TB3239

适用于 PIC®和 AVR®器件的 JSON 解码器

简介

作者：Alexandru Niculae，Microchip Technology Inc.

本文档介绍了 JSON 子集解码器的使用和实现，该解码器面向 PIC®和 AVR®单片机等嵌入式器件。

解码器将 JSON 对象的字符串格式转换为 C 数据结构表示形式，这样编程器便可以访问键值对。使用 JSON 对象可以轻松互连应用程序。

可在 GitHub 上获得 JSON 解码器代码。

目录

1. 概述

JSON 是 Web 和桌面应用程序上最常用的数据序列化格式。几乎所有现代编程语言中都已实现了 JSON，因此是在应用程序之间交换数据的理想工具。它具有易于人类读写以及易于机器解析和生成的文本格式。

JSON 对象是一组键值对。虽然键只能是字符串，但值可以是多种形式。此解码器实现支持字符串、数字和对象值。

注： 为了确保此库能用于存储器较小的器件，我们只选择并实现了一部分功能，因此不支持数组、布尔和空值。

以下代码块给出了此解码器的示例用法。2. API 一章详细介绍了 API 函数。

```
#define TEST_JSON "{\"main\":{\"key\" : 10,\"foo\":\"bar\"}, \"alt\":2}"

...

jsonNode_t *root, *objmain;
char str[64], foo[10];
int alt;

memcpy(str, TEST_JSON, sizeof(TEST_JSON));

JSON_DECODER_fromString(str)

JSON_DECODER_getRoot(&root);
JSON_DECODER_getObject(root, "main", &objmain);

JSON_DECODER_getNumber(root, "alt", &alt)
JSON_DECODER_getString(objmain, "foo", sizeof(foo), foo)
```

注： JSON 中最外层的对象称为根。

2. API

有关 JSON 解码器中公共函数的详细说明，请参见本章。

```
jsonDecoderStatus_t JSON_DECODER_fromString(char *str)
```

说明
将 JSON 字符串解析为 C 表示形式。它支持字符串、数字和对象值。

参数

名称	说明
str	指向 JSON 对象的现有字符串表示形式的指针。解析后，字符串将发生更改。

返回值

名称	说明
JSON_DECODER_OK	解码成功
JSON_DECODER_BAD_FORMAT	输入无效

```
JSON_DECODER_getRoot(jsonNode_t **pNode)
```

说明
找到 JSON 中最外层的对象（称为根）。不得在 JSON_DECODER_fromString 之前调用此函数。

参数

名称	说明
pNode	指向 jsonNode_t 的指针

返回值

名称	说明
JSON_DECODER_OK	pNode 现在指向根

```
jsonDecoderStatus_t JSON_DECODER_getObject(jsonNode_t *current, char *key, jsonNode_t **pNode)
```

说明
通过键在一个 JSON 对象中查找另一个 JSON 对象。不得在 JSON_DECODER_fromString 之前调用此函数。

参数

名称	说明
current	指向 jsonNode_t 的指针
key	待查找对象的键

.....（续）	
名称	**说明**
pNode	指向对象的指针（如果找到）

返回值

名称	说明
JSON_DECODER_OK	已找到对象，pNode 现在指向它
JSON_DECODER_KEY_NOT_FOUND	指定的键不存在

```
jsonDecoderStatus_t JSON_DECODER_getString(jsonNode_t *current, char *key, uint8_t size, char
*pVal)
```

说明
通过键在 JSON 对象中查找字符串。不得在 JSON_DECODER_fromString 之前调用此函数。

参数

名称	说明
current	要搜索的 JSON 对象
key	待查找字符串的键
size	字符串的最大大小。不得大于 pVal 缓冲区的长度
pVal	指向字符串的指针（如果找到）

返回值

名称	说明
JSON_DECODER_OK	已找到字符串，pVal 现在指向它
JSON_DECODER_KEY_NOT_FOUND	指定的键不存在

```
jsonDecoderStatus_t JSON_DECODER_getNumber(jsonNode_t *current, char *key, int *pVal)
```

说明
通过键在 JSON 对象中查找数字。不得在 JSON_DECODER_fromString 之前调用此函数。

参数

名称	说明
电流	要搜索的 JSON 对象
key	待查找字符串的键
pVal	指向数字的指针（如果找到）

返回值

名称	说明
JSON_DECODER_OK	已找到数字，pVal 现在指向它
JSON_DECODER_KEY_NOT_FOUND	指定的键不存在

3. 实现

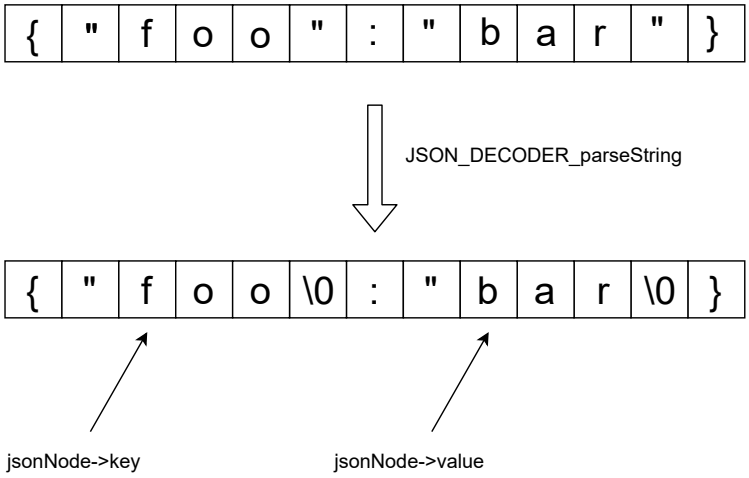
本章介绍了可帮助您深入了解代码的实现细节，还介绍了一些用于优化存储器使用和实现安全性的决策。例如，由于不建议使用递归，因此重构了算法的一部分。

3.1 输入缓冲区

输入缓冲区是通过 MQTT 库、UART 驱动程序或其他通信方法接收 JSON 字符串的缓冲区。此缓冲区将通过 `JSON_DECODER_parseString` 函数传递给 JSON 解码器。当此函数返回时，缓冲区的内容将发生修改。

为了尽可能少占用存储空间，内部 JSON 对象表示形式不包含任何字符串，而是包含指向字符串在输入缓冲区中的原始位置的指针。因此，应将空终止符插入此缓冲区中以隔离字符串。所有键和字符串值均在输入缓冲区内。

图 3-1. 字符串缓冲区



一个重要的影响是，在使用 JSON 数据之前，一定不要重写缓冲区。如果在 `JSON_DECODE_parseString` 和访问 JSON 结构之间重写了缓冲区，则将损坏数据。

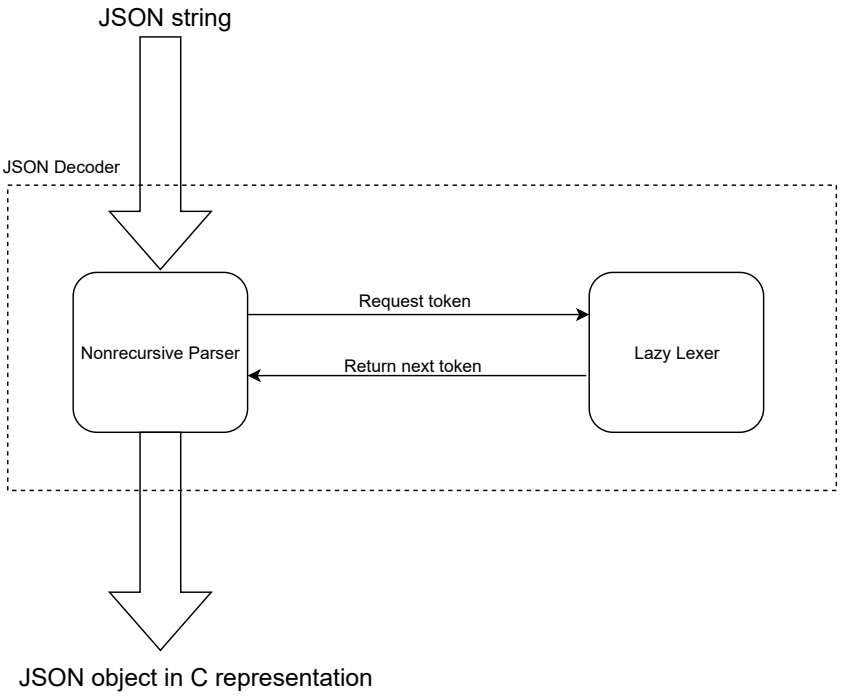
3.2 解码

解码意味着将 JSON 对象的字符串格式转换为 C 数据结构表示形式。这是一个语法分析问题，通常可以通过两个步骤解决：词法分析和语法分析。

词法分析将输入文本拆分为多个标记，并去除多余字符，例如空格。标记可以是单字符（大括号、逗号和冒号），也可以是多字符（字符串和数字）。此实现使用“惰性词法分析器”，这意味着将根据解析器的需要逐一检索标记，而不是创建所有标记的列表。

解析器验证标记序列在 JSON 语法的上下文中是否有意义。它将内部表示形式构造为 C 数据结构。解析器本质上是递归的，因为它处理嵌套的 JSON 对象。为了避免递归（和可能的栈溢出错误），将算法重构为使用堆栈和 while 循环，而不是使用实际的栈。

图 3-2. JSON 解码器 HLD

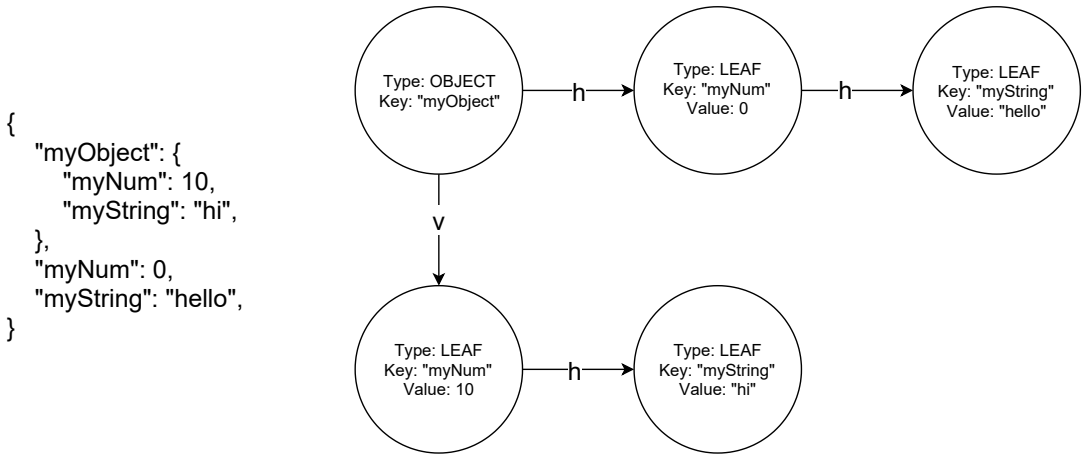


3.3 内部表示形式

解析字符串格式后，JSON 对象以树形结构存储在存储器中。使用 API 函数遍历树节点以通过键名检索值。每个节点都对应一个键，可以是叶节点（具有字符串或数字值）或对象节点。

图 3-3 给出了 JSON 对象及其 `jsonNode_t` 结构树的表示形式。节点结构具有类型字段、键字段、值字段和两个指向相邻节点的指针字段。如果当前节点是类型对象，则 `h` 指针指向同一 JSON 对象中的下一个键，而 `v` 指针指向第一个自己的键。

图 3-3. 内部表示形式



注： JSON 对象可以解码的最大键数受静态定义的 `jsonNode_t` 的数组长度限制。嵌入式系统中应避免动态存储器分配。

注： 嵌套对象的最大数量由解析器使用的堆栈的大小确定。

3.4 编码

尽管此实现不包括对应的编码，但只需使用 `printf`/`sprintf` 函数的形式参数即可轻松完成。以下代码行以字符串形式创建 JSON 对象。可通过 MQTT/UART 等协议发送此代码来报告测量数据。

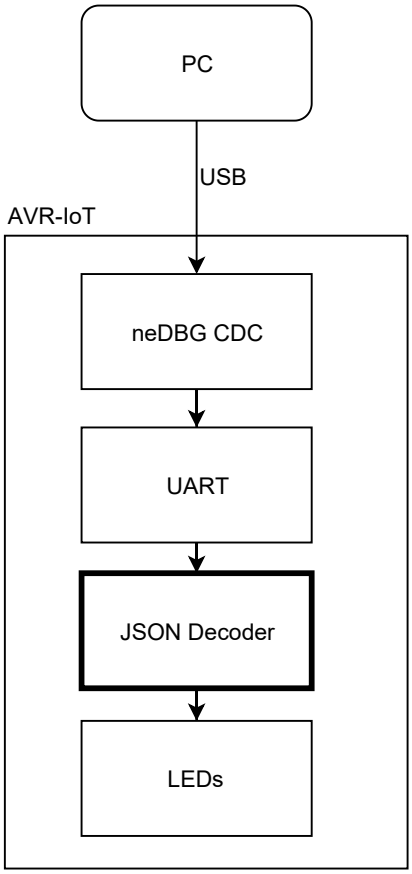
```
char json[30];
sprintf(json, "{\"Light\":%d,\"Temp\":%d}", 10, 25);
```

4. 演示：通过 PC 控制 AVR-IoT 上的四个 LED

本章演示如何将 JSON 解码器添加到一个简单的 MCC 项目中，以及如何通过从 Tera Term 终端仿真器发送 JSON 格式的命令来控制 AVR.IOT-WG 板上的四个 LED。

使用 JSON 对象执行此任务可提高灵活性，虽然是单个命令，但格式和处理逻辑清晰，可以一次控制一个、两个、三个或四个 LED。下图给出了本演示的架构总览图。

图 4-1. JSON 解码器简单用例



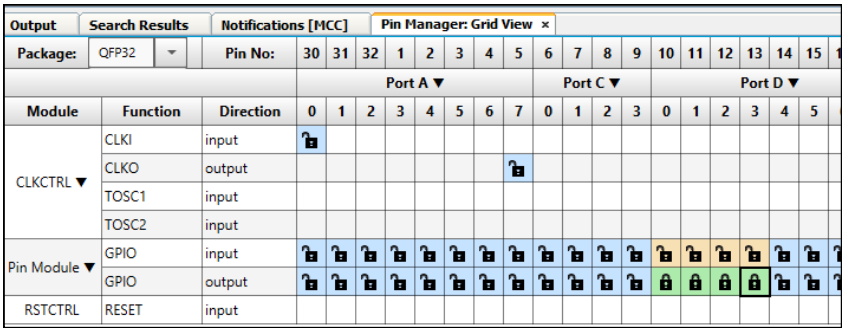
4.1 说明

要按照以下各节中的步骤操作，需要使用 AVR-IoT 板、MPLAB® X 5.25、带有 AVR MCU 外设库 2.0.2（旧版本可能也适用）的 MCC 3.85.1 和 Git。

4.1.1 使用 MCC 自举项目

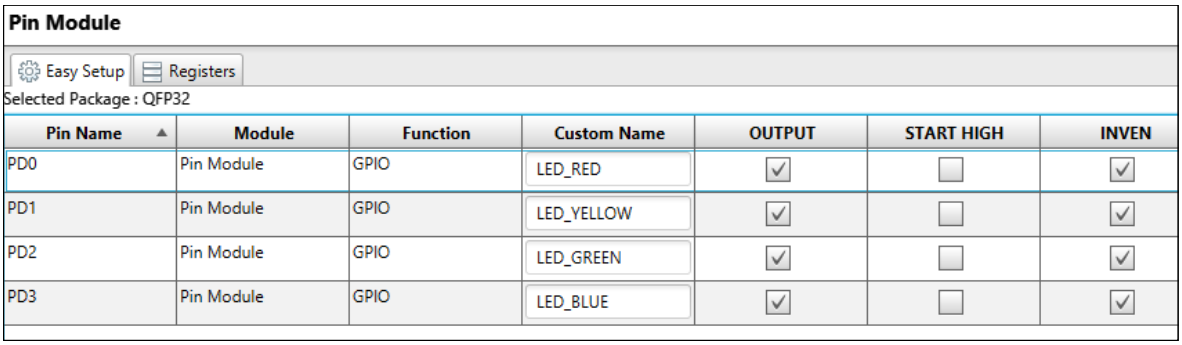
1. 为 ATmega4808 器件新建一个 MPLAB X 项目。
2. 单击 **MCC** 图标开始配置项目。
3. 在 Pin Manager: Grid View（引脚管理器：网格视图）窗口中，将引脚 PD0 至 PD3 标记为输出。

图 4-2. Pin Manager: Grid View



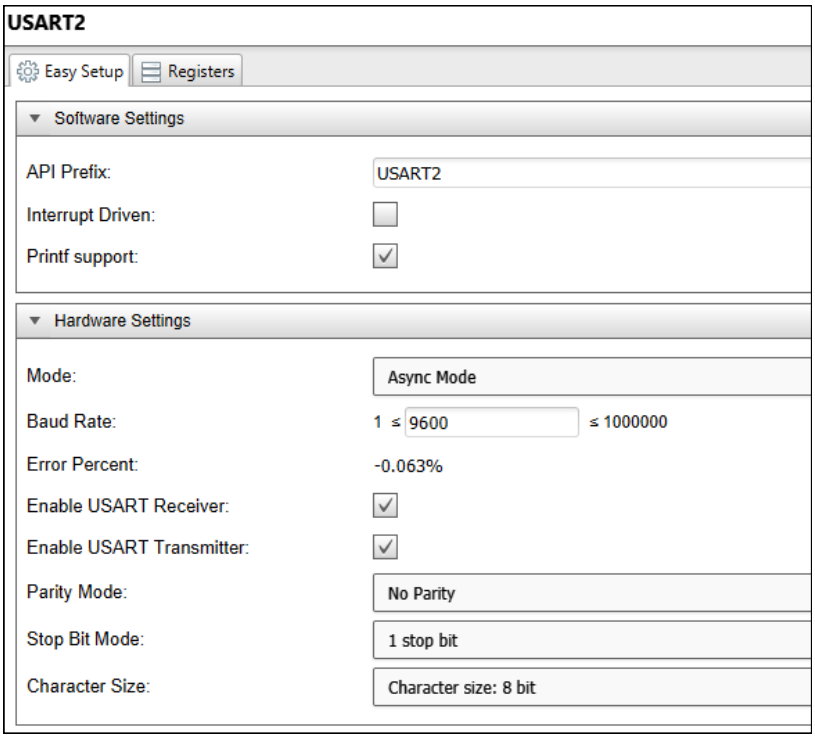
4. 在 Pin Module（引脚模块）窗口中，将引脚重命名为 LED_RED、LED_YELLOW、LED_GREEN 和 LED_BLUE，然后选中所有项的 INVEN。

图 4-3. Pin Module 窗口



5. 从 Device Resources（器件资源）中将 **USART2** 添加到项目中。
6. 在 Pin Manager: Grid View 窗口中，确保 USART2 使用引脚 PF0 和 PF1。
7. 在 USART2 设置窗口中，确保选中 **Printf support**（Printf 支持），并且所有其他设置均为默认设置。

图 4-4. USART2 设置窗口

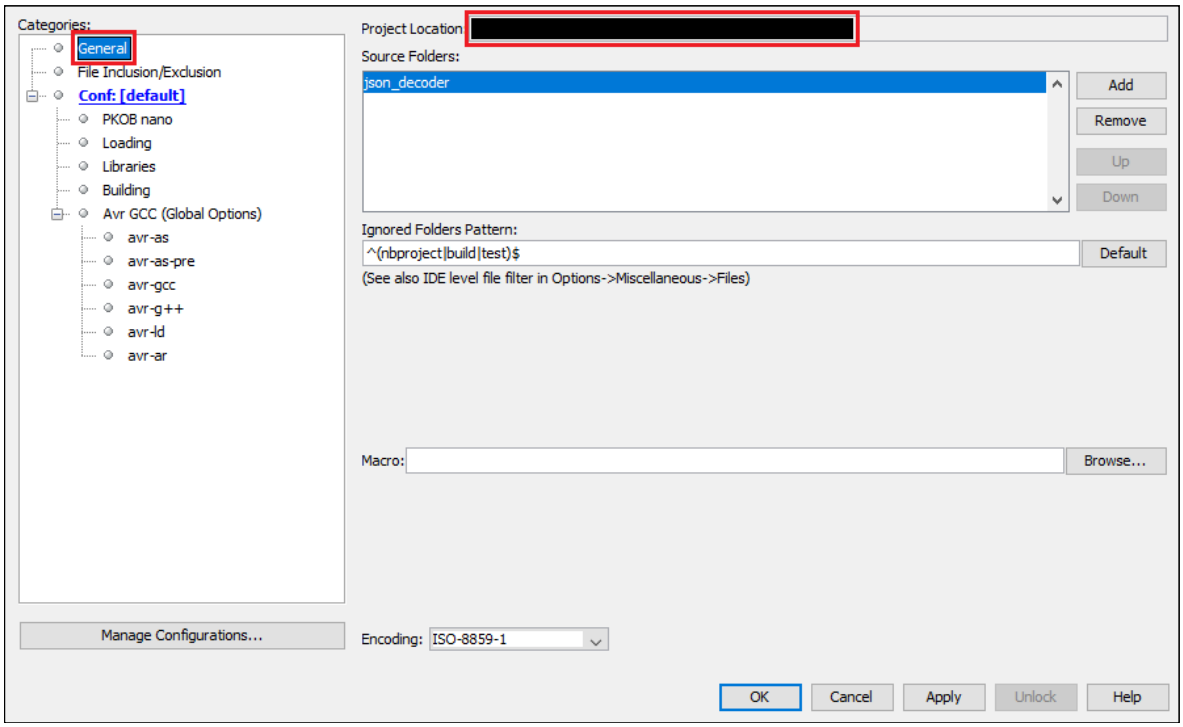


8. 单击 **generate**（生成）。

4.1.2 将 json_decoder 添加到项目

1. 在 MPLAB X 的 Projects（项目）窗口中右键单击项目路径，然后单击 **Properties**（属性）。**General**（常规）选项卡上的第一个元素是 **Project Location**（项目位置）。

图 4-5. MPLAB® X 项目窗口



2. 使用命令行（例如 cmd 或 Power Shell）导航到项目根目录。
3. 使用以下命令将 json_decoder 资源库克隆到项目中：

```
$ git clone https://github.com/MicrochipTech/json_decoder.git
```

4. 在 Projects 窗口中的当前项目下，右键单击 **Source Files**（源文件），然后选择 **New logical folder**（新建逻辑文件夹）。
5. 将创建的文件夹重命名为 **json_decoder**。
6. 右键单击 **json_decoder** 文件夹，然后选择 **Add Existing Item…**（添加现有项…）。从 json_decoder 克隆文件夹中添加所有 .c 源文件。
7. 在 **Header files**（头文件）下添加一个新的逻辑文件夹，将其重命名为 **json_decoder**，然后从 jsone_decoder 克隆文件夹中添加所有 .h 头文件。
8. 在 main.c 中包含 json_decoder/json_decore.h。

```
#include “json_decoder/json_decoder.h”
```

9. 单击 **Clean and Build Main Project**（清除并编译主项目）图标。项目将成功编译。

4.1.3 创建一个简单应用程序

1. 在 main.c 中定义 MAX_COMMAND_LEN。

```
#define MAX_COMMAND_LEN 64
```

2. 在主函数中的 while 循环之前定义以下变量。

```
char command[MAX_COMMAND_LEN];
uint8_t index = 0;
char c;
```

3. 将以下代码添加到 while 循环中。

```
c = USART2_Read();
if(c != '\n' && c != '\r')
{
    command[index++] = c;
    if(index > MAX_COMMAND_LEN)
    {
        index = 0;
    }
}

if(c == '\n')
{
    command[index] = '\0';
    index = 0;
    executeCommand(command);
}
```

注： 要了解有关 UART 的更多信息，请参见 *"TB3216 - Getting Started with USART (DS90003216B)"*。

4. 在主函数上方定义函数 void `executeCommand(char *command)`。
5. 在函数内部，将命令解码为 JSON 对象并获取根。

```
jsonNode_t *root = 0;
jsonDecoderStatus_t ret;
char state[5];

ret = JSON_DECODER_fromString(command);
if(JSON_DECODER_OK != ret)
{
    printf("Invalid JSON string.\r\n");
}

JSON_DECODER_getRoot(&root);
```

6. 检查解码命令中是否存在“red”键，并根据键值点亮或熄灭红色 LED。

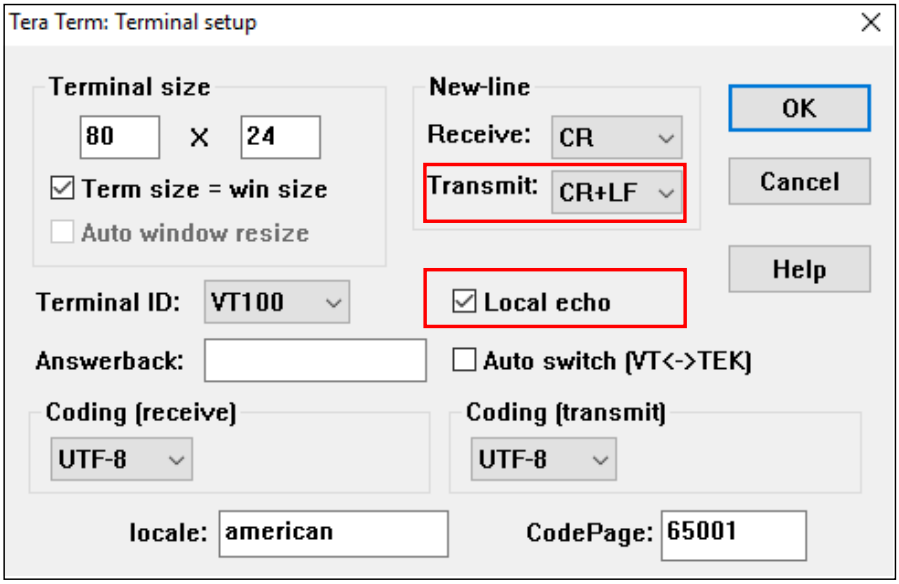
```
ret = JSON_DECODER_getString(root, "red", sizeof(state), state);
if(JSON_DECODER_OK == ret)
{
    if(strcmp(state, "on") == 0)
    {
        LED_RED_SetHigh();
    }
    else if(strcmp(state, "off") == 0)
    {
        LED_RED_SetLow();
    }
}
```

7. 对其他三个 LED 执行类似操作。
   注： 有关 `executeCommand` 函数的完整代码，请参见 6. 附录。

4.1.4 测试应用程序

1. 打开一个终端仿真器，例如 Tera Term。
2. 选择插入计算机的 AVR.IoT-WG 的 COM 端口。
3. 在 Tera Term 菜单中，转到 *Setup → Terminal*（设置→终端），并将其设置为发送 CR + LF 格式的换行符，然后启用 **Local echo**（本地回显）。

图 4-6. Tera Term 终端设置



4. 发送 JSON 字符串命令，例如：

```
{" red" :" on" }
```

- 红色 LED 将点亮。

```
{" blue" :" on" }
```

- 蓝色 LED 将点亮。

```
{" yellow" :" on" ," blue" :" off" }
```

- 黄色 LED 将点亮。
- 蓝色 LED 将熄灭。

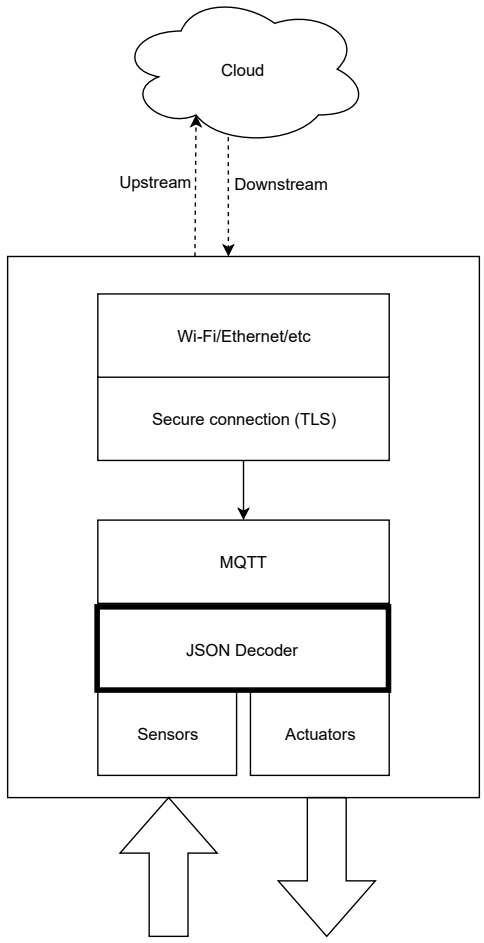
```
{" yellow" :" off" ," red" :" off" ," blue" :" on" ," green" :" on" }
```

- 黄色 LED 和红色 LED 将熄灭。
- 蓝色 LED 和绿色 LED 将点亮。

5. 演示：与 IoT 节点的通用云通信

与 IoT 节点之间的一种常见云通信方式是使用安全的 TLS 协议。应用协议有很多种，但本场景下最常用的是 MQTT。实际的数据通常是 JSON 对象。因此，JSON 解码器模块是 IoT 节点中不可或缺的部分。下图演示了此类应用的通用架构总览图。

图 5-1. JSON 解码器高级示例



6. 附录

软件许可协议

Microchip Technology Incorporated（以下简称“本公司”）在此提供的软件只面向本公司客户，并只用于本公司生产的产品。

本软件为本公司和/或其供应商所有，并受到适用的版权法保护。保留所有权利。用户在使用时如果违反上述限制，可能会依法受到刑事制裁，并可能由于违背本许可证的条款和条件而承担民事责任。

本软件按“原样”提供。本公司不作任何明示、暗示或法定形式的担保，包括但不限于对软件适销性和特定用途的适用性的暗示担保。对于在任何情况下，因任何原因造成的特殊的、偶然的或间接的损失，本公司概不负责。

```
void executeCommand(char *command)
{
    jsonNode_t *root = 0;
    jsonDecoderStatus_t ret;
    char state[5];

    ret = JSON_DECODER_fromString(command);
    if(JSON_DECODER_OK != ret)
    {
        printf("Invalid JSON string.\r\n");
    }

    JSON_DECODER_getRoot(&root);

    ret = JSON_DECODER_getString(root, "red", sizeof(state), state);
    if(JSON_DECODER_OK == ret)
    {
        if(strcmp(state, "on") == 0)
        {
            LED_RED_SetHigh();
        }
        else if(strcmp(state, "off") == 0)
        {
            LED_RED_SetLow();
        }
    }

    ret = JSON_DECODER_getString(root, "yellow", sizeof(state), state);
    if(JSON_DECODER_OK == ret)
    {
        if(strcmp(state, "on") == 0)
        {
            LED_YELLOW_SetHigh();
        }
        else if(strcmp(state, "off") == 0)
        {
            LED_YELLOW_SetLow();
        }
    }

    ret = JSON_DECODER_getString(root, "green", sizeof(state), state);
    if(JSON_DECODER_OK == ret)
    {
        if(strcmp(state, "on") == 0)
        {
            LED_GREEN_SetHigh();
        }
        else if(strcmp(state, "off") == 0)
        {
            LED_GREEN_SetLow();
        }
    }

    ret = JSON_DECODER_getString(root, "blue", sizeof(state), state);
    if(JSON_DECODER_OK == ret)
    {
        if(strcmp(state, "on") == 0)
```

```
        {
            LED_BLUE_SetHigh();
        }
        else if(strcmp(state, "off") == 0)
        {
            LED_BLUE_SetLow();
        }
    }
}
```

Microchip 网站

Microchip 网站（www.microchip.com/）为客户提供在线支持。客户可通过该网站方便地获取文件和信息。我们的网站提供以下内容：

- **产品支持**——数据手册和勘误表、应用笔记和示例程序、设计资源、用户指南以及硬件支持文档、最新的软件版本以及归档软件
- **一般技术支持**——常见问题解答（FAQ）、技术支持请求、在线讨论组以及 Microchip 设计伙伴计划成员名单
- **Microchip 业务**——产品选型和订购指南、最新 Microchip 新闻稿、研讨会和活动安排表、Microchip 销售办事处、代理商以及工厂代表列表

产品变更通知服务

Microchip 的产品变更通知服务有助于客户了解 Microchip 产品的最新信息。注册客户可在他们感兴趣的某个产品系列或开发工具发生变更、更新、发布新版本或勘误表时，收到电子邮件通知。

欲注册，请访问 www.microchip.com/pcn，然后按照注册说明进行操作。

客户支持

Microchip 产品的用户可通过以下渠道获得帮助：

- 代理商或代表
- 当地销售办事处
- 应用工程师（ESE）
- 技术支持

客户应联系其代理商、代表或 ESE 寻求支持。当地销售办事处也可为客户提供帮助。本文档后附有销售办事处的联系方式。

也可通过 www.microchip.com/support 获得网上技术支持。

Microchip 器件代码保护功能

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术规范。
- Microchip 确信：在正常使用的情况下，Microchip 系列产品非常安全。
- 目前，仍存在着用恶意、甚至是非法的方法来试图破坏代码保护功能的行为。我们确信，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这种试图破坏代码保护功能的行为极可能侵犯 Microchip 的知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是“牢不可破”的。代码保护功能处于持续发展中。Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案（Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

法律声明

提供本文档的中文版本仅为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc.及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc.的英文原版文档。

商标

ISBN： 978-1-5224-7353-4

质量管理体系

有关 Microchip 的质量管理体系的信息，请访问 www.microchip.com/quality。

全球销售及服务网点

美洲	亚太地区	亚太地区	欧洲
公司总部 2355 West Chandler Blvd. Chandler, AZ 85224-6199 电话：480-792-7200 传真：480-792-7277 技术支持： www.microchip.com/support 网址： www.microchip.com **亚特兰大** 德卢斯，佐治亚州 电话：678-957-9614 传真：678-957-1455 **奥斯汀，德克萨斯州** 电话：512-257-3370 **波士顿** 韦斯特伯鲁，马萨诸塞州 电话：774-760-0087 传真：774-760-0088 **芝加哥** 艾塔斯卡，伊利诺伊州 电话：630-285-0071 传真：630-285-0075 **达拉斯** 阿迪森，德克萨斯州 电话：972-818-7423 传真：972-818-2924 **底特律** 诺维，密歇根州 电话：248-848-4000 **休斯顿，德克萨斯州** 电话：281-894-5983 **印第安纳波利斯** 诺布尔斯维尔，印第安纳州 电话：317-773-8323 传真：317-773-5453 电话：317-536-2380 **洛杉矶** 米慎维荷，加利福尼亚州 电话：949-462-9523 传真：949-462-9608 电话：951-273-7800 **罗利，北卡罗来纳州** 电话：919-844-7510 **纽约，纽约州** 电话：631-435-6000 **圣何塞，加利福尼亚州** 电话：408-735-9110 电话：408-436-4270 **加拿大 - 多伦多** 电话：905-695-1980 传真：905-695-2078	**澳大利亚 - 悉尼** 电话：61-2-9868-6733 **中国 - 北京** 电话：86-10-8569-7000 **中国 - 成都** 电话：86-28-8665-5511 **中国 - 重庆** 电话：86-23-8980-9588 **中国 - 东莞** 电话：86-769-8702-9880 **中国 - 广州** 电话：86-20-8755-8029 **中国 - 杭州** 电话：86-571-8792-8115 **中国 - 香港特别行政区** 电话：852-2943-5100 **中国 - 南京** 电话：86-25-8473-2460 **中国 - 青岛** 电话：86-532-8502-7355 **中国 - 上海** 电话：86-21-3326-8000 **中国 - 沈阳** 电话：86-24-2334-2829 **中国 - 深圳** 电话：86-755-8864-2200 **中国 - 苏州** 电话：86-186-6233-1526 **中国 - 武汉** 电话：86-27-5980-5300 **中国 - 西安** 电话：86-29-8833-7252 **中国 - 厦门** 电话：86-592-2388138 **中国 - 珠海** 电话：86-756-3210040	**印度 - 班加罗尔** 电话：91-80-3090-4444 **印度 - 新德里** 电话：91-11-4160-8631 **印度 - 浦那** 电话：91-20-4121-0141 **日本 - 大阪** 电话：81-6-6152-7160 **日本 - 东京** 电话：81-3-6880-3770 **韩国 - 大邱** 电话：82-53-744-4301 **韩国 - 首尔** 电话：82-2-554-7200 **马来西亚 - 吉隆坡** 电话：60-3-7651-7906 **马来西亚 - 槟榔屿** 电话：60-4-227-8870 **菲律宾 - 马尼拉** 电话：63-2-634-9065 **新加坡** 电话：65-6334-8870 **台湾地区 - 新竹** 电话：886-3-577-8366 **台湾地区 - 高雄** 电话：886-7-213-7830 **台湾地区 - 台北** 电话：886-2-2508-8600 **泰国 - 曼谷** 电话：66-2-694-1351 **越南 - 胡志明市** 电话：84-28-5448-2100	**奥地利 - 韦尔斯** 电话：43-7242-2244-39 传真：43-7242-2244-393 **丹麦 - 哥本哈根** 电话：45-4485-5910 传真：45-4485-2829 **芬兰 - 埃斯波** 电话：358-9-4520-820 **法国 - 巴黎** 电话：33-1-69-53-63-20 传真：33-1-69-30-90-79 **德国 - 加兴** 电话：49-8931-9700 **德国 - 哈恩** 电话：49-2129-3766400 **德国 - 海尔布隆** 电话：49-7131-72400 **德国 - 卡尔斯鲁厄** 电话：49-721-625370 **德国 - 慕尼黑** 电话：49-89-627-144-0 传真：49-89-627-144-44 **德国 - 罗森海姆** 电话：49-8031-354-560 **以色列 - 若那那市** 电话：972-9-744-7705 **意大利 - 米兰** 电话：39-0331-742611 传真：39-0331-466781 **意大利 - 帕多瓦** 电话：39-049-7625286 **荷兰 - 德卢内市** 电话：31-416-690399 传真：31-416-690340 **挪威 - 特隆赫姆** 电话：47-72884388 **波兰 - 华沙** 电话：48-22-3325737 **罗马尼亚 - 布加勒斯特** 电话：40-21-407-87-50 **西班牙 - 马德里** 电话：34-91-708-08-90 传真：34-91-708-08-91 **瑞典 - 哥德堡** 电话：46-31-704-60-40 **瑞典 - 斯德哥尔摩** 电话：46-8-5090-4654 **英国 - 沃金厄姆** 电话：44-118-921-5800 传真：44-118-921-5820