在 **ESP8266** 上使用轻量级 **JavaScript** 引擎 **JerryScript** 开发应用

Phodal Huang

October 24, 2017

目录

原文链接：https://www.wandianshenme.com/play/javascript-engine-jerryscript-esp8266-build-iot-a

JerryScript 是用于资源受限的设备（如微控制器）的轻量级 JavaScript 引擎。它可以在具有小于 64 KB RAM 和小于 200KB flash 的设备上运行。

JerryScript 的主要特性有：

- 符合 ECMAScript 5.1 标准
- 编译为 ARM Thumb-2 时为 160K 二进制大小
- 针对低内存消耗进行了大量优化
- 使用 C99 编写，实现了最大可能的移植性
- 快照支持将 JavaScript 源代码预编译为字节码
- 成熟的 C API，易于嵌入应用程序

本文将介绍如何为 ESP8266 设备编译 JerryScript。

步骤 1：搭建构建环境

为了编译 ESP8266 上的 JerryScript，我们需要先安装编译所需要的软件，即跨平台编译工具链。

这里的示例是在 Debian/Ubuntu（Linux）上运行的。

1.a) 对于 x86 系统来说，安装下面的工具：

```
sudo apt-get install git autoconf build-essential gperf \
   bison flex texinfo libtool libncurses5-dev wget \
   gawk libc6-dev-i386 python-serial libexpat-dev

sudo mkdir /opt/Espressif
sudo chown $USER /opt/Espressif/
```

1.b) 对于 x64 系统来说，则是：

```
sudo apt-get install git autoconf build-essential gperf \
    bison flex texinfo libtool libncurses5-dev wget \
        gawk libc6-dev-amd64 python-serial libexpat-dev

sudo mkdir /opt/Espressif
sudo chown $USER /opt/Espressif/
```

2) 接着，我们需要先安装 crosstool-NG，步骤如下：

```
cd /opt/Espressif
git clone -b lx106-g++-1.21.0 git://github.com/jcmvbkbc/crosstool-NG.gitcd
    crosstool-NG
./bootstrap && ./configure --prefix=`pwd` && make && make install
./ct-ng xtensa-lx106-elf
./ct-ng build
```

其 crosstool-NG 的路径添加到 .profile 文件中：

```
PATH=$PWD/builds/xtensa-lx106-elf/bin:$PATH
```

3) 随后，则是官方的 Espressif SDK：

```
cd /opt/Esprissif
git clone https://github.com/espressif/ESP8266_RTOS_SDK.git
    ESP8266_RTOS_SDK.git
ln -s ESP8266_RTOS_SDK.git ESP8266_SDK
cd ESP8266_SDK
git checkout -b jerry a2b413ad2996450fe2f173b6afab243f6e1249aa
```

我们使用具有 stdlib.h 等的 1.2.0 版本的 SDK。最新的 1.3.0 版本，在撰写本文档时，则是不能工作的。

设置两个环境变量，如 .profile 文件：

```
export SDK_PATH=/opt/Espressif/ESP8266_SDK
export BIN_PATH=(to output folder path)
```

4) 设置 Xtensa 库及其头文件

```
cd /opt/Espressif/ESP8266_SDK
wget -O lib/libhal.a
    https://github.com/esp8266/esp8266-wiki/raw/master/libs/libhal.a
```

5) ESP 镜像工具

```
cd /opt/Espressif
wget -O esptool_0.0.2-1_i386.deb
    https://github.com/esp8266/esp8266-wiki/raw/master/deb/esptool_0.0.2-1_i386.deb
sudo dpkg -i esptool_0.0.2-1_i386.deb
```

6) ESP 上传工具

```
cd /opt/Espressif
git clone https://github.com/themadinventor/esptool esptool-py
sudo ln -s $PWD/esptool-py/esptool.py
    crosstool-NG/builds/xtensa-lx106-elf/bin/esptool.py
```

步骤 2：为 ESP8266 SDK 打 JerryScript 补丁

由于 iram 相当的小，适合所有的代码，但链接器会试图把代码放在那里。要强制将 JerryScript 代码放置在 irom 部分，需要更改顺序并告诉链接器，如下所示：

```
diff --git a/ld/eagle.app.v6.common.ld b/ld/eagle.app.v6.common.ld
index caf8e32..dadaceb 100644
--- a/ld/eagle.app.v6.common.ld
+++ b/ld/eagle.app.v6.common.ld
@@ -151,6 +151,21 @@ SECTIONS
   } >dram0_0_seg :dram0_0_bss_phdr
  /* __stack = 0x3ffc8000; */

+  .irom0.text : ALIGN(4)
+  {
+    _irom0_text_start = ABSOLUTE(.);
+    *(.irom0.literal .irom.literal .irom.text.literal .irom0.text
    .irom.text)
+    *(.literal.* .text.*)
+    _irom0_text_end = ABSOLUTE(.);
+
+    _jerry_text_start = ABSOLUTE(.);
+    *\libjerryentry.a:*(.text*)
+    *\libjerrycore.a:*(.text*)
+    *\libjerrylibm.a:*(.text*)
+    _jerry_text_end = ABSOLUTE(.);
+
+  } >irom0_0_seg :irom0_0_phdr
+
   .text : ALIGN(4)
   {
```

```
26      _stext = .;
27 @@ -199,13 +214,6 @@ SECTIONS     _lit4_end = ABSOLUTE(.);
28    } >iram1_0_seg :iram1_0_phdr
29
30 -  .irom0.text : ALIGN(4)
31 -  {
32 -    _irom0_text_start = ABSOLUTE(.);
33 -    *(.irom0.literal .irom.literal .irom.text.literal .irom0.text
       .irom.text)
34 -    *(.literal.* .text.*)
35 -    _irom0_text_end = ABSOLUTE(.);
36 -  } >irom0_0_seg :irom0_0_phdr
37  }
```

第二个文件是修改 irom 大小，以便它可以容纳所有的代码和数据。这个可以通过给另一个 SPI_SIZE_MAP 来完成。为此，我使用它来执行修改。

```
1  /* get ROM code address */
2 diff --git a/ld/eagle.app.v6.ld b/ld/eagle.app.v6.ld
3 index 3e7ec1b..4a9ab5b 100644
4 --- a/ld/eagle.app.v6.ld
5 +++ b/ld/eagle.app.v6.ld
6 @@ -26,7 +26,7 @@ MEMORY
7    dport0_0_seg :                        org = 0x3FF00000, len = 0x10
8    dram0_0_seg :                         org = 0x3FFE8000, len = 0x14000
9    iram1_0_seg :                         org = 0x40100000, len = 0x8000
10 -  irom0_0_seg :                         org = 0x40240000, len = 0x3C000
11 +  irom0_0_seg :                         org = 0x40240000, len = 0xB0000
12  }
13
14  INCLUDE "../ld/eagle.app.v6.common.ld"
```

需要得到 **setjmp/longjmp**

这一步可以从 SDK 解压、复制来获取：

```
1 cd ~/harmony/jerryscript/targets/esp8266/libs
2 ar -xv $SDK_PATH/lib/libcirom.a lib_a-setjmp.o
```

步骤 3：构建 JerryScript

在完成了下面的步骤之后，剩下的事件就简单了：

```
cd ~/harmony/jerryscript
# clean build
make -f ./targets/esp8266/Makefile.esp8266 clean
# or just normal build
make -f ./targets/esp8266/Makefile.esp8266
```

输出文件应放在 $BIN_PATH 变量所指定的路径上

步骤 4：烧录到 ESP8266 ESP-01 开发板上

这里使用的是 ESP8266 ESP-01(WiFi) 作为演示。其它开发板也是非常相似的。

GPIO0 and GPIO2

在烧录固件之前，您需要按照步骤操作：

1. 关闭 ESP8266
2. 连接 GPIO0 到 GND、GPIO2 到 VCC
3. 启动 ESP8266
4. 烧录

烧录

通过以下的脚本，即可以实现：

```
make -f ./targets/esp8266/Makefile.esp8266 flash
```

默认 USB 设备是 `/dev/ttyUSB0` 。如果你的结果是不同，就把这个值附给 USBDEVICE，如：

```
USBDEVICE=/dev/ttyUSB1 make -f ./targets/esp8266/Makefile.esp8266 flash
```

步骤 5：运行

运动步骤

1. 关闭 ESP8266
2. 断开 GPIO0、GPIO2
3. 启动 ESP8266

这里的示例程序适用于 LED 和具有以下连接的 SW。

1. 将 GPIO2 连接到 LED -> 4K 电阻 -> GND
2. 在 VCC -> 4K 电阻和 GND 之间连接 GPIO0

如果 GPIO0 为高电平，则 LED 亮起，反之亦然。

附录 A：示例代码

main.js

```
function sysloop(ticknow) {
  blink();
};
print("main js OK");
```

blink.js

```
var check = 1;

function blink() {
  var inp = gpio_get(0);
  var blk = (check > 8) ? 1 - inp : inp;
  gpio_set(2, blk);
  check = check >= 10 ? 1 : check+1;
}

// GPIO 0 as input
// GPIO 2 as output
gpio_dir(0, 0);
gpio_dir(2, 1);

print("blink js OK");
```

原文链接:https://www.wandianshenme.com/play/javascript-engine-jerryscript-esp8266-build-iot-a