# nRF24-L01 + ESP8266 Mesh Network

## I plan on building a mesh network consisting of four 10x7x2.3cm transparent plastic boxes, each equipped with the following hardware:

- 1x Nrf24-L01 Long range 2.4GHz transceiver with antenna [ 0.0009 - 15mA ]
- 1x Nrf24 base module for power regulation/data "smoothing"/interference reduction.
- 1x ESP8266-12F WiFi module/microprocessor [ 0.01 - 170mA ]
- 1x 0.9" 128x32 OLED I2C Display [ 8 - 12mA ] (Will be disabled unless wake button is pressed)
- 1x Adjustable step down converter => ~4.5V to 3.3V [ Advertised efficiency: 97% expected efficiency: 80-90% ]
- 3x AA Battery + Holder => connected in series to reach ~4.5V [ 400 - 2400mAh (* ~0.9 because of efficiency of step down converter) ]
- 1x 2-Position power switch
- 1x momentary push button => Wake display / GUI input
- 1x piezo buzzer for locating the devices or indicating an error/battery-low state.

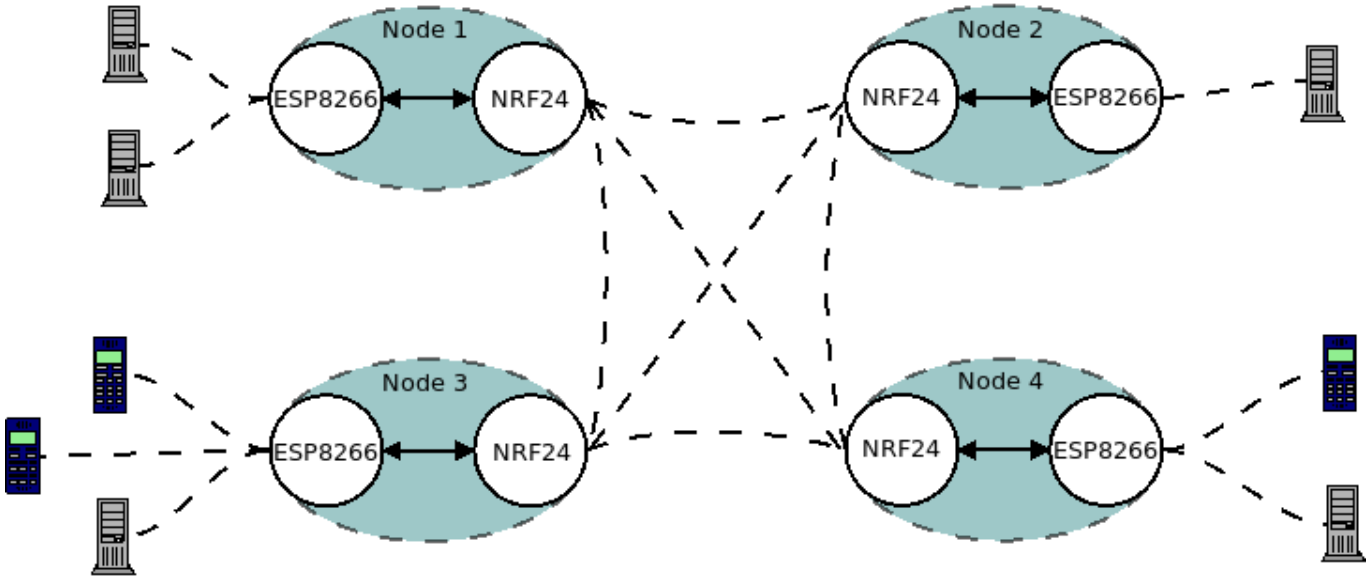
Expected cost for 4pcs: ~30$.
Actual cost: 34$. (All components were ordered from AliExpress.com)

Pessimistic battery-runtime: (400mAh * 0.8) / (15mA + 170mA + 12mA) = ~1.6h
Optimistic battery-runtime: (2400mAh * 0.9) / (4mA + 80mA + 0mA) = ~25.7h
Realistic battery-runtime: (1.6h + 25.7h) / 2 = ~13.7h

## Configuration:

I expect a range of 400-700m between two of the Nrf24 chips and a range of ~8m for the ESP8266s. The idea is to provide a (pretty slow) LAN network by connecting the Nrf24s with each other and using the ESPs, which would each be physically connected to a NRF, as access points for regular WiFi devices. I expect a data transfer speed of around 1-2mBit.

https://tmrh20.github.io/RF24Mesh
https://tmrh20.github.io/RF24Network
https://tmrh20.github.io/RF24Mesh




## Potential issues:

- Interference (ESP/NRF/power-supply) => Shielding / Input voltage smoothing?
- Batteries/PS - Amperage/Voltage
- Lower-than-expected transfer speeds (various reasons)
- How would I even send network traffic over the NRFs?
    - MAC/IP-addresses?

## Hardware setup (used for testing):

I am using two NodeMCU dev boards, each with their own nRF24-L01 chip, for testing. **The nRFs are connected to the ESPs via the hardware SPI interface.**

| NRF24-L01 | NodeMCU |
|---|---|
| GND | GND |
| VCC | VCC |
| CE | D4 |
| SCK | D5 |
| MISO | D6 |
| MOSI | D7 |
| CSN | D8 |

🚨 **IMPORTANT: D8 must pulled low using a ~4.75k Ohm resistor to enable hardware serial.** 🚨

```
// D4 and D8 are the "interesting" pins.
RF24 radio(D4, D8);
```

## Other notes:

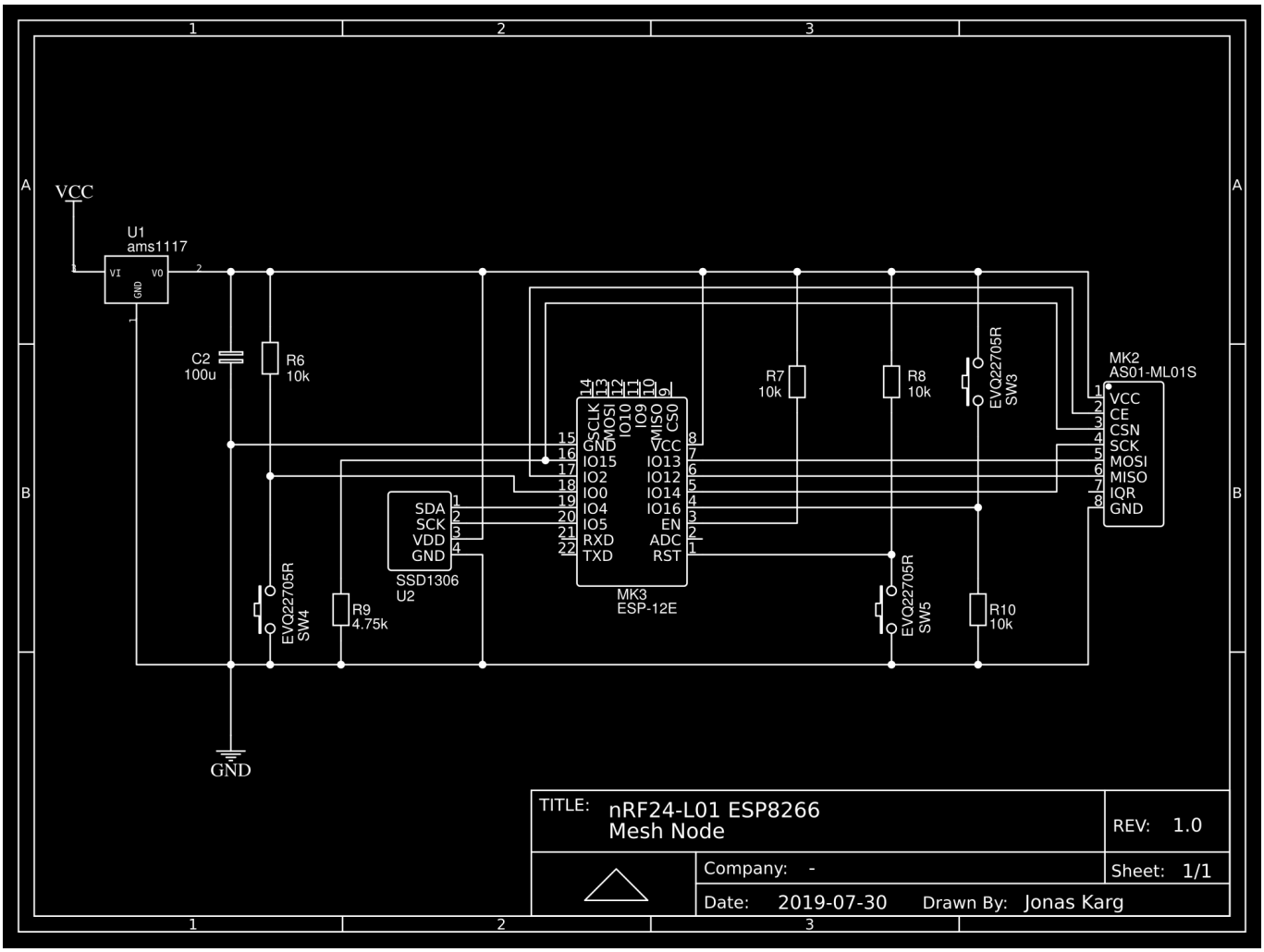
The ESP8266 has a Watch-Dog-Timer (WDT), which aims to prevent the board from freezing in e.g. a "endless" loop. To prevent the timer from expiring and thus resetting the ESP, you have to either disable it using `ESP.wdtDisable();`, adjust it for your purposes with `ESP.wdtEnable(MILLISECOND_AMOUNT);` or just calling `yield();` every once in a while (`yield()` is also called after every `loop()`);
Watch out for blocking operations in the libraries you use (e.g. `NRF24.waitAvailable()`).
A great tool for developing with the ESP8266 is **PlatformIO**.

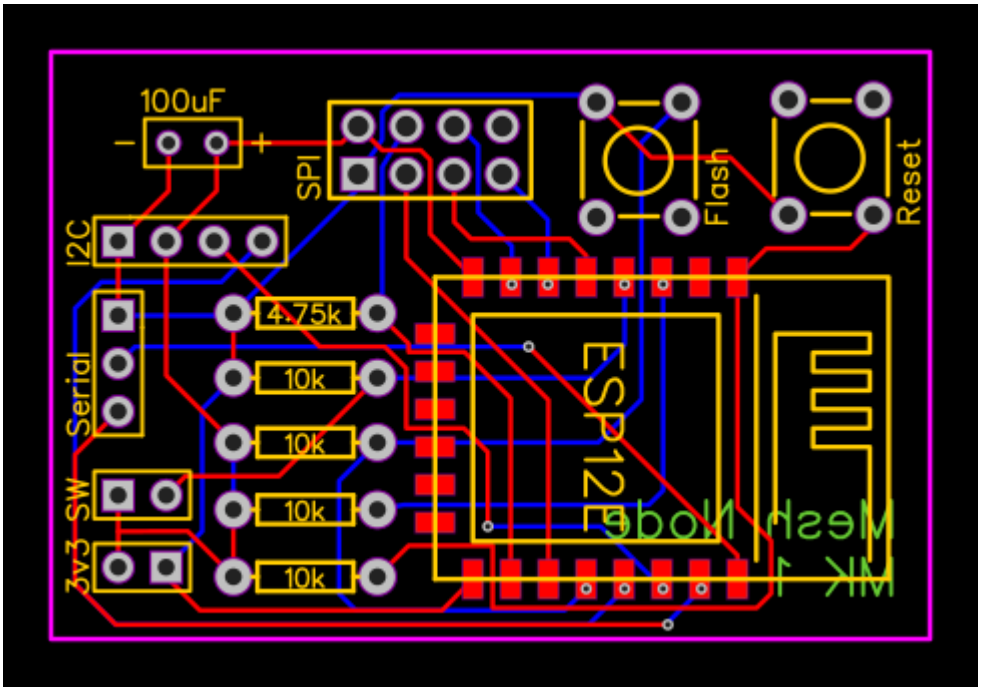
## References/Resources:

https://tmrh20.github.io/RF24Mesh
https://tmrh20.github.io/RF24Network
https://tmrh20.github.io/RF24Mesh (Probably not going to use this lib - I want a flat network)

https://en.wikipedia.org/wiki/Serial_Peripheral_Interface
https://techtutorialsx.com/2017/01/21/esp8266-watchdog-functions/
https://platformio.org/
https://docs.platformio.org/en/latest/boards/espressif8266/nodemcuv2.html?highlight=nodemcu
https://easyeda.com
"dia" - Diagram tool

## Prototype Schematics and PCB




## ToDo:

- Plan layout in case for min interference
- Waterproofing
  - Do I even care about that? (yeah - at least a bit of waterproofing..)
  - What about the antenna?

    - Buttons on inside or outside?
- What stuff on display?
    - Battery status.
    - Connection
        - How many nodes
        - Packet loss
- Wiring
    - Circuit diagram
    - Design a PCB