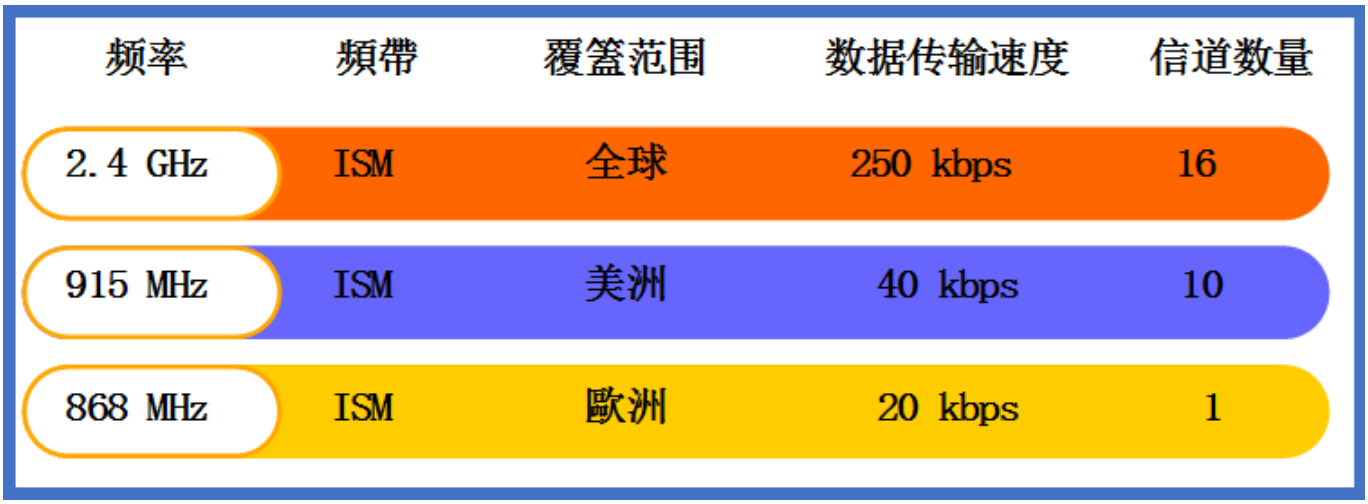
（2）串口通信流程图

ZigBee与IEEE802.15.4

ZigBee网络中的每一个节点都有一个16bit网络地址和一个64bit IEEE扩展地址。

IEEE802.15.4定义了2450MHz、868MHz、915MHz三个频段。对于不同的国家和地区，为其提供的工作频率范围是不同的，其中2450MHz （2.4GHz）频段是全球统一无需申请的的ISM频段，提供 **250kbps** 的通信速率，并提供16个信道;868MHz频段是欧洲的ISM频段，提供1个速率为20kbps的信道，915MH频段是美国的ISM频段，提供10个速率为40kbps的信道。





频率	频带	覆盖范围	数据传输速度	信道数量
2.4 GHz	ISM	全球	250 kbps	16
915 MHz	ISM	美洲	40 kbps	10
868 MHz	ISM	欧洲	20 kbps	1

在ZigBee网络中存在三种节点类型：

- Coordinator(协调器)，
- Router(路由器)
- End-Device(终端设备)。

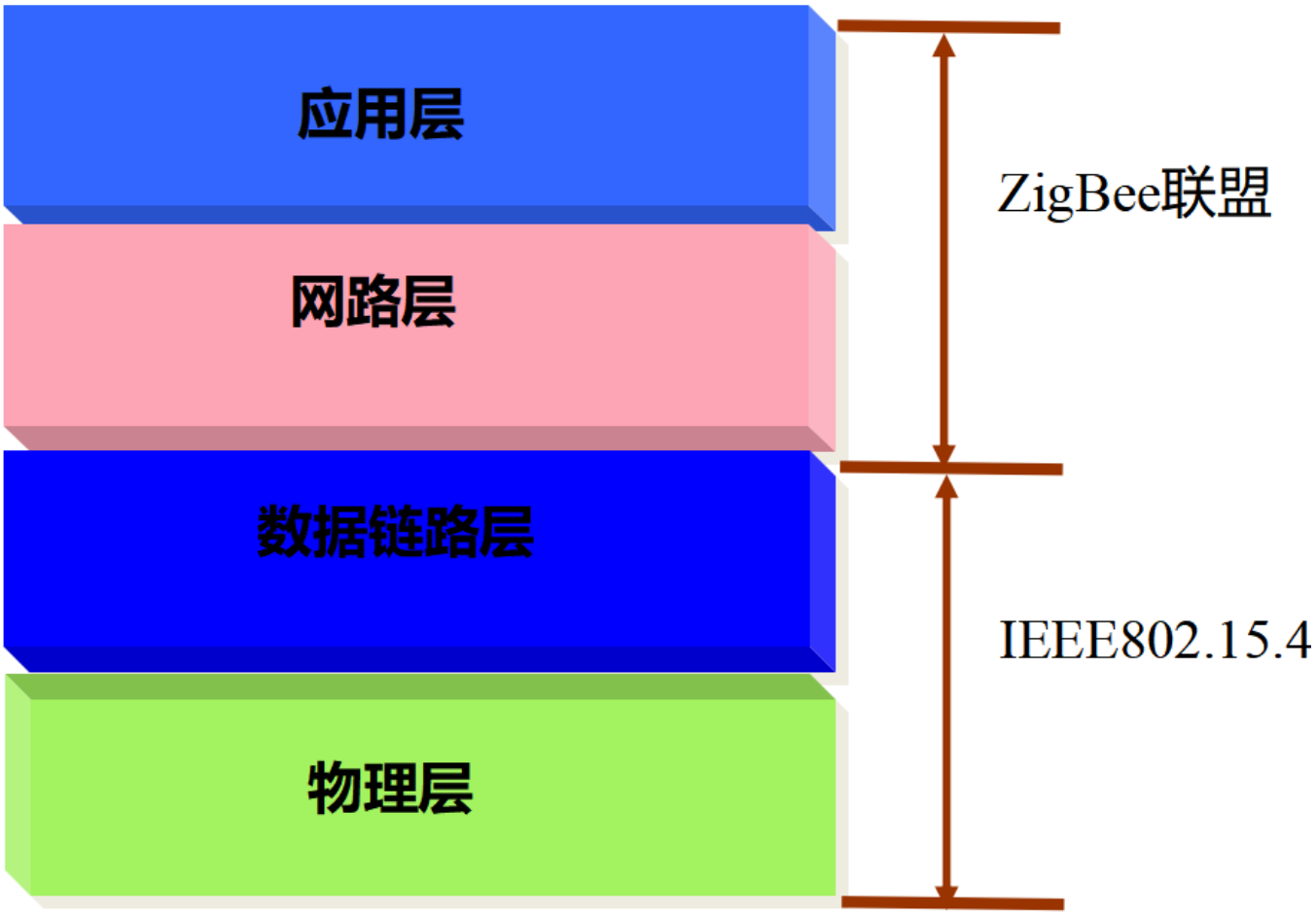
ZigBee网络由一个Coordinator以及多个Router和多个End_Device组成。

ZigBee可采用 **星状** 、**片状** 和 **网状** 网络结构，由一个主节点管理若干子节点，最多一个主节点可管理254个子节点;同时主节点还可由上一层网络节点管理，最多可组成**65000**个节点的大网。

三种设备类型

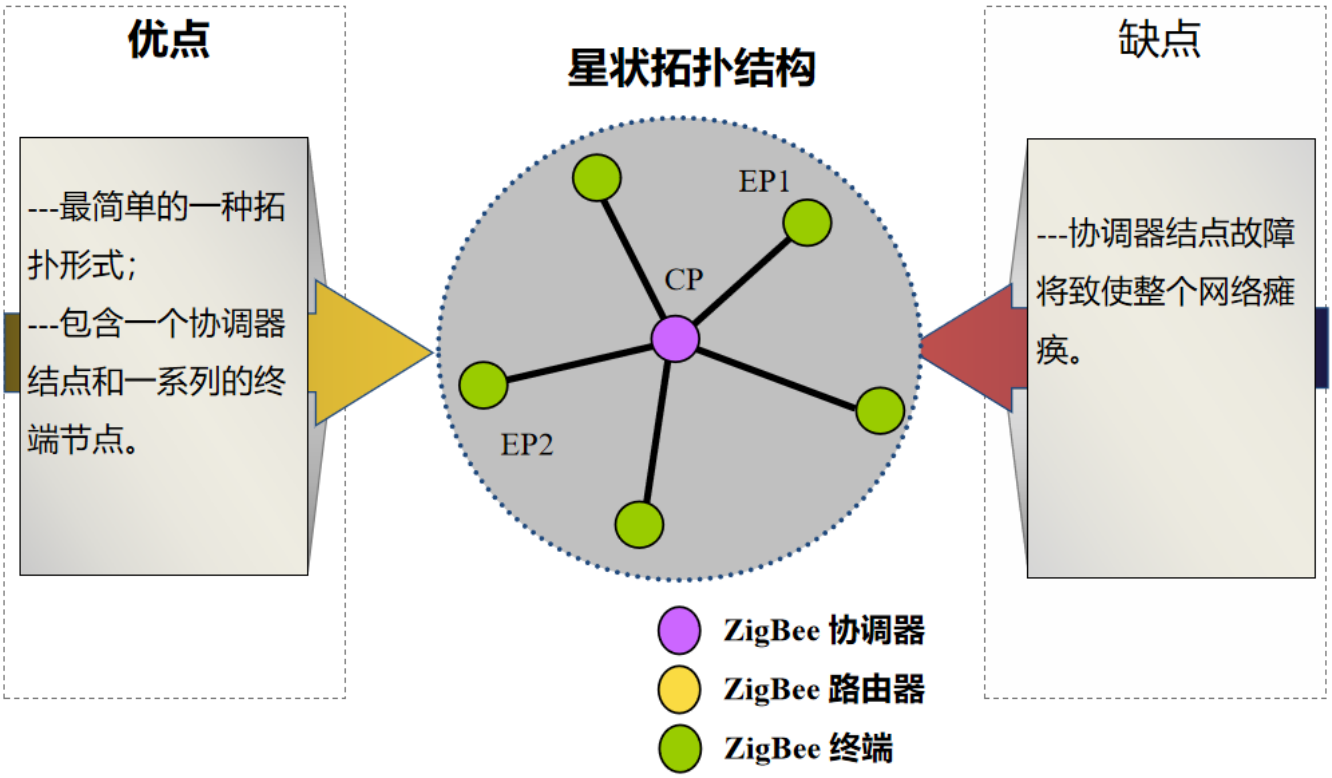
三种节点类型及其案例

三种网络拓扑结构类型



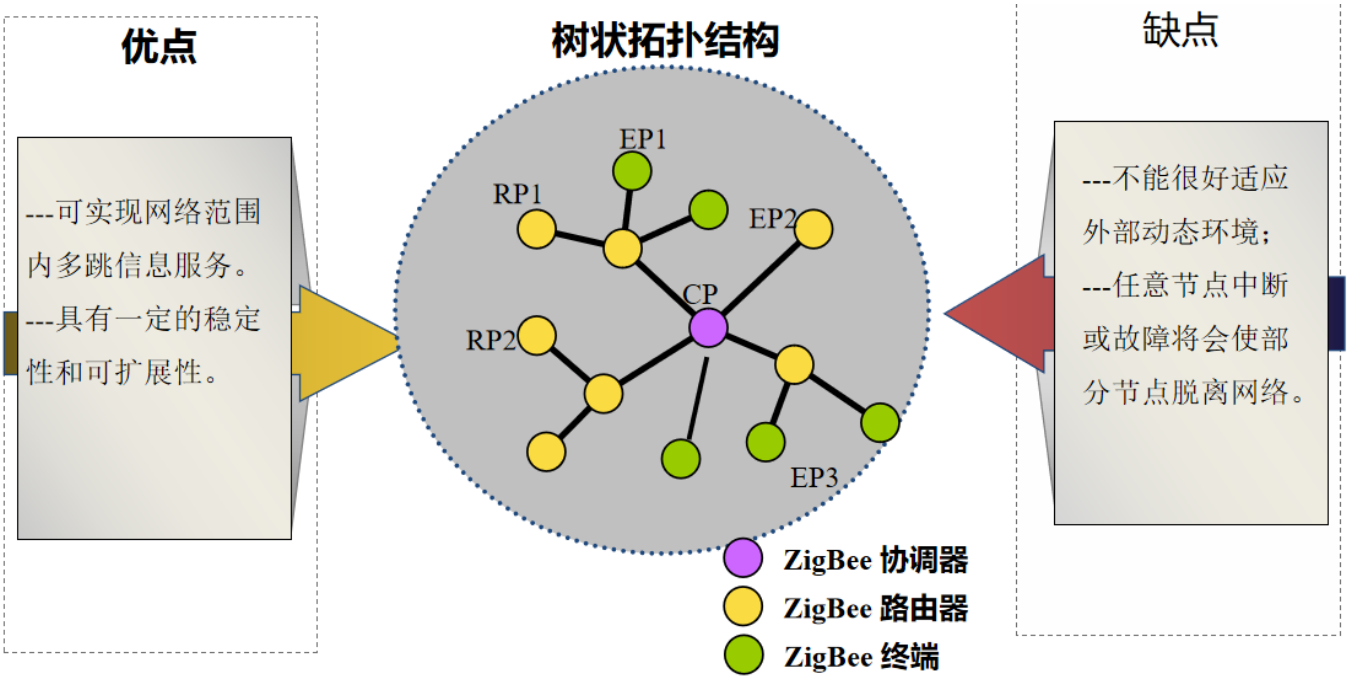
星形拓扑

包含一个Co-ordinator（协调者） 节点和一系列的 End Device（终端）节点，每一个End Device 节点只能和 Co-ordinator 节点进行通讯。如果需要在两个 End Device 节点之间进行通讯必须通过Co-ordinator 节点进行信息的转发。



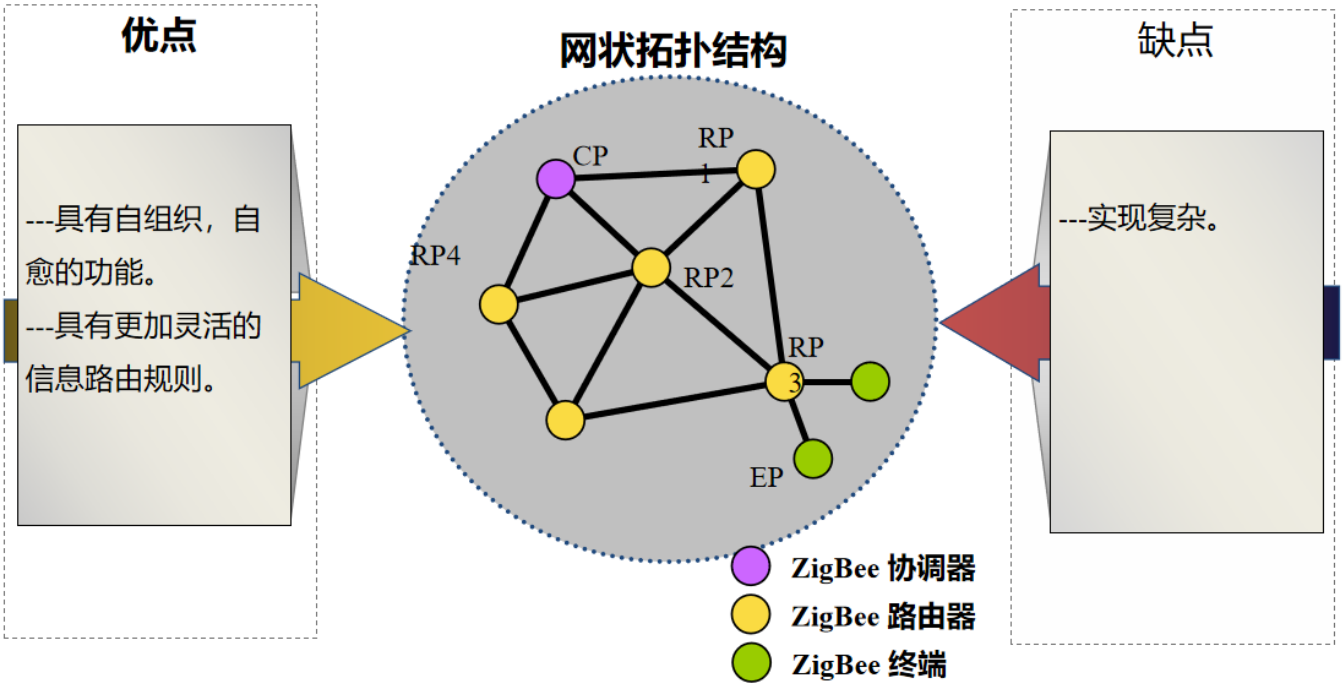
树形网络

树形拓扑包括一个Co-ordinator（协调者）以及一系列的 Router（路由器） 和 End Device（终端）节点。Co-ordinator 连接一系列的 Router 和 End Device， 他的子节点的 Router也可以连接一系列的 Router 和End Device. 这样可以重复多个层级。 Co-ordinator 和 Router 节点可以包含自己的子节点。End Device 不能有自己的子节点。有同一个父节点的节点之间称为兄弟节点。



网型网络（Mesh拓扑）

网形网络（Mesh拓扑） 包含一个Co-ordinator和一系列的Router 和End Device。这种网络拓扑形式和树形拓扑相同；请参考上面所提到的树形网络拓扑。但是，网状网络拓扑具有**更加灵活的信息路由规则**，在可能的情况下，路由节点之间可以直接的通讯。这种路由机制使得信息的通讯变得更有效率，而且意味这一旦一个路由路径出现了问题，信息可以自动的沿着其他的路由路径进行传输。



RFID射频识别技术

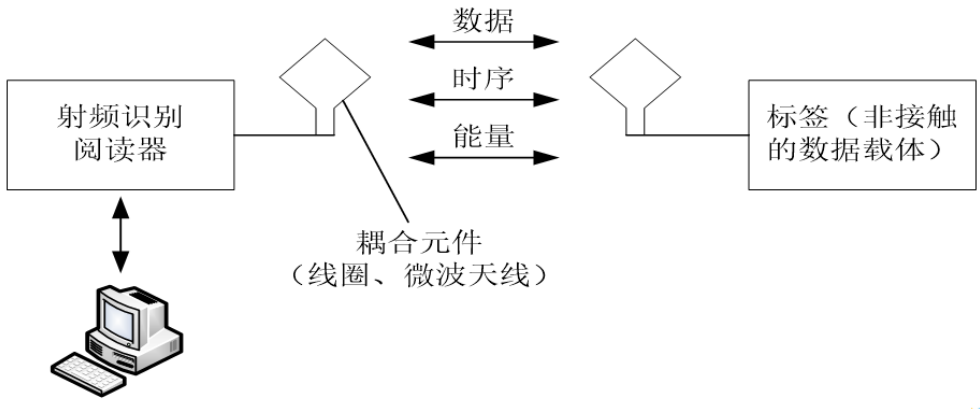
RFID的定义：射频识别（Radio Frequency Identification，RFID）是一种 **非接触式** 的自动识别技术，它通过射频信号自动识别特定目标对象并读写相关数据，而无须识别系统与特定目标之间建立机械或光学接触。

1. RFID系统通常由 **电子标签** 、**读写器** 和 **计算机网路** 三部分组成。
2. RFID系统中的读写器的 **工作频率** 决定了整个RFID系统的工作频率，功率大小决定了整个RFID系统的工作距离。
3. 在RFID系统中常用的频段有4种：低频LF(125kHz)、高频HF(13.56MHz)、超高频UHF(850～910MHz)及微波(2.45GHz)
4. 电子标签分为 **无源标签(passive)**、 **半无源标签(semi-passive)** 和 **有源标签(active)三种类型**。
5. RFID最基本的硬件体系结构由RFID电子标签、RFID读写器和计算机。

RFID系统基本硬件组成

RFID系统的工作原理

利用射频信号的空间耦合（电磁感应或电磁传播）传输特性，实现对静止的、移动的待识别物品的自动识别，如图所示：

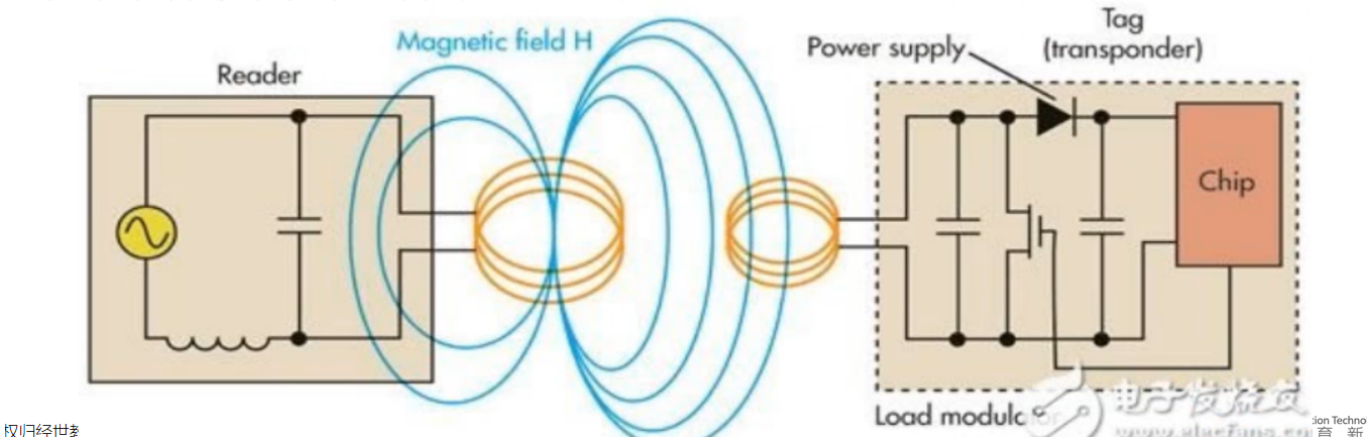


1. RFID网络框架结构

- 通信串口组网、
- 读写器与计算机直接连接、
- 读写器通过中间件连入网络

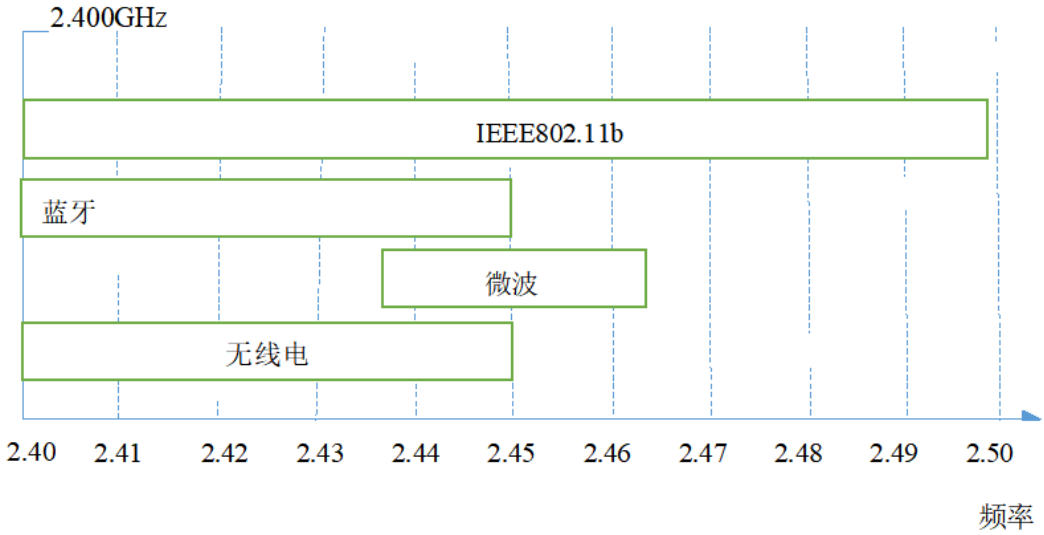
RFID工作过程

读写器向M1卡发一组固定频率的电磁波，卡片内有一个LC串联谐振电路，其频率与读写器发射的频率相同，在电磁波的激励下，LC谐振电路产生共振，从而使电容内有了电荷，在这个电容的另一端，接有一个单向导通的电子泵，将电容内的电荷送到另一个电容内储存，当所积累的电荷达到2v时，此电容可做为电源为其它电路提供工作电压，将卡内数据发射出去或接取读写器的数据。



蓝牙技术

1. 蓝牙技术工作在2.4GHz频段，2.4GHz的ISM波段是一种短距离无线传输技术，供开源使用。

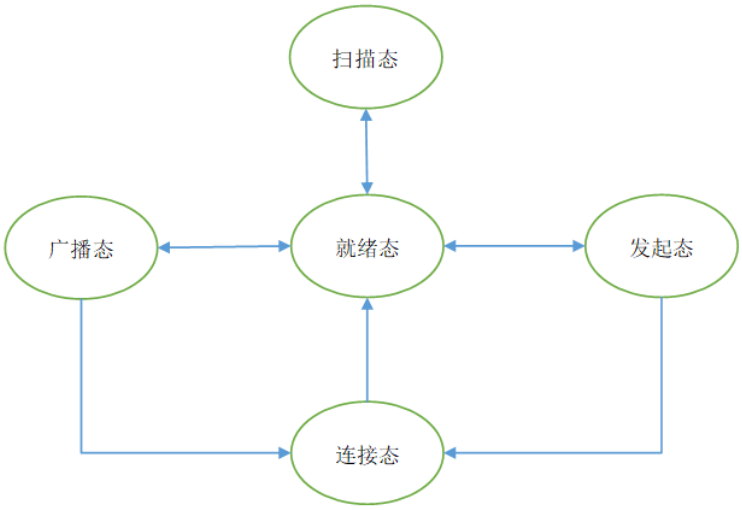


2. 蓝牙使用**跳频技术**，将传输的数据分割成数据包，通过79个指定的蓝牙频道分别传输数据包。每个频道的频宽为1MHz，蓝牙4.0使用2MHz间距，可容纳40个频道。 3. 蓝牙采用**高速跳频(Frequency Hopping)和时分多址(Time Division Multiple Access，TDMA)等先进技术，支持点对点**及**点对多点通信**。其传输频段为全球公共通用的2.4 GHz频段，能提供1 Mb/s的传输速率和10 m的传输距离，并采用**时分双工传输**方案实现全双工传输。 4. 在一个微微网中，所有设备的级别是相同的，具有相同的权限。主设备单元负责提供**时钟同步信号**和**跳频序列**，从设备单元一般是**受控同步**的设备单元。

链路管理协议层利用状态机定义了设备的五中状态

- 就绪
- 扫描
- 广播

- 发起
- 连接



WiFi技术

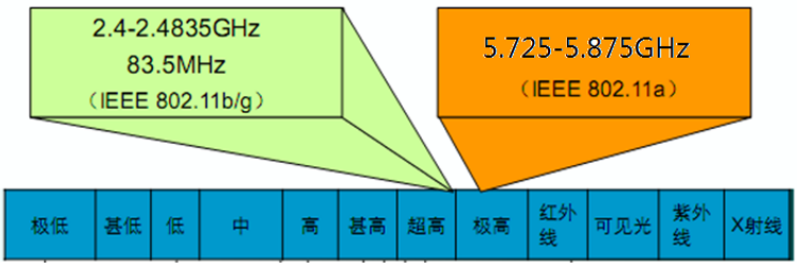
1. WiFi技术的基本概念 **定义：** 一种可以将个人计算机、手机设备等终端以无线方式互相连接的技术。
2. WiFi（wireless Fidelity），在无线局域网领域称为"无线相容性认证"，是一种无线网络通信的品牌，有Wi-Fi联盟所持有，目的是为了改善基于IEEE802.11b标准的无线网络产品之间的互通性。

- WiFi既是一种商业认证，也是一种无线联网技术；
- WiFi技术的优点：可移动、实现简单、可靠性强、环保指数高；

无线局域网（Wireless Local Area Network，WLAN）

无线局域网（WLAN）工作频段

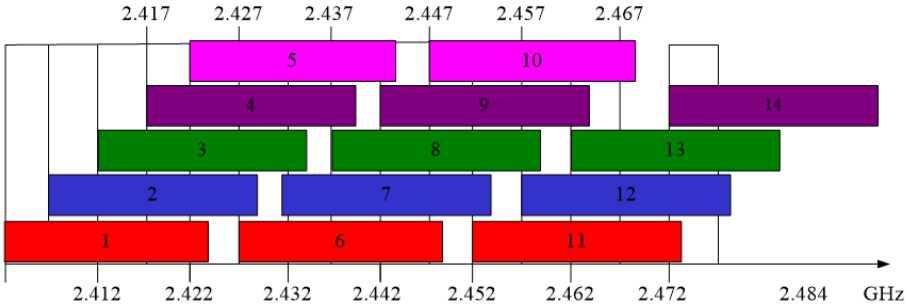
- 按照国家的无线电管理部门规定，WLAN使用2.4GHz和5.8GHz两个频段。这两个频段属于特高频和微博范围，属于微波波段（波长小于一米）



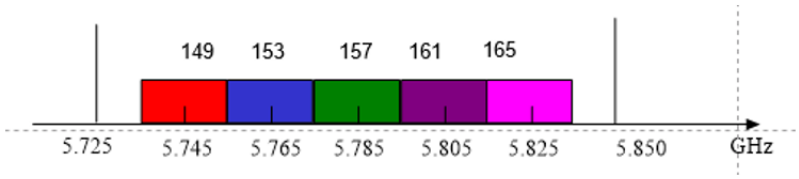
频谱示意图

- **2.4GHz频段信道划分：**

  可用的宽带为83.5MHz，划分13个信道，每个信道宽为22MHz，信道之间有间隔，间隔5MHz。2.4GHz频段信道划分如下图所示：

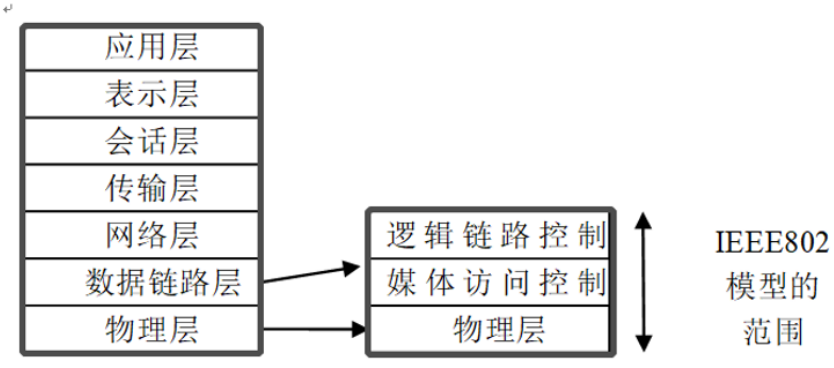


- **5.8GHz频段信道划分：** 在中国大陆，IEEE802.11a使用的是5.98GHz频段，工作频率范围为5725-5875MHz。可用信道为5个，分别为149，153，,157，,161，165，每两个信道之间相差4，信道之间没有间隔。



1. 无线局域网（WLAN）的架构

- 无线局域网参考模型与OSI七层参考模型如下图：
- IEEE802标准遵循ISO/OSI 参考模型的原来规则确定最低两层——物理层与数据链路层的功能，以及与网络层的接口服务、网络互联等有关的高层功能。



2. 无线局域网（WLAN）的重要协议

- IEEE最初制定的无线局域网标准主要用于解决 **办公室局域网** 和 **校园网** 中用户以及 **终端的无线接入** 问题，其业务主要限于数据存取，其速率最高只能达到2Mpbs。
- 由于它在速率和传输距离上都不能满足人们的需要，IEEE小组又相继推出了802.11b、802.11a、802.11g、802.11n、802.11ac等一系列标准。

(1) WLAN技术具有以下优势

- 灵活性强，终端设备可以在WLAN网络的覆盖范围内任意放置和低速移动。
- 部署成本较低，省去了终端设备综合布线的费用。
- 扩展能力好，增加AP就可以增大容量和覆盖范围。
- 技术成熟，在国外已经得到广泛的应用。
- 采用了扩频技术和加密机制，具有较强的抗干扰性和网络保密性，安装简单。

(2）帧的种类和用途如下:

- 数据帧:用来传送数据，不含任何数据的空帧也属于数据帧。
- 控制帧:控制媒体存储的各种帧。
- 管理帧:与无线网络运作有关的帧，例如无线设备的连接，身份认证等等。

ESP8266单片机

- ESP8266模块的开发语言C语言
- ESP8266模块有1个模拟输入管脚

常用函数

1. WiFi.begin(ssid,pass)：
    - ssid: 热点的名字
    - pass：热点的密码
2. WiFi.mode()：设定wifi的操作模式，其参数值为: WIFI_AP,WIFI_STA,WIFI_AP_STA或者WIFI_OFF
    - WIFI_OFF:关闭WiFi
    - WIFI_STA:设置成Wi-Fi终端
    - WIFI_AP:设置成Wi-Fi网络接入点(基站，热点)
    - WIFI_AP_STA:-设置成Wi-Fi网络接入点以版终端
3. WIFI.getMode():返回当前的WIFI模式，上述四种模式中的一种。
4. WIFI.softAP(ssid):创建一个无线局域网热点。这个热点允许其他设备（如手机、电脑等）通过Wi-Fi连接到ESP8266。参数 ssid 是要创建的Wi-Fi热点的名称，是一个字符串类型变量,最多31个字符。
5. WiFi.softAP(ssid,password,channel,hidden,max_connetion)：设置受密码保护的网络或配置其他网络参数，此功能的第一个参数是必须的，其余四个是可选的；

    - **password** 带有密码的可选字符串。对于WPA2-PSK网络，其长度至少应为8个字符。如果未指定，则接入点将打开，任何人都可以连接（最多63个字符）

    - **channel（可选参数）** ：指定SoftAP要使用的Wi-Fi信道。不同的国家和地区有不同的可用信道范围，我国的信道范围规定,选择一个相对空闲的信道有助于提高无线通信质量。

- **hidden（可选参数）** ：如果设为true，表示创建一个**隐藏网络**，即不广播其SSID，这样其他设备不能通过常规方式自动扫描到该热点。用户需要手动输入正确的SSID才能连接。
- **max_connection（可选参数）** ：定义允许同时连接到SoftAP的**最大客户端数量**。超出这个数目的设备将无法再加入此Wi-Fi网络。

6. **softAPgetStationNum**()：获取连接的soft-AP接口的站点数WiFi. softAPdisconnect(wifioff)，功能会将当前配置的SSID和soft_AP的密码设置为空值，该参数wifioff是可选的，如果设置为true，将关闭soft-AP模式
7. **WiFi.softAPIP()**: 返回访问点的网络接口的IP地址，返回值是IPAddress类型。
8. **WiFi.softAPmacAddress(mac):** 函数接受一个参数mac，该参数是指向内存位置的指针（一个uint8_t大小为6个元素的数组），以保存mac地址。函数本身返回相同的指针值。

Station模式

- 首先**AP**发出信标(beacon)帧，意思就是我在这里，谁来连接我啊;
- **移动工作站** 也会发出探(probe)帧，意思是有谁我可以连接啊。这个是工作站主动发出来的，每隔一定时间发出一次。

所以，根据这点，可以产生很多有价值的应用。比如你的带WiFi功能的手机，即使不连接wifi的情况下，只要打开WiFi功能，就可以被路由器截获这帧信息，路由器收集之后，你的信息就会被一个审计的东西发到服务器上，你手机号xxx上线时间xxx下线时间xxx都浏览了那些网页，WiFi建立连接过程都一目了然，你的位置也全都暴露了，这就叫WiFi探针。

具体流程如下所示:

1. 探测请求过程： STA：设备 ---------->Probe Request(探测请求)---> AP：热点
2. 返回应答帧： STA<---------- Probe Response <---------- AP，这个是由wifi（热点）返回的应答帧;

   接下来是身份验证过程，可以使用诸如 WEP、WEP2、WPA 等加密方式应用到认证请求上;
3. 身份验证过程： STA ----------> Authentication Request -------->AP 认证请求中包含认证 Auth类型，Open System , Shared Key等信息，路由器返回认证结果;
4. 连接请求 STA<---------- Authentication Response<--------- AP连接请求; STA----------> Association Request ---------->AP请求与AP建立关联，从而可以进行数据交互;认证过连接请求OK返回。
5. 数据传输 STA<---------- Association Response<---------- AP

```
#include <ESP8266WiFi. h>
#define ssid"ESP8266"
#define pssd"123456789”
const char* stassid= "doit";
const char* stapssd = "doit3305";
void setup(){
// put your setup eode here, to run once :
serial.begin(115200);//波特率设置为115200
wiPi.mode(WIFI_AP_STA);//设置WIFI操作模式为WIFI_AP_STA,即接入点以及终端
wiPi.disconnect();//
wiri.begin(stassid, stagssd);
```

```
while(WiPi.status() !=L_COHIEECTED){
    delay=(500) ;
    Serial.print(""."");
}

serial .println("TIFI connect sucess") ;
serial. println(WiPi.localIP());
WiPi.softAP(ssid,pssd);//连接WIFI
IPAddress ip=wiPi.softAPIP();//softAPIP()返回网络接口IP地址，返回值为IPAddress
serial . println(ip):
Serial . println(wiPi.softAPmacAddress().c_str());
}
void loop(){
//put your main code here, to run repeatedly :
Serial.printf("Station conneted to soft-AP = %d\n"
,WiPi.softAPgetStationJlun());//softAPgetStationJlun获取soft-AP接口的站点数
delay=(5000);
}
```

TCP/IP协议

TCP/IP不是指特定的某一个协议，而是一个协议族，包含众多的协议。

互联网通信的本质是数字通信，任何数字通信都离不开通信协议的制定，通信设备只有按照约定的、统一的方式去封装和解析信息，才能实现通信。互联网通信所要遵守的众多协议，被统称为**TCP/IP** .

Serverd的应用

与之相关的函数：

- Server ：服务器是所有基于WIFI Server的基类，
- Server()函数：创建一个服务器，以侦听指定端口上的连续；

```
Server(port)
参数port:侦听的端口（int类型）；
返回值：无
```

如下为一个Server应用的代码，代码实现了用开发板连接手机开的热点，然后在手机上输入串口监视器打印的IP，然后可以看见显示 "analog input:2" 的网页。

```
#include<ESP8266WiFi.h>
cconst char* ssid = "LW";
const char* pssd = "ping123456";
WiFiServer server(80);//侦听端口，无返回值
void setup() {
// put your setup code here, to run once:
    Serial.begin(115200);
    Serial.println();
```

```
        Serial.printf("connetctint tu %s",ssid);
        WiFi.begin(ssid,pssd);  //连接wifi
        while(WiFi.status()!=WL_CONNECTED)   //等待连接wifi
        {
            delay(500);
            Serial.print(".");
        }
        Serial.println("connected");
        server.begin();      //侦听服务器
        Serial.printf("web server started,open %s in a web
browser\n",WiFi.localIP().toString().c_str());
    }

    String prepareHtmlPage()         //返回网页浏览器
    {
        String htmlPage=
        String("HTTP/1.1 200 OK\r\n")+
        "Contect-Type:text/html\r\n"+
        "Connerction:close\r\n"+  //连接关闭请求
        "Refersh:5\r\n"+      //5秒刷新一次网页
        "\r\n"+
        "<!DOCTYPE HTML>" +
        "<html>"+
        "analog input:" + String(analogRead(A0))+
        "</html>"+
        "\r\n";
        return htmlPage;
    }
    void loop() {
    // put your main code here, to run repeatedly:
        WiFiClient client = server.available();     //检测是否有客户端存在
        if(client)
        {
            Serial.println("\n[Client connected]");
            while(client.connected())       //判断客户端是否连接成功
            {
                String line = client.readStringUntil('\r');    //每行读取到r结束
                //等待服务器的应答
                if(line.length()==1&&line[0]=='\n')
                {
                    client.println(prepareHtmlPage());
                    break;
                }
                }
            }
            delay(1);
    }
```

TCPClient函数

1. 相关函数介绍

connect()描述:
连接到构造函数中指定的IP地址和端口。返回值指示成功或失败。使用域名(ex:google.com) 时，connect()还支持DNS查找。
句法:
- client.connect(ip,port)
- client.connect(URL,port)参数
ip：客户端将连接的ip地址（4个字节的数组）
URL：客户端将连接到的域名（字符串，例如："arduino.cc"）
Port：客户端将连接到的端口（ int）
返回值：如果连接成功，则返回 true;
否则，返回false

---

print()描述:
将数据打印到客户端连接到服务器。将数字打印为数字序列，每个数字为一个ASCI字符句法:
client.print(data)
client.print(data,BASE)参数
data:要打印的数据（char,byte,int,long或string）
BASE（可选）:打印数字的基数:DEC代表十进制（基数10），
OCT代表八进制（基数8），
HEX代表十六进制（基数16>

---

2. **客户端访问服务器流程**

- 连接到Wi-Fi
- 选择服务器
- 实例化客户端WiFiClient client;
- 把客户端连接到服务器
- 请求数据
- 读取来自服务器的回复 实现代码如下：

```
#include <ESP8266WiFi.h>
const char* ssid = "LW";
const char* pssd = "ping123456";
const char* host = "www.baidu.com";
void setup() {
// put your setup code here, to run once:
    Serial.begin(115200);
    Serial.println();
    Serial.printf("connecting to %s ",ssid);
    WiFi.begin(ssid,pssd);
    while(WiFi.status()!=WL_CONNECTED)       //等待WIFI连接成功
    {
        delay(500);
        Serial.print(".");
```

```
        }
        Serial.println("connected");
    }

    void loop() {
    // put your main code here, to run repeatedly:
    /*  String host="";
        while(Serial.available()>0)
        {
            host += (char)Serial.read();
            delay(2)
        }
        */
        //char host[30];
    //     for(int i=0;i<30;i++)
    //     {
    //          host[i]='\0';
    //      }
    //      while(Serial.available()>0)
    //    {
    //        host += (char)Serial.read();
    //        delay(2)
    //      }
    //
        WiFiClient client;
        Serial.printf("\n[connecting to %s ...",host);
        if(client.connect(host,80))      //连接服务器成功
        {
            Serial.println("connected]");
            //访问google.cn的报头，采用GET方式，
            Serial.println("[Sending a request]");
            client.println(String("GET/")+
                           "HTTP/1.1\r\n"+
                           "Host:"+host+"\r\n"+
                           "Connection: close\r\n"+
                           "\r\n"
                           );

            Serial.println("[Response:]");
            while(client.connected()||client.available())       //等待如果客户端已
经连接，并且有数据到来
            {
                    if(client.available())
                    {
                        String line = client.readStringUntil('\n');     //从缓存区读
取服务器的数据
                        Serial.println(line);
                    }
                }
                client.stop();              //停止客户端
                Serial.println("\n[Disconnected]");
        }
        else
        {
```

```
            Serial.println("connection failed !");
            client.stop();
            }
            delay(5000);
    }
```

HTTP协议简介

当使用浏览器(客户端)访问一个网页时,大致经过了以下三步:

- 用户输入网址后,浏览器(客户端)会向服务器 **发出HTTP请求** ;
- 服务器收到请求后会返回HTML形式的文本以响应请求;
- 浏览器收到服务器返回的HTML文本后,将文本转换为网页显示出来。

1. HTTP请求 当客户端访问网页时,会先发起HTTP请求。HTTP请求由三部分组成,分别是 **请求行**、**请求报头** 和 **请求正文**。

2. HTTP响应 服务器在接收到HTTP请求消息后,会返回一个响应消息。HTTP响应也是由三个部分组成,分别是状态行、响应报头和响应正文。

UDP协议

用户数据报UDP有两个字段:数据字段和首部字段。首部字段很简单，只有8个字节(图5-5)，由四个字段组成，每个字段的长度都是两个字节。各字段意义如下:

(1)源端口 源端口号。在需要对方回信时选用。不需要时可用全0。 (2)目的端口 目的端口号。这在终点交付报文时必须使用。 (3）长度 UDP用户数据报的长度，其最小值是8（仅有首部)。 (4)检验和 检测UDP用户数据报在传输中是否有错。有错就丢弃。

程序流程：

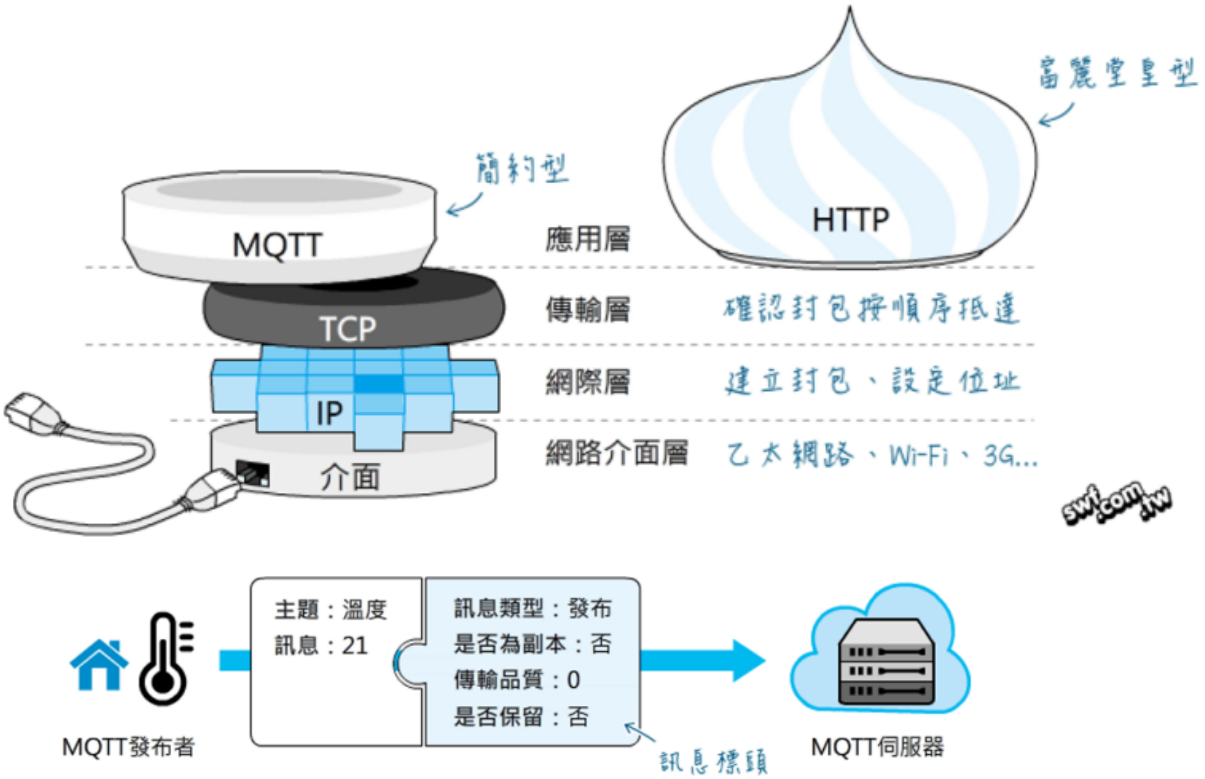
- 设置ESP8266工作在station模式（ WiFi.mode(WIFI_STA)) ;②将ESP8266接入到wIFI网络中 WiFi.begin(SSID,PSSD);
- 实例化WiFiUDP对象指定端口，建立UDP连接，连接后自动开始侦听传入的数据包。 WiFiUDP Udp; Udp.begin(localUdpPort);
- 等待输入的UDP包处理收到的数据并回复。
    - Udp.parsePacket ( ) ;
    - Udp.read(incomingPacket,255);
    - Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
    - Udp.write(replyPacket);
    - Udp.endPacket();

```
    /*
    1、设置ESP8266工作在station模式（WiFi.mode(WIFI_STA)）;
    2、将ESP8266接入到WIFI网络中WiFi.begin(SSID,PSSD);
    3、实例化WiFiUDP对象指定端口，建立UDP连接，连接后自动开始侦听传入的数据包。
    WiFiUDP Udp; Udp.begin(localUdpPort);
    4、等待输入的UDP包处理收到的数据并回复。
    Udp.parsePacket （）;
```

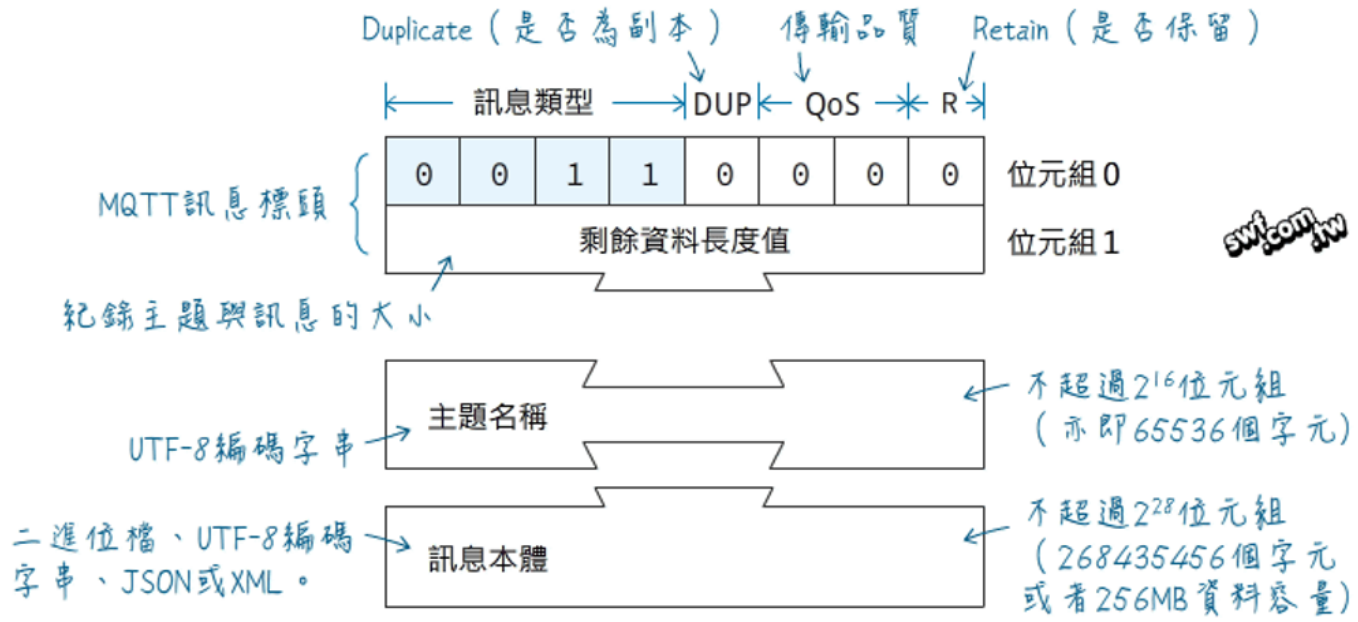
```
    Udp.read(incomingPacket, 255);
    Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
    Udp.write(replyPacket);
    Udp.endPacket();
    */

    #include <ESP8266WiFi.h>
    #include <WiFiUdp.h>

    const char* ssid = "iPhone";
    const char* password = "peng2020";

    WiFiUDP Udp;
    unsigned int localUdpPort = 4210;  // local port to listen on
    char incomingPacket[255];  // buffer for incoming packets
    char  replyPacket[] = "Hi there! Got the message :-)";  // a reply string to
send back

    void setup() {
    // put your setup code here, to run once:
    Serial.begin(115200);
    Serial.println();

    Serial.printf("Connecting to %s ", ssid);
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);             //开始连接WIFI
    while (WiFi.status() != WL_CONNECTED)    //等待连接
    {
        delay(500);
        Serial.print(".");
    }
    Serial.println(" connected");

    Udp.begin(localUdpPort);      //指定本地连接端口号
    Serial.printf("Now listening at IP %s, UDP port %d\n",
WiFi.localIP().toString().c_str(), localUdpPort);

    }

    void loop() {
    // put your main code here, to run repeatedly:
    int packetSize = Udp.parsePacket(); //调用函数解析数据包
    if (packetSize)
    {
        // receive incoming UDP packets
        Serial.printf("Received %d bytes from %s, port %d\n", packetSize,
Udp.remoteIP().toString().c_str(), Udp.remotePort());
        int len = Udp.read(incomingPacket, 255);     //从缓存区读取数据
        if (len > 0)
        {
        incomingPacket[len] = 0;    //给缓存区加结束符
        }
        Serial.printf("UDP packet contents: %s\n", incomingPacket);
```

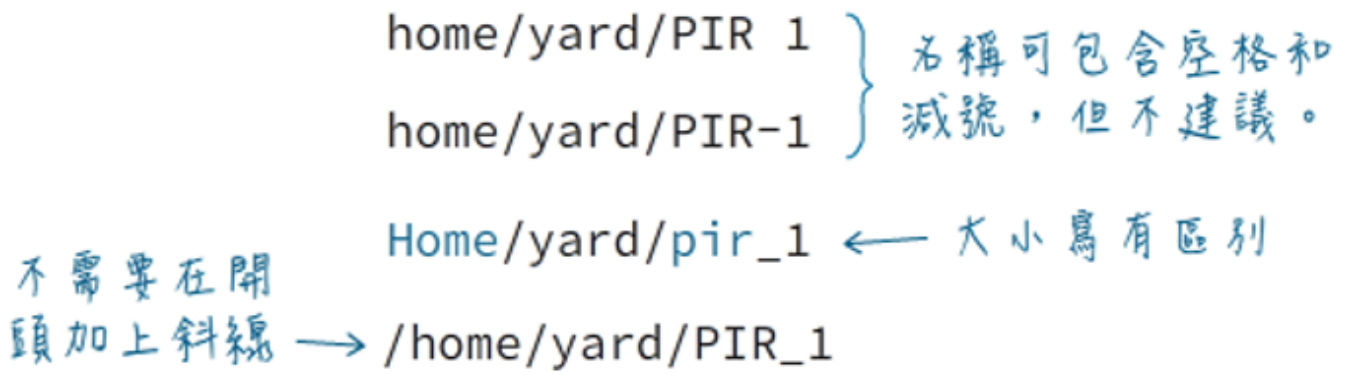
```
        // send back a reply, to the IP address and port we got the packet from
        Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());     //指定远程连接地址和
端口号
        Udp.write(replyPacket);
        Udp.endPacket();
    }
    }
```

HTTP与MQTT协议对比

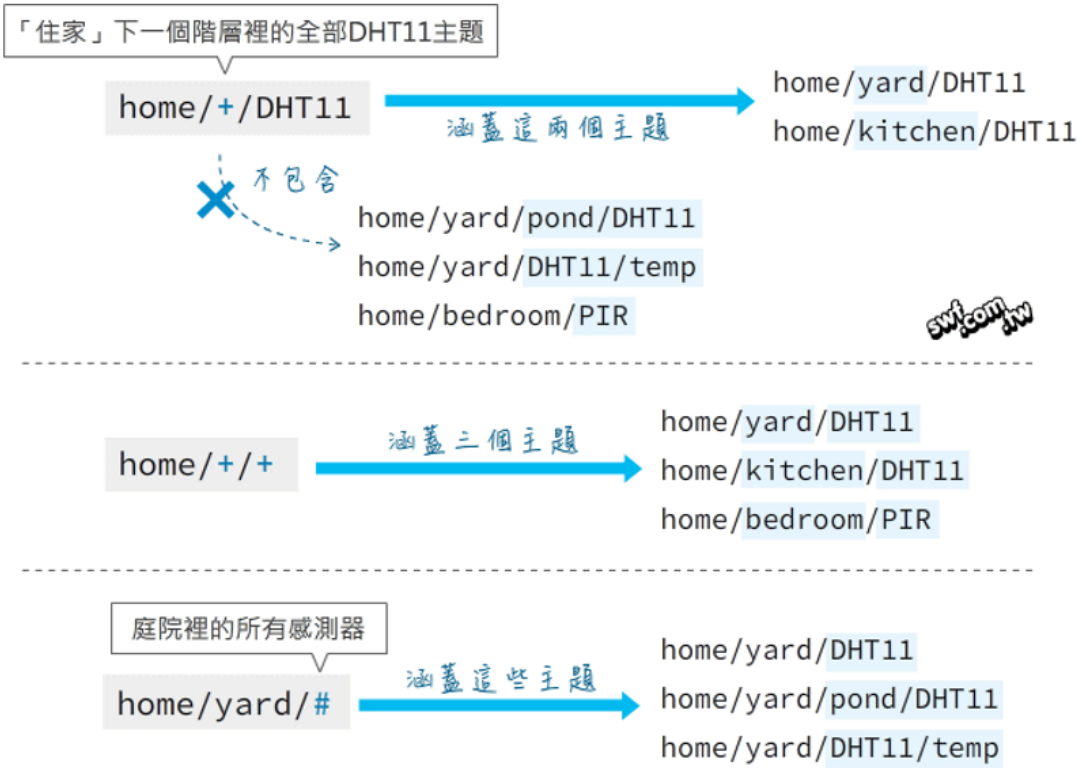


不同于HTTP的标头采用文字描述，MQTT的标头採用数字编码，整个长度只占2位元组，等同两个字元，后面跟著讯息的主题(topic)和内容(payload)，实际格式如下:



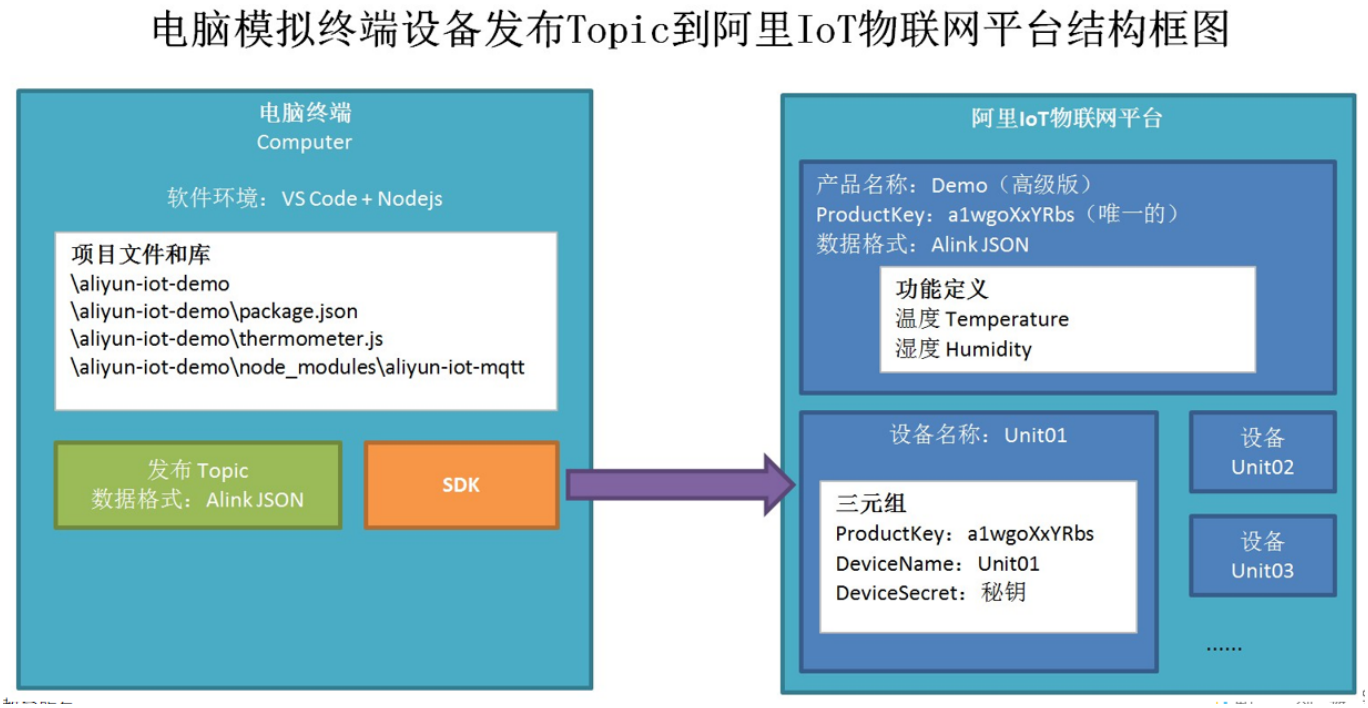


订阅者可透过下列三个叙述，一次订阅多个主题：



QoS(品质)设定 QoS代表发布者与代理人，或者代理人与订阅者之间的传输品质。MQTT定义了0,1和2三个层级的品质设定（实际支援情况依伺服器软体而定，Mosquitto伺服器全都支援):0:最多传送一次(at most once) 1:至少传送一次(at least once)2:确实传送一次(exactly once) 用寄信来比喻，QoS0就像寄平信，不保证讯息会送达。

阿里云平台 物联网云平台：接收设备上报的数据、向设备下发数据、对数据进行转发/分析/计算/显示、管理设备...




```
    #include <ESP8266WiFi.h>//安装esp8266arduino开发环境
    static WiFiClient espClient;

    #include <AliyunIoTSDK.h>//引入阿里云 IoT SDK
    //需要安装crypto库、PubSubClient库

    //设置产品和设备的信息，从阿里云设备信息里查看
    #define PRODUCT_KEY     "k0mzriwvgtZ"//替换自己的PRODUCT_KEY
    #define DEVICE_NAME     "LED2"//替换自己的DEVICE_NAME
    #define DEVICE_SECRET   "674c24074d05a31a672094c0d23dc07c"//替换自己的
DEVICE_SECRET
    #define REGION_ID       "cn-shanghai"//默认cn-shanghai

    #define WIFI_SSID       "LW"//替换自己的WIFI
    #define WIFI_PASSWD     "ping123456"//替换自己的WIFI

    #define PIN_LED   2      //LED

    unsigned long lastMsMain = 0;

    void setup()
    {
    Serial.begin(115200);
    //pinMode(LED_BUILTIN, OUTPUT);
    // digitalWrite(LED_BUILTIN, HIGH);
```

```
    pinMode(PIN_LED, OUTPUT);
    digitalWrite(PIN_LED, HIGH);

    //连接到wifi
    wifiInit(WIFI_SSID, WIFI_PASSWD);

    //初始化 iot，需传入 wifi 的 client，和设备产品信息
    AliyunIoTSDK::begin(espClient, PRODUCT_KEY, DEVICE_NAME, DEVICE_SECRET,
REGION_ID);

    //绑定一个设备属性回调，当远程修改此属性，会触发LED函数
    AliyunIoTSDK::bindData("LED2", LED);
    }

    void loop()
    {
    AliyunIoTSDK::loop();//必要函数

    if (millis() - lastMsMain >= 2000)//每2秒发送一次
    {
        lastMsMain = millis();

        //发送LED状态到云平台（高电平：1；低电平：0）
        AliyunIoTSDK::send("LED2PIN", digitalRead(PIN_LED));
    }
    }

    //wifi 连接
    void wifiInit(const char *ssid, const char *passphrase)
    {
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, passphrase);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(1000);
        Serial.println("WiFi not Connect");
    }
    Serial.println("Connected to AP");
    }

    //灯的属性修改的回调函数
    void LED(JsonVariant L)//固定格式，修改参数1
    {
    int LED = L["LED2"];//参数1，与标识符一致
    if (LED == 0)
    {
        digitalWrite(PIN_LED, HIGH);
    }
    else
    {
        digitalWrite(PIN_LED, LOW);
    }
    Serial.printf("收到的LED是："); Serial.println(LED);
    }
```