Embedded Technoloy and Intelligent Systems 嵌入式技术与智能系统, 2024, 1(2), 51-56
Published Online October 2024 in Hans. https://www.hanspub.org/journal/etis
https://doi.org/10.12677/etis.2024.12006

Hans汉斯

OpenHarmony上利用Paho MQTT连接云平台

安皓楠

北京华清远见科技发展有限公司研发中心，北京

收稿日期：2024年6月28日；录用日期：2024年11月21日；发布日期：2024年11月29日

摘　要

在物联网设备与云端之间的通信中，MQTT作为一种轻量级的、基于发布-订阅模式的通信协议，具备了良好的适用性和灵活性，被广泛应用于物联网领域。在OpenHarmony的LiteOS内核上利用MQTT连接云平台是一项关键的技术任务，它涉及在轻量级操作系统上实现MQTT协议的客户端功能，并与云端平台进行稳定和高效的通信，因此需要选择合适的MQTT库，并进行有效的移植和优化，以保证在资源受限的环境下依然能够实现稳定可靠的通信连接。海思Hi3861芯片采用了LiteOS内核。文章探讨了在海思Hi3861芯片上移植和使用Paho MQTT库连接到华为云的实现过程和关键技术。文章首先介绍了MQTT的相关知识，然后详细讨论了嵌入式Paho MQTT库的内容，接着介绍Hi3861芯片相关功能及其移植Paho MQTT的方式，最后描述了使用移植好的程序连接华为云MQTT的步骤，包括设备鉴权方式和消息发布订阅的实现。实验结果验证了在Hi3861平台上使用Paho MQTT库连接到华为云的可行性和效果。文章的结尾探讨了项目未来的工作。

关键词

MQTT，云，物联网，Paho MQTT，鸿蒙

Using Paho MQTT to Connect to Cloud Platforms on OpenHarmony

Haonan An

Research and Development Center, Beijing Huaqingyuanjian Technology Development Co., Ltd., Beijing

Received: Jun. 28th, 2024; accepted: Nov. 21st, 2024; published: Nov. 29th, 2024

Abstract

In the communication between IoT devices and the cloud, MQTT, as a lightweight, publish-subscribe communication protocol, offers excellent applicability and flexibility, making it widely used in the IoT field. Utilizing MQTT to connect to cloud platforms on OpenHarmony's LiteOS kernel is a critical

文章引用: 安皓楠. OpenHarmony 上利用 Paho MQTT 连接云平台[J]. 嵌入式技术与智能系统, 2024, 1(2): 51-56.
DOI: 10.12677/etis.2024.12006

technical task. It involves implementing MQTT client functionality on a lightweight operating system and establishing stable and efficient communication with the cloud platform. Therefore, choosing an appropriate MQTT library and conducting effective porting and optimization are crucial to ensure reliable communication connections in resource-constrained environments. The Hisilicon Hi3861 chip utilizes the LiteOS kernel. This article discusses the process and key technologies of porting and using the Paho MQTT library to connect to Huawei Cloud on the Hisilicon Hi3861 chip. The article begins with an introduction to MQTT concepts, followed by a detailed discussion of the embedded Paho MQTT library. It then covers the features of the Hi3861 chip and how to port Paho MQTT to it. Finally, it describes the steps to connect to Huawei Cloud MQTT using the ported application, including device authentication methods and the implementation of message publishing and subscription. Experimental results validate the feasibility and effectiveness of using the Paho MQTT library to connect to Huawei Cloud on the Hi3861 platform. The article concludes with a discussion on future project directions.

Keywords

MQTT, Cloud, Internet of Things, Paho MQTT, OpenHarmony

1. 引言

近年来，物联网(Internet of Things, IoT)技术的快速发展改变了各行各业，让设备之间实现了无缝连接和智能互动。从传感器到智能家电，物联网设备需要与云服务进行高效通信，以传输数据、接收指令并实现远程管理。这一转变促使了专门针对物联网部署需求开发的专业物联网平台和通信协议，如 CoAP (Constrained Application Protocol)、AMQP (Advanced Message Queuing Protocol)、DDS (Data Distribution Service)、WebSocket 等。除上述协议外，MQTT (Message Queuing Telemetry Transport)因其适应资源受限设备、简单易用的设计、强大的消息传输能力和广泛的应用支持而成为物联网通信的常用选择。它不仅满足了各种物联网设备和应用程序之间实时通信的需求，还提供了灵活的部署和扩展选项，适应了不同规模和复杂度的物联网解决方案[1]。

Paho MQTT 是一个开源的 MQTT 库，由 Eclipse Paho 项目提供支持。它提供了多种编程语言的实现，包括 C、C++、Java、Python 等，旨在帮助开发者轻松地在各种设备和平台上实现 MQTT 通信协议。本文专注于将 Paho MQTT 客户端库移植到 OpenHarmony 项目中的 LiteOS 内核处理器上，如海思 Hi3861 芯片，实现与云平台的连接。在 OpenHarmony 项目的 LiteOS 内核处理器上集成 MQTT 不仅增强了设备与云端高效通信的能力，还为创建互联物联网生态系统的整体目标作出了贡献。

2. MQTT 协议

MQTT 协议是一种轻量级的、基于发布/订阅模式的消息传输协议，最早由 IBM 开发于 1999 年，用于传感器和施工机器之间的通信。随后，协议被开放，成为 OASIS (Organization for the Advancement of Structured Information Standards，结构化信息标准推动组织)标准，并在众多物联网应用中得到广泛应用。MQTT 协议在物联网领域起初设计用于传感器网络和设备间的低带宽、高效率通信。其轻量级和简单的发布/订阅模式使得设备能够节省能源和带宽，适用于智能家居、工业自动化和智能城市等多种场景。

2.1. MQTT 的架构

MQTT 的客户端是消息发布者或者消息订阅者，它们与 MQTT Broker 进行通信来实现数据的传输和接收[2]。客户端可以是各种设备或应用程序，包括传感器、嵌入式设备、移动应用等。MQTT 的客户端和 Broker 之间的关系如图 1 所示。

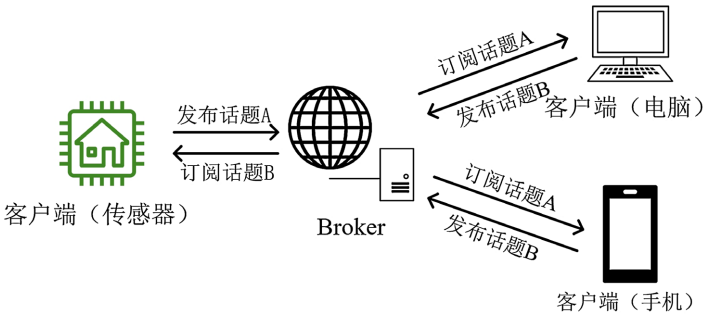

Figure 1. MQTT network architecture
图 1. MQTT 网络架构

客户端在 MQTT 中的主要功能包括：

发布消息：客户端可以向 Broker 发布(或发送)消息，消息可以是任何格式的数据，通常包含有用的传感器数据、控制命令等。

订阅主题：客户端可以向 Broker 订阅(或接收)特定的主题(Topic)，通过主题来过滤和接收感兴趣的消息。主题可以使用通配符来实现更灵活的订阅规则。

接收消息：客户端通过订阅主题，可以接收来自 Broker 转发的消息，并进行相应的处理和分析。

客户端可以选择不同的消息服务质量(QoS)级别来控制消息传递的可靠性和效率，包括至多传递一次(QoS 0)、至少传递一次(QoS 1)、恰好传递一次(QoS 2)。

MQTT 的 Broker 是中介服务器，负责管理客户端之间的消息传递。Broker 接收来自发布者(发布消息的客户端)的消息，并确保将这些消息按照订阅者(订阅消息的客户端)的需求正确分发。Broker 在 MQTT 协议中的关键作用包括：

消息路由和分发：根据订阅关系，将发布者发送的消息分发给所有订阅了相关主题的客户端。

连接管理：管理客户端的连接和状态，确保每个连接的可靠性和安全性。

QoS 管理：根据客户端设置的 QoS 级别，确保消息的按时传递和确认。

2.2. MQTT 的报文

为了让客户端和 Broker 之间进行通信，MQTT 协议定义了不同类型的消息(称为报文)，如 CONNECT、CONNACK、PUBLISH、PUBACK、SUBSCRIBE、SUBACK、UNSUBSCRIBE 等。报文包含固定报头、可变报头(部分报文包含)、负载(部分报文包含)。固定报头定义了报文类型、控制标志和后续数据长度，可变头部则依据报文类型规定了各自的详细信息，如后续消息内容、QoS 级别等。负载用于携带实际数据内容。这种设计使得 MQTT 能够在各种网络条件下高效传输消息，并支持灵活的通信需求和服务质量保证。

3. Paho MQTT 库

3.1. 简介

Paho 项目致力于提供开源的、可扩展的消息传递协议实现，支持 M2M (Machine-to-Machine)和物联

网的各种应用。它特别关注设备连接中的物理限制和成本问题。Paho MQTT C 是 Paho 项目的一部分，是专门为 C 语言开发的 MQTT 客户端库。Paho MQTT C 库旨在提供一个可靠、高效的实现，使开发者能够轻松地在 C 语言环境中实现 MQTT 的发布和订阅功能[3]。

3.2. 库文件内容

应用于嵌入式的 Paho MQTT C 库文件中，能够被移植的代码位于 MQTTPacket、MQTTClient-C、MQTTClient 文件夹。

MQTTPacket 文件夹包含了 MQTT 协议报文的解析和封装功能。这些文件提供了处理 MQTT 协议中不同报文(如 CONNECT、PUBLISH、SUBSCRIBE 等)的代码实现。它们负责将 MQTT 消息编码为字节流(序列化)，或者将接收到的字节流解析为可操作的消息(反序列化)。

MQTTClient-C 文件夹包含了 MQTT 客户端的 C 语言实现的核心部分。这些文件实现了 MQTT 客户端的连接、发布、订阅、断开连接等功能。它们是构建在 MQTTPacket 基础之上的更高级别的抽象，提供了一个易于使用和集成的 MQTT 客户端接口。在移植过程中，我们需要对这个文件夹中的一些网络接口进行修改，才能利用其中的函数正常地与服务器通信。

MQTTClient 是一个最初为 mbed 编写的 C++库，现已移植到其他平台上使用。该库基于并且需要 MQTTPacket。对于仅使用 C 语言的系统来说，可以忽略此文件夹。

表 1 说明了 MQTTPacket 文件夹中与客户端有关的文件的作用。

Table 1. The role of files in the MQTTPacket folder
表 1. MQTTPacket 文件夹中的文件作用

文件	描述
MQTTConnectClient.c	封装 MQTT 客户端连接相关的报文，如连接建立、参数设置等。
MQTTSerializePublish.c	将要发布的消息封装为 MQTT 报文格式，以便网络传输。
MQTTDeserializePublish.c	解析 MQTT 发布的消息，将报文数据解析为应用消息。
MQTTSubscribeClient.c	处理订阅请求和响应的报文。
MQTTPacket.c	包含了对 MQTT 报文的封装和解析的详细实现。
MQTTFormat.c	将 MQTT 报文的二进制字节流转换为易于阅读和分析的字符串形式。

由于这些文件只负责处理报文格式，不涉及与单片机有关的通信接口，因此在移植过程中直接复制到工程中即可，不需要修改。

MQTTClient-C 文件夹中使用的主要是 MQTTClient.c 文件。该文件是 MQTT 客户端库的核心实现，负责实现 MQTT 协议的各种功能，提供了连接 Broker、发布和订阅消息等高级抽象接口。它通过调用 MQTTPacket 文件夹中的程序来封装报文，并通过调用相关的网络接口将这些报文发送出去。在该文件中，初始化函数 MQTTClientInit()需要调用 Network 结构体来传入有关 Socket 网络接口。该结构体涉及到 Socket 接口编号、接收函数、发送函数。因此需要编写程序来初始化 Socket 接口，并编写接收和发送函数，以供 MQTTClient.c 文件中的函数调用。还需要编写供超时判断用的计时函数。

3.3. 移植说明

官方的 MQTTClient-C 文件夹中已经打包好了一些例程，它们适配于 FreeRTOS、Linux、CC3200。移植时需要依据这些例程编写一个程序文件(此处名称为 Hi3861_PahoMQTT.c)，为 MQTTClient.c 文件中

的函数配置好网络接口和计时函数。网络接口利用 Socket 通过 TCP 的方式连接到服务器并进行通信，计时函数用于通信超时判断。要编写的文件中的程序内容如表 2 所示。

图 2 通过发布消息的流程来说明在 Hi3861_PahoMQTT.c 文件中增加的程序的作用。

Table 2. Programs that need to be written during migration
表 2. 移植时需要编写的程序

函数	作用
NetworkRead()	利用 Socket 从网络中读取数据。
NetworkWrite()	利用 Socket 向网络中写入数据。
NetworkInit()	初始化网络结构体，包括 Socket 编号和读写数据的函数接口。
NetworkConnect()	利用 IP 地址和端口号连接到指定主机。
NetworkDisconnect()	断开网络连接。
其它计时函数	用于判断通信是否超时

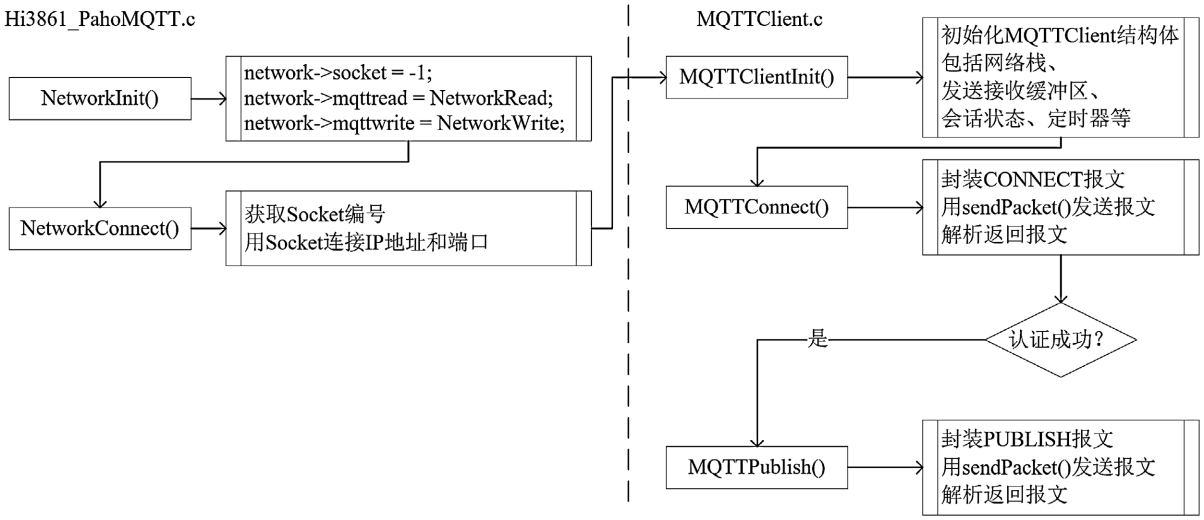

Figure 2. The process of publishing messages using the paho.mqtt.c library
图 2. 利用 paho.mqtt.c 库发布消息的流程

从图 2 中可以看到，为了移植嵌入式的 Paho MQTT C 库而新增的几个函数的主要作用是初始化 Socket 接口、设置利用 Socket 进行接收和发送数据的函数，然后利用 Socket 连接服务器。配置完成后，设备后续与 MQTT Broker 进行连接认证、发布消息都依靠的是 MQTTClient.c 文件中的相关函数。这些函数通过调用 MQTTPacket 文件夹中的函数来进行报文的封装或解析，利用与 Socket 相关的函数进行数据的接收和发送。

4. 在 Hi3861 上移植 Paho MQTT

Hi3861 是海思推出的一款支持 WiFi 功能的处理器，其内核采用 OpenHarmony 架构下的 LiteOS 操作系统[4]。在官方提供的工程文件中，已经实现了在 LiteOS 系统下的 WiFi 连接接口和用于实现 TCP/IP 协议栈的 LwIP (Light weight IP)库。通过调用这些接口，Hi3861 可以连接到 WiFi 热点上面，获取到 IP 地址，并创建 Socket 套接字[5]。移植前需要将 Paho MQTT C 库相关文件复制到工程中，并设置好 BUILD.gn 项目构建文件。

在 LiteOS 操作系统中，setsockopt()函数用于配置套接字的选项，如协议级别、接收发送的超时时间、保活机制等，recv()用于从 Socket 接收数据，send()用于向 Socket 发送数据。因此，在第 3.3 节提到的 NetworkRead()函数中需要先通过 setsockopt()设置接收超时时间，然后调用 recv()函数进行数据的接收；在 NetworkWrite()函数中需要先通过 setsockopt()设置发送超时时间，然后调用 send()函数进行数据的发送。LiteOS 中的 socket()函数用来创建一个新的套接字(Socket)，connect()函数用于在客户端套接字上建立与远程服务器的连接。因此这两个函数需要在 NetworkConnect()函数中对这两个函数进行调用，从而连接到云平台的服务器上。

另外还需要移植用于超时判断的定时器相关函数，方法主要是通过内核 Tick 计数获取函数 osKernelGetTickCount()和系统定时器计数获取函数 osKernelGetSysTimerCount()，配合计数频率来计算秒数和微秒数，这涉及到对 MQTTClient.c 中的 TimerIsExpired()、TimerCountdownMS()、TimerCountdown()、TimerLeftMS()这几个函数的重定义。

配置好 MQTTClient.c 中的函数接口后，即可调用 MQTTClient.c 中的相关函数进行 MQTT 的认证、订阅、取消订阅、发布、接收相关操作了。

5. 实验部署

本文采用华清远见研发的 FS-Hi3861 开发板进行 Hi3861 工程的部署，使用华为云平台的设备接入 IoTDA 资源作为 MQTT 服务器。

首先需要在华为云的设备接入 IoTDA 资源中创建产品及相关属性、命令。属性为设备上报的数据，命令为向设备下发的数据，这两种数据的话题和格式是不一样的。然后注册一个设备，获取到设备的鉴权信息，包括客户端 ID、用户名、密码。

设备需要利用 MQTTClient.c 中的 MQTTConnect()函数包装好鉴权信息，与服务器进行认证。认证通过后，在华为云中会显示设备处于在线状态。然后，设备可以利用 MQTTPublish()函数按照华为云规定的格式向指定的话题发布属性数据。设备还可以利用 MQTTSubscribe()函数订阅指定的话题，并通过 MQTTRun()函数接收云平台下发的数据。

6. 未来工作

本文以 Hi3861 为例，介绍了在 OpenHarmony 项目的 LiteOS 内核上面通过移植 Paho MQTT 连接 MQTT 云平台的方法。由于云平台通常使用 JSON 格式进行数据传递，因此在未来的项目中，将通过移植 cJSON 库的方式进行数据的序列化和反序列化，从而更加方便地传输传感器数据并解析相关指令。另外，为了便于传感器网络的建立，未来的项目将探讨利用 Hi3861 进行 Mesh 组网，从而扩展传感范围的方式。为了适应低功耗需求，未来的项目也将探讨低功耗 WiFi 模式下 MQTT 的应用。

参考文献

[1] Vanani, K., Patoliya, J. and Patel, H. (2016) A Survey: Embedded World around MQTT Protocol for IoT Application. *International Journal for Scientific Research and Development*, **4**, 26-29.

[2] Light, R.A. (2017) Mosquitto: Server and Client Implementation of the MQTT Protocol. *The Journal of Open Source Software*, **2**, Article 265. https://doi.org/10.21105/joss.00265

[3] Eclipse Paho (2024) Eclipse Paho. https://eclipse.dev/paho/

[4] 海思官网. Hi3861LV100 产品简介[EB/OL]. https://www.hisilicon.com/cn/products/connectivity/short-range-IoT/wifi-nearlink-ble/Hi3861V100, 2024-06-27.

[5] 何进, 谢松巍. 基于 Socket 的 TCP/IP 网络通讯模式研究[J]. 计算机应用研究, 2001(8): 134-135.