# **第三章 Microbot框架需求分析与概要设计**

## Microbot项目整体概述

Microbot框架是一个基于Node.js的支持微服务的物联网开发框架，帮助开发者使用Javascript直接与嵌入式设备交互，进行物联网应用的开发，而不需要学习底层的程序语言。

如图3.1所示，Microbot的功能主要分为以下两大块：

第一部分是对于底层硬件设备进行一个抽象处理，把硬件设备接口化，提供一个简单统一的接口，使得开发者不需要投入额外的时间成本去学习关于底层设备的硬件实现，从而实现快速开发。Microbot现在提供对Arduino、Raspberry Pi平台的支持和基于GPIO和I2C总线的设备元件的实现。Arduino等平台全部继承自框架封装的Adaptor，而基于GPIO等总线的元件全部继承自框架封装的Driver。 Driver和Adaptor就是Microbot的主体血肉。

第二部分是对微服务架构的支持。Microbot实现了基于MQTT的发布、订阅消息服务和基于HTTP、REST API和 Socket.io这三种类型的服务。系统中的各个微服务可被单独部署，彼此之间是松耦合的，每个任务代表着一个小的业务能力。开发者可以根据业务需要配置参数，选择服务类型，直接调用相应的接口就可以轻松实现服务的发布或者订阅，不需要关注协议的具体实现。

## 3.2 Microbot框架的需求分析

### 3.2.1 Microbot 框架用例图

Microbot框架的涉众只有唯一的一类：物联网应用开发者。他们通过框架来开发物联网应用，主要有两大类的需求类别：第一部分是利用微服务架构，配置服务，对服务进行发布、订阅和取消。另外还有一个可视化界面，方便开发者查看当前运行的设备和正在使用中的服务。第二部分是通过配置连接件、配置设备、访问并操作设备来对硬件进行操作和管理，实现具体应用的业务逻辑。具体用例入下图所示：

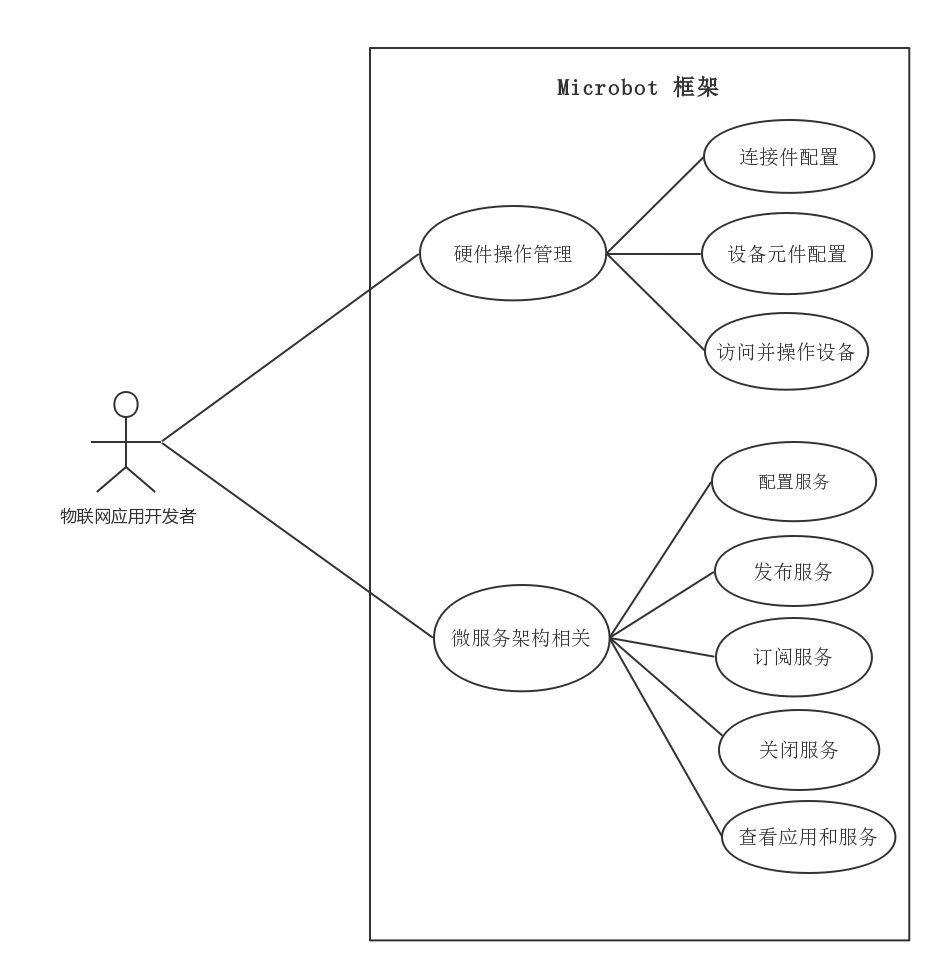


图3.1 Microbot框架用例图

### 3.2.2 Microbot 框架用例描述

对 Microbot 框架的“连接件配置”、“设备元件配置”、“访问并操作设备”、“配置服务”、“发布服务”、“订阅服务”、“取消服务”、“查看应用和服务”的用例描述分别如表3.2，表3.3，表3.4，表3.5，表3.6，表3.7，表3.8以及表3.9所示。

表3.2是对连接件配置的用例描述。开发者对连接件的 connection 进行定义，在其中定义具体的连接件名称和平台类型以及端口号。

表3.2 连接件配置用例描述

|  |  |
| --- | --- |
| UC1 连接件配置 | |
| 参与者 | 物联网应用开发者，目标是配置嵌入式设备的连接件 |
| 触发条件 | 开发者需要配置连接件 |
| 前置条件 | 无 |
| 后置条件 | 与连接件建立连接 |
| 正常流程 | 1.开发者将连接件接入 pc 的 USB 端口  2.系统要求开发者定义connection 中具体的连接件名称和平台类型以及端口号  3.开发者输入连接件名称、平台类型和端口号 |
| 扩展流程 | 2a.开发者需要添加更多的连接件  1.开发者将更多的连接件接上 pc  3a.开发者输入的连接件平台系统不支持  1.系统提示该连接件平台不支持  3b.开发者输入的端口号与 pc 系统不匹配  1.系统提示该端口号无法找到 |

表3.3是对连接件配置的用例描述。开发者对设备元件的 devices进行定义，在其中定义具体的元件名称、驱动类型、与连接点所建立的连接以及针脚号。

表3.3 设备元件配置用例描述

|  |  |
| --- | --- |
| UC2 设备元件配置 | |
| 参与者 | 物联网应用开发者，目标是配置设备元件 |
| 触发条件 | 开发者需要配置设备元件 |
| 前置条件 | 开发者已经将连接件连接配置完成 |
| 后置条件 | 与设备元件建立连接 |
| 正常流程 | 1.开发者将设备元件接上连接件  2.系统要求开发者定义devices中具体的设备元件名称和驱动类型以及针脚号  3.开发者输入设备元件名称、驱动类型和针脚号 |
| 扩展流程 | 1a. 开发者设备元件电路连接错误  2.系统提示元件连接错误  2a.开发者需要添加更多的设备元件  1.开发者将更多的元件接上连接件  3a.开发者输入的元件类型统不支持  1.系统提示该连接件平台不支持  3b.开发者输入的针脚号与元件连接的针脚号不匹配  1.系统提示该针脚号无应答 |

表3.4是对访问并操作设备的用例描述。开发者将已经配置好的连接件和设备元件组合成一个整体，访问该整体所开放的功能，并使用这些功能对设备整体进行操作。设备根据开发者的操作指令做出对应的响应。

表3.4 访问并操作设备用例描述

|  |  |
| --- | --- |
| UC3访问并操作设备 | |
| 参与者 | 物联网应用开发者，目标是访问并操作设备 |
| 触发条件 | 开发者需要访问并操作设备 |
| 前置条件 | 开发者已经正确配置好连接件和设备元件 |
| 后置条件 | 设备元件根据指令做出响应 |
| 正常流程 | 1.开发者将连接件和设备元件组合成一个设备整体  2.系统要求开发者给出对整体设备的指令，对设备进行操作  3.开发者给出指令操作设备 |
| 扩展流程 | 1a.开发者组合的设备元件和连接件不匹配  2.系统提示连接件和设备元件不匹配，组合失败  3a.开发者给出的操作指令设备不支持  1.系统提示不支持该操作指令 |

表3.5是配置服务用例描述。开发者对服务进行定义，在其中定义具体的服务名称和通信协议类型以及代理主机ip。

表3.5 配置服务用例描述

|  |  |
| --- | --- |
| UC4配置服务 | |
| 参与者 | 物联网应用开发者，目标是配置服务 |
| 触发条件 | 开发者需要配置服务 |
| 前置条件 | 开发者已经可以访问和操作设备 |
| 后置条件 | 无 |
| 正常流程 | 1.开发者添加一个服务  2.系统要求开发者给出服务的具体名称、通信协议类型和代理的主机ip  3.开发者给出服务名称、通信协议类型和代理主机ip |
| 扩展流程 | 3a. 开发者定义的通信协议类型系统不支持  1.系统提示不支持该类通信协议  3b. 开发者给出的代理主机地址错误  1.系统提示主机地址无响应 |

表3.6是发布服务用例描述。开发者对服务进行定义，在其中定义具体的服务主题地址和服务发布所携带的有效数据以及服务的发布频率。

表3.6发布服务用例描述

|  |  |
| --- | --- |
| UC5发布服务 | |
| 参与者 | 物联网应用开发者，目标是对服务进行发布 |
| 触发条件 | 开发者需要发布服务 |
| 前置条件 | 开发者已经成功配置服务 |
| 后置条件 | 服务被发布，按给定频率发布数据 |
| 正常流程 | 1.开发者请求发布一个服务  2.系统要求开发者给出具体的服务主题地址和服务发布所携带的有效数据以及服务发布的频率  3.开发者给出具体的服务主题地址和服务发布所携带的有效数据以及服务发布的频率 |

表3.7是订阅服务用例描述。开发者请求订阅服务，给出所需订阅的服务所在代理的ip和服务的主题地址。

表3.7订阅服务用例描述

|  |  |
| --- | --- |
| UC6订阅服务 | |
| 参与者 | 物联网应用开发者，目标是对某个特定服务进行订阅 |
| 触发条件 | 开发者需要订阅服务 |
| 前置条件 | 开发者有已经成功配置可以订阅服务的设备 |
| 后置条件 | 开发者通过订阅服务获得的数据，进行进一步开发操作 |
| 正常流程 | 1.开发者请求订阅一个服务  2.系统要求开发者给出所要订阅的服务的主题地址和服务所在的代理地址  3. 开发者给出所要订阅的服务的主题地址和服务所在的代理地址 |
| 扩展流程 | 3a. 开发者给出的的服务的主题地址和服务所在的代理地址不匹配  1.系统提示服务的主题地址和代理地址不匹配，订阅失败 |

表3.8是取消服务用例描述。开发者请求取消订阅某特定服务，给出所需取消订阅的服务所在代理的ip和服务的主题地址。

表3.8取消服务用例描述

|  |  |
| --- | --- |
| UC7取消服务 | |
| 参与者 | 物联网应用开发者，目标是取消对某个特定服务的订阅 |
| 触发条件 | 开发者需要取消对某服务的订阅 |
| 前置条件 | 开发者有已经订阅该服务 |
| 后置条件 | 开发者不再收到该服务发布的数据 |
| 正常流程 | 1.开发者请求取消订阅某服务  2.系统要求开发者给出所要取消订阅的服务的主题地址和服务所在的代理地址  3. 开发者给出所要取消订阅的服务的主题地址和服务所在的代理地址 |
| 扩展流程 | 3a. 开发者给出的的服务的主题地址和服务所在的代理地址不匹配  1.系统提示服务的主题地址和代理地址不匹配，取消订阅失败 |

表3.9是查看应用和服务用例描述。开发者请求查看当前运行中的设备、运行设备的详细信息、已经发布的服务和已经订阅的服务以及服务的详细信息

表3.9 查看应用和服务用例描述

|  |  |
| --- | --- |
| UC8查看应用和服务 | |
| 参与者 | 物联网应用开发者，目标是查看当前应用和服务 |
| 触发条件 | 开发者需要查看当前应用和服务 |
| 前置条件 | 开发者的设备能正常运行，连接到服务器 |
| 后置条件 | 无 |
| 正常流程 | 1.开发者请求查看应用和服务  2.系统给出当前在运行中的应用和服务  3.开发者请求查看当前在运行中的设备的详细信息，包括设备名、设备连接件的平台类型、连接端口、设备所连接的元件名、以及元件连接的针脚号  4.系统提供当前在运行中的设备的详细信息，包括设备名、设备连接件的平台类型、连接端口、设备所连接的元件名、以及元件连接的针脚号  5.开发者请求查看当前在运行中的服务的详细信息，包括服务名、服务所采用的通信协议类型、连接端口号、代理 ip、订阅该服务的主题地址和发布该服务的设备名  6.系统给出当前在运行中的服务的详细信息，包括服务名、服务所采用的通信协议类型、连接端口号、代理 ip、订阅该服务的主题地址和发布该服务的设备名 |

### 3.2.3 Microbot 框架功能性需求

分析用例图和关键用例，总结出Microbot 的功能需求列表，如表3.10所示。

表3.10 Microbot 框架功能性需求列表

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 需求名称 | 需求描述 | 优先级 |
| R1 | 配置连接件信息 | 开发者能够根据开发的需要对所要用到的硬件连接件设备进行配置，并在程序启动时完成连接件硬件连接。 | 高 |
| R2 | 配置设备元件信息 | 开发者能够根据开发需要，对需要用到的硬件设备元件的信息进行配置，并在程序启动时完成连接，确保设备元件与连接件的正确衔接和正常通行。 | 高 |
| R3 | 访问并操作设备 | 开发者能够在主程序中访问并操作设备，调用设备开放的接口，实现具体应用中的业务逻辑。 | 高 |
| R4 | 配置服务信息 | 开发者能够通过对应用的相关信息进行配置，然后将其封装成微服务。 | 高 |
| R5 | 发布服务 | 开发者能够通过 MQTT 通信协议发布服务到 Broker 或者通过 HTTP 协议，开放 RESTful API 接口接受客户端请求。 | 高 |
| R6 | 订阅服务 | 开发者能在应用的业务逻辑中订阅MQTT 服务或者通过 HTPP 通信协议调用其他服务提供的 REST API 来发送请求 | 高 |
| R7 | 关闭服务 | 开发者能够根据应用业务逻辑的需要关闭服务 | 中 |
| R8 | 查看运行中的应用和服务 | 开发者能够查看当前正在运行的设备信息和子应用以及与其相关的服务信息 | 中 |

### 3.2.4 Microbot 框架非功能性需求

除了功能需求之外，Microbot还有非功能性需求，从需求类别来划分主要包括性能需求和质量需求这两类。如表3.11所示，Microbot 的非功能性需求主要来自以下若干方面：

第一、Microbot 支持 Websocket 的实时通信，所以在性能方面需求方面要求服务的实时性，当发送请求后，需要在2s 内在嵌入式设备上得到响应。

第二、Microbot 作为一个框架，需要满足开发者对同时操作多个嵌入式设备和硬件元件的需求，所以需要考虑框架可以承担的并行工作量，有负载的需求。框架应该允许至少能同时操作6个嵌入式设备和硬件元件。

第三、Microbot 作为一个框架类项目，用户将使用框架进行二次开发，需要将各层次之间进行封装，开放出统一接口，对框架自身的易用性要求较高。

第四、Microbot 是针对物联网应用开发的框架，物联网硬件平台和设备种类繁多，并且目前还没有统一公共标准。框架除了要支持对目前主流嵌入式设备和硬件元件的支持外，还必须考虑框架自身的可扩展性。当开发者需要加入新的硬件平台时，可拓展性可以尽量降低支持某个硬件平台或硬件元件的开发成本。

第五、Microbot 支持网络通信，所以对发布和订阅服务的可靠性有一定的要求。框架应该保证在客服端与服务器端通信时，如果网络故障，系统不能出现故障。

第六、Microbot 作为一个框架，需要支持各平台的使用，所以对可移植性也有要求，本框架只需要安装 Node.js 环境即可在任意操作系统上运行。

第七、在开发者使用 Microbot 框架进行物联网应用开发的时候，需要框架对可用性做一个保证，框架在投入使用时可操作性和可访问的程度要达到95%。

表3.11 Microbot框架非功能需求列表

|  |  |  |
| --- | --- | --- |
| 编号 | 非功能需求 | 需求描述 |
| PR1 | 实时性 | 框架支持网络实时通信，所以在性能方面需求方面要求服务的实时性，当发送请求后，需要在2s 内在嵌入式设备上得到响应 |
| PR2 | 负载 | 框架可以同时对多个嵌入式设备和硬件元件进行操作，框架也支持同时发布或订阅多个服务。 |
| QA1 | 易用性 | 框架的学习成本低。接口和参数简单明了，命名规范和注释说明完整，易于理解；有合理的错误提示，便于开发者区分。学习框架1天的开发者可以使用框架便捷的开发物联网应用。 |
| QA2 | 可拓展性 | 框架可以灵活的删减和增加对硬件品台和设备的支持，不需要修改大量原有代码，不会对其他模块造成影响。 |
| QA3 | 可靠性 | 在客户端与服务器通信时，如果网络故障，框架不能出现故障。 |
| QA4 | 可移植性 | 只需要安装 Node.js 环境，框架可以在多个操作系统运行，包括 Windows，Mac OSX，Linux 等操作系统。 |
| QA5 | 可用性 | 框架在投入使用时要求框架所提供的功能的可用性要达到95%。 |

### 3.3 Microbot框架的概要设计

### 3.3.1 系统框架结构

下图展示了Microbot框架的整体架构：

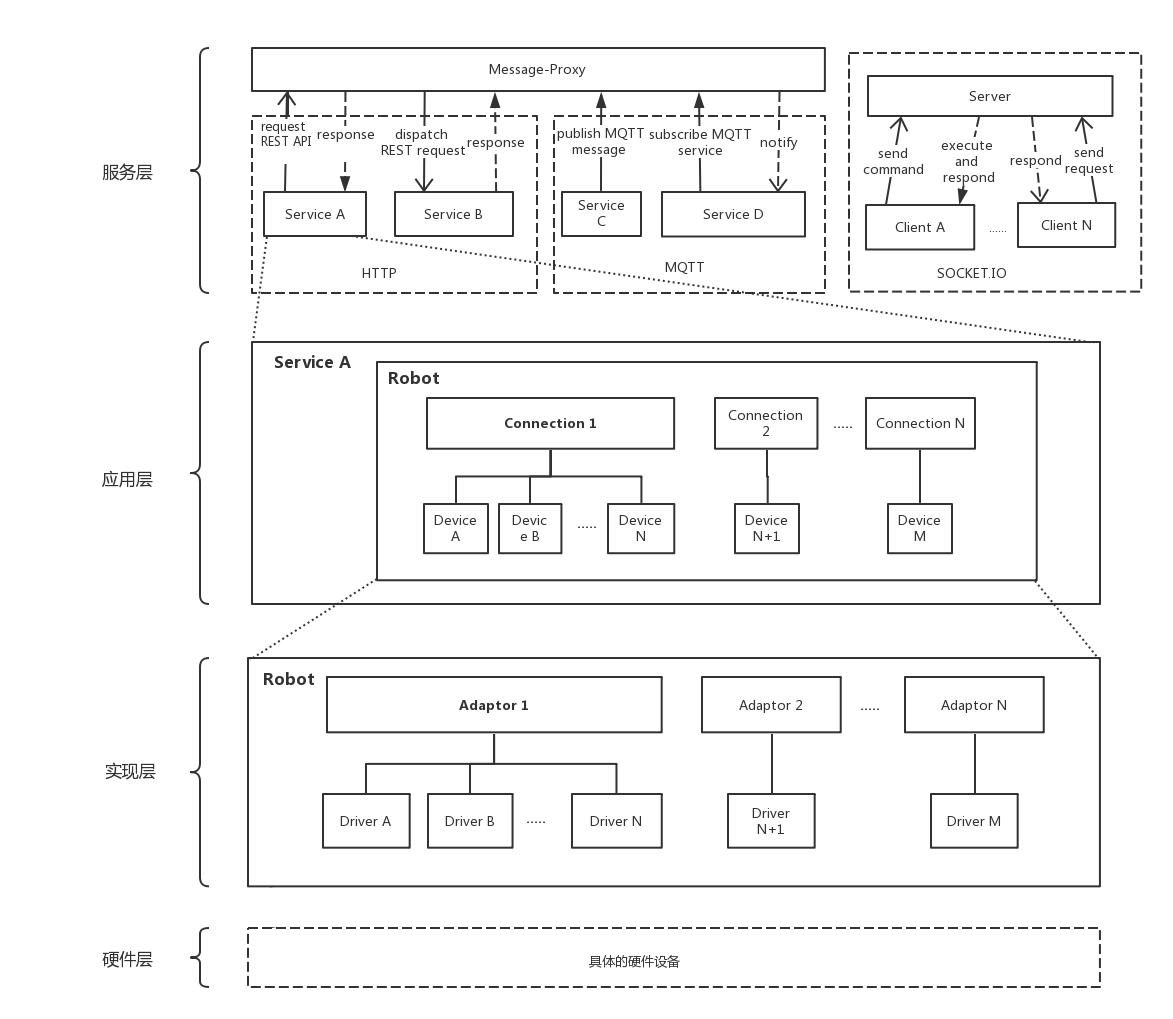


图3.2 Microbot框架架构分层视图

如图3.2所示：Microbot 框架由服务层、应用层、实现层、硬件层这四层构成。以下对这四层分别加以说明：

首先是服务层，服务层主要提供对微服务框架的支持。微服务架构的顶层是一个服务代理中心Message-Proxy，所有服务间的 MQTT 和 HTTP 通信都通过消息代理完成。服务实例通过对所使用的协议的配置，按照接口说明传入参数并调用即可实现服务的请求或发布。

图3.2中 Service A 和 Service B 展示了对 REST API请求的处理：服务请求方调用服务的请求接口，发送请求； Message-Proxy 转发请求，Microbot内部通过服务代理中心接受 Service A的请求并转发该请求到最终调用的服务地址，获得最终服务地址即Service B的响应后再将该响应转发给服务请求方 Service A。整个流程如如图3.3所示。这一过程虽然增加了一次服务转发，但是通过服务代理中心，可以完成诸如统一接口、格式、权限验证等增值服务，节约了框架使用者在开发过程中的开发成本，也使得代码更好的封装，使得最终的实现代码更简洁明了。

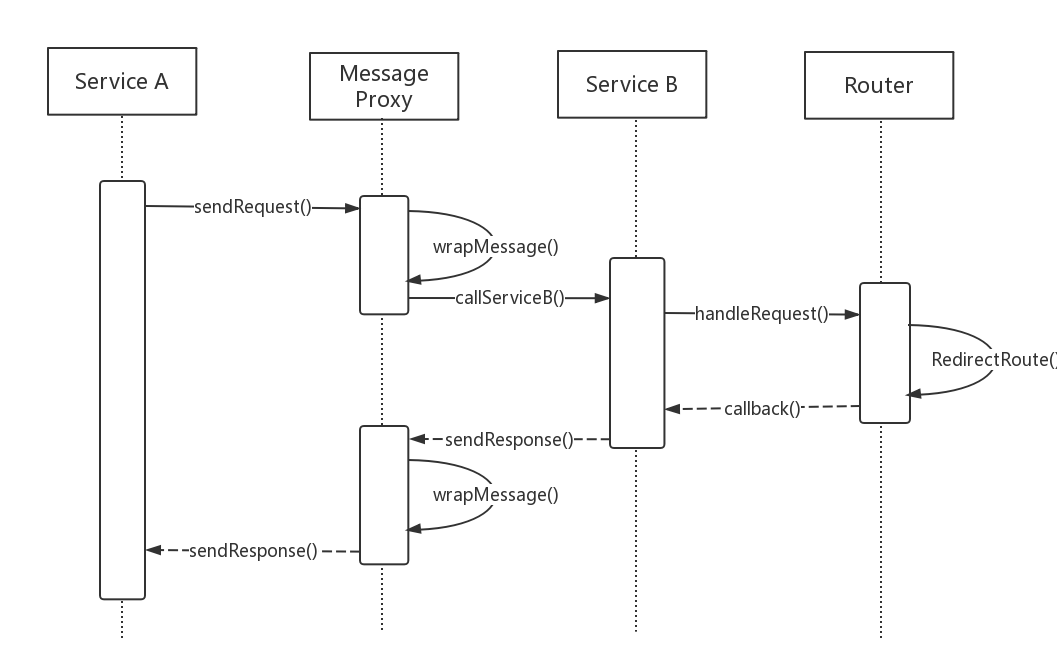


图3.3 Microbot框架微服务框架HTTP消息代理

继续阐述服务层的 MQTT 通信部分。在图3.2中Service C 和 Service D则展示了 MQTT 服务的实现，服务发布方调用发布接口，设置服务的主题和消息，经过配置的服务由服务代理中心接管，服务代理中心维护了一份订阅每个主题的客户列表，服务代理中心根据该列表将其发布给服务订阅方。另外，当有新的服务订阅方请求订阅某服务时，服务代理中心会更新该服务的订阅列表信息。

具体如图3.4所示。Service C请求订阅Service D服务，在发起订阅请求时注册一个接受应对服务消息的 API 供Message-Proxy调用；Message-Proxy 接收到 Service C 的请求，将Service C作为客户加入，更新针对Service D的服务订阅列表。对于服务发布方 Service D 而言，他的订阅者是 Message-Proxy；Service D 通过 MQTT 发布消息给他的客户 Message-Proxy, Message-Proxy再通过 REST API 将封装过后的消息分发给订阅者。

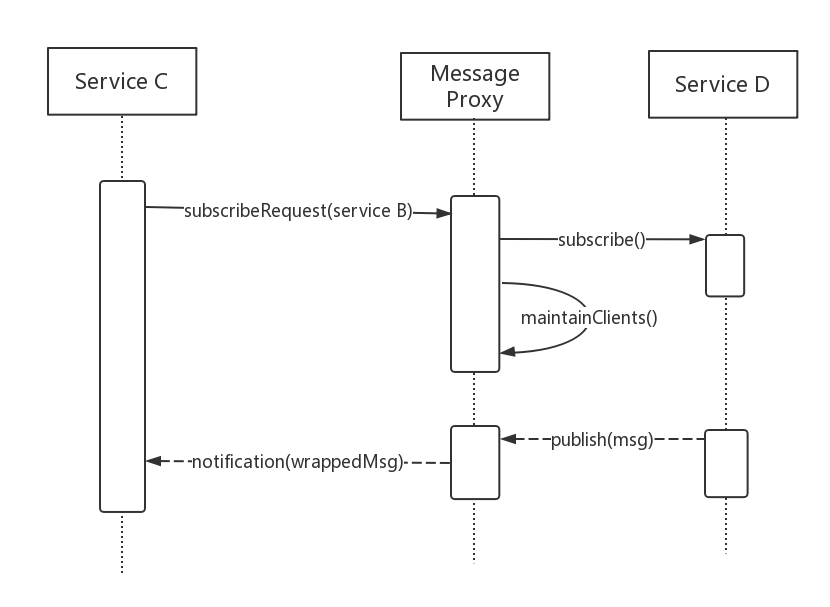


图3.4 Microbot框架微服务框架 MQTT发布/订阅消息代理

继续阐述服务层的 Socket.io 通信部分。与 HTTP 和 MQTT 不同，Socket.io没有采取消息代理的实现方式，而是使用了 C/S 架构，使用客户端和服务器的结构。运行中的设备作为一个服务器端，同时可以连接多个客户端，现在我们采用的客服端的平台是浏览器。可以将客户端理解为一个基于浏览器的浏览器应用，通过网页端与作为服务器的设备进行信息指令的实时交互，如图3.5所示。

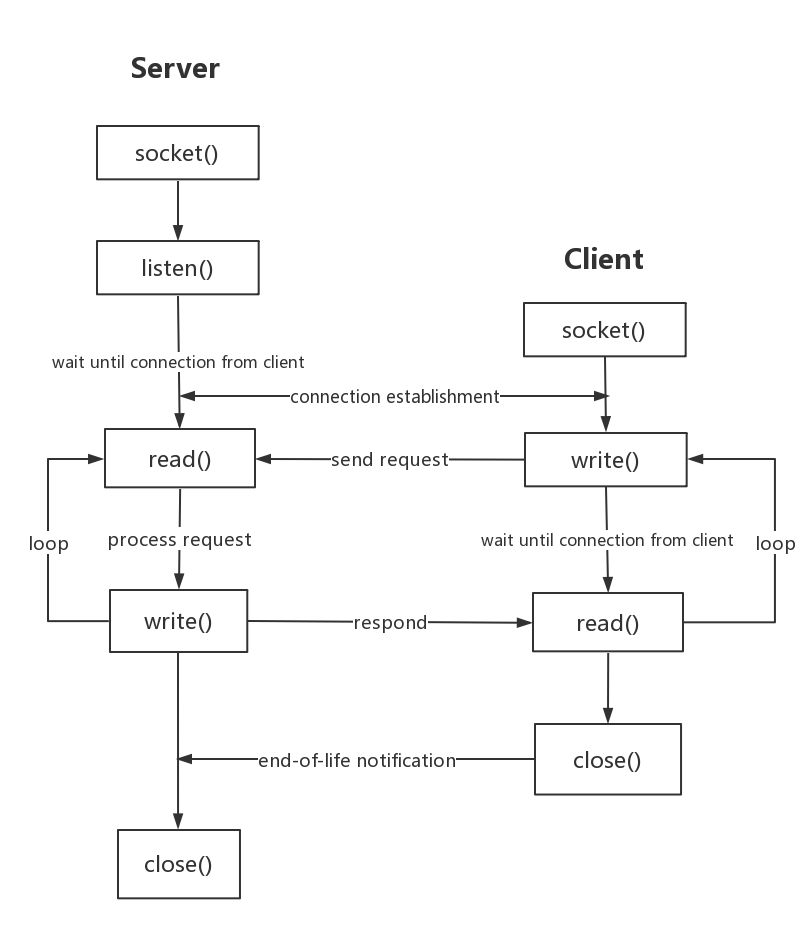


图3.5 Microbot框架微服务框架 Socket.io通信交互图

其次是应用层，从本质上来说应用层是现实连接的硬件设备在代码层次的整体高层抽象。Microbot 中把一个完整的物联网应用定义为一个 Robot，每个Robot 可以由一个或多个连接件（Connection）组成，每个连接件从属于某个硬件平台，例如 Arduino 板、Raspberry Pi 等。与现实中一个硬件开发板可以连接多个硬件设备元件一样，每个连接件又可以连接多个设备（Device）。连接件和设备通过在应用初始化时配置每个连接件对应的适配器（Adaptor）和设备对应的驱动（Driver）等相关信息，由框架初始化模块完成实例化过程。

接着是实现层，实现层是比应用层更偏底层的一些硬件抽象，涉及到与平台相关的具体实现。适配器（Adaptor）是硬件平台的抽象，每个硬件平台（如 Arduino 等）的具体实现就是一个适配器。适配器本身开放出标准的统一接口，这些接口就是使用 Javascript实现的硬件的功能，适配器将硬件功能封装成接口提供给上层使用。驱动（Driver）和适配器的原理相似，但是驱动是硬件元件设备的抽象，他调用 Adaptor 开放的接口实现每个硬件元件的具体功能，所以驱动对适配器有依赖关系。

最后是硬件层，硬件层对应现实中的硬件设备，实际上这并不属于框架的实现范围，但是放入分层架构图中更有助于理解整个应用。目前框架只支持Arduino 和 Raspberry Pi这两个硬件平台，但是由于框架设计良好的可拓展性，只需要添加Adaptor对需要增加的硬件平台进行封装即可。

### 3.3.2 Microbot 框架模块间接口

上一小节从分层视图的角度介绍了Microbot的体系架构，本小节则从模块组成的角度分析Microbot的架构和模块间接口。

Microbot 框架按照功能主要划分为Service（服务）、Application（应用）和 Initialization（初始化）三大模块。各个模块职责如表3.12所示。

表3.12 Microbot框架各模块职责表

|  |  |
| --- | --- |
| 模块 | 职责 |
| Initialization（启动模块） | 负责初始化主程序，实例化服务代理中心和适配器以及驱动工场的单例，启动应用。 |
| Service（服务模块） | 负责微服务实例化和通信 |
| Application（应用模块） | 负责物联网应用各组成部分的组建和初始化，提供业务逻辑实现接口。 |

由于微处理器设备的特殊性，他们的功能通常有较大的差异性，接口也天差地别，所以很难抽象出一个父类拥有大量的公共接口。因此，Microbot 在抽象类 Adaptor 中，只抽象出了 connect 和 disconnect 这两个公共方法供Robot调用。仅用于程序启动时负责连接件的连接与程序结束时连接件的断开，其他具体的功能的接口由负责其他平台的子类提供。Driver 类也相当类似，由于有种类繁多的硬件元件，抽象出一个高度统一的接口是不可能的。于是 Driver 的抽象类中只包含 start 和 halt 这两个公共方法，用于开启和终止设备，其他的功能接口由具体的硬件元件相对应的驱动子类提供。

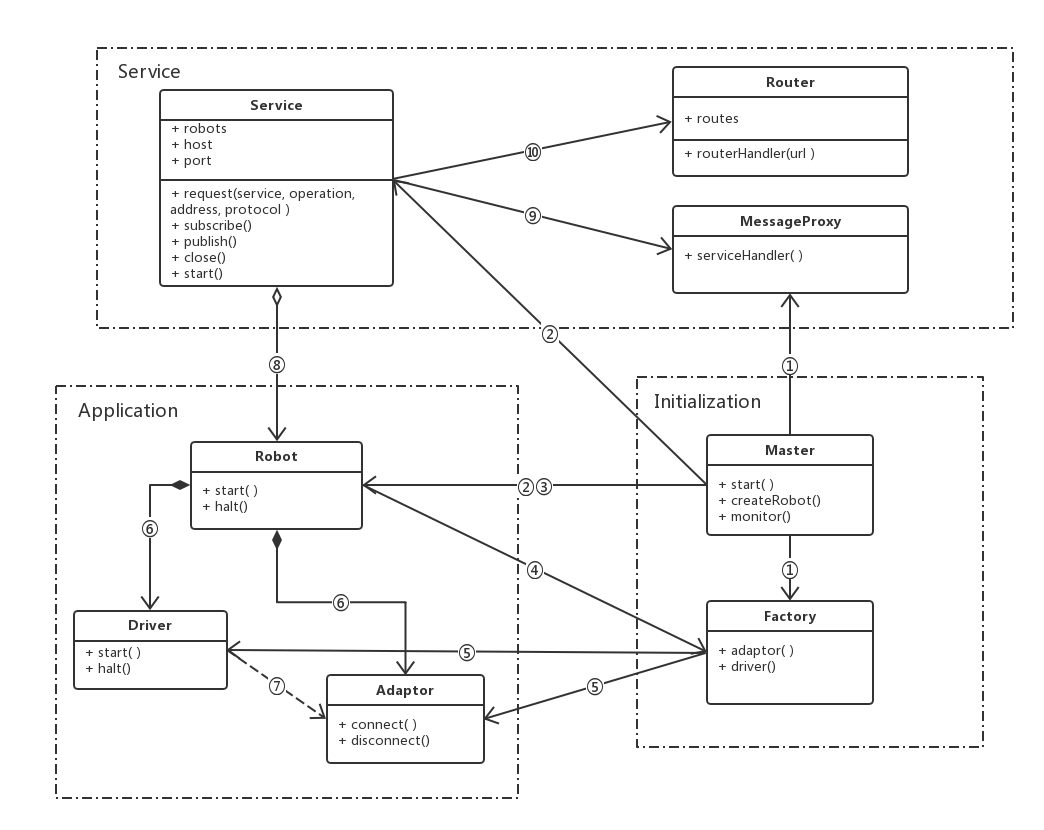


图3.6 Microbot框架模块视图及关键类图

上图展示了Microbot 框架内部只要模块间接口的关系，具体说明如下：

① Master 的 start 方法是程序运行时的主入口，负责对 Factory 和 MessageProxy 的单例的初始化。

② Master 维护了当前正在运行的 Service 和 Robot 的实例，通过访问浏览器本地地址可以查看当前运行的服务和应用的详细信息。

③ Master 的 createRobot 方法是创建 Robot 的开放接口，开发者通过调用该方法可以实现对 Robot 实例的创建。

④ Robot 在启动时调用 Factory 的工厂方法完成对连接件和设备的初始化。

⑤ Factory是一个应用了工场模式的单例，根据传入的参数require 对应的Adaptor和Driver子类，将相应的子类实例化。

⑥ Robot中聚合了Driver和Adaptor，这一点使得开发者使得可以在robot 中对适配器和驱动进行操作和访问，实现robot的具体功能。

⑦ 每一个具体的Driver子类都包含对相应Adaptor的依赖，具体的关系在配置 robot 的过程中被定义，在舒适化过程中绑定。Driver通过调用其connect和disconnect连接或断开与其绑定的连接件。

⑧ Service中聚合了Robot，这使得开发者可以直接从Service的API访问到robot实例，获取数据实现业务逻辑，对外提供服务；在robot实例也包含对service的依赖，所以在robot主程序运行时，robot可以发布服务或者请求外部服务，以此来实现自己内部的复杂功能。

⑨ Service实例调用MessageProxy的接口实现服务的发布、订阅和请求。

⑩ Service请求服务是时调用Router进行重定向连接到相应的服务发布方的路径。

### 3.4 本章小结

本章首先介绍了Microbot框架的基本情况与架构，然后对Microbot框架的整体需求进行了分析，并对微服务架构相关需求和硬件配置连接需求做了详细需求分析。接着描述了根据需求分析的结果得到的功能模块设计，介绍了框架中各模块职责，并且介绍了框架架构设计和开发中采用的模块视图和关键类图。