# **第一章 引言**

## 1.1 项目背景

[国际电信联盟](http://baike.baidu.com/item/%E5%9B%BD%E9%99%85%E7%94%B5%E4%BF%A1%E8%81%94%E7%9B%9F)( ITU[[1]](#footnote-1))，对物联网进行了定义：通过二维码识读设备、射频识别([RFID](http://baike.baidu.com/item/RFID)[[2]](#footnote-2))装置、红外感应器、全球定位系统和激光器等信息传感设备，按约定的协议，把互联网与任何物品相链接，进行信息交换和通信，以实现智能化识别、定位、跟踪、监控和管理的一种网络[1]。

随着物联网技术的不断发展和进步，物联网应用的开发 受到IT 界越来越多的重视。于是，越来越多的用于机器人以及物联网的开发框架被应用。百花齐放，但是每个框架都有自己的侧重点，相对的也会有各自的缺点，鲜少有框架做到同时满足将底层抽象和顶层封装集成。并且由于机器人的开发需要对大量各式各样的微处理器[[3]](#footnote-3)进行应用开发来实现功能，这就要求框架还能提供对多种微处理器的底层支持。于是能满足以上需求的框架更是少之又少。

虽说物联网和机器人的概念现在被炒的火热，然而传统的嵌入式开发要求的对底层语言的掌握，门槛高，碎片化严重，平台复杂，大大增加了开发成本与难度，对开发者要求高开发效率低，这将很多人拒之门外。所以就有一种迫切的需求，是否能降低这种开发成本，或许相较于 C/C++，脚本语言是一个更不错的选择。物联网与互联网、移动互联网的本质是类似的，生态圈的繁荣取决于在该平台上应用的数量和开发者的活跃度。不得不承认使用 C/C++来开发应用是低效和昂贵的，解释性语言对开发应用来说才是更好的选择。因此，物联网的未来需要一门解释性语言。相较于PHP、Python、Java 来说，JavaScript[[4]](#footnote-4) 是最有可能在物联网大行其道的解释性语言，因为 Javascript 在浏览器端一枝独秀，Node.js[[5]](#footnote-5)在服务器端的热门与活跃也让人心动，这一切基本上组建了一个完整的应用开发生态环境。Javascript 已经跨界到物联网，受到了行业巨头的关注，各种项目发展的如火如荼。而目前已被大量应用在物联网框架开发中的 Node.js是一个基于 Chrome V8[[6]](#footnote-6)引擎的Javascript 运行环境，使用了事件驱动，非阻塞式 I/O 的模型，轻量又高效[2]。但是由于Node.js没有为各式各样的嵌入式设备提供一个统一的、经过抽象的、可以直接使用的解决方案，所以它不能直接适用于物联网或机器人应用开发。因为，一批以Node.js 为基础的物联网框架陆续出现。比较具有代表性的就是 Cylon.js[[7]](#footnote-7) 和 IoT.js[[8]](#footnote-8)。Cylon.js 对各种嵌入式设备的底层抽象做的十分完备，但是在对外提供独立功能调用，和微服务提供方面并没有给出一个适用的解决方案。IoT.js使用 JerryScript[[9]](#footnote-9) 引擎来运行Javascript代码，使用 libuv 库来实现异步 I/O。这样的结构让开发者能够创建物联网服务，让设备与设备、外界之间交互。然而 IoT.js 的架构又分为 Javascript 实现的上层API 和 C 语言实现的核心层两大层，这样一来 IoT.js 的架构就稍显复杂，核心层的拓展编写依旧无法脱离 C/C++。

## 1.2 典型类似的物联网开发框架应用场景及优缺点

基于Node.js 的机器人和物联网应用开发框架有很多，每一种都有各自的功能侧重点和优缺点。下面分析两个比较主流的物联网应用框架Cylon.js 和 IoT.js 的特征以及优劣势。

Cylon.js是一个基于 Javascript 的机器人及物联网开发框架。Cylon.js 是基于 Node.js的更高一层的框架，对于底层协议相关的模块，大多是引用 Node 的拓展包，然后做进一步封装。Cylon.js可以使用 Node.js 进行机器人开发和物理计算。它提供了一个简易但有力的方案来解决同一时间处理不同硬件设备的问题。Cylon.js 支持 Arduino、Intel Edison、AR.Drone[[10]](#footnote-10) 等在内的18个硬件平台。开发人员可以使用统一的开放 API 来对不同的嵌入式设备进行开发和操作。Cylon 的优势在于：1、基本框架结构简单，学习成本低；2、可扩展性强，支持自定义拓展新的硬件平台和元件；3、分模块，采用模块组装的形式，支持自定义框架，根据需要选择已有的模块；4、开源框架，贡献者众多生态圈欣欣向荣。但是 Cylon.js 也有缺点：基于 Cylon.js 的程序无法独立运行在类似 Arduino 这样的开发板上，因为Cylon 本质上是运行 Node.js利用协议和开发硬件通信。普通的开发板无法运行 Node.js，也就意味着它需要一直连接电脑，使用情景有一点的局限性。

同样作为基于 JS 的物联网应用平台，IoT.js的核心是向下兼容 Node.js，底层采用了 Samsung 自己设计的 Javascript 引擎 JerryScript，通过调用libuv 库来实现异步 I/O。总的架构由上层 API 和 C 语言实现的核心层两部分组成。在 API 层IoT 提供了 ADC/BLE/GPIO/PWM/SPI/UART/UDP 这些通信协议模块，底层用 C 语言自身的 binding 机制抽象到 API 层。但是其实基于 Javascript 语言这件事情本身也有局限性，因为这从一定程度上限制了用户的硬件设备范围。

## 1.3 微服务架构应用于JS框架开发的情况

### **1.2.1 微服务架构定义**

微服务是一种架构风格，在一个大型的复杂度较高的应用可以有一个或多个微服务构成。系统的组件化和服务化是微服务架构的重点所在，系统中的各个微服务可以被单独部署，从前端到控制层、逻辑层、数据库都是完全独立的。原有的单个系统被划分为若干个彼此间松耦合的，可以独立设计开发运行的小型应用。每个小应用在实现自身的逻辑功能的同时，将自身所能提供的服务对外进行发布，也调用外界应用的开放服务，通过服务真正的实现应用的组件化。在微服务架构拓展性强，我们可以直接在某个特定服务中添加功能，不会对整体系统的进程造成影响。

### **1.2.2 微服务架构的优势**

微服务结构有很多好处。首先，微服务通过把巨大单体式应用分解为多个服务解决了程序开发的复杂性问题。每个服务都只关注一个业务功能，相对简单。服务之间是松耦合的，具有较高的灵活性。功能没有改变的条件下，将系统切割为多个更便于理解、管理和维护的模块和服务。

其次，该结构让不同的技术开发团队可以负责不同的服务，不必关注具体实现服务的技术，只需要将重点放在开放的 API 接口。因此即使技术变换，用新的技术重构从前的代码也不会需要巨大的开发成本。

同时，微服务机构为持续交付的提供推力，这保证了，即使在高频发布的情况下，系统的其余模块依然能保持可用性和稳定性。

### **1.2.3微服务架的缺陷**

当然，这个世界上没有银弹。如果是作为一个整体应用去部署，可能只需要一小片服务区。然而微服务架构可能使得运维成本大幅增加，最后需要构建、测试、部署、运营等十数个彼此独立的服务，还需要提供对不同程序语言的支持，可能导致进程数的翻倍。

## 1.4 论文的主要工作和组织结构

本文主要介绍了基于Node.js与微服务的物联网应用框架Microbot，主要提供对Arduino 和 Raspberry P 平台的支持和基于 GPIO 和 I2C总线的设备元件的实现。介绍了项目架构的背景以及框架所用到的技术、框架和概念，并且分析了类似项目的优缺点。然后按照模块对系统功能做了一个大致的介绍，并且阐述了各模块的系统详细设计和实现。最后，对项目进行总结，阐释了框架的应用范围，并对框架的前景提出了设想和展望。

第一章：概述和前言部分，主要介绍了项目背景，当前物联网以及机器人开发领域热门框架的功能，分析了其优势与不足。简要描述了本项目的架构以及选择的运行环境。描述了本论文的主要工作和组织结构。

第二章：主要介绍本基于 Node.js与微服务架构的物联网框架在开发过程汇总使用的技术。

第三章：从项目概述、需求分析以及概要设计这三个方面，对Microbot 进行了详细的介绍。项目概述介绍了项目的模块划分，利用用例和需求列表对项目的功能性需求和非功能性需求进行分析，描述项目的整体的设计框架以及通过功能划分对项目进行模块设计和接口设计。

第四章：对本人负责的系统模块的详细设计和实现进行描述，包括对外接口以及模块内部的类设计图，同时对关键部分的代码进行分析解释。

第五章：对项目期间自己所做工作进行总结，探讨项目的缺点和不足，并指出项目的拓展前景和发展方向。

1. 关于国际电信联盟(ITU)的详细信息，参见：http://www.itu.int/en/pages/default.aspx [↑](#footnote-ref-1)
2. 关于RFID的详细信息，参见：http://www.baike.com/wiki/RFID [↑](#footnote-ref-2)
3. 关于微处理器的详细信息，参见：<http://www.mscbsc.com/cidian/baikeajh> [↑](#footnote-ref-3)
4. 关于JavaScript的详细信息，参见：http://www.w3school.com.cn/js/index.asp [↑](#footnote-ref-4)
5. 关于Nodejs的详细信息，参见：http://nodejs.cn/ [↑](#footnote-ref-5)
6. 关于 chrome V8引擎的详细信息，参见：https://developers.google.com/v8/?hl=zh-CN [↑](#footnote-ref-6)
7. 关于Cylon.js的详细信息，参见：https://cylonjs.com [↑](#footnote-ref-7)
8. 关于IoT.js的详细信息，参见：http://iotjs.net [↑](#footnote-ref-8)
9. 关于JerryScript的详细信息，参见：http://jerryscript.net [↑](#footnote-ref-9)
10. 关于ARDrone的详细信息，参见：https://www.parrot.com/us/drones/parrot-ardrone-20-elite-édition [↑](#footnote-ref-10)