

ESP8266 + JavaScript 操作系统

Mongoose OS 制作物联网项目

Phodal Huang

October 24, 2017

目录

步骤 0: Mongoose OS	3
步骤 1: 开始	4
克隆 LOSANT APP	4
步骤 2: 更新配置	4
步骤 3: 代码	5
步骤 3: 构建固件	6
步骤 4: 烧录固件	7
步骤 5: STREAM LOGS	8
步骤 6: 打开 WEB UI	8
步骤 7: 测试	8
下一步?	8

玩点什么: <https://www.wandianshenme.com>

原文链接: <https://www.wandianshenme.com/play/javascript-os-mongoose-os-esp8266-build-iot-pr>

Mongoose OS, 这是一款支持 **JavaScript** 的硬件开源操作系统。它能提供一个 **Web UI** 界面 (包含 **IDE** 和设备管理功能)、可以直接运行在 **ESP8266** 上。文章介绍了如何编译、烧录 **Mongoose OS** 与 **Mongoose OS Losant** 应用, 并上传数据到 **Losant** 后台。

由于 **JavaScript** 充满活力的社区, 它已经通过诸如 **Johnny-Five**, **Espruino** 和 **JerryScript** 这样的项目进入了硬件领域。在今年早些时候, **Cesanta** 宣布推出 **Mongoose OS**, 这是一款支持 **JavaScript** 的硬件开源操作系统。

Mongoose OS 在构建可靠设备固件方面, 是一个很棒的工具链。当它与 **Losant** 一起使用时, 您可以通过强大的数据处理、可视化、仪表板和云集成等, 来扩展设备的覆盖面。

在本文中, 我们将介绍 **Mongoose OS** 的一些功能, 以及如何将其与 **Losant** 集成。

步骤 0: **Mongoose OS**

Mongoose OS 中最棒的功能之一是 **Web UI** 工具。该 **Web** 界面是一个完整的 **IDE** 和设备配置管理器, 可让您构建和烧录固件。

Mongoose 操作系统还支持 **JavaScript**。但是, **Mongoose OS JavaScript**, 即 **mJS**, 它仅实现了语言的一部分。这样做的目的是, 保证其程序占用的内存足够低。由于内存占用空间小, 在诸如 **ESP8266** 等低内存设备上的 **TLS**, 变得更容易实现。

下面是一个使用 **JavaScript** 编写的、简单的 **Mongoose OS** 操作系统应用程序示例, 用于开关 **LED**:

```
1 load('api_gpio.js'); // load is a special function in Mongoose OS
2 load('api_timer.js');
3
4 let led = 4;
5
6 // Blink built-in LED every second
7 GPIO.set_mode(led, GPIO.MODE_OUTPUT);
8 Timer.set(1000 /* 1 sec */, true /* repeat */, function() {
9   print("Toggling LED");
10  let value = GPIO.toggle(led);
11  }, null);
```

如果您熟悉 JavaScript，这可能看起来有点奇怪。Mongoose OS 暴露了一个加载函数用于包含（引入，include）文件，这与 Node.js 中的 require 类似。除此，Mongoose OS 还暴露了其他有用的对象和功能，如 GPIO、打印（print）和定时器（Timer）。

最后，Mongoose OS 附带了一个名为 mos 的完整 CLI 工具。Web UI 是围绕 CLI 工具构建的。mos CLI 工具为您提供了与硬件交互的终端界面。

步骤 1: 开始

既然，你已经对 Mongoose 操作系统有了基本的了解。那么，让我们配合 Losant 一起来使用它。

开始之前：

- 您将需要一个基于 ESP8266 的设备。
- 您必须安装 mos 工具。有关更多信息，请参阅 [mos 安装说明](#)。

克隆 LOSANT APP

为了简单起见，我们为 Losant 创建了一个 [Mongoose 应用程序](#)。这个 Mongoose OS 应用程，可以序作为您的项目的样板模板。该示例应用程序可在 [GitHub](#) 下载：

```
1 $ git clone https://github.com/Losant/losant-mqtt-mongoose-os.git
```

步骤 2: 更新配置

Mongoose OS 支持运行时多层（multi-layer）配置。这意味着，您可以提供多个编号的配置文件，并且操作系统将根据优先级合并，并正确地对它们进行排序。在这个程序中，我们提供了 conf1.json。如下所示：

```
1 {
2   "wifi": {
3     "sta": {
4       "enable": true,
5       "ssid": "WIFI_SSID",
6       "pass": "WIFI_PASSWORD"
7     }
8   },
9   "device": {
```

```
10     "id": "LOSANT_DEVICE_ID"
11 }, "debug": {
12     "stdout_topic": "",
13     "stderr_topic": ""
14 },
15 "mqtt": {
16     "enable": true,
17     "client_id": "LOSANT_DEVICE_ID",
18     "user": "LOSANT_ACCESS_KEY",
19     "pass": "LOSANT_ACCESS_SECRET",
20     "ssl_ca_cert": "ca.pem"
21 }
22 }
```

接着, 替换下面的值:

- LOSANT_DEVICE_ID
- LOSANT_ACCESS_KEY
- LOSANT_ACCESS_SECRET
- WIFI_SSID
- WIFI_PASSWORD

你需要从 [Losant](#) 获取 LOSANT_DEVICE_ID、LOSANT_ACCESS_KEY 和 LOSANT_ACCESS_SECRET。如果您还没有, 请在 Losant 中[创建一个帐户](#)和一个应用程序。接下来, 添加一个设备到您项目的 Losant 应用程序中。然后, 您将可以获得访问密钥和密码。

步骤 3: 代码

我们来看看 Losant Mongoose 模板应用程序中包含的主要代码:

```
1 load('api_config.js');
2 load('api_gpio.js');
3 load('api_mqtt.js');
4 load('api_sys.js');
5 load('api_timer.js');
6
```

```
7 // Helper C function get_led_gpio_pin() in src/main.c returns built-in LED
  GPIO
8 let led = ffi('int get_led_gpio_pin()')();
9 let getInfo = function() {
10   return JSON.stringify({data:{ total_ram: Sys.total_ram(), free_ram:
      Sys.free_ram() }});
11 };
12
13 // Blink built-in LED every second
14 GPIO.set_mode(led, GPIO.MODE_OUTPUT);
15 Timer.set(1000 /* 1 sec */, true /* repeat */, function() {
16   let value = GPIO.toggle(led);
17   print(value ? 'Tick' : 'Tock', 'uptime:', Sys.uptime(), getInfo());
18 }, null);
19
20 // Publish to MQTT topic on a button press. Button is wired to GPIO pin 0
21 GPIO.set_button_handler(0, GPIO.PULL_UP, GPIO.INT_EDGE_NEG, 200, function()
  {
22   let topic = '/losant/' + Cfg.get('device.id') + '/state';
23   let message = getInfo();
24   let ok = MQTT.pub(topic, message, 1);
25   print('Published:', ok ? 'yes' : 'no', 'topic:', topic, 'message:',
      message);
26 }, null);
```

首先，上述的代码每秒闪烁一次 **LED**，类似于我们在上面的示例代码中看到的。接下来，每次按下按钮，它会向 **Losant** 发送消息。此消息中包含的是 **ESP8266** 的 `total_ram` 和 `free_ram`。

步骤 3: 构建固件

我们下载了应用程充，更新了配置，并对代码进行了审核。

现在，我们现在可以构建固件。执行：

```
1 $ mos build --arch esp8266
```

```
fox@Foxbook: ~/mos/apps/losant-mqtt-mongoose-os
▲ losant-mqtt-mongoose-os at master ✓ mos build --arch esp8266
Handling lib "rpc-loopback"...
The --lib flag was not given for it, checking repository
Clean, skipping (will be handled remotely)
Handling lib "rpc-mqtt"...
The --lib flag was not given for it, checking repository
Clean, skipping (will be handled remotely)
Handling lib "rpc-service-config"...
The --lib flag was not given for it, checking repository
Clean, skipping (will be handled remotely)
Handling lib "rpc-service-fs"...
The --lib flag was not given for it, checking repository
Clean, skipping (will be handled remotely)
Handling lib "rpc-service-gpio"...
The --lib flag was not given for it, checking repository
Clean, skipping (will be handled remotely)
Handling lib "rpc-service-i2c"...
The --lib flag was not given for it, checking repository
Clean, skipping (will be handled remotely)
Handling lib "rpc-service-ota"...
The --lib flag was not given for it, checking repository
Clean, skipping (will be handled remotely)
Handling lib "rpc-uart"...
The --lib flag was not given for it, checking repository
Clean, skipping (will be handled remotely)
Handling lib "spi"...
The --lib flag was not given for it, checking repository
Clean, skipping (will be handled remotely)
Handling lib "vfs-dev-spi-flash"...
The --lib flag was not given for it, checking repository
Clean, skipping (will be handled remotely)
Handling lib "mjs"...
The --lib flag was not given for it, checking repository
Clean, skipping (will be handled remotely)
Handling lib "atca"...
The --lib flag was not given for it, checking repository
Clean, skipping (will be handled remotely)
Handling lib "dns-sd"...
The --lib flag was not given for it, checking repository
Clean, skipping (will be handled remotely)
Handling lib "rpc-service-atca"...
The --lib flag was not given for it, checking repository
Clean, skipping (will be handled remotely)
Connecting to http://mongoose.cloud, user test
Uploading sources (3409 bytes)
Success, built losant-mqtt-mongoose-os/esp8266 version 1.0 (20170628-155204/???)
Firmware saved to build/fw.zip
▲ losant-mqtt-mongoose-os at master ✓
```

步骤 4: 烧录固件

一旦固件建成, 我们能烧录固件到设备上。

```
1 $ mos flash esp8266
```

```
fox@Foxbook: ~/mos/apps/losant-mqtt-mongoose-os
▲ losant-mqtt-mongoose-os at master ✓ mos flash
Loaded losant-mqtt-mongoose-os/esp8266 version 1.0 (20170628-155204/???)
Using port /dev/cu.SLAB_USBtoUART
Opening /dev/cu.SLAB_USBtoUART...
Connecting to ESP8266 ROM, attempt 1 of 10..
Connected
Running Flasher @ 460800..
Flasher is running
Flash size: 4194304, params: 0x0240 (dio,32m,40m)
Deduping...
 2544 @ 0x0 -> 0
 128 @ 0x3fc000 -> 0
Writing...
 4096 @ 0x1000
 671744 @ 0x8000
 131072 @ 0xdb000
 4096 @ 0xfb000
Wrote 811008 bytes in 17.81 seconds (355.73 KBit/sec)
Verifying...
 2544 @ 0x0
 4096 @ 0x1000
 669968 @ 0x8000
 131072 @ 0xdb000
 4096 @ 0xfb000
 128 @ 0x3fc000
Booting firmware...
All done!
▲ losant-mqtt-mongoose-os at master ✓
```

步骤 5: **STREAM LOGS**

一旦烧录完成，固件将立即运行。**Mongoose OS** 操作系统带有很多有用的日志记录。您可以通过在终端中运行下面的命令查看：

```
1 $ mos console
```

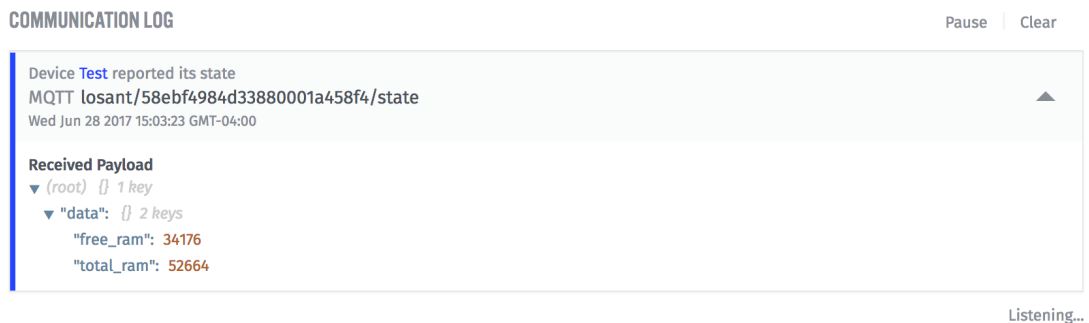
步骤 6: 打开 **WEB UI**

正如我之前提到的，**Mongoose** 的 **Web UI** 界面是非常棒。您可以使用终端中运行命令，就可以使用 **Web UI** 界面。要启动它，运行：

```
1 $ mos
```

步骤 7: 测试

要测试，可以转到 **Losant** 的主应用页面。你应该看到你的通讯日志。如果按下 **ESP8266** 上的按钮，您将看到一条消息。如下：



The screenshot shows a 'COMMUNICATION LOG' interface with 'Pause' and 'Clear' buttons. The log entry states: 'Device Test reported its state', 'MQTT losant/58ebf4984d33880001a458f4/state', and 'Wed Jun 28 2017 15:03:23 GMT-04:00'. Under 'Received Payload', it shows a tree structure: '(root) {} 1 key' expanded to '"data": {} 2 keys', which contains '"free_ram": 34176' and '"total_ram": 52664'. The status 'Listening..' is visible at the bottom right.

您刚刚成功地烧录您的设备，并将其连接到云端。你太棒了。

下一步？

您应该在 **Losant** 文档中查看许多的项目、教程。将设备连接到 **Losant** 后，您可以构建工作流和仪表盘，并将您的设备集成到几乎所有的云服务中，其可能性是无止境的。

相关链接：

- [Losant Workflows](#)
- [Losant Dashboards](#)
- [Mongoose OS Documentation](#)

原文: <https://www.losant.com/blog/getting-started-with-mongoose-os-esp8266-and-losant>

原文链接: <https://www.wandianshenme.com/play/javascript-os-mongoose-os-esp8266-build-iot-pr>

玩点什么: <https://www.wandianshenme.com>