NodeMCU

De Borrekens Gauthier

Intro

The NodeMCU hardware, not to be confused with the same-titled firmware, is a single-board microcontroller. It uses the 'ESP8266' processor which is a Wi-Fi micro-chip with fully integrated TCP protocol.
The original manufacturers stopped producing the board but a lot of spinoffs are still on the market for a very affordable price. It comes with the NodeMCU firmware, which is programmed in lua. It is not necessary to keep this firmware as the MCU is flashable and a lot of other IDE's exist which make it possible to program in C++, python, Arduino IDE and others.

Materials and Methods

As this is just research, no materials were used. The sources will be added.

Information & conclusion

Specifications

Architecture: Xtensa lx106

CPU frequency: 80MHz overclockable to 160MHz

Total RAM available: 96KB (part of it reserved for system)

BootROM: 64KB

Internal FlashROM: None

External FlashROM: code and data, via SPI Flash. Normal sizes 512KB-4MB.

GPIO: 16 + 1 (GPIOs are multiplexed with other functions, including external FlashROM, UART, deep sleep wake-up, etc.)

UART: One RX/TX UART (no hardware handshaking), one TX-only UART.

SPI: 2 SPI interfaces (SPI/HSPI) (one used for FlashROM).

I2C: No native extenal I2C (bitbang implementation available on any pins).

I2S: 1.

Programming: using BootROM bootloader from UART. Due to external FlashROM and always-available BootROM bootloader, ESP8266 is not brickable.

WiFi (interface): 1 shared controller, supports 2 interfaces - AP (Access Point, 4 client limit) and STA (Station/Client Mode)

WiFi (security): Open, WEP, WPA/WPA2, limited support for 802.1x/WPA2-Enterprise

(Petl, sd)

Programming languages

The ESP8266 SDK is developed in C but this is on a low abstract level. This is why there are environments or 'script interpreters' that are written in more user-friendly languages.

As the first paragraph briefly mentioned, The NodeMCU firmware is written in 'lua' but there are some other environments made for the ESP8266 chip so that other languages can be used. A brief summary:

- **AT command SET**: An old serial communication protocol, takes least amount of RAM
- **MicroPython**: Environment that uses python and is user-friendly. Next to the serial port, it can also be communicated through Wi-Fi.

- **Arduino IDE**: Good for programmers who are used to programming in Arduino. Also very convenient to use. Contains a lot of libraries which we need, like the UDP WiFi communication. Also written in C so this makes it relatively light on the RAM.
- **NodeMCU**: Written in lua which makes it easy to use but causes unpredictable bugs according to a lot of people online. Also takes a relatively large chunck of RAM.

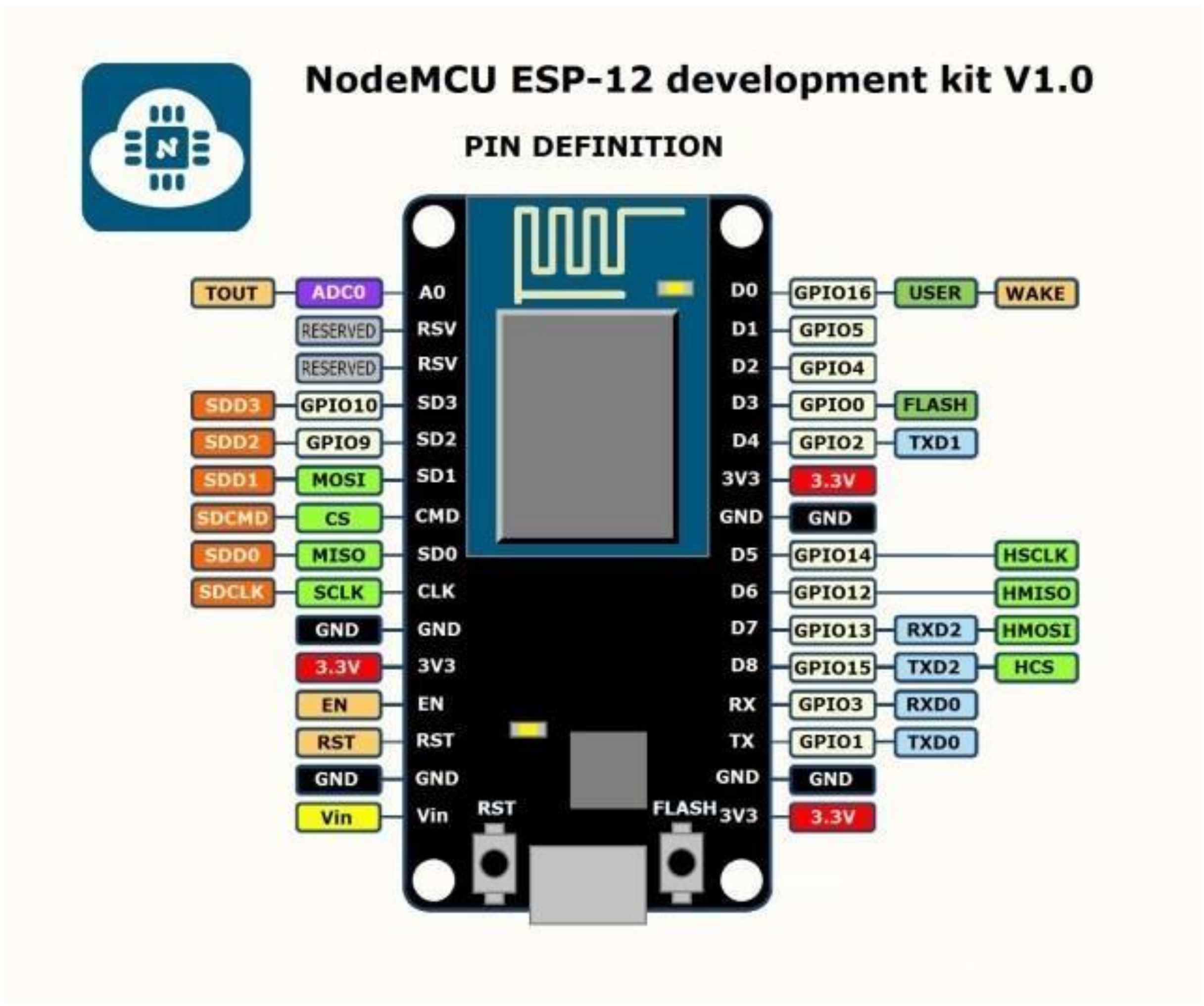
(neonzeon, sd)

Conclusion:

At this point we have not experimented with any environments so we cannot say for sure which one to use. Once we review some existing projects, define our software needs and do some testing we can start choosing our environment.

Pinout

Note: This can differ according to the spinoff we purchase. As we haven't got our NodeMCU's at this time, we will add the pinout of the original NodeMCU V2. Once we get our spinoff we will update this accordingly.



(nodeMCU, sd)

Referencelist

BIBLIOGRAFIE

neonzeon. (sd). *nodeMCU vs microPython*. Opgeroepen op Maart 13, 2018, van https://electronics.stackexchange.com/questions/286328/esp8266-elua-nodemcu-vs-micropython nodeMCU. (sd). Opgeroepen op Maart 13, 2018, van node mcu devkit instruction: https://github.com/nodemcu/nodemcu-devkitv1.0/blob/master/Documents/NODEMCU-DEVKIT-V1.0-INSTRUCTION-EN.pdf

Petl. (sd). *ESP8266 wiki*. (Reddit) Opgeroepen op Maart 13, 2018, van https://www.reddit.com/r/esp8266/wiki/index