ELECTRICALENGINEERING

ESP8266 eLUA (NodeMCU) vs MicroPython

13

7

I am searching for an unbiased, up-to-date comparison of the eLUA (NodeMCU) and MicroPython on ESP8266.

I can only find very superficial reports/blogs of users trying out one or the other. - All of which lacking entirely of technical details.

The closest thing I could find is this probably hopelessly outdated and hard to understand comparison by the MicroPython project.

I would be interested in the obvious questions:

- flash usage
- RAM usage of VM after bootup
- RAM usage in usual use
- execution model (i.e. how are the ESP8266 "tasks" mapped?)
- execution performance
- ease of extension (i.e. module addition)
- anything else that might be relevant

From studying the documentation I believe to have understood the following:

- NodeMCU has fairly fine-grained build options that allows building only required modules. This appears to allow working with small flash sizes. For Micropython 512 KB seems to be the absolute lower limit in which case no space for user-defined code remains. Not sure how this compares to NodeMCU.
- MicroPython has a built-in WebREPL that is auto-configured by default. NodeMCU seems to have nothing similar built-in.
- NodeMCU seems to benefit currently from a larger community, presumably due to it having been around longer.
- MicroPython documentation is fairy informal, lacking entirely when it comes to extending the C code. NodeMCU documentation seems to be excellent.

esp8266 python

edited Feb 15 '17 at 16:56

asked Feb 13 '17 at 18:37
ARF
2,892 ● 3 ● 33 ● 57

Can you state exactly your MUST HAVE requirements and nice to have :>) please? Ram size, array size, floating or integer, RAM efficiency, CPU speed , response times, etc. etc – Sunnyskyguy EE75 Apr 15 '17 at 16:56

1 Answer

16

Here's a slightly different approach in stead of a Lua vs. Python shootout:

Six of the most popular ESP8266 "runtimes":

1. **AT Command SET.** Popular when the 8266 is paired with another MCU. Communicates via the serial port. ~64k of 128k RAM available.
2. **MicroPython.** A MicroPython script interpreter with user friendly GUI that can be accessed via the serial port or WIFI/IP. ~30k of 128k RAM available.
3. **Lua/NodeMCU.** A LUA script interpreter with user friendly GUI that can be accessed via the serial port. ~40k of 128k RAM available.
4. **JavaScript/Espruino.** A JavaScript interpreter with user friendly GUI that can be acess via the serial port or WiFi/IP. ~20k of 128k RAM available.
5. **C/IDE-12E.** ESP8266 flashing tool & C libraries/tools using the standard Arduino IDE. ~80k of 128k RAM available.
6. **C/ESP8266_SDK.** C libraries/tools from the manufacturer. Also a collection of example applications. ~512k Flash. Guesstimate 80k of 128k RAM available.

The key insight is that the **bulk of the code is common.** The primary libraries in 1-5 all originate from 6. Beneath a thin layer of AT/Python/LUA/JavaScript/C **the primary code is practically identical.** That means performance (RAM, FLASH, execution) is similar too.

Since you seem concerned about speed and RAM (flash is generally OK) how about option #5? Arduino is a usable IDE with a large collection of examples. You could have your first code running in less than an hour and would likely outperform any of the scripting engines.

In the absence of significant memory usage differences **I would choose MicroPython** due to the greater number of libraries and an active online community with IRC webchat. Documentation for adding C modules has improved.

Option #6 provides you with the **highest potential for optimization** but at higher complexity and steeper learning curve.

Lastly a good ESP8266 rule-of-thumb: Each TCP/IP connection can consume up to ~3k of memory. **Always expect less than 5 *simultaneous* connections!**

TL;DR: ESP8266 applications have most of their code in common and perform similarly. So choose the script engine you like or step up to C/IDE-12E. Don't expect more than 5 simultaneous IP connections.

edited May 2 '17 at 19:34

answered May 2 '17 at 9:17
neonzeon
1,609 ● 7 ● 12

Relevant: Interesting story on how Ivan Grokhotkov and the ESP8266 community ported the ESP8266 to be programmed from the Arduino IDE : makezine.com/2015/04/03/… – neonzeon May 2 '17 at 19:31

Just wanted to say thank you for summarizing this so. I was actually searching for this information and you saved me a bunch of time, plus added some perspectives that are good to know.. – Crossfit_and_Beer Mar 7 at 3:42