# 使用 **IoT.js** 和 **Raspberry Pi** 开发物联网应用

Phodal Huang

September 8, 2017

# 目录

原文链接：https://www.wandianshenme.com/play/jerryscript-iotjs-raspberry-pi-build-iot-applicatio

IoT.js 是一个使用 JavaScript 语言编写的物联网应用平台；JerryScript 是一个适用于嵌入式设备的小型 JavaScript 引擎。本文将介绍如何在 Raspberry Pi 上，编译 IoT.js，并使用 JerryScript 来编写物联网应用。

> IoT.js 平台使用 JerryScript 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。IoT.js 目前运行在 Linux 和 NuttX（一个实时操作系统），目标设备为树莓派 2（Raspberry Pi 2）和意法半导体开发板（ST board），后续计划支持其他微控制器（MCU）和物联网设备。

IoT.js 当前支持两种构建类型：

- 在桌面操作系统上编译，支持 Linux（Ubuntu）和 macOS 上的交叉编译
- 在 Raspberry Pi 2 上构建

## 步骤 **1**：设置 **Raspberry Pi**

IoT.js 官方支持 Raspbian。而为了编译、使用 IoT.js 我们需要先进行一些设置。

### 允许 **I2C** 接口

为了使用 I2C 模块，我们需要先允许 I2C 接口。从命令行运行：

```
1 sudo raspi-config
```

这将运行起 `raspi-config` 工具，然后：

- 选择 “9 Advanced Options”
- 选择 “A6 I2C”

屏幕上将问你是否允许 I2C 接口，分别选择『YES』、『OK』、『Finish』来返回到命令行。

然后重启 Raspberry Pi。

### 允许 **PWM** 接口

Raspberry Pi 2 在以下引脚上有两个 PWM 输出：

| PWM 号 | GPIO 引脚 (功能) |
|---|---|
| PWM0 | GPIO12(4), GPIO18(2) |
| PWM1 | GPIO13(4), GPIO19(2) |

要使用 PWM 模块，必须在 `/boot/config.txt` 文件中添加 PWM 叠加。

例如，要在 GPIO 18 上获得单个 PWM，请添加如下所示的叠加（overlays）：

```
1 dtoverlay=pwm,pin=18,func=2
```

例如，要在 GPIO 18 和 GPIO 19 上获取多个 PWM，请添加如下所示的叠加（overlays）：

```
1 dtoverlay=pwm-2chan,pin=18,func=2,pin2=19,func2=2
```

有关叠加（overlays）的详细信息，请参考官方的 README。

注意：为了运行 PWM 模块，有必要拥有 root 权限。

### 允许 **UART** 接口

要使用 UART 模块，必须要使用 UART 接口。

在 `/boot/config.txt` 文件中，将 `enable_uart` 的值从 0 更改为 1 。

```
1 enable_uart=1
```

要禁用串行控制台，请编辑文件 `/boot/cmdline.txt`。删除 `console = serial0,115200` 或 `console = ttyAMA0,115200`。

要启用串行控制台，请编辑文件 `/boot/cmdline.txt`。添加 `console = serial0,115200` 或 `console = ttyAMA0,115200`。

再重启你的 Raspberry Pi。

注意：在 Raspberry Pi 3 上，你应该使用 `/dev/ttyS0` 替换 `/dev/ttyAMA0`。

## 步骤 **2**：在你的电脑上构建 **IoT.js**

首先，我们要行 clone 代码：

```
1 git clone https://github.com/Samsung/iotjs
```

**Linux** 条件准备

安装 arm linux 交叉编译器

```
sudo apt-get install gcc-arm-linux-gnueabihf
```

**macOS** 条件准备

通过以下博客了解如何安装 arm linux 交叉编译器：ARM Cross Compiling with Mac OS X

arm linux 编译器工具链的默认位置是 `/usr/local/linaro/arm-linux-gnueabihf-raspbian`。

然后，你需要锁定 C 编译（c_compiler）器。在文件 `./cmake/config/arm-linux.cmake` 中：

```
SET(EXTERNAL_CMAKE_C_COMPILER
    /usr/local/linaro/arm-linux-gnueabihf-raspbian/bin/arm-linux-gnueabihf-gcc)
```

文件 `/deps/libtuv/cmake/config/config_arm-linux.cmake`：

```
SET(CMAKE_C_COMPILER
    /usr/local/linaro/arm-linux-gnueabihf-raspbian/bin/arm-linux-gnueabihf-gcc)
```

构建 **IoT.js**（交叉编译）

在运行 `build.py` 文件时，设置 `target-arch`、`target-os` 和 `target-board` 选项，然后你就可以休息片刻了：

```
./tools/build.py --buildtype=[release|debug] --target-arch=arm \
 --target-os=linux --target-board=rpi2
```

在 **Raspberry Pi 2** 上运行

上述的脚本将会编译产生 `build/arm-linux/release/bin/iotjs` 也有可能是 `build/arm-linux/debug/bin/iotjs`。使用 scp 命令或者你喜欢的工具，复制二进制文件到你的 Raspberry Pi 2 上：

```
scp build/arm-linux/release/bin/iotjs pi@(your RPi2 IP):/home/pi/.
```

再 SSH 到 RPi 上：

```
ssh pi@(your RPi2 IP)
```

然后，让我们编写一个 hello,world：

```
console.log('Hello, world!');
```

最后，运行我们的测试程序。

```
./iotjs hello.js
```

## 步骤 2：在 **Raspberry Pi** 上构建

首先，安装编译工具：

```
sudo apt-get update
sudo apt-get install cmake
```

然后，执行构建命令：

```
./tools/build.py --target-board=rpi2
```

原文链接：https://www.wandianshenme.com/play/jerryscript-iotjs-raspberry-pi-build-iot-applicatio