基于STM32节点和阿里云IoT平台的物联网应用开发 系列课程

第三章

基于STM32的节点设备接入阿里云IoT平台

life.augmented

阿里云

课程内容下载、观看

- 视频观看：AI电堂、阿里云大学IoT课堂
- 课件胶片下载：STMCU中文官网、阿里云大学IoT课堂
- 课件项目下载：STMCU中文官网、阿里云大学IoT课堂

STM32公众号

STM中文官网

电堂公众号

阿里云大学

第三章内容简介

- 第一节：基于STM32的节点端介绍
  - 硬件平台，软件开发环境
- 第二节：使用Paho MQTT客户端协议栈直连阿里云IoT平台
  - 适用于资源受限的节点设备
- 第三节：使用Linkkit C-SDK和TLS通过MQTT协议直连阿里云IoT平台
  - 适用于资源丰富的节点设备

STM32-阿里云IoT　联合课件开发

第三章 . 第二节

基于Paho.MQTT直连阿里云IoT平台

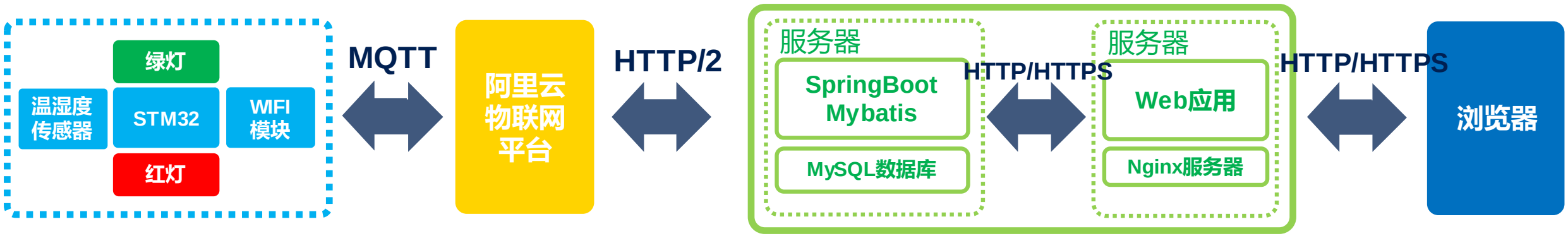
life.augmented

阿里云

第三章.第二节 内容简介

- 项目例程演示
  - 项目例程流程图
  - 演示视频
- MQTT协议介绍
  - 协议特性
  - 协议模型和报文格式
  - 报文使用序列
  - 阿里云IoT云平台侧MQTT协议实现
- 项目例程介绍
  - 软件包和项目结构
  - 使用CubeMX生成系统初始化框架和代码
  - 网络通信的管理（网络通信抽象层和wifi驱动）
  - 使用Paho MQTT客户端协议栈连接阿里云IoT平台
  - 例程参数的存储及Sensor数据的读取

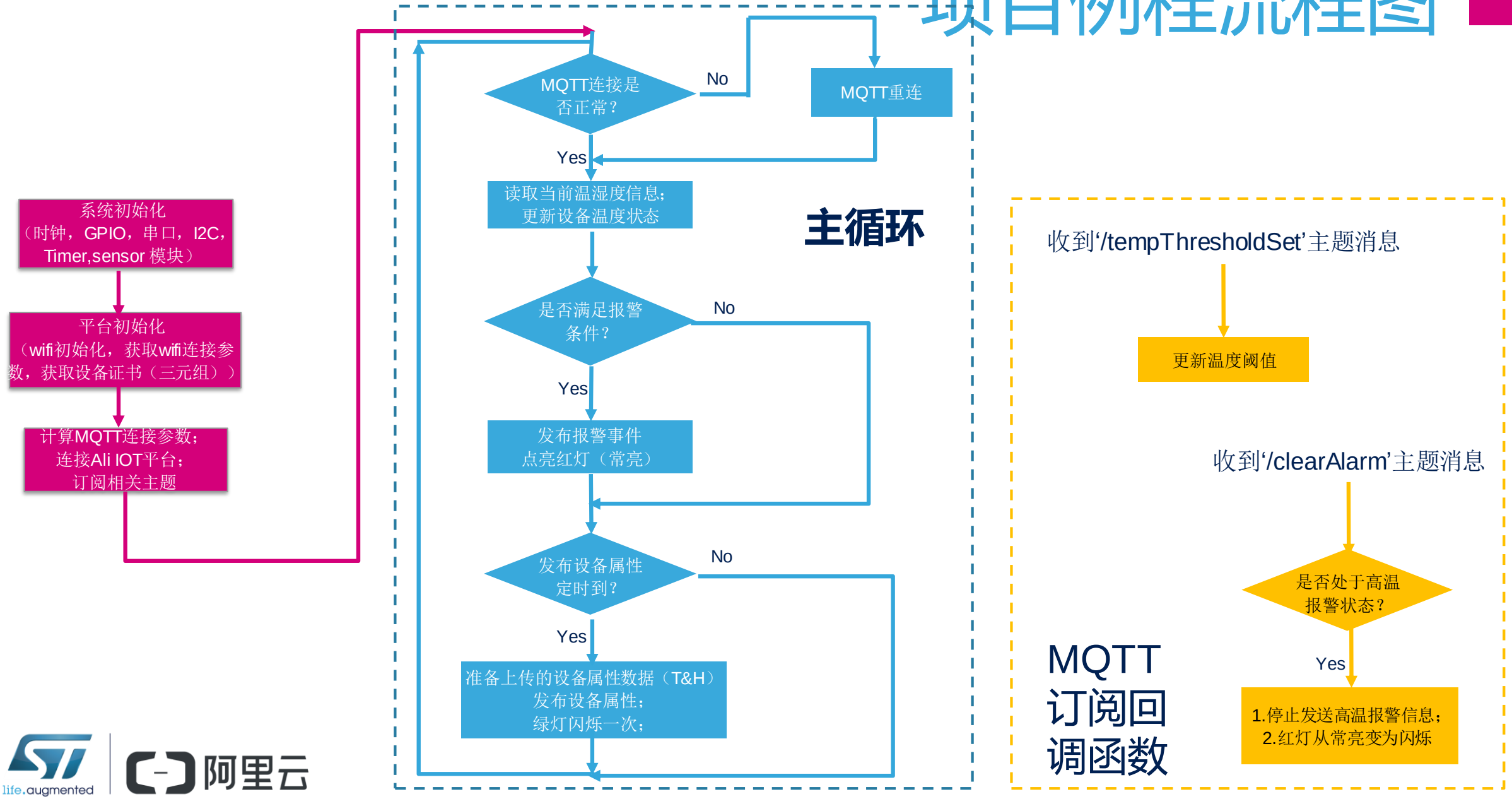
课程场景



- 每5秒上报温湿度值，闪烁绿灯
- 温度超【阈值】亮红灯，并在每10秒向用户服务器报警，直到温度恢复【阈值】以下或者收到警报解除消息
- 收到警报解除信息后红灯闪烁
- 温度恢复到【阈值】以下灭红灯
- 湿度值被阿里云IoT转发到用户服务器，进行数据库存储，同时在web端显示近期温湿度数据曲线
- 报警消息被阿里云IoT转发到用户服务器，在web端显示
- 用户通过web端页面解除报警
- 用户通过web端页面设置【阈值】参数

life.augmented | 阿里云

项目例程流程图



life.augmented

阿里云

运行效果 – 节点端串口打印


```
*
***                  STM32 based AliIoT Client Demo                  **
*
***                           Without TLS                            **
*
***              FW version 0.3.0 - Mar 22 2019, 15:21:40            **
*
*******************************************************************************
*

Application parameter init. Send alarm when temprature>= 28 degrees Celsi
us,turn off Red LED when temprature<27 degrees Celsius

*** WIFI connection ***

Push the User button (Blue) within the next 5 seconds if you want to upda
te the WiFi network configuration.

Your WiFi parameters need to be entered to proceed.

Enter SSID: [redacted]
You have entered [redacted] as the ssid.

Enter Security Mode (0 - Open, 1 - WEP, 2 - WPA, 3 - WPA2):3
You have entered 3 as the security mode.

Enter password: [redacted]

Initializing the WiFi module
firmware version is : basic_AT_v2.1.2
OK

> WiFi module MAC address is: B0:F8:93:17:BC:E2


Connecting to AP: [redacted]  Attempt 1/3 ...
Connected to AP [redacted]
Retrieving the IP address.
IP address: 192.168.43.203
```

若要重新配置wifi热点，需要5秒内按下user键

输入热点名称和密码

热点连接成功

```
Push the User button (Blue) within the next 5 seconds if you want to upda
te the device security parameters or credentials.


Start to enter Ali Device Parameters: Region ID,Product Key,Device Name a
nd Device Secret

Enter Region ID: (example: cn-shanghai)
cn-shanghai
read: --->
cn-shanghai
<---

Enter Product Key: (example: a1b05Uexxxx)
a1jAqFa0Zng
read: --->
a1jAqFa0Zng
<---

Enter device name: (example: mydevicename)
smartthermometer
read: --->
smartthermometer
<---

Enter device secret: (example: 7o7GJ3odUE7pPnie07dzxxxxxxxxxxx)
0TuurK4rWK3x[redacted]
read: --->
0TuurK4rWK3x[redacted]
<---
MQTT server address is :a1jAqFa0Zng.iot-as-mqtt.cn-shanghai.aliyuncs.com

*** Start connecting to MQTT server ***
Server address: a1jAqFa0Zng.iot-as-mqtt.cn-shanghai.aliyuncs.com : 1883

TCP Connection in progress:   Attempt 1/3 ...

connected to server
```

若要连接到自己的iot设备，需要5秒内按下user键

输入regionID，ProductKey device name，device secret

计算MQTT服务器地址

region ID字符串查询页面

life.augmented

阿里云

运行效果 – 节点端串口打印

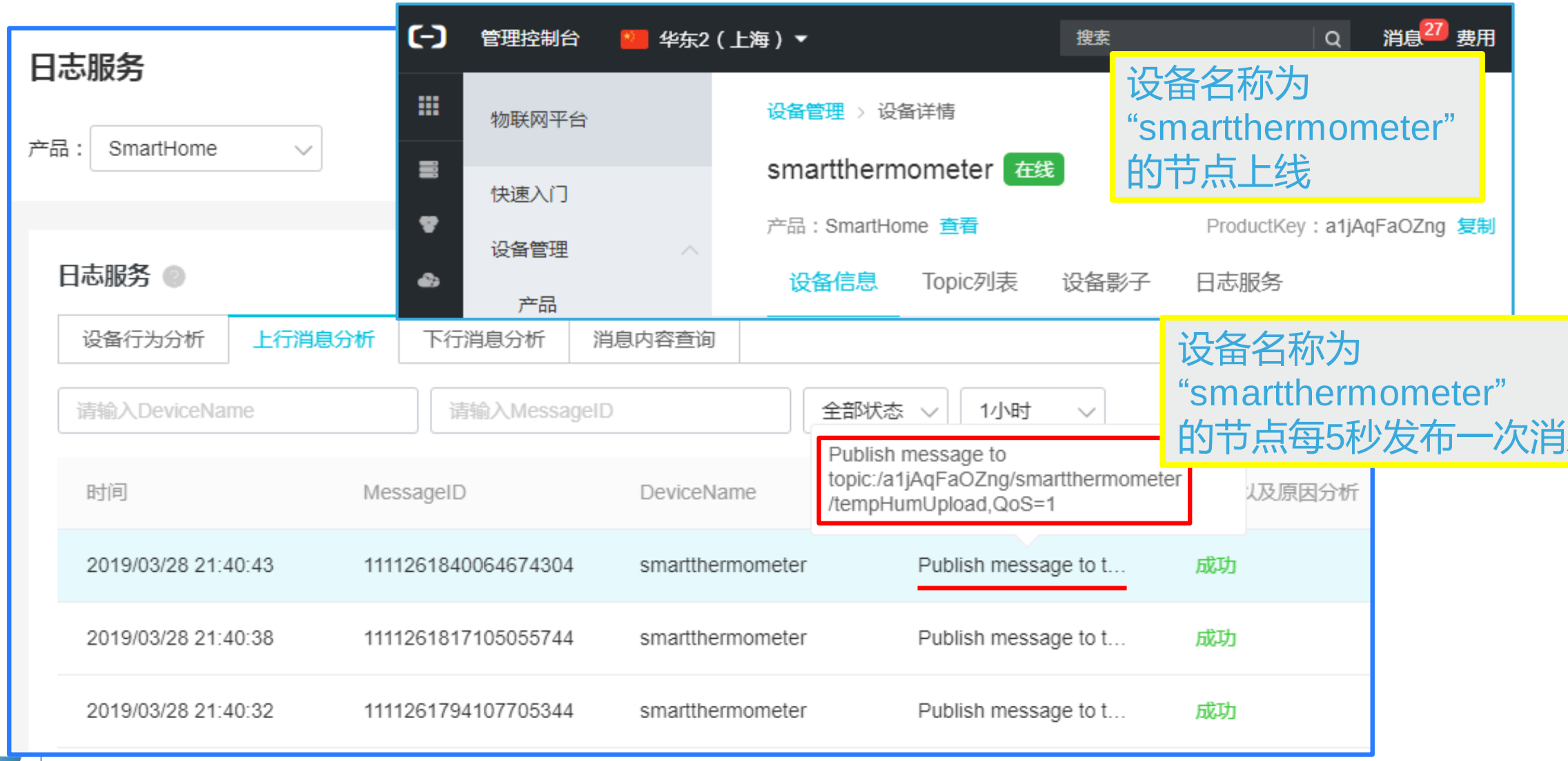

```
Processing MQTT Connection

MQTT Client ID is :b0f89317bce2|securemode=3,signmethod=hmacsha1,timestamp=8237098|
MQTT Username is :smartthermometer&a1jAqFaOZng
content:clientIdb0f89317bce2deviceNamesmartthermometerproductKeya1jAqFaOZngtimestamp8237098
key: 0TuurK4rWK3x2[redacted]
MQTT password generated successfully:809cfd7bfc1fa018[redacted]
MQTT Connection Attempt 1/3 ...
subscribe to topic: /a1jAqFaOZng/smartthermometer/tempThresholdSet
subscribe to topic: /a1jAqFaOZng/smartthermometer/clearAlarm
publish device status successfully : temprature = 25, humidity = 36

publish device status successfully : temprature = 24, humidity = 36

publish device status successfully : temprature = 24, humidity = 36

publish device status successfully : temprature = 24, humidity = 36

publish device status successfully : temprature = 24, humidity = 36

publish device status successfully : temprature = 24, humidity = 36

publish device status successfully : temprature = 24, humidity = 36

publish device status successfully : temprature = 24, humidity = 36

publish device status successfully : temprature = 24, humidity = 36

publish device status successfully : temprature = 24, humidity = 36

publish device status successfully : temprature = 24, humidity = 36

publish device status successfully : temprature = 24, humidity = 36
```

计算阿里云IoT平台MQTT连接参数：
MQTT Client ID
MQTT Username
MQTT password

订阅两个主题：
1.温度报警门限值设置
2.清除报警

每5秒上报节点温湿度值

运行效果 – 阿里云IoT平台侧



life.augmented

阿里云

运行效果 – 应用服务器侧

设备上线，发布消息

运行效果 – 应用服务器侧

用户修改温度报警阈值

1. 收到修改“温度报警阈值”→ 20，并下发

life.augmented

阿里云

运行效果 –节点端串口打印

温度阈值被改变，触发节点端报警行为

```
publish device status successfully : temprature = 24, humidity = 37
publish device status successfully : temprature = 24, humidity = 37
publish device status successfully : temprature = 24, humidity = 37
publish device status successfully : temprature = 24, humidity = 37

[D]. MQTT payload received is:

received Temprature threshold:20
publish temperature alarm successfully : temprature = 24
publish device status successfully : temprature = 24, humidity = 37

publish temperature alarm successfully : temprature = 24
publish device status successfully : temprature = 24, humidity = 37
publish device status successfully : temprature = 24, humidity = 37

publish temperature alarm successfully : temprature = 24
publish device status successfully : temprature = 24, humidity = 37
publish device status successfully : temprature = 24, humidity = 37

[D]. MQTT payload received is:

receive ClearAlarm message
publish device status successfully : temprature = 24, humidity = 37
publish device status successfully : temprature = 24, humidity = 37
publish device status successfully : temprature = 24, humidity = 37
publish device status successfully : temprature = 24, humidity = 37
```

收到订阅的主题的消息：温度阈值 → 20

节点设备发送“报警”事件给IoT平台

收到订阅的主题的消息：清除报警

节点设备停止发送“报警”事件给IoT平台

内容简介

- Demo演示
  - Demo流程图
  - 演示视频
- MQTT协议介绍
  - 协议特性
  - 协议模型和报文格式
  - 报文使用序列
  - 阿里云IoT平台侧MQTT协议实现
- 项目例程绍
  - 软件包和项目结构
  - 使用CubeMX生成系统初始化框架和代码
  - 网络通信的管理（网络通信抽象层和wifi驱动）
  - 使用Paho MQTT客户端协议栈连接阿里云IoT平台
  - 例程参数的存储及Sensor数据的读取

MQTT协议介绍

- MQTT (Message Queuing Telemetry Transport) 消息队列遥测传输
- 协议关键特性
  - 轻量级协议，开销小
  - 异步通信模式，解耦通信双方
  - 基于TCP

IBM

EUROTECH

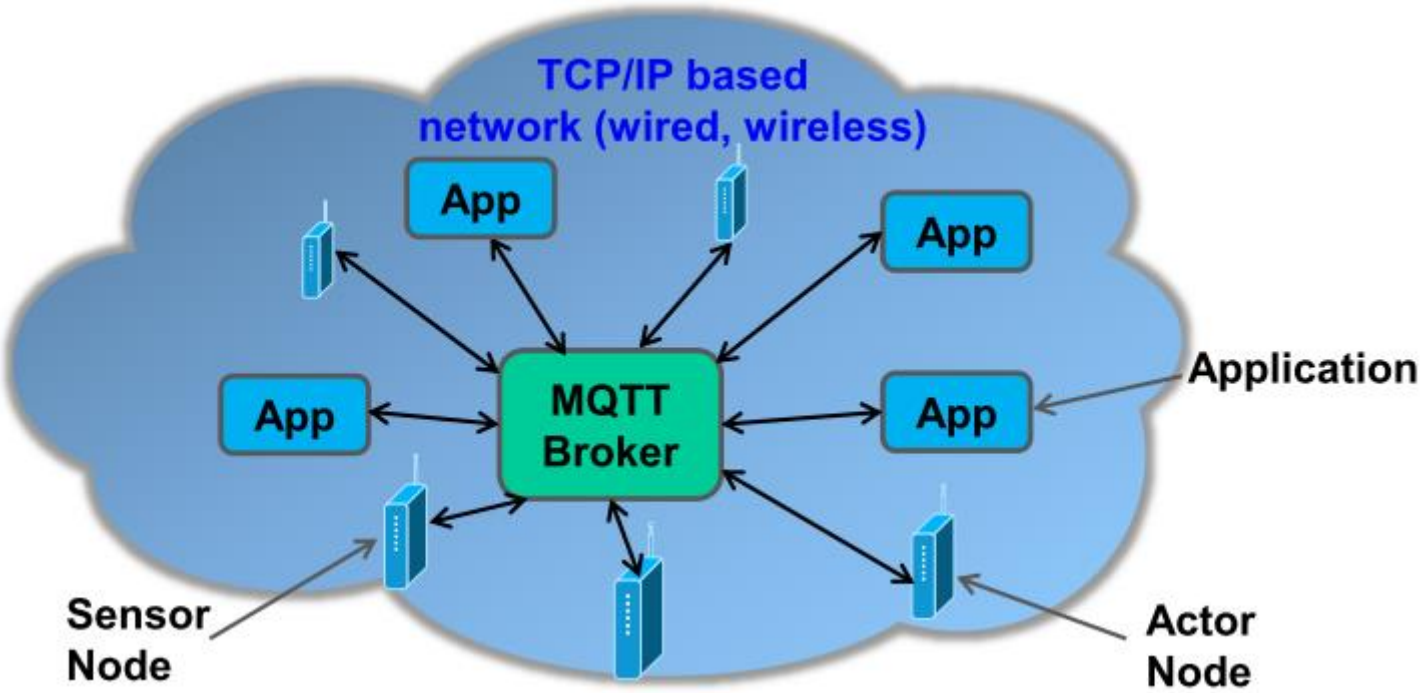
MQTT V3.1

MQTT V3.1.1

OASIS

life.augmented

阿里云



MQTT协议的通信模型

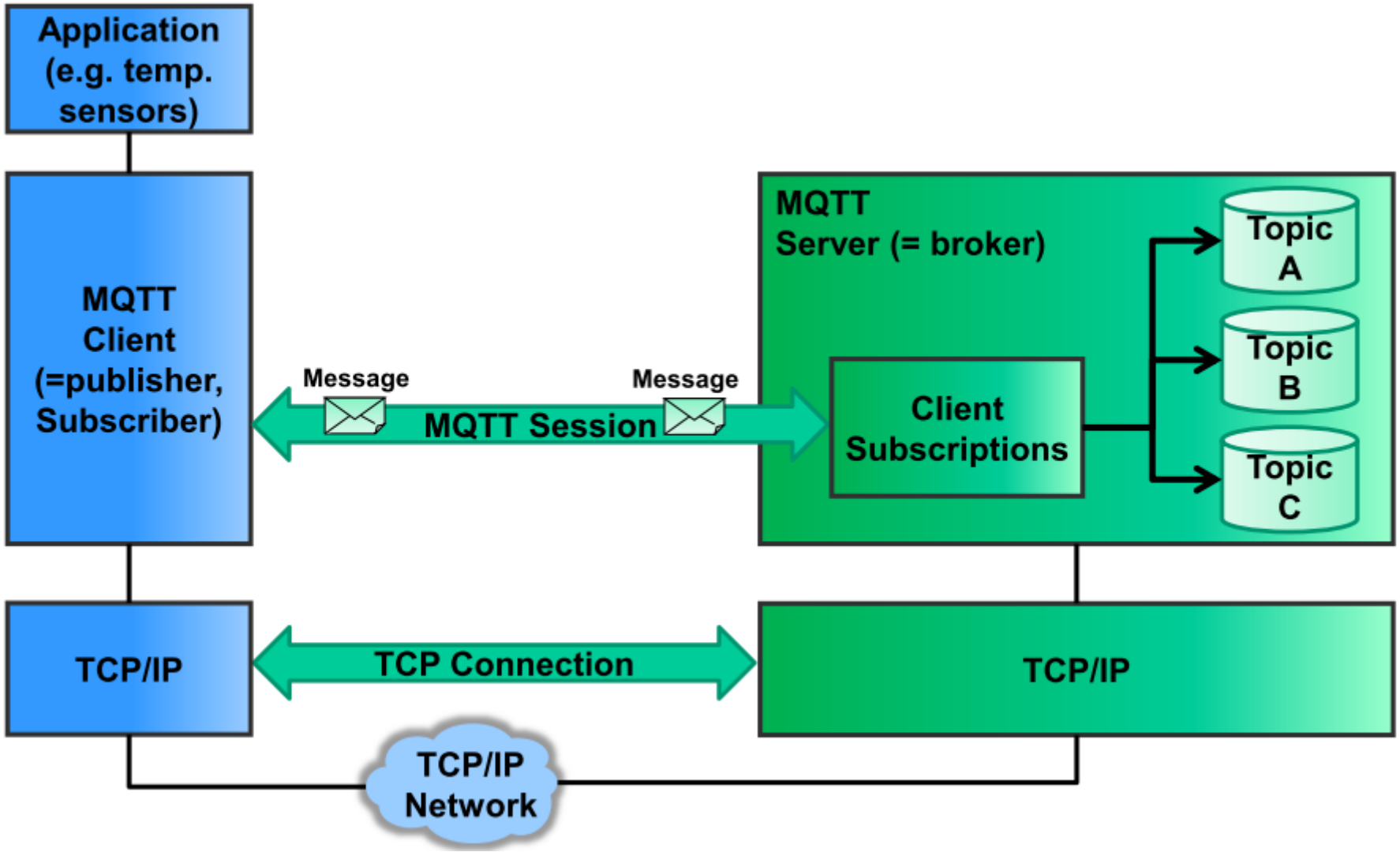
协议关键字

客户端(client)
服务器端/代理(server/broker)

会话(session)

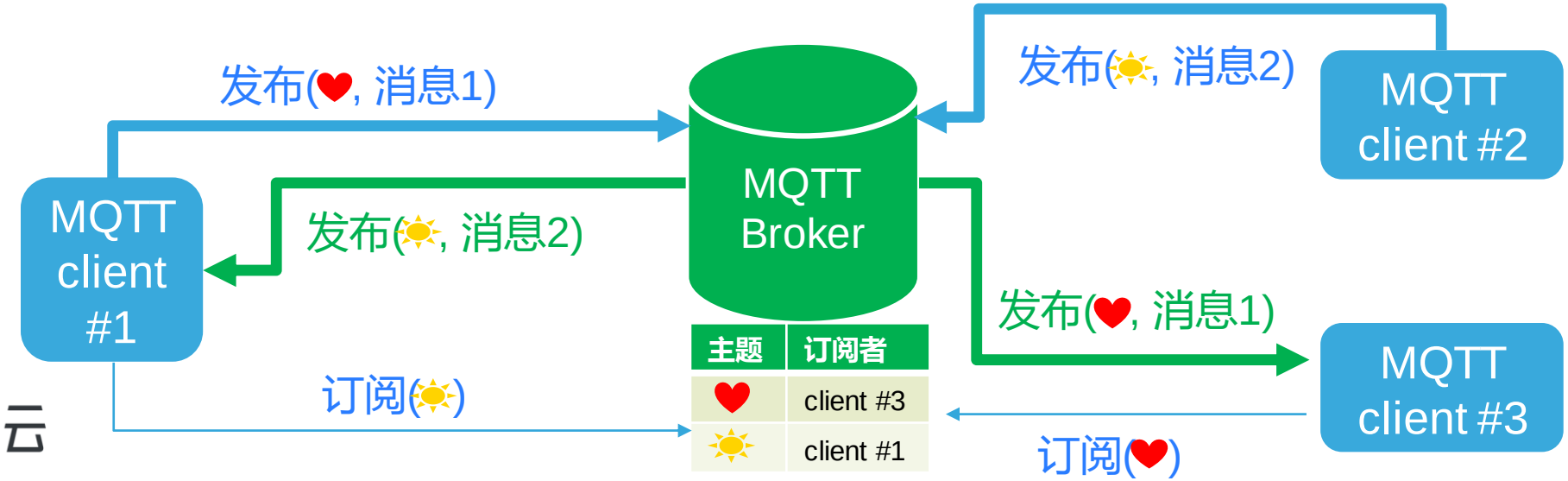
消息(message)
主题(topic)

订阅(subscribe)
发布(publish)



MQTT协议的主题和消息

- 主题具有层级属性，支持通配符

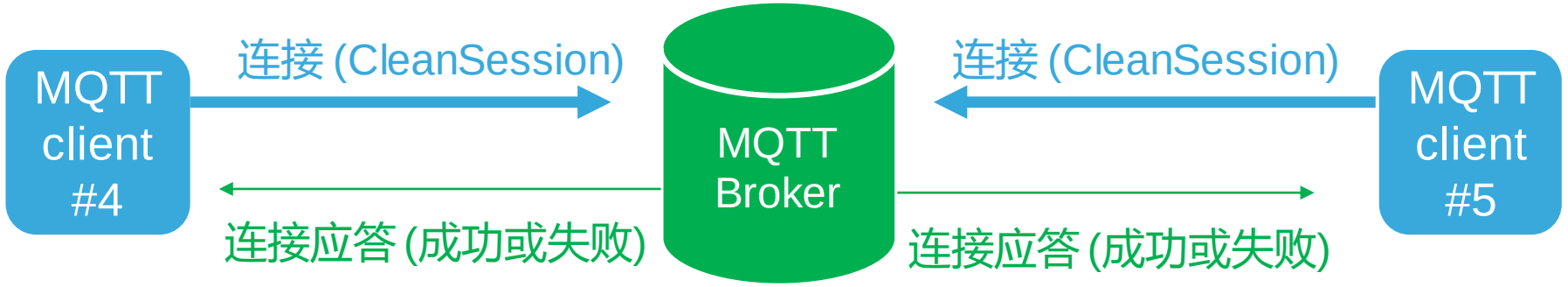


life.augmented

阿里云

MQTT协议的连接和会话

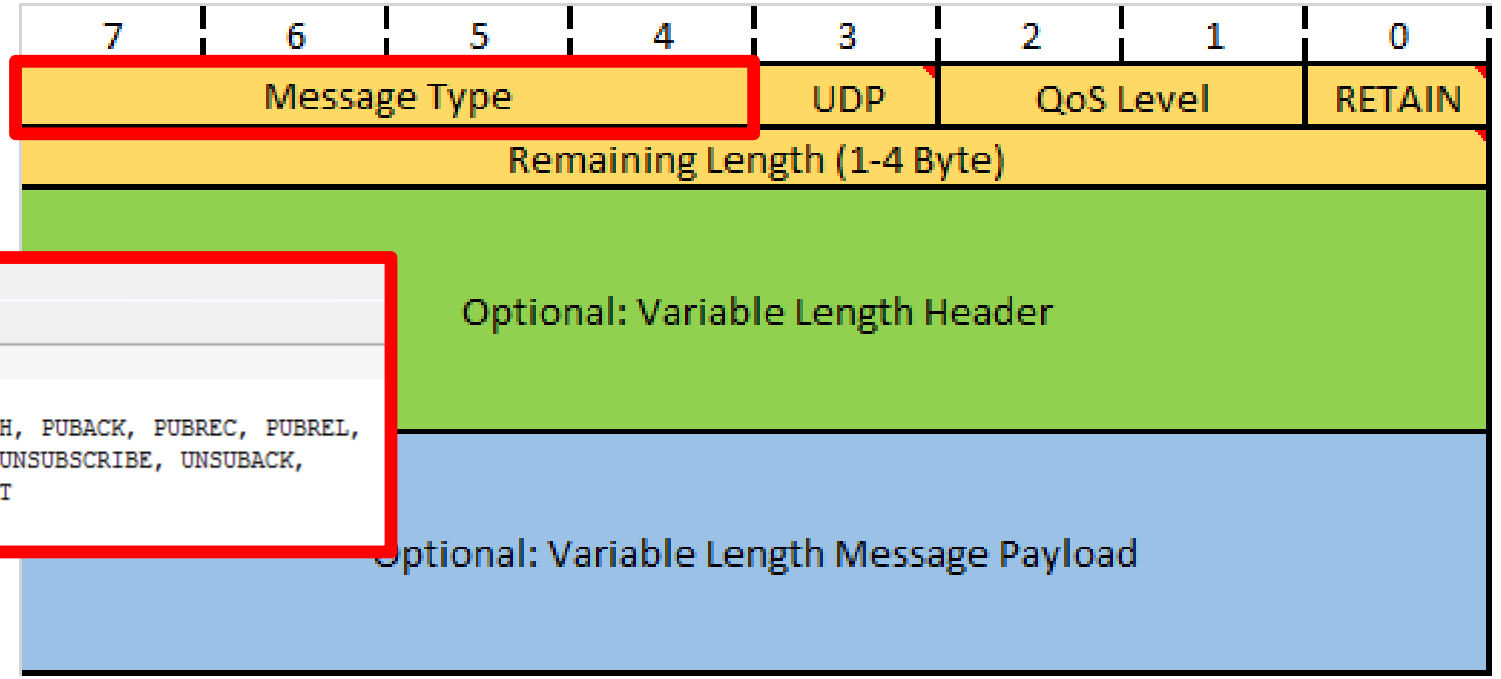
- 连接由客户端发起，会建立一个会话，把客户端附着到服务器上
- 服务器根据连接参数(ClientID, 用户名, 密码)对客户端进行鉴权和授权
- 连接参数(CleanSession)决定此次会话是否是持久会话(Persistent Session)



职责	依据的参数
鉴权	用户名、密码
授权	客户端ID
监测心跳	心跳间隔
遗嘱事宜	Will flag/主题/消息/…
会话保持	CleanSession

MQTT报文格式

- 固定报头，2~5字节，所有报文都包含
- 可变报头，长度不固定，部分报文才包含
- 有效负载，长度不固定，部分报文才包含

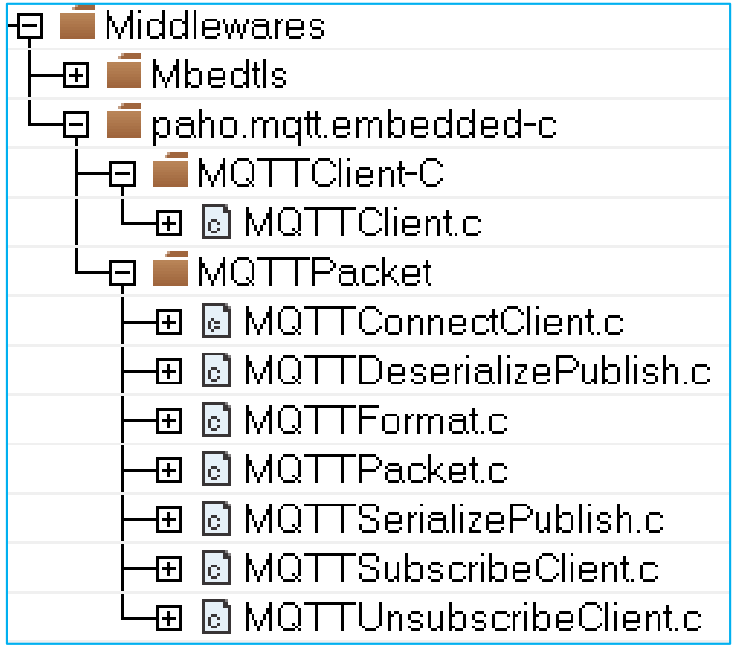


MQTTPacket.h

```
43  enum msgTypes
44  {
45          CONNECT = 1, CONNACK, PUBLISH, PUBACK, PUBREC, PUBREL,
46          PUBCOMP, SUBSCRIBE, SUBACK, UNSUBSCRIBE, UNSUBACK,
47          PINGREQ, PINGRESP, DISCONNECT
48  };
```

报文类型汇总

- 14种报文类型



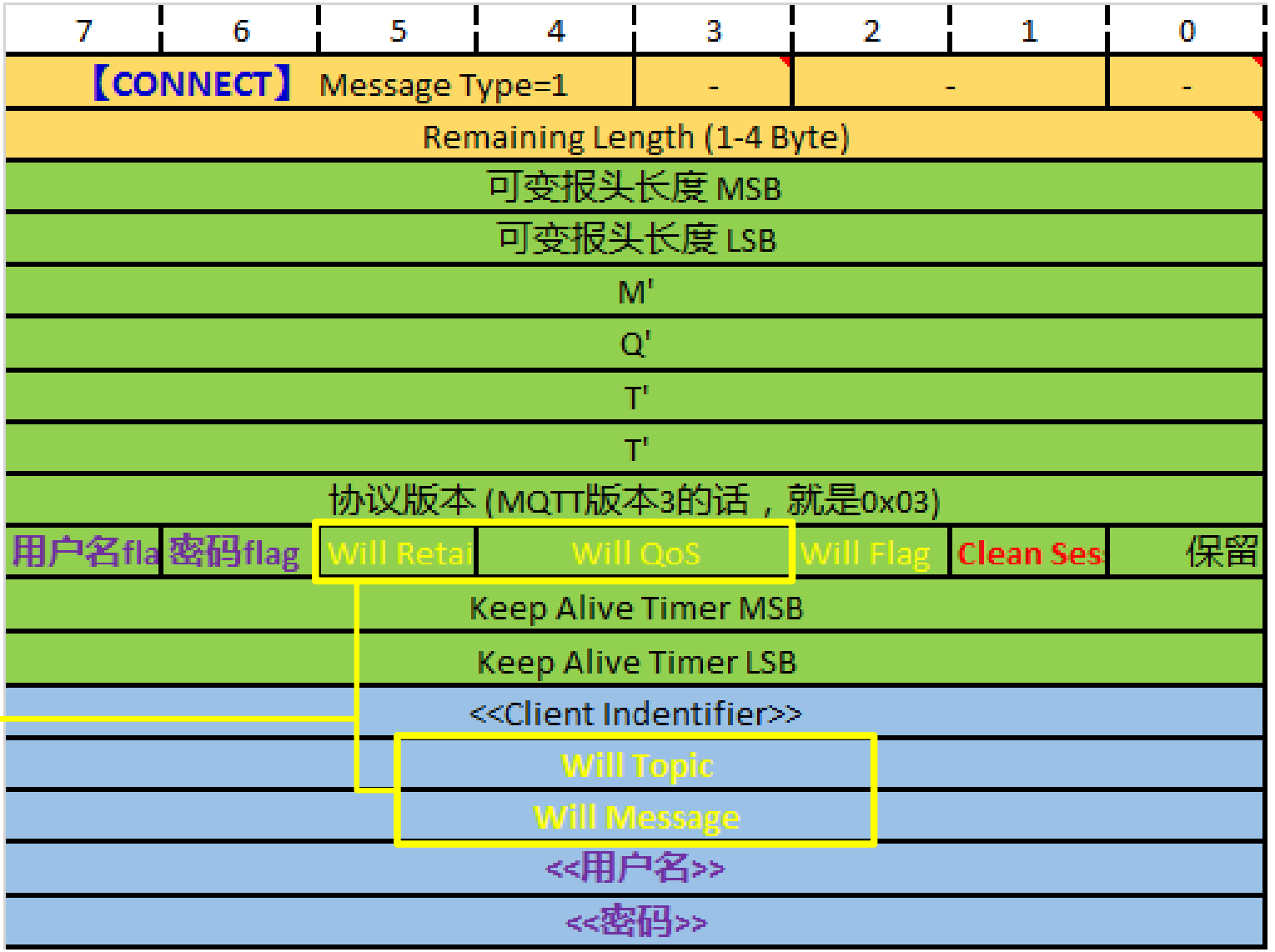
	Message Type value	Direction		Part 1			Part 2	Part 3
				Flag in fix-header			variable-header	payload
		C-->S	S-->C	UDP	QoS level	RETAIN	Pkt ID	
reserved	0							
CONNECT	1							
CONNACK	2							
PUBLISH	3							
PUBACK	4							
PUBREC	5				2			
PUBREL	6				2			
PUBCOMP	7				2			
SUBSCRIBE	8							
SUBACK	9							
UNSUBSCRIBE	10							
UNSUBACK	11							
PINGREQ	12							
PINGRESP	13							
DISCONNECT	14							
reserved	15							

连接报文：CONNECT

- MQTTConnect()

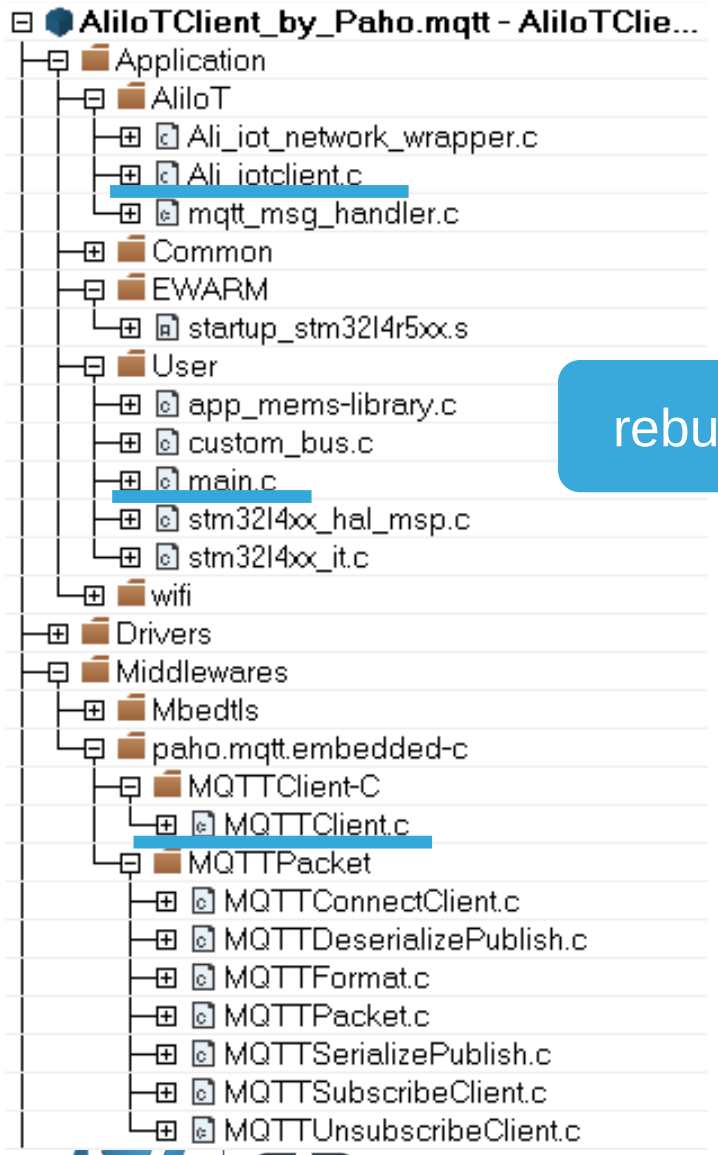
```
int MQTTConnect(MQTTClient* c, MQTTPacket_connectData* options)
{
    MQTTConnackData data;
    return MQTTConnectWithResults(c, options, &data);
}
```

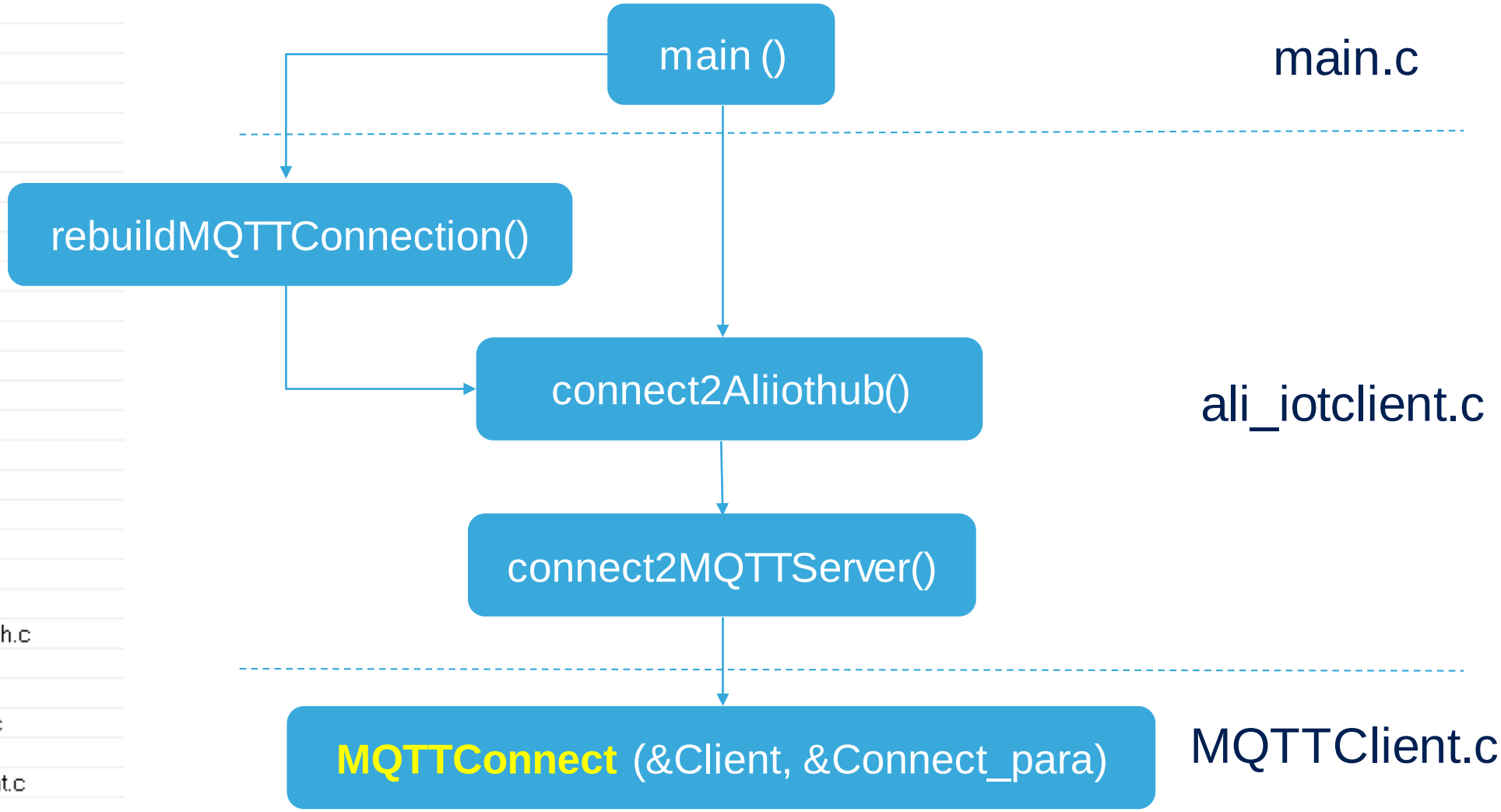
```
typedef struct
{
        /** The eyecatcher for this structure.  must be MQTC. */
        char struct_id[4];
        /** The version number of this structure.  Must be 0 */
        int struct_version;
        /** Version of MQTT to be used.  3 = 3.1 4 = 3.1.1
          */
        unsigned char MQTTVersion;
        MQTTString clientID;
        unsigned short keepAliveInterval;
        unsigned char cleansession;
        unsigned char willFlag;
        MQTTPacket_willOptions will;
        MQTTString username;
        MQTTString password;
} MQTTPacket_connectData;
```



life.augmented 阿里云

协议栈中的调用



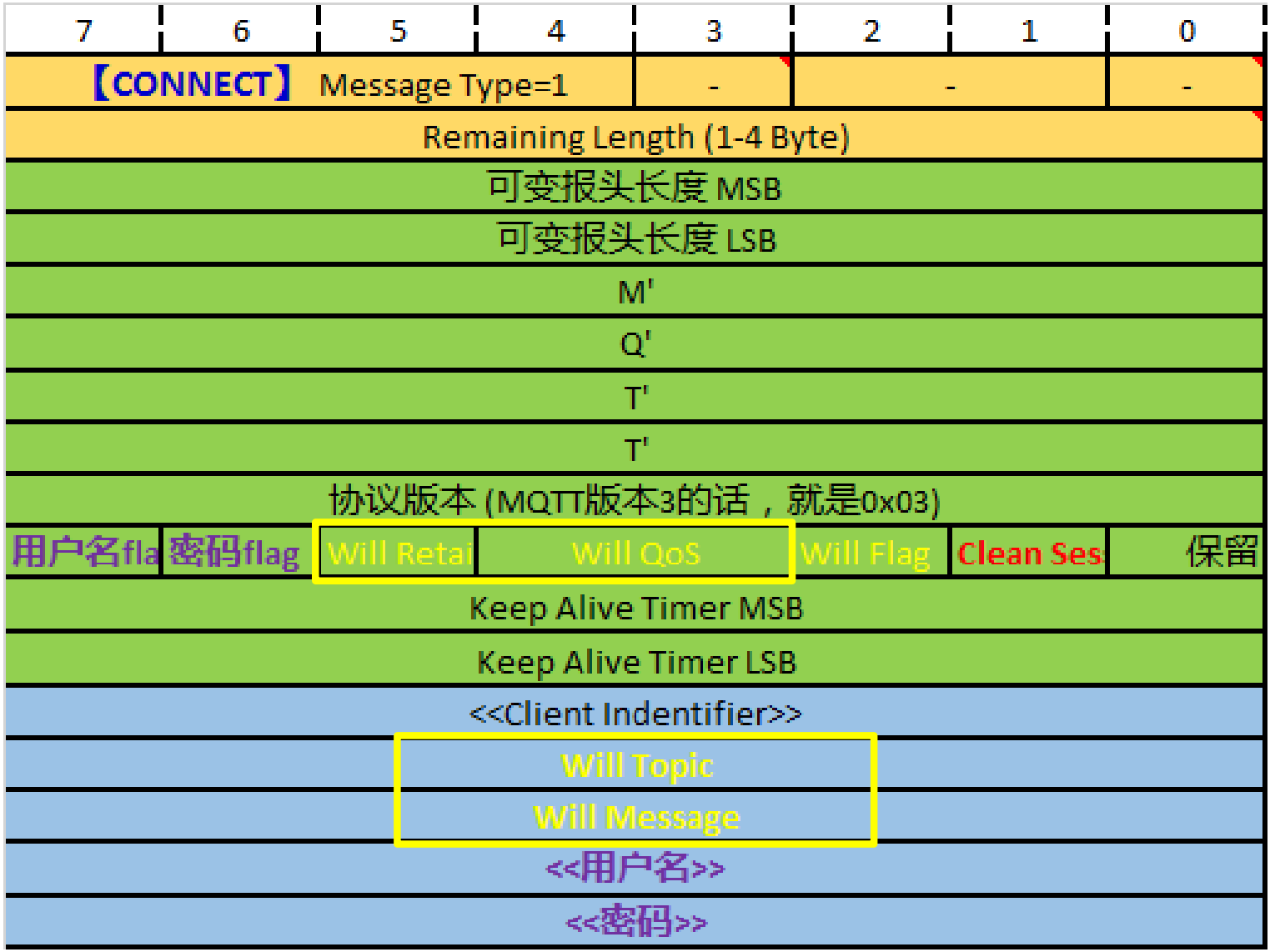


life.augmented

阿里云

遗嘱(Will)消息

- Will Flag
- Will Retain
- Will QoS
- Will Topic
- Will Message



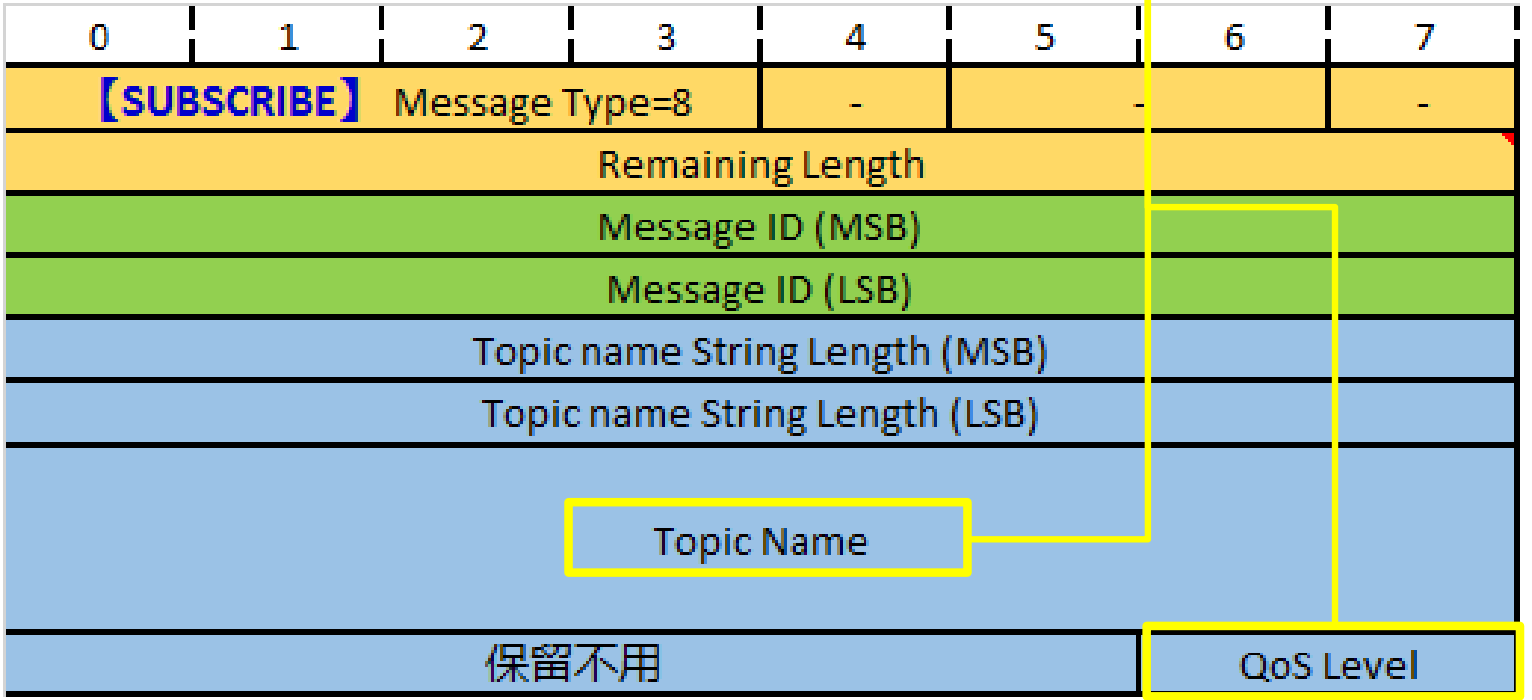
订阅报文：SUBSCRIBE

- MQTTSubscribe()

MQTTSubscribe
(
&Client, threshold_topic,
QOS0,
Parameters_message_handler)

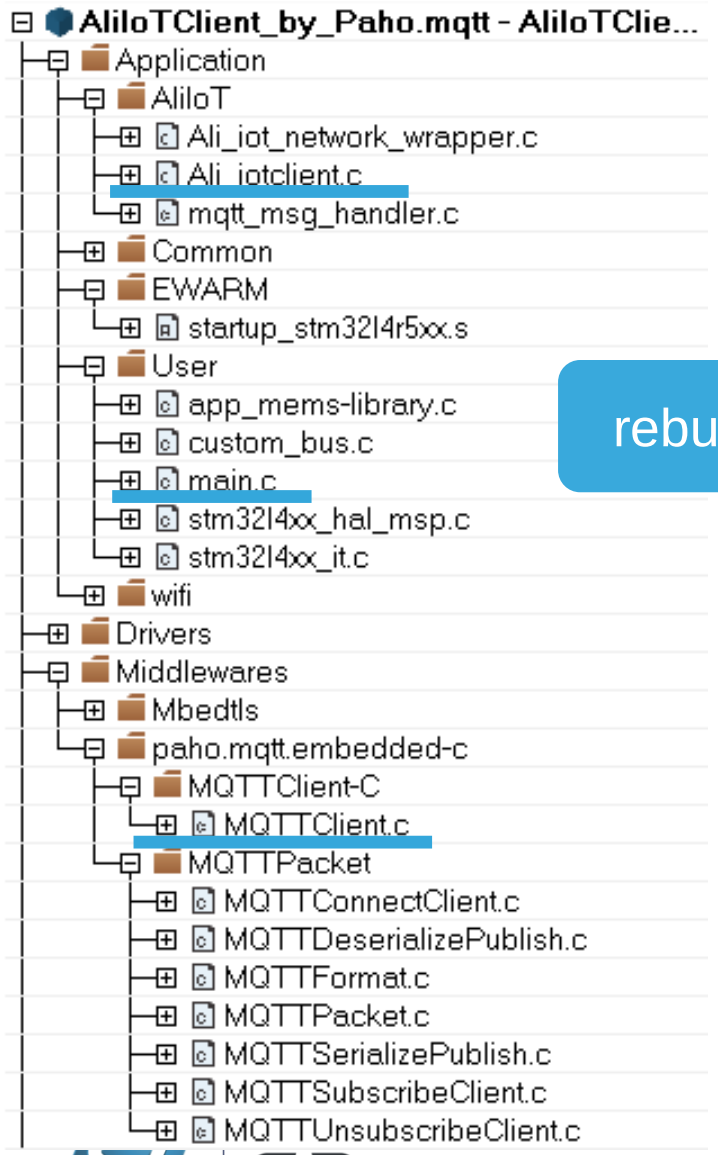
MQTTSubscribe
(
&Client,
clearAlarm_topic,
QOS0,
Service_message_handler)

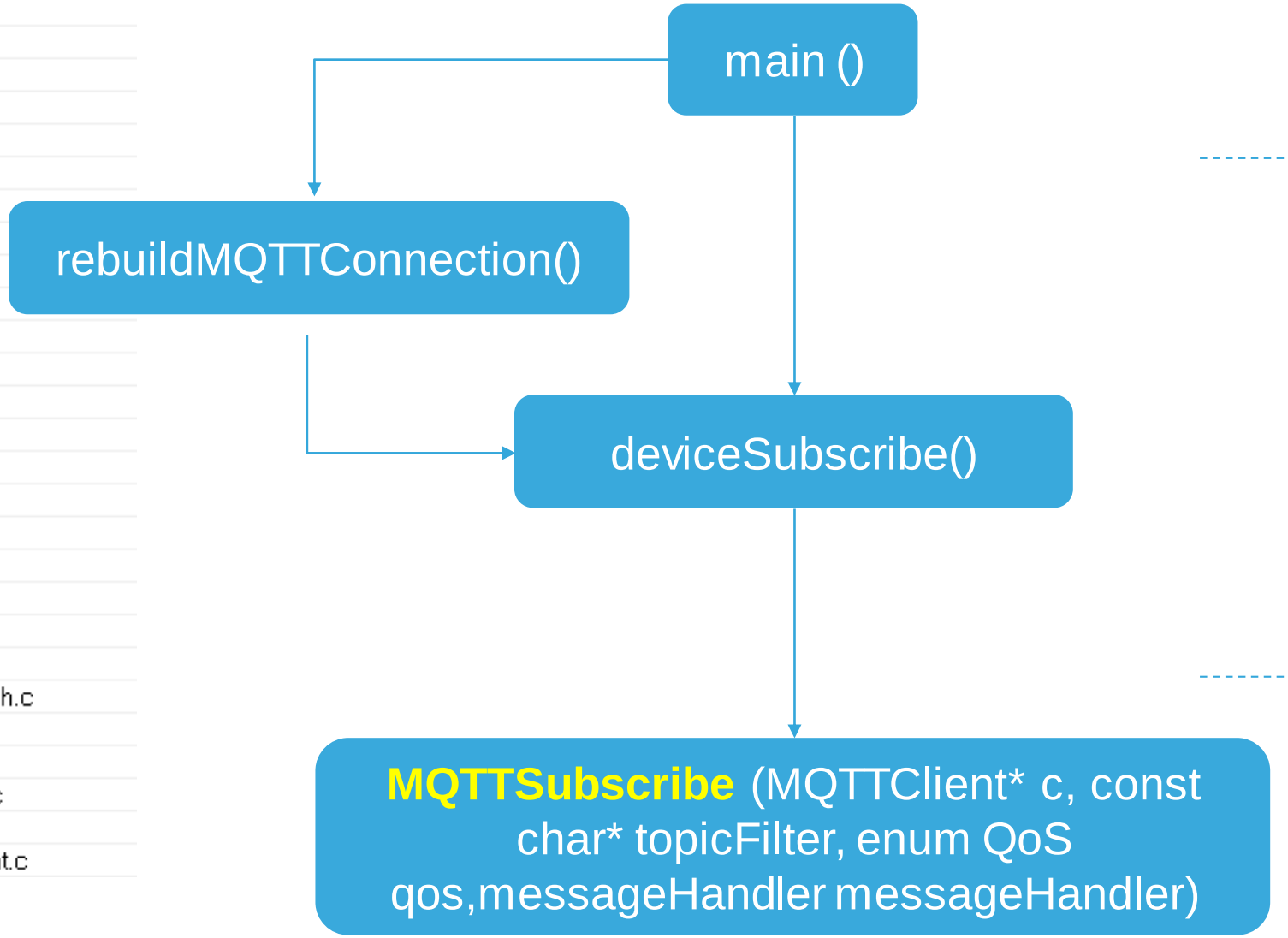
```
int MQTTSubscribe(MQTTClient* c, const char* topicFilter, enum QoS qos,
        messageHandler messageHandler)
{
    MQTTSubackData data;
    return MQTTSubscribeWithResults(c, topicFilter, qos, messageHandler, &data);
}
```



life.augmented | 阿里云

协议栈中的调用





main.c

ali_iotclient.c

MQTTClient.c

life.augmented

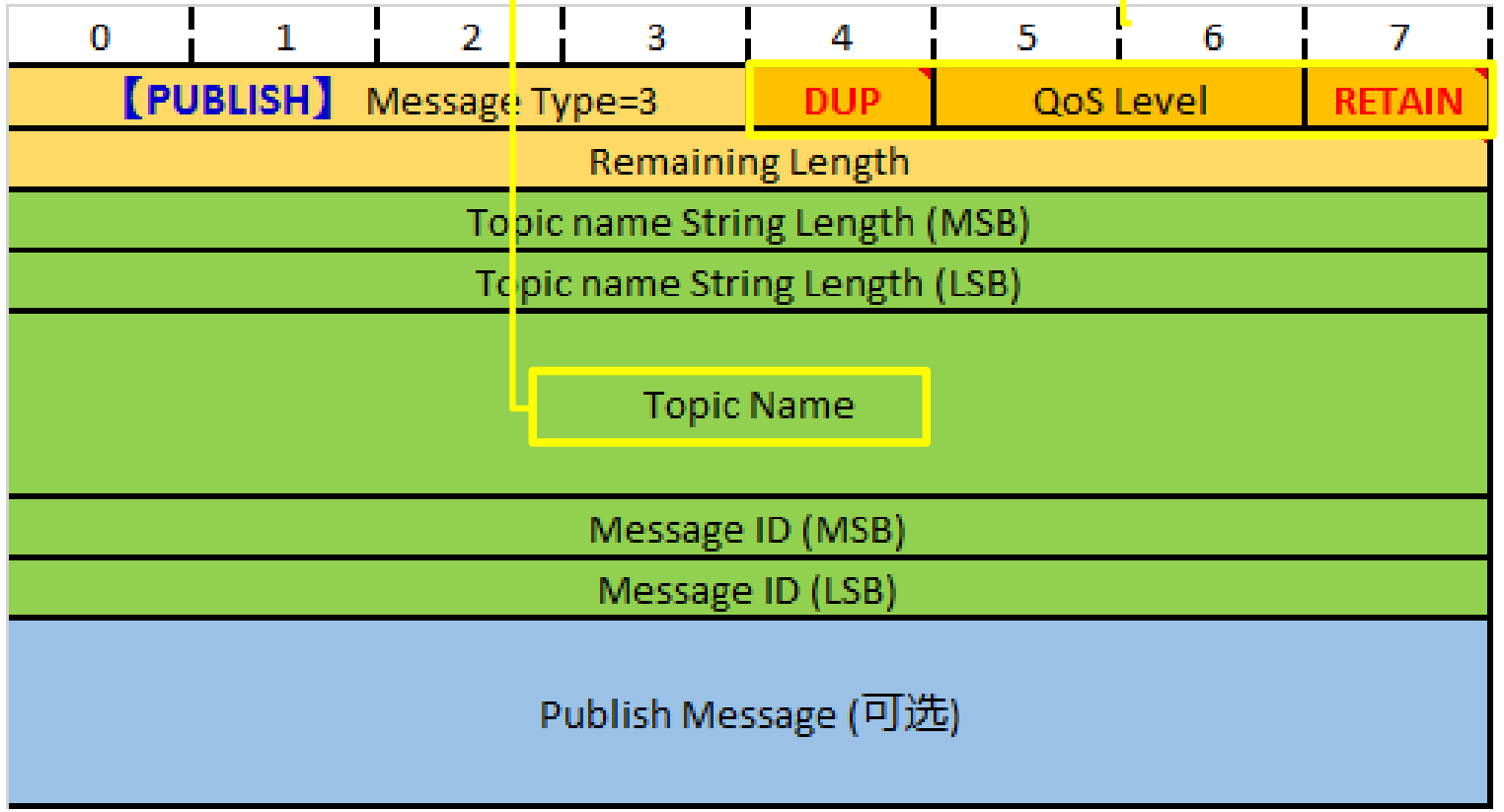
阿里云

发布报文：PUBLISH

- MQTTPublish()

```
typedef struct MQTTMessage
{
    enum QoS qos;
    unsigned char retained;
    unsigned char dup;
    unsigned short id;
    void *payload;
    size_t payloadlen;
} MQTTMessage;
```

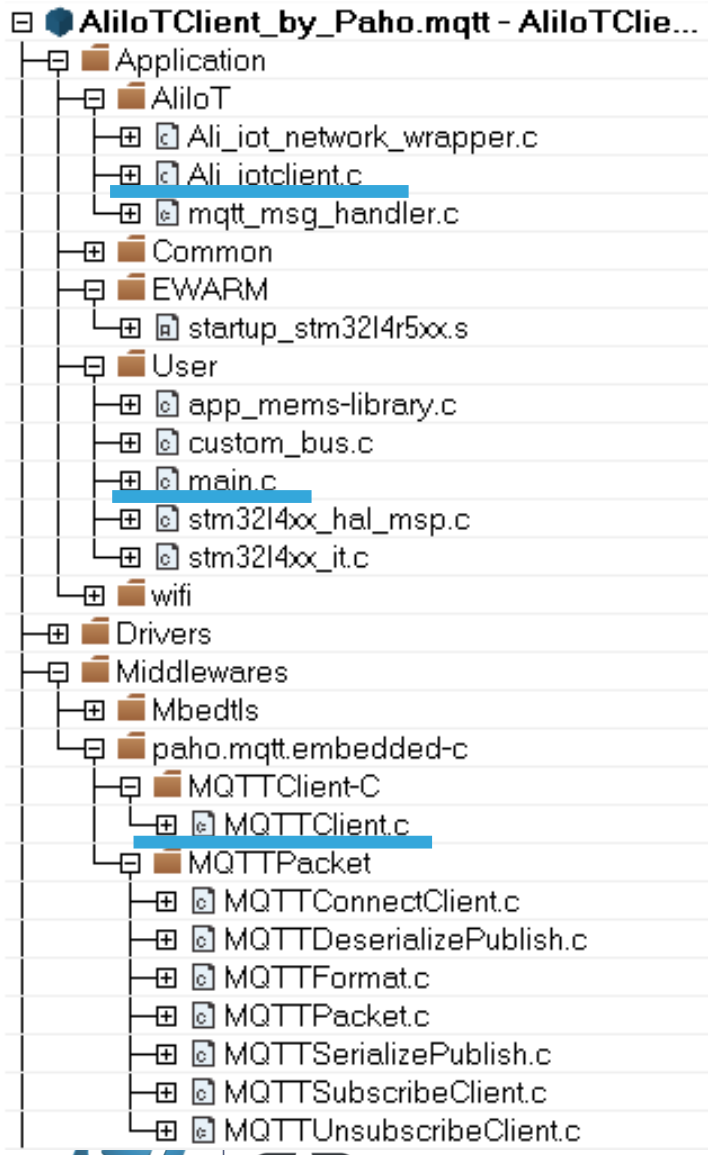
```
int MQTTPublish(MQTTClient* c, const char* topicName, MQTTMessage* message)
{
```

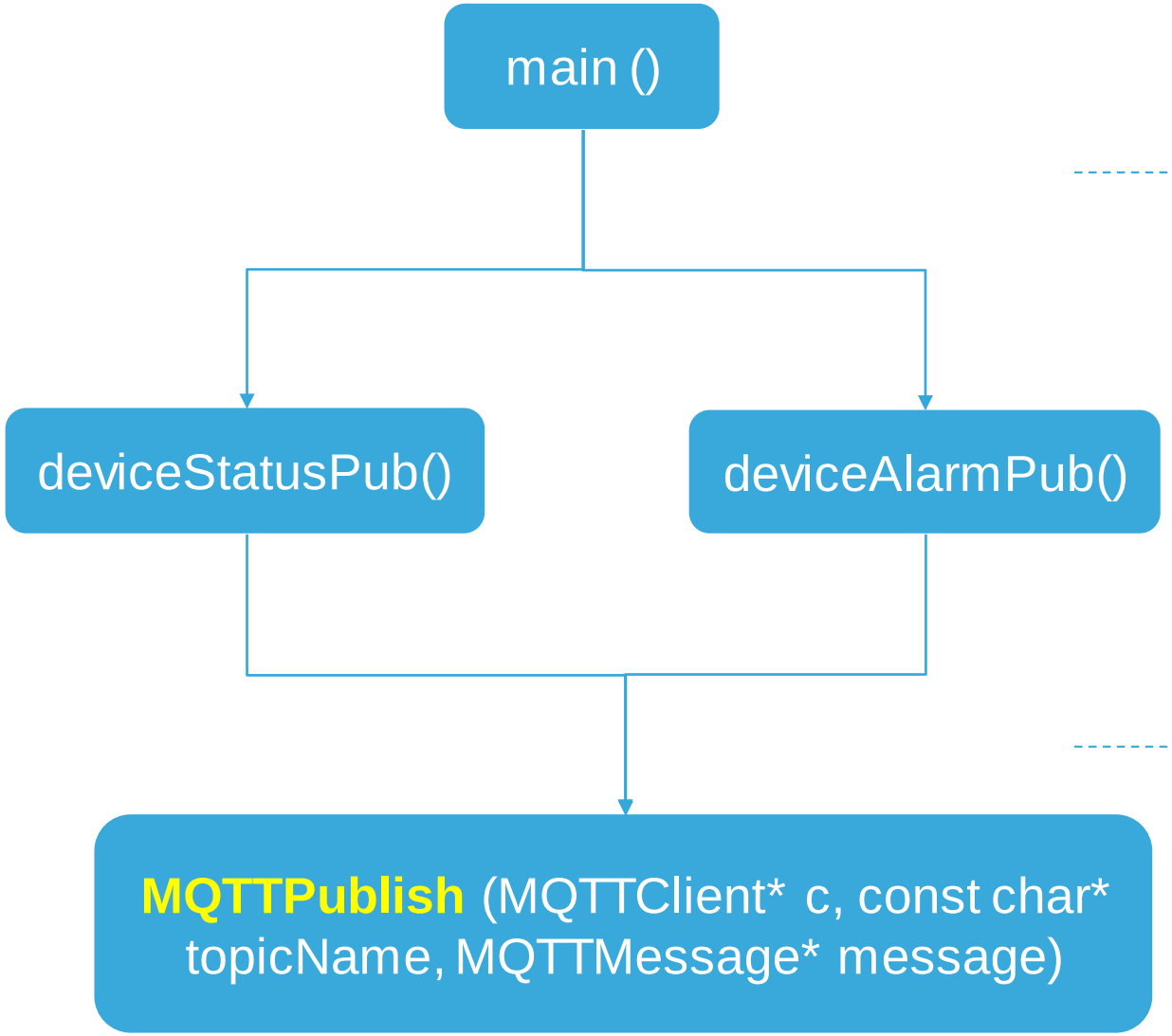


MQTTPublish(
&Client, temp_hum_topic,
&MQTT_msg)

MQTTPublish(
&Client, tempAlarm_topic,
&MQTT_msg)

协议栈中的调用





main.c

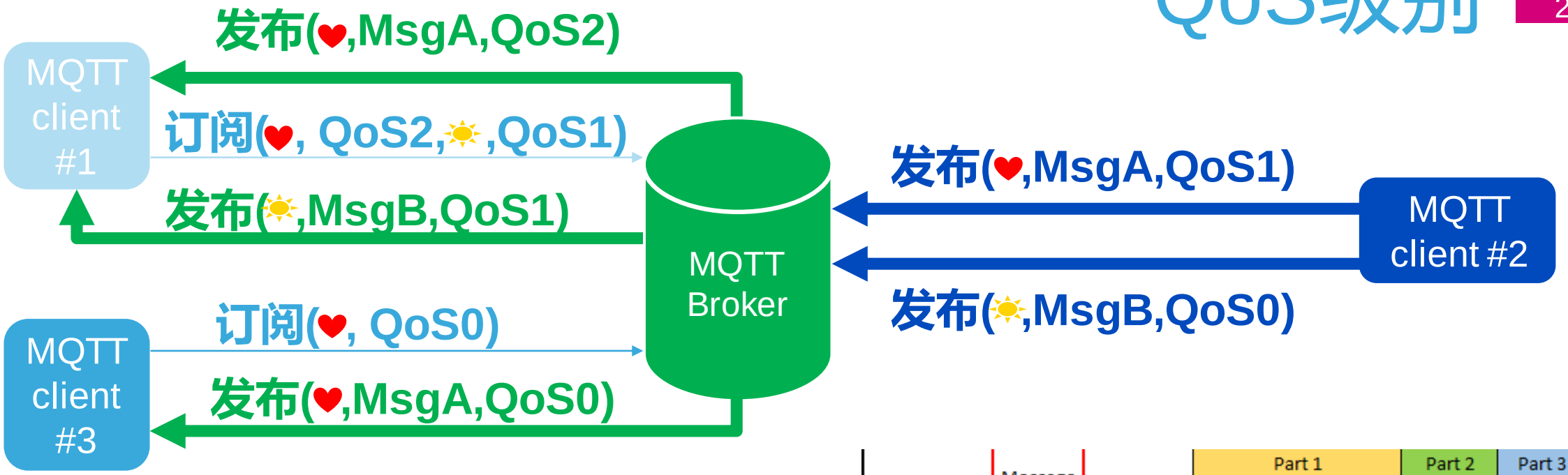
ali_iotclient.c

MQTTClient.c

life.augmented

阿里云

QoS级别



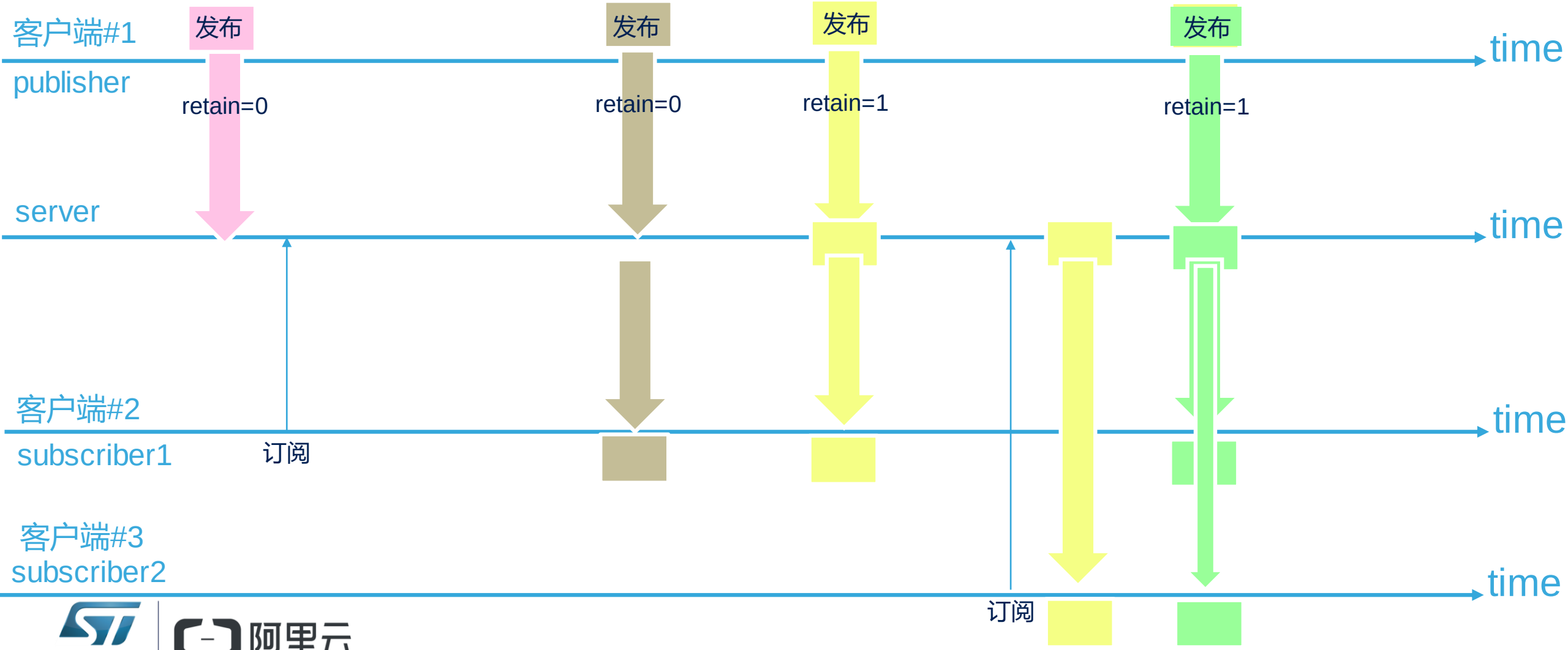
		发送方报文	接收方报文
QoS0	最多收到一次	Publish	
QoS1	最少收到一次	Publish	Puback
QoS2	收且仅收到一次	Publish Pubrel(ease)	Pubrec(ord) Pubcomp(lete)

	Message Type value	Direction		Part 1			Part 2	Part 3
				Flag in fix-header			variable-header	payload
		C-->S	S-->C	UDP	QoS level	RETAIN	Pkt ID	
PUBLISH	3							
PUBACK	4							
PUBREC	5				2			
PUBREL	6				2			
PUBCOMP	7				2			
SUBSCRIBE	8							
SUBACK	9							
UNSUBSCRIBE	10							
UNSUBACK	11							

life.augmented 阿里云

消息的Retain

	Message Type value	Direction		Part 1			Part 2	Part 3
				Flag in fix-header			variable-header	payload
		C-->S	S-->C	UDP	QoS level	RETAIN	Pkt ID	
PUBLISH	3							



资料下载

- Paho.mqtt.embedded-c
  - MQTTPacket
    - 底层的C代码，提供基本的解析数据，以及将数据串行化的功能
    - 是上层接口的基础，也可以单独使用
  - MQTTClient-C 提供C的上层接口，针对那些不支持C++编程的平台

	src
MQTTClient-C	[FreeRTOS] MQTTFreeROTS.c/.h MQTTClient.c/.h
MQTTPacket	MQTTConenct.h, MQTTConnectClient.c, MQTTConnectServer.c, MQTTPublish.h, MQTTDeserializePublish.c, MQTTSerializePublish.c, MQTTSubscribe.h, MQTTSubscribeClient.c, MQTTSubscribeServer.c, MQTTUnsubscribe.h, MQTTUnsubscribeClient,c, MQTTUnsubscribeServer.c, StackTrace.h, MQTTFormate.c/.h, MQTTPacket.c/.h,

- MQTT协议规范 v3.1.1

阿里云IoT平台侧的MQTT实现

支持的MQTT协议版本	兼容3.1和3.1.1版本
与标准MQTT的区别	不支持遗嘱消息
	不支持retained message
	不支持QoS2
	心跳间隔范围：30~1200秒，建议300秒以上
	在原生MQTT topic上支持RRPC同步模式，服务器可以对设备进行同步访问(得到设备回执)
安全等级	支持TLS v1, v1.1, v1.2版本
	支持非加密通道连接
MQTT连接方式	MQTT客户端域名直连**（本课程使用此种连接方式）**
	使用HTTPS认证再连接模式**（参见ST资料 I-Cube-Aliyun）**

内容简介

- 例程演示
  - 例程流程图
  - 演示视频
- MQTT协议介绍
  - 协议特性
  - 协议模型和报文格式
  - 报文使用序列
  - 阿里云IoT平台侧MQTT协议实现
- 项目例程介绍
  - 软件包和项目结构
  - 使用CubeMX生成系统初始化框架和代码
  - 网络通信的管理（网络通信抽象层和wifi驱动）
  - 使用Paho MQTT客户端协议栈连接阿里云IoT平台
  - 例程参数的存储及Sensor数据的读取

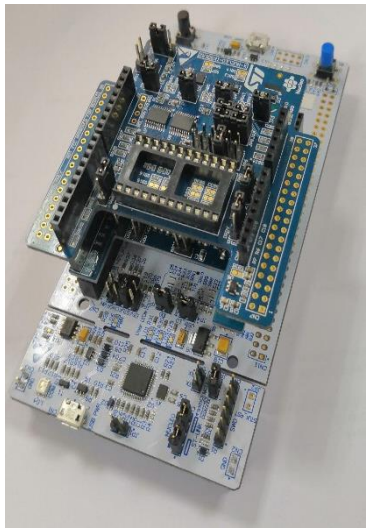
节点端系统框图

X-NUCLEO-IKS01A2 Sensor扩展板

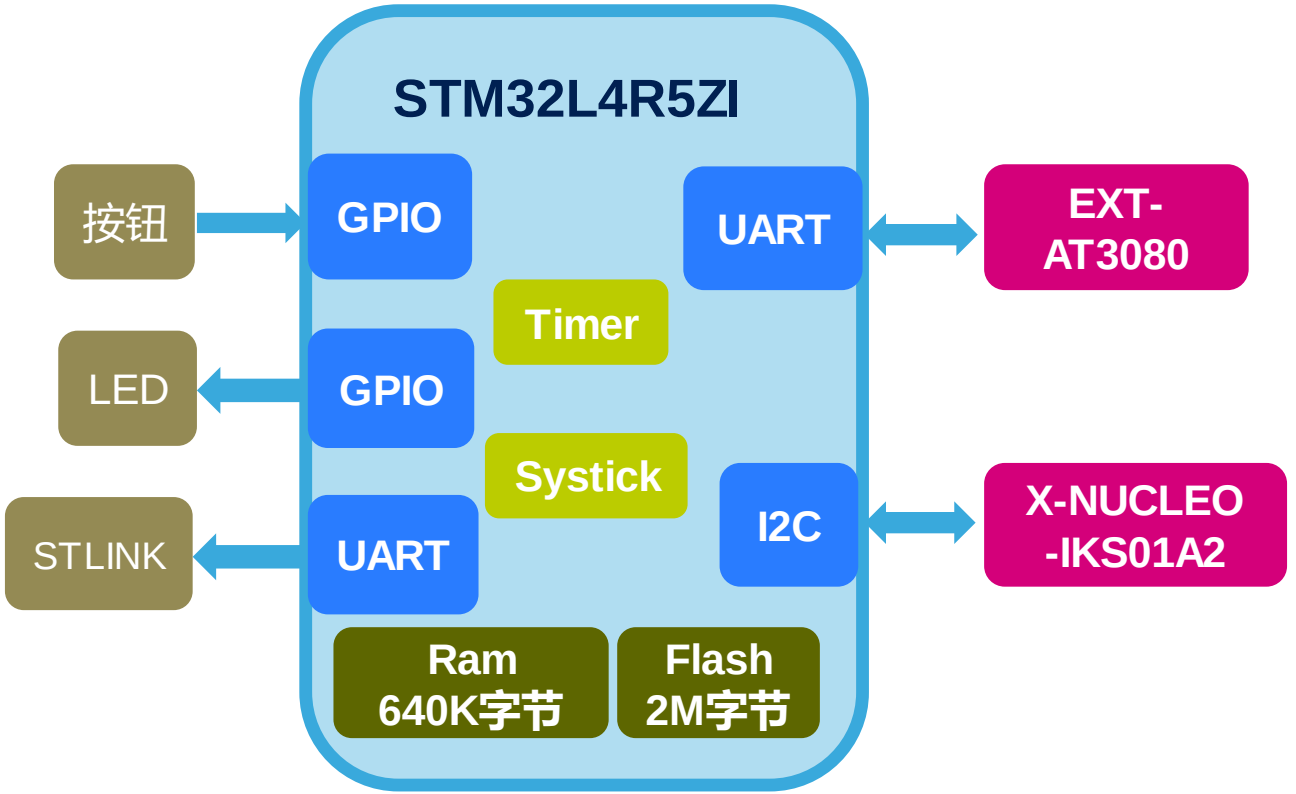
STM32L4R5ZI

NUCLEO-L4R5ZI

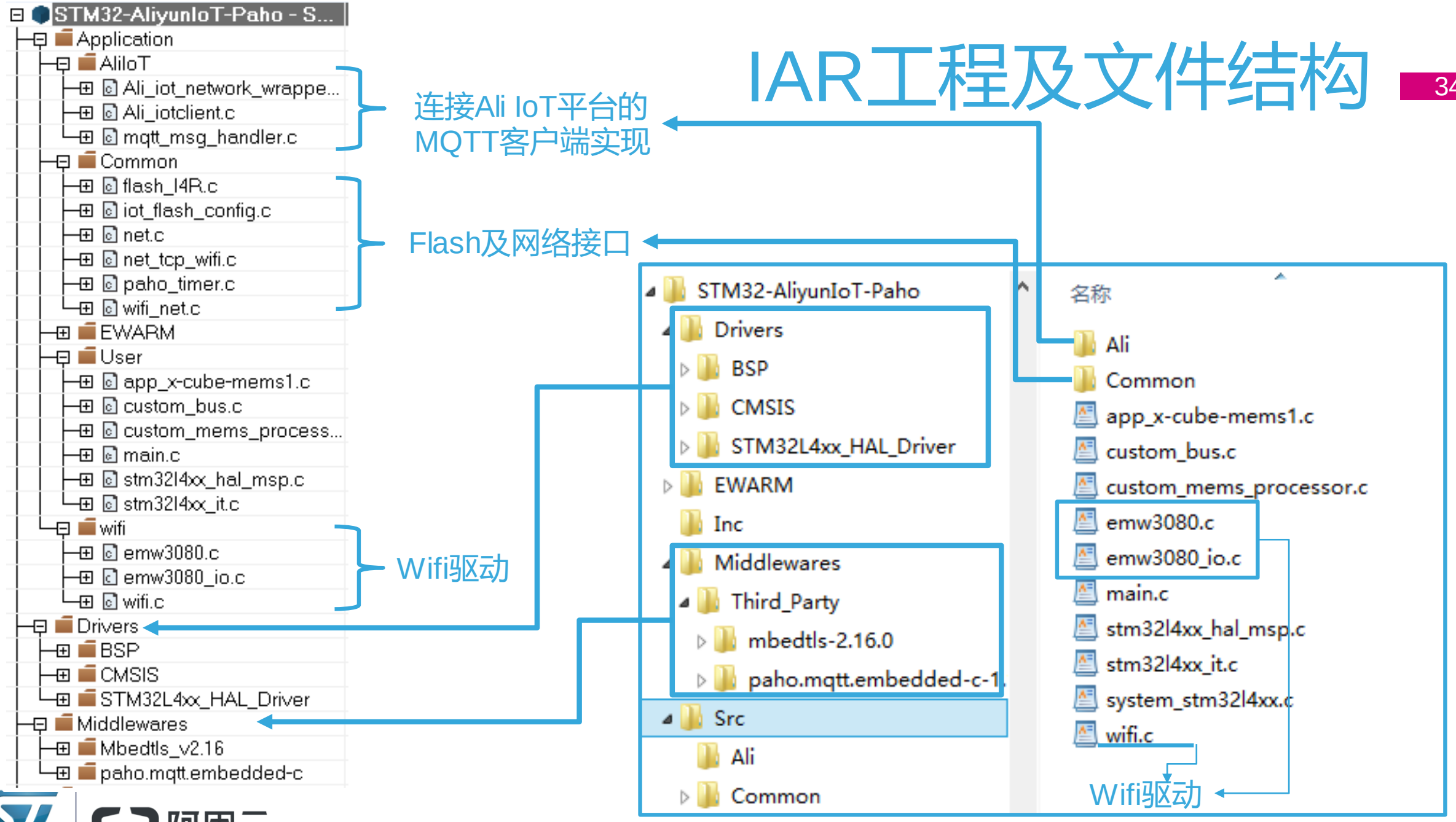
Arduino Uno接口



EXT-AT3080 Wifi扩展板



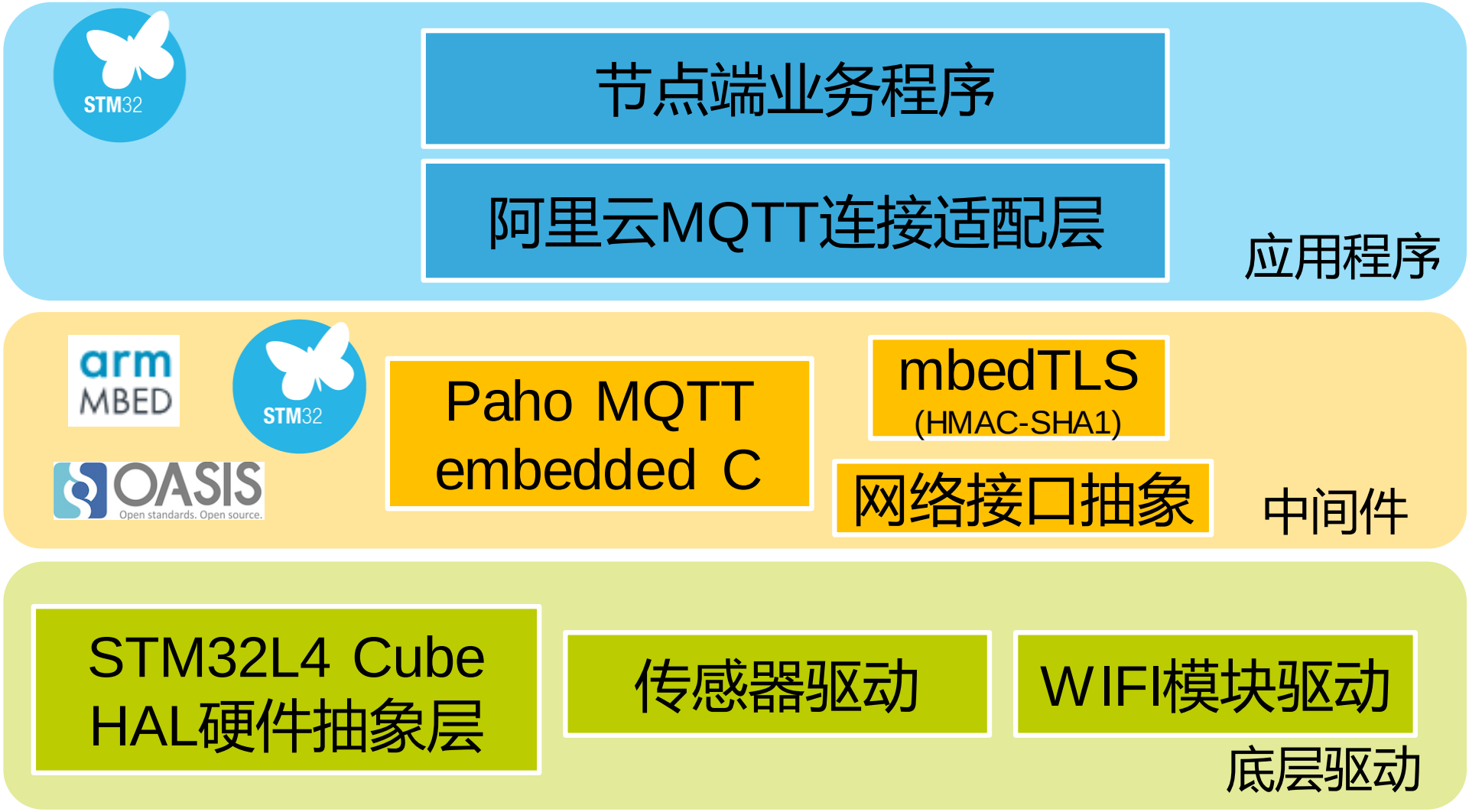
IAR工程及文件结构



life.augmented

阿里云

项目例程软件架构


节点端业务程序
阿里云MQTT连接适配层
应用程序
Paho MQTT
embedded C
mbedTLS
(HMAC-SHA1)
网络接口抽象
中间件
STM32L4 Cube
HAL硬件抽象层
传感器驱动
WIFI模块驱动
底层驱动

MCU外设的使用

- 与WIFI扩展板的接口定义

NCULEO板接口号	接口引脚号	接口引脚名	STM32引脚	STM32外设配置
CN10	14	D1	PD8	USART3_TX
CN10	16	D0	PD9	USART3_RX

- 与传感器扩展板的接口定义

NCULEO板接口号	接口引脚号	接口引脚名	STM32引脚	STM32外设配置
CN7	2	D15	PB8	I2C1_SCL
CN7	4	D14	PB9	I2C1_SDA

- 虚拟串口接口定义

	STM32引脚	STM32外设配置
连到STLINK USB 虚拟串口	PG7	LPUART1_TX
	PG8	LPUART1_RX

- 与USER按键接口定义

NCULEO板	STM32引脚	STM32外设配置	功能
蓝色User按键	PC13	GPIO外部中断，下降沿触发	控制进入虚拟串口输入模式，输入WIFI配网，三元组等信息

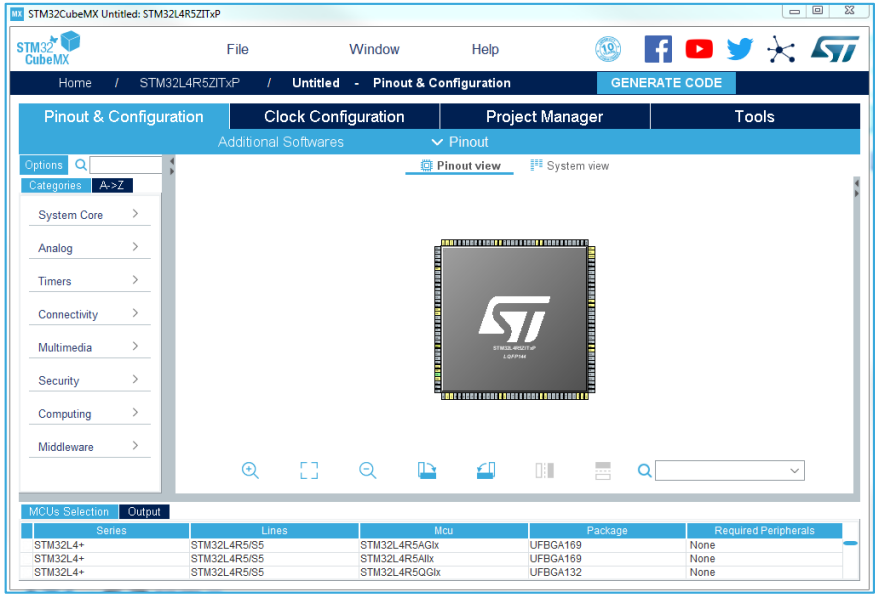
- 与LED灯的接口定义

NCULEO板	STM32引脚	STM32外设配置	功能
LD1(绿)	PC7	GPIO 输出	每次上传温湿度信息时，闪烁一次
LD3(红)	PB14	TIM15CH1，PWM输出	高温报警提醒

- Systick的用途：应用的延时功能，Paho协议栈Timer

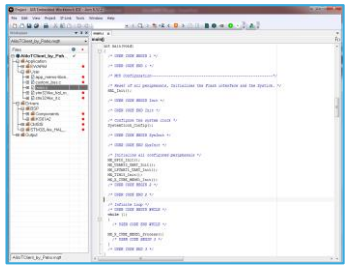
使用CubeMX初始化系统

(1）选择MCU型号



(2）

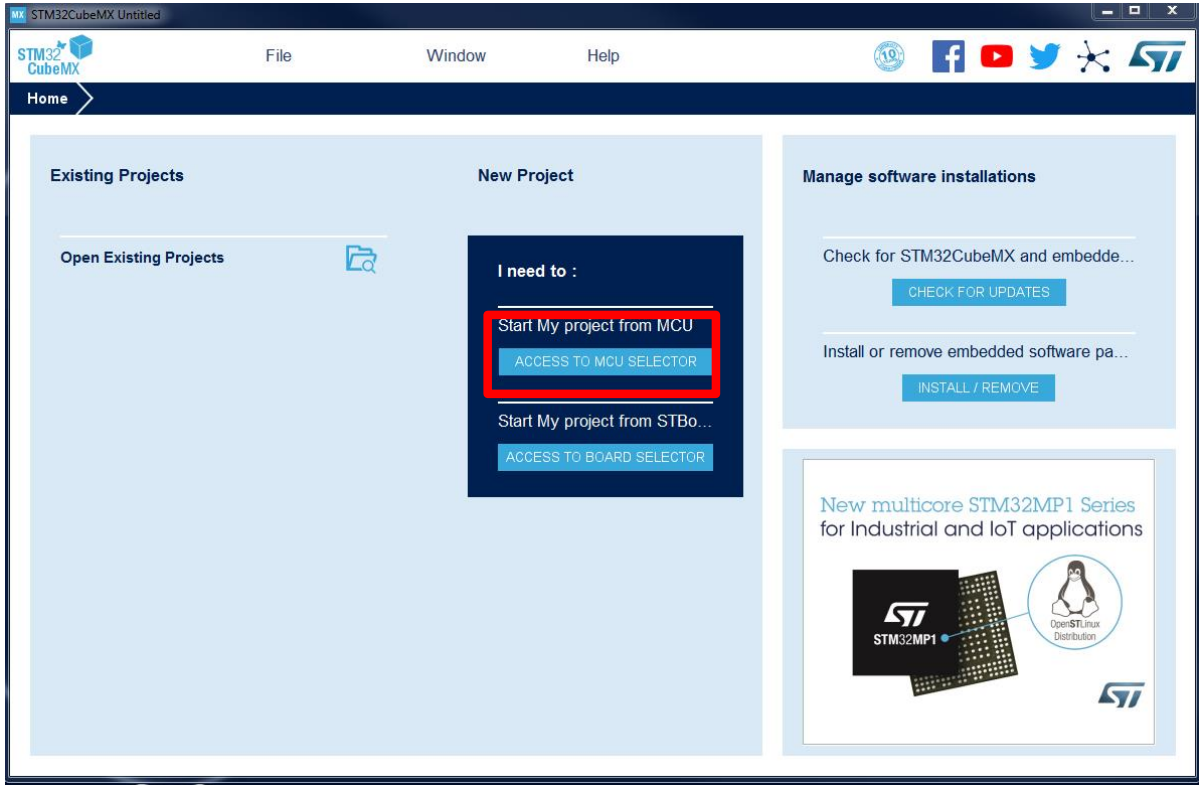
- 引脚/外设配置
  (UART
  /I2C
  /EXT
  /TIMER)
- 时钟配置
- 插件配置

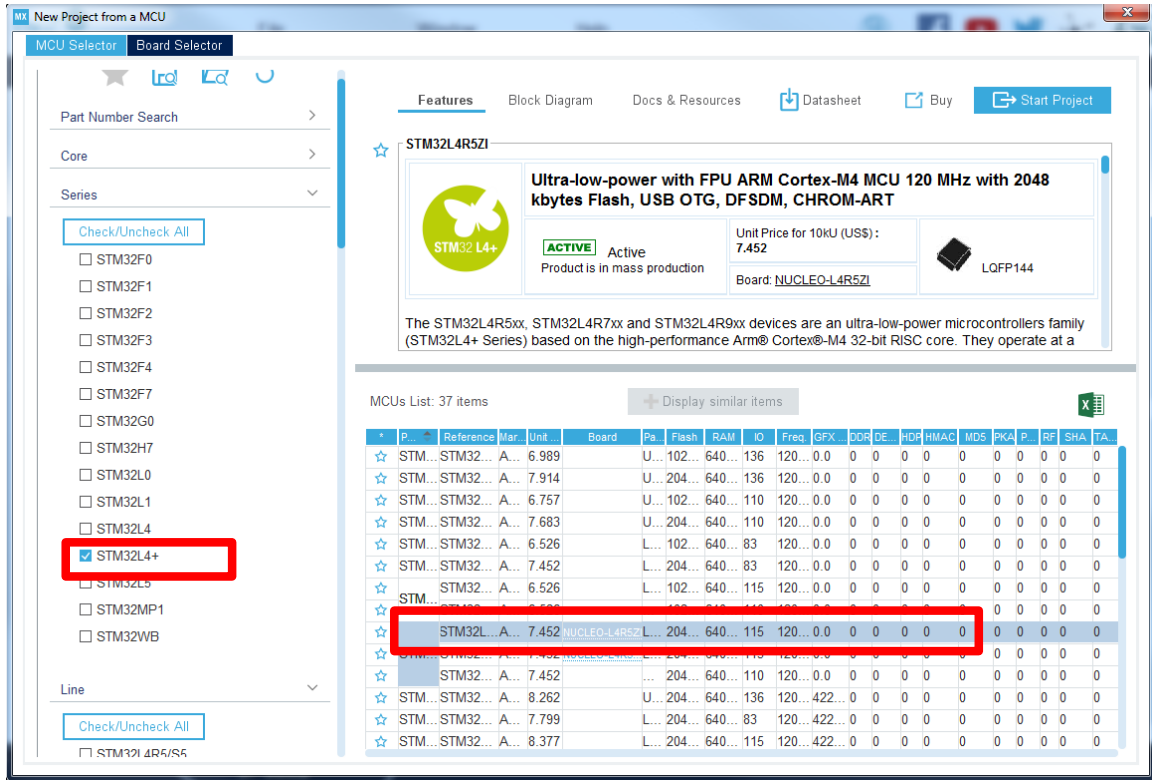


(3）生成初始化工程

使用CubeMX初始化系统（1）

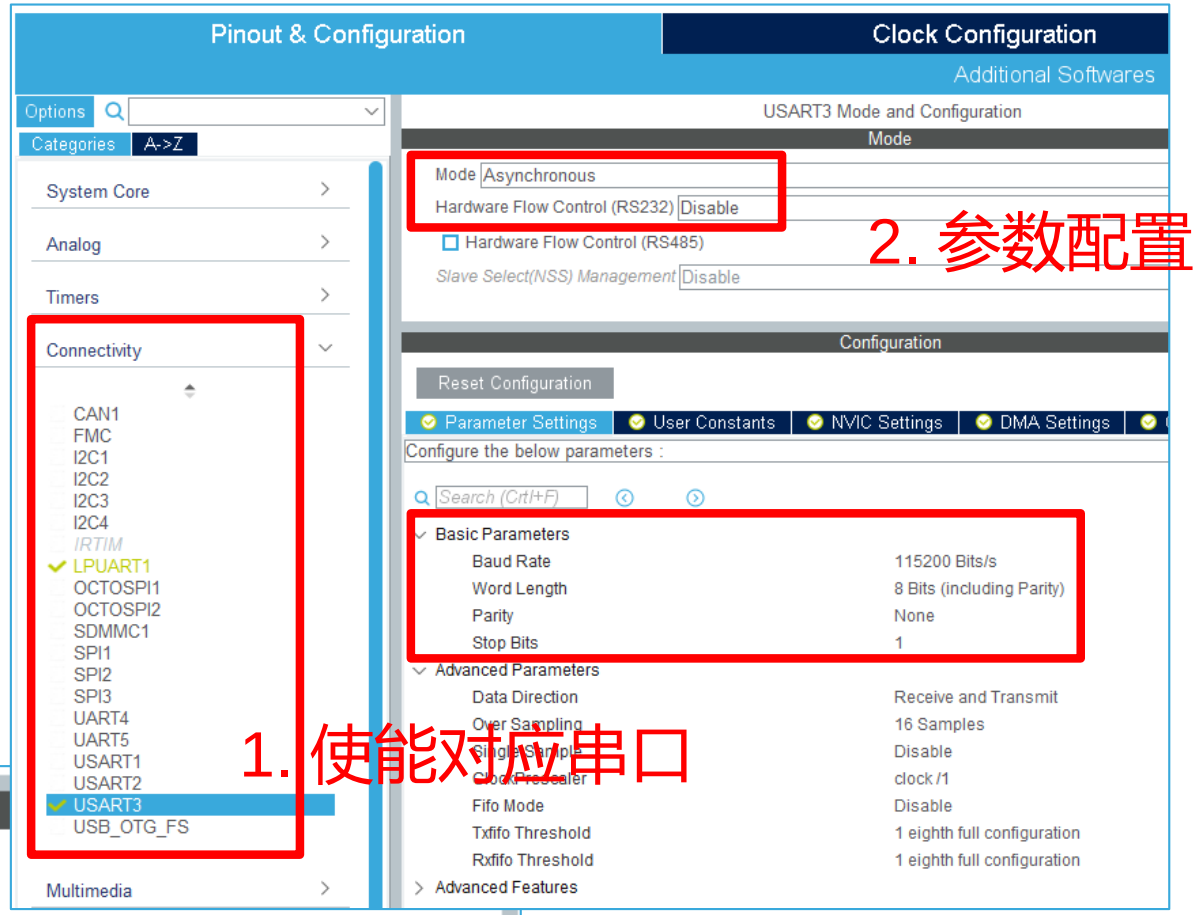
- MCU型号选择：STM32L4R5ZITx

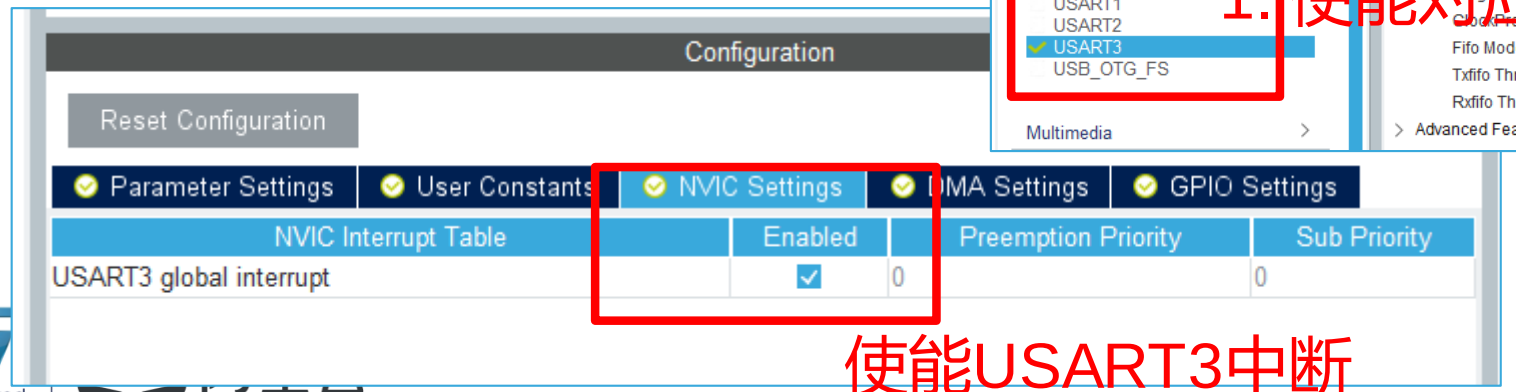




使用CubeMX初始化系统（2）：外设配置

- 分别使能两个串口
  - 与WIFI模块通信（USART3）
  - 打印程序运行信息（LPUART1）
- 参数配置
  - 波特率：115200
  - 数据长度：8bit
  - 1位停止位
  - 无奇偶校验



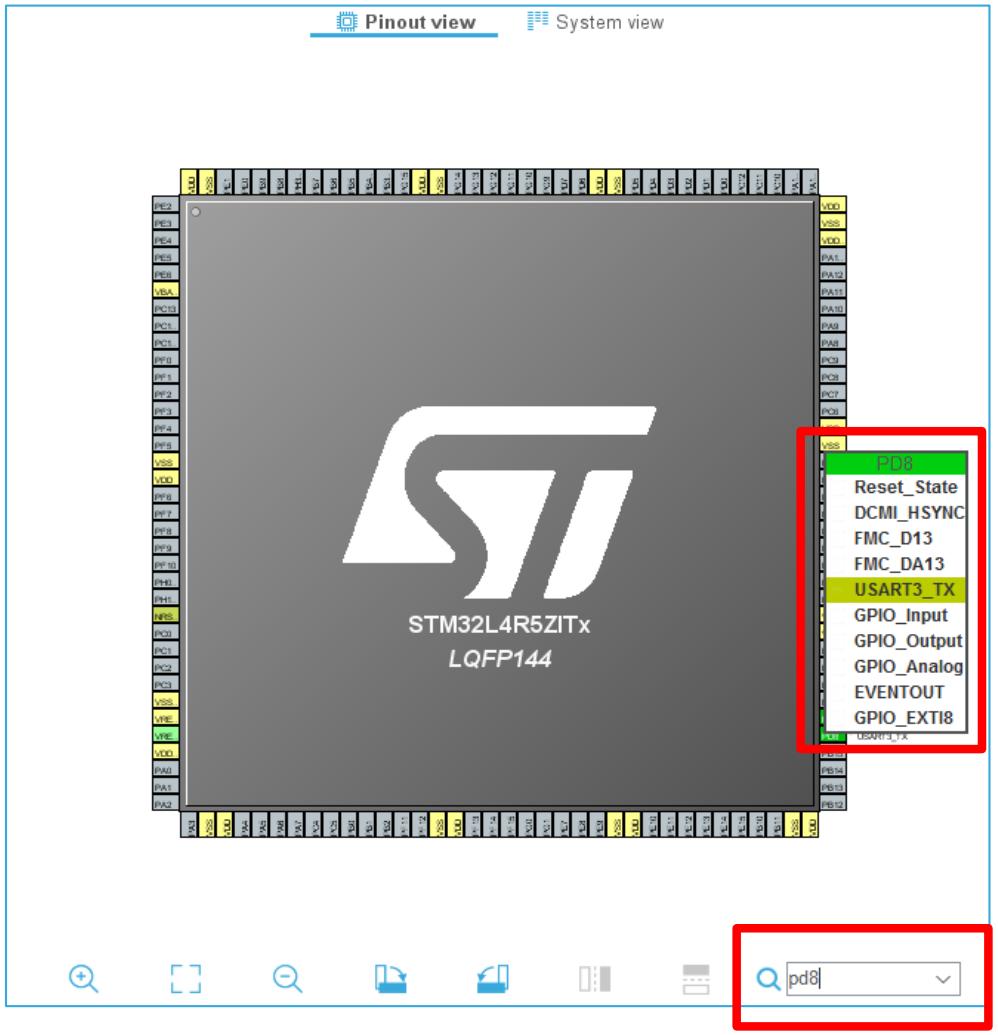


例程实际使用的引脚

- 根据实际所用引脚修改引脚定义

Wifi扩展板接口	外设配置	CubeMX默认分配的引脚	例程实际使用的引脚
D1	USART3_TX	PC4	PD8
D0	USART3_RX	PC5	PD9

	外设配置	CubeMX默认分配的引脚	例程实际使用的引脚
STLINK虚拟串口	LPUART1_TX	PC1	PG7
	LPUART1_RX	PC0	PG8



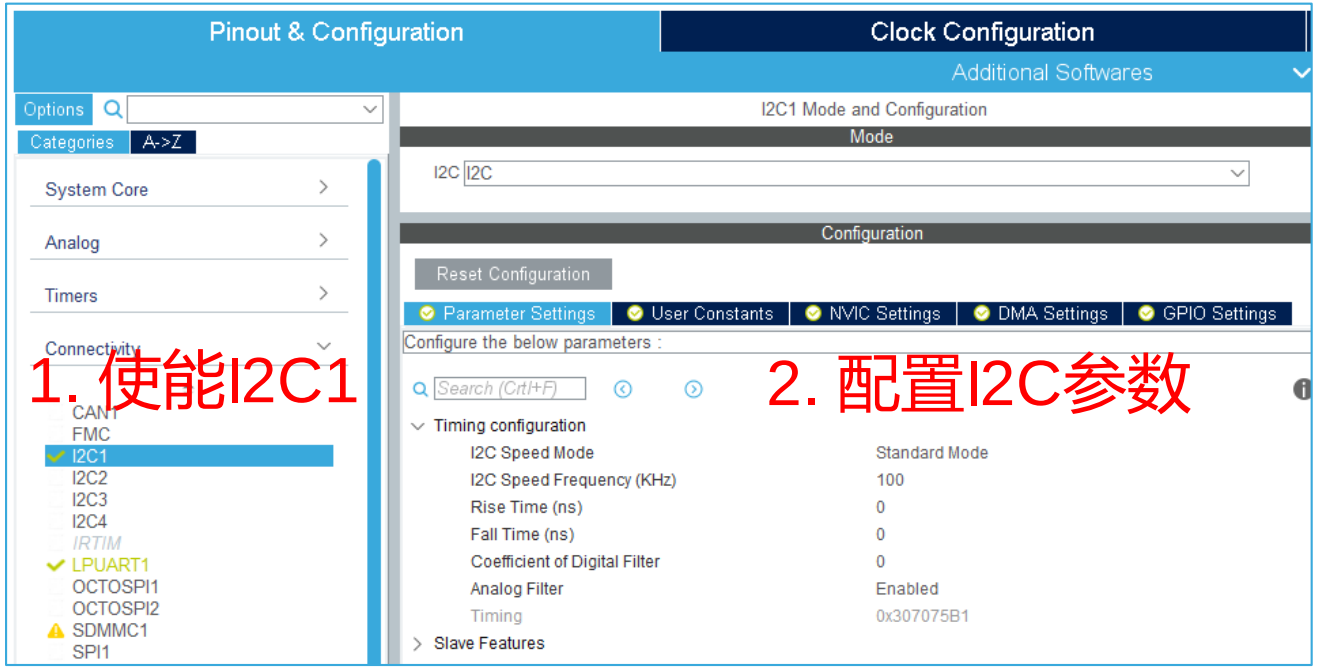
2. 选择对应引脚，配置成串口功能

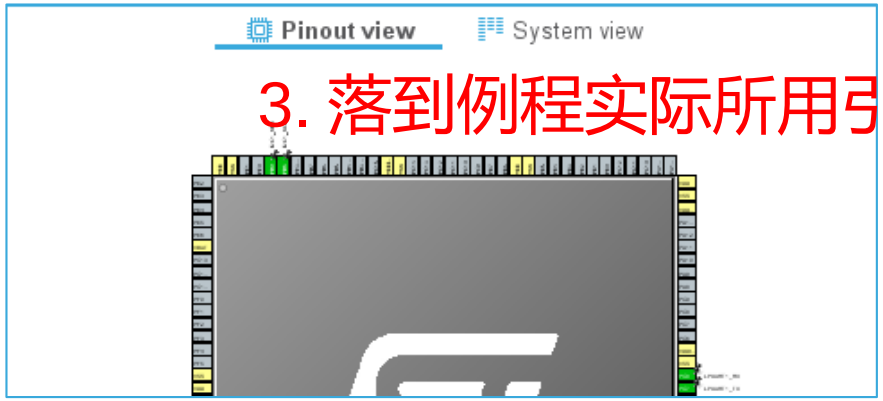
1. 查找需要配置的引脚

使用CubeMX初始化系统（2）

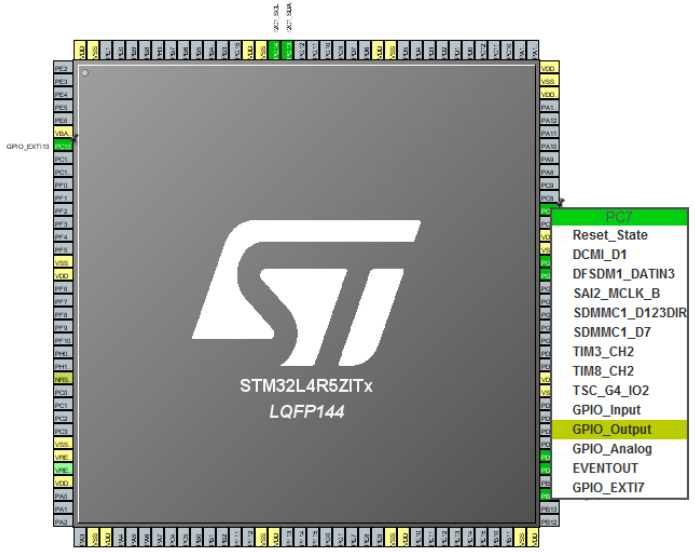
- 使能I2C1
- 参数配置
- 修改引脚定义

Sensor扩展板接口	外设配置	CubeMX默认分配的引脚	例程实际使用的引脚
	I2C1_SDA	PG13	PB9
	I2C1_SCL	PG14	PB8



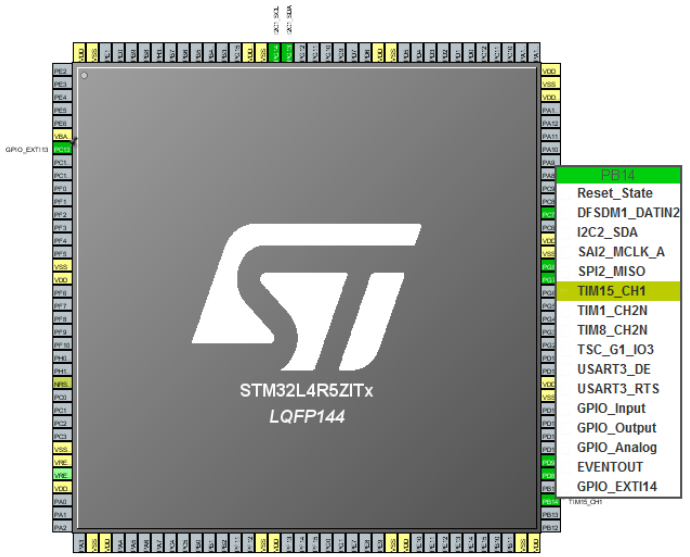


使用CubeMX初始化系统（2）



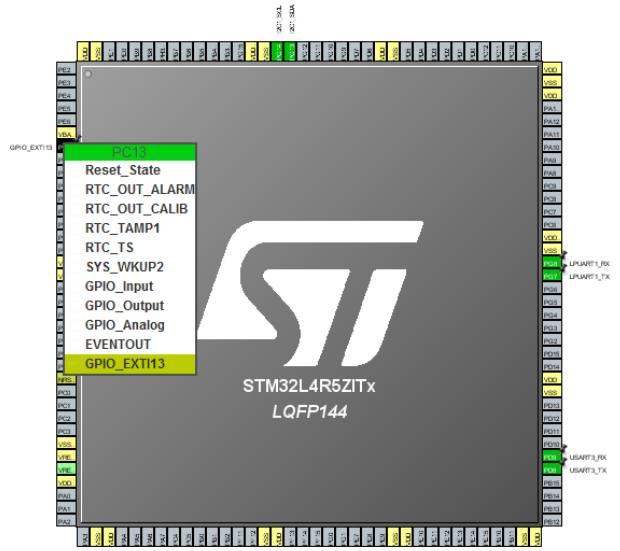
- 与LED灯的接口定义

LED	STM32引脚	STM32外设配置
LD1（绿）	PC7	GPIO 输出
LD3（红）	PB14	TIM15CH1，PWM输出



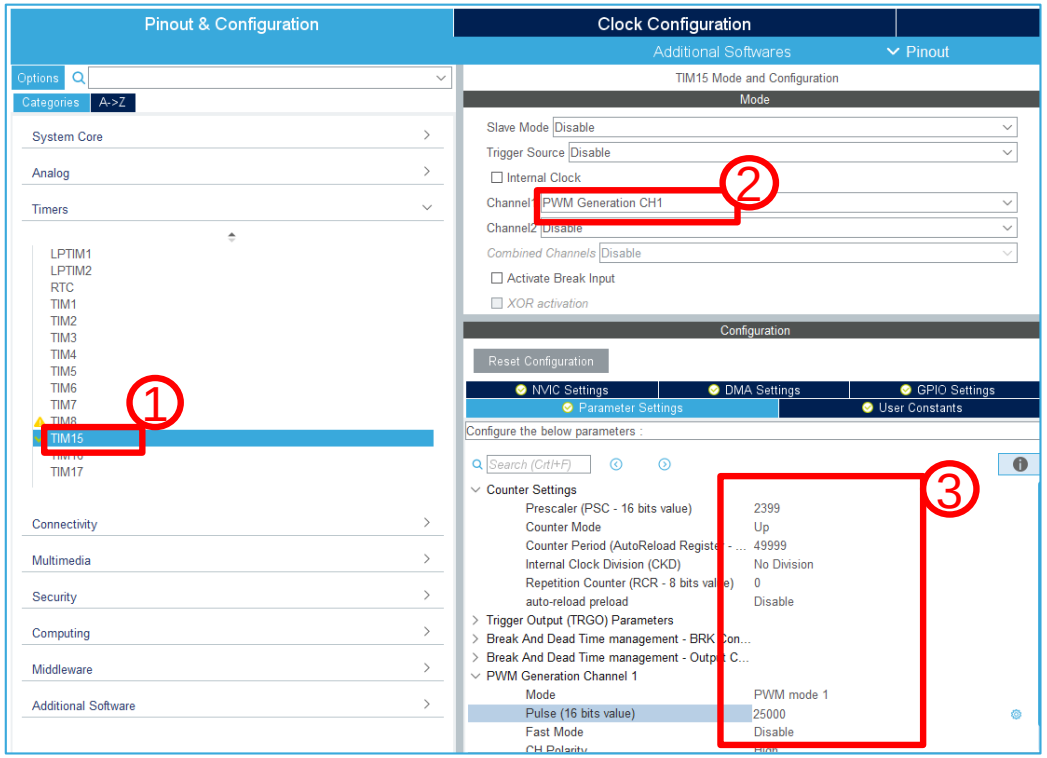
- 与USER按键接口定义

USER按键	STM32引脚	STM32外设配置
USER_BUTTON	PC13	GPIO外部中断，下降沿触发

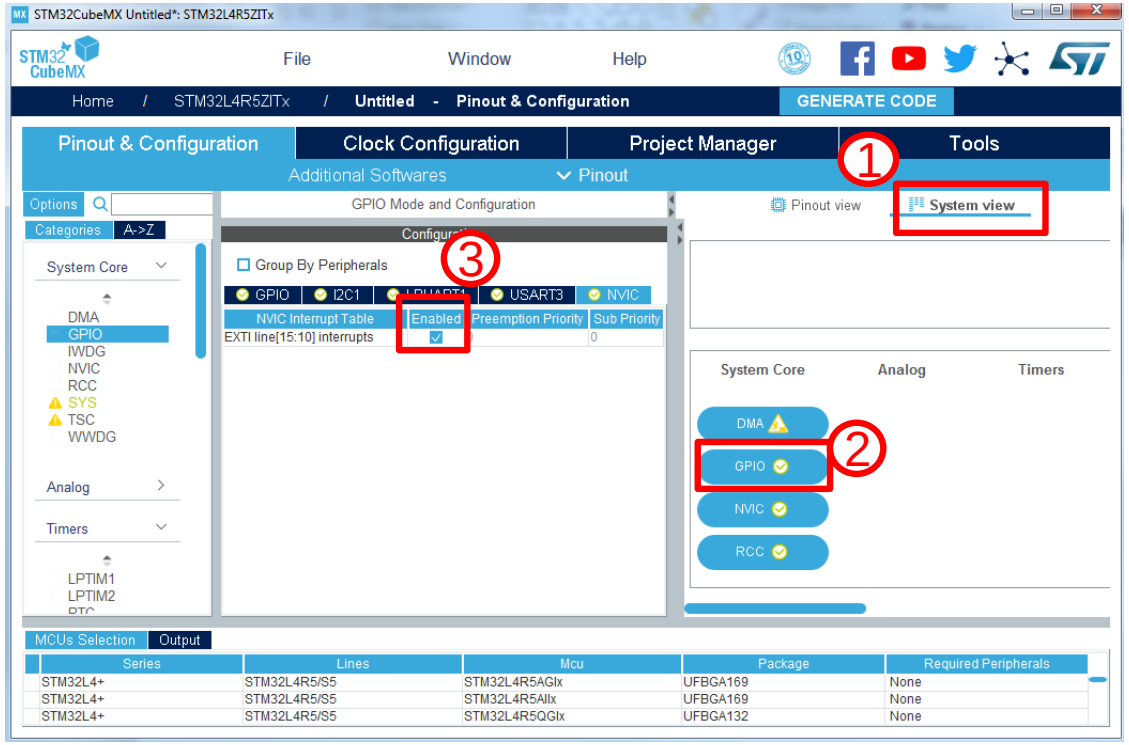


外部中断和定时器配置

- Timer15配置
  - PWM输出模式



- 使能EXT13外部中断



红灯0.5秒闪烁间隔 ← PWM ← 周期1s；占空比50%； ← 50KHz计数时钟；自动重载寄存器=50000-1；捕获/比较寄存器=25000；

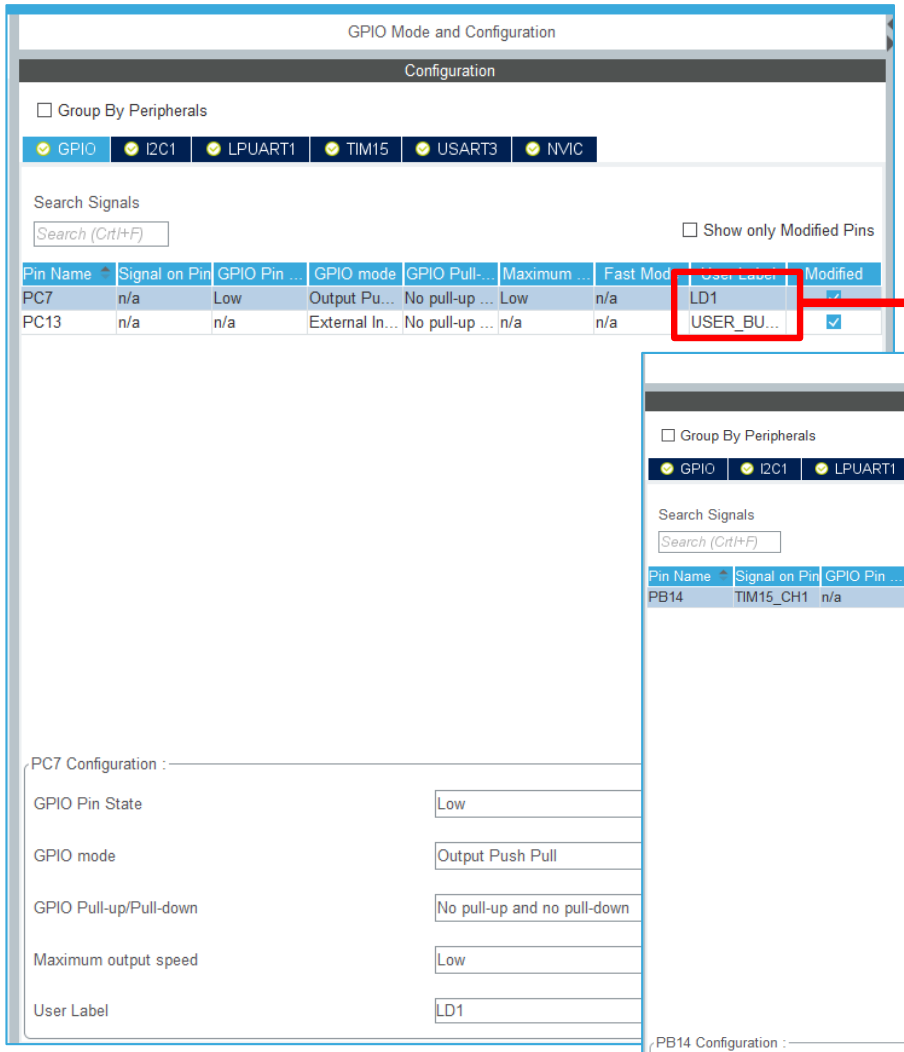
预分频=2400-1

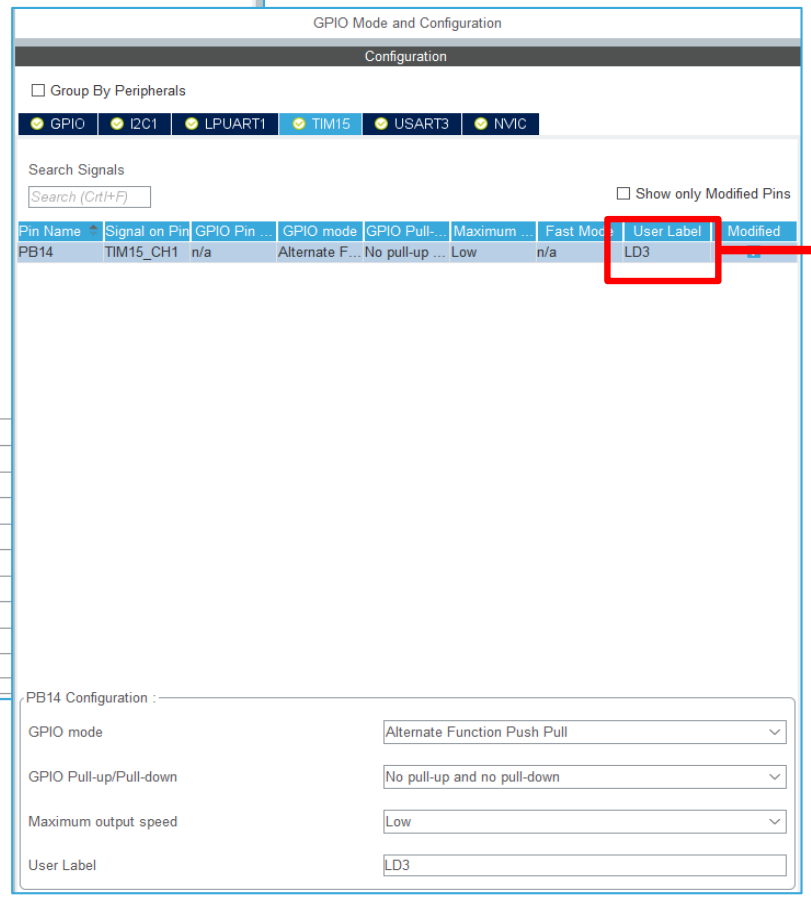
系统时钟=120MHz

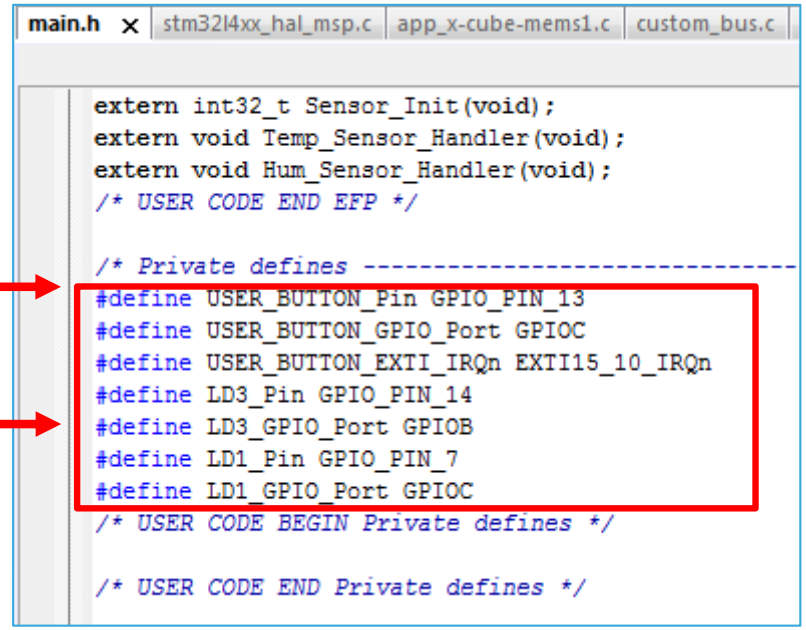
life.augmented | 阿里云

STM32 CubeMX

为GPIO口定义用户标签





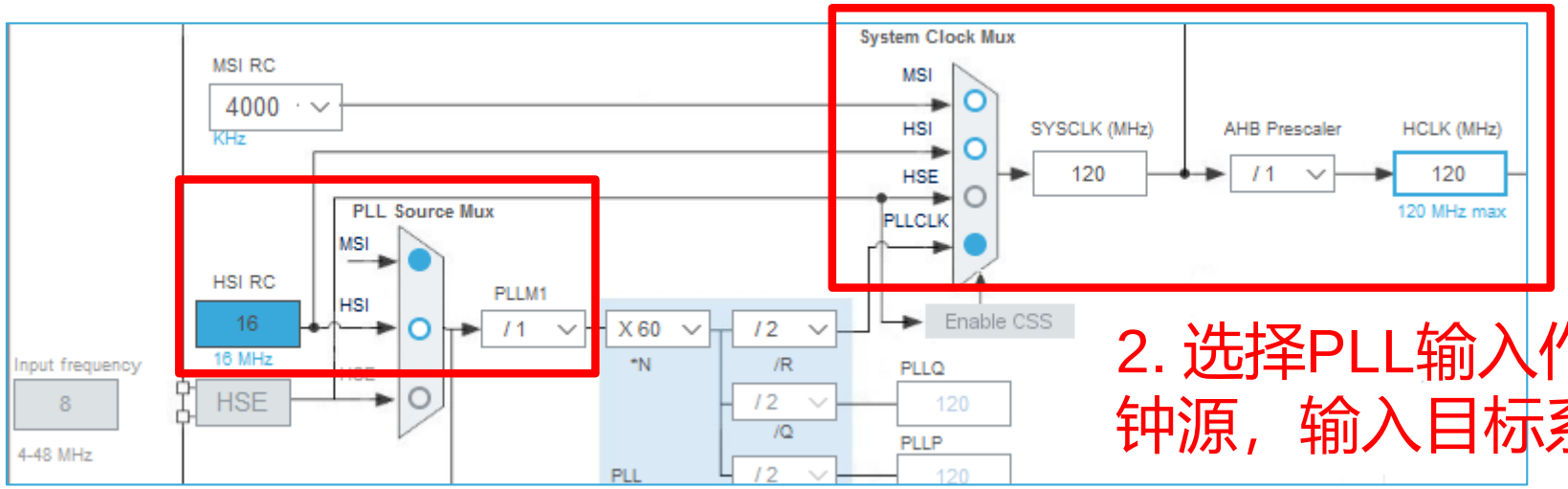


life.augmented

阿里云

使用CubeMX初始化系统（2）：时钟配置

- 时钟配置
  - PLL源选择（MSI）
  - 系统时钟120MHz
  - CubeMX自动计算时钟配置各参数

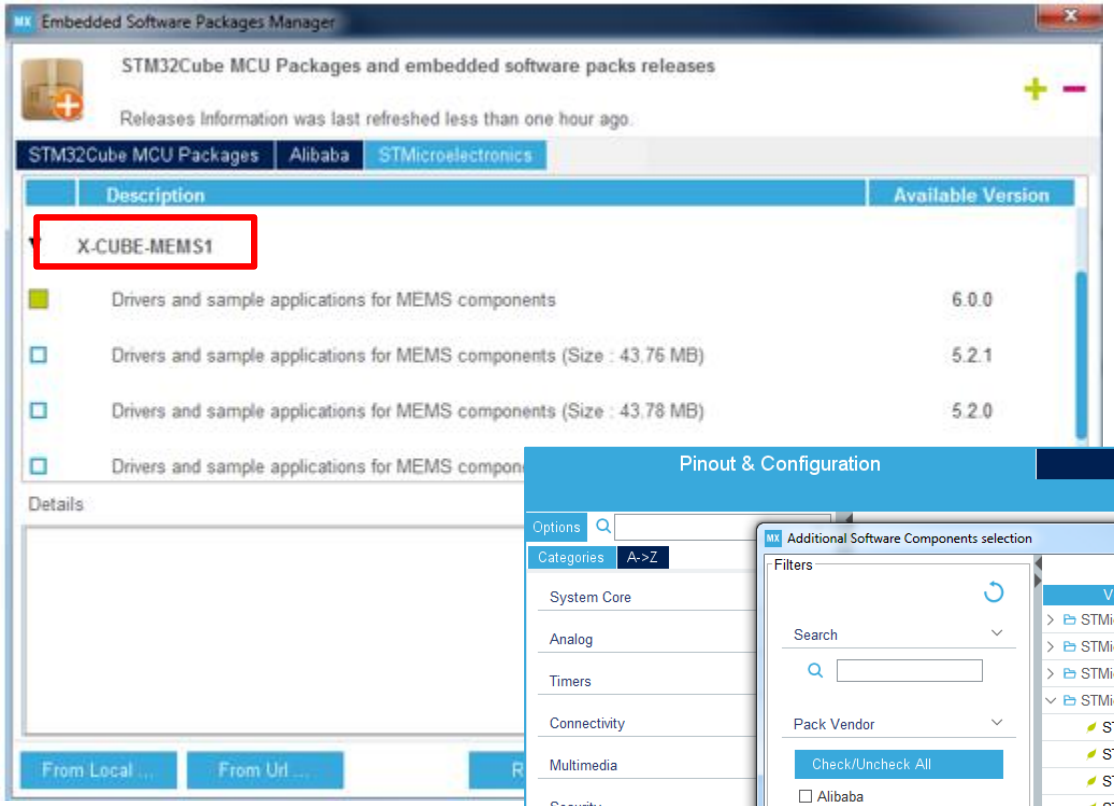


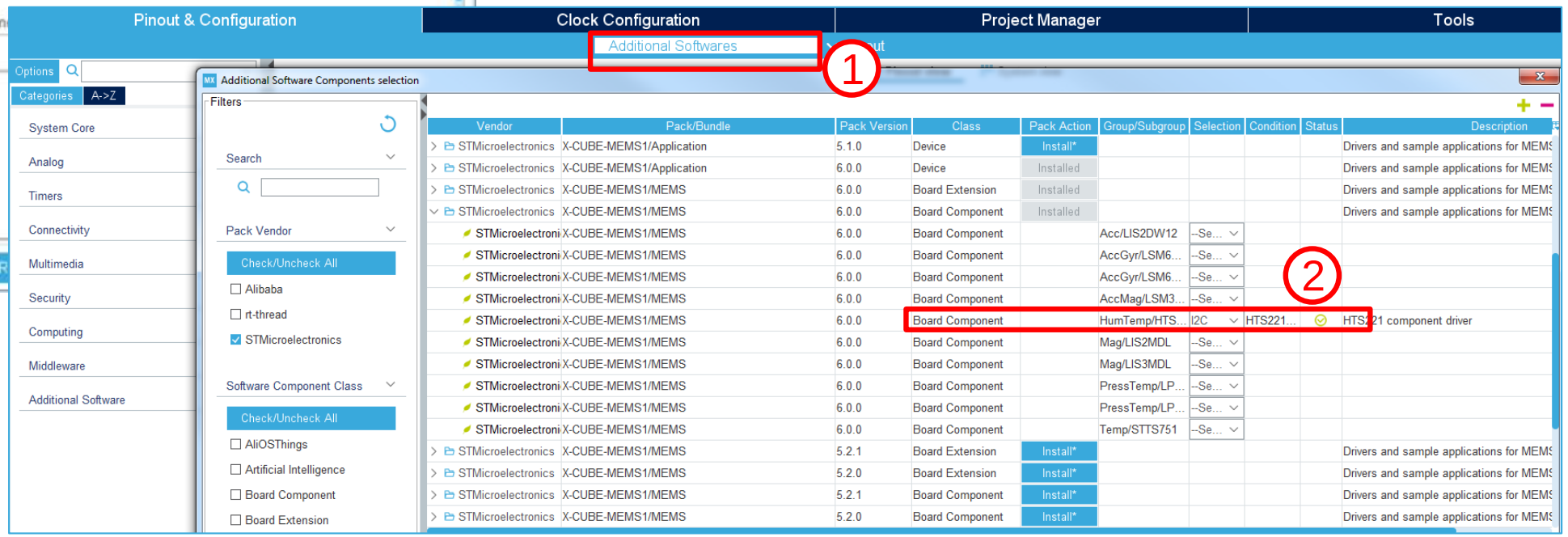
2. 选择PLL输入作为系统时钟源，输入目标系统时钟值

1. 选择MSI作为PLL输入

使用CubeMX初始化系统（2）

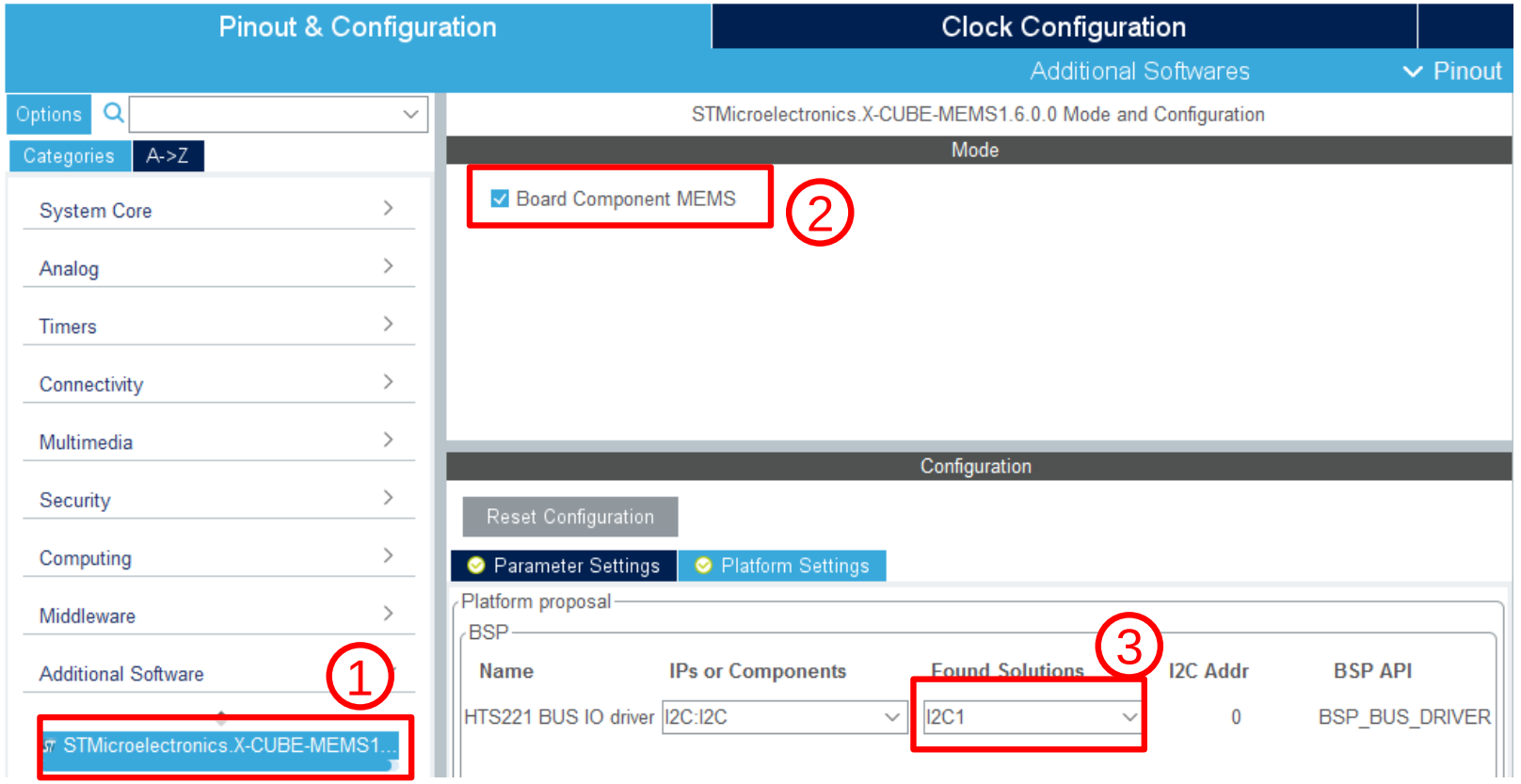
- 添加MEMS插件



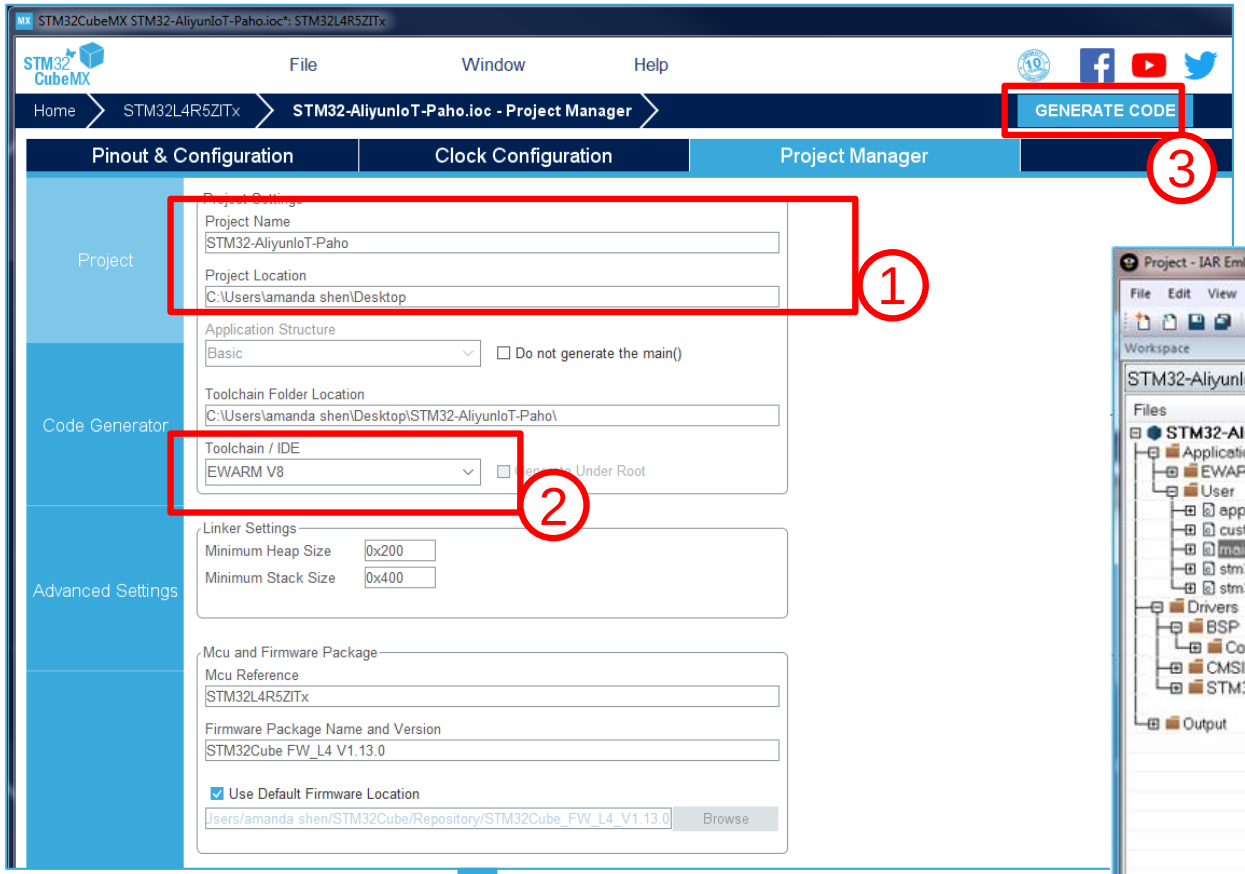


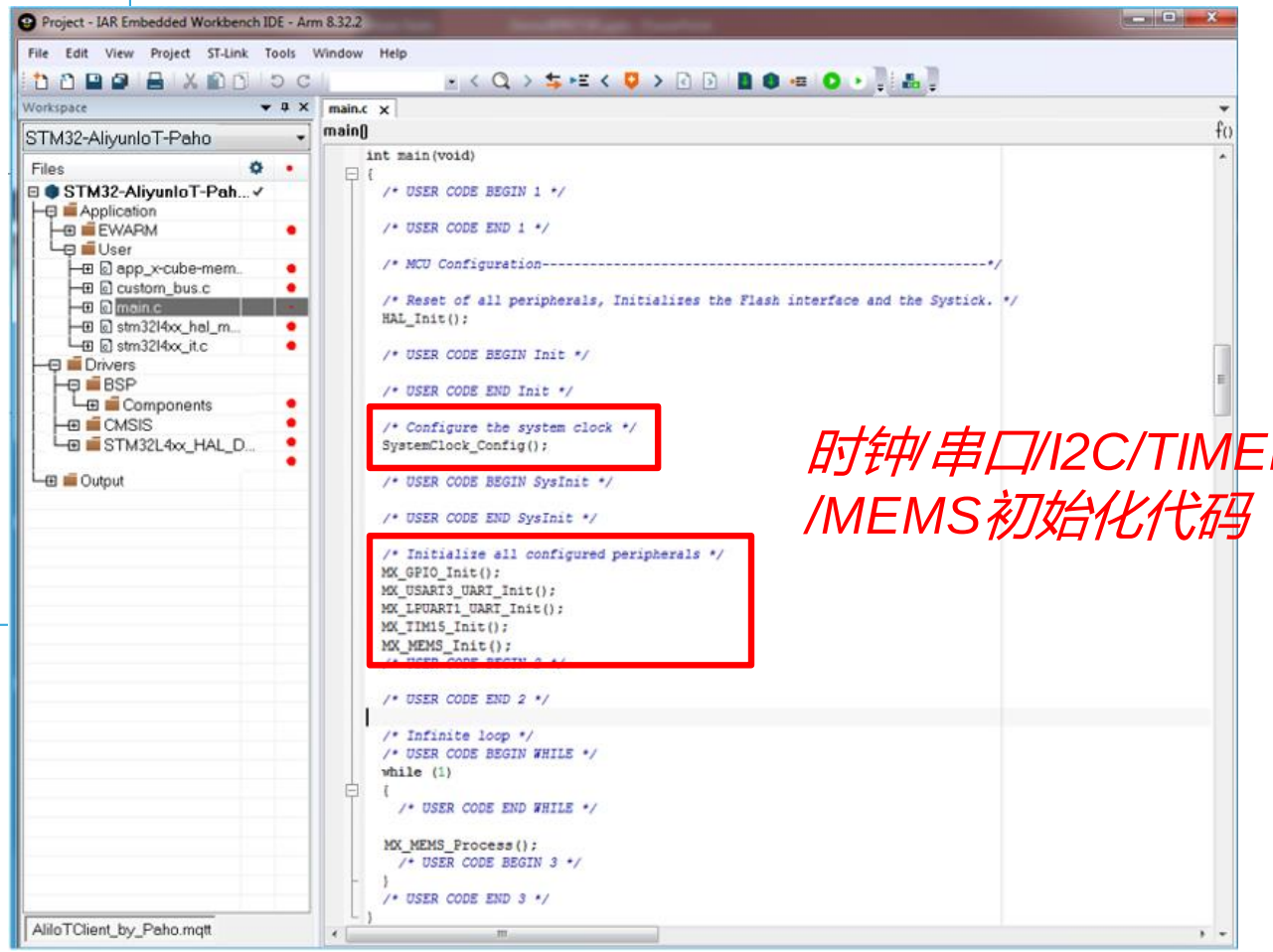
使用CubeMX初始化系统（2）

- 配置X-CUBE-MEMS驱动
  - 使用I2C1读取sensor数据



使用CubeMX初始化系统（3）





生成初始化好的IAR工程

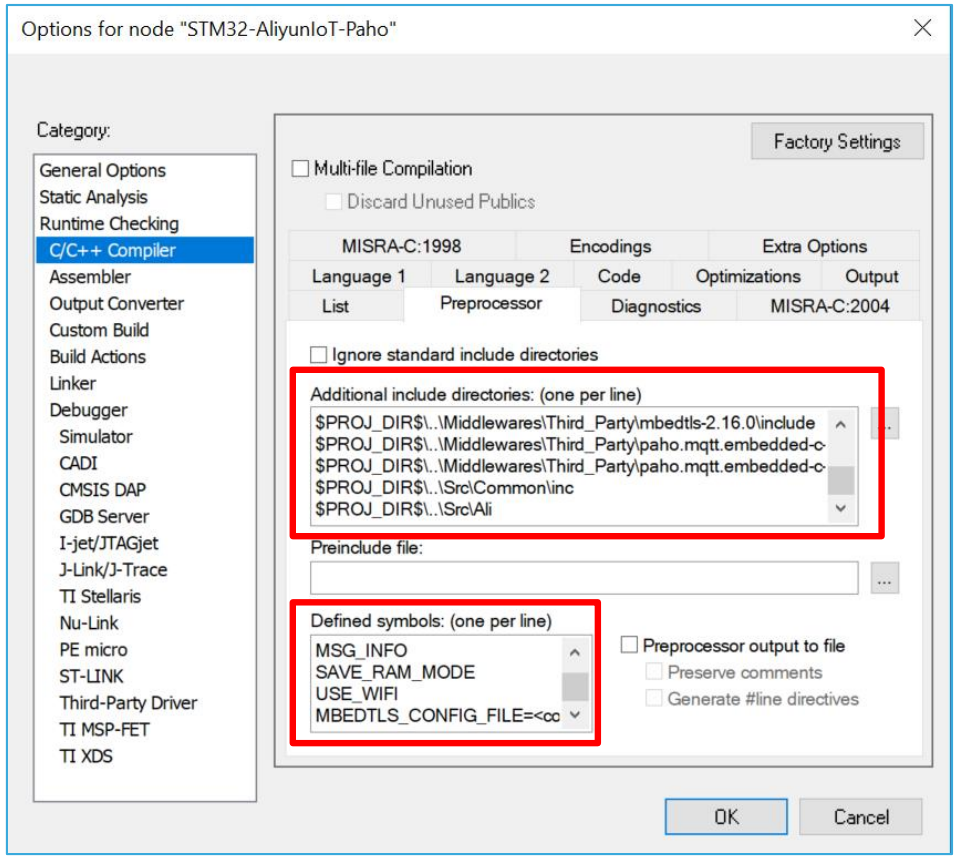
时钟/串口/I2C/TIMER /MEMS初始化代码

life.augmented

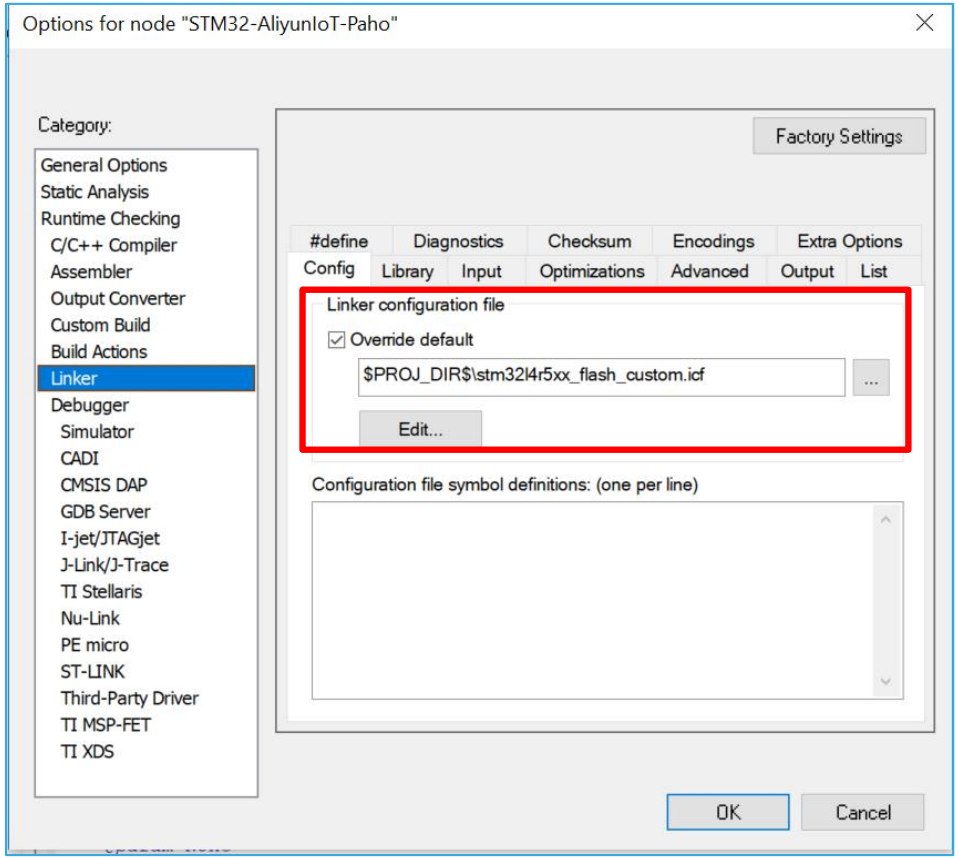
阿里云

使用CubeMX初始化系统（4）

- 配置IAR工程
  - 添加include路径
  - 添加预编译宏

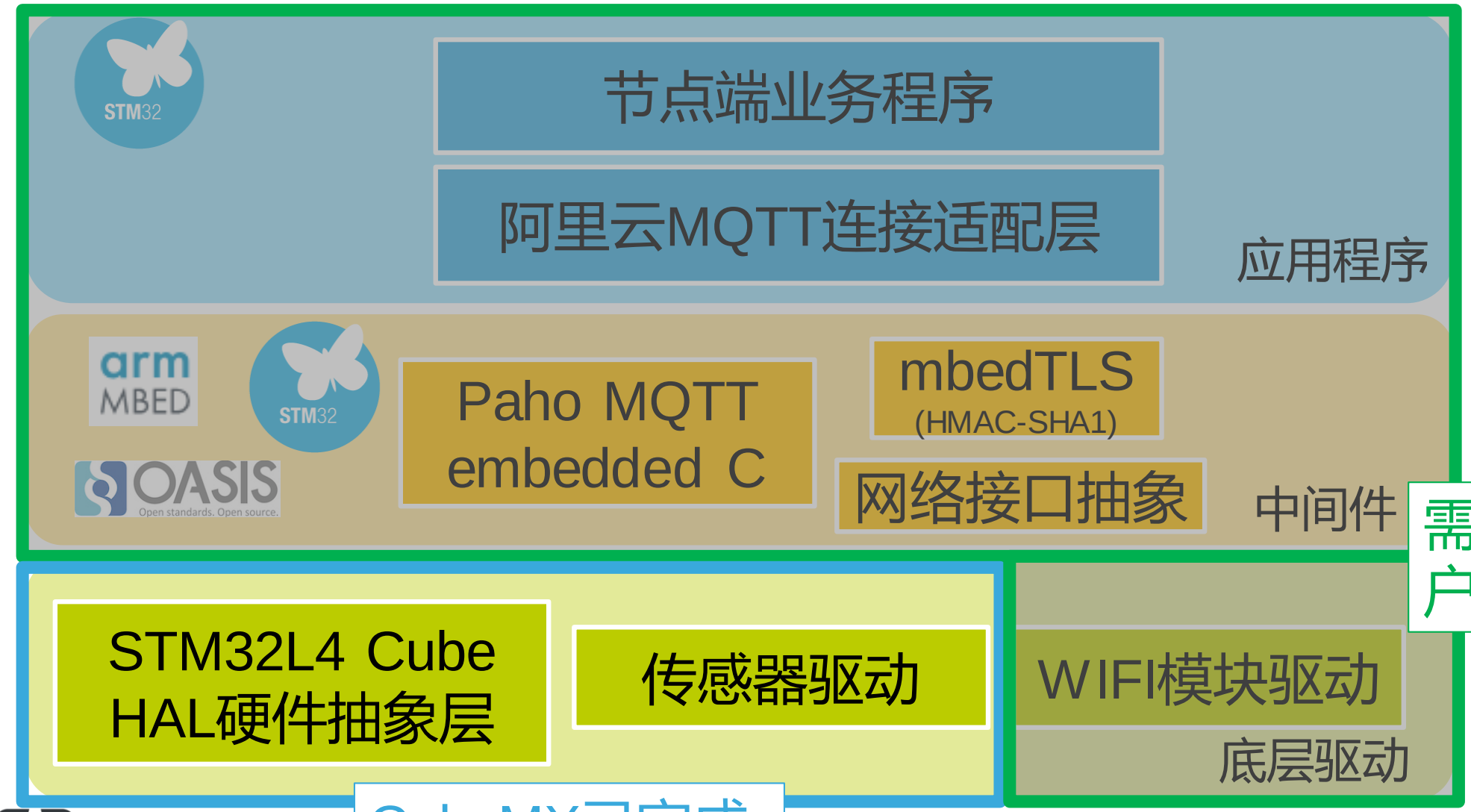


- 指定link文件

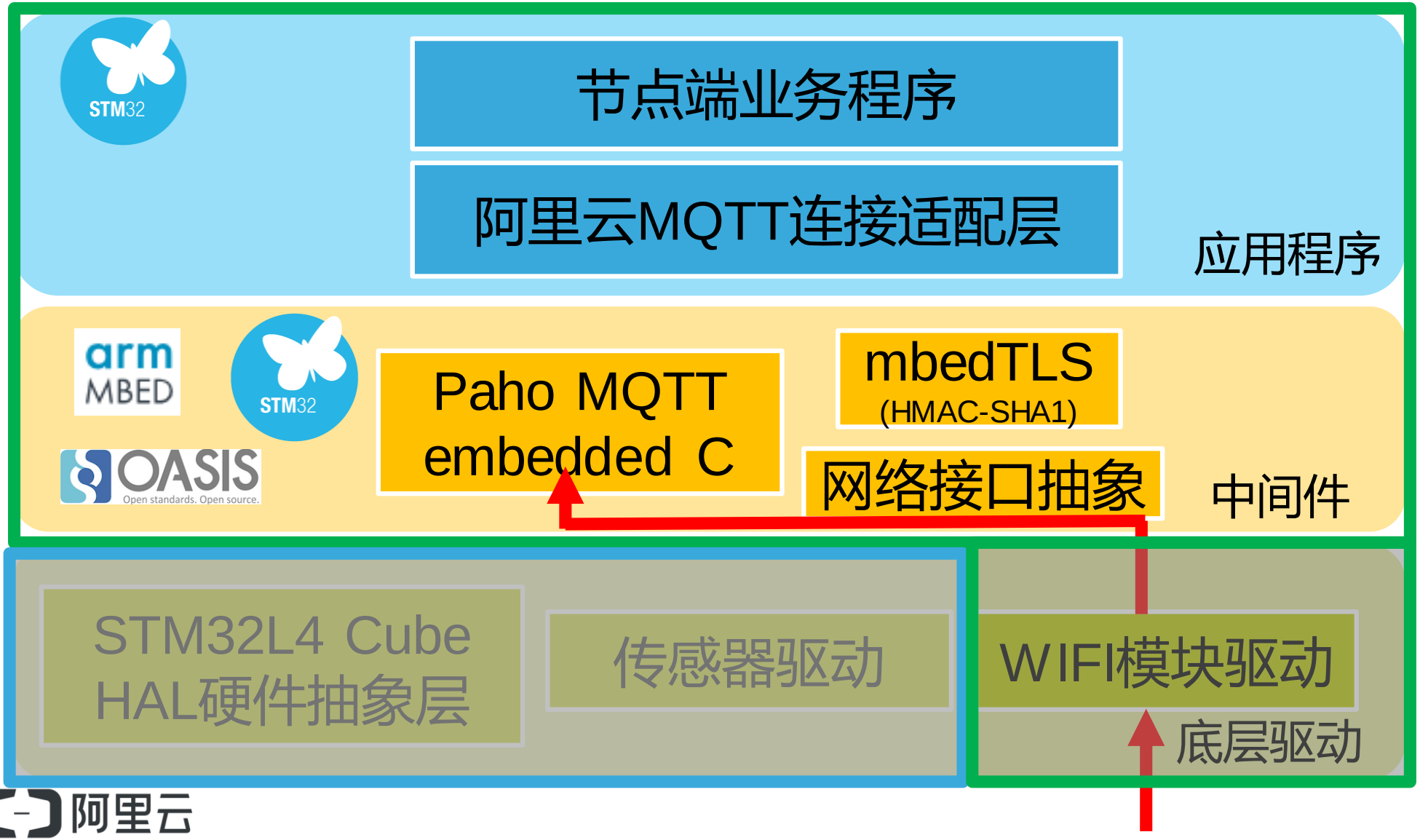


life.augmented
阿里云

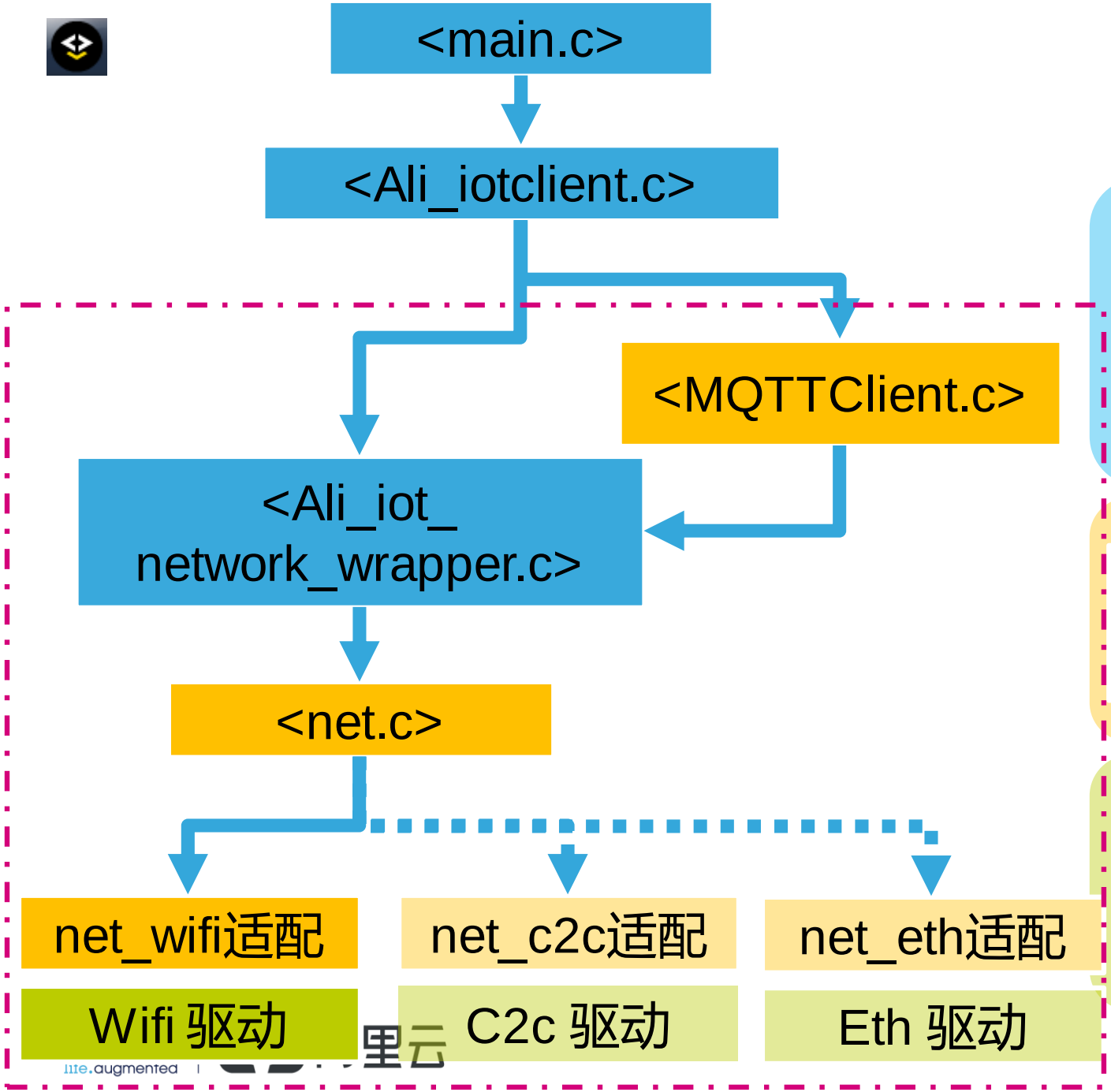
项目例程软件架构


节点端业务程序
阿里云MQTT连接适配层
应用程序
Paho MQTT
embedded C
mbedTLS
(HMAC-SHA1)
网络接口抽象
中间件
需要由用户添加…
STM32L4 Cube
HAL硬件抽象层
传感器驱动
WIFI模块驱动
底层驱动
CubeMX已完成

项目例程软件架构


节点端业务程序
阿里云MQTT连接适配层
应用程序
Paho MQTT
embedded C
mbedTLS
(HMAC-SHA1)
网络接口抽象
中间件
STM32L4 Cube
HAL硬件抽象层
传感器驱动
WIFI模块驱动
底层驱动

项目例程软件架构



节点端业务程序

阿里云MQTT连接适配层

Paho MQTT
embedded C

mbedTLS
(HMAC-SHA1)

网络接口抽象

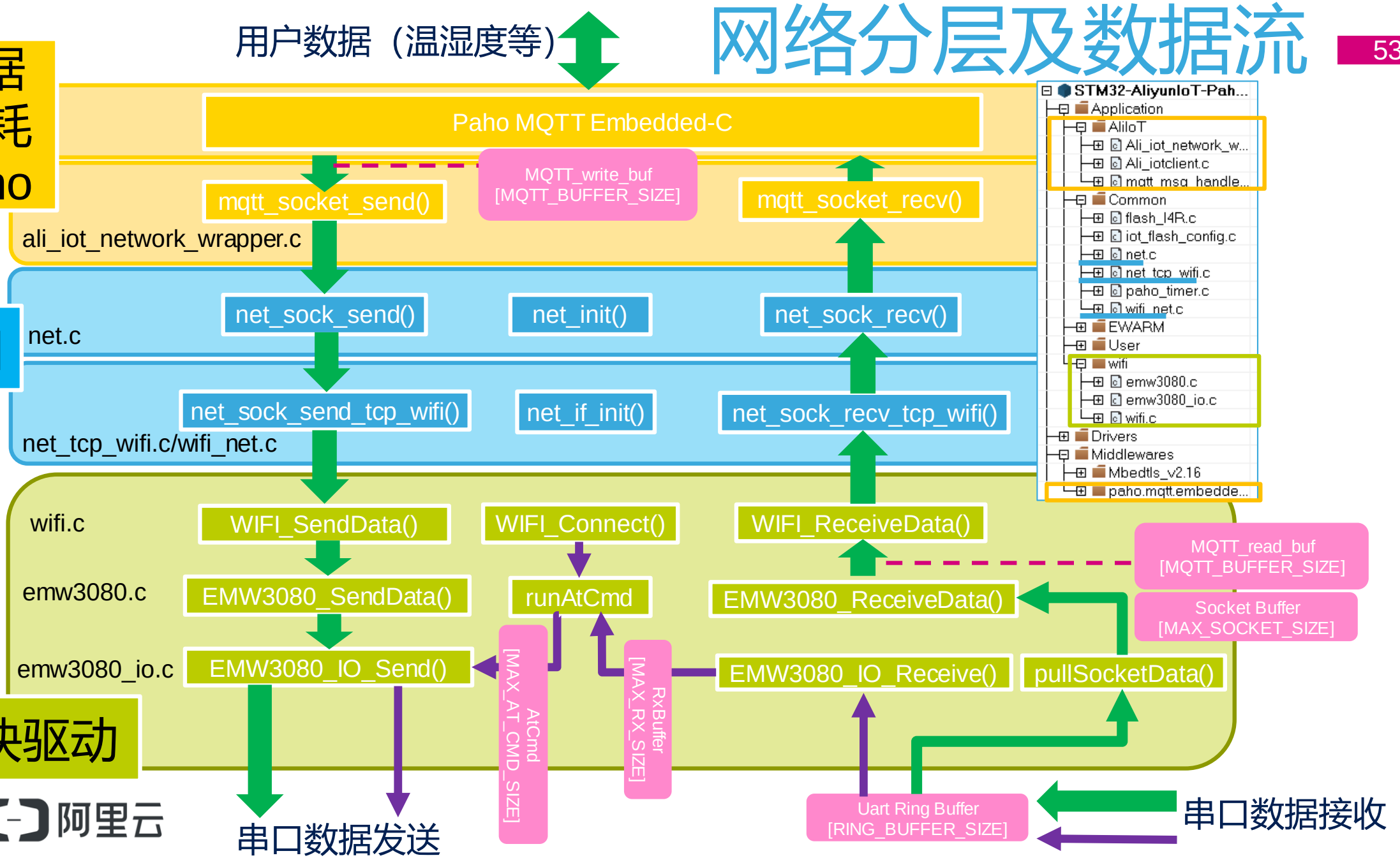
WIFI模块驱动

网络分层及数据流

网络数据产生/消耗层：Paho

网络接口

WIFI模块驱动



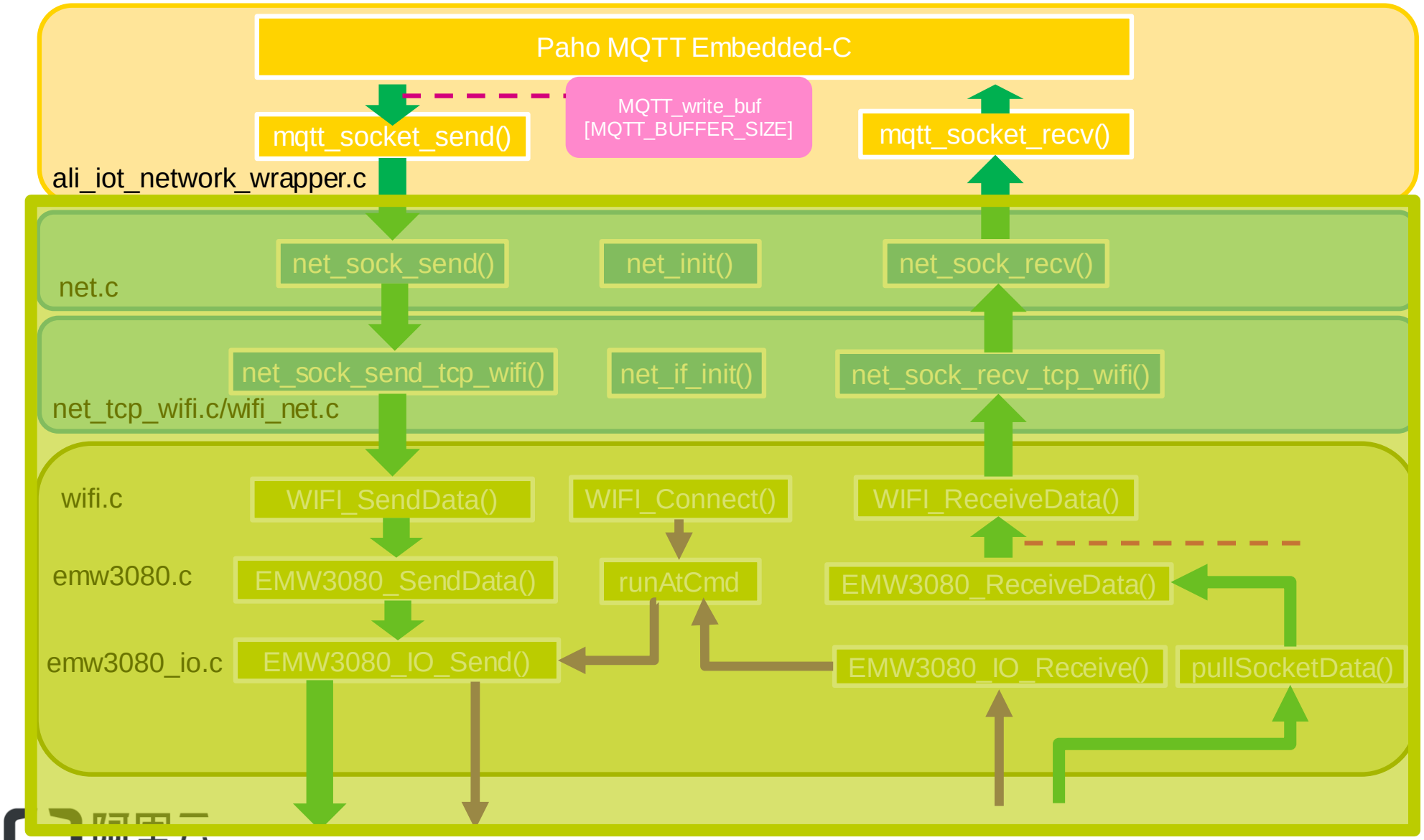
life.augmented

阿里云

驱动和应用层数组定义

Buffer名称	默认大小	定义所在位置	注释
MQTT_write_buf	MQTT_BUFFER_SIZE (512字节)	Ali_iotclient.c	Paho ==MQTTPublish()==> MQTT_write_buf 串口驱动 <==HAL_UART_Transmit_IT()== MQTT_write_buf
MQTT_read_buf	MQTT_BUFFER_SIZE (512字节)	Ali_iotclient.c	WIF驱动 ==EMW3080_ReceiveData()==> MQTT_read_buf Paho <==readPacket()== MQTT_read_buf
AtCmd	MAX_AT_CMD_SIZE(256字节)	emw3080.c	AT指令函数 ==EMW3080_JoinAccessPoint()==> AtCmd 串口驱动 <==HAL_UART_Transmit_IT()== AtCmd
RxBuffer	MAX_RX_SIZE(1500字节)	emw3080.c	WIFI驱动中用到，暂时保存从串口ringbuffer中取出的数据，然后判断AT指令返回值
WiFiRxBuffer	RING_BUFFER_SIZE(1024字节)	emw3080_io.h	串口中断 ==UART_RxISR_8BIT()==> WiFiRxBuffer WIF驱动 <==EMW3080_IO_Receive()== WiFiRxBuffer
“sock buffer”	MAX_SOCKET_SIZE (512字节)	emw3080.h WIFI_OpenClientConnection 函数中分配内存	WIF驱动 ==pullSocketData()==> ”sock buffer” WIF驱动 <==EMW3080_ReceiveData()== “sock buffer”

网络分层及数据流


Paho MQTT Embedded-C
MQTT_write_buf
[MQTT_BUFFER_SIZE]
mqtt_socket_send()
mqtt_socket_recv()
ali_iot_network_wrapper.c
net_sock_send()
net_init()
net_sock_recv()
net.c
net_sock_send_tcp_wifi()
net_if_init()
net_sock_recv_tcp_wifi()
net_tcp_wifi.c/wifi_net.c
wifi.c
WIFI_SendData()
WIFI_Connect()
WIFI_ReceiveData()
emw3080.c
EMW3080_SendData()
runAtCmd
EMW3080_ReceiveData()
emw3080_io.c
EMW3080_IO_Send()
EMW3080_IO_Receive()
pullSocketData()

Paho MQTT客户端对下(网络连接)的适配

- 网络接口的适配

```
typedef struct Network Network;
struct Network
{
  net_sockhnd_t my_socket;
  int (*mqttread) (Network*, unsigned char*,int,int);
  int (*mqttwrite) (Network*,unsigned char*,int,int);
  int (*disconnect) (Network*);
};
```

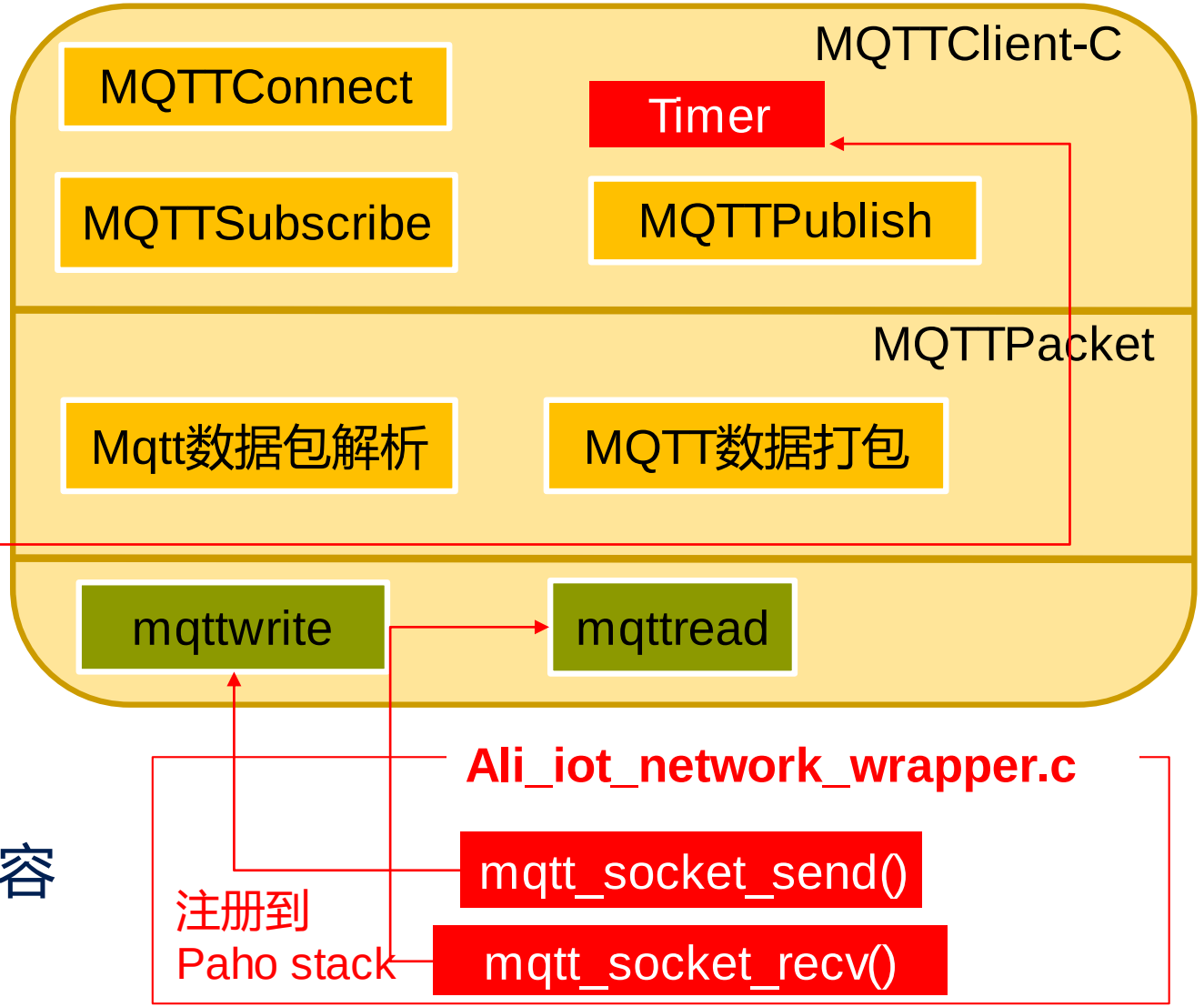
- Timer的适配

Paho_timer.c/h

同名函数体的实现

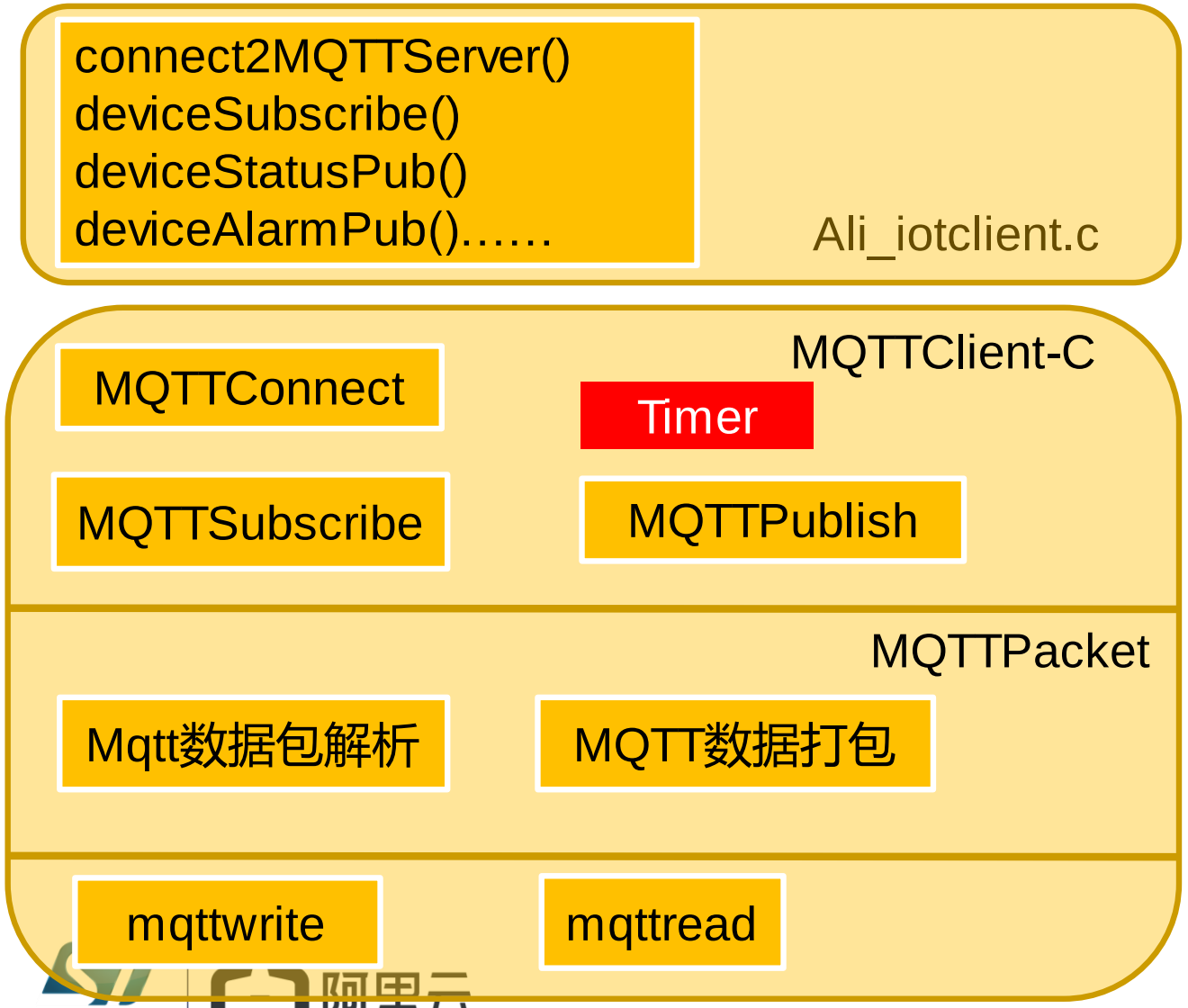
```
struct Timer {
  uint32_t init_tick;
  uint32_t timeout_ms;
};

typedef struct Timer Timer;

void TimerCountdownMS(Timer* timer, unsigned int timeout_ms);
void TimerCountdown(Timer* timer, unsigned int timeout);
int TimerLeftMS(Timer* timer);
char TimerIsExpired(Timer* timer);
void TimerInit(Timer* timer);
```

- returnCode枚举与ST HAL库的不兼容



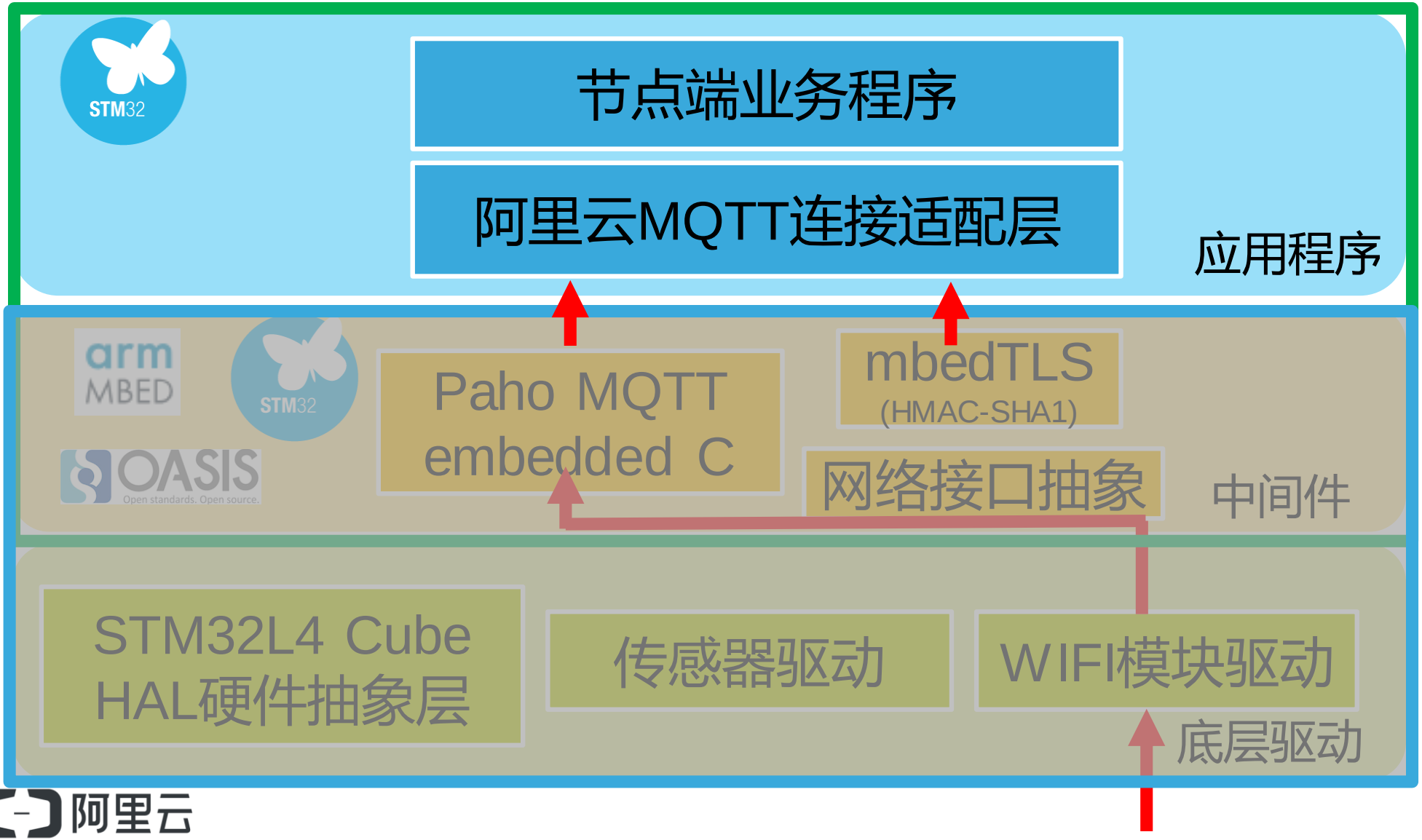
life.augmented 阿里云

Paho MQTT Client的调用



- MQTTClientInit
  - 初始化MQTT客户端
- MQTTConnect
  - 与服务器建立MQTT连接
- MQTTSubscribe
  - 向服务器订阅消息主题，并注册收到消息后的回调函数
- MQTTPublish
  - 向服务器发布某个主题的消息
- MQTTYield
  - 根据应用调整周期调用的间隔

项目例程软件架构


节点端业务程序
阿里云MQTT连接适配层
应用程序
Paho MQTT
embedded C
mbedTLS
(HMAC-SHA1)
网络接口抽象
中间件
STM32L4 Cube
HAL硬件抽象层
传感器驱动
WIFI模块驱动
底层驱动

Paho MQTT客户端对上(阿里云IoT)的适配

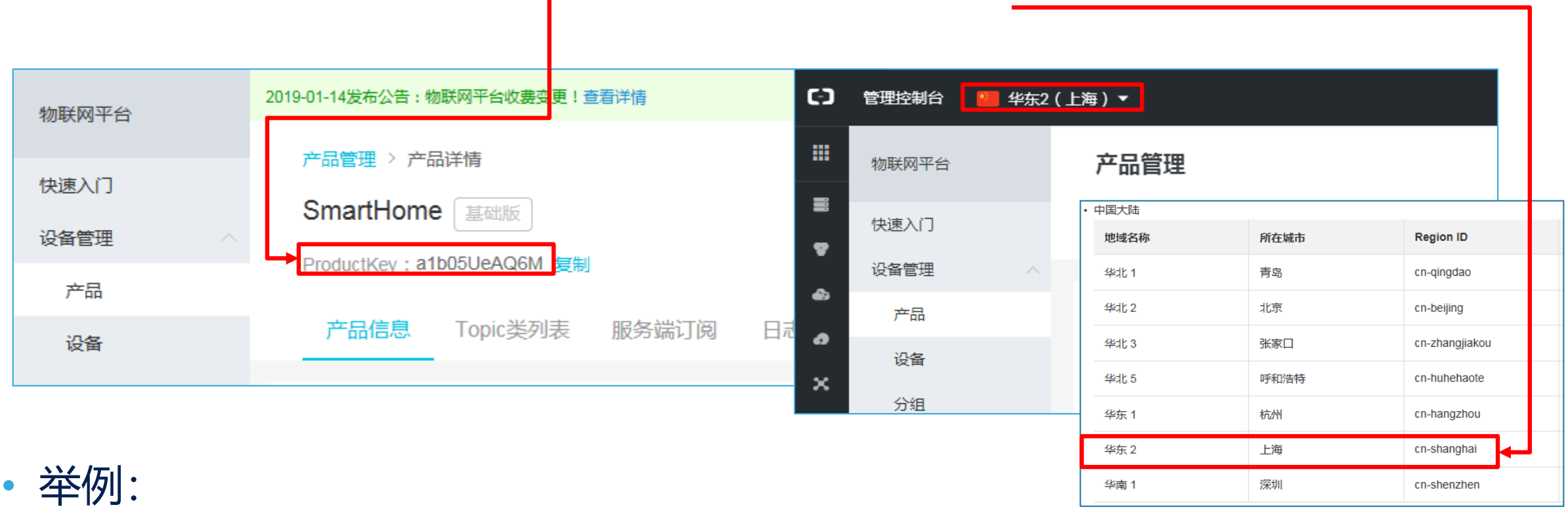
- 与阿里云MQTT服务器连接需要的参数
  - 用户名/密码
  - MQTT ClientID
  - 保活时间
  - Cleansession
  - MQTT服务器域名

	Description
Byte 1	Message Type
Byte 2	Remaining Length
Byte 3~8 (Protocol Name)	Length MSB(0)
	Length LSB(4)
	"M"
	"Q"
	"T"
	"T"
Byte9	Protocol Version(4)
Byte10	Connect Flag
Byte11	Keep Alive MSB
Byte12	Keep Alive LSB
Byte13 ~ Byte n	Client Identifier
	Will Topic
	Will Message
	Username
	Password

```
typedef struct
{
        /** The eyecatcher for this structure.  must be MQTC. */
        char struct_id[4];
        /** The version number of this structure.  Must be 0 */
        int struct_version;
        /** Version of MQTT to be used.  3 = 3.1 4 = 3.1.1
          */
        unsigned char MQTTVersion;
        MQTTString clientID;
        unsigned short keepAliveInterval;
        unsigned char cleansession;
        unsigned char willFlag;
        MQTTPacket_willOptions will;
        MQTTString username;
        MQTTString password;
} MQTTPacket_connectData;
```

构建MQTT服务器域名

- MQTT服务器域名： ${YourProductKey}.iot-as-mqtt.${YourRegionId}.aliyuncs.com:1883



- 举例：
  - 服务器域名：a1b05UeAQ6M.iot-as-mqtt.shanghai-cn.aliyuncs.com
  - 端口：1883

life.augmented 阿里云

构建MQTT ClientID

- MQTT ClientID：
  clientId+“|securemode=3,signmethod=hmacsha1,timestamp=132323232|”

客户端自己定义的ID号。可以使用MAC地址

安全模式，可选2（TLS直连）和3（TCP直连）

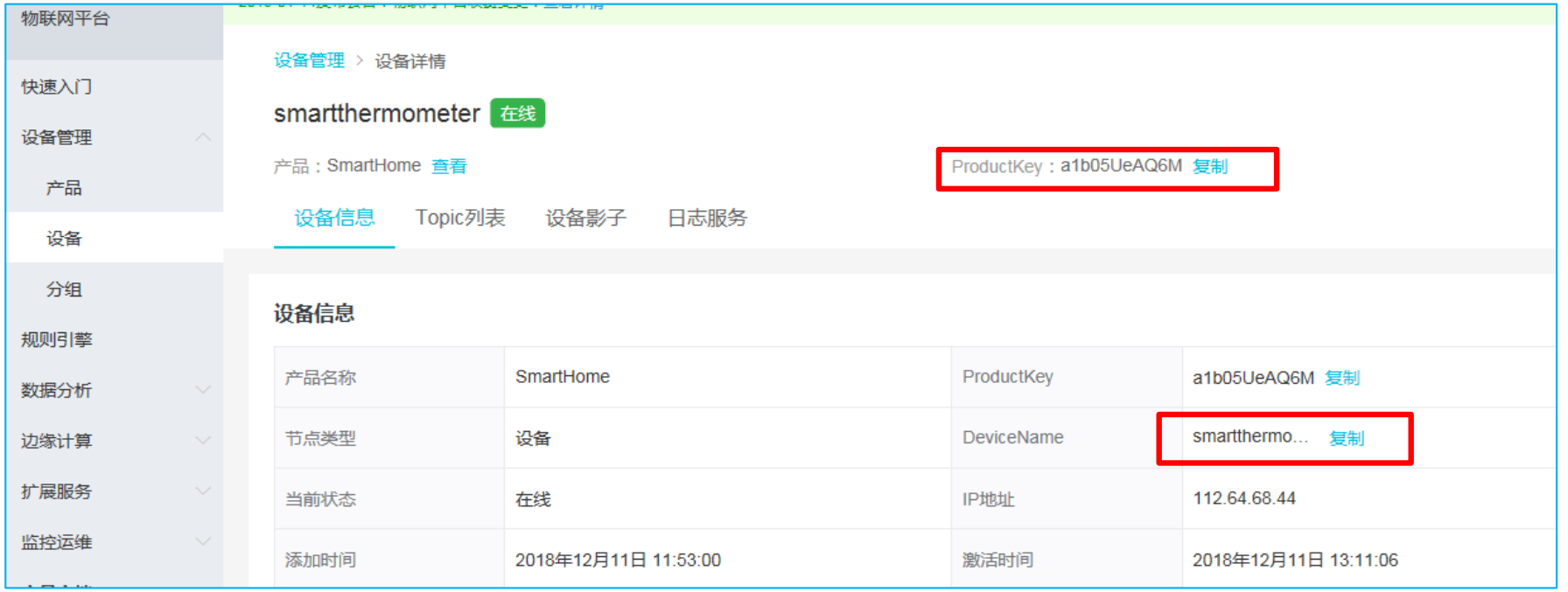
签名算法支持：
hmacmd5
hmacsha1
hmacsha256

当前的时间戳。可以通过HAL_GetTick获取当前时间戳。

- 如果clientId为“b0f8933b9467”，签名算法选择hmacsha1,当前时间戳为24081，则MQTT Client ID为：“b0f8933b9467|securemode=3,signmethod=hmacsha1,timestamp=24081|”

构建MQTT用户名

- MQTT 用户名： DeviceName+"&"+ProductKey



- 举例：MQTT用户名就是“smartthermometer&a1b05UeAQ6M”

构建MQTT密码

MQTT 密码 = sign_hmac(**DeviceSecret**, **content**)

MAC地址： b0f8933b9467

DeviceName：
smartthermometer

ProductKey：
a1b05UeAQ6M

timestamp： 24081

clientId${clientId}deviceName${YourDeviceName}productKey${YourProductKey}timestamp${timestamp}

拼接成content

clientIdb0ff8933b9467deviceNamesmartthermometerproductKeya1b05UeAQ6Mtimestamp24081

DeviceSecret：
7o7GJ3odUE7pPnie0
7dzlXDKDZTzTQVe

key

hmacsha1

digest

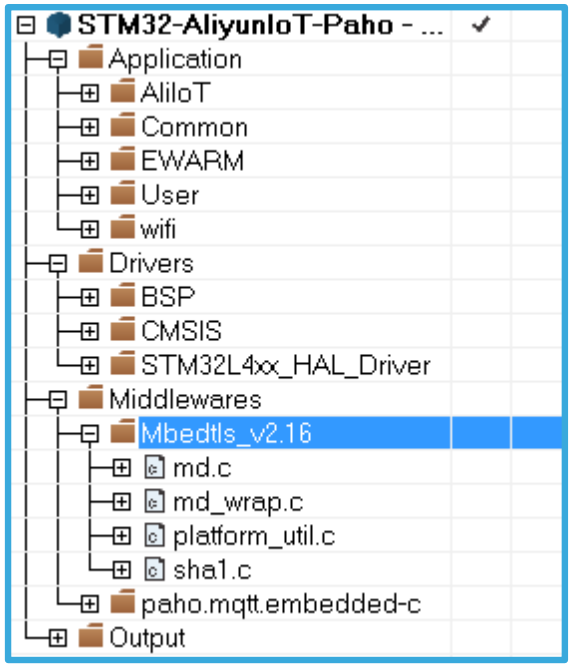
8DA3F48A54A816D01C462B07CD3BEF1E5A87B356

MQTT密码

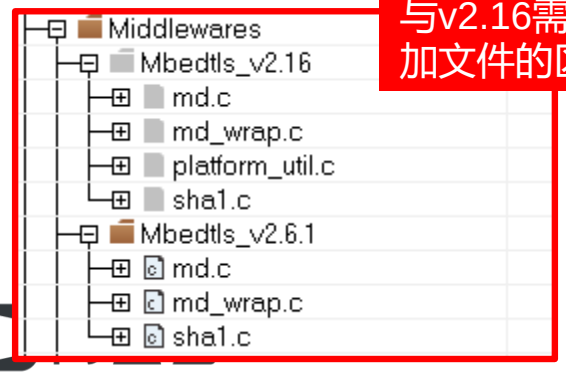
Mbedtls

Download mbed TLS

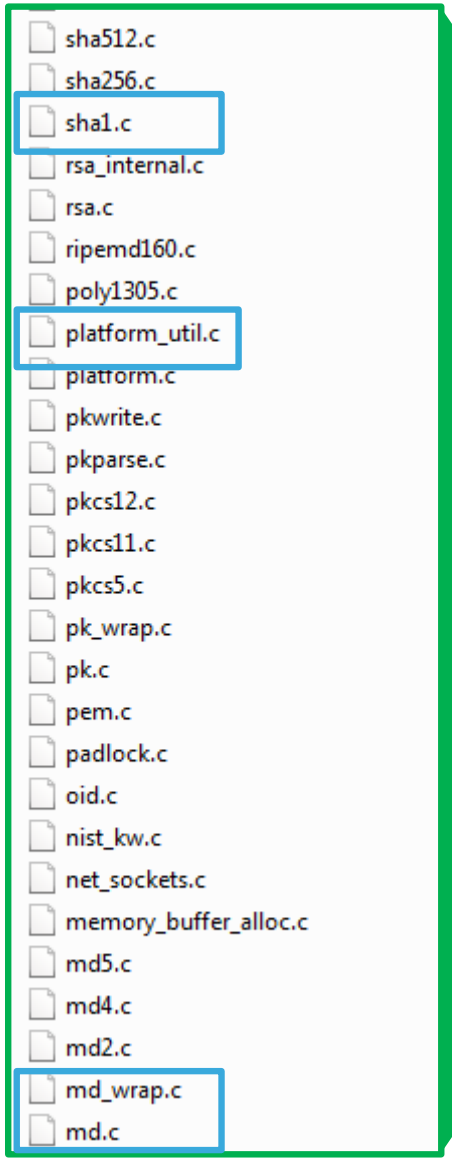
Demo IAR工程



HMACSHA1



mbedtlsV2.6.1与v2.16需要添加文件的区别



Hash

AES

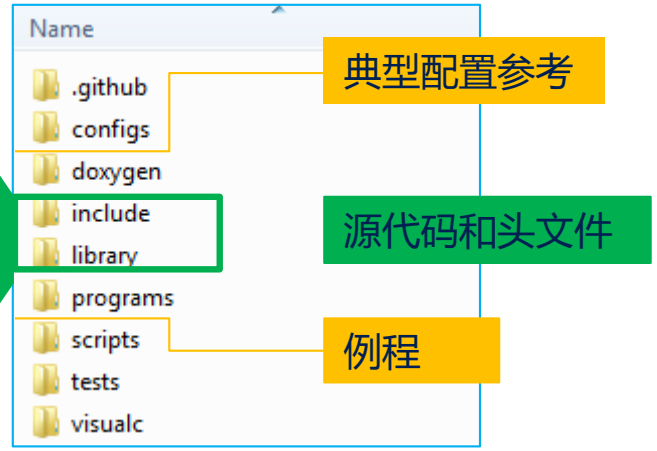
RSA

Elliptic Curve

SSL/TLS

Random Number

Mbedtls协议包



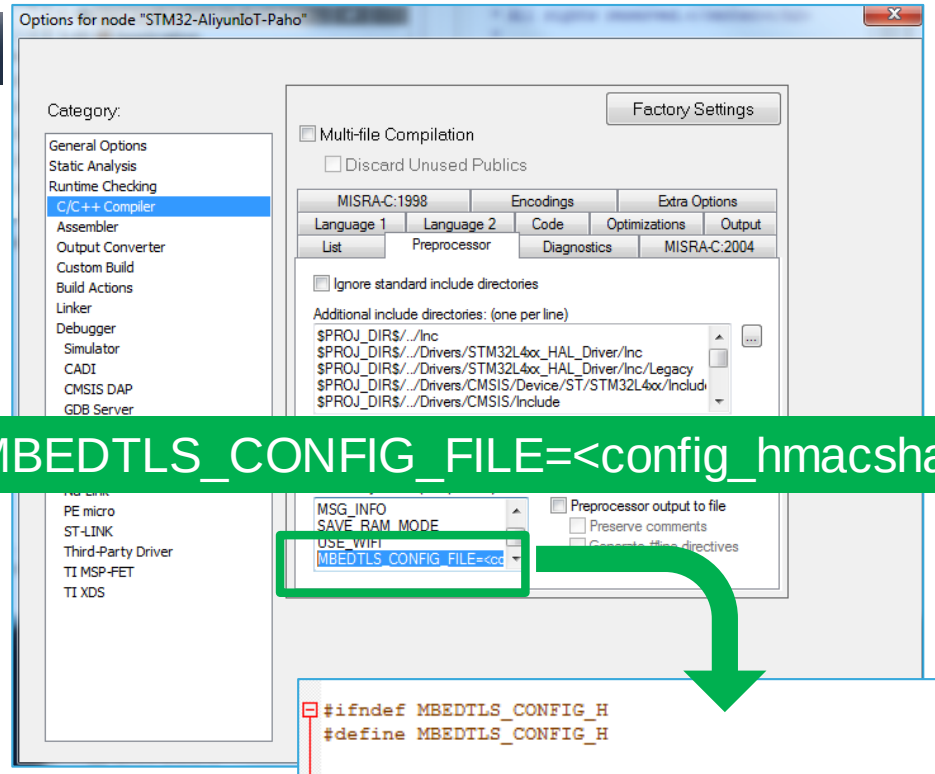
典型配置参考

源代码和头文件

例程

life.augmented

使用Mbedtls计算签名



MBEDTLS_CONFIG_FILE=<config_hmacsha1.h>

```
#ifndef MBEDTLS_CONFIG_H
#define MBEDTLS_CONFIG_H

/* System support */
#define MBEDTLS_HAVE_ASM
#define MBEDTLS_HAVE_TIME

/* mbed TLS feature support */
// #define MBEDTLS_CIPHER_MODE_CBC
// #define MBEDTLS_PKCS1_V15
// #define MBEDTLS_KEY_EXCHANGE_RSA_ENABLED
// #define MBEDTLS_SSL_PROTO_TLS1_1

/* mbed TLS modules */
#define MBEDTLS_MD_C
#define MBEDTLS_SHA1_C

/* For test certificates */
#define MBEDTLS_BASE64_C
#define MBEDTLS_CERTS_C
#define MBEDTLS_PEM_PARSE_C

/* For testing with compat.sh */
#define MBEDTLS_FS_IO

#include "mbedtls/check_config.h"

#endif /* MBEDTLS_CO
```

config_hmacsha1.h

Ali_iotclient.c

Mbedtls_SHA1_HMAC_Compute(uint8_t *, uint32_t, uint8_t *, uint32_t, uint8_t *, int32_t *)

```
421 int32_t Mbedtls_SHA1_HMAC_Compute(uint8_t* InputMessage,
422                                    uint32_t InputMessageLength,
423                                    uint8_t *HMAC_key,
424                                    uint32_t HMAC_keyLength,
425                                    uint8_t *MessageDigest,
426                                    int32_t* MessageDigestLength)
427 {
428   mbedtls_md_context_t sha1_ctx;
429   int32_t ret = 0;
430
431   mbedtls_md_init(&sha1_ctx);
432
433   ret = mbedtls_md_setup(&sha1_ctx,mbedtls_md_info_from_type(MBEDTLS_MD_SHA1),1)
434
435   /* check for initialization errors */
436   if (ret == 0)
437   {
438     /* Add data to be hashed */
439     mbedtls_md_hmac_starts(&sha1_ctx,HMAC_key,HMAC_keyLength);
440     mbedtls_md_hmac_update(&sha1_ctx,InputMessage,InputMessageLength);
441     mbedtls_md_hmac_finish(&sha1_ctx,MessageDigest);
442     *MessageDigestLength = strlen(MessageDigest);
443   }
444
445   return ret;
446 }
```

1 2 3

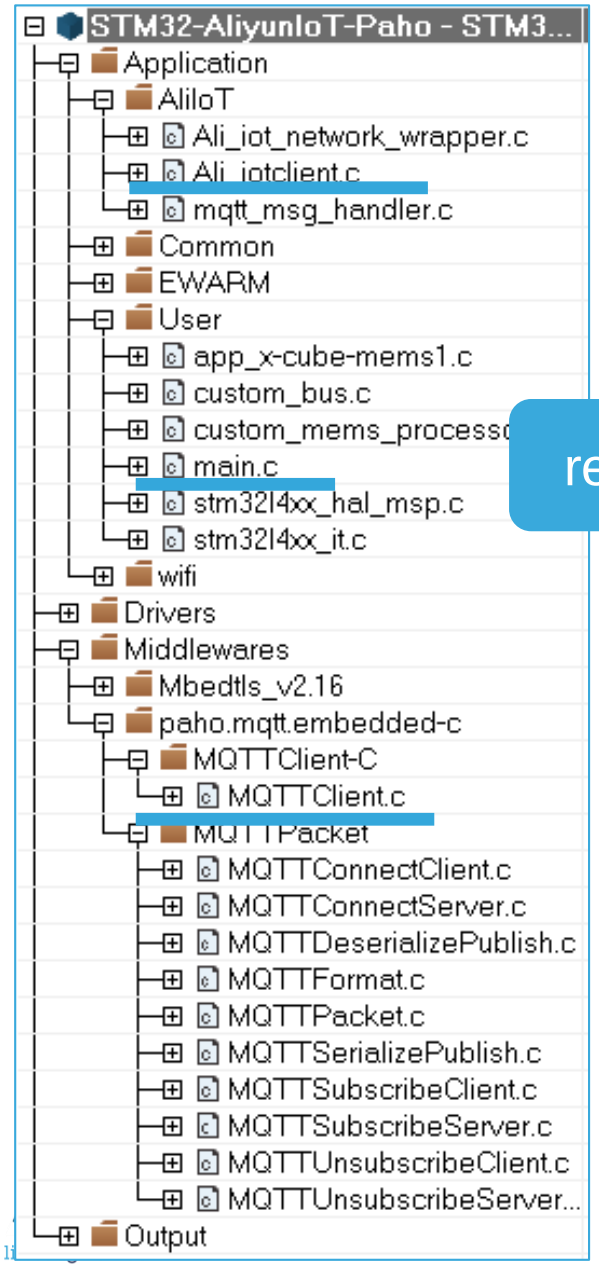
MHACSHA1计算

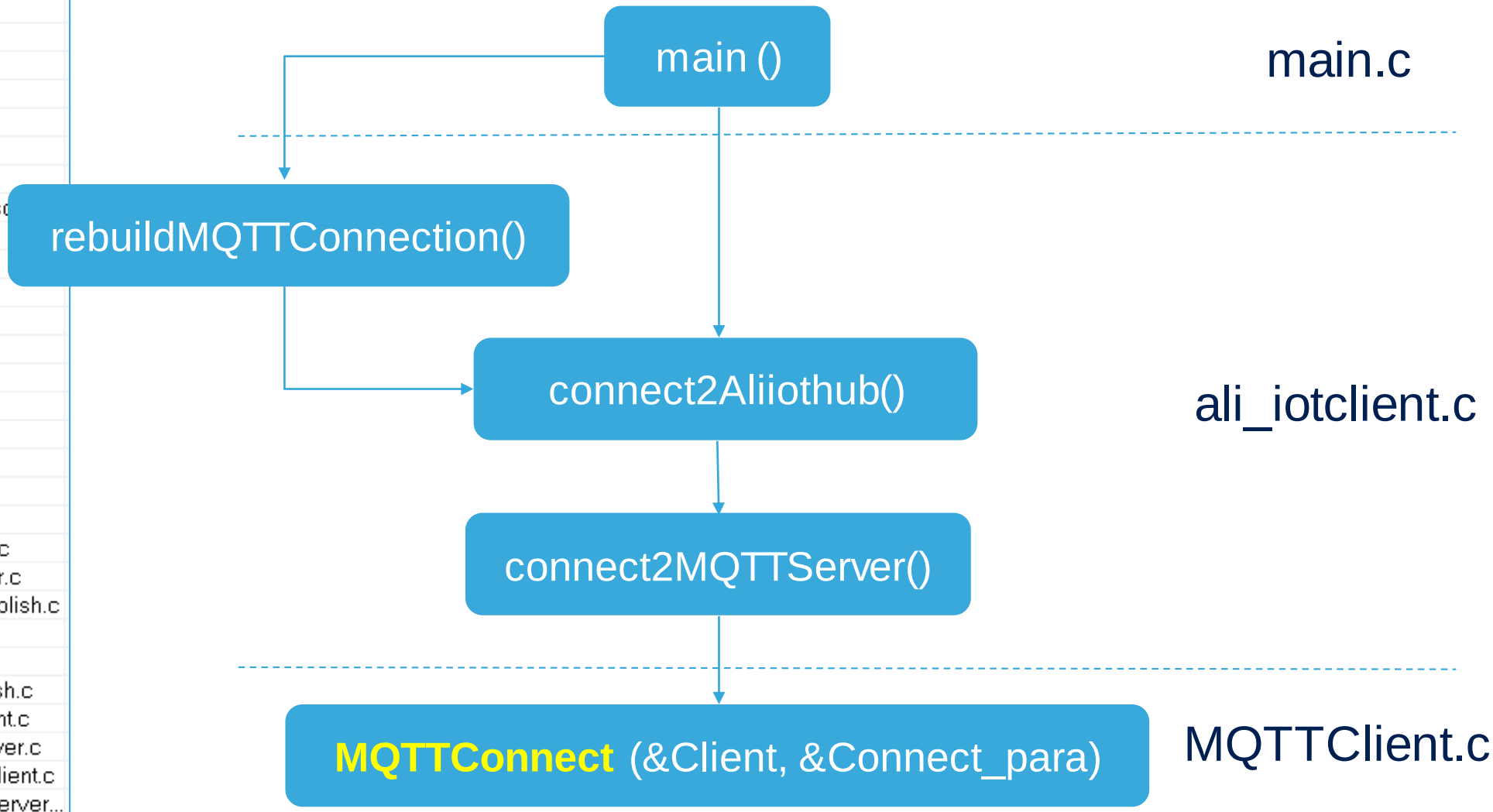
life.augmented

阿里云IoT平台连接参数小结

int MQTTConnect(MQTTClient* c, MQTTPacket_connectData* options)	
struct_id	MQTC
struct_version	
MQTTVersion	
clientID	clientId+“|securemode=3,signmethod=hmacsha1,timestamp=132323232|”
keepAliveInterval	
cleansession	
WillFlag	
will	
username	DeviceName+"&"+ProductKey
password	content = clientId${clientId}deviceName${YourDeviceName}productKey${YourProductKey}timestamp${timestamp} sign_hmac(**DeviceSecret**, **content**)
int mqtt_connect_network(Network* n, const char * host_address, int port)	
host_address	${YourProductKey}.iot-as-mqtt.${YourRegionId}.aliyuncs.com:1883

代码中的调用


STM32-AliyunIoT-Paho - STM3...
Application
AliIoT
Ali_iot_network_wrapper.c
Ali_iotclient.c
mqtt_msg_handler.c
Common
EWARM
User
app_x-cube-mems1.c
custom_bus.c
custom_mems_process
main.c
stm32l4xx_hal_msp.c
stm32l4xx_it.c
wifi
Drivers
Middlewares
Mbedtls_v2.16
paho.mqtt.embedded-c
MQTTClient-C
MQTTClient.c
MQTTPacket
MQTTConnectClient.c
MQTTConnectServer.c
MQTTDeserializePublish.c
MQTTFormat.c
MQTTPacket.c
MQTTSerializePublish.c
MQTTSubscribeClient.c
MQTTSubscribeServer.c
MQTTUnsubscribeClient.c
MQTTUnsubscribeServer...
Output


main ()
main.c
rebuildMQTTConnection()
connect2Aliiothub()
ali_iotclient.c
connect2MQTTServer()
MQTTConnect (&Client, &Connect_para)
MQTTClient.c

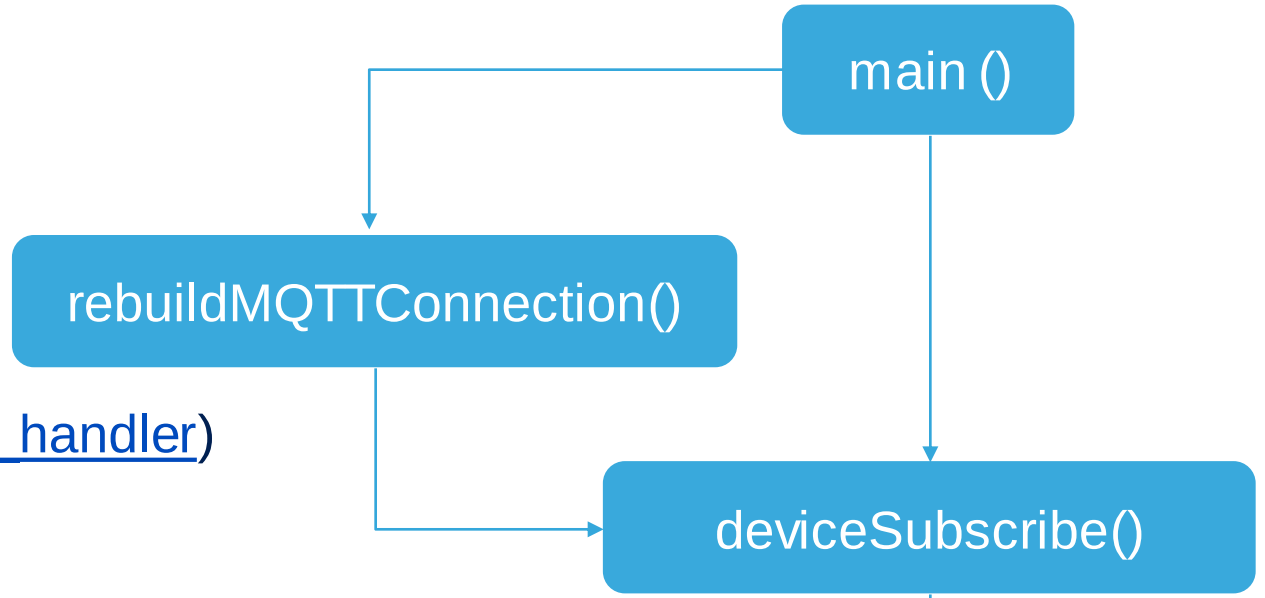
MQTT主题与消息负载格式

功能	MQTT主题	操作权限（设备）	消息方向	负载格式
设置温度阈值	${productKey}/${deviceName}/user/tempThresholdSet	订阅	下行	一个字节：温度阈值。直接二进制传输，例如：0x1E代表30℃
解除警报	${productKey}/${deviceName}/user/clearAlarm	订阅	下行	一个字节，固定为0x01
高温报警	${productKey}/${deviceName}/user/tempAlarm	发布	上行	一个字节，固定为0x01
上报属性	${productKey}/${deviceName}/user/tempHumUpload	发布	上行	Byte1 温度值 Byte2 湿度值 Byte3 温度阈值

MQTT订阅消息的回调函数

MQTTSubscribe
(
&Client,
threshold_topic,
QOS0,
Parameters_message_handler)

MQTTSubscribe
(
&Client,
clearAlarm_topic,
QOS0,
Service_message_handler)



main.c

ali_iotclient.c

MQTTClient.c

life.augmented 阿里云

Demo参数输入

- 需要保存在MCU闪存中的信息
  - WIFI配网参数（串口输入）
  - 阿里云IoT平台三元组信息（串口输入）
  - 温度报警阈值（云端下发）

```
typedef struct {
  uint64_t magic;                          /**< The USER_CONF_MAGIC magic word signals
  char ssid[USER_CONF_WIFI_SSID_MAX_LENGTH];  /**< Wifi network SSID. */
  char psk[USER_CONF_WIFI_PSK_MAX_LENGTH];    /**< Wifi network PSK. */
  uint8_t security_mode;                   /**< Wifi network security mode. See @ref wi
} wifi_config_t;

typedef struct {
  uint64_t magic;                          /**< The USER_CONF_MAGIC magic word signals
  char product_key[USER_CONF_PRODUCT_KEY_LENGTH];
  char device_name[USER_CONF_DEVICE_NAME_LENGTH];
  char device_secret[USER_CONF_DEVICE_SECRET_LENGTH];
  char region_id[USER_CONF_REGION_ID_LENGTH];
} iot_config_t;

typedef struct {
  uint64_t magic;                          /**< The USER_CONF_MAGIC magic word signals
  uint8_t temprature_threshold;
}app_config_t;
/** Static user configuration data which must survive reboot and firmware update. */
typedef struct {
  wifi_config_t wifi_config;
  iot_config_t iot_config;
  app_config_t app_config;
} user_config_t;
```

iot_flash_config.h

```
COM36 - Tera Term VT
File  Edit  Setup  Control  Window  Help
*** WIFI connection ***

Push the User button (Blue) within the next 5 seconds if you want to update the WiFi netw
ork configuration.

Your WiFi parameters need to be entered to proceed.

Enter SSID: amanda
You have entered amanda as the ssid.

Enter Security Mode (0 - Open, 1 - WEP, 2 - WPA, 3 - WPA2):3
You have entered 3 as the security mode.

Enter password: 12345678

Initializing the WiFi module
firmware version is : basic_AT_v2.1.2
OK

> WiFi module MAC address is: B0:F8:93:3B:94:67


Connecting to AP: amanda  Attempt 1/3 ...
Connected to AP amanda
Retrieving the IP address.
IP address: 192.168.43.70
Push the User button (Blue) within the next 5 seconds if you want to update the device se
curity parameters or credentials.

Enter Region ID: (example: cn-shanghai)
cn-shanghai
read: --->
cn-shanghai
<---

Enter Product Key: (example: a1b05Uexxxx)
a1b05Uxxxxx
read: --->
a1b05Uxxxxx
<---

Enter device name: (example: mydevicename)
SmartThermometer
read: --->
SmartThermometer
<---

Enter device secret: (example: 7o7GJ3odUE7pPnie07dzxxxxxxxxxxxx)
7o7GJ3odUE7pPnie07dzxxxxxxxxxxxx
read: --->
7o7GJ3odUE7pPnie07dzxxxxxxxxxxxx
<---
```

WIFI配网信息

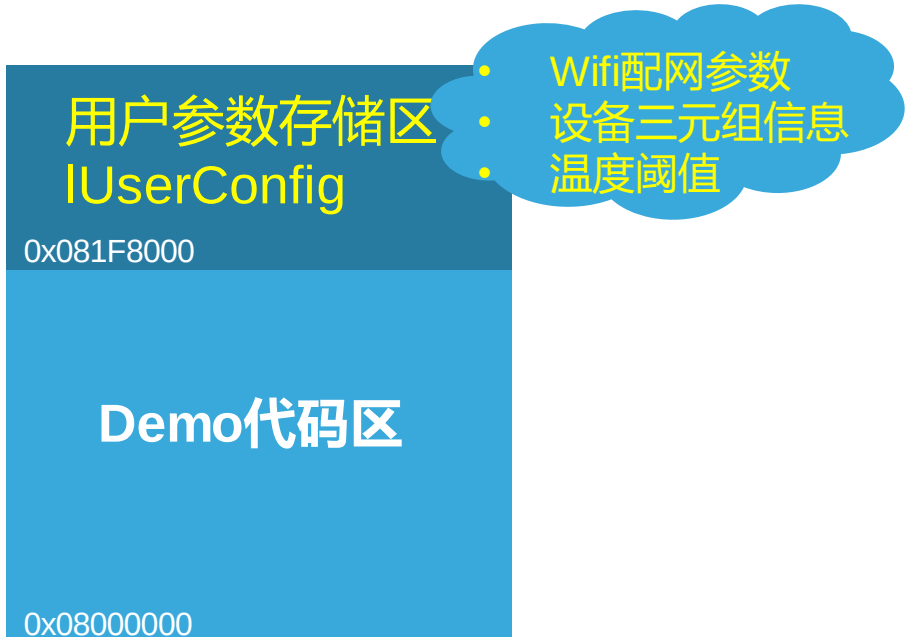
阿里云IoT平台区域ID及三元组

life.augmented

阿里云

Demo参数存储

- MCU用户闪存容量2M，页大小4K(双bank下)
- 取末尾32K作为用户参数存储区



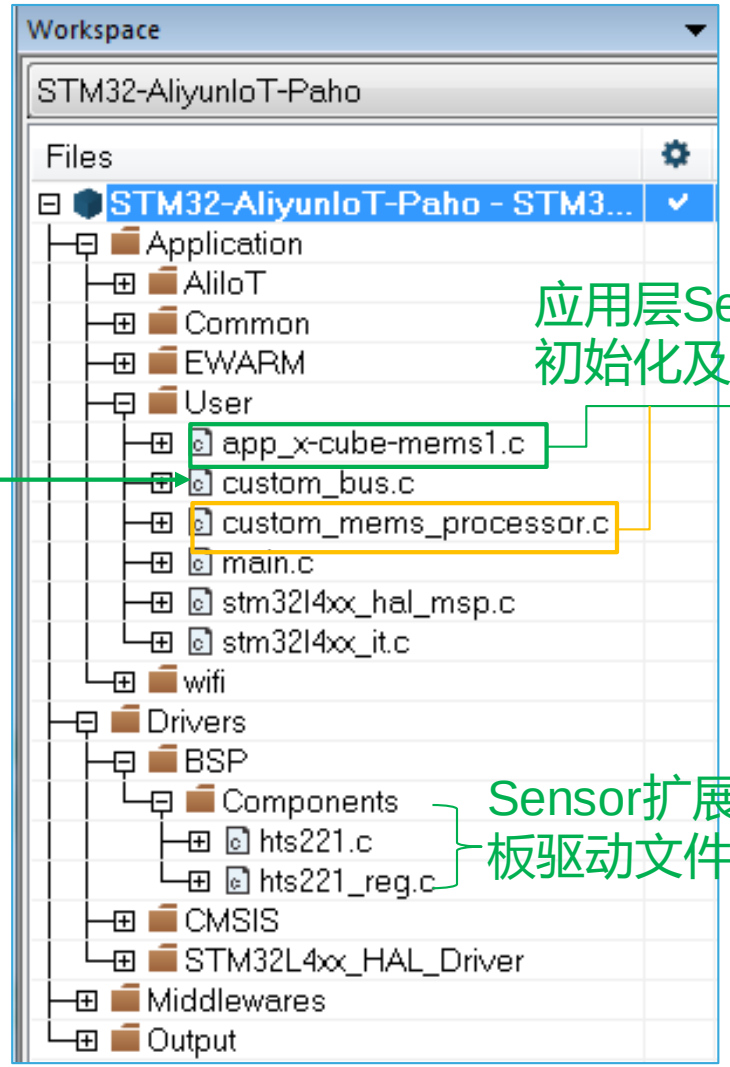
- link文件中定义该区域

```
define symbol __ICFEDIT_region_FIXED_LOC_start__ = 0x081F8000;
define region uninit_fixed_loc = mem:[from __ICFEDIT_region_FIXED_LOC_start__ size 32K];
```

- iot_flash_config.c将全局变量指定在此区域

```
#ifdef __ICCARM__  /* IAR */
__no_init const user_config_t lUserConfig @ "UNINIT_FIXED_LOC";
#elif defined ( __CC_ARM   )/* Keil / armcc */

#elif defined ( __GNUC__ )      /*GNU Compiler */

#endif
```

Sensor数据的读取（1）



I2C驱动

应用层Sensor板初始化及数据读取

Sensor扩展板驱动文件

Main函数

```
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_LPUART1_UART_Init();
MX_USART3_UART_Init();
MX_TIM15_Init();
MX_MEMS_Init();
/* USER CODE BEGIN 2 */
msg_info("\n");
msg_info("*****************************************************************************\n");
msg_info("***                    STM32 based AliIoT Client Demo                     ***\n");
msg_info("***                              Without TLS                              ***\n");
msg_info("***              FW version %d.%d.%d - %s, %s              ***\n",
         FW_VERSION_MAJOR, FW_VERSION_MINOR, FW_VERSION_PATCH, __DATE__, __TIME__);
msg_info("*****************************************************************************\n");

initApplication();

//WIFI SSID/passowrd config and initialization
//and config device certificate
ret = initPlatform();
if(ret!=0)
{
  msg_info("wifi initial failed!\n");
  while(1);
}
//connect to Ali Iot platform
ret = connect2Aliiothub();
if(ret!=0)
{
  msg_info("MQTT connection failed!\n");
  while(1);
}
//subscribe to topics
deviceSubscribe();
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
  //MQTT reconnection check
  rebuildMQTTConnection();
  doMqttYield();
  /* USER CODE END WHILE */
  MX_MEMS_Process();
  /* USER CODE BEGIN 3 */
    //update device temperature status
```

sensor扩展板初始化

sensor数据读取及处理

life.augmented

阿里云

Sensor数据的读取（2）

app_x-cube-mems1.c

```
void MX_MEMS_Init(void)
{
  /* USER CODE BEGIN SV */

  /* USER CODE END SV */

  /* USER CODE BEGIN MEMS_Library_Init_PreTreatment */

  /* USER CODE END MEMS_Library_Init_PreTreatment */

  /* Initialize the peripherals and the MEMS components */

  /* USER CODE BEGIN SV */
  Sensor_Init();
  /* USER CODE END SV */

  /* USER CODE BEGIN MEMS_Library_Init_PostTreatment */

  /* USER CODE END MEMS_Library_Init_PostTreatment */
}
/*
 * LM background task
 */
void MX_MEMS_Process(void)
{
  /* USER CODE BEGIN MEMS_Library_Process */
  Temp_Sensor_Handler();
  Hum_Sensor_Handler();
  /* USER CODE END MEMS_Library_Process */
}

#ifdef __cplusplus
}
#endif
```

初始化sensor模块

```
/* Configure the environmental sensor driver */
io_ctx.BusType     = HTS221_I2C_BUS; /* I2C */
io_ctx.Address     = HTS221_I2C_ADDRESS;
io_ctx.Init        = BSP_I2C1_Init;
io_ctx.DeInit      = BSP_I2C1_DeInit;
io_ctx.ReadReg     = BSP_I2C1_ReadReg;
io_ctx.WriteReg    = BSP_I2C1_WriteReg;
io_ctx.GetTick     = BSP_GetTick;

if (HTS221_RegisterBusIO(&hts221_obj_0, &io_ctx) != HTS221_OK)
{
  ret = BSP_ERROR_UNKNOWN_COMPONENT;
}
else if (HTS221_ReadID(&hts221_obj_0, &id) != HTS221_OK)
{
  ret = BSP_ERROR_UNKNOWN_COMPONENT;
}
else if (id != HTS221_ID)
{
  ret = BSP_ERROR_UNKNOWN_COMPONENT;
}
else
{
  HTS221_GetCapabilities(&hts221_obj_0, &cap);
  /* Initialize the HTS221 sensor */
  if(HTS221_COMMON_Driver.Init(&hts221_obj_0)!= HTS221_OK)
    ret = BSP_ERROR_COMPONENT_FAILURE;
  else
    ret = BSP_ERROR_NONE;

  if(cap.Temperature==1)
  {
    if(HTS221_TEMP_Driver.Enable(&hts221_obj_0)!=HTS221_OK)
      ret = BSP_ERROR_COMPONENT_FAILURE;
  }
  if(cap.Humidity==1)
  {
    if(HTS221_HUM_Driver.Enable(&hts221_obj_0)!=HTS221_OK)
      ret = BSP_ERROR_COMPONENT_FAILURE;
  }
}
```

注册传感器模块读写操作函数

初始化传感器模块
使能温度和湿度传感器

Custom_mems_processor.c

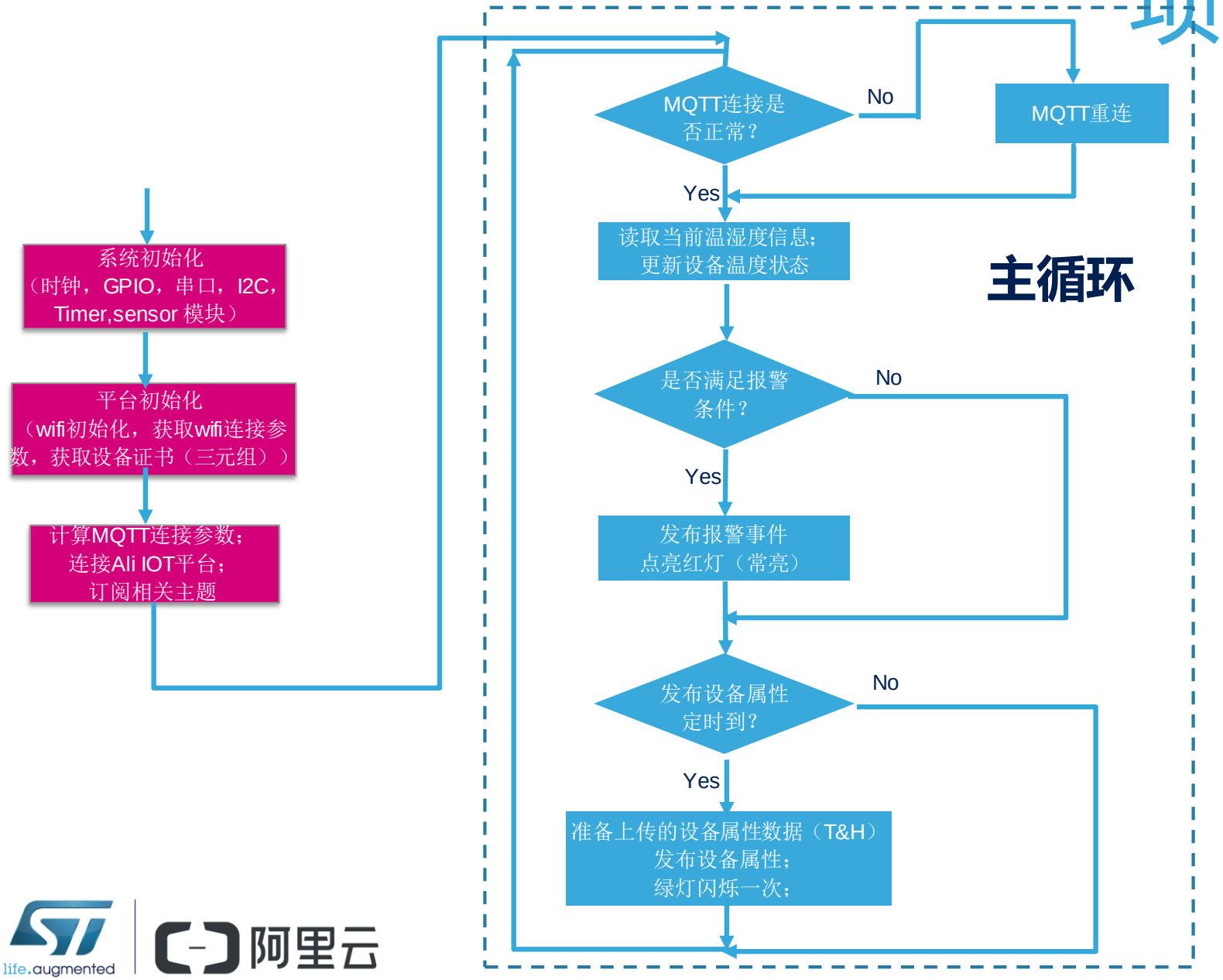
```
void Temp_Sensor_Handler(void)
{
  float temperature;

  if((HTS221_TEMP_Driver.GetTemperature(&hts221_obj_0, &temperature))==0)
  {
    SetTemperatureValue((int8_t)temperature);
  }
}

/**
  * @brief  Handles the humidity data getting/sending
  * @param  Instance the device instance
  * @retval None
  */
void Hum_Sensor_Handler(void)
{
  float humidity;

  if ((HTS221_HUM_Driver.GetHumidity(&hts221_obj_0, &humidity))==0)
  {
    SetHumValue((uint8_t)humidity);
  }

}
```

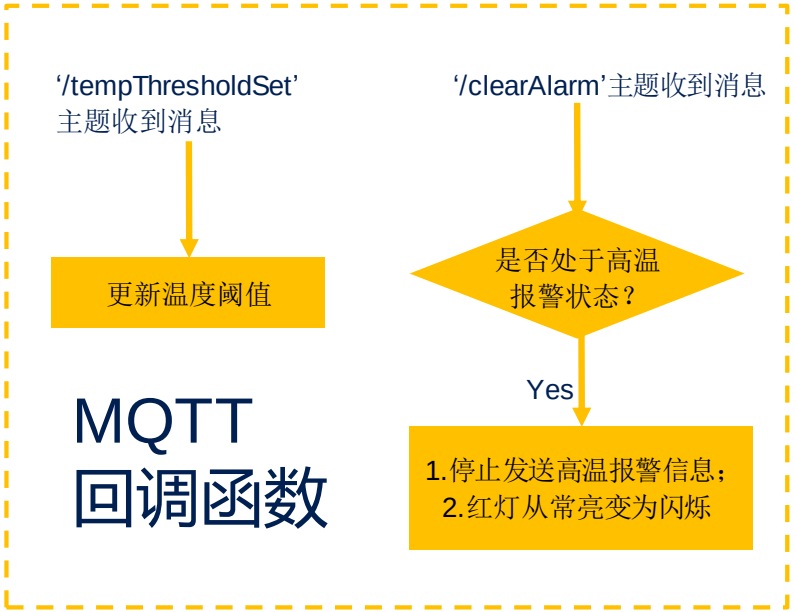
读温湿度信息

life.augmented

阿里云

项目例程流程图





life.augmented

阿里云

项目例程内存占用

- 总内存占用（使用IAR v8.32.3，最高优化等级）
  - Flash：52924字节
  - Ram：10874字节(包括4KB堆栈)
- 主要模块内存占用

模块	Flash(字节)	Ram(字节)	
Wifi驱动	2846	2873	
传感器驱动	1970	176	
STM32L4 HAL	15828	40	
ST网络抽象层	1794		
Paho MQTT协议栈	3858	4	
HMAC计算	4717	4	
Ali Iot Client	3284	1948	

下节预告

- 第一节：基于STM32的节点端介绍
  - 硬件平台，软件开发环境
- 第二节：使用Paho MQTT客户端协议栈直连阿里云IoT平台
  - 适用于资源受限的节点设备
- 第三节：使用Linkkit C-SDK和TLS通过MQTT协议直连阿里云IoT平台
  - 适用于资源丰富的节点设备