GizWits（机智云）平台 WiFi 通讯通用数据协议

（V3.0.2）

Contents

修改历史

版本	修改内容	修改时间	修改人
v1.0	确定各通讯实体间数据协议	2013-8-23	刘冠华 (Johnson)
v1.04	描术并举例不同协议间的嵌套和转换关系；调整了 WiFi 模组与设备主控 MCU 通讯的协议消息头;解决大循环下一对一回复的问题	2013-10-16	刘冠华 (Johnson)
v1.05	WiFi 模组响应发现请求中加入 MAC（二 4.）;手机请求获取绑定设备列表（三 8.）;WiFi 模组上电后广播自己的存在（二 5.）;手机与设备绑定会有确认结果（三 5.）	2014-2-11	刘冠华 (Johnson)
v1.06	明确 username 的命名规范（p11）；添加了“手机定阅修改用户名和密码响应”协议（三.9）	2014-3-13	刘冠华 (Johnson)
v1.07	添加 WiFi 模组向设备主控 MCU 请求 productKey 协议（五.6，五.7）	2014-3-13	刘冠华 (Johnson)
v1.08	修改了手机或 WiFi 模组与设端的通信的 topic（三，四）；说明当手机修改了 username 和 password 之后必须重连接云端（四.1）；	2014-3-14	刘冠华 (Johnson)

v1.09	修改了用户向云端注册时的返回码（三.1）；修改了 WiFi 模组向云端注册时的返回码（四.1）；明确了手机与云端及 WiFi 模组与云端 MQTT 连接中的 ClientId 的值（三.1.2.3，四.1.2.3）；	2014-3-16	刘冠华 (Johnson)
v1.10	修改了三.9，三.10 cmd 的值（解决了 cmd 冲突）；修改了用户向云端修改用户名和密码的返回码（三.10）；WiFi 模组与设备主控 MCU 的通讯协议中添加了数据据长度；	2014-3-18	刘冠华 (Johnson)
v1.11	修改了协议中 protocolVer 的值（由原 0x00000002 变为 0x00000003，P0 中的 protocolVer 除外）	2014-3-19	刘冠华 (Johnson)
v1.12	优化了手机与云端及 WiFi 模组与云端的 MQTT Topic 格式；	2014-3-20	刘冠华 (Johnson)
v1.13	更正了 productKey 的说明文字（p13，改为“hard code 在设备主控 MCU 的固件中”）；	2014-3-20	刘冠华 (Johnson)
v1.14	添加了“设备上线和下线通知”协议（p23，“三.13”和“三.14”）	2014-3-26	刘冠华 (Johnson)
v1.2.1	版本号改为以“x.x.x”的方式命名；在“7.手机定阅获取绑定设备列表请求的响应（云端 ->手机）”中添加了 devProtocolVer(4B)字段(p23)；在“13.设备上线通知（云端 ->手机）”中添加了 devProtocolVer(4B)字段(p23)；	2014-3-29	刘冠华 (Johnson)
v1.2.2	1. 明确指出手机或设备 MQTT CONNECT 云端时的 username 和 password 参数(三.3,四.3)； 2. 在“三 7.手机定阅获取绑定设备列表请求的响应”中添加了 busiProtocolVer(4B)字段和 isOnline 字段(p23)； 3. 合并了原“三.13 设备上线通知”和“三.14 设备下线通知”为“三.13 设备上线下线状态变更通知”（p24）； 4. 添加了“二.13 手机向 WiFi 模组查询版本号”和“二.14 WiFi 模组向手机返回所支持的版本号”（p22）； 5. 在“四.8 WiFi 模组请求获取最新的 WiFi 模组固件 Id (OTA)”中添加了版号号参数（p26）； 6. 把“五.6 WiFi 模组向设备主控 MCU 请求 productKey”更名为“五.6 WiFi 模组向设备主控 MCU 请求设备信息”（p22）； 7. 把“五.7 设备主控 MCU 向 WiFi 模组回复 productKey”更名为“五.7 设备主控 MCU 向 WiFi 模组回复设备信息”（p22）； 8. 在“五.7 设备主控 MCU 向 WiFi 模组回复设备信息”加添了版本号字段（p28）；	2014-4-11	刘冠华 (Johnson)
v1.2.3	1. 添加了协议“五.8 设备主控 MCU 请求 WiFi 模组进入 onboarding 模式”和“五.9 WiFi 模组通知设备主控 MCU onboarding 结果”	2014-5-4	刘冠华 (Johnson)
v1.2.4	1. 把 username 的最大长度变更为 64B.（p23）	2014-5-14	刘冠华 (Johnson)
v1.2.5	1. 添加了对 protocolVer 和 cmd 的说明，说明是按字节数组的方式发送和解释（而不是看成一个整数）。（p12）	2014-6-5	刘冠华 (Johnson)

	2. 添加了协议“五.10 设备主控 MCU 请求 WiFi 模组进入可配对模式”。（p28）		
v1.2.6	1. 修改“二、8. WiFi 模组响应手机的 Passcode 请求”，加入 result 字段，用于当 WiFi 模组没有处在“配对”模式时返回不成功的结果。（p22） 2. 修改“三、13. 设备上线下线状态变更通知”，加入了 varLen 和 flag 字段，即使用 Hi 的包头方式，以便手机 SDK 收到 topic 后可以通过统一的包头格式处理数据包。（p25）	2014-6-10	刘冠华 (Johnson)
v1.2.7	1. 明确“四、8. WiFi 模组请求获取最新的 WiFi 模组固件 Id (OTA)”中在设备首次上电且不确定当前的固件 id 时关于参数“currentFirmwareId”的值。（p27） 2. 明确“四、7. WiFi 模组定阅获取最新的 WiFi 模组固件 Id 的响应(OTA)”中，当云端没对应的固件时，“latestFirmwareId”的值为 0，设备无需作后续的获取固件请求。（p27）	2014-6-18	刘冠华 (Johnson)
v1.2.8	1. 添加“二、15.手机向 WiFi 模组请求 WiFi 热点列表”和“二、16. WiFi 模组向手机响应 WiFi 热点列表”。 2. 添加“三、14.手机定阅取消用户与设备间绑定响应”和“三、15. 手机取消用户与设备间的绑定”。	2014-7-1	刘冠华 (Johnson)
v1.2.9	1. 在“三、14.手机定阅取消用户与设备间绑定响应”中添加了返回结果 result=2(other error)的情况。	2014-7-9	刘冠华 (Johnson)
v1.2.10	1. 删除“四、5WiFi 模组变更 Passcode（WiFi 模组->云端）”。 2. 添加“五、11 设备主控 MCU 请求重置 WiFi 模组”。	2014-7-16	刘冠华 (Johnson)
v1.2.11	1. 在“二、4. WiFi 模组响应发现请求”中添加 product_key 返回。 2. 在“二、5. WiFi 模组上电后广播自己的存在”中添加 product_key 返回。	2014-7-17	刘冠华 (Johnson)
v1.2.12	1. 在“三、7. 手机定阅获取绑定设备列表请求的响应”中添加 product_key 返回。 2. 在“三、13. 设备上线下线状态变更通知”中添加 product_key 返回。 3. 把列表的表示方式修改为：[{列表元素}...]	2014-7-24	刘冠华 (Johnson)
v1.2.13	1. 在“六、手机与设备主控 MCU 的通讯协议（P0）”的各 P0 协议中，如协议包涵数组，则数组前统一加上数组元素的个数 elementCount(int, 2B)。	2014-7-25	刘冠华 (Johnson)
v1.2.14	本版本覆盖 v1.2.12 所做的修改。 1. 在“三、7. 手机定阅获取绑定设备列表请求的响应”中取消 product_key 返回。 2. 在“三、13. 设备上线下线状态变更通知”中取消 product_key 返回。 3. 添加“三、16. 手机定阅获取绑定设备列表请求的响应 v2”。	2014-7-28	刘冠华 (Johnson)

	4. 添加“三、17. 手机请求获取绑定设备列表 v2”。		
v1.2.15	去除 username 不能是全数字的限制。	2014-7-31	刘冠华 (Johnson)
v1.2.16	添加更多的 username 的允许字符：username 允许由‘@’，‘-’或‘.’组成。p14	2014-8-4	刘冠华 (Johnson)
v1.2.17	1.添加“五、12.设备主控 MCU 请求 WiFi 模组的状态信息”。 2.添加“五、13. WiFi 模组向设备主控 MCU 发送 WiFi 状态”。 3. 添加“二、17.手机向 WiFi 模组设置 WiFi 模组的串口通讯参数”。 4. 添加“二、18.WiFi 模组向手机响应设置 WiFi 模组的串口通讯参数结果”。	2014-8-4	刘冠华 (Johnson)
v1.2.18	1. 添加“二、19.手机设置 WiFi 模组的日志与指示灯等各种开关”。 2. 添加“二、20. WiFi 模组对手机设置 WiFi 模组日志与指示灯等各种开关的响应”。 3. 添加“二、21. WiFi 模组向手机发送日志”。 4. 添加“三、18.手机通过云端透传命令到 WiFi 模组”。 5. 添加“三、19. 手机通过云端接收来自 WiFi 模组的命令”。 6. 添加“四、14.WiFi 模组接收来自手机对 WiFi 模组的命令”。 7. 添加“四、15.WiFi 模组通过云端透 WiFi 模组命令到特定的手机”。 7. 添加“四、16.WiFi 模组通过云端传送命令到所有与设备有关联的手机”。	2014-8-12	刘冠华 (Johnson)
v1.2.19	1. 删除“五、12. 设备主控 MCU 请求 WiFi 模组的状态信息”； 2. 删除“五、13. WiFi 模组向设备主控 MCU 发送 WiFi 状态”； 3. 添加“五、12. WiFi 模组状态变化后，向设备主控 MCU 发送最新状态”； 4. 添加“五、13. 设备主控 MCU 给 WiFi 模组返回对收到 WiFi 状态的 ACK”； 5. 添加“五、14. WiFi 模组向设备主控 MCU 发送心跳”； 6. 添加“五、15. 设备主控 MCU 给 WiFi 模组返回心跳 ACK”；	2014-8-15	陈松峰、刘冠华
v1.3.0	1. 添加“二、22. 手机向 WiFi 模组请求 WiFi 模组信息”； 2. 添加“二、23. WiFi 模组响应手机的信息请求”； 3. 添加“二、24. 手机向 WiFi 模组发送心跳请求”；	2014-9-5	刘冠华

	4. 添加“二、25. WiFi 模组响应手机的心跳请求”； 5. 添加“三、1. 手机定阅用户注册响应”中的错误代码 2,3,4,5； 6. 修改“三、3. 用户登陆云端”中 MQTT Username 和 MQTT Password 字段的填写格式，添加了以 uid+token+appid 的登陆方式； 7. 添加“四、1. WiFi 模组定阅设备注册响应”中的错误代码 2,3,4,5,6； 8. 不推荐使用“二、13. 手机向 WiFi 模组查询版本号”； 9. 不推荐使用“二、14. WiFi 模组向手机返回所支持的版本号”；		
v1.3.1	1. 修改了“三、3. 用户登陆云端”的描述，把“2$appid$username”修正为“2$appid$uid”； 2. 修改了“二、24. 手机向 WiFi 模组发送心跳请求”的描述，细化了手机与 WiFi 模组对心跳的处理方式；	2014-9-5	刘冠华
v1.3.2	1. 把原 uid 更名为 username，避免与新版的 uid 存在歧义。	2014-9-8	刘冠华
v1.3.3	1. 向“五、12. WiFi 模组状态变化后，向设备主控 MCU 发送最新状态”添加了 WiFi 信号强度值，且定时每 10 分种发一次。	2014-9-25	刘冠华
V1.3.4	1. 添加“四、17. WiFi 模组定阅最新的设备主控 MCU 固件 Id 消息 (OTA)”； 2. 添加“四、18. WiFi 模组定阅获取最新的设备主控 MCU 固件 Id 的响应 (OTA)”； 3. 添加“四、19. WiFi 模组请求获取最新的设备主控 MCU 固件 Id (OTA)”； 4. 添加“四、20. WiFi 模组定阅获取设备主控 MCU 固件响应(OTA)”； 5. 添加“四、21. WiFi 模组获取设备主控 MCU 固件(OTA)”； 6. 修改“二、5. WiFi 模组上电后广播自己的存在”的 udp 端口为 2415； 7. 修改“二、8. WiFi 模组响应手机的 Passcode 请求（用户绑定设备）”把 result 字段的位置后移了；	2014-9-29	刘冠华
V1.3.5	1. 删除废弃的“四、16. WiFi 模组定阅最新的设备主控 MCU 固件 Id 消息 (OTA)” 2. 删除废弃的“四、17. WiFi 模组定阅获取最新的设备主控 MCU 固件 Id 的响应 (OTA)” 3. 删除废弃的“四、18. WiFi 模组请求获取最新的设备主控 MCU 固件 Id (OTA)” 4. 删除废弃的“四、19. WiFi 模组定阅获取设备主控 MCU 固件响应(OTA)” 5. 删除废弃的“四、20. WiFi 模组获取设备主控 MCU 固件(OTA)” 6. 删除废弃的“四、5. WiFi 模组定阅最新的固件 Id 消息 (OTA)” 7. 添加 OTA v4 的“四、15. WiFi 模组定阅固件升级通知 (OTA v4)”	2014-10-23	刘冠华

V1.3.6	1. 删除废弃的“二、17.手机向 WiFi 模组设置 WiFi 模组的串口通讯参数” 2. 删除废弃的“二、18. WiFi 模组向手机响应设置 WiFi 模组的串口通讯参数结果”	2014-11-20	刘冠华
V1.3.7	1. 添加“四、16. WiFi 模组收到在线手机数量变化通知” 2. 在“五、12. WiFi 模组状态变化后，向设备主控 MCU 发送最新状态”中添加“是否有手机在线”标志位	2014-11-29	刘冠华
V1.3.8	1. 修改“四、17. WiFi 模组收到在线手机数量变化通知”的指令命令字 2. 添加“四、16. WiFi 模组请求返在线手机数量”	2014-12-05	刘冠华
V1.3.9	1. 明确如何在 WiFi 模组工作状态中判断是使用了 SoftAP/Web 方式还是使用了 AirLink 方式进行 OnBoarding （五、12）	2014-12-09	刘冠华
V3.0.0	1. 在“五、12. WiFi 模组状态变化后，向设备主控 MCU 发送最新状态”中添加“是否处于产测模式”标志位 2. 添加“五、16.设备主控 MCU 请求 WiFi 模组进入产测模式” 3. 添加“五、17. WiFi 模组向设备主控 MCU 返回产测模式 ACK” 4. 添加“二、24. 手机请求 WiFi 模组退出产测模式” 5. 添加“二、25. WiFi 模组响应手机的退出产测模式请求” 6. 把“五、15. WiFi 模组向设备主控 MCU 返回心跳 ACK”中的命令字由 0x030E 更正为 0x030F 7. 由于此为 v3 版协议，协议号统一由 v3.0.0 开始命名	2014-12-28	刘冠华
V3.0.1	添加“二、26. WiFi 模组广播已通过 AirLink 成功配置”	2015-01-05	刘冠华
V3.0.2	规定 MQTT ClientId 的字符取值范围为只能包含大小写英文字母，数字 0 到 9，即字符[0-9a-zA-Z]	2015-07-07	刘冠华

一. 总述

1. 本协议涉及到的主要通讯实体

1) **手机**:以 App 的形式为用户提供交互界面（图 1 中 Mobile Phone）

2) **WiFi 模组:**安装在设备端的无线通讯单元（图 1 中 WiFi Module）
3) **设备主控 MCU:**负责具体设备业务逻辑的控制单元（图 1 中 Device MCU）
4) **云端**:云端服务器（图 1 中 m2m.gizwits.com）

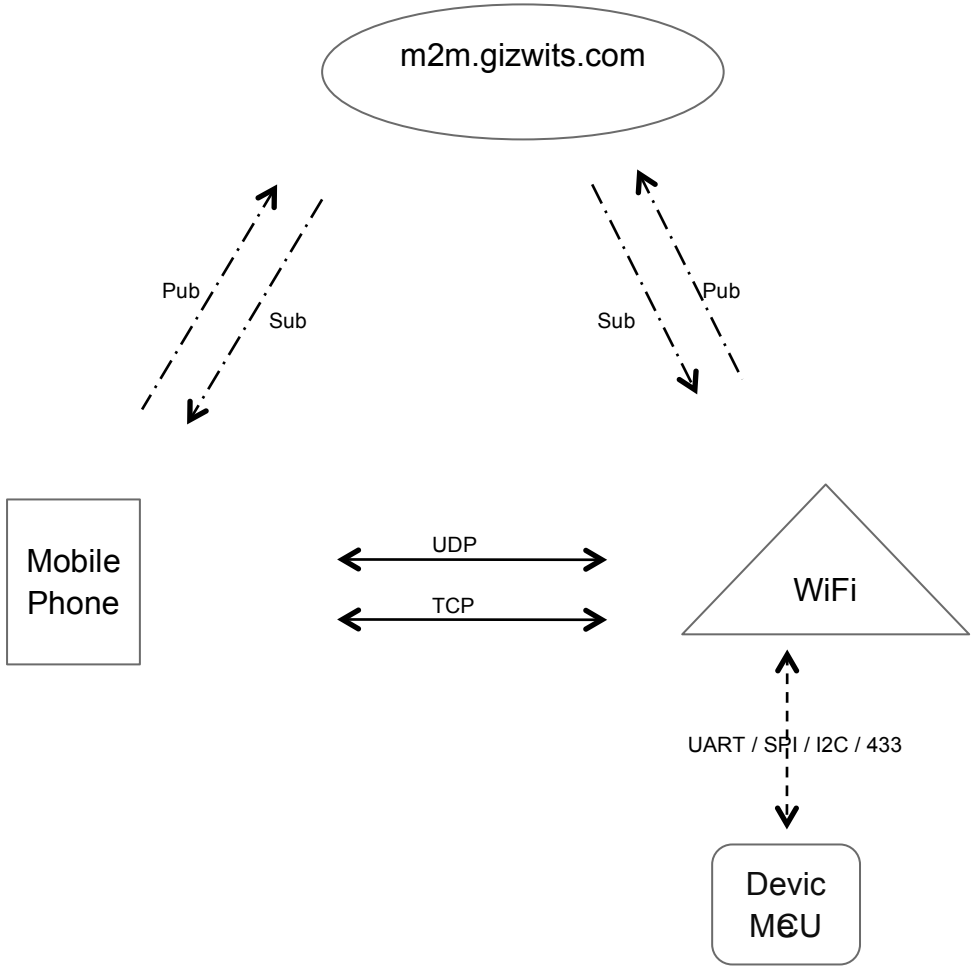


图 1：通讯过程物理连接图

2. 通讯实体间的通讯通道类型

1) **手机与 WiFi 模组**:局域网的 UDP/TCP 通讯
2) **手机与云端**: MQTT 发布与定阅
3) **WiFi 模组与云端**: MQTT 发布与定阅
4) **WiFi 模组与设备主控 MCU**: UART/SPI/I2C/443 等

3. 本协议覆盖内容

1) 手机与 WiFi 模组间的局域网通讯协议（图 2 的线 1）
   - 手机初始化 WiFi 模组（OnBoarding）
   - 手机发现 WiFi 模组
   - 用户绑定 WiFi 模组
   - 用户登陆 WiFi 模组
   - 手机通过 WiFi 模组透传命令到设备主控 MCU
   - WiFi 模组向手机透传来自设备主控 MCU 的命令

2) 手机与云端间的通讯协议（图 2 的线 2）
   - 用户向云端注册
   - 用户登陆云端
   - 用户在云端保存与设备间的绑定信息
   - 手机通过云端（然后再通过 WiFi 模组）透传命令到设备主控 MCU
   - 手机通过云端（通过 WiFi 模组再到云端）接收来自设备主控 MCU 的命令

3) WiFi 模组与云端间的通讯协议（图 2 的线 3）
   - WiFi 模组向云端注册

- WiFi 模组登陆云端
- WiFi 模组变更自身 Passcode
- WiFi 模组接收新固件通知、查询及下载 WiFi 模组固件及设备主控 MCU 固件进行 OTA 升级
- WiFi 模组经云端透传设备主控 MCU 的命令到手机
- WiFi 模组向设备主控 MCU 透传来自手机的命令（首先经过云端透传）

4) WiFi 模组与设备主控 MCU 的通讯协议（图 2 的线 4）
   - WiFi 模组向设备主控 MCU 透传来自手机的命令（直接来自手机或间接通过云端）
   - 设备主控 MCU 请求 WiFi 模组向手机转发设备主控 MCU 的命令（直接向手机或间接通过云端向手机）
   - WiFi 模组与设备主控 MCU 的其它交互

5) 手机与设备主控 MCU 的通讯协议（间接通过云端或 WiFi 模组，图 2 的线 5 和线 6）
   - 手机读取设备的属性
   - 手机更改设备的属性（或手机控制设备）
   - 设备推送状态到手机

   注意：此部分的协议（图 2 的线 5 和线 6）可以由设备生产厂商根据产品特性定制，不一定需要按照此执行，本部分的手机与设备主控 MCU 的通讯协议只是建议的基于实体属性读写的通用协议格式。

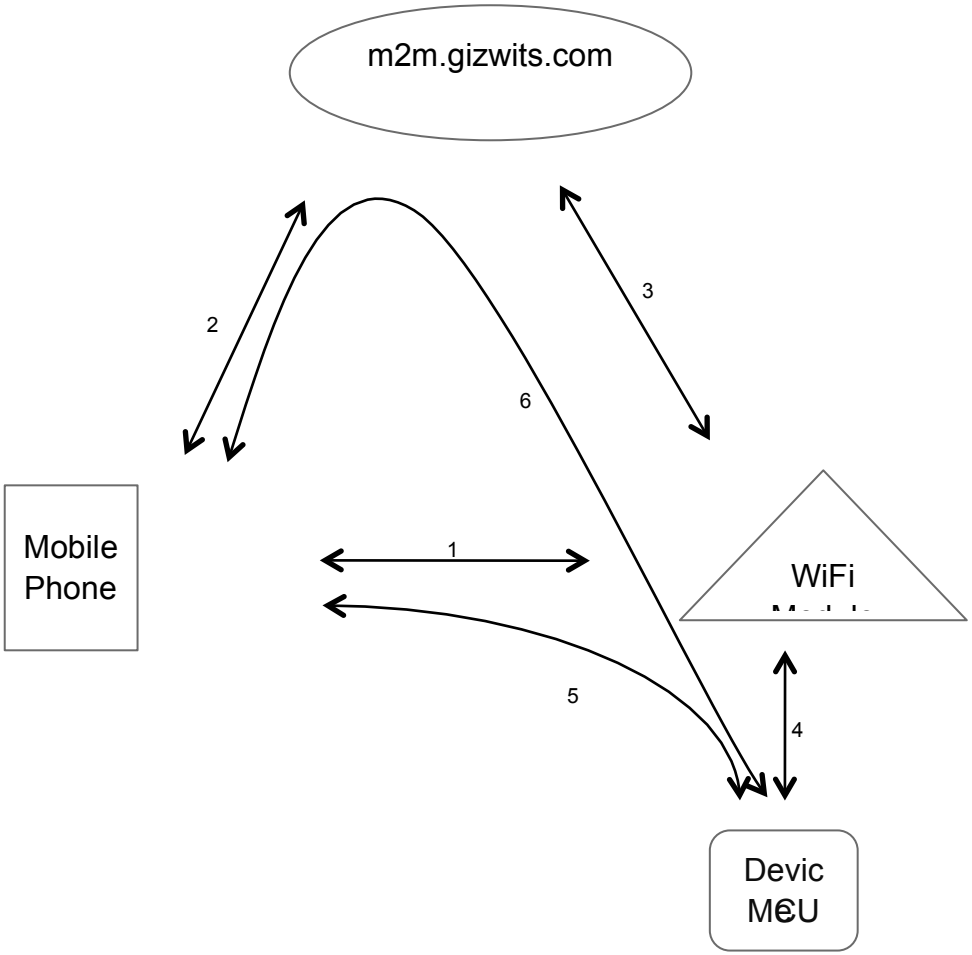


图 2：通讯协议的种类

4. 协议阅读说明

1) 发送命令的动作

- **pub**: 表示发布(publish)MQTT 消息（请参阅附件《MQTT_V3.1_Protocol_Specific.pdf》P19），消息的主题为大括号{}中的定符串，如{a/b}表示主题为”a/b”，冒号（：）后接发布的内容(Payload)

  例：pub{cli2ser/<did>}: [Pc] protocolVer(4B)=0x00000003, cmd(2B)=0x0205, productKeyLen(2B), productKey(str, max 32B), currentFirmwareId(int, 8B)

- **sub**: 表示定阅(subscribe)MQTT 消息（请参阅附件《MQTT_V3.1_Protocol_Specific.pdf》P26），消息的主题为大括号{}中的定符串，如{a/b}表示主题为”a/b”，冒号（：）后接此定阅会收到的消息的内容(Payload)

  例：sub{register/<clientId>}: [Pc] protocolVer(4B)=0x00000003, cmd(2B)=0x0102, result(1B)=0(success) | 1(fail) | 2(wrong format)

- **tcp**: 表示通过 TCP 发送命令，IP 和端口号在大括号{}号中指定，如{WiFiModIP:8080}表示 WiFi 模组 IP 和 8080 端口，冒号（：）后接发送字节的内容

  例：tcp{WiFiModIP:12416}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x0007, passcodeLen(2B), passcode(str, max 32B)

- **udp**: 表示通过 UDP 发送命令，IP 和端口号在大括号{}号中指定，如{255.255.255.255:12414}表示广播 IP 到 12414 端口，冒号（：）后接发送字节的内容

  例：udp{PhoneIP:PhoneUDPPort}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x0002

- **local**: 表示通过 UART / SPI / I2C/443 等方式发送命令，冒号（：）后接发送字节的内容

  例：local: [Hd+P0] protocolVer(4B)=0x00000003, cmd(2B)=0x0301, socketId(4B)=与手机连接的 SocketId, P0

2) 数据类型

- **str:** 表示字符串(string)类型，后接占用的固定字节长度或最大字节长度，例 1: str, 32B 表示固定 32 bytes，例 2: str, max 23B 表示最大 23 bytes
- **int:** 表示整数(integer)类型，采用大端(big enidan)网络字节序，后接占用的字节长度，例: int, 2B 表示占用 2 bytes 的整型数据

3) 缩写

- **flag:** 表示标志位，最高 bit 为 1 时，表示后续还有 flag，例: flag(1B)=0x00，0x00 为值
- **ver:** 表示版本号(version)，后接占用的字节长度，例: ver(2B)=0x0001，0x0001 为值
- **protocolVer:** 表示协议版本号(protocol version)，为了后续升级协议及兼容不同产品的定制协议，后接占用的字节长度，例: protocolVer(4B)=0x00000003，0x00000003 为值，即第 0 个字节为 0x00，第 1 个字节为 0x00，第 2 个字节为 0x00，第 3 个字节为 0x03。
- **cmd**：表示命令字(command)，后接占用的字节长度，例: cmd(2B)=0x0001，0x0001 为值，即第 0 个字节为 0x00，第 1 个字节为 0x01。
- **varLen**:表示后接数据包的长度（不包括 varLen 本身），占 1~4 个 bytes，每个 byte 的最高位为 1 时表示有还有后续字节表示长度。定义详见 MQTT 的可变长度的定义：附件《MQTT_V3.1_Protocol_Specific.pdf》Page6 - Remaining Length
- **len**: 表示后接数据包的长度（不包括 len 本身），后接所占字节的长度，例:len(2B)
- **username**: 表示用户名称，字符串，只能是由字母(a-z,A-Z)，数字(0-9)，下划线(_)，@，连线符(-)，英文名点(.)组成。最大长度为 64B，例:username(str, max 64B)
- **newUsername**: 表示新用户名称

- **oldUsername**：表示原用户名称
- **password**:表示用户的密码，字符串，后接数据类型及占用的字节最大长度，例:password(str, max 32B)
- **newPassword**:表示新用户的密码
- **did**：表示设备 ID(device Id)，字符串，在 WiFi 模组向云端注册时由云端统一分配，后接数据类型及占用的字节最大长度，例:did(str, max 23B)
- **Passcode**: WiFi 模组的密码，字符串，最大长度为 32B，在 WiFi 模组向云端注册时由 WiFi 模组以随机的方式生成，当手机登陆 WiFi 模组及 WiFi 模组向云端登陆都使用此密码，此密码可以由设备重新生成并保存到云端
- **oldPasscode**：原 WiFi 模组的密码，变更 WiFi 模组的密码时传递的参数
- **newPasscode**：新 WiFi 模组密码，变更 WiFi 模组的密码时传递的参数
- **wifiModIP**：表示 WiFi 模组 IP
- **phoneIP**：表示手机 IP
- **phoneUDPPort**：表示手机的 UDP 端口号
- **phoneTCPPort**：表示手机的 TCP 端口号
- **socketId**：表示 TCP Socket 的文件描述符(File Descriptor)，为一整型，4B
- **entityId**：表示实体的 ID，后接占用的字节长度，例: entityId(2B)=0x0001，0x0001 为值
- **attrId**：表示实体的属性 ID，后接占用的字节长度，例: attrId(2B)=0x0001，0x0001 为值

- **attrVal:** 表示实体的属性值
- **checksum:** 表示对后续数据的冗余校验，计算方式为对数据按 2B 分组，若不是 2B 的倍数，在数据的最后添加一个 0x00，然后对这些 2B 分组求和，取结果的低 2B 为 checksum 的值
- **seq:** 消息序列号，用于对 QoS=1(非 MQTT)消息的标识。接收方负责以确认帧的方式返回该序列号到发送方
- **productKey**: 产品 Key，字符串，由云端统一分配给特定的产品，hard code 在设备主控 MCU 的固件中
- **mac:** MAC 地址，字符串，由大写字母和数字组成, 12B，如“00224D7AEC73”
- **clientId:**指 MQTT 中的 Client Id
- **phoneClientId**：指手机端的 MQTT Client Id

4) OTA 相关

- **firmwareId**：命令所请求的 WiFi 模组或设备主控 MCU 的固件 ID
- **requestFirmwareId**：命令所请求的 WiFi 模组或设备主控 MCU 的固件 ID
- **latestFirmwareId**：云端保存的最新 WiFi 模组或设备主控 MCU 的固件 ID
- **currentFirmwareId**：WiFi 模组或设备主控 MCU 当前运行的固件 ID
- **firmwareVer**：WiFi 模组或设备主控 MCU 的固件版本号，字符串
- **firmwareBin**：固件的内容，二进制内容

- **firmwareMD5**：对固件内容的 MD5 运算，也就是 MD5(firmwareBin)，16B

5) 其它

- **QoS=x**: 表示该发送的 MQTT 消息 QoS 级别，没有注明 QoS 的 pub 默认为 QoS=0 即可，例: QoS=1 表示 QoS 级别为 1
- **##B**: 表示字段的字节长度，等于## bytes，例:23B 表示 23 bytes
- **[{列表元素}...]**: 表示列表或数组，例:[{id(int, 4B), verLen(2B), ver(str, max 32B)}...]表示一个列表，列表的每一项都由 4B 的整型 Id, 2B 的版本字符串字节长度和最大 32B 的版本字符串组成
- **xxxLen**:表示后接字段的字节长度，一般占用 2B，例：didLen(2B)

5. 协议的种类与关系

本文档所覆盖的通讯协议中，由于不同实体间的通讯方式和通信需求不同，出现多种协议格式。

1) 各种协议格式定义（参见下图 3）：

- **P0**：Protocol0，表示手机与设备的业务逻辑通讯的通讯协议内容。
  协议格式：P0 = protocolVer(2B), len(2B) , flag(1~nB), cmd(2B), parameters
  注：P0 协议的内容可以由设备生产厂商根据产品特性定制，上为建议的基于实体属性读写的通用协议格式。

- **Hi**：Header for Internal，表示手机与 WiFi 模组（通过局域网或云端）通讯的协议消息头
  协议格式：Hi = *protocolVer(4B), varLen(1~4B) , flag(1~nB), cmd(2B)*

- **Pi**：Payload for Internal，表示手机通过局域网与 WiFi 模组通讯的协议有效负载
  对应不同 cmd，内容不同。对特定 cmd，Pi 等于 P0

- **Pc**：Payload to Cloud，表示手机与云端及 WiFi 模组与云端通讯的通讯协议内容
  对应不同 cmd，内容不同

- **He**：Header for External，表示手机通过云端与 WiFi 模组通讯的 MQTT 协议消息头
  MQTT 协议消息头（请参阅附件《MQTT_V3.1_Protocol_Specific.pdf》）

- **Hd**：Header for Device，表示 WiFi 模组与设备主控 MCU 通讯的协议消息头
  协议格式： Hd =*protocolVer(4B), cmd(2B)*

- **Pd**：Payload for Device，表示 WiFi 模组与设备主控 MCU 通讯的协议有效负载
  对应不同 cmd，内容不同。对特定 cmd，Pd 等于 P0

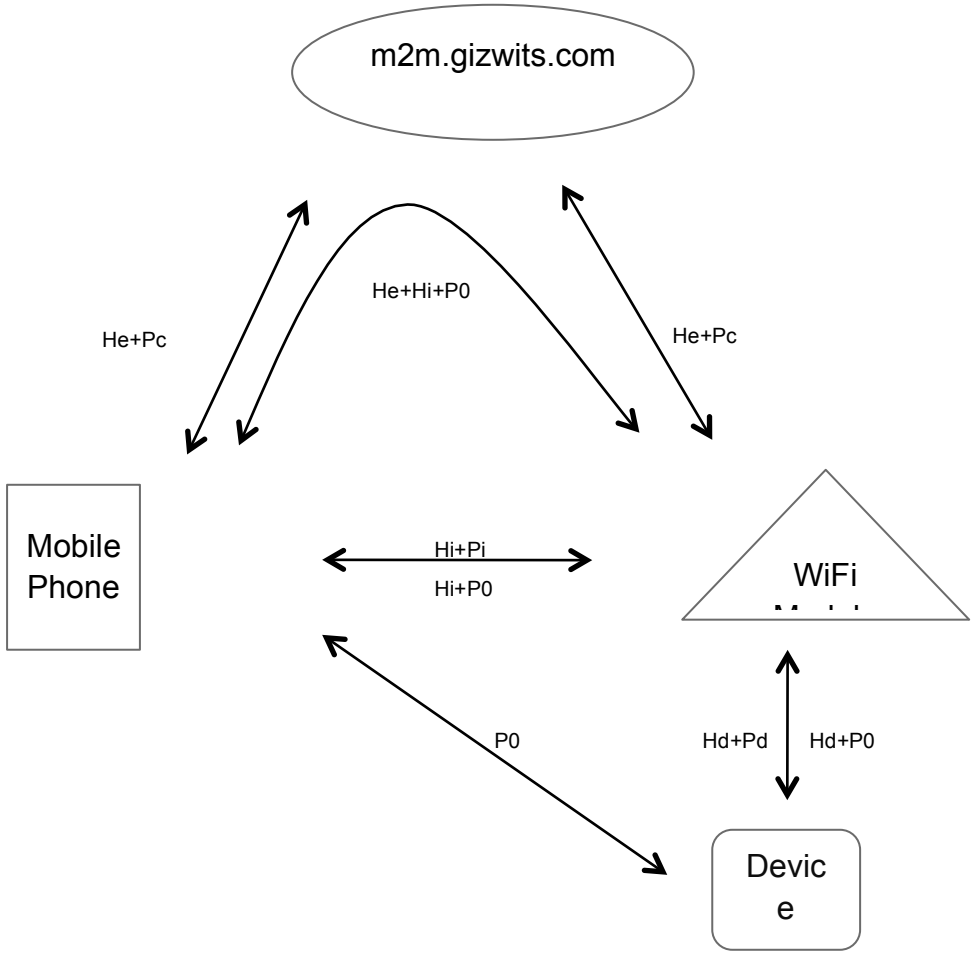


图 3：实体通讯协议的种类

2) 协议间的嵌套和转换关系

消息在通讯实体间转输时，可能会需要对协议消息头进解释与转换。现针对不同的通讯场景，下面就图 2 中的线 1，2，3，4，5，6 中的协议通讯关系进行说明。

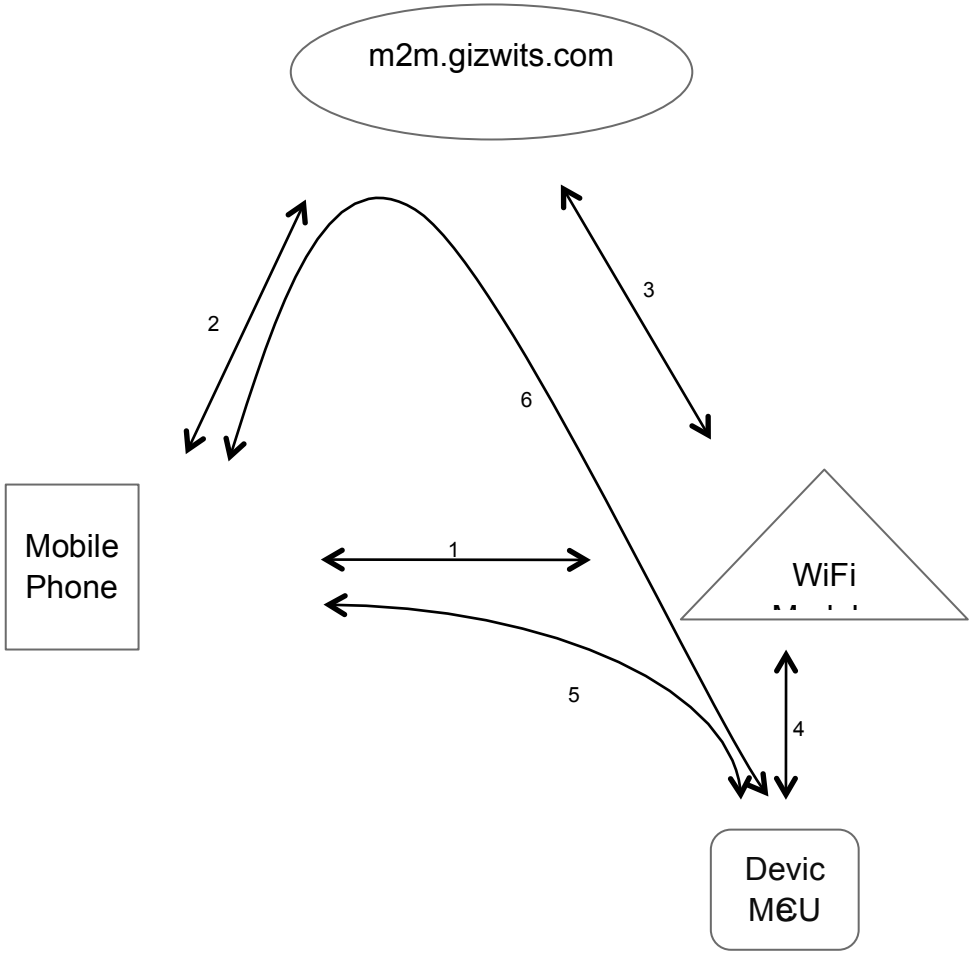
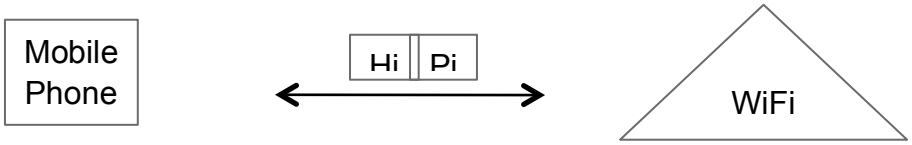
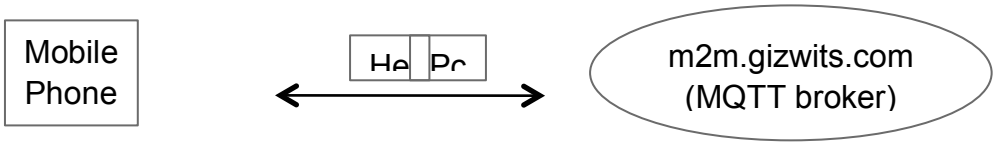


图 2：通讯协议的种类

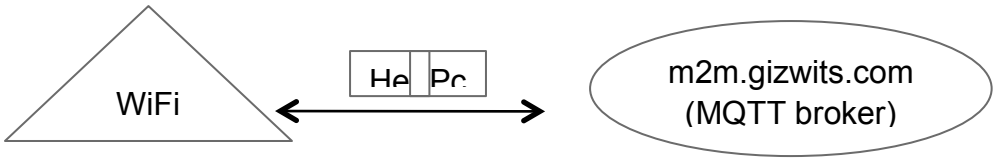
- 手机与 WiFi 模组间的局域网通讯（上图 2 的线 1）



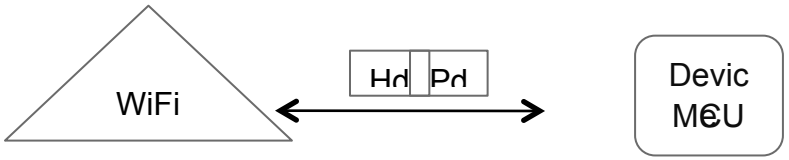
- 手机与云端间的通讯（上图 2 的线 2）



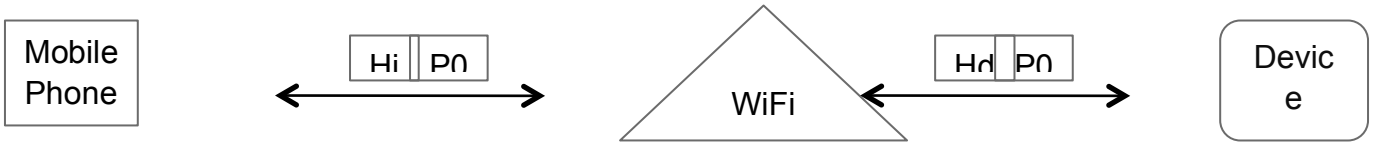
- WiFi 模组与云端间的通讯（上图 2 的线 3）



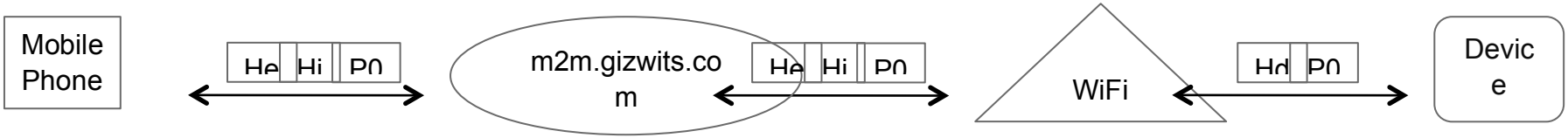
- WiFi 模组与设备主控 MCU 的通讯（上图 2 的线 4）



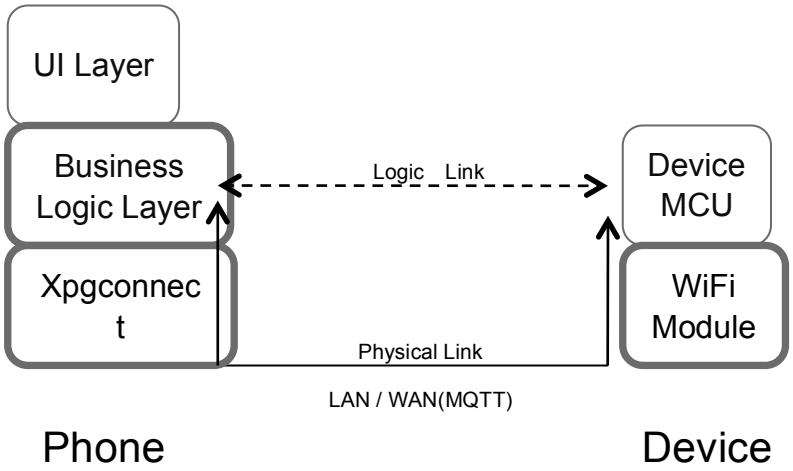
- 手机与设备主控 MCU 间接通过 WiFi 模组的通讯（上图 2 的线 5）



- 手机与设备主控 MCU 间接通过云端及 WiFi 模组的通讯（上图 2 的线 6）



- 对于上图 2 的线 5 和线 6，发即设备主控 MCU（下图中 Device MCU）和手机业务逻辑层（下图中 Business Logic Layer）之间的通讯，虽然底层经过了 WiFi 模组、云端和手机 SDK（下图 Xpgconnect SDK）等（下图中实线箭头），由于使用了嵌套协议的方式，在剥离了外层的数据协议消息头之后，逻辑上形如直接通讯一样（图 4 中虚线箭头）。



6. 关于安全的约定

1) 手机在局域网 TCP 连接上 WiFi 模组后，如没有执行登陆 WiFi 模组动作而不作任何数据请求，5 秒后 WiFi 模组应视此为非法连接而中断该 TCP 连接。
2) 手机在局域网 TCP 连接上 WiFi 模组后，如没有执行登陆 WiFi 模组动作而直接发送控制命令，WiFi 模组应视此为非法动作而中断该 TCP 连接。但标注为“手机无需登陆 WiFi 模组即可执行”的除外，这此命令可以在登陆前执行。
3) 手机与云端间及 WiFi 模组与云端间的 MQTT 通讯全部经过 TLS/SSL 加密。

二、手机与 WiFi 模组的通讯协议

1. 手机保存路由器的 SSID 和 Password 到 WiFi 模组（OnBoarding）（手机 ->WiFi 模组）

udp{255.255.255.255:12414}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x0001, ssidLen(2B), ssid(str, max 32B), passwordLen(2B), password(str, max 32B)

注：当 WiFi 模组连接不上路由器时，WiFi 模组自动进入 softAP 模式。手机发送此命令时，要求手机首先连接到 WiFi 模组的 softAP。

2. WiFi 模组向手机确认收到了路由器的 SSID 和 Password（OnBoarding）（WiFi 模组 ->手机）

udp{phoneIP:phoneUDPPort}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x0002

3. 手机发现 WiFi 模组请求（UDP 广播方式）（手机 ->WiFi 模组）

udp{255.255.255.255:12414}:[Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B) , flag(1B)=0x00, cmd(2B)=0x0003

4. WiFi 模组响应发现请求（UDP 单播方式）（WiFi 模组 ->手机）

udp{phoneIP:phoneUDPPort}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x0004, didLen(2B), did(str, max 23B), macLen(2B), mac(str, max 32B), firmwareVerLen(2B), firmwareVer(str, max 32B), productKeyLen(2B), productKey(str, max 32B)

5. WiFi 模组上电后广播自己的存在（UDP 广播方式）（WiFi 模组 ->手机）

udp{255.255.255.255:2415}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x0005, didLen(2B), did(str, max 23B), macLen(2B), mac(str, max 32B), firmwareVerLen(2B), firmwareVer(str, max 32B), productKeyLen(2B), productKey(str, max 32B)

6. 手机发现 WiFi 模组请求（Bonjour 方式）（手机<->WiFi 模组）

请参阅 https://developer.apple.com/bonjour/。

7. 手机请求 WiFi 模组的 Passcode（用户绑定设备）（手机 ->WiFi 模组）

tcp{wifiModIP:12416}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1B)(1~4B), flag(1B)=0x00, cmd(2B)=0x0006

注：手机无需登陆 WiFi 模组即可执行。

8. WiFi 模组响应手机的 Passcode 请求（用户绑定设备）（WiFi 模组 ->手机）

tcp{phoneIP:phoneTCPPort}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x0007, passcodeLen(2B), passcode(str, max 32B), result(1B)=0(success) | 1(fail)

注：只有当 WiFi 模组工作在“配对”模式下才返回 Passcode。WiFi 模组可以通过让用户按下设备上的按钮进入“配对”模式。当 WiFi 模组没有工作在“配对”模式时，result=1，此时，passcodeLen 和 passcode 无意义。

9. 用户登陆 WiFi 模组（手机 ->WiFi 模组）

tcp{wifiModIP:12416}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x0008, passcodeLen(2B), passcode(str, max 32B)

10. WiFi 模组响应用户的登陆请求（WiFi 模组 ->手机）

tcp{phoneIP:phoneTCPPort}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x0009, result(1B)=0(success) | 1(fail)

11. 手机通过 WiFi 模组透传命令到设备主控 MCU（手机 ->WiFi 模组）

tcp{wifiModIP:12416}: [Hi+P0] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x0090,P0（请见“六、手机与设备主控 MCU 的通讯协议”）

12. WiFi 模组透传设备主控 MCU 命令到手机（WiFi 模组 ->手机）

tcp{phoneIP:phoneTCPPort}: [Hi+P0] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x0091,P0（请见“六、手机与设备主控 MCU 的通讯协议”）

13. 手机向 WiFi 模组查询版本号（手机 ->WiFi 模组）- [不推荐使用]

tcp{wifiModIP:12416}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x000A

注：手机无需登陆 WiFi 模组即可执行。

14. WiFi 模组向手机返回所支持的版本号（WiFi 模组 ->手机）- [不推荐使用]

tcp{phoneIP:phoneTCPPort}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x000B, genProtocolVer(4B), busiProtocolVerLen(2B), busiProtocolVer(int)

注：参数中的“genProtocolVer”是 Wifi 模组所支持的通用数据协议的版本号，“busiProtocolVer”是对设备所支持的业务逻辑协议的版本号。

15. 手机向 WiFi 模组请求 WiFi 热点列表（手机 ->WiFi 模组）

tcp{wifiModIP:12416}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x000C

注：

1. 所请求到的 WiFi 热点列表是指 WiFi 模组所能监测到的周围的 WiFi 热点列表。
2. 手机无需登陆 WiFi 模组即可执行。

16. WiFi 模组向手机响应 WiFi 热点列表（WiFi 模组 ->手机）

tcp{phoneIP:phoneTCPPort}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x000D, [{ssidLen(2B), ssid(max 32B), signalStrength(1B)}...]

注：singalStrength 是指该 WiFi 的信号强度。

17. 手机设置 WiFi 模组的日志与指示灯等各种开关（手机 ->WiFi 模组）

tcp{wifiModIP:12416}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x0010,values1(1B),values2(1B)

注：

1. 开关值按位设置，共 16 位，编号规则为 values2 的最低位至最高位为 bit0 至 bit7，values1 的最低位至最高位为 bit8 至 bit16。每个位(bit)的定义如下：
   bit0: error 日志级别的开与关，0 为关，1 为开；
   bit1: warning 日志级别的开与关，0 为关，1 为开；
   bit2: info 日志级别的开与关，0 为关，1 为开；
   bit3: WiFi 模组所有指示灯的总开关，0 为关，1 为开；
   bit4- bit15: 保留。
2. 手机无需登陆 WiFi 模组即可执行。

18. WiFi 模组对手机设置 WiFi 模组日志与指示灯等各种开关的响应（WiFi 模组 ->手机）

tcp{phoneIP:phoneTCPPort}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x0011,result(1B)=0(success) | 1 (error)

19. WiFi 模组向手机发送日志（WiFi 模组 ->手机）

tcp{phoneIP:phoneTCPPort}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x0012,logLevel(1B)=1(error) | 2 (warning)| 3

(info),tagLen(2B),tag(str),sourceLen(2B),source(str),contentLen (2B),content(str)

注：

1. logLevel 表示日志的级别，tag 表示日志的标签，source 表示日志的来源，content 表示日志的内容。
2. 手机无需登陆 WiFi 模组即可执行。

20. 手机向 WiFi 模组请求 WiFi 模组信息（手机 ->WiFi 模组）

tcp{wifiModIP:12416}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x0013

注：手机无需登陆 WiFi 模组即可执行。

21. WiFi 模组响应手机的信息请求（WiFi 模组 ->手机）

tcp{phoneIP:phoneTCPPort}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x0014, wifiHardVer(str, 8B), wifiSoftVer(str, 8B),mcuHardVer(str, 8B), mcuSoftVer(str, 8B), p0Ver(str, 8B), firmwareId(int, 8B), firmwareVerLen(2B), firmwareVer(str, max 32B), productKeyLen(2B), productKey(str, max 32B)

22. 手机向 WiFi 模组发送心跳请求（手机 ->WiFi 模组）

tcp{wifiModIP:12416}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x0015

注：当手机超过一定时间没有从 WiFi 模组接收到数据后，需要向 WiFi 模组发送心跳包。手机心跳超时时间建议设定为 50s，当手机在 10s 内没有收到 WiFi 模组的心跳响应，可以认为已断开了与 WiFi 模组的连接。当 WiFi 模组在 60s 还没有收到手机的心跳请求可以认为与手机的连接已断开。

23. WiFi 模组响应手机的心跳请求（WiFi 模组 ->手机）

tcp{phoneIP:phoneTCPPort}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x0016

24. 手机请求 WiFi 模组退出产测模式（手机 ->WiFi 模组）

tcp{wifiModIP:12416}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x0017

注：手机无需登陆 WiFi 模组即可执行。

25. WiFi 模组响应手机的退出产测模式请求（WiFi 模组 ->手机）

tcp{phoneIP:phoneTCPPort}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x0018

26. WiFi 模组广播已通过 AirLink 成功配置（UDP 广播方式）（WiFi 模组 ->手机）

udp{255.255.255.255:2415}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x0019, macLen(2B), mac(str, max 32B), productKeyLen(2B), productKey(str, max 32B) , didLen(2B), did(str, max 23B)

注：当设备还没有 did 时，didLen 的值为零。

三、手机与云端的通讯协议

1. 手机定阅用户注册响应（云端 ->手机）

sub{register/<clientId>}: [Pc] protocolVer(4B)=0x00000003, cmd(2B)=0x0102, result(1B)=0(success) | 1(other error) | 2(wrong username length) | 3(invalid username chars) | 4(wrong password length) | 5(user is already registered)

注：注册时，MQTT 连接云端的 CONNECT 消息中无需包含用户名和密码，ClientId 填“ano”+最大 20B 的随机字符串。如该 ClientId 已被占用，MQTT 的 CONNACK 消息中的 return code 会是 0x02(identifier rejected)。

2. 用户向云端注册（手机 ->云端）

pub{register/user}: [Pc] protocolVer(4B)=0x00000003, cmd(2B)=0x0101, usernameLen(2B), username(str, max 64B), passwordLen(2B), password(str, max 32B)

注：注册时，MQTT 连接云端的 CONNECT 消息中无需包含用户名和密码，ClientId 填“ano”+最大 20B 的随机字符串（字符取值范围为只能包含大小写英文字母，数字 0 到 9，即字符[0-9a-zA-Z]）。如该 ClientId 已被占用，MQTT 的 CONNACK 消息中的 return code 会是 0x02(identifier rejected)。请注意 username 和 password 的格式要求（见“协议阅读说明”部分）。

3. 用户登陆云端（手机 ->云端）

使用 MQTT 的 CONNECT 消息，请参阅附件《MQTT_V3.1_Protocol_Specific.pdf》P15。CONNECT 中的 ClientId 填“usr”+最大 20B 的随机字符串（字符取值范围为只能包含大小写英文字母，数字 0 到 9，即字符[0-9a-zA-Z]）。

MQTT Username 和 MQTT Password 字段的填写分以下 2 种格式，

V3 版用户登陆：MQTT Username 字段填 username，MQTT Password 字段填 password。

V4 版用户登陆：MQTT Username 字段填 2$appid$uid，MQTT Password 字段填 token。

如该 ClientId 已被占用，MQTT 的 CONNACK 消息中的 return code 会是 0x02(identifier rejected)。

4. 心跳包（手机<->云端）

使用 MQTT 的 PINGREQ/PINGRESP 消息。请参阅附件《MQTT_V3.1_Protocol_Specific.pdf》P33，P34。

5. 手机定阅用户与设备绑定响应（云端 ->手机）

sub{ser2cli_res/<clientId>}: [Pc] protocolVer(4B)=0x00000003, cmd(2B)=0x0104, result(1B)=0(success) | 1(wrong passcode) | 2(did does not exist)

6. 手机在云端保存用户与设备间的绑定信息（手机 ->云端）

pub{cli2ser_req}: [Pc] protocolVer(4B)=0x00000003, cmd(2B)=0x0103, didLen(2B), did(str, max 23B), passcodeLen(2B), passcode(str, max 32B)

7. 手机定阅获取绑定设备列表请求的响应（云端 ->手机）

sub{ser2cli_res/<clientId>}: [Pc] protocolVer(4B)=0x00000003, cmd(2B)=0x0106, [{didLen(2B), did(str, max 23B), macLen(2B), mac(str, max 32B), passcodeLen(2B), passcode(str, max 32B), isOnline(1B)=0(offline)|1(online), genProtocolVer(4B), busiProtocolVerLen(2B), busiProtocolVer(int)}...]

注：列表中的“genProtocolVer”是 Wifi 模组所支持的通用数据协议的版本号，“busiProtocolVer”是对设备所支持的业务逻辑协议的版本号，当设备不在线时，字段“genProtocolVer”和“busiProtocolVer”的值没有意义。

8. 手机请求获取绑定设备列表（手机 ->云端）

pub{cli2ser_req}: [Pc] protocolVer(4B)=0x00000003, cmd(2B)=0x0105

9. 手机定阅修改用户名和密码响应（云端 ->手机）

sub{ser2cli_res/<clientId>}: [Pc] protocolVer(4B)=0x00000003, cmd(2B)=0x0108, result(1B)=0(success) | 1(invalid username len) | 2(invalid username format) | 3(invalid password len) | 4(fail with other reason)

注：修改用户名和密码后，手机应该断开与云端的连接并用新用户名和密码重新连接。

10. 手机修改用户名和密码（手机 ->云端）

pub{cli2ser_req}: [Pc] protocolVer(4B)=0x00000003, cmd(2B)=0x0107, newUsernameLen(2B), newUsername(str, max 23B), newPasswordLen(2B), newPassword(str, max 32B)

11. 手机通过云端（及 WiFi 模组）透传命令到设备主控 MCU（手机 ->云端）

pub{app2dev/<did>/<clientId>}: [Hi+P0] 见“二、11.手机通过 WiFi 模组透传命令到设备主控 MCU”

12. 手机通过云端（及 WiFi 模组）接收来自设备主控 MCU 的命令（云端 ->手机）

sub{dev2app/<did>/#}: [Hi+P0] 见“二、12.WiFi 模组透传设备主控 MCU 命令到手机”

13. 设备上线下线状态变更通知（云端 ->手机）

sub{dev2app/<did>}: [Hi+Pi] protocolVer(4B)=0x00000003, varLen(1~4B), flag(1B)=0x00, cmd(2B)=0x0092, macLen(2B), mac(str, max 32B), passcodeLen(2B), passcode(str, max 32B)), isOnline(1B)=0(offline)|1(online), genProtocolVer(4B), busiProtocolVerLen(2B), busiProtocolVer(binary)

注：参数中的“genProtocolVer”是 Wifi 模组所支持的通用数据协议的版本号，“busiProtocolVer”是对设备所支持的业务逻辑协议的版本号，当设备不在线时，字段“genProtocolVer”和“busiProtocolVer”的值没有意义。

14. 手机定阅取消用户与设备间绑定响应（云端 ->手机）

sub{ser2cli_res/<clientId>}: [Pc] protocolVer(4B)=0x00000003, cmd(2B)=0x010A, result(1B)=0(success) | 1 (binding does not exist) | 2(other error)

15. 手机取消用户与设备间的绑定（手机 ->云端）

pub{cli2ser_req}: [Pc] protocolVer(4B)=0x00000003, cmd(2B)=0x0109, didLen(2B), did(str, max 23B)

16. 手机定阅获取绑定设备列表请求的响应 v2（云端 ->手机）

sub{ser2cli_res/<clientId>}: [Pc] protocolVer(4B)=0x00000003, cmd(2B)=0x010C, [{didLen(2B), did(str, max 23B), macLen(2B), mac(str, max 32B), passcodeLen(2B), passcode(str, max 32B), productKeyLen(2B), productKey(str, max 32B), isOnline(1B)=0(offline)|1(online), genProtocolVer(4B), busiProtocolVerLen(2B), busiProtocolVer(int)}...]

注：列表中的“genProtocolVer”是 Wifi 模组所支持的通用数据协议的版本号，“busiProtocolVer”是对设备所支持的业务逻辑协议的版本号，当设备不在线时，字段“genProtocolVer”和“busiProtocolVer”的值没有意义。

17. 手机请求获取绑定设备列表 v2（手机 ->云端）

pub{cli2ser_req}: [Pc] protocolVer(4B)=0x00000003, cmd(2B)=0x010B

18. 手机通过云端透传命令到 WiFi 模组（手机 ->云端）

pub{app2dev/<did>/<clientId>}: [Hi+P0] 见“二、19. 手机设置 WiFi 模组的日志与指示灯等各种开关”

19. 手机通过云端接收来自 WiFi 模组的命令（云端 ->手机）

sub{dev2app/<did>/#}: [Hi+P0] 见“二、20. WiFi 模组对手机设置 WiFi 模组日志与指示灯等各种开关的响应”

sub{dev2app/<did>/#}: [Hi+P0] 见“二、21. WiFi 模组向手机发送日志”

四、WiFi 模组与云端的通讯协议

1. WiFi 模组定阅设备注册响应（云端 ->WiFi 模组）

sub{register/<clientId>}: [Pc] protocolVer(4B)=0x00000003, cmd(2B)=0x0202, result(1B)=0(success) | 1(other error) | 2(wrong mac length) | 3(wrong passcode length) | 4(wrong product key length) | 5(invalid product key) | 6 (not permitted), didLen(2B), did(str, max 23B)

注：注册时，MQTT 连接云端的 CONNECT 消息中无需包含用户名和密码，ClientId 填“ano”+最大 20B 的随机字符串（字符取值范围为只能包含大小写英文字母，数字 0 到 9，即字符[0-9a-zA-Z]）。如该 ClientId 已被占用，MQTT 的 CONNACK 消息中的 return code 会是 0x02(identifier rejected)。只有返回值中的 result 的值为 0 时，didLen 和 did 才有意义。

2. WiFi 模组向云端注册（WiFi 模组->云端）

pub{register/dev}: [Pc] protocolVer(4B)=0x00000003, cmd(2B)=0x0201, macLen(2B), mac(12B), passcodeLen(2B), passcode(str, max 32B), productKeyLen(2B),

productKey(str, max 32B)

注：注册时，MQTT 连接云端的 CONNECT 消息中无需包含用户名和密码，ClientId 填“ano”+最大 20B 的随机字符串（字符取值范围为只能包含大小写英文字母，数字 0 到 9，即字符[0-9a-zA-Z]）。如该 ClientId 已被占用，MQTT 的 CONNACK 消息中的 return code 会是 0x02(identifier rejected)。参数中的 productKey 从设备主控 MCU 处获得（见“五.6，五.7”）。

3. WiFi 模组登陆云端（WiFi 模组->云端）

使用 MQTT 的 CONNECT 消息。CONNECT 中的 ClientId 填 did，Username 填 did，Password 填 passcode。请参阅附件《MQTT_V3.1_Protocol_Specific.pdf》P15。

4. 心跳包（WiFi 模组<->云端）

使用 MQTT 的 PINGREQ/PINGRESP 消息。请参阅附件《MQTT_V3.1_Protocol_Specific.pdf》P33，P34。

5. WiFi 模组定阅获取最新的 WiFi 模组固件 Id 的响应 (OTA)（云端 ->WiFi 模组）

sub{ser2cli_res/<clientId>}: [Pc] protocolVer(4B)=0x00000003, cmd(2B)=0x0206, latestFirmwareId(int, 8B)

注：当云端没对应的固件时，“latestFirmwareId”的值为 0，设备无需作后续的获取固件请求。

6. WiFi 模组请求获取最新的 WiFi 模组固件 Id (OTA)（WiFi 模组->云端）

pub{cli2ser_req}: [Pc] protocolVer(4B)=0x00000003, cmd(2B)=0x0205, productKeyLen(2B), productKey(str, max 32B), currentFirmwareId(int, 8B), genProtocolVer(4B), busiProtocolVerLen(2B), busiProtocolVer(int)

注：参数中的“genProtocolVer”是 Wifi 模组所支持的通用数据协议的版本号，“busiProtocolVer”是对设备所支持的业务逻辑协议的版本号。当设备在首次上电且不确定当前的固件 id 时，“currentFirmwareId”可以填值 0，代表当前固件 id 未知。

7. WiFi 模组定阅获取固件响应(OTA)（云端 ->WiFi 模组）

sub{ser2cli_res/<clientId>}: [Pc] protocolVer(4B)=0x00000003, cmd(2B)=0x0208, firmwareId(int, 8B), firmwareVerLen(2B), firmwareVer(str, max 32B), firmwareMD5(16B),firmwareBinLen(4B), firmwareBin

8. WiFi 模组获取固件(OTA)（WiFi 模组->云端）

pub{cli2ser_req}: [Pc] protocolVer(4B)=0x00000003, cmd(2B)=0x0207, requestFirmwareId(int, 8B)

9. WiFi 模组通过云端透传来自设备主控 MCU 的命令到特定的手机（WiFi 模组->云端）

pub{dev2app/<did>/<phoneClientId>}: [Hi+P0] 见“二、12.WiFi 模组透传设备主控 MCU 命令到手机”

注：phoneClientId 是从设备主控 MCU 传过来数据中的 Hd 中得到

10. WiFi 模组通过云端透传来自设备主控 MCU 的命令到所有与设备有关联的手机（WiFi 模组->云端）

pub{dev2app/<did>}: [Hi+P0]见“二、12.WiFi 模组透传设备主控 MCU 命令到手机”

11. WiFi 模组接收并透传来自手机对设备主控 MCU 的命令（云端 ->WiFi 模组）

sub{app2dev/<did>/#}: [Hi+P0]见“二、11.手机通过 WiFi 模组透传命令到设备主控 MCU”

注：topic 中的“#”代表手机的 clientId(即 phoneClientId)。

12. WiFi 模组接收来自手机对 WiFi 模组的命令（云端 ->WiFi 模组）

sub{app2dev/<did>/#}: [Hi+P0]见“二、19. 手机设置 WiFi 模组的日志与指示灯等各种开关”

注：Topic 中的#指 phoneClientId

13. WiFi 模组通过云端透 WiFi 模组命令到特定的手机（WiFi 模组->云端）

pub{dev2app/<did>/<phoneClientId>}: [Hi+P0]见“二、20. WiFi 模组对手机设置 WiFi 模组日志与指示灯等各种开关的响应”

注：phoneClientId 见“四、14.WiFi 模组接收来自手机对 WiFi 模组的命令”

14. WiFi 模组通过云端传送命令到所有与设备有关联的手机（WiFi 模组->云端）

pub{dev2app/<did>}: [Hi+P0]见“二、21. WiFi 模组向手机发送日志”

15. WiFi 模组收到固件升级通知 (OTA v4)（云端 -> WiFi 模组）

sub{ser2cli_res/<clientId>}: [Pc] protocolVer(4B)=0x00000003, cmd(2B)=0x020E, firmwareType(int, 1B)=1(wifi)|2(mcu), targetFirmwareId(int, 4B),

downloadUrlLen(2B), downloadUrl(str, max 512B)

16. WiFi 模组请求返在线手机数量（WiFi 模组 -> 云端）

pub{cli2ser_req}: [Pc] protocolVer(4B)=0x00000003, cmd(2B)=0x020F

注：云端通过“WiFi 模组收到在线手机数量变化通知”指令返回结果。

17. WiFi 模组收到在线手机数量变化通知（云端 -> WiFi 模组）

sub{ser2cli_res/<clientId>}: [Pc] protocolVer(4B)=0x00000003, cmd(2B)=0x0210, onlineDeviceCount(2B)

注：onlineDeviceCount 是指与设备有绑定关系的在线设备数量。

五、WiFi 模组与设备主控 MCU 的通讯协议

1. WiFi 模组向设备主控 MCU 转发直接来自手机的命令（WiFi 模组 ->设备主控 MCU）

local: [Hd+P0] protocolVer(4B)=0x00000003, len(int, 2B), cmd(2B)=0x0301, socketId(4B)=与手机连接的 SocketId, P0（见“六、手机与设备主控 MCU 的通讯协议（P0）”）

2. WiFi 模组向设备主控 MCU 转发间接通过云端来自手机的命令（WiFi 模组 ->设备主控 MCU）

local: [Hd+P0]protocolVer(4B)=0x00000003, len(int, 2B), cmd(2B)=0x0302, phoneClientIdLen(2B), phoneClientId(str, max 23B), P0（见“六、手机与设备主控 MCU 的通讯协议（P0）”）

注：phoneClientId 从 MQTT Topic 中获取

3. 设备主控 MCU 请求 WiFi 模组直接向手机转发命令（设备主控 MCU ->WiFi 模组）

local: [Hd+P0]protocolVer(4B)=0x00000003, len(int, 2B), cmd(2B)=0x0303, socketId(4B)=与手机连接的 SocketId, P0（见“六、手机与设备主控 MCU 的通讯协议

（P0）”）

4. 设备主控 MCU 请求 WiFi 模组间接通过云端向手机转发命令（设备主控 MCU ->WiFi 模组）

local:[Hd+P0]protocolVer(4B)=0x00000003, len(int, 2B), cmd(2B)=0x0304, phoneClientIdLen(2B), phoneClientId(str, max 23B), P0（见“六、手机与设备主控 MCU 的通讯协议（P0）”）

注：phoneClientId 从 MQTT Topic 中获取

5. 设备主控 MCU 请求 WiFi 模组向所有直接连接 WiFi 模组的手机及云端转发命令（设备主控 MCU ->WiFi 模组）

local:[Hd+P0]protocolVer(4B)=0x00000003, len(int, 2B), cmd(2B)=0x0305, P0（见“六、手机与设备主控 MCU 的通讯协议（P0）”）

6. WiFi 模组向设备主控 MCU 请求设备信息（WiFi 模组 ->设备主控 MCU）

local:[Hd+Pd] protocolVer(4B)=0x00000003, len(int, 2B), cmd(2B)=0x0306

7. 设备主控 MCU 向 WiFi 模组回复设备信息（设备主控 MCU ->WiFi 模组）

local:[Hd+Pd] protocolVer(4B)=0x00000003, len(int, 2B), cmd(2B)=0x0307, productKeyLen(2B), productKey(str, max 32B)
), busiProtocolVerLen(2B), busiProtocolVer(int)

注：参数中的“busiProtocolVer”是对设备所支持的业务逻辑协议的版本号。

8. 设备主控 MCU 请求 WiFi 模组进入 onboarding 模式（设备主控 MCU ->WiFi 模组）

local:[Hd+Pd] protocolVer(4B)=0x00000003, len(int, 2B), cmd(2B)=0x0308

9. WiFi 模组通知设备主控 MCUWiFi 模组 onboarding 的结果（WiFi 模组 ->设备主控 MCU）

local:[Hd+Pd] protocolVer(4B)=0x00000003, len(int, 2B), cmd(2B)=0x0309, result(1B)=0(success) | 1(fail)

10. 设备主控 MCU 请求 WiFi 模组进入可配对模式（设备主控 MCU ->WiFi 模组）

local:[Hd+Pd] protocolVer(4B)=0x00000003, len(int, 2B), cmd(2B)=0x030A

11. 设备主控 MCU 请求重置 WiFi 模组（设备主控 MCU ->WiFi 模组）

local:[Hd+Pd] protocolVer(4B)=0x00000003, len(int, 2B), cmd(2B)=0x030B

注：重置 WiFi 模组包指清空 WiFi 模组中保存的 did, passcode, 无线路由 ssid 及其密码

12. WiFi 模组状态变化后，向设备主控 MCU 发送最新状态（WiFi 模组 ->设备主控 MCU）

local:[Hd+Pd] protocolVer(4B)=0x00000003, len(int, 2B), cmd(2B)=0x030C, wifiStatus(2B)

注：

1. wifiStatus 用两个字节描述，从右向左依次是第 0 位，第 1 位，..... 第 15 位；

    第 0 位：是否开启 softAP 模式，0：关闭，1：开启；

    第 1 位：是否开启 station 模式，0：关闭，1：开启；

    第 2 位：是否开启 onboarding 模式，0：关闭，1：开启（可以通过第 0 位的 softAP 模式有没有开启来判断具体的 onboarding 是什么，当 softAP 此时为开启时，为 softAP/web onboarding 方式；当 softAP 此时为关闭时，为 AirLink onboarding 方式）；

    第 3 位：是否开启 binding 模式，0：关闭，1：开启；

    第 4 位：WiFi 模组是否成功连接路由器，0：未连接，1：连接；

    第 5 位：WiFi 模组是否成功连接云端(connect 成功)，0：未连接，1：连接；

    第 6 和第 7 位：MQTT 状态枚举，00：start，01：registered，10：login，11：running；

    第 8, 9, 10 位：仅当 WiFi 模组已成功连接路由器（请看上第 4 位）时值才有效，三个位合起来表示一个整型值，值范围为 0~7，表示 WiFi 模组当前连接 AP 的信号强度（RSSI），0 为最低，7 为最高；

    第 11 位：是否有手机在线，0 为否，1 为是；

    第 12 位：是否处于产测模式，0 为否，1 为是；

    第 13-15 位：预留；

2. WiFi 模组状态发生变化后，或者重启后，均需要向设备主控制 MCU 汇报当前的状态，并且要得到 ACK（见指令五.13），否则 200 毫秒进行重发，最多重发 3 次。

3. 本指令在即使没有发生状态化也会定时每 10 分钟发一次

13. 设备主控 MCU 给 WiFi 模组返回对收到 WiFi 状态的 ACK（设备主控 MCU ->WiFi 模组）

local:[Hd+Pd] protocolVer(4B)=0x00000003, len(int, 2B), cmd(2B)=0x030D

注：设备主控制 MCU 收到 WiFi 状态后，要先回复 ACK，再进行处理。

14. WiFi 模组向设备主控 MCU 发送心跳（WiFi 模组 ->设备主控 MCU）

local:[Hd+Pd] protocolVer(4B)=0x00000003, len(int, 2B), cmd(2B)=0x030E

15. 设备主控 MCU 给 WiFi 模组返回心跳 ACK（设备主控 MCU ->WiFi 模组）

local:[Hd+Pd] protocolVer(4B)=0x00000003, len(int, 2B), cmd(2B)=0x030F

注：

1、心跳由 WiFi 模组发起，设备主控 MCU 收到后立即回复，设备主控 MCU 不主动发起心跳；

2、WiFi 模组和设备主控 MCU 都可以判断心跳超时，超时后重启 WiFi 模组程序，MCU 不重启。

16. 设备主控 MCU 请求 WiFi 模组进入产测模式（设备主控 MCU -> WiFi 模组）

local:[Hd+Pd] protocolVer(4B)=0x00000003, len(int, 2B), cmd(2B)=0x0310

17. WiFi 模组向设备主控 MCU 返回产测模式 ACK（WiFi 模组 ->设备主控 MCU）

local:[Hd+Pd] protocolVer(4B)=0x00000003, len(int, 2B), cmd(2B)=0x0311

六、手机与设备主控 MCU 的通讯协议 (P0)

本 P0 协议的内容可以由设备生产厂商根据产品特性定制，此为建议的基于实体属性读写的通用协议格式。

1. 手机请求读取设备的属性（手机 ->设备主控 MCU）

[P0] protocolVer(2B)=0x0001, len(int, 2B), flag(1B)=0x00, cmd(2B)=0x0001,elementCount(int, 2B), [{entityId(2B)=0x0000,attrId(2B)}...], checksum(2B)

注：elementCount(int, 2B)表示后接数组的数组元素个数，下同。

2. 设备主控 MCU 回复手机的读取请求（设备主控 MCU ->手机）

[P0] protocolVer(2B)=0x0001, len(int, 2B), flag(1B)=0x00, cmd(2B)=0x0002,elementCount(int, 2B), [{entityId(2B), attrId(2B),attrValLen(2B),attrVal}...], checksum(2B)

3. 手机更改设备的属性（或手机控制设备）- QoS=0（手机 ->设备主控 MCU）

[P0] protocolVer(2B)=0x0001, len(int, 2B), flag(1B)=0x00, cmd(2B)=0x0003,elementCount(int, 2B), [{entityId(2B), attrId(2B),attrValLen(2B),attrVal}...], checksum(2B)

4. 手机更改设备的属性（或手机控制设备）- QoS=1（手机 ->设备主控 MCU）

[P0] protocolVer(2B)=0x0001, len(int, 2B), flag(1B)=0x01, seq(4B), cmd(2B)=0x0003,elementCount(int, 2B), [{entityId(2B), attrId(2B),attrValLen(2B),attrVal}...], checksum(2B)

5. 设备主控 MCU 推送状态到手机 - QoS=0（状态推送/Push notification）（设备主控 MCU ->手机）

[P0] protocolVer(2B)=0x0001, len(int, 2B), flag(1B)=0x00, cmd(2B)=0x0004,elementCount(int, 2B), [{entityId(2B), attrId(2B),attrValLen(2B),attrVal}...], checksum(2B)

6. 设备主控 MCU 推送状态到手机 - QoS=1（设备报警/Alert）（设备主控 MCU ->手机）

[P0] protocolVer(2B)=0x0001, len(int, 2B), flag(1B)=0x01, seq(4B), cmd(2B)=0x0004,elementCount(int, 2B), [{entityId(2B), attrId(2B),attrValLen(2B),attrVal}...], checksum(2B)

7. 对 QoS=1 命令的确认（设备主控 MCU ->手机或手机 ->设备主控 MCU）

[P0] protocolVer(4B)=0x0001, len(int, 2B)=0x0006, cmd(2B)=0x0005, seq(4B)

七、附件

1. 《MQTT_V3.1_Protocol_Specific.pdf》