Micropython Tutorial for esp32

Pascal Deneaux

19.01.2019

Inhaltsverzeichnis:

KAPITEL 1

Vorwort

Dies ist ein Micropython Tutorial für den esp32-Mikrocontroller.

Warnung: Dieses Tutorial befindet sich noch im Aufbau und wird zur Zeit von einer Person angelegt. Wenn du Lust hast mit zu machen, kontaktiere mich bitte über Github.

KAPITEL 2

An wen richtet sich dieses Tutorial ?

Natürlich an dich! Dieses Tutorial ist entstanden mit dem Gedanken, es im Informatikunterricht in der Mittelstufe einzusetzen. Wenn du außerhalb der Schule auf diese Seite gestoßen bist, bist du natürlich ebenfalls willkommen. Jeder Autodidakt darf die Inhalte, die hier geboten werden gerne frei verwenden. Ich hoffe das dieses Tutorial dich etwas lehrt und dich dazu inspiriert, etwas großartiges damit zu erschaffen.

Micropython ist eine wundervolle Programmiersprache, die es dir sehr leicht macht, tolle Anwendungen zu entwickeln. Ich wünsche dir viel Spaß!

KAPITEL 3

Entstehung

Zum Zeitpunkt der Entstehung dieses Tutorials fand ich im Internet keine deutschsprachigen Anleitungen zu Micropython.

Nach intensiver Beschäftigung mit dem esp8266 und dem Nachfolger esp32 bin ich zu der Entscheidung gelangt, dass endlich ein deutschsprachiges Tutorial für den Schulunterricht oder das Selbststudium her muss.

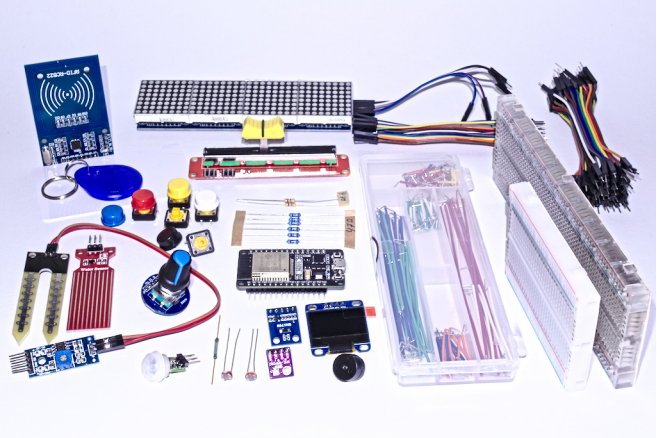
KAPITEL 4

Aufbau dieses Tutorials

KAPITEL 5

Voraussetzungen

1. Hardware
2. Software (Firmware, Treiber, Bibliotheken, Tools)
3. Dieses Tutorial



5.1 Vorbereitungen

In diesem Kapitel bekommst Schritt für Schritt gezeigt, was du tun musst, um deinen Computer fürs Programmieren vorzubereiten.

Wenn du Linux benutzt, musst du zunächst ein paar Programme installieren um dann in einem Terminal die Treiber auf den ESP32 zu kopieren. Das Windows Setup ist vergleichsweise einfach. Unter Windows nutzt du die grafische App **uPyCraft** - unter Linux arbeitest du mit der Konsolenanwendung **mpfshell**.

Wähle hier, je nachdem welches Betriebssystem du nutzt, ein Setup aus und befolge die Anleitung.

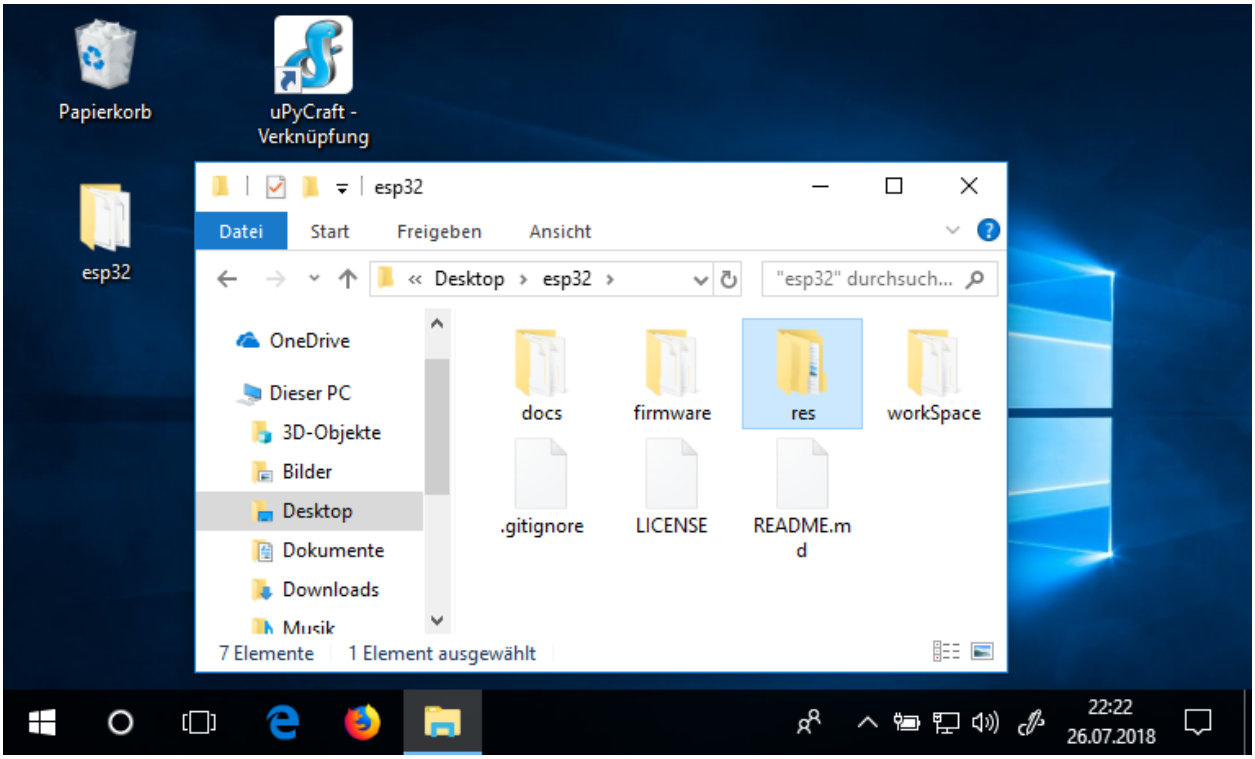
5.1.1 Software installieren

Windows Setup

Um los zu legen, lädst du dir die zip-Datei mit der App und allen Treibern herunter und öffnest sie.

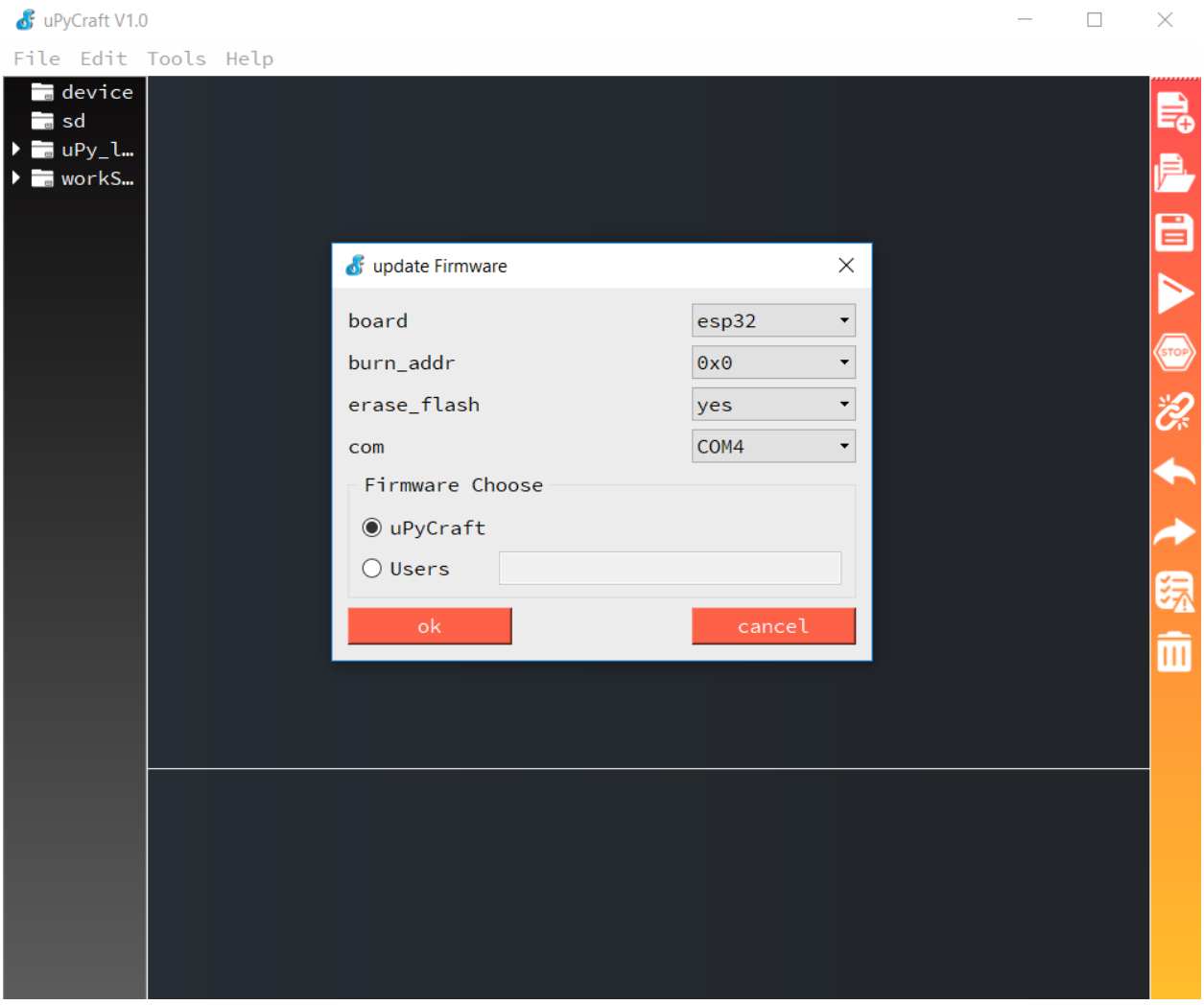
Entpacke anschließend den Ordner `Micropython-Tutorial-for-esp32-master` aus der zip-Datei auf deinen Desktop und benenne ihn in `esp32` um.

Öffne als nächstes den Ordner und gehe in das Verzeichnis `Desktop > esp32 > res > Software` und erstelle von der Datei `uPyCraft.exe` eine Verknüpfung auf deinen Desktop.

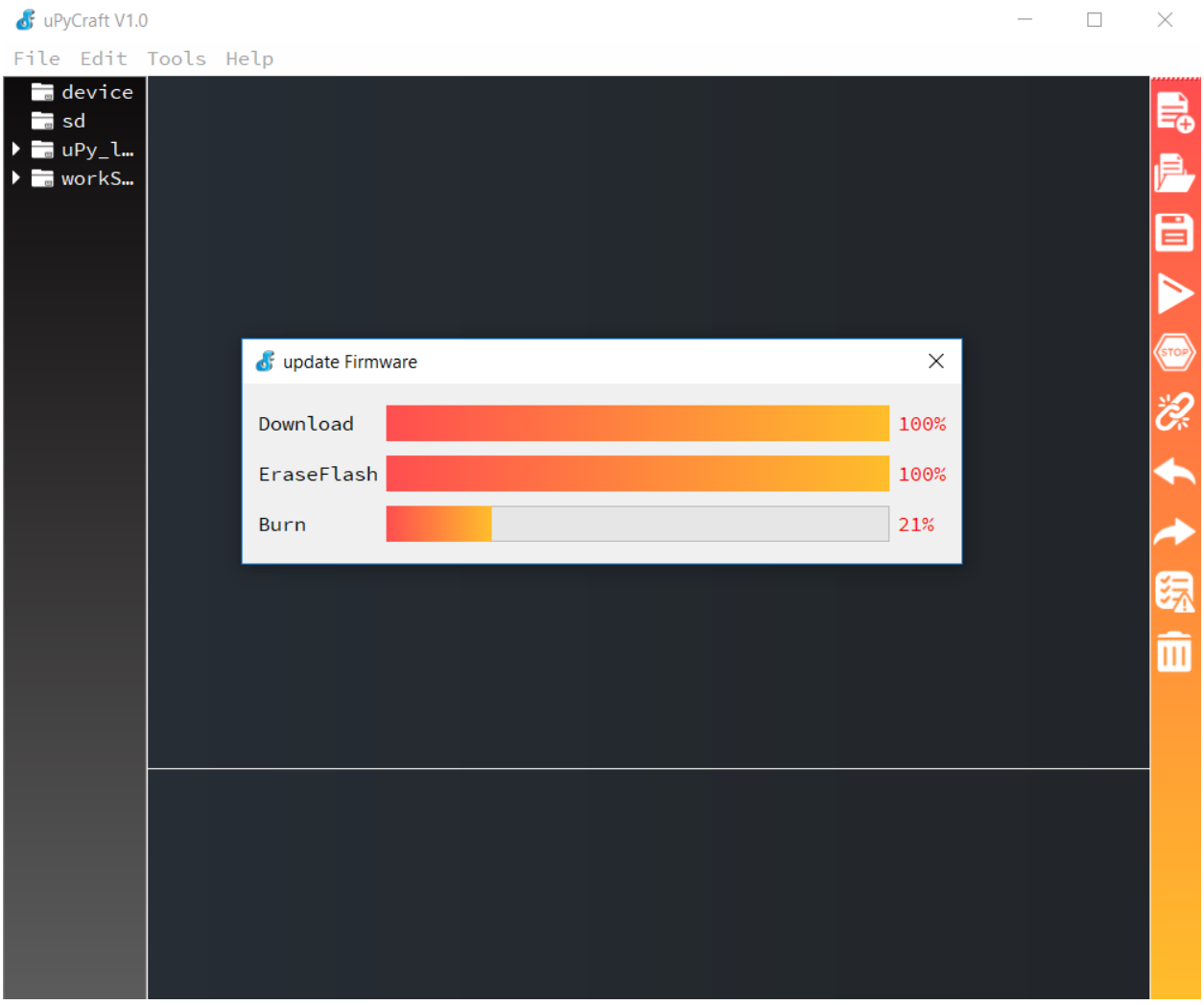


Schließe das ESP32-Board jetzt an den Computer an und installiere den Treiber aus der zip-Datei `Desktop > esp32 > res > Software > CP210x_Universal_Windows_Driver.zip` indem du zu erst die zip-Datei entpackst und anschließend das Setup `CP210xVCPInstaller_x64` ausführst.

Wenn du zum ersten mal das **uPyCraft** startest, wirst du gebeten eine Schriftart zu installieren. Im Anschluss begrüßt dich die App mit einer grafischen Benutzeroberfläche. Klicke jetzt in der Menüleiste auf **Tools -> Serial** und wähle einen COM-Port aus. Das Dialogfenster *update Firmware* öffnet sich so wie in der nächsten Abbildung.



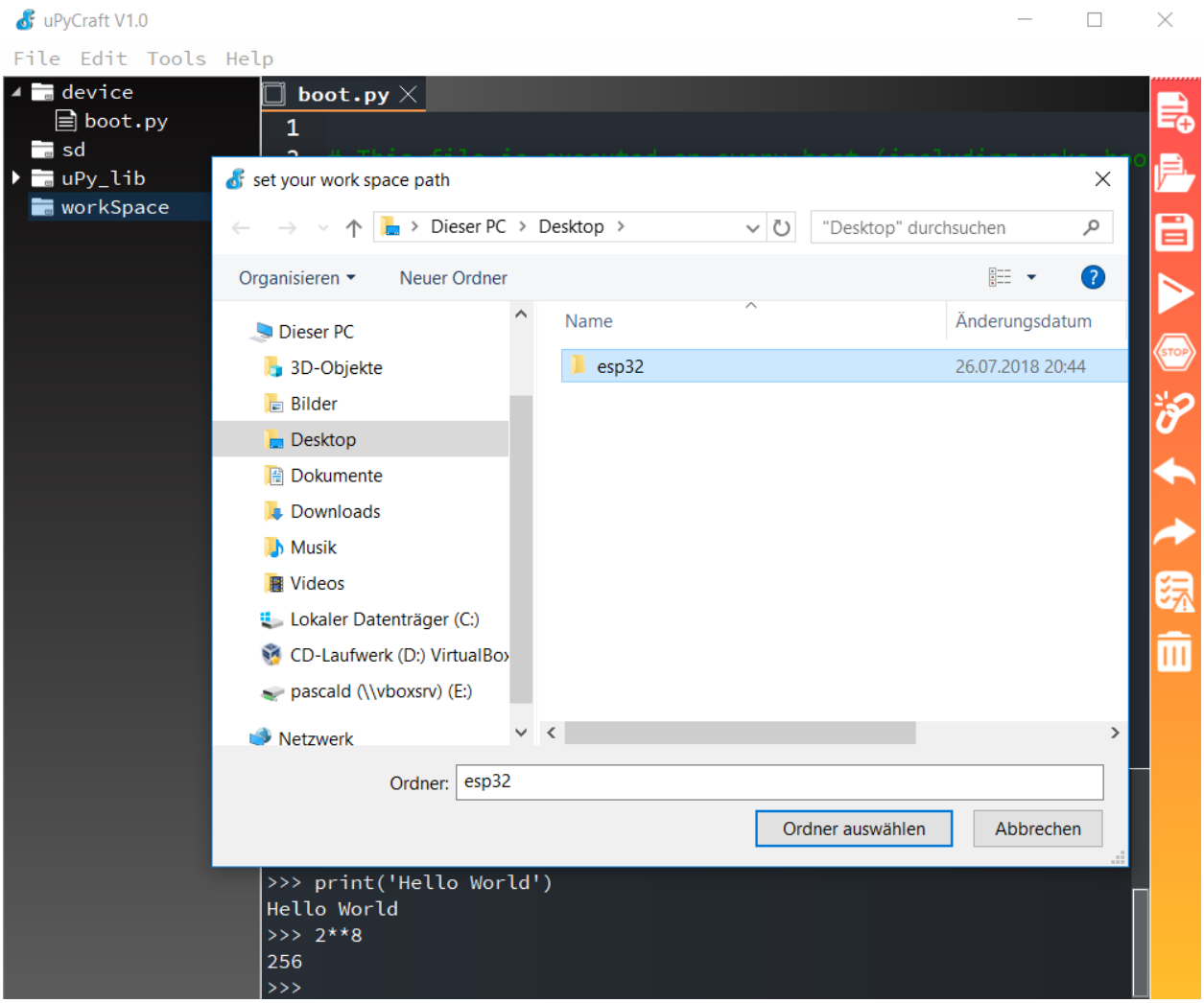
Achte darauf, dass du das richtige board (esp32) auswählst und lasse uPyCraft die Firmware selbst auswählen und aus dem Internet laden indem du auf **ok** klickst. Zunächst wird die neuste Micropython-Firmware heruntergeladen und im Anschluss wird der Speicher des ESP32 mit der neuen Firmware überschrieben. Die kleine blaue LED zeigt diesen Vorgang durch ein flackern an.



Wenn alles geklappt hat, sollte im unteren Bereich ein Python-Promt >>> auftauchen.

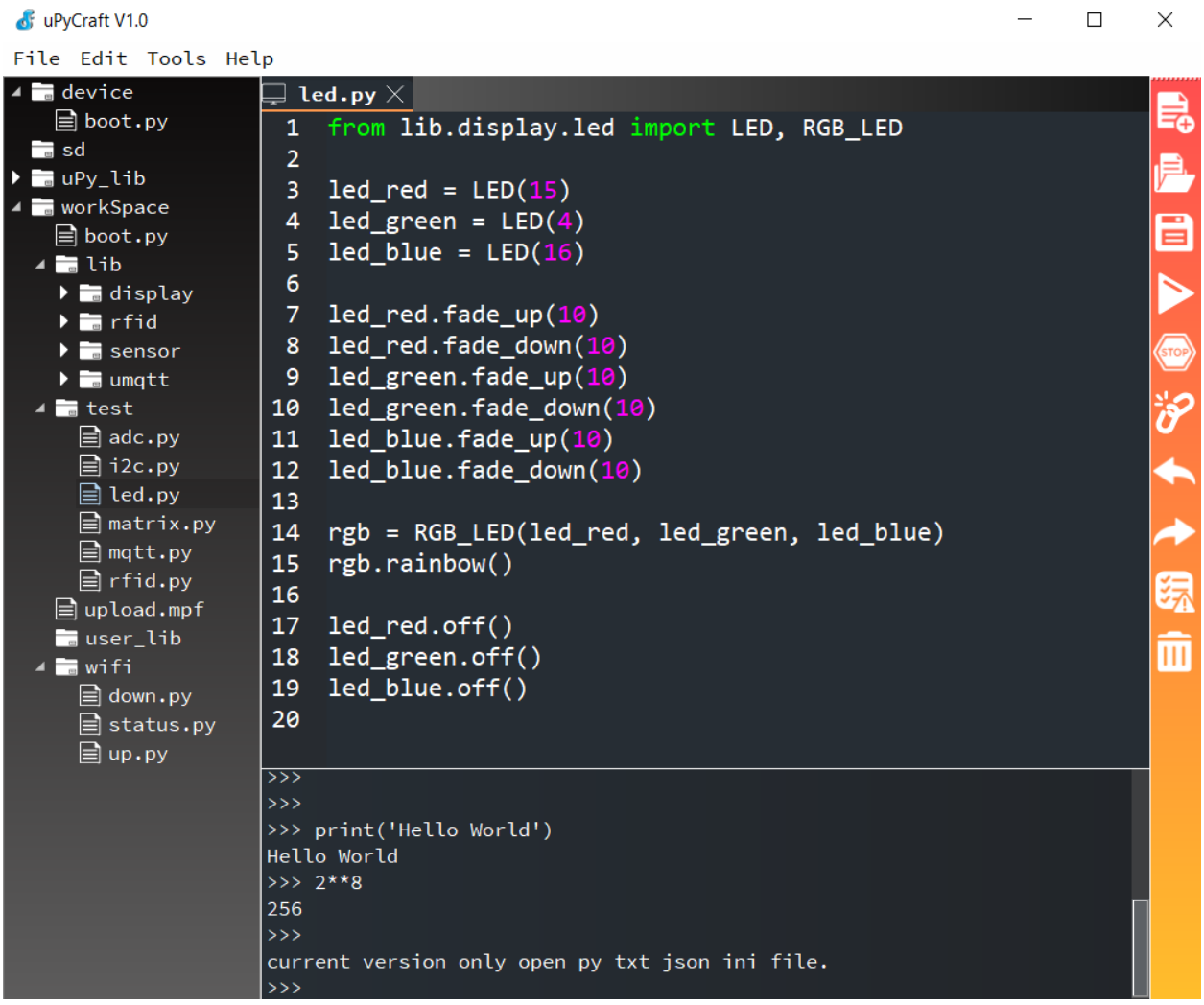
Jetzt wählst du noch den Ordner `esp32` als Arbeitsverzeichnis aus indem du links im Verzeichnisbaum auf **workSpace** klickst.

Bemerkung: Achtung! wähle **nicht** den Ordner `Desktop > esp32 > workSpace` sondern `Desktop > esp32`. Um eine falsche Wahl rückgängig zu machen, klicke in der Menüleiste auf `Tools > InitConfig`



Im nächsten Bild siehst du, wie eine Datei aus dem Arbeitsverzeichnis geöffnet wurde. Auch im Python-Terminal (REPL) wurden schon ein paar Befehle ausprobiert.

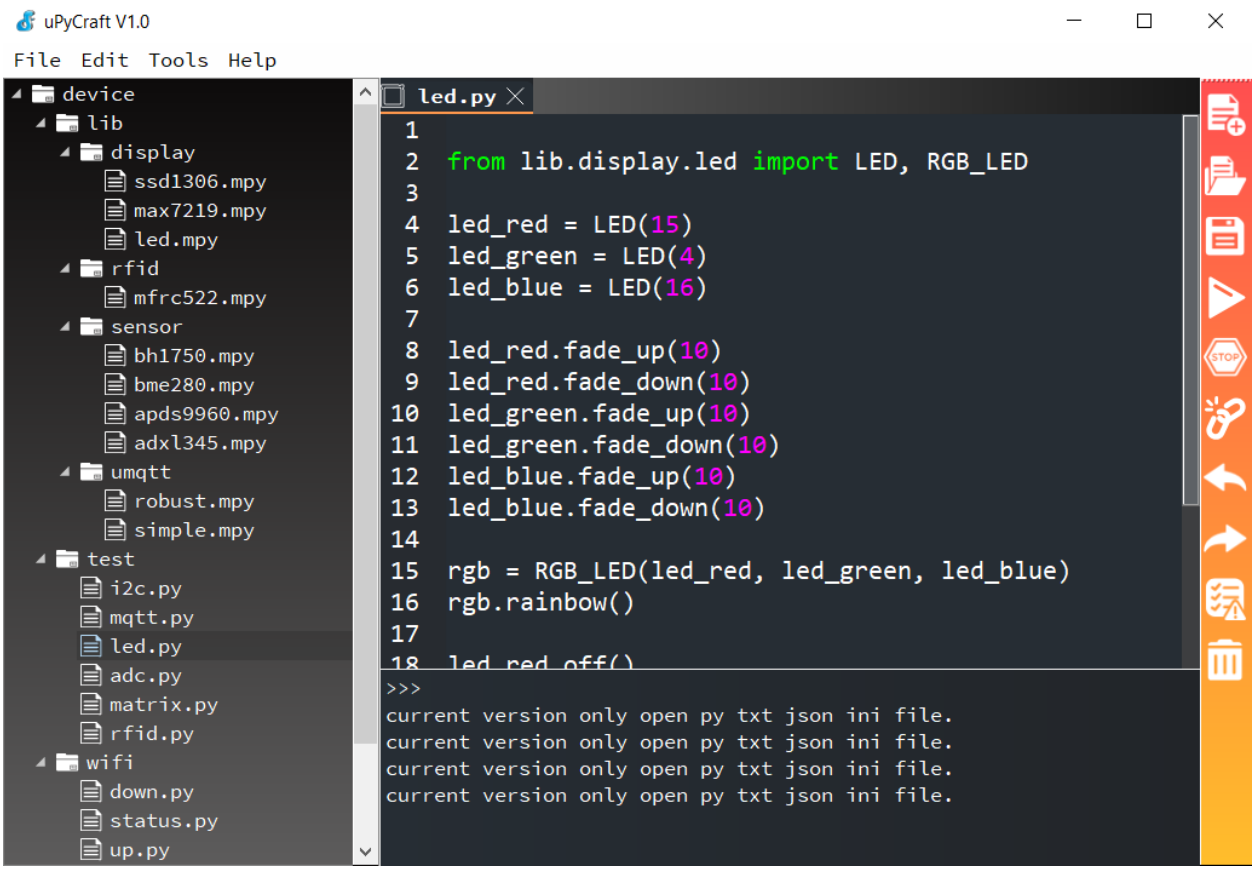
Manchmal musst du in der Menüleiste auf **File > Reflush Directory** klicken, um den Verzeichnisbaum links zu aktualisieren.



Zu guter Letzt musst du noch die drei Verzeichnisse `lib`, `test` und `wifi` auf den ESP32 kopieren. Leider geht das nicht in einem Zug. Jeder Ordner muss von Hand angelegt und jede Datei muss einzeln kopiert werden.

Warnung: Achtung! Alle `py`-Dateien müssen kopiert werden. Einzige Ausnahme sind die `py`-Dateien im Verzeichnis `lib`. Diese Dateien **nicht** kopieren. Statt dessen müssen die `mpy`-Dateien kopiert werden. Das sind die Treiber für die verschiedenen Module.

Überprüfe ob alles passt, indem du deinen Verzeichnisbaum mit dem nächsten Bild vergleichst.



Linux Setup

Software installieren

Auf einem Computer muss zunächst das Programm **esptool** installiert werden. Damit lässt sich die Firmware auf den Mikrocontroller kopieren. Man sagt zu diesem Vorgang auch *flashen*. Am einfachsten ist es, du führst die drei folgenden Befehle auf jedem Computer aus.

Bemerkung: Vergiss nicht **MeinBenutzername** durch den Benutzernamen zu ersetzt. Der Benutzername steht übrigens im Terminal vor dem @-Zeichen.

```
$ sudo apt-get install python-pip wget fritzing
$ sudo pip install mpfshell
$ sudo pip install esptool
$ sudo adduser MeinBenutzername dialout
```

Die erste Zeile installiert drei nützliche Programme auf dem Computer. Mit dem zweiten und dritten Befehl werden die beiden Programme **esptool** und **mpfshell** installiert.

Mit dem vierten Befehl wird der Benutzer der Gruppe *dialout* hinzugefügt. Dadurch erhält der Benutzer die Rechte um auf die serielle Schnittstelle des Computers zugreifen zu dürfen.

Bemerkung: Damit die Änderung des vierten Befehls wirksam wird, muss sich der Benutzer einmal ab- und wieder

anmelden.

Genauso wie ein Arduino benötigt auch ein ESP32-Mikrocontroller eine Firmware. Aber anders dieser, wird ein ESP32 nicht von Werk aus mit der Micropython-Firmware ausgestattet. Diese muss zunächst in den Speicher des Prozessors geladen (*geflasht*) werden. Erst wenn das getan ist, lassen sich Micropython-Programme darauf ausführen. In den Speicher des ESP32 muss sozusagen ein neues Betriebsystem: **Micropython** installiert werden.

Bei einem Arduino wird die Firmware jedes mal, wenn am Programm eine Änderung vorgenommen wurde, überschrieben. Beim ESP32 passiert das nur ein einziges Mal. Das Micropython-*Betriebssystem* hat nämlich ein Dateisystem, in das man mit Hilfe des Programms **mpfshell** ganz leicht Programm-Dateien kopieren oder löschen kann.

Frimware flashen

Lade dir zuerst die zip-Datei mit den Treibern herunter und entpacke sie in dein home-Verzeichnis. Öffne jetzt ein Terminal in wechsle in den home-Verzeichnis (erster Befehl). Benenne den Ordner `Micropython-Tutorial-for-esp32-master` um in `esp32` (zweiter Befehl).

```
$ cd ~
$ mv Micropython-Tutorial-for-esp32-master esp32
$ cd esp32/firmware/
```

Du befindest dich nach dem dritten Befehl im Ordner `~/esp32/firmware`.

Wenn du aktuellste Micropython-Firmware herunter laden möchtest, dann speichere sie in diesen Ordner ab.

Verbinde jetzt des ESP32-Board mit deinem Computer. Mit dem ersten Befehl `ls /dev/ttyUSB*` überprüfst du, ob das Board vom Computer erkannt wird. Wird im Terminal `/dev/ttyUSB0` oder `/dev/ttyUSB1` angezeigt, dann ist alles in Ordnung. Führe jetzt die beiden Befehle mit `esptool.py` aus, um die Firmware auf das ESP32-Board zu flashen.

Bemerkung: Achtung! Denke daran, vorher `firmware.bin` durch den Dateinamen der Firmware zu ersetzen.

```
$ ls /dev/ttyUSB*
/dev/ttyUSB0
$ esptool.py --port /dev/ttyUSB0 erase_flash
$ esptool.py --chip esp32 --port /dev/ttyUSB0 write_flash -z 0x1000 firmware.bin
```

Treiber auf das Board kopieren

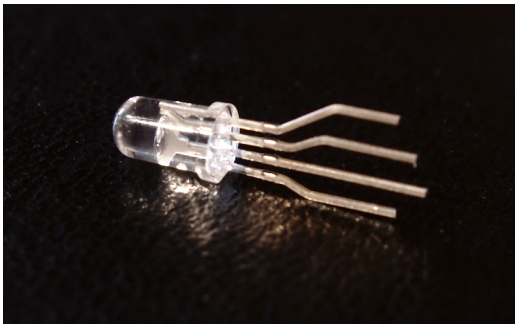
Im Verzeichnis `esp32` befinden sich drei wichtige Ordner: `lib`, `test` und `wifi`. Der Inhalt dieser Ordner muss nun auf den ESP32 kopiert werden. Dazu hast du vorhin das Programm `mpfshell` installiert. Führe den folgenden Befehl aus um alle erforderlichen Dateien aus den drei Ordnern auf das ESP32-Board zu kopieren.

```
$ mpfshell -s upload.mpf
```

Wenn alles geklappt hast, bist du von der Software-Seite her jetzt bereit um los zu legen. Nimm dir aber noch die Zeit um auch schon die Hardware vorzubereiten. Du darfst jetzt auch alle Tütchen auspacken und den Lötkolben einschalten.

5.1.2 Hardware Vorbereiten

Im Sortiment befinden sich 5 Taster und zwei RGB-LEDs. Die Beinchen der Leuchtdioden sind viel zu lang und viel zu nah beieinander. Versuche mit etwas Geschick die Beinchen so wie auf dem Bild zu biegen. Im Anschluss kannst du die Beinchen mit einem Seitenschneider auf die halbe Länge kürzen. Jetzt passt die LED super in das Steckbrett.



Warnung: Achte unbedingt darauf, dass die Länge der Beinchen vor und nach dem Kürzen das gleiche Verhältnis zueinander haben. Das längste Beinchen sollte also auch nach dem Kürzen länger als die anderen sein.

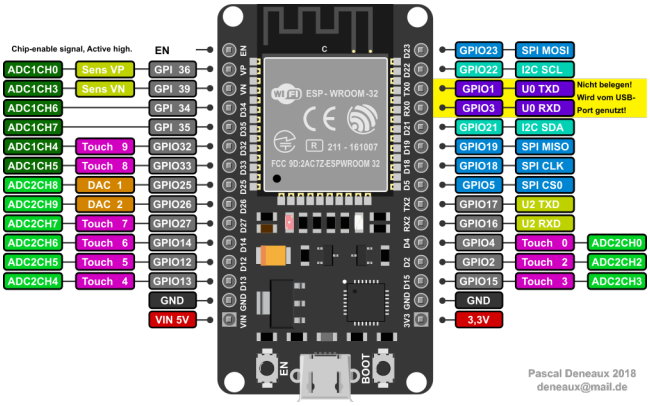
Die vier Beinchen der Taster musst du mit der Pinzette um 90° verdrehen damit sie in das Steckbrett passen. Schau dir am besten die beiden Fotos an. Dann weißt du was du machen musst. Alle Taster sollen so wie der rechts im Bild aussehen.

Schau dir auf dem nächsten Bild die Beinchen an. Sie sind genau um 90° verdreht.

Fertig? Noch nicht ganz. Einige Sensoren werden mit Stiftleisten geliefert. Diese Stifte müssen noch an die Platine gelötet werden. Lass dir helfen wenn du noch nie gelötet hast.

5.1.3 Pinout-Diagramm ausdrucken

Last but not least solltest du dir das Pinout-Diagramm ausdrucken. Klicke dazu mit der rechten Maustaste auf das Bild und wähle anschließend *Bild anzeigen* im Kontextmenü deines Browsers aus. Danach öffnet sich ein neuer Tab in deinem Browser. Drucke das Bild aus indem du auf deiner Tastatur die Tasten **Strg** und **P** gemeinsam drückst.



5.2 Erste Schritte

Gleich kann es los gehen. Im ersten Kapitel wirst du eine RGB-LED zum blinken bringen. Das tolle an einer RGB-LED ist, dass sie in allen Farben leuchten kann. Jede Farbe kann durch das Mischen der drei Grundfarben Rot, Grün und Blau erzeugt werden.

Aber bevor es los geht, gibt es noch paar Dinge die du wissen solltest.

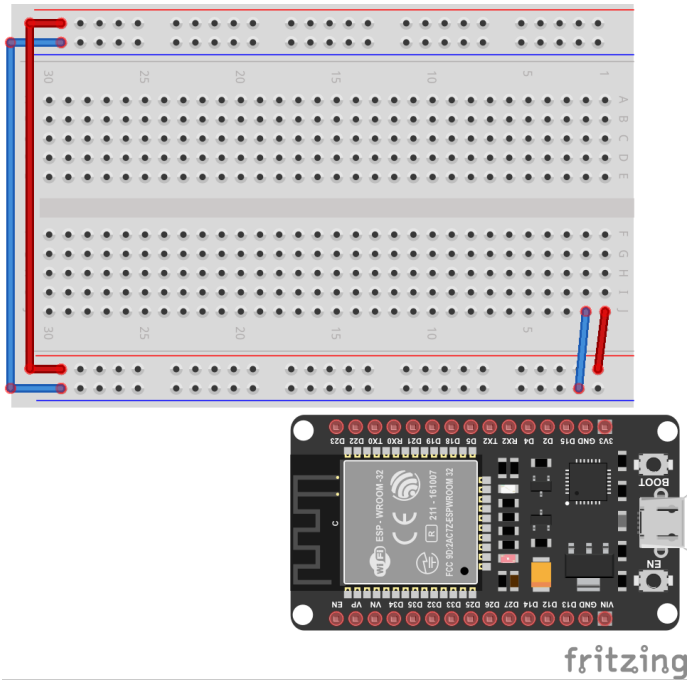
5.2.1 Das Steckbrett

Wie du sicher weißt, fließt der Strom immer vom Plus-Pol zum Minus-Pol. Plus heißt, dass das Potential hier bei 3,3 Volt liegt und Minus steht für 0 Volt. Auf deinem Steckbrett stehen die roten Linien für Plus (3,3V) und die beiden blauen Linien für Minus (0V). Die Farbe Schwarz deutet auch manchmal auf den Minus-Anschluss hin.

Bemerkung: Wenn du einen Blick auf den ESP32 wirfst, wirst du erkennen, dass zwei Pins mit dem Aufdruck `GND` beschriftet sind. `GND` steht für *Ground* und das heißt im Grunde nichts anderes als Minus.

Grundaufbau

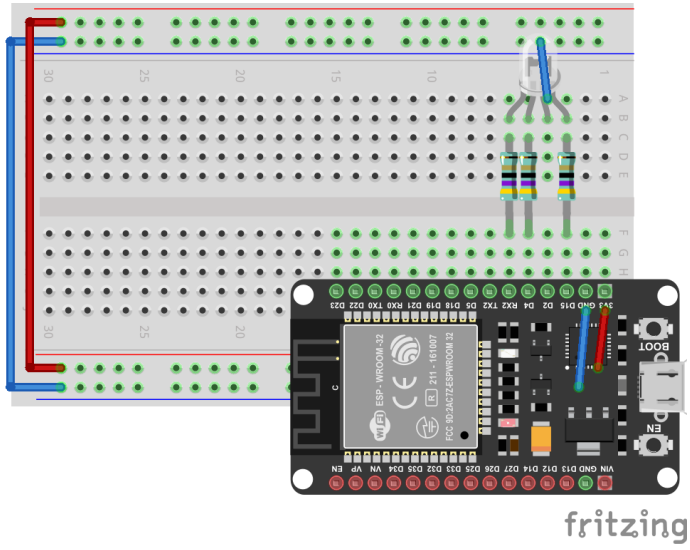
Baue als erstes die Schaltung genau wie auf dem Bild nach. Achte ganz penibel darauf, dass alle Kabel genau dort sind, wo sie hin gehören. Lass dich von den Farben auf dem Bild nicht irritieren. Du wirst in deinem Steckbrücken-Set selten die gewünschte Farbe in der richtigen Länge finden. Wähle die Steckbrücke einfach nach der Länge aus.



Die beiden Steckbrücken rechts unten verbinden den Plus- und Minus-Pol des ESP32 mit dem Steckbrett. 3,3V wird mit der roten Versorgungsleitung und GND mit der blauen Versorgungsleitung verbunden.

Die beiden Kabel links verbinden den Plus- (rot) und Minus-Pol (blau) der oberen und unteren Hälfte des Steckbretts miteinander.

Wenn du als nächstes das ESP32-Board einsetzt, wirst du mit ein wenig Druck und Feinfühligkeit vorgehen müssen um weder das Board noch das Steckbrett zu beschädigen. Bei manchen Steckbrettern ist das gar nicht so einfach. Wenn du hier nicht vorsichtig bist, kann es passieren, dass du später Probleme bekommst. Handle also unbedingt mit Gefühl!



Auf dem zweiten Bild kannst du sehen, wie du die RGB-LED und die drei Widerstände mit dem ESP32 verbindest.

Warnung: Ließ dir unbedingt zu erst den nächsten Abschnitt durch bevor du die RGB-LED falsch herum einsetzt! Die Länge der vier Anschlusspins spielt nämlich eine wichtige Rolle.

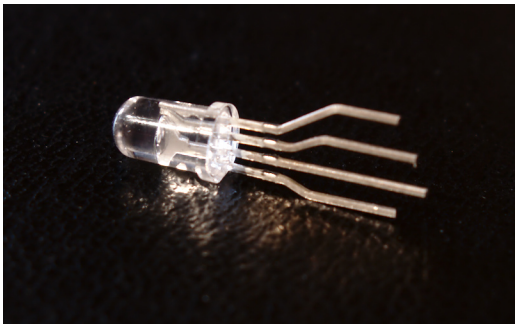
5.2.2 Wissenswertes über LEDs

Es zwei wichtige Dinge, die man über LEDs wissen sollte. Normale LEDs haben zwei Anschlüsse. Eine Kathode (Minus) und eine Anode (Plus). RGB-LEDs bestehen im Prinzip aus drei normalen LEDs - eine rote, eine blaue und eine grüne. Müsste eine RGB-LED dann nicht 6 Anschlüsse, für drei Anoden und drei Kathoden, haben?

Die Antwort ist ganz einfach: Nein, denn innerhalb der RGB-LED werden die drei Kathoden (Minuspole) der drei LEDs miteinander verbunden. Bei der RGB-LED im Set handelt es sich um ein Bauteil vom Typ *common cathode* also gemeinsame Kathode. Die Minuspole der drei LEDs haben also einen gemeinsamen Anschluss. **Man erkennt ihn am langen Pin!**

Dieser lange Anschlusspin sollte also immer mit Minus (blau) verbunden sein.

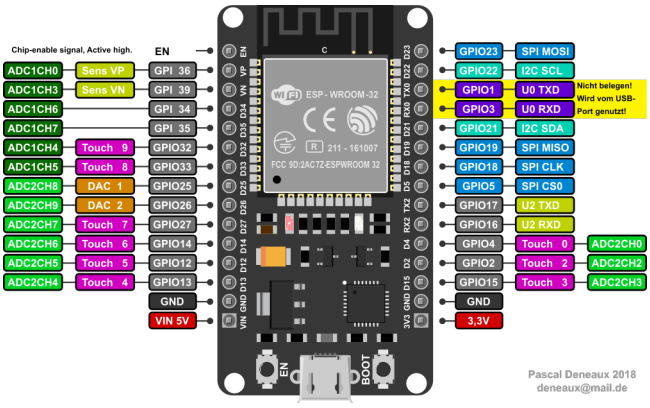
Es gibt auch RGB-LEDs vom Typ *common anode* zu kaufen. Sie haben logischerweise einen gemeinsamen Pluspol.



Eine zweite Sache sollte man auch immer im Hinterkopf behalten. LEDs dürfen niemals ohne einen Vorwiderstand in einen Stromkreis eingebaut werden. Die 47Ohm Widerstände sind auf die 3,3V des ESP32 abgestimmt. Jede LED bekommt einen eigenen Widerstand. Am besten kürzt du die Beinchen der Widerstände mit einem Seitenschneider auf die halbe Länge.

5.2.3 Das Pinout-Diagramm

Das Schaubild zeigt dir welche Funktion(en) die 30 Pins deines ESP32-Boards haben. Nimm dir einen Moment Zeit und mache dir klar welche fünf Pins du aktuell in deiner Schaltung benutzt.



5.2.4 Dein erstes Programm

Hier ist ein kleines Python-Programm, das die LED dreimal rot blinken lässt.

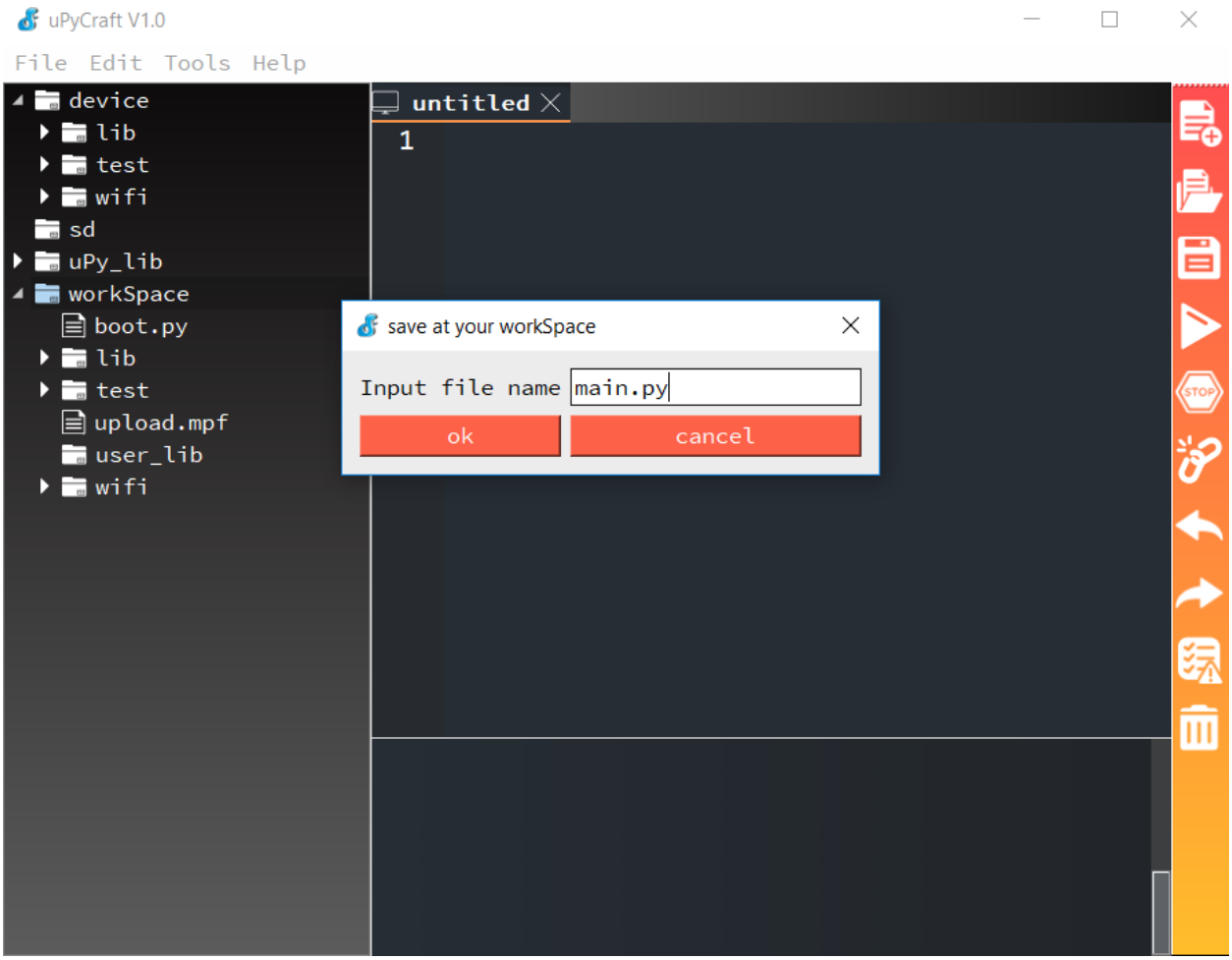
```
from lib.display.led import LED
from time import sleep_ms

led_red = LED(15)
led_green = LED(4)
led_blue = LED(16)

for x in range(3):
    led_red.on()
    sleep_ms(500)
    led_red.off()
    sleep_ms(500)
```

Wie du das Programm auf den ESP32 kopierst, hängt von deinem Betriebssystem ab. Wähle Windows oder Linux.

Windows (uPyCraft)

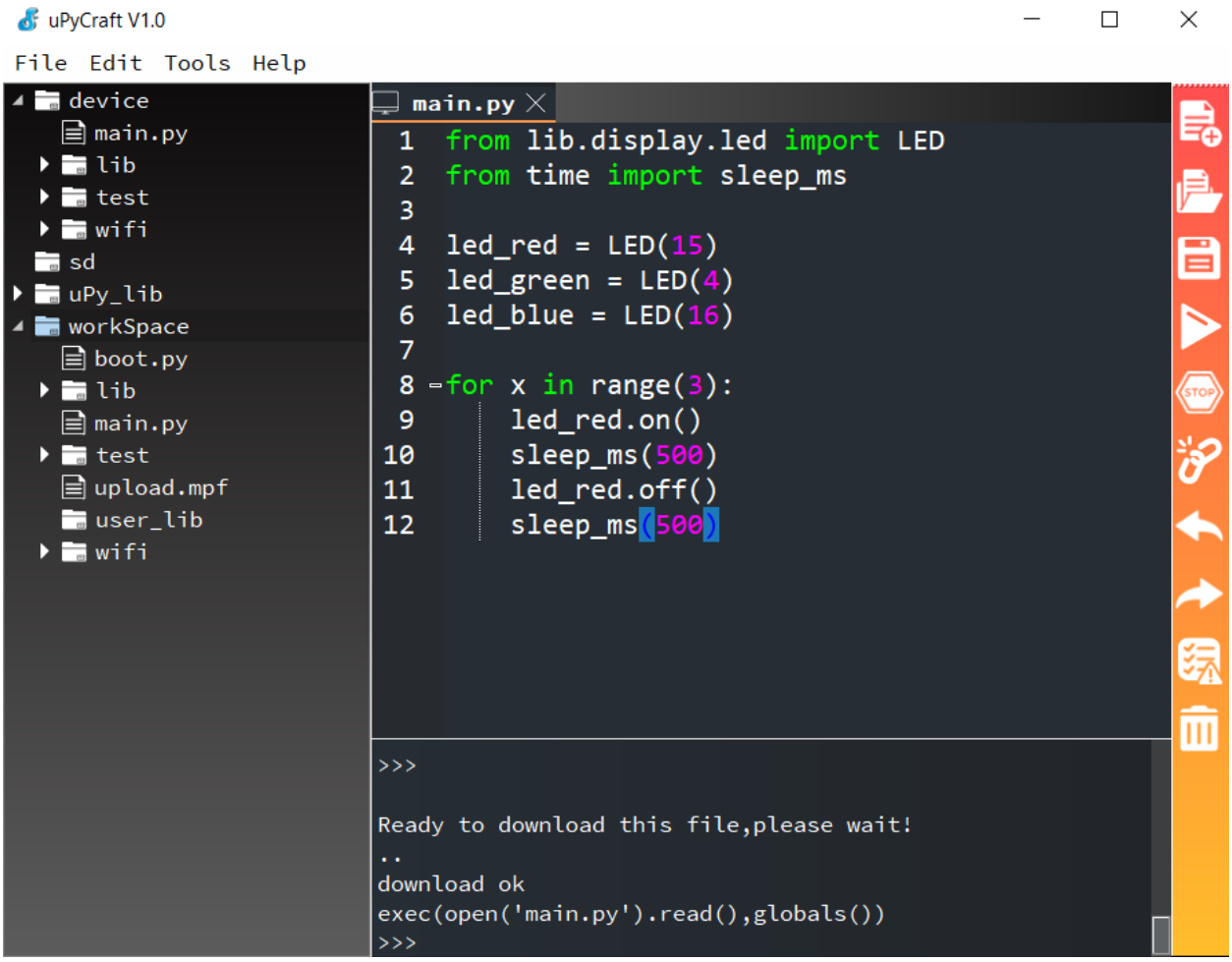
Erstelle zu erst eine neue Datei indem du in der Symbolleiste rechts das oberste Symbol wählst.

Speichere die Datei anschließend in deinem *WorkSpace* indem du auf das Diskettensymbol rechts klickst und den Dateinamen `main.py` eingibst.



Im letzten Schritt klickst du auf das *Play*-Symbol (Nr. 4 von oben) um die Datei in den Speicher des ESP32 zu kopieren

und auszuführen.



So einfach geht das. Bedenke, dass du jede Datei, die du bereits auf den ESP32 kopiert hast auch einfach durch einen Rechtsklick auf die Datei im Verzeichnisbaum links starten kannst.

Linux (mpfshell)

Speicher das Programm in einer neuen Datei mit dem Dateinamen `main.py` im Verzeichniss `~/esp32 ab` (~ steht für dein home-Verzeichnis).

Öffne im Verzeichnis `esp32` ein Terminal und startest dort das Programm `mpfshell`.

Das Bild zeigt dir, was alles schief gehen kann. Der erste Befehl `ls` kann nicht ausgeführt werden, da noch keine Verbindung zum Board hergestellt wurde. Im zweiten Anlauf wurde vergessen das Board an den Computer anzuschließen und dann teilt der Computer im dritten Anlauf dem Board den Namen `ttyUSB1` zu. Das passiert aber eher selten. Schlussendlich steht die Verbindung und nachdem Kommando `repl` begrüßt dich der typische Python-Promt `>>>`.

```
[user@host esp32]$ ls
boot.py  docs  firmware  lib  LICENSE  main.py  README.md  test  upload.mpf  wifi
[user@host esp32]$ mpfshell

** Micropython File Shell v0.8.1, sw@kaltpost.de **
-- Running on Python 3.6 using PySerial 3.4 --

mpfs [/]> ls

Not connected to device. Use 'open' first.

mpfs [/]> open ttyUSB0

Failed to open: ser:/dev/ttyUSB0

mpfs [/]> open ttyUSB1
Connected to esp32
mpfs [/]> repl
>
*** Exit REPL with Ctrl+] ***

MicroPython v1.9.3-620-geb88803a on 2018-05-10; ESP32 module with ESP32
Type "help()" for more information.
>>> print("Hallo Welt")
Hallo Welt
>>>
```

Um die Datei `main.py` kopieren zu können, musst du die REPL (*Read–Eval–Print Loop*) wieder verlassen. Drücke dazu Die Taste `9 ]` während du die Tasten `Strg` und `Alt Gr` gedrückt hältst. Auf einer englischsprachigen Tastatur macht diese Tastenkombination mehr Sinn und ist einfacher zu finden. Aber damit müssen wir uns abfinden.

Im nächsten Bild siehst du, wie du dir den Verzeichnisinhalt des ESP32 und des lokalen Ordners anzeigen lassen kannst. Erkennst du den Unterschied? Versuche zu verstehen, was genau `ls` und was `lls` macht.

Wie du sicher schon bemerkt hast, hat sich der Ordnerinhalt auf dem ESP32 nach dem **put**-Befehl verändert. Die Datei ist kopiert worden. Um sie auszuführen muss der ESP32 jetzt neu gestartet werden.

```
mpfs [/]> ls

Remote files in '/':

 <dir> lib
 <dir> test
 <dir> wifi

mpfs [/]> lls

Local files:

 <dir> docs
 <dir> firmware
 <dir> lib
 <dir> test
 <dir> wifi
       .gitignore
       LICENSE
       README.md
       boot.py
       main.py
       upload.mpf

mpfs [/]> put main.py
mpfs [/]> ls

Remote files in '/':

 <dir> lib
 <dir> test
 <dir> wifi
       main.py

mpfs [/]>
```

Gib erneut `repl` ein und drücke `Strg` und `D` um den ESP32 neu zu starten. Die Datei `main.py` wird immer automatisch gestartet wenn der ESP32 neu bootet.

Wenn alles geklappt hat, müsste die LED jetzt blinken. Gratulation!

5.2.5 Aufgaben

- Betrachte die Animation am Anfang der Seite. Um welchen Typ von RGB-LED handelt es sich. *common cathode* oder *common anode*?
- Verändere das Programm so, dassder Pin 2 also `LED(2)` angesprochen wird. Was passiert?
- Lass abwechselnd die rote, die blaue und die grüne LED leuchten.
- Ändere die Blinkgeschwindigkeit.
- Lass die LED abwechselnd in den 6 Farben: Rot, Violett, Blau, Cyan, Grün und Gelb leuchten.

5.3 Ausgabe

Jedes datenverarbeitende System - wie ein Mensch oder ein Computer - gliedert sich in die drei Teile: Eingabe, Verarbeitung und Ausgabe. Eine Möglichkeit etwas auszugeben haben wir im letzten Kapitel kennen gelernt. In diesem Kapitel gehen wir etwas genauer auf die Klasse LED ein und lernen im Anschluss, wie man ein Display und einen Summer an den ESP32 anschließt.

Bemerkung: Da sich dieses Tutorial noch im Aufbau befindet, findest du hier nur ein paar Beispielprogramme. Experimentiere mit den Programmen, um herauszufinden, wie du die Befehle einsetzten kannst.

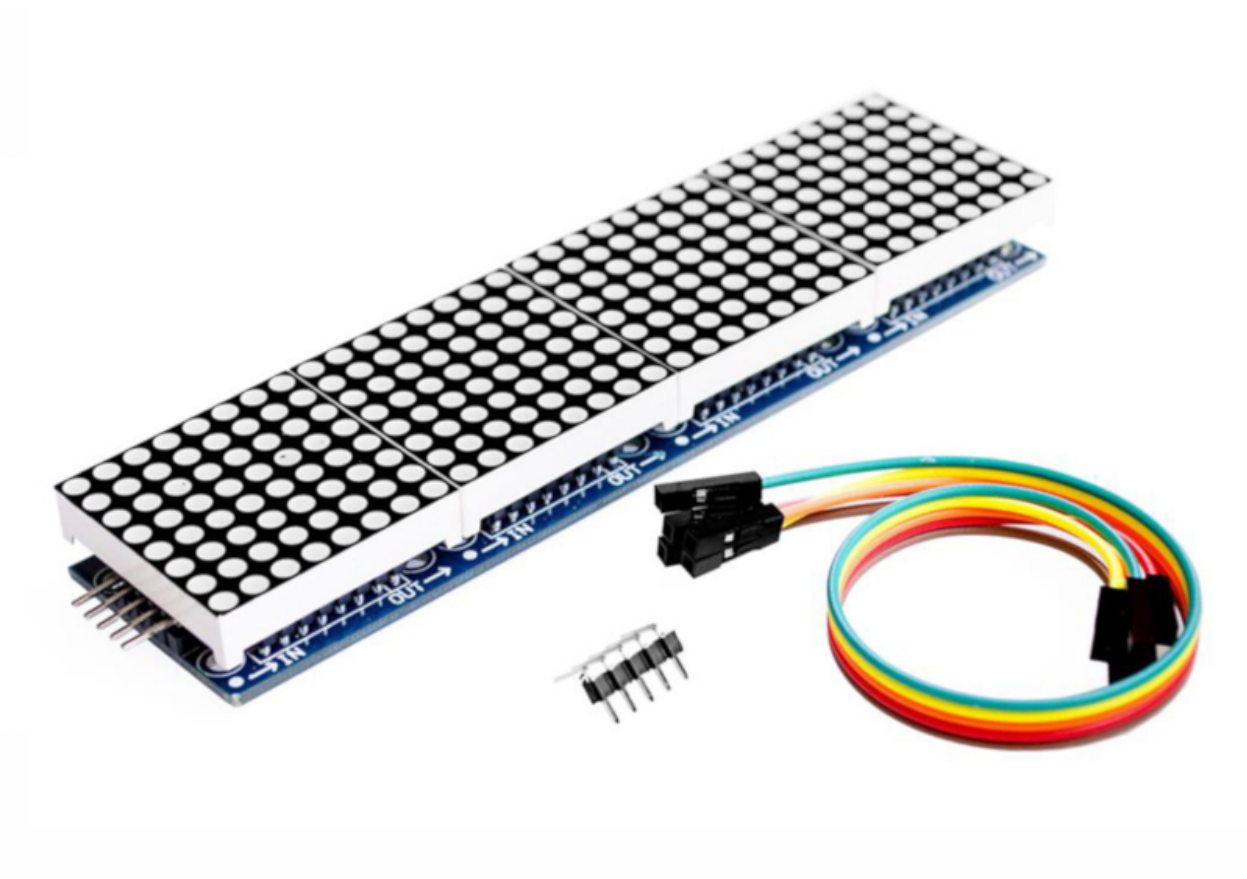
5.3.1 Spaß mit LEDs

Die Datei `test/led.py` demonstriert die Möglichkeiten der Klassen `LED` und `RGB_LED`

```
from lib.display.led import LED, RGB_LED

led_red = LED(15)
led_green = LED(4)
led_blue = LED(16)

led_red.fade_up(10)
led_red.fade_down(10)
led_green.fade_up(10)
led_green.fade_down(10)
led_blue.fade_up(10)
led_blue.fade_down(10)

rgb = RGB_LED(led_red, led_green, led_blue)
rgb.rainbow()

led_red.off()
led_green.off()
led_blue.off()
```

```
import test.led
```

5.3.2 Das LED-Matrix Display



Verbinde **VCC** mit der roten Versorgungsspannung auf den Steckbrett und **GND** mit der blauen. Verbinde **DIN** mit **GPIO23 (SPI MOSI)**, **CS** mit **GPIO2** und **CLK** mit **GPIO18 (SPI CLK)**.

```
from machine import Pin, SPI
from lib.display.max7219 import Matrix8x8
from time import sleep_ms

# Initialisiere den SPI-Bus
sck = Pin(18, Pin.OUT)
mosi = Pin(23, Pin.OUT)
miso = Pin(19, Pin.IN)
cs = Pin(2, Pin.OUT)
spi = SPI(baudrate=100000, polarity=0, phase=0, sck=sck, mosi=mosi, miso=miso)

display = Matrix8x8(spi, cs, 4)
display.brightness(0)

display.fill(1)
display.text('Halo',0,0,0)
display.show()
sleep_ms(2000)

display.fill(0)
display.text('Welt',0,0,1)
display.show()
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
sleep_ms(2000)

display.fill(0)
display.rect(0,0,32,8,1)
display.show()
```

```
import test.matrix
```

Quellen

- https://github.com/mcauser/micropython-max7219

5.3.3 Das OLED-Display

```
from machine import I2C, Pin
from lib.display.ssd1306 import SSD1306_I2C
from time import sleep_ms

# Initialisiere den I2C-Bus
bus = I2C(scl=Pin(22), sda=Pin(21))

oled = SSD1306_I2C(128, 64, bus)

oled.fill(0)
oled.text("Hallo Welt", 10, 20)
oled.show()
sleep_ms(2000)

oled.fill(1)
oled.text("Hallo Welt", 10, 20, 0)
oled.show()
```

Quellen

- https://github.com/adafruit/micropython-adafruit-ssd1306
- https://github.com/peterhinch/micropython-samples/tree/master/SSD1306

5.3.4 Töne Erzeugen mit einem Summer

5.4 Eingabe Teil 1

Bemerkung: Da sich dieses Tutorial noch im Aufbau befindet, findest du hier nur ein paar Beispielprogramme. Experimentiere mit den Programmen, um herauszufinden, wie du die Befehle einsetzten kannst.

5.4.1 Analoge Eingabe (ADC)

Regensensor

Schiebewiderstand

Für dieses Modul benötigst du die beiden 470 Ohm Widerstände (gelb - lila - braun - gold). Verbinde den **VCC**-Pin über einen 470 Ohm Widerstand mit Plus (rot), den **GND**-Pin über einen 470 Ohm Widerstand mit Minus (blau) und den **OTA**-Pin mit **GPIO 36 (VP)**.

```
from machine import Pin, ADC
from time import sleep_ms
from lib.display.led import LED, RGB_LED

led = LED(2)

adc = ADC(Pin(36))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_9BIT)
try:
    while True:
        value = adc.read() // 2
        led.brightness(value)
        print(value)
        sleep_ms(50)
except KeyboardInterrupt:
    led.off()
    print("Bye")
```

```
import test.adc
```

Fotowiderstand (LDR)

Touch

5.4.2 Digitale Eingabe

Taster

Bodenfeuchtesensor

Bewegungssensor

Inkrementalgeber

5.5 Eingabe Teil 2

5.5.1 Umweltsensoren

Die beiden Umweltsensoren bme280 und bh1750 ... Am besten schließt du vorher das oled-Display an.

```
import test.i2c
```

Quellen

- https://github.com/kevbu/micropython-bme280
- https://github.com/PinkInk/upylib/tree/master/bh1750

5.5.2 Beschleunigungssensor

Quellen

- http://andidinata.com/download/

5.5.3 RFID

Verbinde **3.3V** mit der roten Versorgungsspannung auf den Steckbrett und **GND** mit der blauen. Verbinde **MOSI** mit **GPIO23 (SPI MOSI)**, **MISO** mit **GPIO19 (SPI MISO)**, **SCK** mit **GPIO18 (SPI CLK)** und **SDA** mit **GPIO5 (SPI CS0)**. Die Pins **IRQ** und **RST** brauchst du nicht anzuschließen.

```
from machine import Pin, SPI
from lib.rfid.mfrc522 import MFRC522
from time import sleep_ms

# Initialisiere den SPI-Bus
sck = Pin(18, Pin.OUT)
mosi = Pin(23, Pin.OUT)
miso = Pin(19, Pin.OUT)
sda = Pin(5, Pin.OUT)
spi = SPI(baudrate=100000, polarity=0, phase=0, sck=sck, mosi=mosi, miso=miso)

try:
    while True:
        rdr = MFRC522(spi, sda)
        uid = ""
        (stat, tag_type) = rdr.request(rdr.REQIDL)
        if stat == rdr.OK:
            (stat, raw_uid) = rdr.anticoll()
            if stat == rdr.OK:
                uid = ("0x%02x%02x%02x%02x%02x" % (raw_uid[0], raw_uid[1], raw_uid[2],␣
↪raw_uid[3]))
                print(uid)
                sleep_ms(100)
except KeyboardInterrupt:
    print("Bye")
```

```
import test.rfid
```

Quellen

- https://github.com/wendlers/micropython-mfrc522

5.6 Internet

5.6.1 Verbindung mit dem WLAN herstellen

5.6.2 Kommunikationsprotokolle im Internet

HTTP

MQTT

5.6.3 IOT-Plattformen

Adafruit IO

IFTTT

5.7 Kaufberatung

Tab. 1: Teile

Was	Wo	Wie viel
ESP32	AliExpress	4,57 €
Steckplatine groß	AliExpress	1,95 €
Steckplatine klein	AliExpress	1,10 €
Jumper für Steckplatine	AliExpress	1,79 €
Dupont Kabel m zu m 40 Stk	AliExpress	0,57 €
Dupont Kabel m zu w 40 Stk	AliExpress	0,56 €
Punkt-Matrix LED-Modul 8x32	AliExpress	3,31 €
OLED-Display 128x64	AliExpress	1,91 €
Drehimpulsgeber	AliExpress	2,02 €
RFID Modul	AliExpress	1,13 €
Gyrosensor ADXL-345	AliExpress	0,81 €
Kombisensor Luftfeuchte Baro Temp	AliExpress	2,75 €
Helligkeitssensor	AliExpress	1,02 €
PIR Bewegungssensor	AliExpress	1,75 €
Reedschalter	AliExpress	0,23 €
Bodenfeuchtesensor	AliExpress	0,47 €
Wasser-Sensor	AliExpress	0,21 €
5 Drucktaster	AliExpress	0,73 €
2 LDR Photodiode	AliExpress	0,04 €
Pinzette	AliExpress	0,86 €
2 RGB-LED gemeinsame Anode	AliExpress	0,10 €
Wisent Organizer 3 Plus	Bauhaus	7,95 €
Magnet 10x2mm	Amazon	0,20 €
Diverse Widerstände und Kondensatoren		

5.8 MIT License