基于 Node. js 的 BLE 可穿戴医疗设备管理中间件研究与实现

陈 刚[1] 闫 航[1] 张亚兵[1] 王俊红[2]

[1](郑州大学互联网医疗与健康服务协同创新中心 河南 郑州 450052)

[2](郑州大学护理学院 河南 郑州 450001)

摘 要 为了解决可穿戴医疗设备在通用计算平台下的通信与管理问题,更好地在智能病房系统中支持对病患的生命体征监测,设计基于 Node. js 的低功耗蓝牙 BLE(Bluetooth Low Energy)可穿戴医疗设备管理中间件,实现对 BLE 可穿戴医疗设备的设备连接、操作控制、数据处理等管理功能。同时还实现多设备的并发控制,满足医疗场景下多设备协同监测的需求。未来,各类可穿戴医疗设备借助于该系统可便捷地接入到智能病房系统中,进一步扩展可穿戴医疗设备在智能化医疗服务中的应用。

关键词 低功耗蓝牙 可穿戴医疗设备 健康监测 智能病房

中图分类号 TP391.7 **文献标识码** A DOI:10.3969/j. issn. 1000-386x. 2019. 06. 003

RESEARCH AND IMPLEMENTATION OF BLE WEARABLE MEDICAL DEVICE MANAGEMENT MIDDLEWARE BASED ON NODE. JS

Chen Gang[1] Yan Hang[1] Zhang Yabing[1] Wang Junhong[2]

[1](*Internet Medical and Health Service Collaborative Innovation Center, Zhengzhou University, Zhengzhou 450052, Henan, China*)

[2](*College of Nursing, Zhengzhou University, Zhengzhou 450001, Henan, China*)

Abstract In order to solve the communication and management problems of wearable medical devices under general-purpose computing platform, and better support the vital signs monitoring for patients in the intelligent ward system, we designed a bluetooth low energy(BLE) wearable medical device management middleware based on Node. js. It realized the management functions of device connection, operation control and data processing of BLE wearable medical device. It also realized the concurrent operation of multiple devices to meet the needs of multi-device collaborative monitoring in medical scenarios. In the future, wearable medical devices can be easily accessed into the smart ward system by means of this system, which further expand the application of wearable medical devices in intelligent medical services.

Keywords Bluetooth low energy(BLE) Wearable medical device Health monitoring Intelligent ward

0 引 言

可穿戴医疗设备是可以直接穿在身上或作为配件穿戴并能传输监测数据的一种计算设备[1]。各类可穿戴医疗设备在医院病房中发挥着关键的作用。首先,可穿戴医疗设备是护士对病人进行日常健康监测的重要工具;其次,在健康方面对特定的生命体征需要进行 24 h 监护[2]。由于采取有线通信方式的传统可穿戴医疗设备在治疗过程中导线会限制患者的活动,越来越多的可穿戴医疗设备开始采用无线的通信方式。低功耗蓝牙是蓝牙 4.0 开始在其标准的基础上针对低功耗应用进行优化后的标准,凭借快速连接与超低功耗的显著特点,已经广泛应用于医疗保健、可穿戴设备等领域[3]。

近年来绝大多数 BLE 可穿戴医疗设备由智能手

收稿日期:2018 - 12 - 14。国家重点研发计划项目(2017YFB1401200);河南省科技攻关项目(182102310137);赛尔网络下一代互联网技术创新项目(NGII20170716);2019 年河南省高等学校重点科研项目(19A520039)。**陈刚**,工程师,主研领域:网络安全,语义网络,互联网与移动医疗服务。**闫航**,硕士生。**张亚兵**,硕士生。**王俊红**,硕士生。

机管理。甘广辉等[4]提出了一种基于低功耗蓝牙的家用胎儿监护系统,实现了对胎心率、宫缩压和胎动生理信号的采集与传输,并通过手机对数据进行显示和存储以实现胎儿的实时监护。张金玲等[5]设计了一种基于蓝牙低功耗协议的手机无线心电和血氧健康监护系统,通过手机客户端程序实现人体心电和血氧参数的实时监控。而通用平台 Windows 或 Linux 上的开发应用较少,谢佳柏等[6]设计了一款基于 BLE 与 WebSocket 的数据网关,实现了低功耗蓝牙网络对手机、平板和笔记本电脑的兼容性,但是该数据网关采用的是 51 内核,处理能力较弱,仅支持同时连接 3 个简单的传感器节点。在医院病房下的健康监测场景中,多设备并发使用、健康数据处理、患者管理等操作需要强大的计算能力与通用平台编程需求。此外,由于各个厂家的设备开发标准体系不互通,智能手机中的一个 APP 只能连接同厂家的可穿戴医疗设备,难以跨厂家、跨平台管理多种设备[7]。

鉴于此,本文提出了基于 Node. js 的 BLE 可穿戴医疗设备管理系统。由于 Node. js 具备跨平台的特点,本系统可以灵活部署于 Windows 和 Linux 系统上。通过该中间件系统可以集中进行多种可穿戴医疗设备的管理,同时系统对应用程序提供调用接口,从而应用程序无需安装专用的 BLE 软件。本系统实时性强,响应速度快,支持并发操作,能够以一种更加高效的方式进行 BLE 可穿戴医疗设备的管理。

1　相关技术

1.1　Node. js 环境

Node. js 是一个基于 V8 引擎的 JavaScript 运行环境,采用轻量和高效的事件驱动、非阻塞 I/O 模型,常用于构建快速、可扩展的网络应用程序[8]。它由 Ryan Dahl 于 2009 年设计并提出,用异步 I/O 和事件驱动取代多线程,不仅大幅度提升了性能,还减少了多线程开发的复杂性。Node. js 不但采用 Chrome V8 作为引擎,还使用了高效的 libev 和 libeio 库支持事件驱动和异步 I/O,并在此基础上抽象出了层 libuv,实现了高性能的运行机制。

Node. js 基于事件触发的方式能够快速接收可穿戴医疗设备发来的监测数据,相比于其他平台有着显著的优势,并且能够适用于高并发的运作方式,所以本文所设计的中间件系统选择 Node 作为运行支撑环境。

1.2　BLE 协议简介

BLE 即蓝牙低功耗技术,是一种低成本、短距离、超低功耗的无线传输技术[3]。如图 1 所示为 BLE 协议栈,BLE 应用的开发所要关注的是通用访问配置(GAP)子协议与通用属性配置(GATT)子协议。

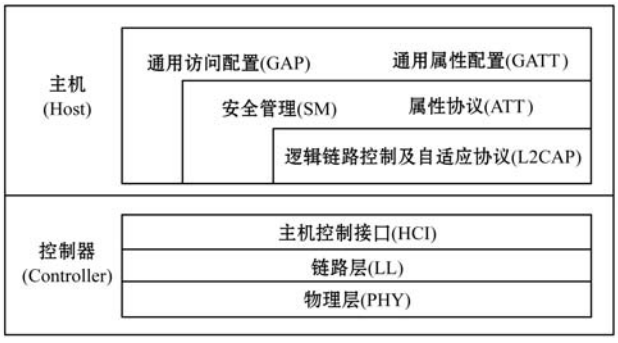


图 1　BLE 协议栈

GAP 协议负责控制设备连接和广播。通常将设备分为中心设备(Central)与外围设备(Peripheral),中心设备负责扫描并连接其他外围设备,外围设备则将自身的设备信息对外广播[9]。本设计方案中,可穿戴医疗设备作为外围设备,而部署中间件系统的计算机则作为中心设备。

GATT 协议是负责双方数据传输的通用规范,把各种属性表示为服务(Service)的集合,每个服务都有一个 128 bit 的 UUID 作为这个服务的标识。为了提高传输效率,蓝牙联盟也定义了较短的 16 bit 的 UUID 来使用。服务由若干个特征值(Characteristic)组成,每一个特征值也有唯一的 UUID 作为标识符。特征值由一个 value 和零个或多个对 value 的描述组成,相关的监测数据即存储于特征值中[9]。

1.3　WebSocket 通信技术

WebSocket 是基于 TCP 协议实现 Web 浏览器和服务器之间的实时双向通信技术[10]。它支持持久连接,客户端与服务器进行一次合法的握手建立 WebSocket 连接后,服务器便可以主动地向客户端发送数据[11]。本文采用 WebSocket 技术实现中间件系统与应用层之间的双向数据传输,在此基础上进一步封装管理可穿戴医疗设备的通信接口。

2　中间件架构设计

2.1　体系拓扑结构

本文提出的体系拓扑结构如图 2 所示,该体系由 BLE 可穿戴医疗设备、设备管理中间件和调用方组成。可穿戴医疗设备管理中间件是核心组成部分,一方面它通过 BLE 协议接入各类可穿戴医疗设备如血压计、体温计和心率计等,实现对可穿戴设备的统一管理与数据处理,同时协调设备资源的调度以实现多设备并

发运行。另一方面,中间件通过 WebSocket 通信方式响应调用方应用程序的操作请求,调用方通过中间件封装的控制指令完成对可穿戴医疗设备的操作以及健康监测数据的实时传输。中间件能够同时响应多个 Socket 客户端的请求,进一步适应了对可穿戴设备的并发操作,而调用方只需实现 Socket 客户端访问中间件即可,能够兼容各主流平台如 Windows、Android 以及 MacOS 等。

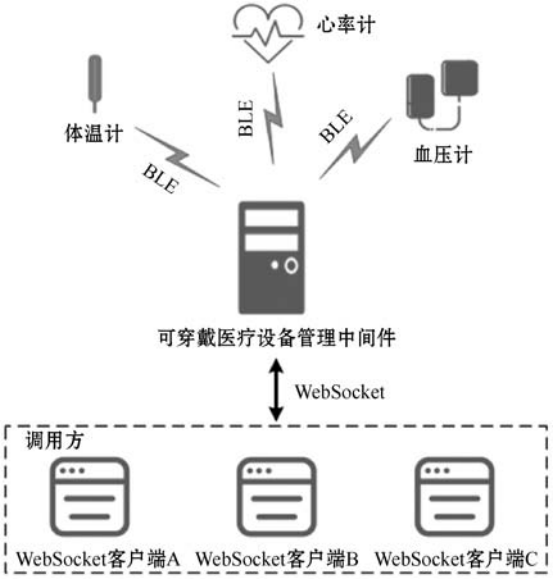


图 2 体系结构拓扑

2.2 软件架构

本文设计的中间件系统软件架构如图 3 所示。Node 环境下的程序保持单进程单线程执行,系统由设备驱动层、任务调度层、服务接口层组成。设备驱动层主要是 BLE 开发程序,通过 BLE 协议栈实现对低功耗可穿戴医疗设备的基本操作,包括设备扫描、连接、控制以及获取监测数据。任务调度层由用户指令解析程序、多设备并发调度程序和监测数据处理程序组成。其中,用户指令解析程序处理客户端发来的操作指令,进而执行相应的操作方式。设备并发调度程序负责调度设备资源与客户端请求,用以实现高效的设备并发操作。监测数据处理程序对设备驱动层获取的原始监测数据进行规范化处理,将数据转换为直观可读的健康数据。服务接口层主要是 WebSocket 服务器程序,用来提供中间件与调用者交互的接口,实现两者的双向数据传输。WebSocket 服务器程序一方面接收调用者发来的消息并进行 json 解析,将解析后的数据传递给任务调度层;另一方面将任务调度层发来的健康数据与连接状态封装为 json 格式并发送给调用者。此外,系统还包含一个全局的进程守护程序,负责解决 Node 可能出现的异常进而防止崩溃,监视全局异常的发生,包括回调函数发生的异常,保证中间件系统的可靠运行。

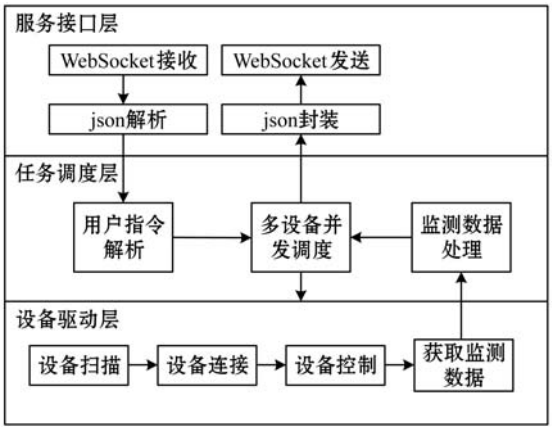


图 3 中间件软件架构

3 系统实现

本系统以生活中典型的 BLE 血压计、BLE 心率计和 BLE 体温计作为 BLE 可穿戴医疗设备进行系统的实现与演示。血压计、心率计、体温计是三种不同类型的可穿戴医疗设备,也是进行疾病预防与诊断的重要设备。血压计与非接触式体温计由专业厂家生产,而心率计为实验室自制,以上设备都实现了 BLE4.0 协议,具有极低的运行和待机功耗。

3.1 系统的环境准备

系统基于 Node 平台采用 JS 语言编写,可以将 Node 环境部署于 Windows 或 Linux 系统上从而实现跨平台的可穿戴医疗设备接入。首先,部署系统的平台需要通过外接蓝牙适配器来实现 BLE 功能,中间件系统在 BLE 开发中就充当了中心设备的角色。

近年来随着 Node.js 的广泛应用,Node 社区也涌现了很多优秀的资源。第三方库 noble 是基于 Node 环境封装的低功耗蓝牙库,它基于 GAP 协议与 GATT 协议对 BLE 蓝牙的基础功能进行了封装,并结合 C ++ 与 python 语言进行了低功耗蓝牙底层协议的开发。ws 库是 Node 环境中流行的用以实现 WebSocket 协议的开发包,它具有快速、易用和稳定的特点,并且同时实现了服务器与客户端。因此本文基于 noble 库进一步丰富低功耗医疗类型设备的开发工作,实现对可穿戴医疗设备的扫描、连接、特征值发现等 BLE 开发基本操作,引入 ws 库实现 WebSocket 传输协议,示意代码如下:

```
const WebSocket = require('ws');
var noble = require('./lib/noble');
```

3.2 可穿戴医疗设备基本操作实现

中间件系统在低功耗设备开发中实现的基础功能

依次为外围设备扫描、设备连接、发现服务、发现特征值以及数据写入与读取,相应地就可以完成系统对可穿戴医疗设备的连接、测量控制与数据传输工作。

3.2.1　可穿戴医疗设备开发基本流程

本文借助 noble 库实现低功耗蓝牙的基础功能,可穿戴医疗设备的基本操作开发主要包括以下步骤:

(1) 中间件系统开启蓝牙扫描。通过调用方法 noble.startScanning(serviceUUID)扫描相应的 BLE 设备,参数为所要连接设备主服务的 UUID 号。

(2) 设备连接。成功扫描到所要连接的 BLE 可穿戴医疗设备后会触发设备发现的事件,在事件的回调函数中通过返回的 peripheral 对象执行设备连接操作并停止蓝牙扫描。peripheral 即是通过扫描获取到的外围设备对象,示意代码如下所示:

```
noble.on('discover', function(peripheral) {
noble.stopScanning();                          //停止扫描
peripheral.connect(function(err) {…})   //连接 BLE 设备
})
```

连接成功后才能进行后续的一系列操作,此时可穿戴医疗设备被系统占用,且不可被其他中心设备所连接。

(3) 发现服务。系统要获取的服务是与健康监测数据相关的服务,通过方法 peripheral.discoverServices(serviceUUID)发现服务,参数为所要获取服务的 UUID 值。

(4) 发现特征值。在发现服务方法的回调函数中通过返回的 service 对象调用 service.discoverCharacteris tics()方法来发现特征值,此方法的回调函数中返回该服务下的所有特征值,然后遍历所有特征值来匹配到健康数据相关的特征值,主要包括 write 特征值和 notify 特征值。write 特征值用来向可穿戴医疗设备发送数据,而 notify 特征值则是从可穿戴医疗设备中获取健康监测数据。

(5) 设备控制与数据读取。通过 write 特征值调用方法 characteristic.write()并传入控制指令来控制可穿戴医疗设备,主要包括启动设备、关闭设备的命令。通过 notify 特征值调用 characteristic.subscribe()方法启动 BLE 设备特征值变化时的 notify 功能,实现对监测数据的监听。如果可穿戴医疗设备有新的数据变化,notify 特征值的回调函数就会触发,系统从回调函数中读取监测数据。

系统从可穿戴医疗设备直接收到的数据为原始的字节流数据,必须要对数据进行解析以及规范化处理。BLE 健康数据传输格式一般由起始位、数据位和结束位组成,需要从数据位中获取健康数据[12]。然而,不同厂家、不同类型的设备对健康数据也存在着不同的编码方式,必须针对具体的可穿戴医疗设备采取对应的算法进行健康数据的解析与处理。通常情况下,数据解析与处理的流程如图 4 所示。

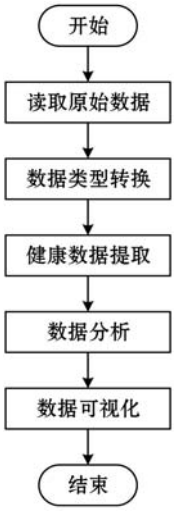


图 4　数据处理流程

(6) 断开连接。完成测量之后通过 write 特征值向可穿戴医疗设备发送关闭设备的指令,之后通过步骤(2)中的 peripheral 对象执行 peripheral.disconnect()方法断开连接,解除设备占用并释放系统资源。

3.2.2　可穿戴医疗设备的控制

系统与可穿戴医疗设备建立连接并且匹配好读写相关的特征值后,系统将控制指令写入 write 特征值对可穿戴医疗设备进行控制,包括开始测量、停止测量、关闭设备等常用的操作。不同的可穿戴医疗设备对应着不同的指令编码,写入的数据应转换为 byte 类型的数组。

根据可穿戴医疗设备实际控制需求,本文所使用的血压计和心率计的开始测量、关闭设备指令以十六进制表示的编码如表 1 所示。而系统对体温计只提供监测数据、关闭连接的控制指令,开始测量、关闭设备的操作由设备自身完成。

表 1　血压计与心率计控制编码

设备类型	开启测量指令	关闭设备指令
血压计	[0xFD,0xFD,0xFA,0x05, 0X0D, 0x0A]	[0x73,0x74,0x61, 0x72,0x74]
心率计	[0xFD,0xFD,0xFE,0x06, 0X0D, 0x0A]	[0x73,0x74,0x6F, 0x70]

3.3　多设备并发实现

相较于大多数智能手机上的可穿戴医疗设备管理方式,本系统实现了多用户、多设备并发使用的功能。设备并发使用包含两种场景,一是单个用户同时使用多台可穿戴医疗设备,二是多个用户同时使用各自的可穿戴医疗设备。在可穿戴医疗设备基本操作实现的基础上,本文提出了以设备池为中心的调度策略来解

决多用户、多设备并发使用问题。结合实际场景考虑，一台可穿戴医疗设备同时只能服务一个用户，当用户选择某台设备时，分配给用户与该设备相关的资源，直到用户使用完毕后才能释放资源，设备调度工作流程如图 5 所示。

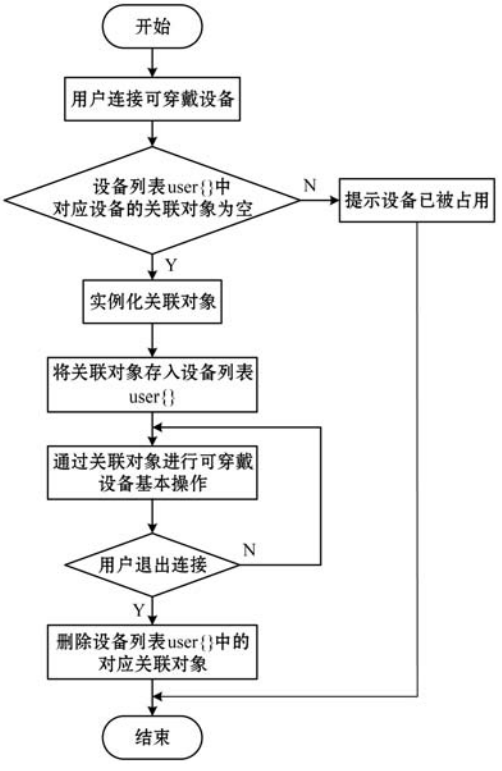


图 5 设备调度工作流程图

本研究处理设备多连接的方法是为每一个已连接的客户端生成一个包含四个关键属性的对象，对象包含 WebSocket 客户端接口、设备对象、设备 write 特征值和设备 notify 特征值，实例化代码如下所示：

```
function user_info( user_ws, peripheral,ble_notify, ble_write) {
    this. user_ws = user_ws;
    this. peripheral = peripheral;
    this. ble_notify = ble_notify;
    this. ble_write = ble_write;
}
```

此对象的作用是保存控制可穿戴设备所需要的关键资源，并建立调用者与可穿戴设备的对应关系。多用户同时进行各自的健康监测时，每台可穿戴医疗设备都会从自身的关联对象中进行设备控制、数据处理以及数据传输工作，能够保证用户只与所使用的可穿戴医疗设备进行交互，不同设备之间的操作互不影响。当一个用户同时使用多台可穿戴医疗设备时，则建立多个 user_info 对象来关联多台设备，用户同时与多台设备进行互不影响的交互，实现多设备联动健康监测。具体方法如下：

（1）客户端选择可穿戴医疗设备，中间件系统完成设备连接、服务与特征值发现后，调用函数 user_info()实例化一个对象实现用户与设备之间的关联。同时将该对象以 key-value 的形式保存在 user{}列表中，key 设置为可穿戴医疗设备名称，value 为用户关联对象，user{}设备池中保存所有用户关联对象。

（2）客户端发起测量指令后，在 user{}列表中索引对应设备的关联对象，调用关联对象的 ble_write 特征值启动测量任务。中间件系统通过关联对象的 ble_notify 特征值监听可穿戴医疗设备发送来的血压或心率等监测数据，同时调用关联对象的 WebSocket 接口发送给使用此设备的用户。

（3）测量任务结束后，中间件系统通过关联对象的 peripheral 属性来执行 peripheral. disconnect() 函数断开所连接的健康设备，解除设备资源占用。

（4）最后将 user{}列表中对应设备的关联对象清空，该可穿戴医疗设备可重新分配给其他的客户端。

3.4 可穿戴设备管理接口设计

本文基于 WebSocket 通信技术为调用方提供管理设备的接口，中间件系统在 WebSocket 通信中作为服务器端，而调用方的应用程序则作为客户端。如图 6 所示为中间件设备管理流程图，首先使用方法 const wss = new WebSocket. Server({port:2800}) 创建 WebSocket 服务器并将端口设备为 2800。服务器端基于事件驱动完成客户端的请求，主要事件函数包括'connection'、'message'、'close'，分别对应连接建立、收到数据、连接断开的事件回调[13]。

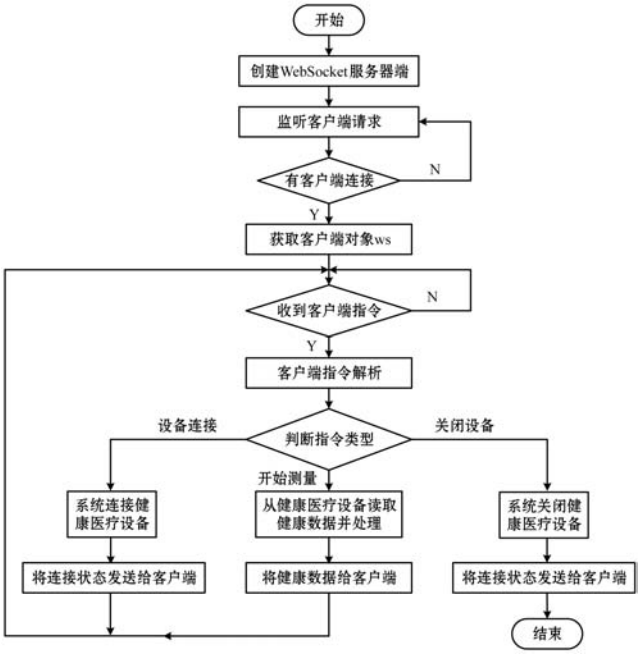


图 6 可穿戴医疗设备管理流程图

服务器实时处于监听状态，当有客户端建立连接后触发'connection'事件并返回客户端对象 ws。服务器

在'message'事件回调中解析用户指令,中间件提供的操作指令接口分别为设备连接、开始测量和关闭设备。其中,设备连接指令用来连接用户所选择的可穿戴医疗设备,并将连接状态发送给用户。开始测量指令则启动可穿戴医疗设备以进行生命体征测量,中间件实时获取监测数据并进一步处理后发送给用户。关闭设备指令是将可穿戴医疗设备关闭并断开与设备之间的蓝牙连接以释放资源,同时将设备连接状态发送给用户。服务器通过 ws. send(msg)方法向客户端发送数据,参数 msg 为发送的内容并封装为 JSON 格式,msg 的内容为设备连接状态和健康监测数据。客户端退出后触发'close'事件,在此回调函数中首先检查用户所关联的设备是否关闭,若存在没有关闭的设备则强制关闭以上设备,最后进行设备资源的释放。

客户端与服务器建立连接之后就可以进行双向通信,双方只需要进行一次请求/响应操作,之后每次通信传输的都为控制指令和健康数据,加快了数据传输的速度,进一步提高数据传输的实时性[14]。

4　中间件系统测试

本文将中间件系统部署于 Windows10 操作系统,首先从 Windows 命令行程序中进入系统项目的根目录,执行 node healthy_central. js 运行中间件程序,中间件系统就会以后台的方式挂起并开始监听 2800 端口。

本系统以接入的血压计、体温计和心率计来进行演示,设备实物图如图 7 所示。BLE 血压计为专业厂家深圳奥又美公司生产;BLE 体温计由深圳智子云守护科技公司生产的新一代非接触式体温计;BLE 心率计由团队通过心率传感器、温度传感器、电源和低功耗蓝牙 RF 芯片制作而成。血压计能够测量实时血压值并得出舒张压、收缩压和平均心率,非接触式体温计能够快速、精准测量体温,心率计能够实时进行心率和室内温度的测量。

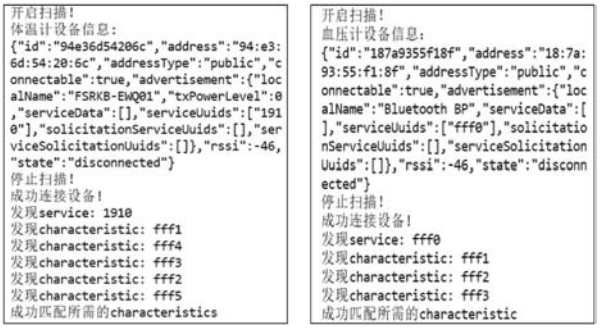
图 7　设备实物图

目前主流的浏览器如 Chrome、Firefox 等均已支持 WebSocket API,所以本文采用 HTML5 + JavaScript 语言编写了简易的 Web 界面通过 WebSocket 客户端来访问系统接口,从而作为调用方来进行便捷的演示。该客户端同时适应了电脑与移动端设备,页面设计如图 8 所示。客户端通过浏览器访问中间件系统来管理可穿戴医疗设备,主要实现了 onopen( )、onmessage( )、onclose( )等回调函数,触发事件分别为连接建立、接收中间件数据、连接关闭,并通过 send( ) 函数主动向中间件发送数据。当客户端与中间件建立连接后,用户可发送指令来管理可穿戴医疗设备,客户端页面会实时显示设备连接状态与健康数据。

图 8　客户端界面设计

4.1　可穿戴医疗设备连接测试

本文通过所设计的客户端分别连接 BLE 体温计、血压计和心率计设备,中间件系统立即执行外围设备扫描、设备连接以及特征值发现操作。为便于观察,采用 VS Code 编辑器进行输出结果的显示,体温计与血压计连接操作的中间件系统执行结果如图 9 所示,系统扫描设备后返回设备的广播信息如设备名、设备 MAC 地址、设备连接状态等,测试表明中间件系统均能够快速扫描、连接可穿戴医疗设备,并能够成功匹配操作设备所需的特征值。



(a) 体温计　　(b) 血压计

图 9　可穿戴医疗设备连接状态图

4.2 可穿戴医疗设备并发操作以及数据传输测试

为更好地演示客户端跨平台的效果以及多设备并发操作的性能,在 Windows10 平台中通过 Chrome 浏览器启动一个客户端连接心率计设备,同时在两台 Android7.1 移动终端通过 QQ 浏览器启动客户端分别连接体温计与血压计设备。三个客户端在同时段进行各自的健康监测,通过 Date()方法获取当前的时间进行对照。可穿戴医疗设备健康监测的效果如图 10 所示。测试表明三个客户端并发运行下均能够独立地操作可穿戴医疗设备,中间件正确执行连接设备、开始测量和关闭设备的指令,解析健康监测数据并实时传输到客户端。

(a) 体温计监测数据 (b) 血压计监测数据 (c) 心率计监测数据

图 10 并发操作与数据传输测试

中间件系统各项测试结果表明,基于 Node.js 的可穿戴医疗设备管理系统能够进行便捷有效的设备操控,系统运行稳定,能够正常完成可穿戴医疗设备连接、控制、数据传输等工作。相比于安卓,在通用平台上进行可穿戴医疗设备的管理有着更强的响应速度和更快的数据处理能力,并能够实现设备并发的操作,非常适合有着计算需求和并发测量的应用场景。

5 结 语

本文提出并开发了一种适用于通用平台的可穿戴医疗设备管理中间件,实现了病房场景下多传感器设备的综合管理功能,进而扩展了可穿戴医疗设备的应用场景。基于本文工作,后续将会进一步增强对传感器设备的管理能力,通过定义标准化接口来兼容更多厂家、类型的可穿戴医疗设备,设计兼顾结构化、半结构化数据的数据预处理模块来支持复杂多样的传感器数据类型,并进一步优化支持多传感器的并发管理任务,为可穿戴医疗设备的管理提供一种灵活易用、更趋智能化的方式。

参 考 文 献

[1] 樊瑜波. 可穿戴医疗/健康技术——生物医学工程的机遇和挑战[J]. 生物医学工程学杂志, 2016,33(1):1-1.

[2] 王瑞, 吕蓉, 梁涛. 可穿戴设备在疾病管理中的应用进展[J]. 中华护理杂志, 2018(1):114-116.

[3] 王启林, 李小鹏, 郁滨,等. 基于连接认证的低功耗蓝牙泛洪攻击防御方案[J]. 计算机应用研究, 2017, 34(2):499-502.

[4] 甘广辉, 童蕾, 陈超敏. 基于低功耗蓝牙网络的家用胎儿监护系统设计[J]. 电子技术应用,2015,41(8):34-36.

[5] 张金玲, 高志新. iOS 平台无线健康监护系统[J]. 北京邮电大学学报, 2016,39(6): 17-21.

[6] 谢佳柏, 陈贤祥, 胡欣宇, 等. 基于低功耗蓝牙和 WebSocket 的物联网数据网关[J]. 仪表技术与传感器, 2016(1):76-78.

[7] Zachariah T, Klugman N, Campbell B, et al. The Internet of Things Has a Gateway Problem[C]//Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications. ACM, 2015:27-32.

[8] 王伶俐, 张传国. 基于 NodeJS + Express 框架的轻应用定制平台的设计与实现[J]. 计算机科学, 2017, 44(b11):596-599.

[9] Gomez C, Oller J, Paradells J. Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology[J]. Sensors, 2012, 12(9):11734-11753.

[10] 陈炜, 苏厚勤, 柴炯. 基于 WebSocket 技术水文资源监管系统的研究与实现[J]. 计算机应用与软件,2016,33(3):104-108,113.

[11] Ma K, Sun R. Introducing WebSocket-Based Real-Time Monitoring System for Remote Intelligent Buildings[J]. International Journal of Distributed Sensor Networks, 2013, 2013(12):867693.

[12] 王骥, 任肖丽. 基于蓝牙低功耗技术的智能健康监测手表系统[J]. 生物医学工程学杂志, 2017(4):557-564.

[13] Zhang W, Stoll R, Stoll N, et al. An mHealth Monitoring System for Telemedicine Based on WebSocket Wireless Communication[J]. Journal of Networks,2013,8(4):955-962.

[14] Pimentel V, Nickerson B G. Communicating and Displaying Real-Time Data with WebSocket[J]. IEEE Internet Computing, 2012, 16(4):45-53.