

Řešení čtvrtého kola

1. Další kolo nezačalo příliš slibně. Přesto že jsme se drželi doporučeného zdroje [1], nastal problém při konfiguraci sériového portu počítače Raspberry. Bylo totiž nutné upravit systémové soubory, které v naší instalované verzi Raspbianu (Jessie) již nebyly v adresáři, který popisoval návod. Proto jsme Raspberry nejdříve přeinstalovali na Raspbian Wheezy. Dále už bylo dohledání souborů snadné.

Příkazem `sudo nano /etc/inittab` jsme otevřeli soubor *inittab*, ve kterém jsme zakomentovali poslední řádek týkající se sériové linky. *Inittab* vypadá tedy takhle:

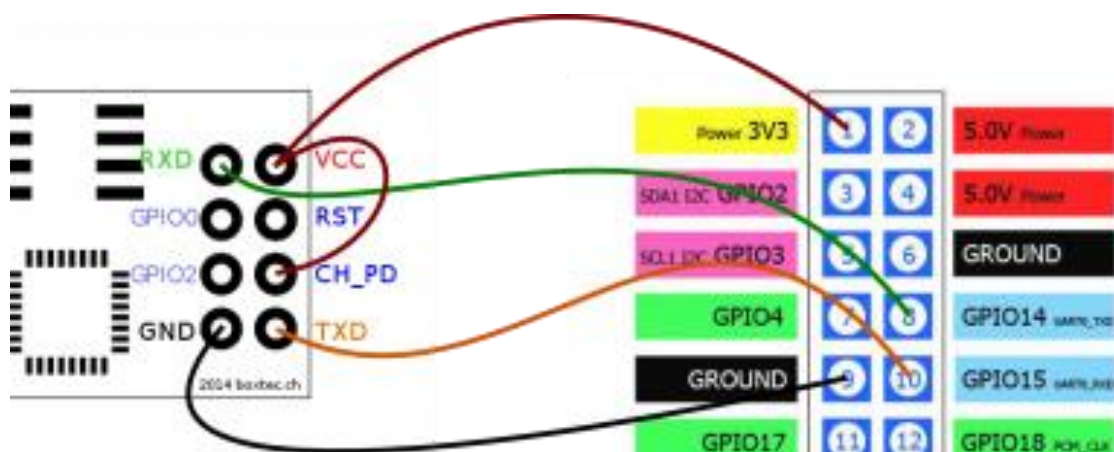
```
#Spawn a getty on Raspberry Pi serial line
#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

Další soubor, *cmdline.txt* jsme editovali příkazem `sudo nano /boot/cmdline.txt`. Z tohoto souboru opět odstraníme referenci na sériovou linku – `console=ttyAMA0, 115200`.

Výsledek vypadá následovně:

```
dwc_otg.lpm_enable=0 root=/dev/mmcblk0p2 rootfstype=ext4
elevator=deadline rootwait
```

Ted' se můžeme pustit do fyzického zapojování modulu k Raspberry, které vypadalo takto:



Obrázek č. 1: Zapojení ESP modulu

GPIO0 a GPIO2 fungují stejně jako na Raspberry, využívají 3,3V logickou úroveň. Je možné využít i pulsně šířkovou modulaci.

Nyní potřebujeme program, který bude po sériové lince komunikovat s modulem. Náš tým zvolil program *minicom*. Program jsme nainstalovali příkazem `sudo apt-get install minicom`. ESP modul se spouští ve dvou módech, běžném a flashovacím. Nyní potřebujeme modul spustit ve flashovacím módu, proto modul odpojíme z napájení, uzemníme pin GPIO0 a opět připojíme napájení.

Po instalaci a vepsání tohoto příkazu do RPi konzole `sudo minicom -b 115200 -o -D /dev/ttyAMA0` se zobrazí terminál modulu ESP. V defaultním stavu je na modulu firmware odpovídající na AT příkazy [2], takže po zadání některého z nich by modul měl odpovědět.



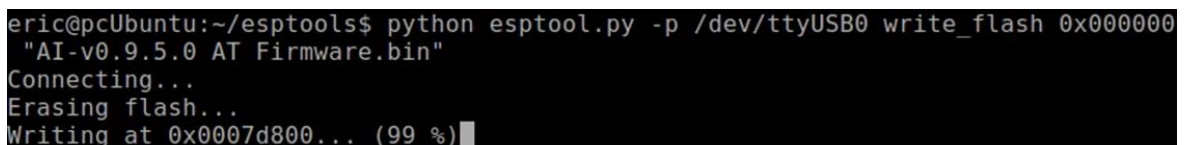
```
AT+GMR
002009.5 (b1)
compiled @ Dec 25 2014 21:40:28
AI-THINKER Dec 25 2014
```

Obrázek č. 2: Odezva AT příkazů

Tak se také stalo a náš tým pokračoval v zadání. Konkrétně jsme flashovali ESP modul doporučeným NodeMCU. Nejdříve jsme stáhli firmware z Githubu [3] (je důležité stáhnout verzi 0.9.5, v poslední verzi totiž nefunguje teplotní čidlo DS18B20), stáhli jsme verzi float, integer totiž podporuje pouze celá čísla, což není praktické, pokud chceme firmwarem komunikovat s teplotním čidlem. Program, kterým jsme flashovali, se nazývá *esptool* [4]. Dále jsme stažený firmware uložili do stejného adresáře s *esptool.py* a flashovali jsme příkazem:

```
python esptool.py -p /dev/ttyAMA0 write_flash 0x000000
nodemcu_float_0.9.5_20150627.bin
```

a taková byla odezva:



```
eric@pcUbuntu:~/esptools$ python esptool.py -p /dev/ttyUSB0 write_flash 0x000000
"AI-v0.9.5.0 AT Firmware.bin"
Connecting...
Erasing flash...
Writing at 0x0007d800... (99 %)█
```

Obrázek č. 3: Flashování

Po flashování se změní modulační rychlost, kterou modul komunikuje z 115200 na 9600 baudů, proto k opětovnému přístupu k terminálu modulu tuto rychlost musíme změnit.

2. Po zprovoznění modulu jsme hned chtěli vyzkoušet funkčnost přeflashovaného firmwaru NodeMCU. Ideální se jevil právě druhý úkol únorového zadání, rozblikání LED diody pomocí modulu. Odpojili jsme tedy GPIO0 od GND a místo toho jsme GPIO0 připojili k LED diodě chráněné malým 300 ohmovým rezistorem. Po přeflashování již modul neovládáme AT příkazy, nýbrž skriptovacím jazykem LUA. Navíc už na GitHubu jsme našli spoustu zajímavých příkladů LUA kódů, takže z této stránky jsme zkopírovali kostru kódu:

```
pin = 3
gpio.mode(pin, gpio.OUTPUT)
gpio.mode(pin, gpio.HIGH)
```

Hodnotu proměnné je nutno přepsat podle pinu, na kterém se LEDka nachází. NodeMCU značí GPIO0 jako pin 3 a GPIO2 jako pin 4 [5], poněvadž byla naše LEDka na GPIO0, dosadili jsme za jedničku trojku.

Těchto pár řádků nám tedy rozsvítí připojenou LED. To je fajn, ale při psaní delšího kódu tento způsob není dvakrát pohodlný a proto jsme se v dalším kroku rozhodli použít program *luatool*.

3. K zapojení DS18B20 potřebujeme druhý pin GPIO2 (viz proměnná *pin* v kódu *dallas.lua*) na něj jsme dali opět malý 300 ohmový rezistor a za něj datový kabel teploměru. Nakonec jsme připojili ještě zbývající vývody, VCC k napájení a GND na zem. Na obrázku č. 6 je celé zapojení.

zapojení	DS18b20
GPIO2 (sériově s rezistorem)	Data
GND	GND (na obr č. 6 vpravo)
Vcc	Vcc (na obr č. 6 vlevo)

Jak jsem již naznačil, *luatool* je program, který nahraje do modulu spustitelný soubor, což práci velice usnadní. Tento program lze stáhnout přímo: `git clone https://github.com/4refr0nt/luatool.git`. Poté jsme vytvořili v Raspberry soubor *dallas.lua* a nahráli jej pomocí *luatoolu*, příkazem `./luatool.py --port /dev/ttyUSB0 --src dallas.lua --dest dallas.lua -verbose`.

Sobury lze do modulu nahrávat **pouze pokud je vypnutý *minicom*** a po nahrání souboru je nutno modul restartovat, snadno to lze provést příkazem `node.restart()` v konzoli vyvolané *minicomem*. Také jsme editovali soubor *init.lua*, jenž jsme stáhli společně s *luatoolem*, *init.lua* je soubor spouštějící se ihned

po spuštění ESP modulu. V souboru je kód, který připojí ESP modul na WiFi síť a vypíše svou IP adresu. Při editaci *init.lua* musí programátor dávat pozor na chyby, aby se kód nezacyklil a nerestartoval modul ihned po spuštění.

4. Naměřené hodnoty s ovládáním LEDky jsme měli ve čtvrtém úkolu zobrazit na webovém rozhraní ESP modulu. Proto jsme zamířili opět na GitHub, kde jsme stáhli NodeMCU a zde jsme se inspirovali kódem rozsvěčejícím LEDku. O něco těžší bylo však implementovat funkci měřící teplotu do webového rozhraní. I to se nám však podařilo a webové rozhraní je přístupné, pokud uživatel připojený na stejné síti zadá do webového prohlížeče IP adresu ESP modulu. Kód v jazyce Lua má název *tepweb2.lua*. Zde je náhled jednoduchého webového rozhraní:

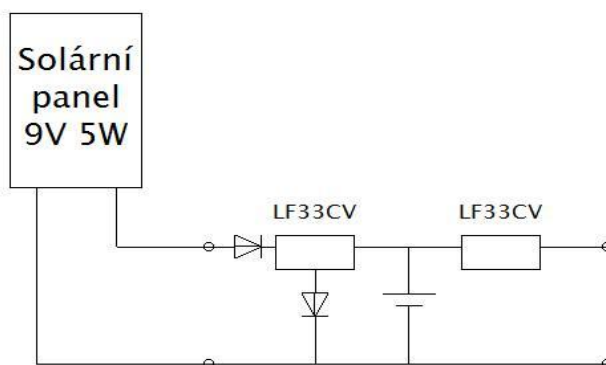


Obrázek č. 4: Webové rozhraní modulu

Nakonec jsme se snažili také posílat aktuální teplotu z ESP přímo na webové stránky www.ioe.zcu.cz, proto jsme stáhli dvě knihovny zprovozňující http protokol na ESP – *http.lua* a *httpDL.lua*. S těmito knihovnami jsme pracovali v kódu *rozhrani.lua*, bohužel tento kód nefungoval spolehlivě, teplotu z ESP sice zapsal, ale po dalších pokusech se modul restartoval.

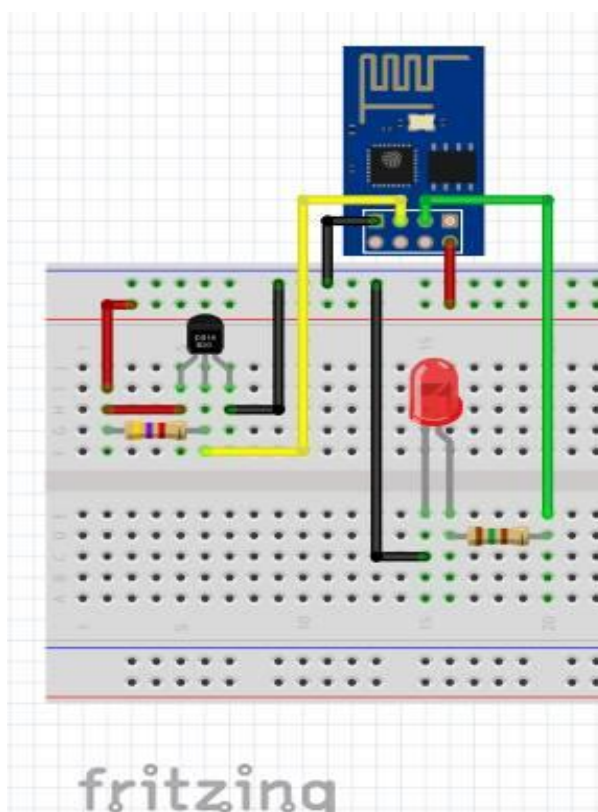
5. V posledním úkolu jsme měli udělat bezdrátový snímač teploty s dálkově ovládanou LED. Napájení jsme měli řešit monočlánky, ale poté co jsme si spočetli, že by vydržely asi půl dne (při neustálém snímání teploty), rozhodli jsme se místo toho, sestavit zdroj vlastní – solární panel. Zde nastal problém, poněvadž panel napájel v závislosti na proměnné intenzitě slunečního světla (za slunného dne by ESP, stavěné na 3,3 V, shořelo a v noci by se zase vypínalo). Proto jsme zdroj doplnili nabíjecími monočlánky. Panel musel být chráněn předřadnou usměrňovací diodou, hned za ní se nachází stabilizátor LF33CV který nám s další diodou (paralelně k monočlátku) stabilizuje napětí na 4 V, kterými se článek musí nabíjet. Li-Ion monočlánek má výstupní napětí 3,7 V, tudíž musí být také omezen, k čemuž jsme opět použili

stabilizátor stejného typu. Na výstupu tohoto zdroje je tedy 3,3 V dostačujících na napájení ESP modulu.



Obrázek č. 5: Schéma zdroje

Toto zapojení jsme nakreslili na cupertitovou desku a vyřešili jsme ho plošným spojem. Poté jsme modul s monočlánkem umístili do jídelního boxu a umístili na střechu školy, jak můžete vidět v příloženém videu.



Obrázek č. 6: Schéma zapojení ESP 1

Zdroje:

- [1] http://www.esp8266.com/wiki/doku.php?id=raspberrypi:getting_started
- [2] http://wiki.iteadstudio.com/ESP8266_Serial_WIFI_Module
- [3] <https://github.com/nodemcu/nodemcu-firmware/releases>
- [4] <https://github.com/themadinventor/esptool>
- [5] <https://learn.adafruit.com/adafruit-huzzah-esp8266-breakout/using-nodemcu-lua>