MA35D0 edge device WEB server example on Linux

NuMicro® 64/32位系列微控制器范例程序代码介绍

文件信息

应用简述	本范例程序说明如何于 MA35D0 系列开发平台透过网页接口进行远程监控和控制，利用 Node.js 和 HTML5 实现对系统指针和跨多个装置管理的实时更新。
BSP 版本	Linux-5.10.x
开发平台	NuMaker-IoT-MA35D0 V1.0

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. 概述

此范例代码的目的是提供在系统内使用 Node.js 和 HTML5 建构远程控制接口的方式。此范例代码能够透过网页浏览器联机，提供系统状态的实时更新，如装置 IP 地址、UID、CPU 负载、内存容量，以及 IO 状态和控制。另外，它还具备监控重要事件和进行远程 LED 控制的能力。此范例的特色在于支持多台 MA35D0 EVB 装置之间的信息共享，增强协同作业和管理。

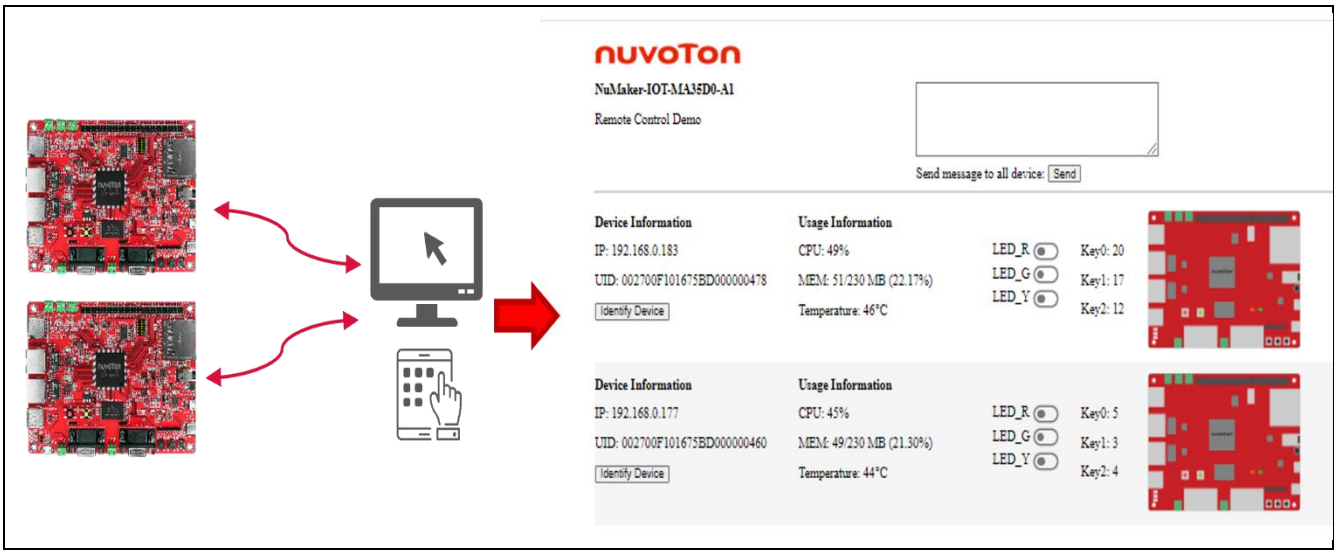

Figure 1-1 物联网装置远程监控与控制概览

1.1 软件架构

- **init.sh** 初始化示范环境设定并启动 Web 服务器(server.js)，并呼叫 set_serverIP.sh 为客户端 JavaScript 档案分配 IP。
- **set_serverIP.sh** 设定 public 目录中的 HTML layout 和客户端 JavaScript 档案。
- **server.js** 服务器端代码。
- **sys_monitor.sh** 负责获取 CPU、内存和温度等信息。
- **LEDControl.sh** 控制 LED 的开启/关闭状态。
- **identify_device.sh** 装置已收到对应的命令后闪烁 LED 6 次。
- **get_dev_info.sh** 取得装置的 IP 和 UID 信息。

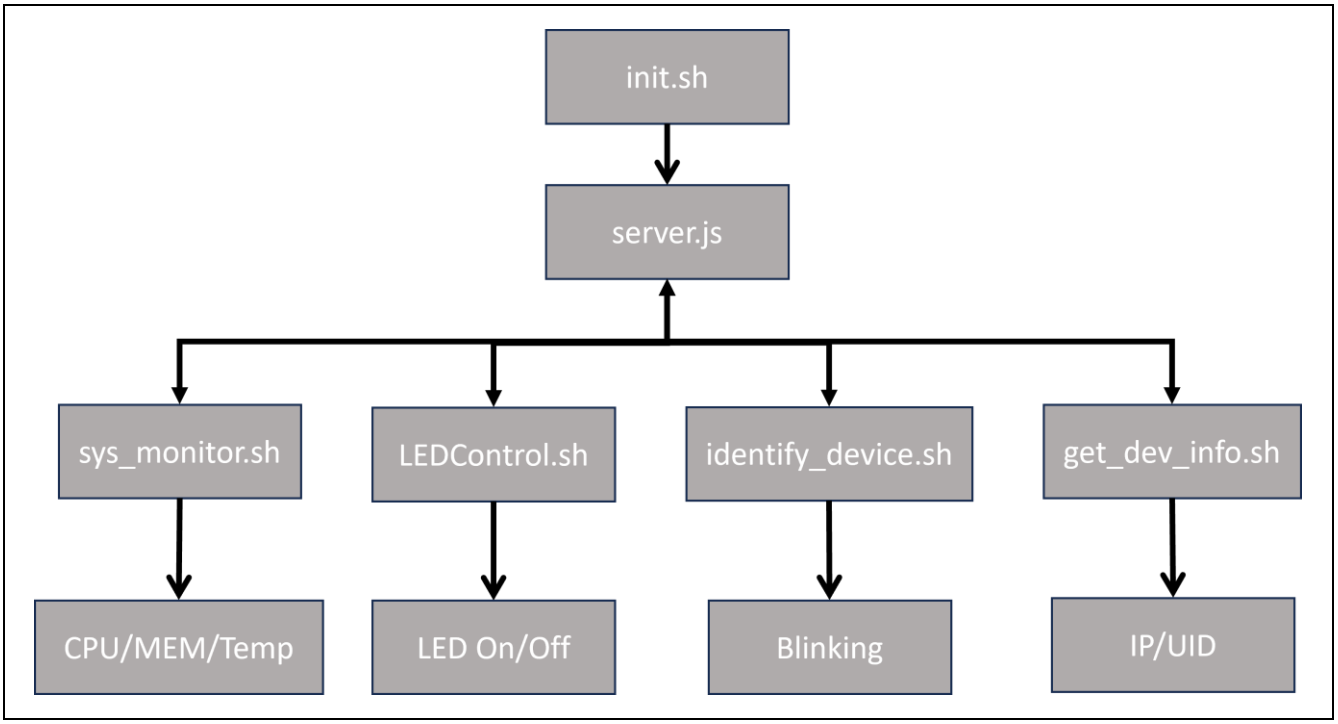

Figure 1-2 软件框架

1.2 环境设定

MA35D0环境架设请参考 Nuvoton MA35D0 Evaluation Board Quick Start.

Buildroot

在 Buildroot menuconfig 启用 MA35 支持所需的套件，请按照以下步骤操作：

1. 执行 make menuconfig 命令来开启 Buildroot 设定选单。
2. 透过在套件选择选单中选择对应选项并按下空格键来启用每个套件。可以使用'/'键按名称搜寻套件。
    1) 将 BR2_ROOTFS_OVERLAY="…" 修改为 BR2_ROOTFS_OVERLAY="*board/nuvoton/ma35d0/overlay*"。
    2) 启用 BR2_PACKAGE_NODEJS 以包含 Node.js 套件。
    3) 启用 BR2_PACKAGE_ NODEJS_NPM 以包含 Node.js 套件。
    4) 启用 BR2_PACKAGE_ NODEJS_MODULES_ADDITIONAL= “express ws” 以包含” express” 与 “ws” Nodes.js 套件 。

3. 选定需要的套件后，退出 menuconfig 并储存设定。

4. 在 Buildroot 执行 命令编译固件. 完成后使用 NuWriter 将固件下载至 MA35D0 开发板。

Linux Device Tree

禁用 gpio_keys_test 驱动程序，直接读取缓存器来获取 IO 状态。

ma35d0-iot-256m.dts

```
gpio_keys_test {
        compatible = "gpio-keys";
        status = "disabled"
  …
}
```

1.3 为范例代码建立 Overlay 文件夹

在路径: ***${Buildroot TOPDIR}/board/nuvoton/ma35d0/*** 建立一个新文件夹，命名为"overlay"，将范例代码子文件夹(“opt”与 “etc”)放置在此目录下，在编译之前，Buildroot 会将 overlay 文件夹的内容复制到 *output/target/*路径下。

完成 1.2 和 1.3 步骤后，请进入 ***${Buildroot}/*** 目录并运行 make 命令来编译固件

编译完成后，使用 **NuWriter** 将固件下载至 MA35D0 开发板。

2. 代码介绍

2.1 系统初始化 “inittab”

inittab 文件有以下功能：

- **系统初始**：设定系统初始的过程，这包括挂载 proc 文件系统，重新挂载根文件系统为可擦写模式，创建 */dev/pts* 和 */dev/shm* 目录，挂载所有在 /etc/fstab 文件中指定的文件系统，开启交换分区（swap）等。
- **创建符号链接**：创建了指向当前处理进程文件描述符的 */dev/fd* 和标准输入/输出/错误的符号链接。
- **设定主机名**：使用 */etc/hostname* 文件中的内容来设定系统的主机名。
- **执行自定义初始化脚本**：执行 */opt/init.sh* 脚本，用于特定硬件或应用程序的自定义初始化指令。
- **执行 rc 脚本**：执行 */etc/init.d/rcS* 脚本，这是系统提供的标准运行级别初始化脚本。
- **配置串口控制面板**：在 ttyS0 串口上启动 getty 程序，以便可以登录。

2.2 init.sh

开机过程中设定 GPIO 引脚、网络配置，并启动 Node.js 应用程序：

- **设定 GPIO**： 这个脚本定义了两个函数 export_gpio 和 set_direction，分别用于启用特定的 GPIO 并设定其方向。
- **连接网络**： 脚本会执行 udhcpc 命令，透过 DHCP 协议，闲置网络接口 eth0 从 DHCP 服务器取得一个 IP 地址。如果这个命令失败，脚本会输出一个错误消息并终止执行。
- **导出 Node.js 路模块**：通过 export NODE_PATH=/usr/lib/node_modules 指令，这行命令设置了 Node.js 模块的路径环境变量。
- **运行服务器应用程序**：通过 node server.js & 命令在后台启动一个 Node.js 服务器应用程序。

2.3 set_serverIP.sh

将目前 eth0 网络接口的 IP 地址更新到 JavaScript 档案(/opt/public/index.js) 。

- **取得 IP 地址**：利用 ifconfig 命令获取 eth0 网络接口的 IP 地址，并储存到 eth0_IP 变量中。
- **检查 IP 地址是否取得成功**：如果没有成功取得 IP 地址（即 eth0_IP 变量内容为空），则输出错误消息并结束程序执行。
- **检查 /opt/public/index.js 是否存在**：如果文件不存在，则输出错误消息并结束程序执行。
- **创建临时文件**：使用 mktemp 命令创建一个临时文件，并把路径储存到 TMP_FILE 变量中。
- **输入新的第一行**：将含有 eth0 IP 地址的新第一行写入临时文件。
- **释出 index.js 的原始内容**：使用 tail 命令将 /opt/public/index.js 文件的第二行开始的所有内容追加到临时文件。
- **用更改后的文件覆盖原文件**：使用 mv 命令将临时文件重命名为 /opt/public/index.js，达到替换原始 index.js 文件的目的。

2.4 server.js

设置服务器应用，提供与客户端建立 WebSocket 联机，使用 shell 脚本完成与客户端之间的通讯，包含监控和控制系统的 IO、系统状态、LED 以及装置识别。

- **引入模块**：引入所需的 Express、WebSocket 以及 child process 模块，以便创建网页服务器并能够执行 Shell 脚本。
- **构建 Express 应用**：设置一个基于 Express 的应用，用于处理 HTTP 请求和提供静态档案服务。
- **监听端口**：在 3000 端口启动 HTTP 服务器，以便客户端可以通过网页访问服务。
- **设置服务器 IP**：执行 Shell 脚本 set_serverIP.sh 来设定服务器的 IP 地址，这样客户端就可以连接到正确的服务器。
- **获取装置信息**：执行 get_dev_info.sh 脚本来获取装置的 IP 地址和 UID，并储存于 devInfo 对象中，这些信息用于识别连接的装置。

- **启动 WebSocket 服务器：**在 121 端口建立一个 WebSocket 服务器，用于与客户端建立实时通信。
- **处理客户端连接和断开：**当客户端连接或断开时，会有相应的处理函数，用于追踪连接的客户端并在有必要时更新客户端的状态。
- **定时发送 IO 和系统状态：**使用定时器定期检查和发送装置的 IO 状态与系统状态给所有连接的客户端，确保客户端了解最新的装置状态。
- **处理 WebSocket 消息：**定义了处理客户端发来的 WebSocket 消息的函数，根据收到的指令执行相应的装置控制或反馈操作。

2.5 sys_monitor.sh

监控 GPIO 引脚、LED，以及 CPU、RAM 等各种系统资源的状态：

- **read_gpio_value:** 读取指定 GPIO 引脚的值。
- **io_status:** 输出连接到 GPIO 的按键状态，以及连接到三个 LED 的亮度值。
- **sys_status:** 输出系统的 CPU 使用率、RAM 使用情况和温度值。
- **cpu_usage:** 使用 top 命令取得 CPU 的使用情况。
- **ram_usage:** 使用 free 命令取得 RAM 的使用情况。

2.6 LEDControl.sh

控制名称为 LED0、LED1 或 LED2 的三个 LED 灯。

- 检查用户是否输入一个合法的 LED 名称，也就是 "LED0"、"LED1" 或 "LED2" 中的一个。如果没有，脚本就会输出错误消息 "Invalid LED"
- 如果输入的 LED 名称是合法的，脚本就会验证第二个输入参数，也就是 LED 的目标状态。如果目标状态是 "On"，脚本就会在设定在对应的 */sys/class*

2.7 identify_device.sh

让三个 LED 灯（LED0、LED1 和 LED2）闪烁效果。

2.8 get_dev_info.sh

取得装置的 eth0 网络接口的 IP 地址和唯一标识符（UID）：

- **获取 eth0 的 IP 地址**：使用 ifconfig 命令取得 eth0 的 IP 地址，并将其储存在 eth0_IP 变量。
- **读取内存地址以形成 UID**：使用 devmem 命令读取特定的三个内存地址，然后把这些地址串接起来形成 UID。

2.9 index.html

网页版面设计，这个网页主要包括以下部分：

- **设备信息**：这部分将显示设备的 IP 地址和唯一标识符（UID）。阅读器可以将这些信息用于辨认连接的设备。
- **设备使用信息**：这部分将显示设备的 CPU 使用率、内存使用状况和温度。用户可以用这些信息来了解设备的运行状况。
- **LED 灯和按键状态**：这部分允许用户远程控制并查看设备上的 LED 灯状态，包含红色（LED0）、绿色（LED1）及黄色（LED2）的灯。「On」代表打开，「Off」代表关闭。网页上也将显示 Key0，Key1 和 Key2 的当前状态。
- **发送消息给所有设备**：用户可以通过在文本区域中输入消息，然后点击'发送'按钮来将消息发送给所有设备。

2.10 index.js

在本地网络上透过WebSocket连接发现和互动多个设备。它允许用户控制LED灯，识别设备，并向设备的控制面板发送消息。用户界面会根据从设备收到的状态信息动态更新，主要功能如下：

- **建立 WebSocket 联机**：程序代码会扫描特定的 IP 范围，并尝试对每个 IP 地址建立 WebSocket 联机。若成功建立联机，程序代码便会建立对应的设备对象并存放于 deviceProfile 数组中。
- **处理 WebSocket 消息**：对每个建立的 WebSocket 联机都注册了事件处理器来处理开启、错误、关闭和接收消息的情况。如果 WebSocket 从服务器收到消息，程序代码会进行解析并更新对应设备的状态和用户接口。

- **动态生成设备状态接口**：透过 duplicateElement 函数，为每个发现的设备复制并插入一个新的设备状态块到网页上，并且该设备状态块会显示该设备的 IP、UID、按键次数和温度等信息。
- **设备识别和 LED 控制**：identifyDevice 与 controlLED 函数分别用于发送识别设备的请求和控制 LED 灯的开或关。
- 发送信息至所有设备：sendMessageToAllDevices 函数会向所有已连接的设备发送特定的消息。

3. 软件与硬件需求

3.1 软件需求

- BSP 版本
  - Linux-5.10.x

3.2 硬件需求

- 电路组件
  - NuMaker-IoT-MA35D0 V1.0

4. 目录信息

EC_MA35D0_Edge_Device_WEB_Server_Example_On_Linux_V1.00	
SampleCode	
etc	
inittab	Script to handle boot-up process
opt	
get_dev_info.sh	Procedure to obtain IP and UID
identify_device.sh	Logic to handle LED blinking
init.sh	Custom script for handling boot procedures
LEDControl.sh	Script for managing LEDs
server.js	Server-side application
set_serverIP.sh	Update IP in the index.js file
sys_monitor.sh	Procedure to obtain CPU/MEM usage and I/O status
Images	Images for the web interface
public	Design of the web page layout
index.html	The client-side application
index.js	

Figure 4-1 目录信息

5. 范例程序执行

初始设置和 LED 闪灯行为:

将 MA35D0 的以太网络端口(RMII0)连接到路由器，并将计算机端连接到控制面板，并使用 TYPE C 供电。开机后，默认系统将成功运行该示例，且 LED 将同步闪烁 6 次。

通过控制面板和浏览器访问 MA35D0 装置信息:

如果演示运行成功，则会显示 IP 地址和相关信息:
使用相同网域的 PC 或手机，输入${MA35D0 IP}:3000 进入浏览器。

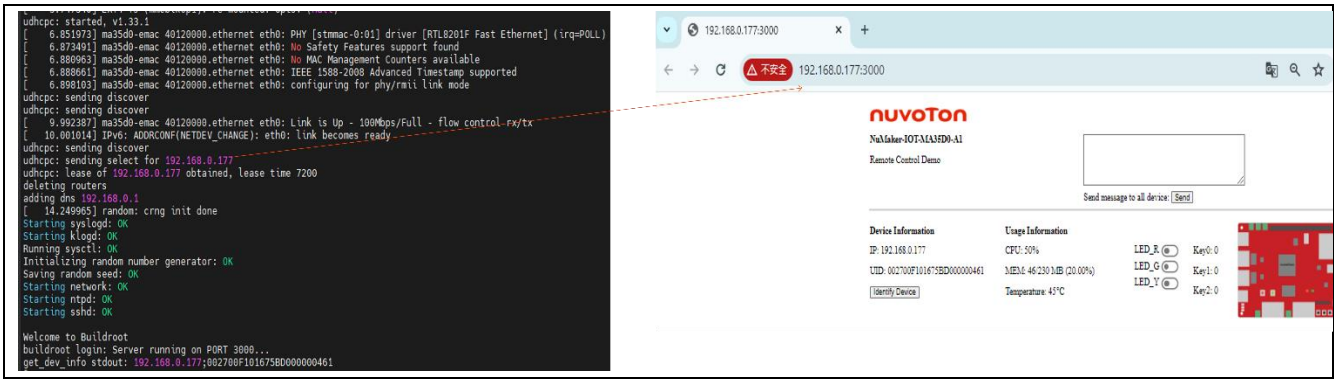

Figure 5-1 访问 MA35D0 装备信息

广播:

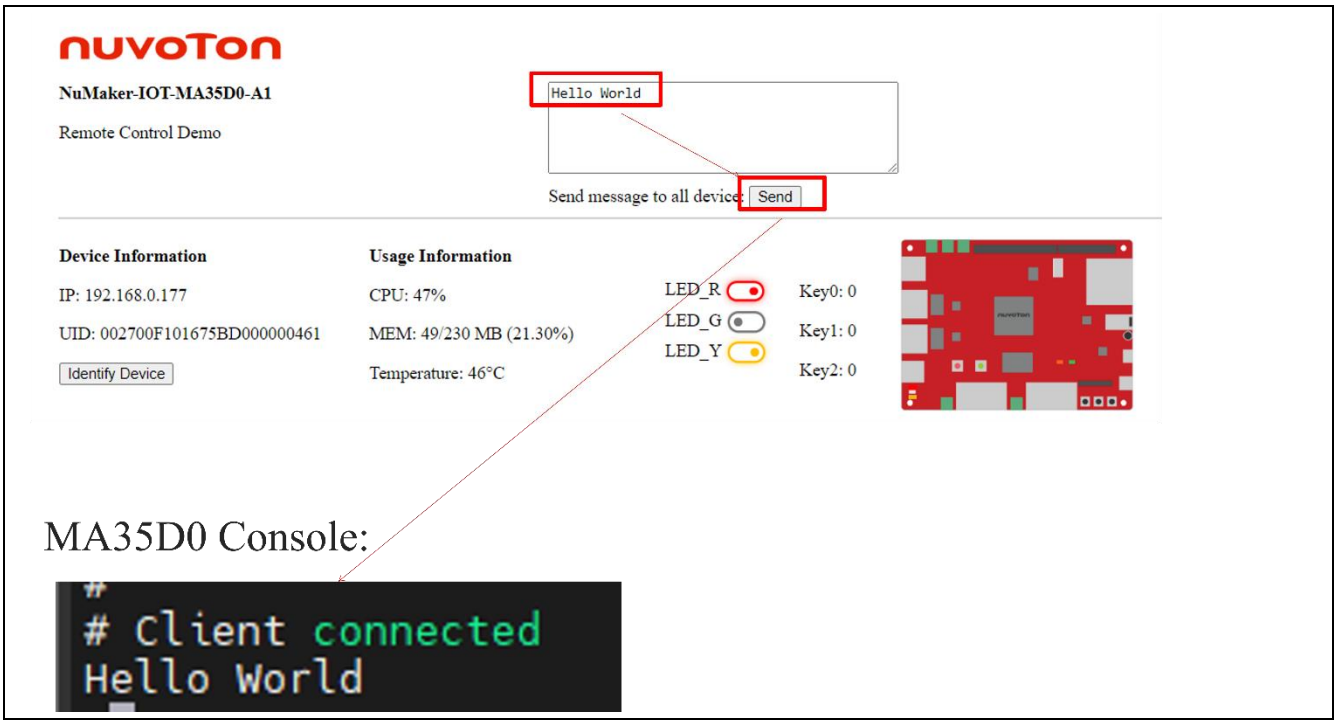

Figure 5-2 广播

控制与按键状态:

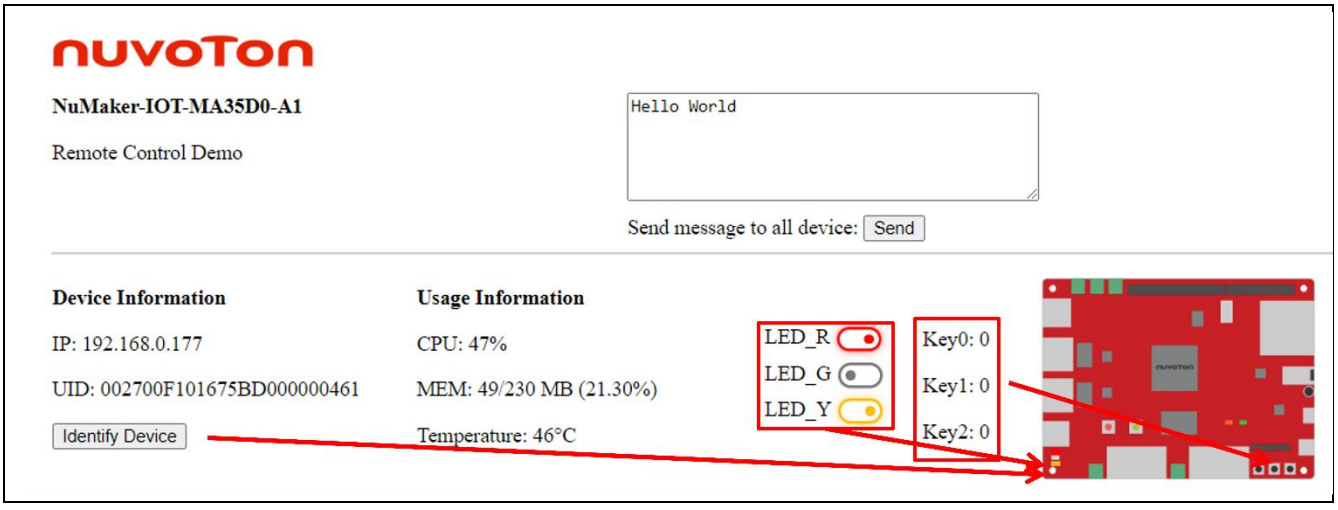

Figure 5-3 控制与按键状态

多个 MA35D0 EVB 共享信息:

开启另一个 EVB 并连接到控制台，在*/opt/init.sh* 修改初始化脚本以更改 MAC 地址。

init.sh 代码片段

```
#Set new MAC address
ifconfig eth0 down
ifconfig eth0 hw ether 22:33:44:55:66:77
```

重新启动 EVB 并检查新的 IP 地址。

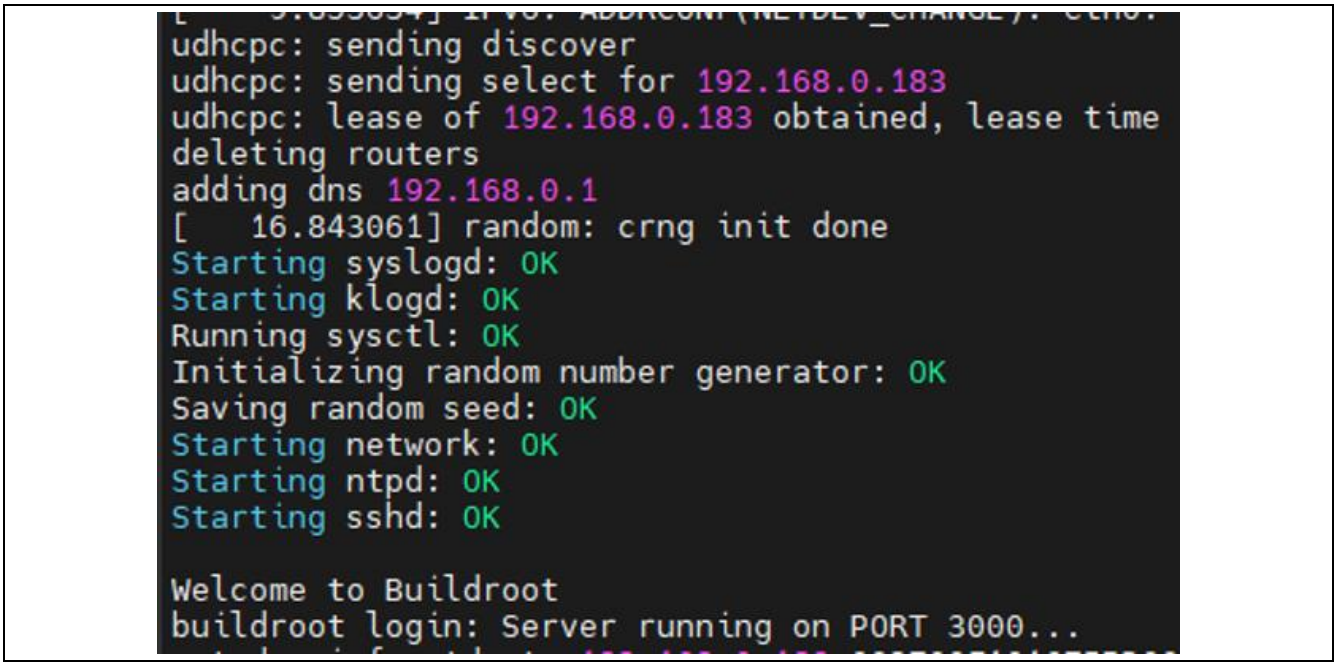

Figure 5-4 检查新 IP 地址

输入 ${MA35D0 IP}:3000 到浏览器，以进入每个 EVB 的网页服务器。

通过浏览器，可以在两个 EVB 的服务器上观察每个 MA35D0 的状态。

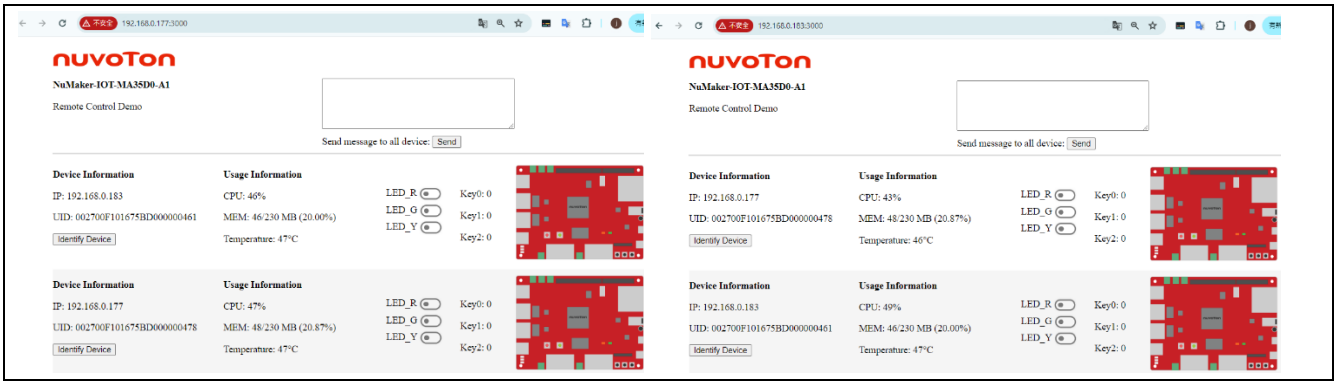

Figure 5-5 多个 EVB 状态信息

6. 修订纪录

Date	Revision	Description
2024.05.13	1.00	初始版本。

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.