WIFI Controllable Textile LED Circuit
DAAN

This prototype demonstrates how to control a high power LED circuit with MOSFET's operated by a ESP2866. The circuit can therefore also work for other applications where higher power is needed while the regulating microcontroller can still run on low power. While building the circuit, we had to experiment with different components and if this circuit gets reproduced either the exact use of parts or good reading the Datasheets of the similar components is advised.

Max Current : 0,8 A
Max Voltage: 3,3 Volt
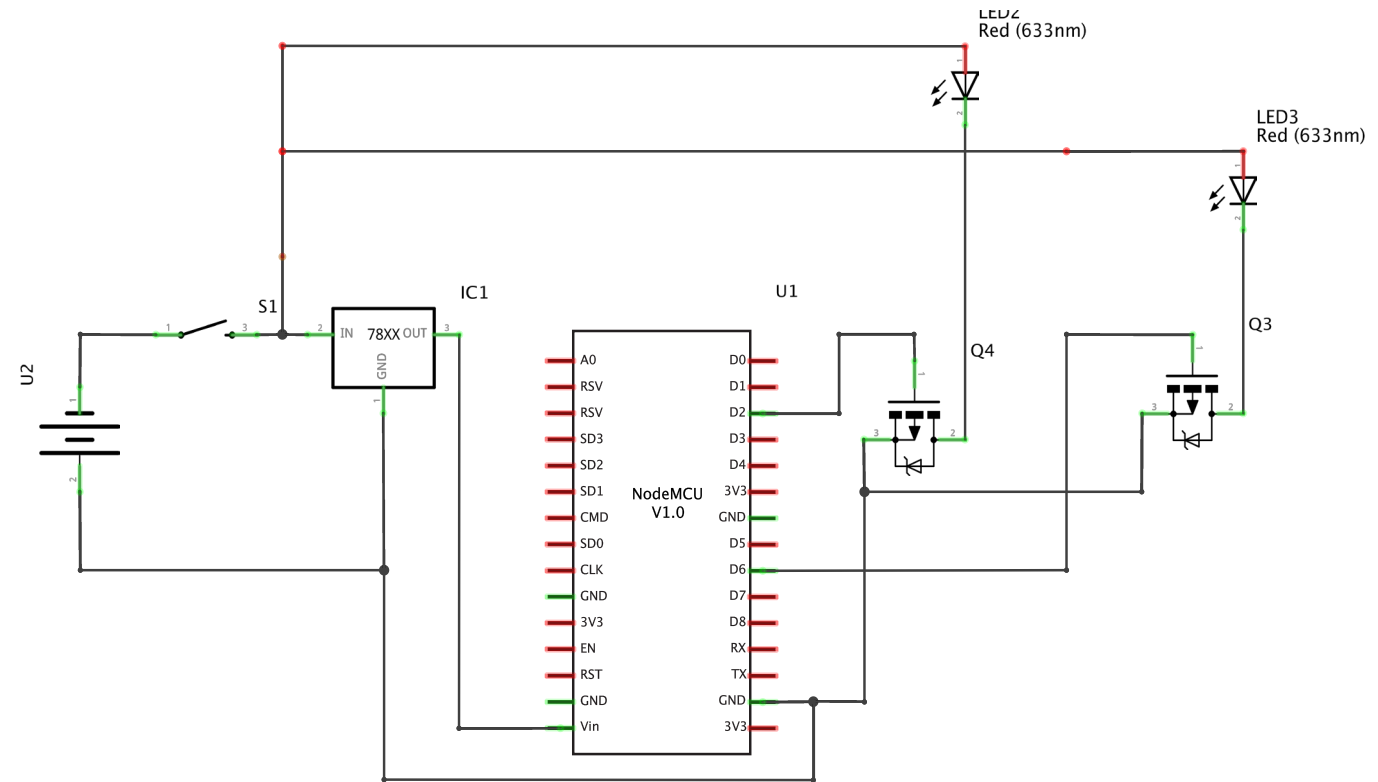**Regulated Voltage: 3,2 Volt**

List of parts:

Voltage Regulator:
LM1084IT-5.0

MOSFET:
MOSFET Infineon Technologies IRLZ34NSPBF
1 HEXFET 3.8 W D2PAK

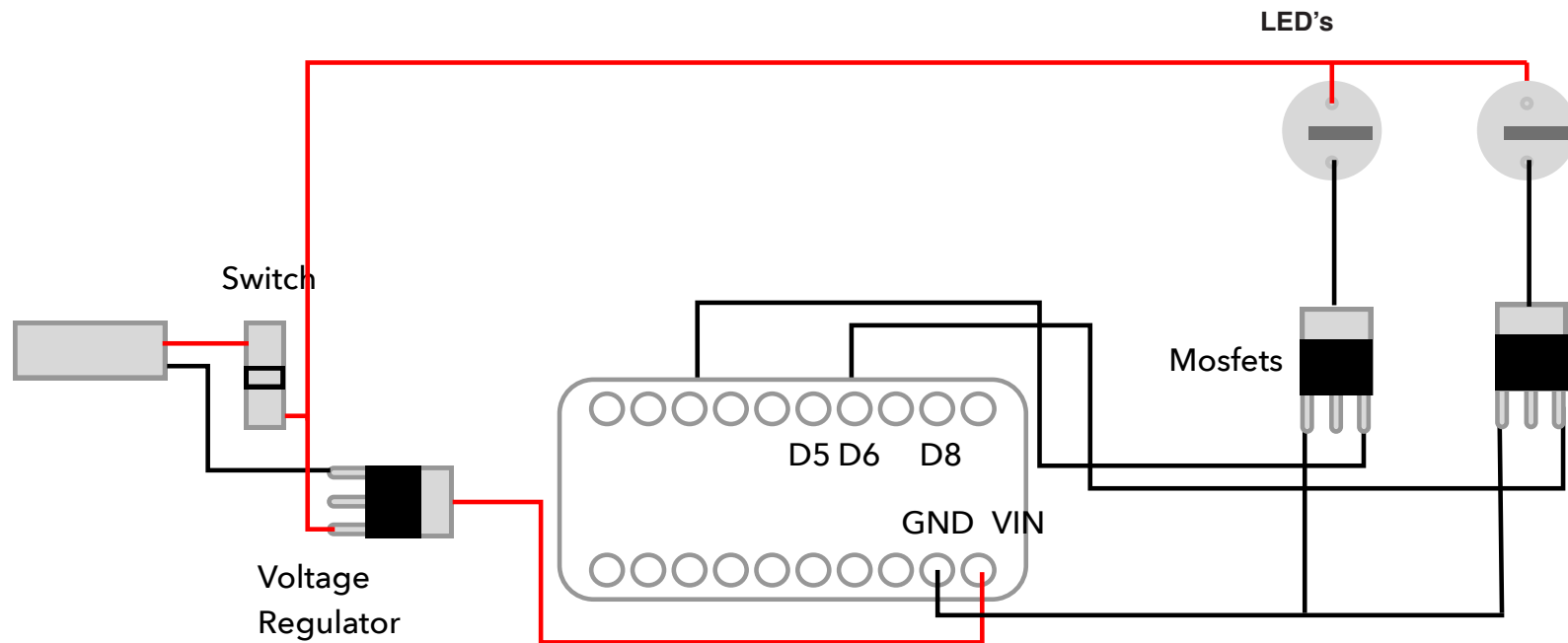Microcontroller:
NodeMCU-DevKit 1.0
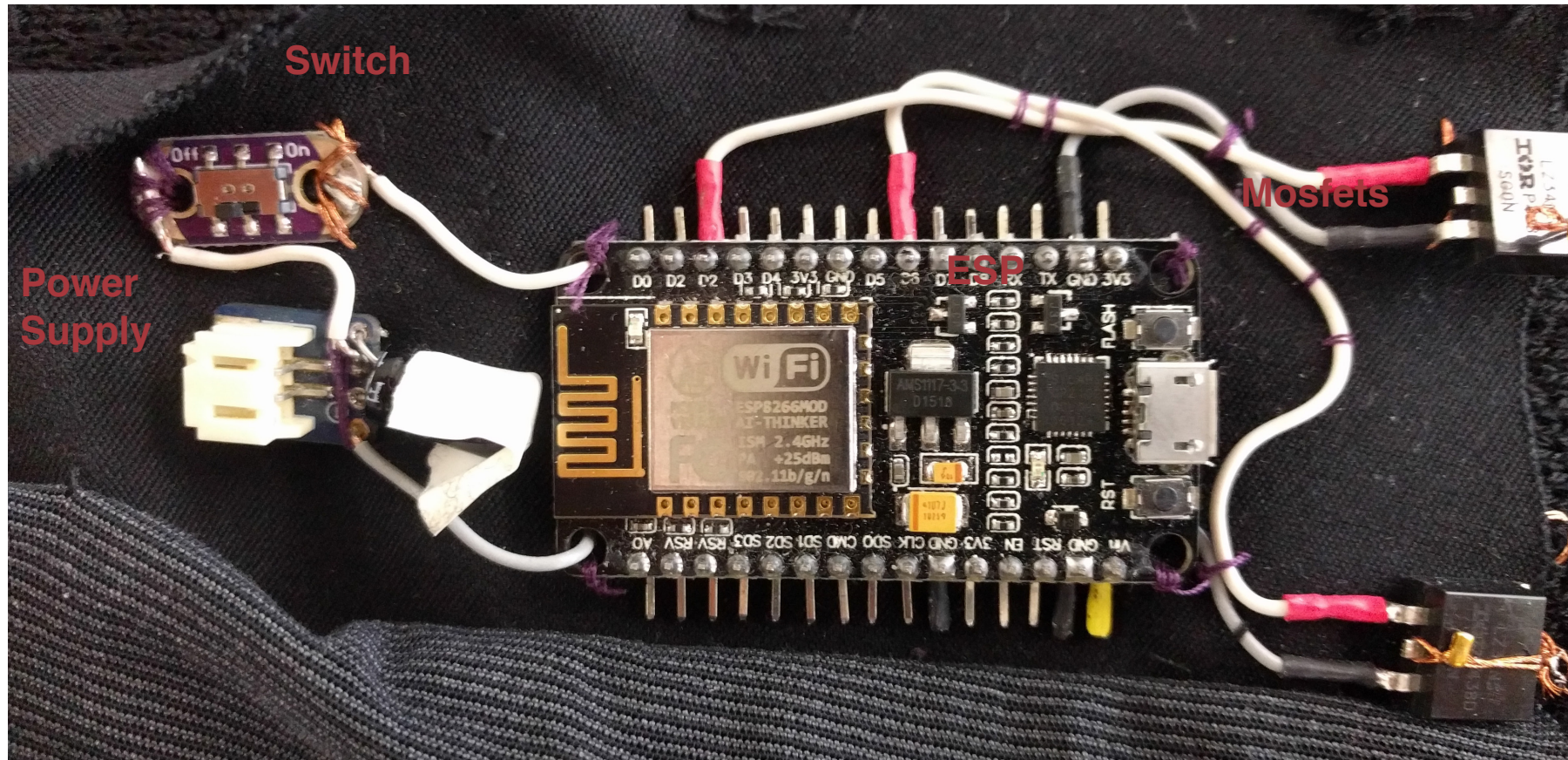
LED's:
BEKINOX VN12-3x275-175s

Circuit

LED2
Red (633nm)

LED3
Red (633nm)

S1

IC1

U1

IN 78XX OUT
GND

U2

NodeMCU V1.0	
A0	D0
RSV	D1
RSV	D2
SD3	D3
SD2	D4
SD1	3V3
CMD	GND
SD0	D5
CLK	D6
GND	D7
3V3	D8
EN	RX
RST	TX
GND	GND
Vin	3V3

Q4

Q3

fritzing

Circuit/Plan

LED's

Switch

Mosfets

D5 D6   D8

GND   VIN

Voltage
Regulator

Circuit/Plan

General

The circuit is designed for regulating the electrical current which powers high current LED's . Therefore the LED's are connected almost directly to the powersource. The Mosefets, which can switched by the micro-controller regulatate the draining from the them. Although the system operates in the spectrum of Extra-low voltage (< 120V DC) it can serve up to 0,5 A from the battery and therefore should be handeled with caution. Touching or direct skin contract with the LED Elements when full powered is not advised. (and not tested) Furthermore this is a Proto-type and should be handled like one.

Function

In our prototype we used a Node MCU, an open source IoT platform. The Hardware of this Single-board microcontroller is based on a ESP-12 module. The ESP-12 module serves as a low-cost Wi-Fi chip with full TCP/IP stack and MCU (microcontroller unit) capability. While keeping microcontroller functionalities like Pulse-width modulation, the Node MCU can also provide websockets and is therefore compatible with a lot of protocols already in use in different IOT and smart home environments like MQTT. The Prototype can work in three ways:

1. As a standalone server and access point in an Ad Hoc Network:

The Node MCU is programmed to work as the access point (for other devices to connect) in a small Ad-hoc-Network. It serves a HTML website to a fixed IP. The website provides gui elements which can be used to control the shawl. This connection structure does not rely on a pre-existing infrastructure but shows, when used, a high number of devic-disconnects and proves therefore to be to fragile for extensive user testing.

2. As a standalone server in existing Wifi Network:

The Node MCU is programmed to work as a web server serving http in the local wifi network. It serves a HTML website to a fixed IP. The website provides gui elements which can be used to control the shawl. Every Prototype has a different IP Address and a custom GUI elements tailored to its functions.

3. As a MQTT Client in in an existing Wifi Network with :

MQTT (Message Queue Telemetry Transport) is an publish-subscribe-based messaging protocol for use on top of the TCP/IP protocol. Thereby a central broker is responsible for distributing messages to interested clients based on the topic of a message. The Node MCU is programmed as a client in an existing MQTT Network: It can subscribe to a specific topic, which is managed and updated by the broker. So the Prototype can be easily integrated in bigger and complex IOT Environments.

Microcontroller

The microcontroller, a Node MCU is pro-
grammed with mircopython. After deploying
the firmware to it, the Node can be pro-
grammed via REPL over the serial port di-
rectly. REPL works like an interactive python
interpreter. Programms can also be uploaded
via the provided scipt. (webrepl_cli.py)
To have the programm executed when the
controller starts, it needs to be named:
main.py

While working with the protoype we discov-
ered that the Node MCU serves http much
more stable when integrated in a existing wifi
network. Therefore we gave it a fixed ip in
the Lab Network.

The used sourcecode can be find attached.

https://docs.micropython.org/en/latest/
esp8266/esp8266/tutorial/intro.html

http://docs.micropython.org/en/latest/
esp8266/esp8266/tutorial/intro.html#deploy-
ing-the-firmware

https://github.com/micropython/webrepl