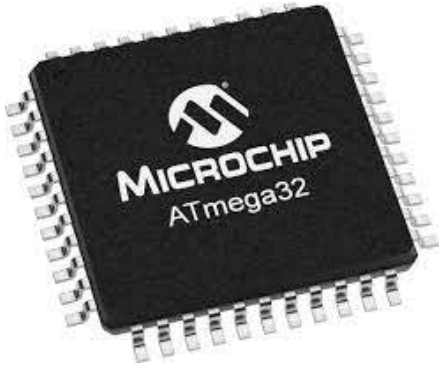
Carl von Ossietzky
Universität
Oldenburg

Soft Skills und Technische Kompetenz

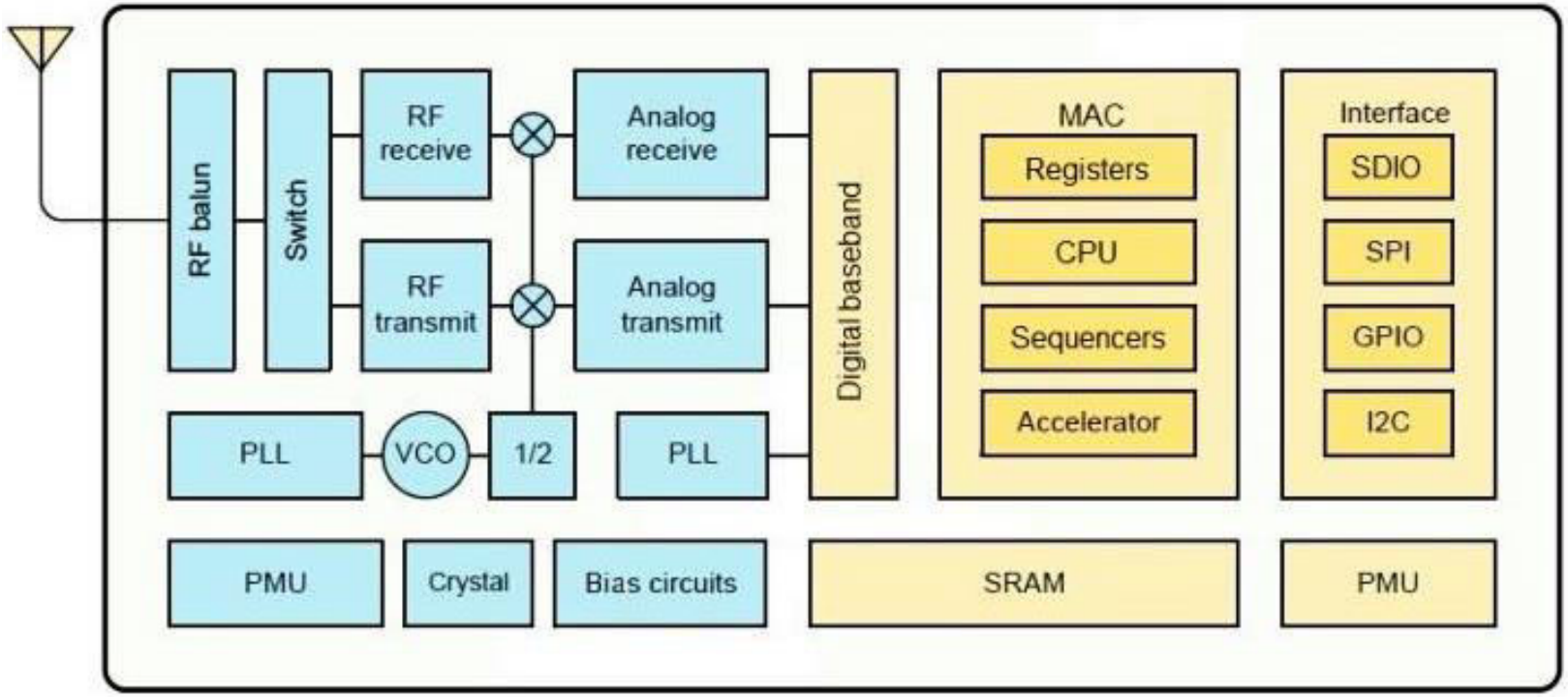
Übung
Einführung Mikrocontroller Programmierung

Anatolij Fandrich, 01.12.2020

Was ist ein Mikrocontroller?

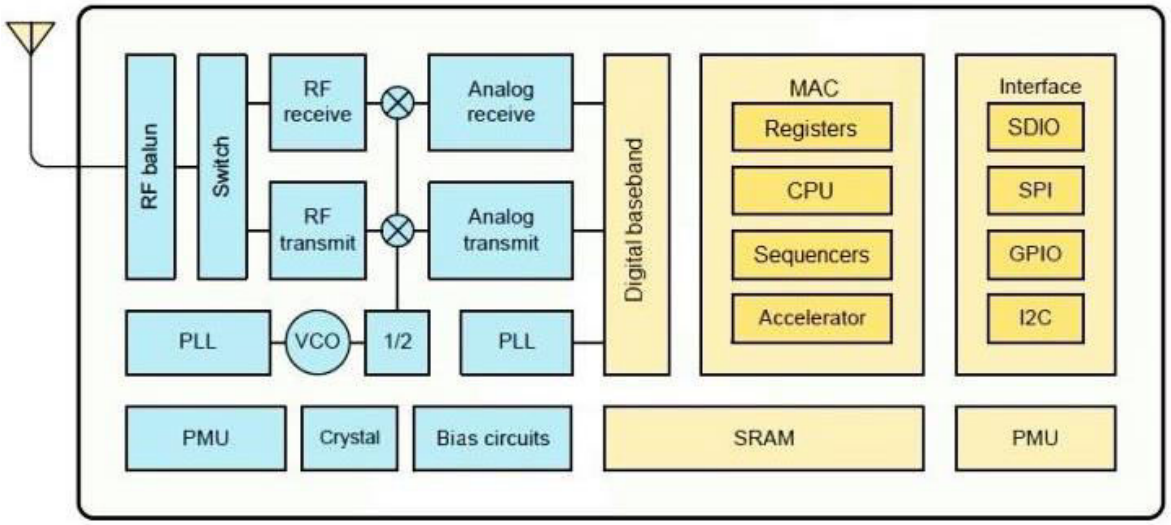


Aber stimmt das denn?



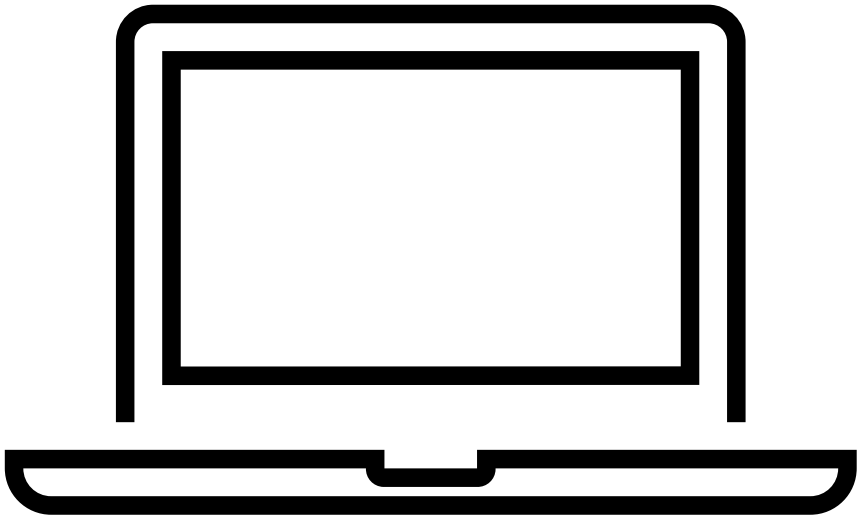
Aber stimmt das denn?

- CPU – wie ein Computer
  - Holt Befehle und Daten aus Register
  - ALU rechnet
- Speichert Daten im flüchtigen Speicher zwischen
  - z.B. Variablen
- Schnittstellen zur analogen Welt
- Programmierbar!

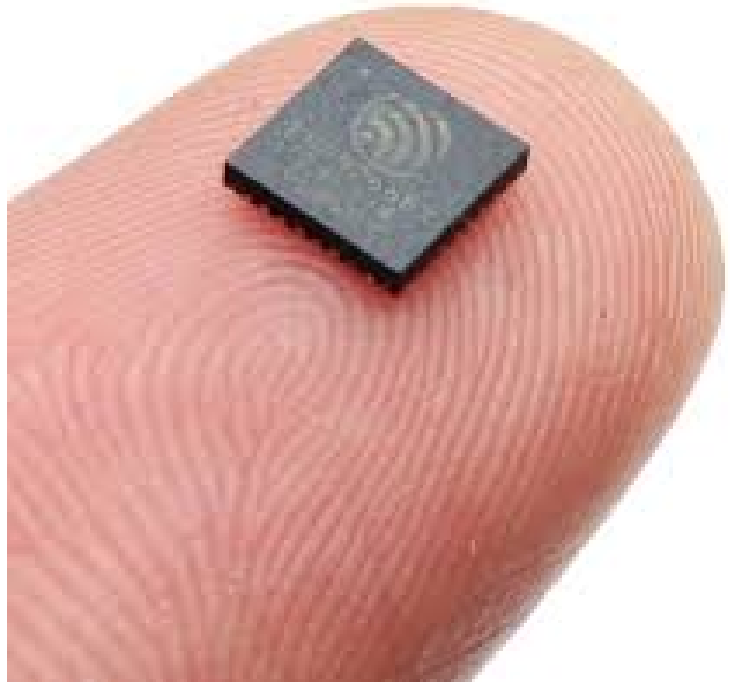


https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

Aber stimmt das denn?



?



Ja! Das stimmt, aber es sind sehr kleine Computer

- Benötigen einen „normalen“ Computer um programmiert zu werden
- Programmspeicher stark limitiert
  - Manche µC haben nur 1KB Speicher (Attiny 15)
  - Firmware (Software auf Mikrocontroller) muss der Hardware angepasst sein
  - Keine Ressourcen für ein Java Enterprise Banking System
- Arbeitsspeicher limitiert
  - 80 KiB Nutzerdaten RAM beim ESP8266
  - Nicht mehr Speicher als nötig reservieren!

Ja! Das stimmt, aber es sind sehr kleine Computer

- Systemtakt im MHz Bereich
  - 1Hz ist ein Takt pro Sekunde
  - 1 bis 20 MHz sind typisch für AVR Mikrocontroller
  - 80 bis 160MHz beim ESP8266
  - Mein erster PC hatte einen Pentium II Prozessor mit 266MHz
- Auch physisch klein
  - Keine VGA oder HDMI Ports für Bildschirme, keine USB Schnittstellen für Maus und Tastatur
  - Keine Soundkarte

Mikrocontroller im Alltag?

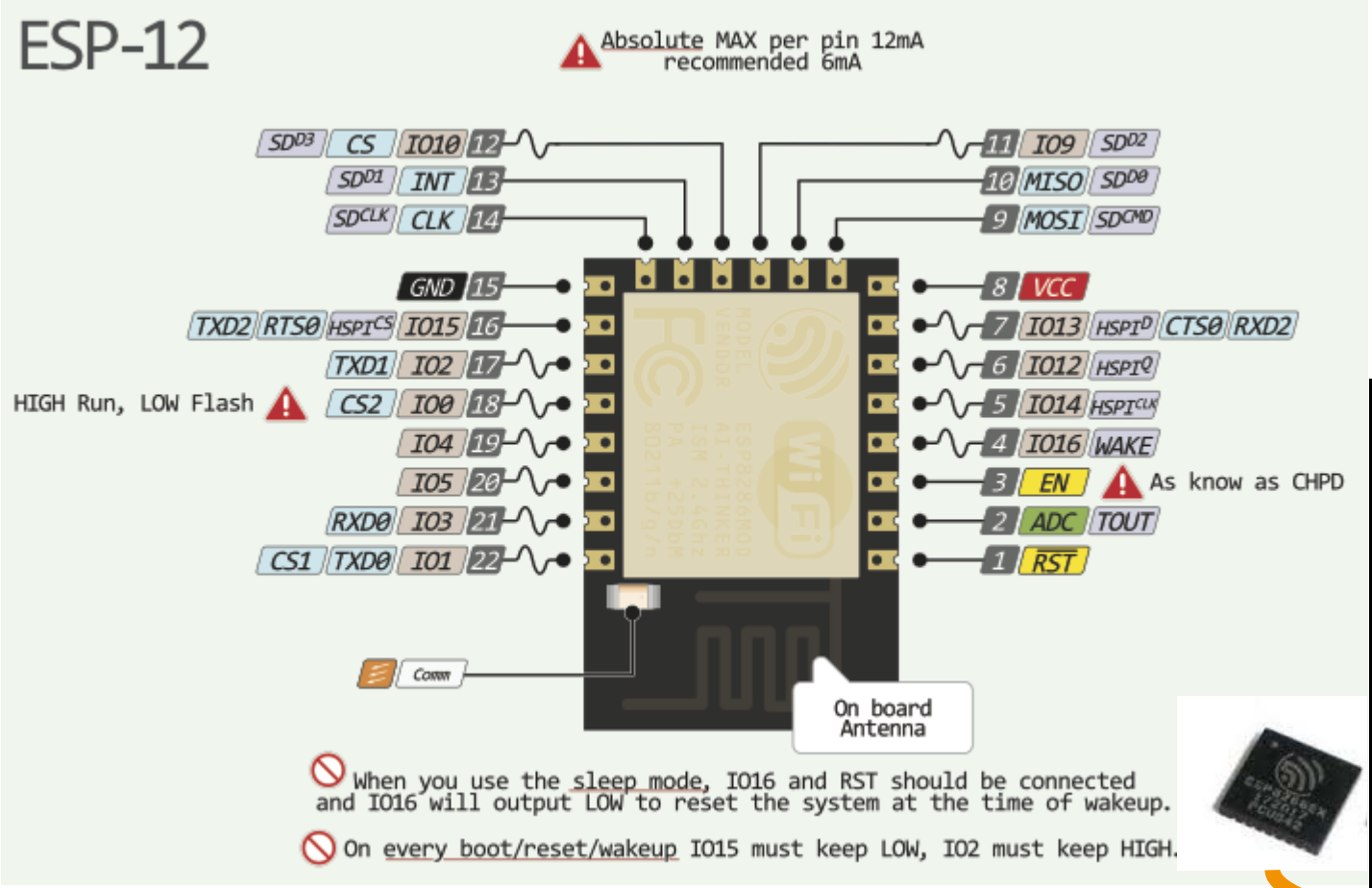
Mikrocontroller als Bauteil

– Wie schließe ich die an?

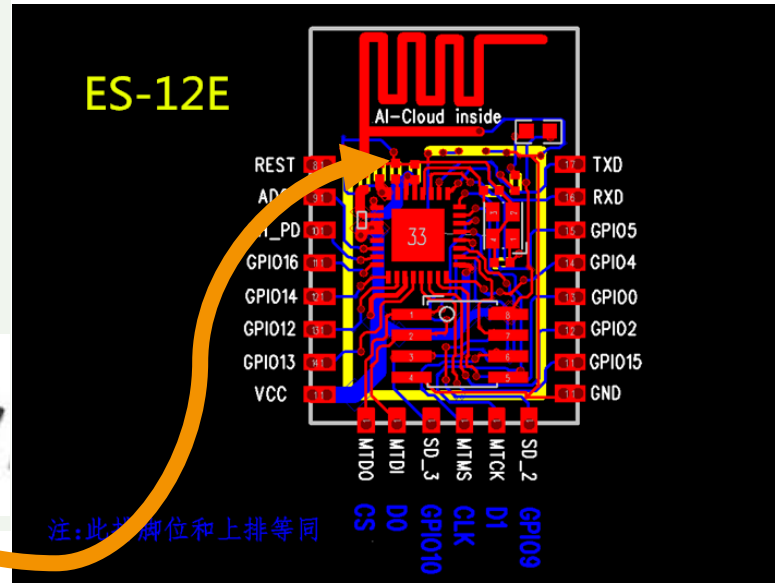
– Wie programmiere ich die?

– Worauf muss ich achten?

µC ESP8266 im Modul ESP12-F auf Wemos D1 Mini

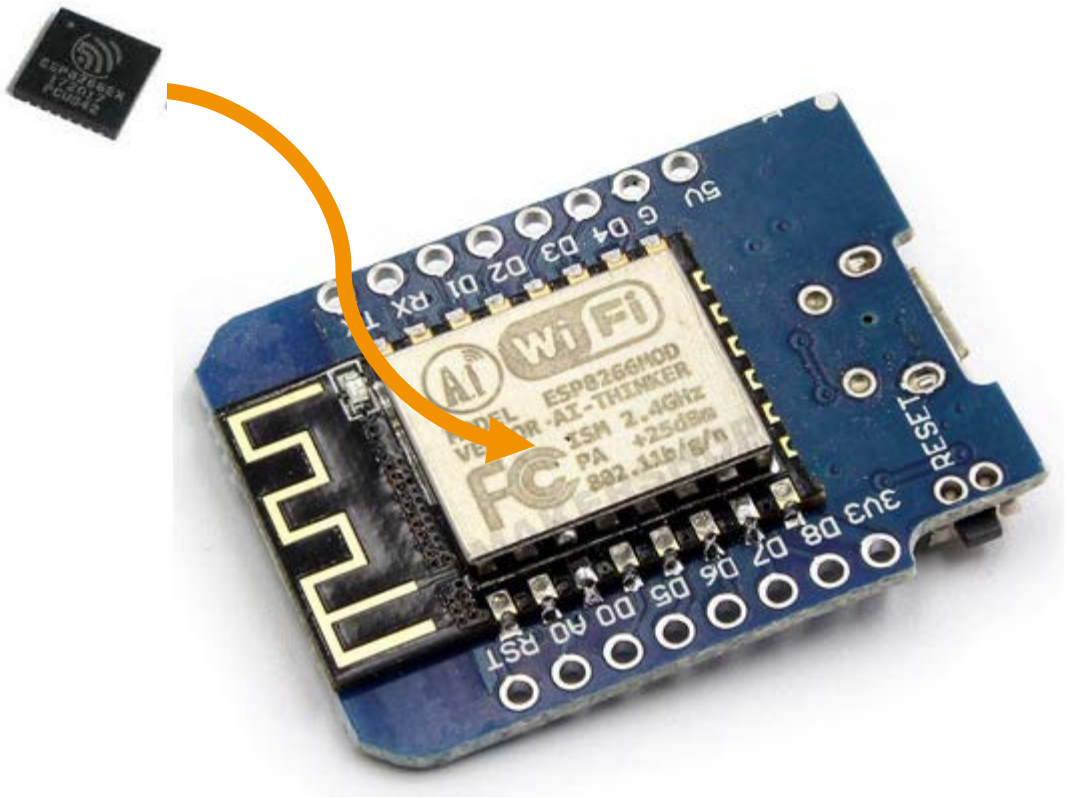


ESP12F = ESP8266EX
4MB Flash
Antenne
Quarz
LED
Platine



http://cholla.mmto.org/esp8266/esp12/

µC ESP8266 im Modul ESP12-F auf Wemos D1 Mini



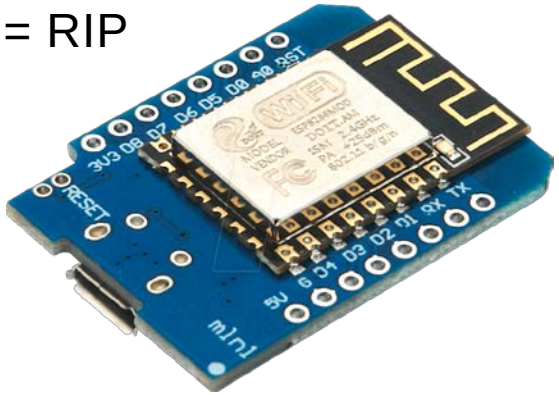
Wemos D1 Mini =

- ESP12-F
- USB Interface
- USB2Serial Konverter
- Boot/Flash Schaltung
- Spannungsregler
- Reset Taster
- Pins (2.54mm Pitch)

Operating Voltage	3.3V
Digital I/O Pins	11
Analog Input Pins	1(3.2V Max)
Clock Speed	80/160MHz
Flash	4M Bytes
Size	34.2*25.6mm
Weight	3g

5V an IO = RIP

Wemos D1 Mini Pins

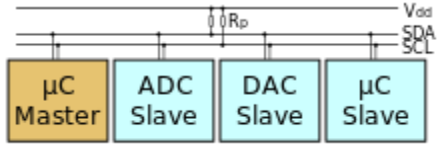


- *GPIO – GeneralPurposeInputOutput*
- *IO – Input/Out*
- *D0 bis D8* – Digitale Pins
  - Lesen 0 (LOW) oder 3.3V (HIGH)
  - Schreiben 0 (LOW) oder 3.3V (HIGH)
- *A0* – analoger Pin
  - Lesen 0 bis 3.2V

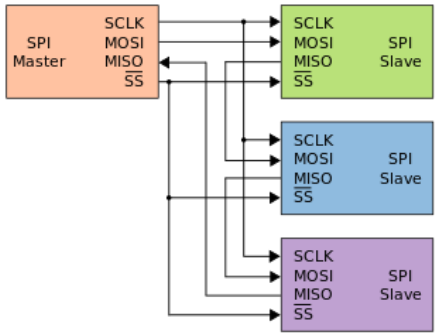
Pin	Function	ESP-8266 Pin
TX	TXD	TXD
RX	RXD	RXD
A0	Analog input, max 3.2V	A0
D0	IO	GPIO16
D1	IO, SCL	GPIO5
D2	IO, SDA	GPIO4
D3	IO, 10k Pull-up	GPIO0
D4	IO, 10k Pull-up, BUILTIN_LED	GPIO2
D5	IO, SCK	GPIO14
D6	IO, MISO	GPIO12
D7	IO, MOSI	GPIO13
D8	IO, 10k Pull-down, SS	GPIO15
G	Ground	GND
5V	5V	-
3V3	3.3V	3.3V
RST	Reset	RST

Wemos D1 Mini GPIO

- 10 Bit ADC (AnalogDigitalConverter)
- 0V bis 3.2V in 2^10 diskrete Werte aufteilen
- SDA/SCL (I2C Bus)



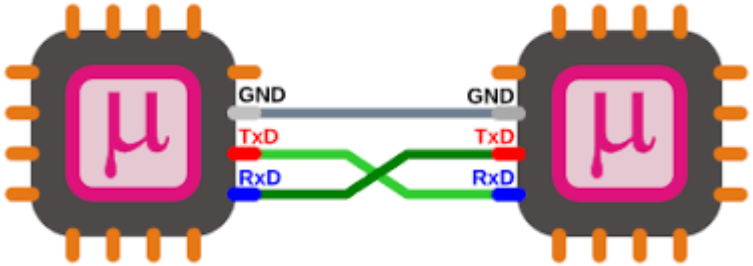
- SCK, MISO, MOSI, SS (SPI Bus)



Pin	Function	ESP-8266 Pin
TX	TXD	TXD
RX	RXD	RXD
A0	Analog input, max 3.2V	A0
D0	IO	GPIO16
D1	IO, SCL	GPIO5
D2	IO, SDA	GPIO4
D3	IO, 10k Pull-up	GPIO0
D4	IO, 10k Pull-up, BUILTIN_LED	GPIO2
D5	IO, SCK	GPIO14
D6	IO, MISO	GPIO12
D7	IO, MOSI	GPIO13
D8	IO, 10k Pull-down, SS	GPIO15
G	Ground	GND
5V	5V	-
3V3	3.3V	3.3V
RST	Reset	RST

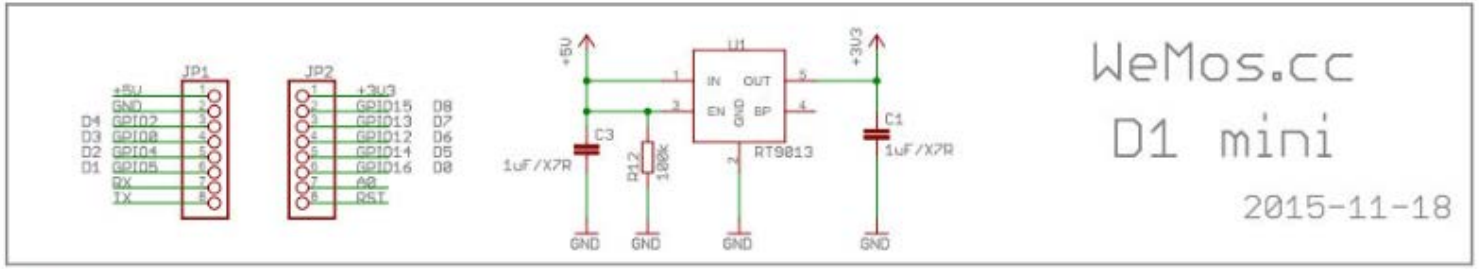
Wemos D1 Mini GPIO

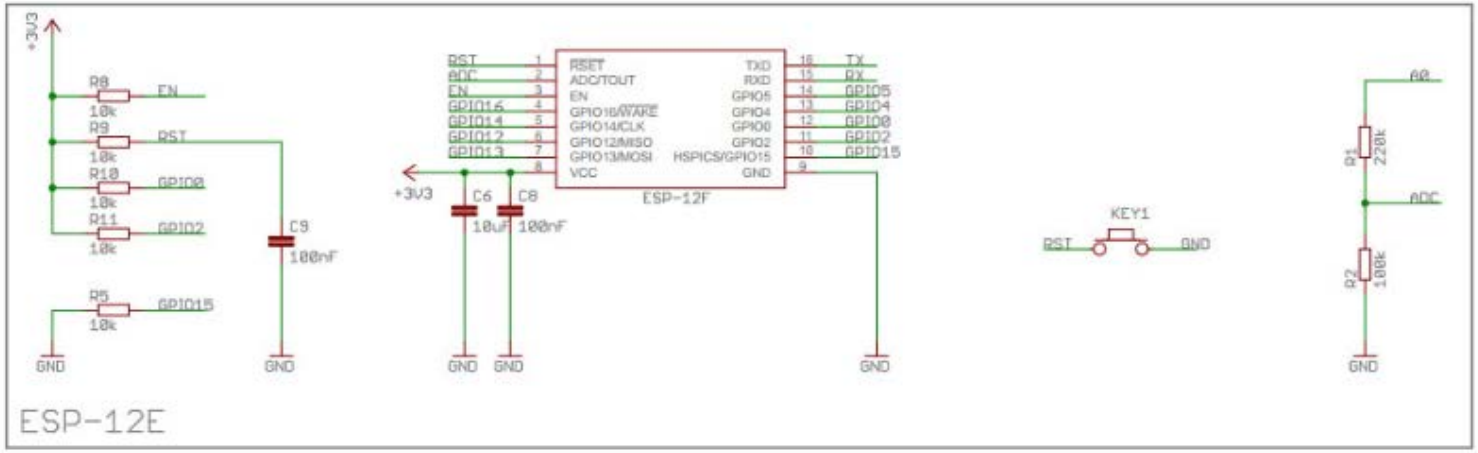
- TX/RX (UART)
- TX -Transmit
- RX - Receive
- Für die Serielle Kommunikation (Zwischen z.B PC und µC oder Sensoren)

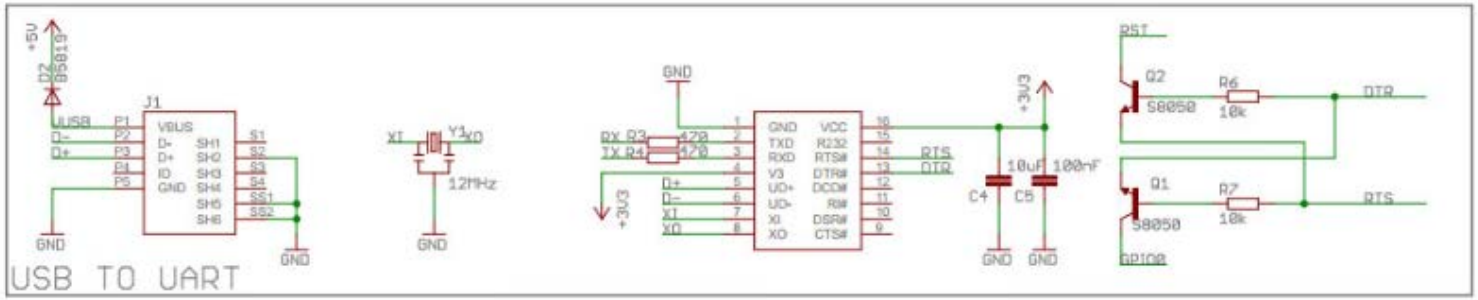


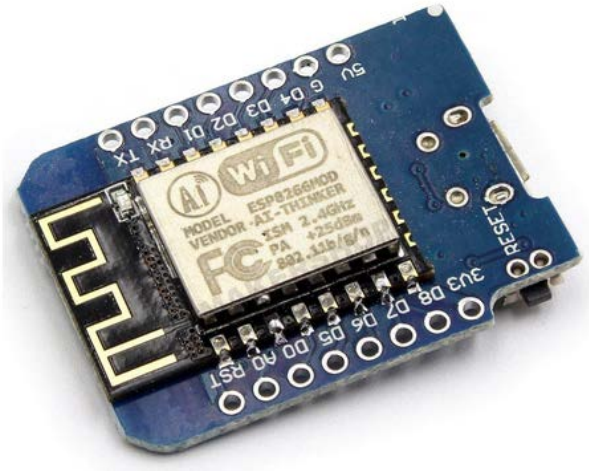
Pin	Function	ESP-8266 Pin
TX	TXD	TXD
RX	RXD	RXD
A0	Analog input, max 3.2V	A0
D0	IO	GPIO16
D1	IO, SCL	GPIO5
D2	IO, SDA	GPIO4
D3	IO, 10k Pull-up	GPIO0
D4	IO, 10k Pull-up, BUILTIN_LED	GPIO2
D5	IO, SCK	GPIO14
D6	IO, MISO	GPIO12
D7	IO, MOSI	GPIO13
D8	IO, 10k Pull-down, SS	GPIO15
G	Ground	GND
5V	5V	-
3V3	3.3V	3.3V
RST	Reset	RST

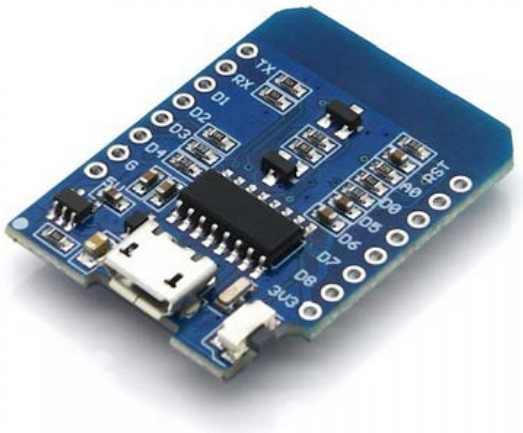
Schaltplan


JP1
JP2
U1
RT9013
WeMos.cc
D1 mini
2015-11-18

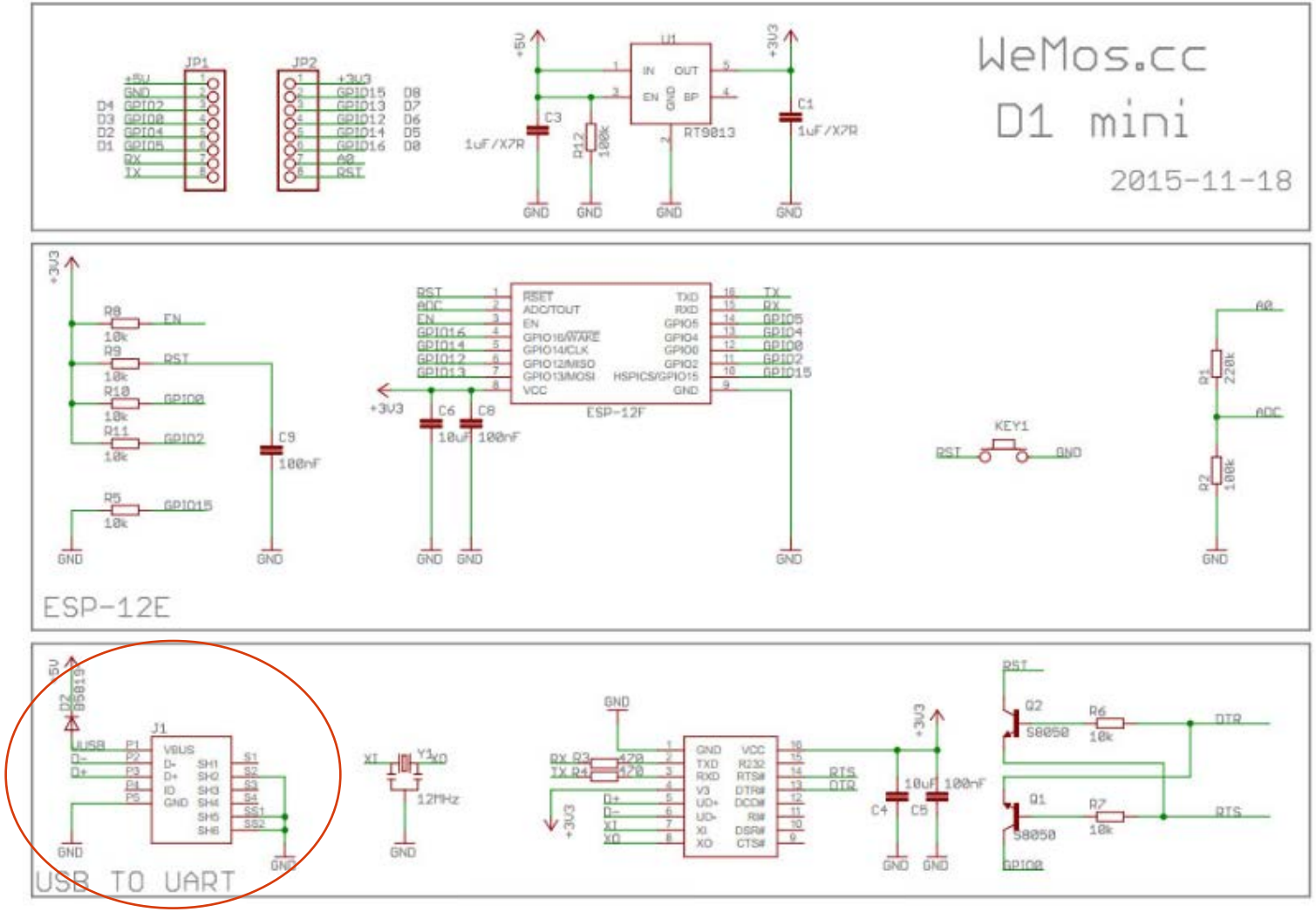

ESP-12F
KEY1
ESP-12E

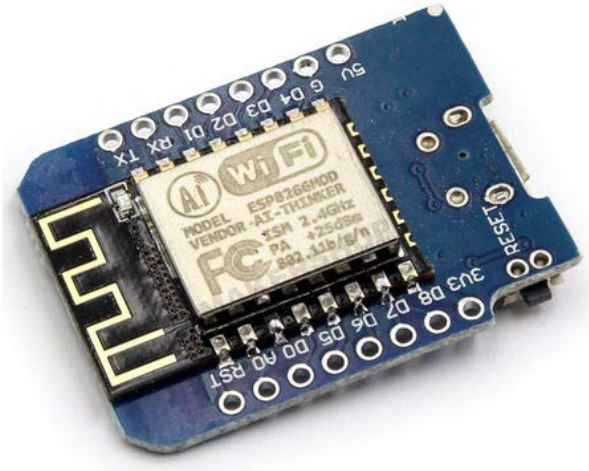

J1
Y1
12MHz
Q2
Q1
S8050
USB TO UART

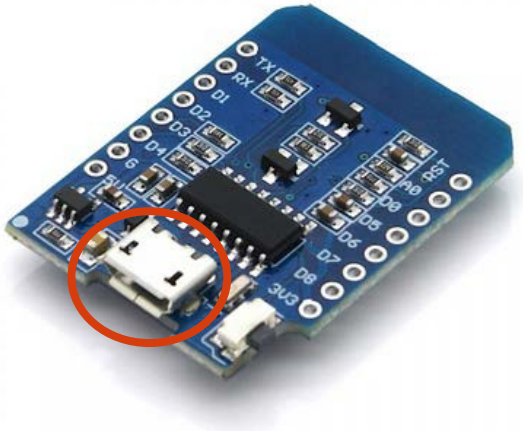




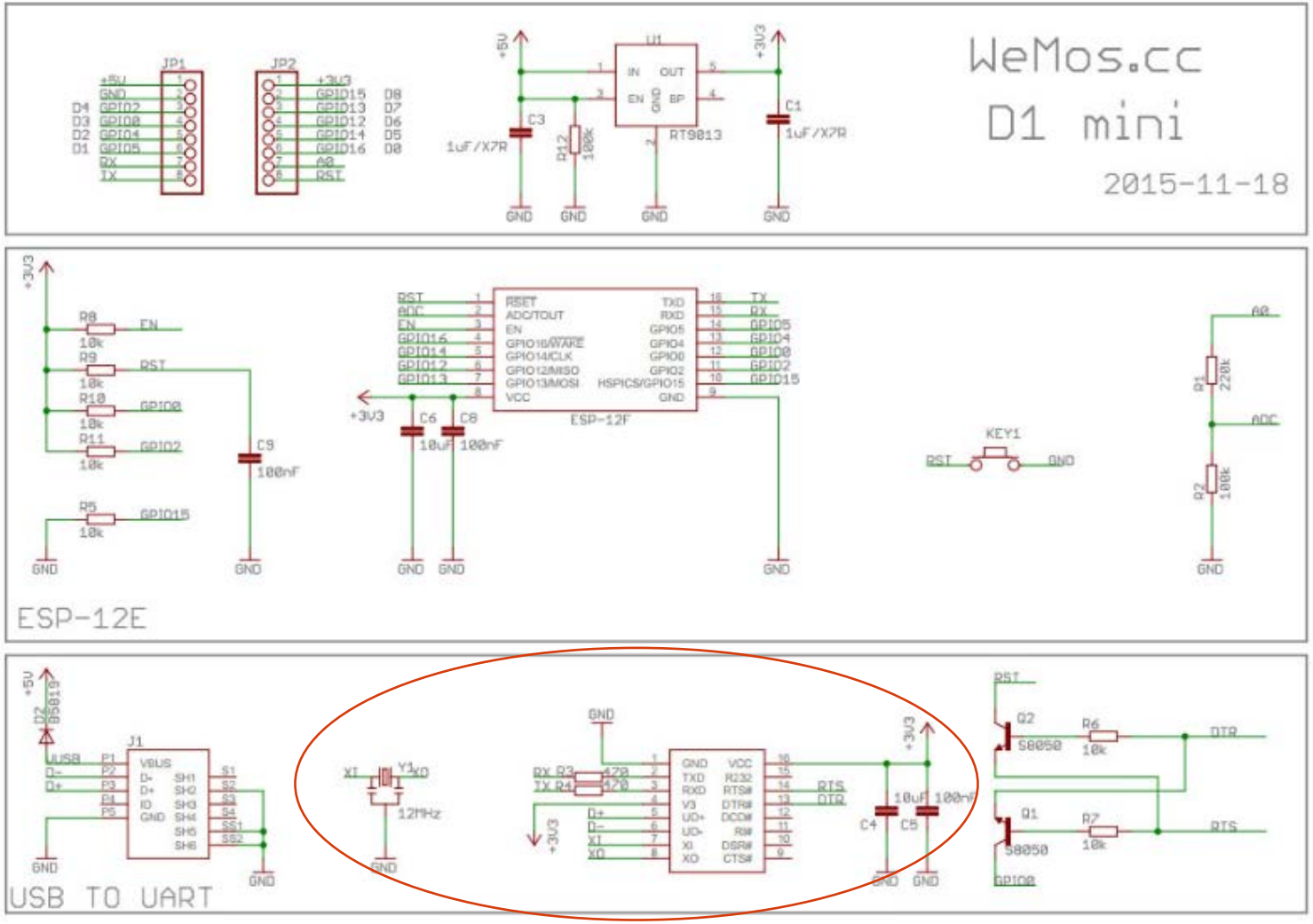
Schaltplan

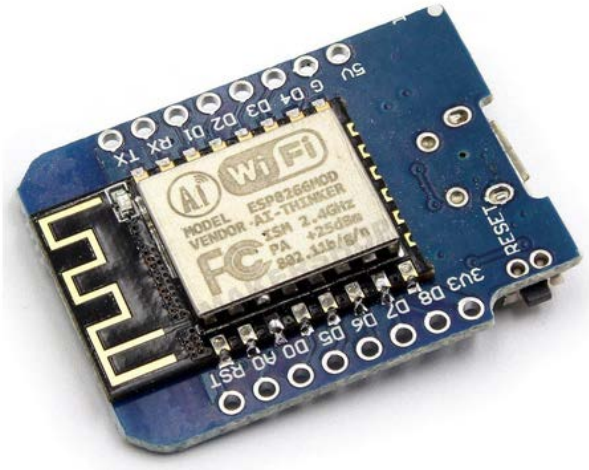


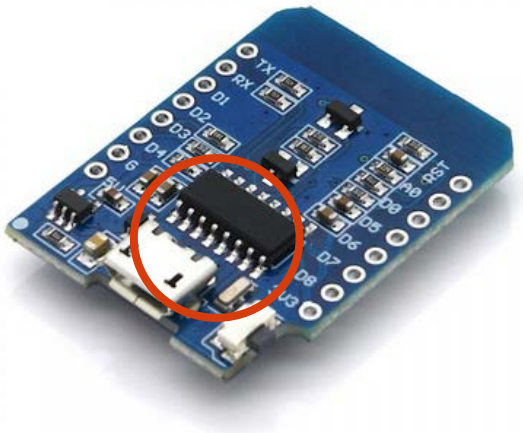




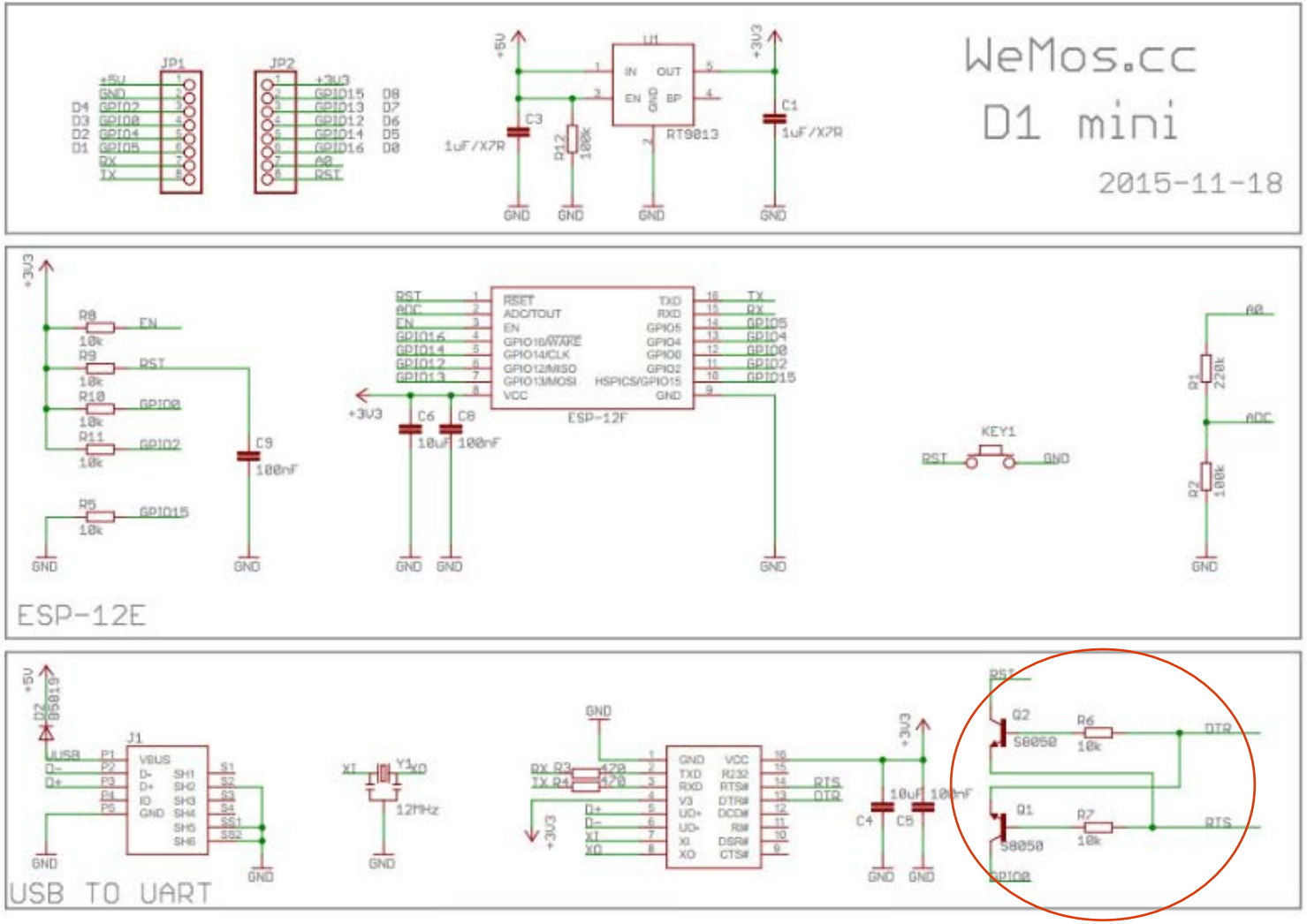
Schaltplan

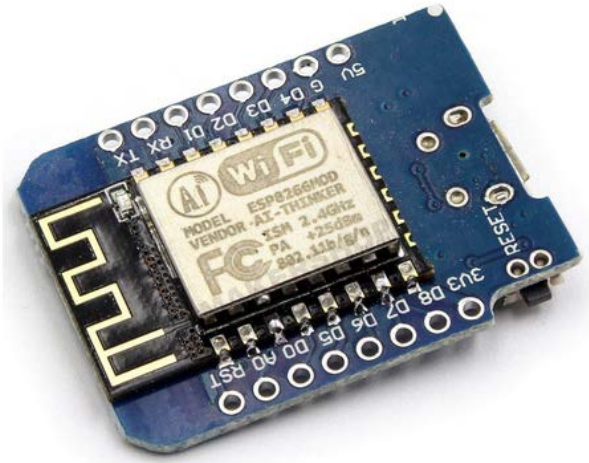


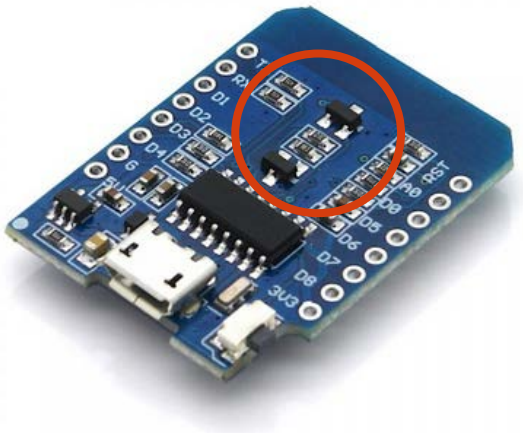




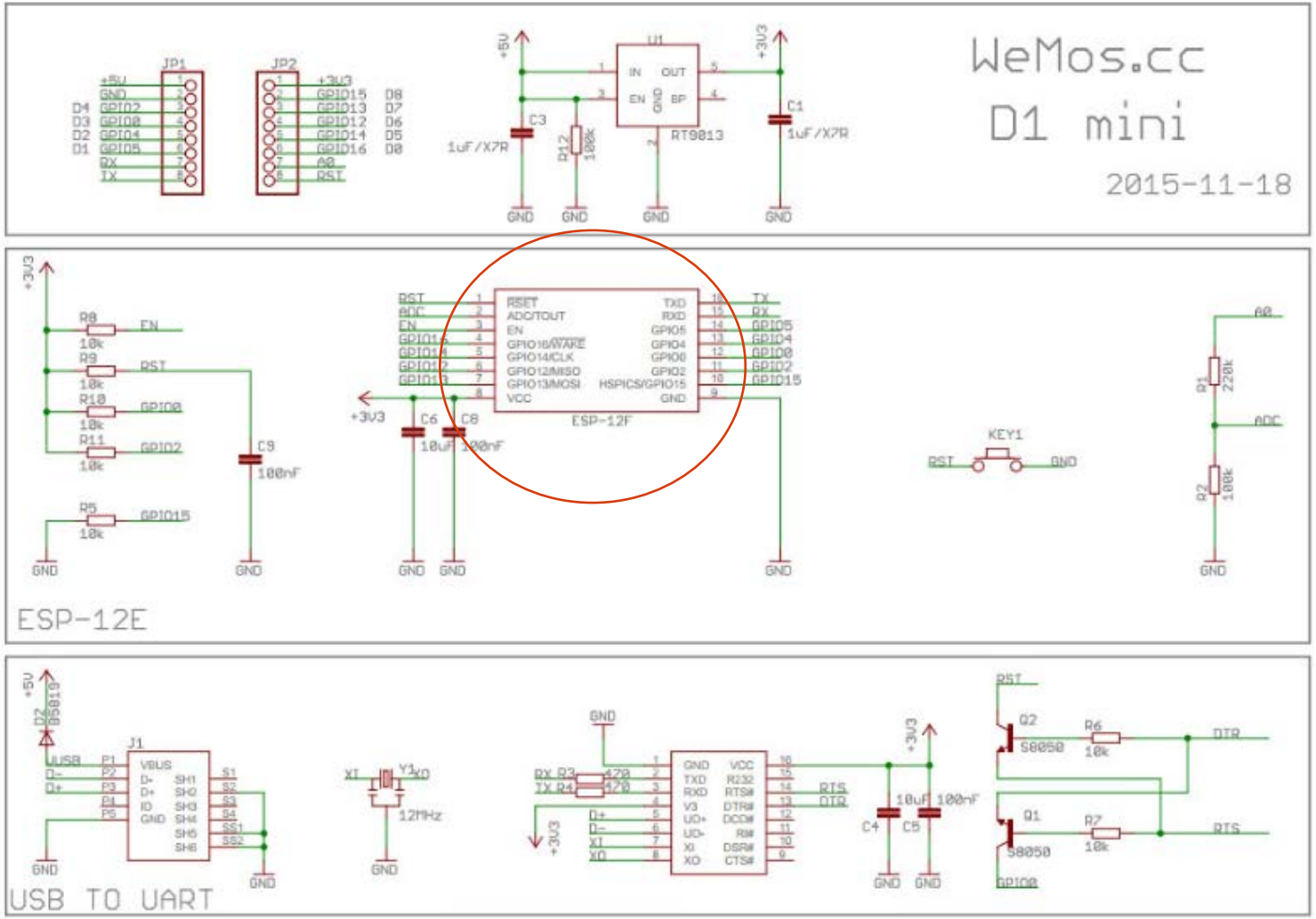
Schaltplan

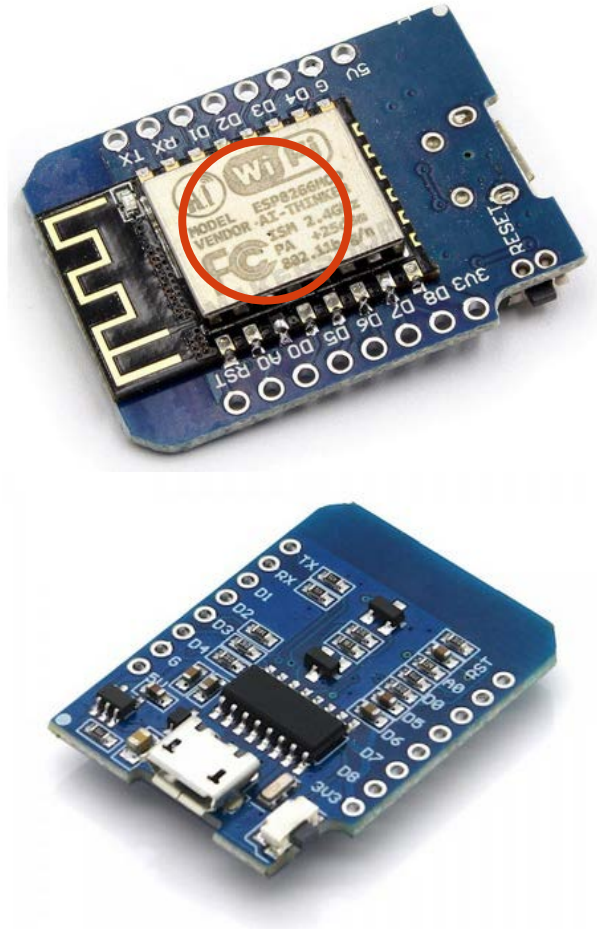




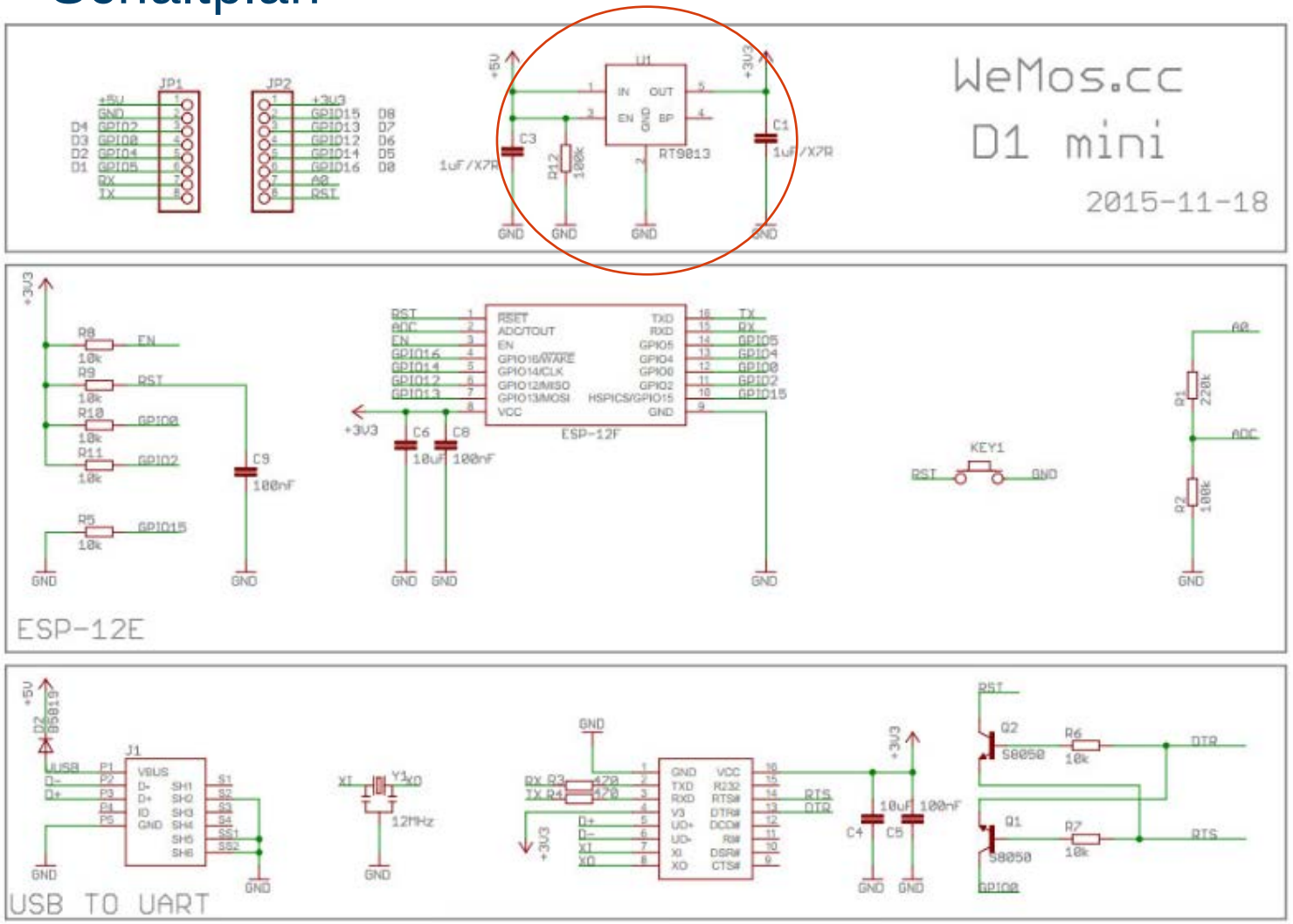


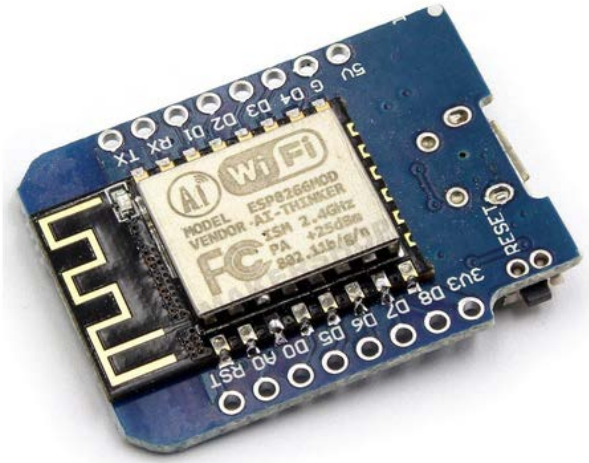
Schaltplan

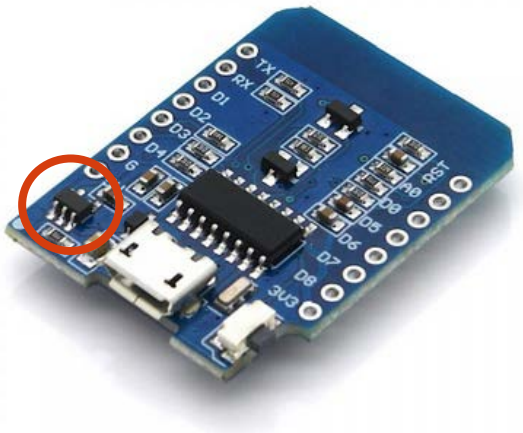




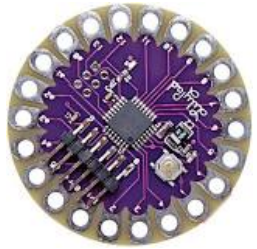
Schaltplan







Programmierung



- Arduino Framework
  - Diverse Open Source Entwicklungsboards
  - Eigene und überschaubare Entwicklungsumgebung
  - Toolchain zum Programmieren
- Programmierung in C/C++ mit Arduino spezifischen Vereinfachungen
- Große Open Source Community: Zahlreiche Beispiele!
- Einfache Arduino Bibliotheken für (fast) jeden Sensor und Aktor

Arduino Framework: Datentypen

Variables, Arrays, and Data

Data Types

boolean	true | false
char	-128 - 127, 'a' '$' etc.
unsigned char	0 - 255
byte	0 - 255
int	-32768 - 32767
unsigned int	0 - 65535
word	0 - 65535
long	-2147483648 - 2147483647
unsigned long	0 - 4294967295
float	-3.4028e+38 - 3.4028e+38
double	currently same as float
void	i.e., no return value

Strings

```
char str1[8] =
  {'A','r','d','u','i','n','o','\0'};
  // Includes \0 null termination
char str2[8] =
  {'A','r','d','u','i','n','o'};
  // Compiler adds null termination
char str3[] = "Arduino";
char str4[8] = "Arduino";
```

Numeric Constants

123	decimal
0b01111011	binary
0173	octal - base 8
0x7B	hexadecimal - base 16
123U	force unsigned
123L	force long
123UL	force unsigned long
123.0	force floating point
1.23e6	1.23*10^6 = 1230000

Qualifiers

static	persists between calls
volatile	in RAM (nice for ISR)
const	read-only
PROGMEM	in flash

Arrays

```
int myPins[] = {2, 4, 8, 3, 6};
int myInts[6];    // Array of 6 ints
myInts[0] = 42;   // Assigning first
                  // index of myInts
myInts[6] = 12;   // ERROR! Indexes
                  // are 0 though 5
```

https://i.redd.it/8m27xkluynd31.png

Arduino Framework: Programmkopf

- Bibliotheken einbinden
- Konstanten definieren
- Objekte instanziieren
- Globale Variablen anlegen und initialisieren

Baum | Arduino 1.8.13

Datei Bearbeiten Sketch Werkzeuge Hilfe

Baum

```
#include <ESP8266WiFi.h>
#include <ESP8266mDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include <FastLED.h>
#include <ArduinoJson.h>
#include <PubSubClient.h>

#define FASTLED_ALLOW_INTERRUPTS 0
#define FASTLED_ESP8266_RAW_PIN_ORDER

#define DATA_PIN    D3
#define NUM_LEDS    300
CRGB leds[NUM_LEDS];

#define BRIGHTNESS          96
#define FRAMES_PER_SECOND  120

// COOLING: How much does the air cool as it rises?
// Less cooling = taller flames.  More cooling = shorter flames.
// Default 50, suggested range 20-100
#define COOLING  55

// SPARKING: What chance (out of 255) is there that a new spark will be lit?
// Higher chance = more roaring fire.  Lower chance = more flickery fire.
// Default 120, suggested range 50-200.
#define SPARKING 120

WiFiClient espClient;
PubSubClient client(espClient);


const char* ssid = "XXX";
const char* password = "XXX";
const char* mqtt_server = "3L15t4.xyz";
const char* mqtt_topic = "ddiBaum";

enum power {
  ON, OFF
};
```

Arduino Framework: Setup

- Jedes Arduino Programm hat eine Setup-Methode
- Wird nur 1x aufgerufen beim Start
- Initalisierung
  - DataDirectionRegister setzen
  - Zustände setzen
  - Objekte erzeugen/konfigurieren
  - Weiteren Betrieb vorbereiten

```
void setup() {
  delay(3000); // 3 second delay for recovery
  pinMode(D5, INPUT);
  digitalWrite(D5, HIGH);
  FastLED.addLeds<WS2811, DATA_PIN, GRB>(leds, NUM_LEDS).setCorre
  FastLED.setMaxPowerInVoltsAndMilliamps(5, 1800);
  FastLED.setBrightness(90);
  Serial.begin(115200);
  Serial.println("Booting");
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.waitForConnectResult() != WL_CONNECTED) {
    Serial.println("Connection Failed! Rebooting...");
    delay(5000);
    ESP.restart();
  }
```

Arduino Framework: Loop

- Jedes Arduino Programm hat eine Loop-Methode
- Wird in Endlosschleife nach Setup ausgeführt
- Systembetrieb
  - Messungen durchführen
  - Zustände ändern
  - Auf Daten warten oder absenden
  - Aktoren steuern
  - …

```
void loop() {
  ArduinoOTA.handle();

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  if (animation_enabled) {
    if (cur_animation == RAINBOW) {
      rainbow();

    }
    if (cur_animation == JUGGLE) {
      juggle();
    }
    if (cur_animation == FIRE) {
      fire();
    }
    if (cur_animation == CYCLON) {
      if (cyclon_dir == 0) {
        cyclon_i++;
      }
      if (cyclon_dir == 1) {
        cyclon_i--;
      }

      if (cyclon_i == 300) {
        cyclon_dir = 1;
      }
```

Arduino Framework: Konstanten

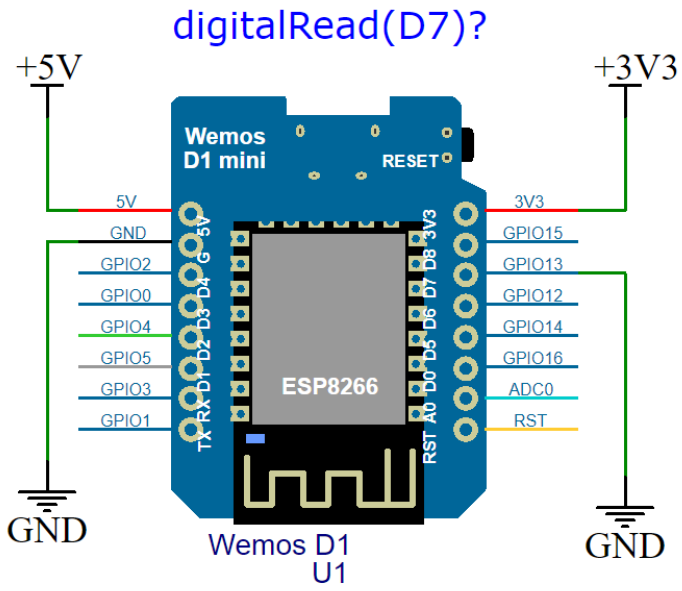
- HIGH
  - High Pegel = 3.3V
  - max 20mA GPIO
  - Synonym zu 1 und true bei if-Abfragen
- LOW
  - Low Pegel = 0V
- INPUT
  - INPUT_PULLUP: falls Pin internen Pull-Up Widerstand hat
- OUTPUT
- LED_BUILTIN

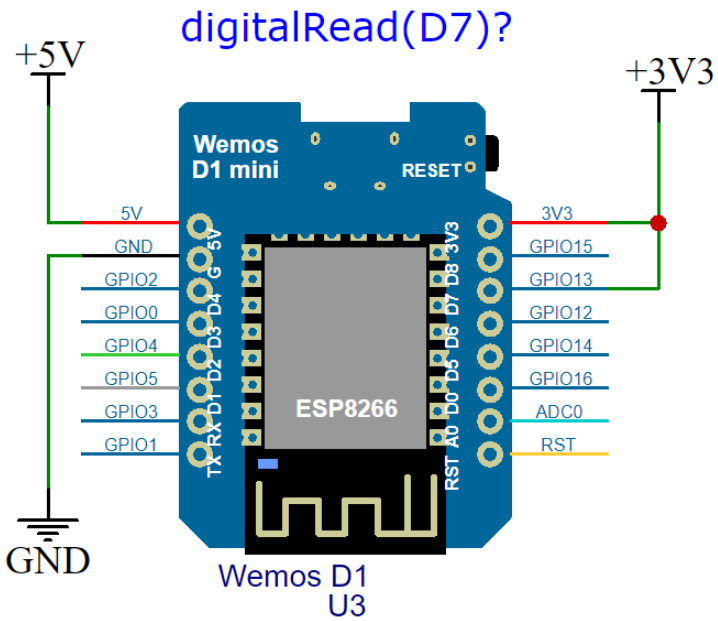
Arduino Framework: Digital IO

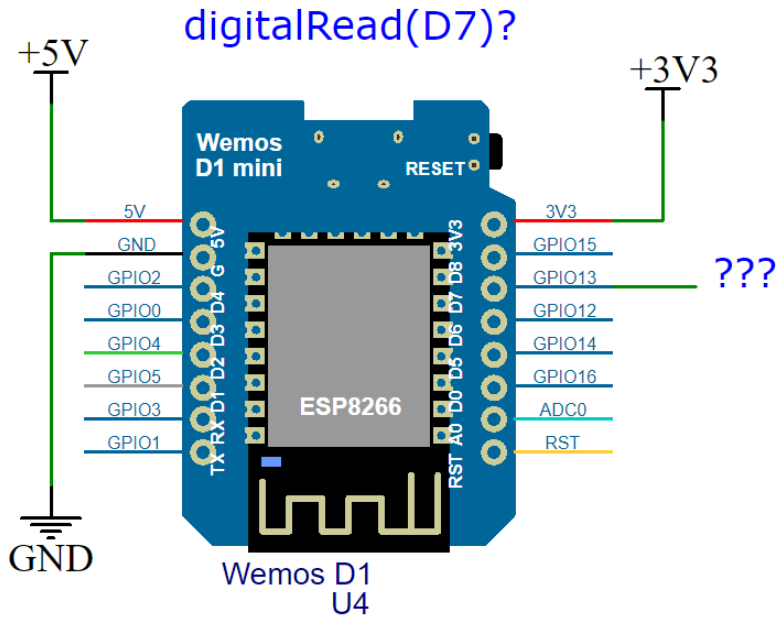
- pinMode([PIN], [INPUT, OUTPUT]);
  - Setze Pin PIN als Eingang (INPUT) oder Ausgang (OUTPUT)
- digitalWrite([PIN], [HIGH, LOW]);
  - Setze Spannungspegel am Ausgangspin PIN auf HIGH (3.3V) oder LOW (0V)
- digitalRead([PIN]);
  - Lese Spannungspegel an Eingangspin PIN
    - 0V bis 0.825V LOW
    - 2.475V bis 3.3V HIGH

```
pinMode(D1, INPUT);
pinMode(D2, OUTPUT);
pinMode(D3, INPUT);

digitalWrite(D2, HIGH);

int zustand = digitalRead(D1);
if(zustand == HIGH){
  ...
}
if(digitalRead(D1)){
  ...
} else {
  ...
}
```

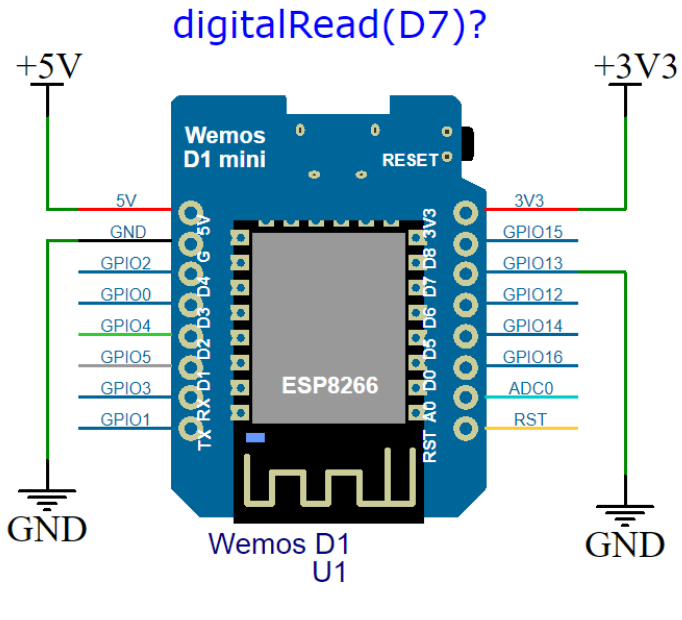
Arduino Framework: Digital IO



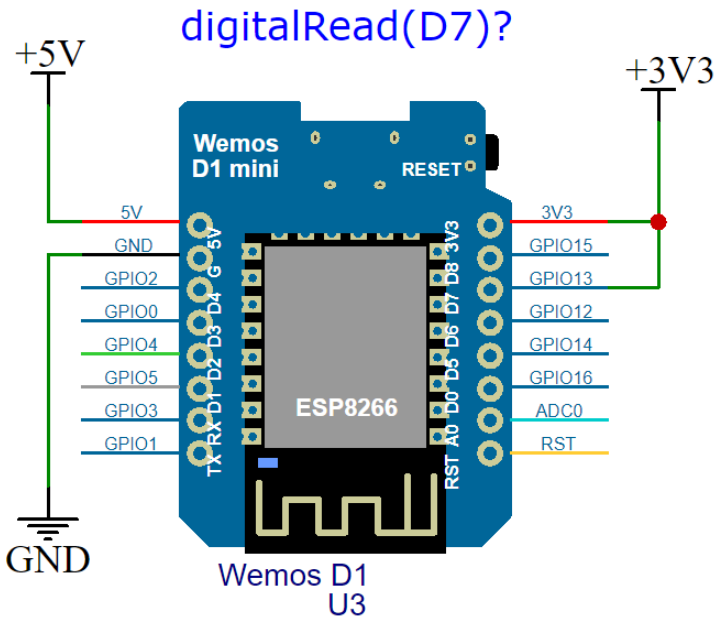




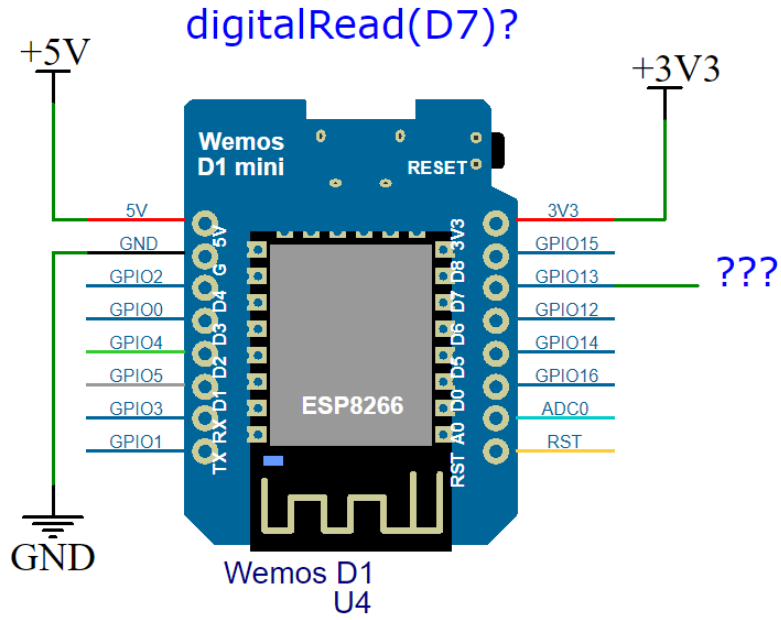
Arduino Framework: Digital IO



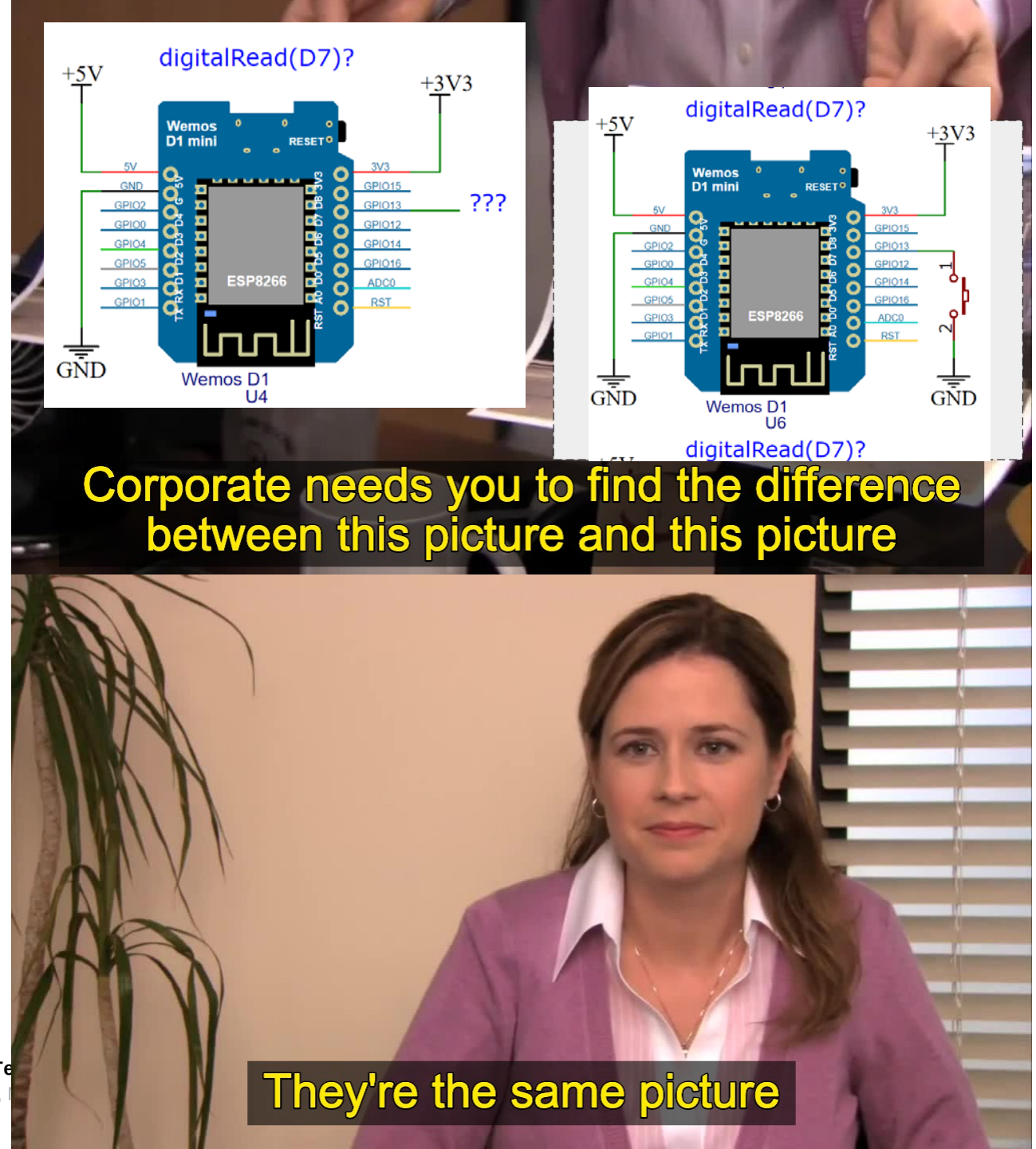
LOW

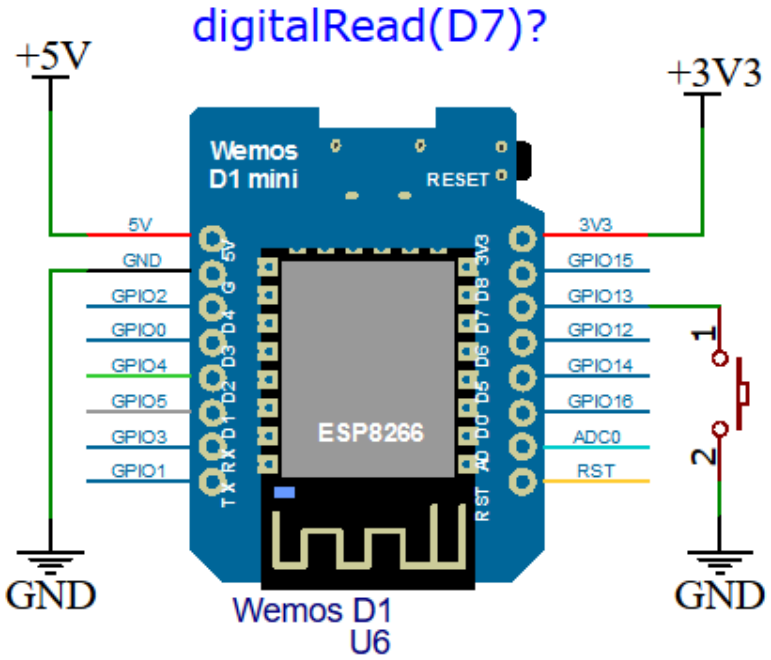


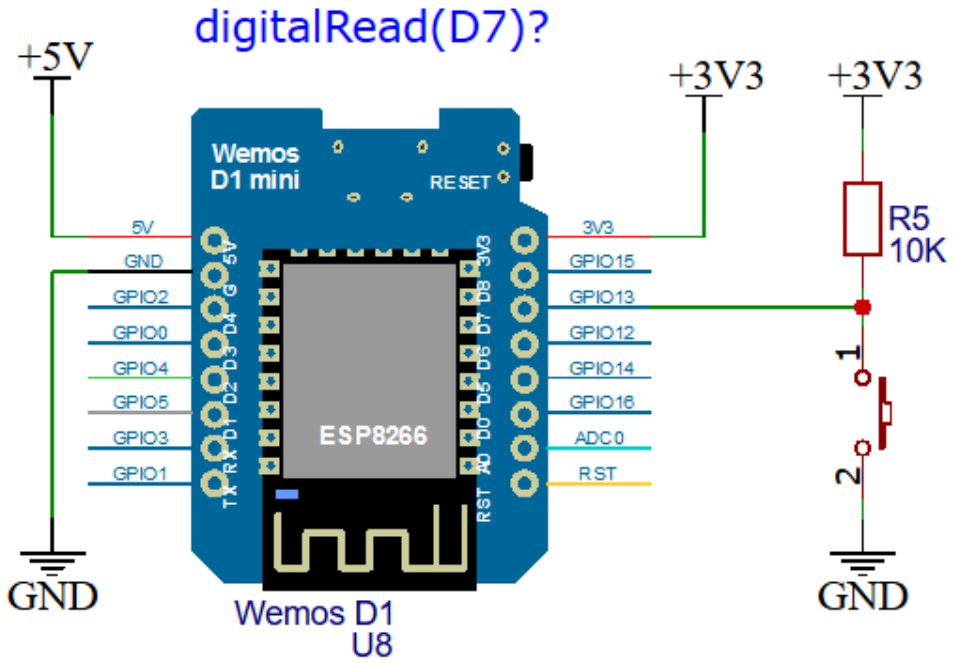
HIGH

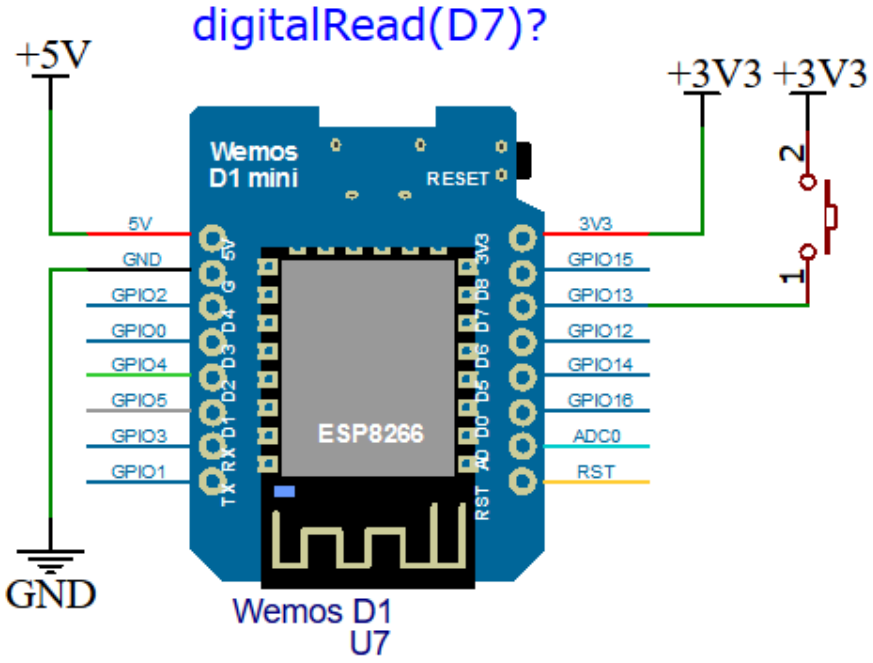


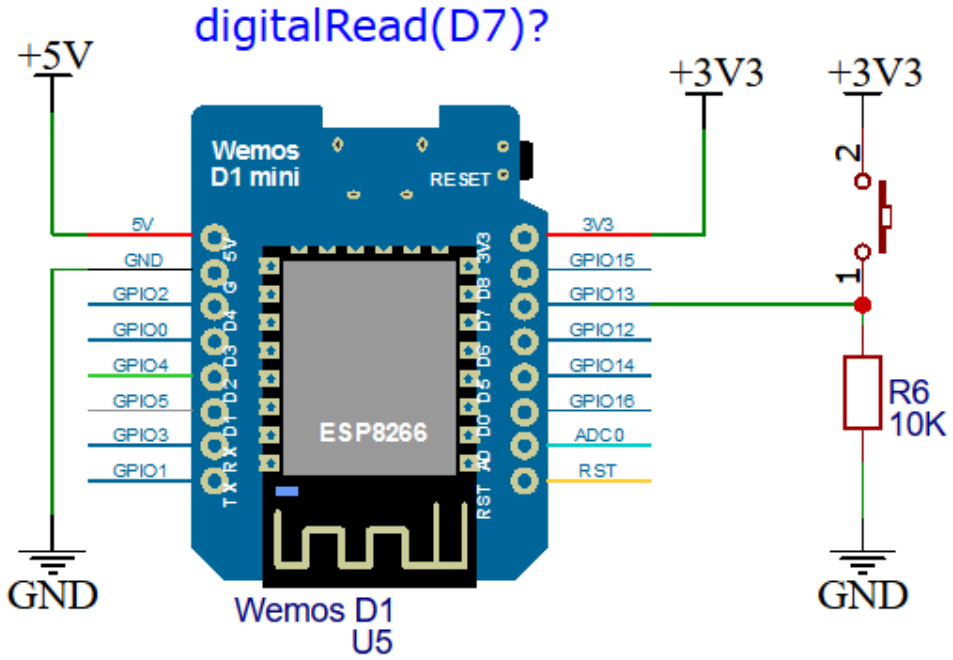
LOW?
HIGH?


digitalRead(D7)?
+5V
+3V3
Wemos
D1 mini
RESET
ESP8266
???
GND
Wemos D1
U4
digitalRead(D7)?
+5V
+3V3
Wemos
D1 mini
RESET
ESP8266
GND
GND
Wemos D1
U6
digitalRead(D7)?
Corporate needs you to find the difference
between this picture and this picture
They're the same picture


digitalRead(D7)?
+5V
+3V3
Wemos
D1 mini
RESET
5V
GND
GPIO2
GPIO0
GPIO4
GPIO5
GPIO3
GPIO1
3V3
GPIO15
GPIO13
GPIO12
GPIO14
GPIO16
ADC0
RST
ESP8266
1
2
GND
GND
Wemos D1
U6


digitalRead(D7)?
+5V
+3V3
+3V3
Wemos
D1 mini
RESET
5V
GND
GPIO2
GPIO0
GPIO4
GPIO5
GPIO3
GPIO1
3V3
GPIO15
GPIO13
GPIO12
GPIO14
GPIO16
ADC0
RST
ESP8266
R5
10K
1
2
GND
GND
Wemos D1
U8

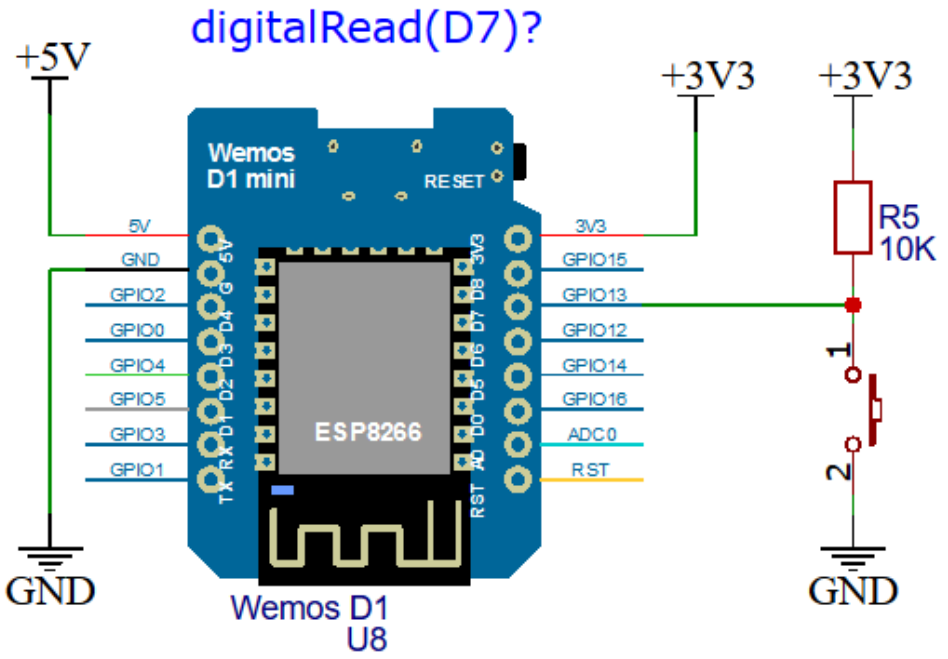

digitalRead(D7)?
+5V
+3V3
+3V3
Wemos
D1 mini
RESET
5V
GND
GPIO2
GPIO0
GPIO4
GPIO5
GPIO3
GPIO1
3V3
GPIO15
GPIO13
GPIO12
GPIO14
GPIO16
ADC0
RST
ESP8266
2
1
GND
GND
Wemos D1
U7

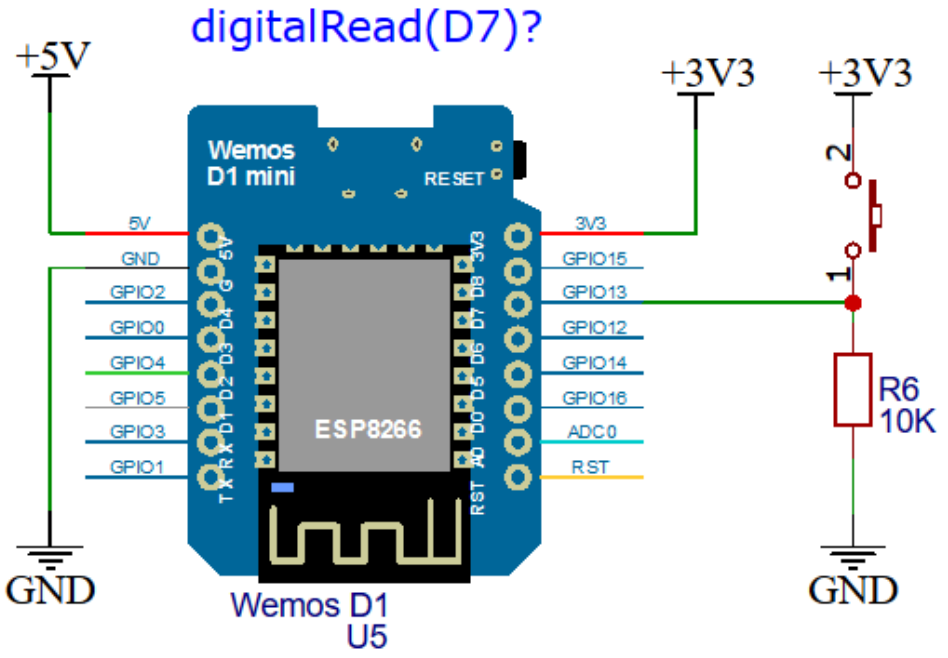

digitalRead(D7)?
+5V
+3V3
+3V3
Wemos
D1 mini
RESET
5V
GND
GPIO2
GPIO0
GPIO4
GPIO5
GPIO3
GPIO1
3V3
GPIO15
GPIO13
GPIO12
GPIO14
GPIO16
ADC0
RST
ESP8266
2
1
R6
10K
GND
GND
Wemos D1
U5

Arduino Framework: Digital IO

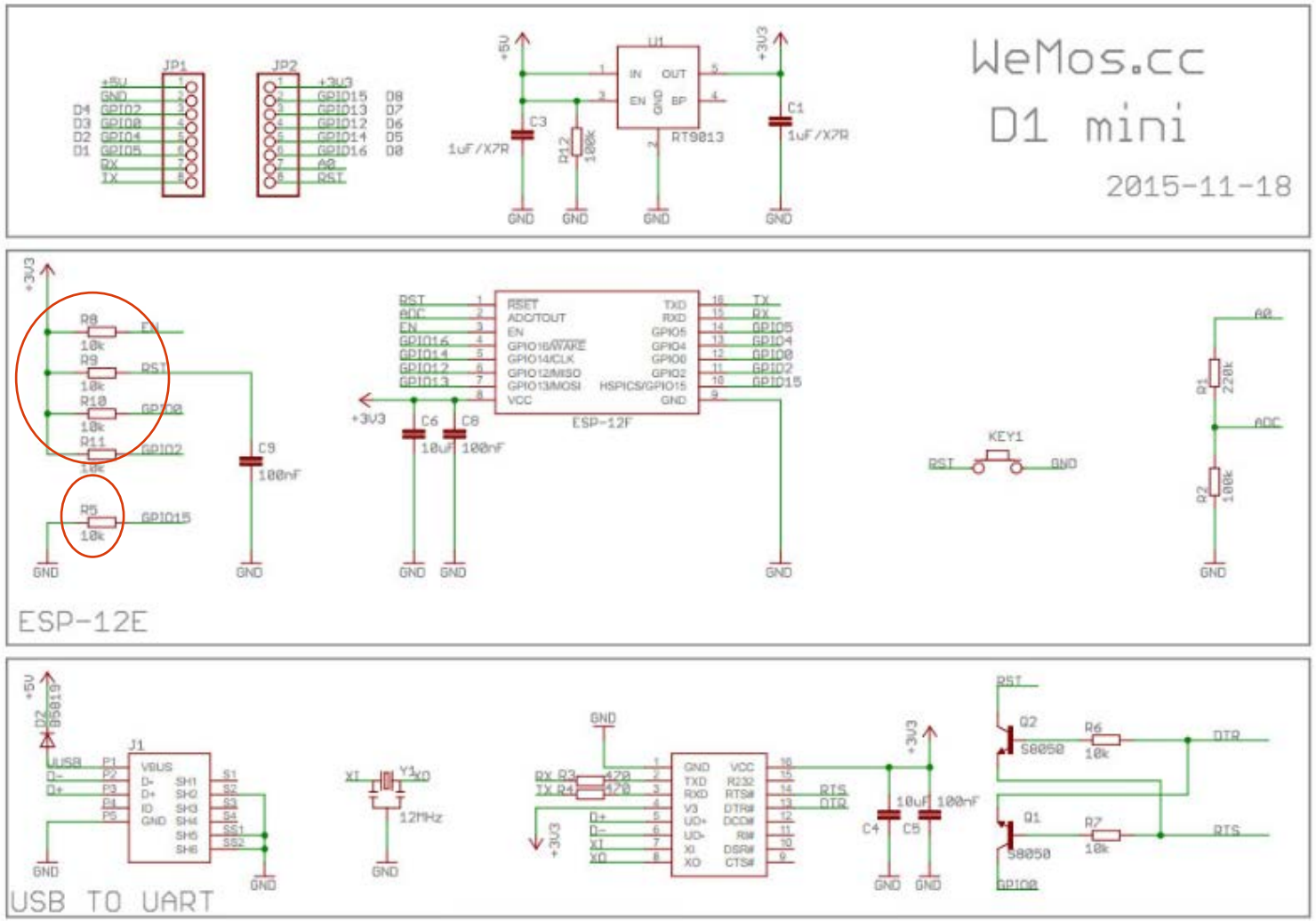
- digitalRead(D7)
  - HIGH, wenn Taster offen
  - LOW, wenn Taster geschlossen
  - PULL-UP Widerstand

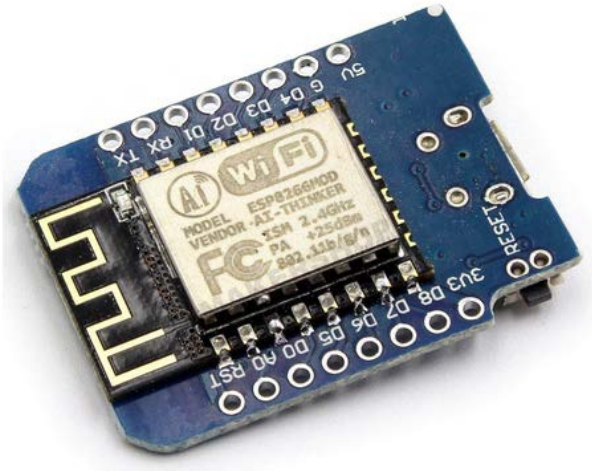
- digitalRead(D7)
  - LOW, wenn Taster offen
  - HIGH, wenn Taster geschlossen
  - PULL-DOWN Widerstand

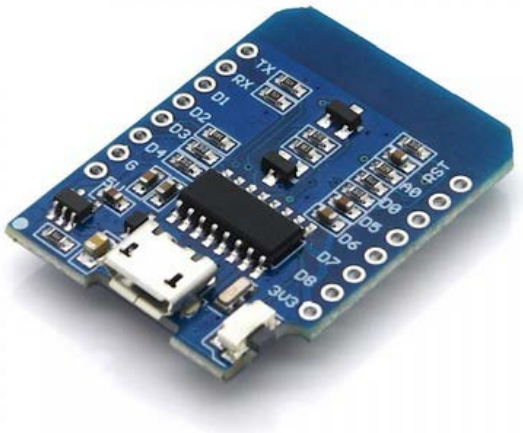




Schaltplan



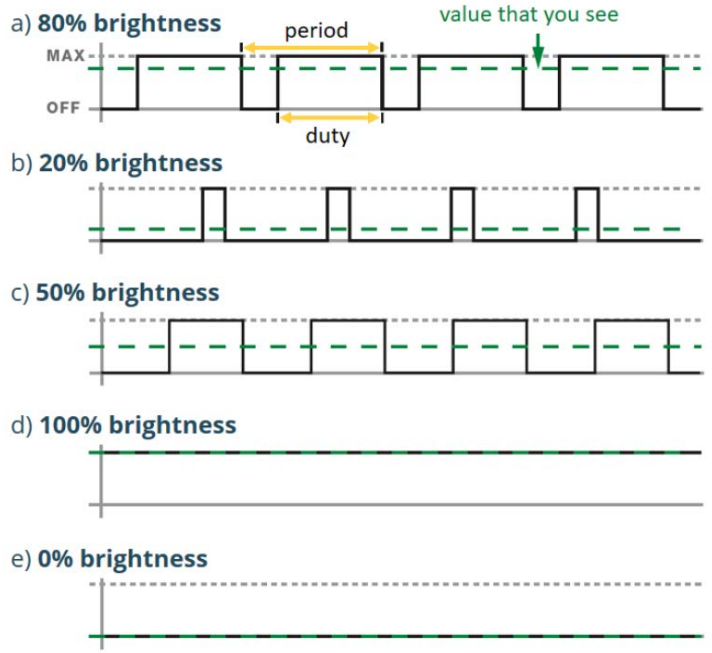




Arduino Framework: Analog IO

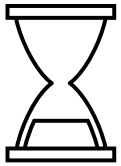
- analogRead(A0);
  - Lese Spannungspegel an Pin A0 und gebe diskreten Wert zwischen 0 und 1023 zurück
  - 10 Bit ADC
  - Angenommen analogRead gibt 500 zurück
  - 500/1023 = x / 3.2V
- analogWrite([PIN], [0 bis 1023]);
  - analogWrite(D2, 512);
    - 50% Duty Cycle ~ „1.65V“

```
pinMode(A0, INPUT);
int sensorWert = analogRead(A0);
if(sensorWert < 200){
  ...
}

if (analogRead(A0) > 500){
  ...
}
```



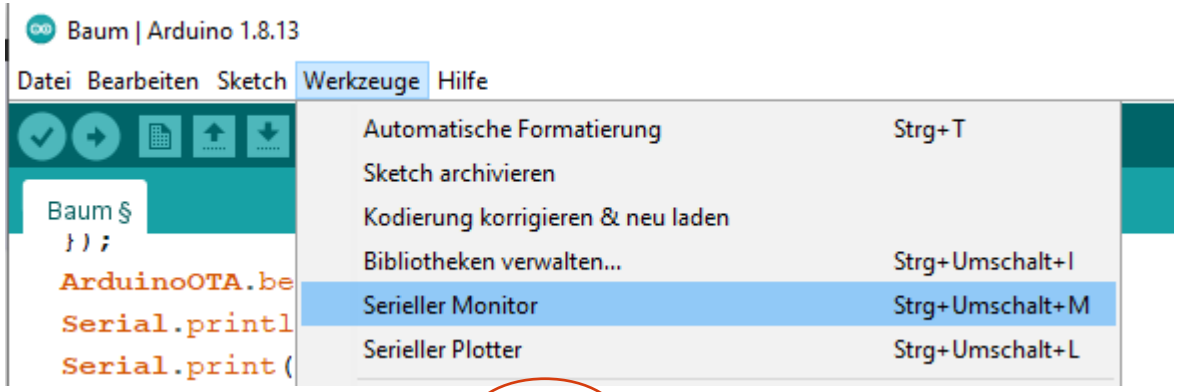
Arduino Framework: Zeit

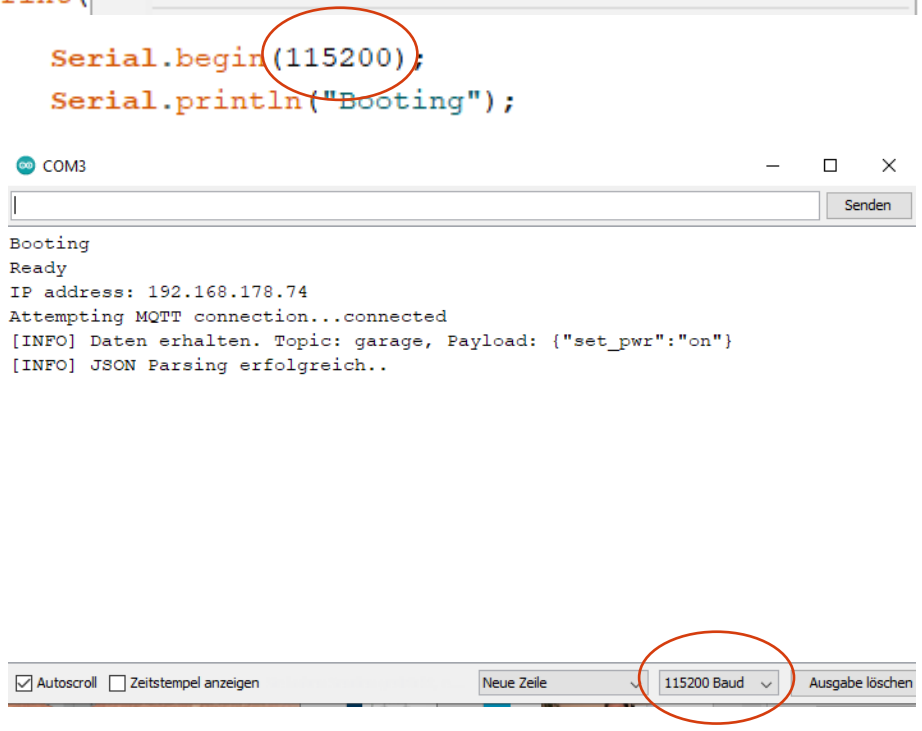
- unsigned long millis()
  - Systemzeit in ms
- unsigned long micros()
  - Systemzeit in µs
- delay([ZEIT IN MS])
  - delay(500) = warte 500ms
  - Aufruf ist blockierend! Nichts anderes wird in der Zwischenzeit ausgeführt



Arduino Framework: Serielle Schnittstelle

- Keine Haltemarken zum Debuggen
- Kein Bildschirm zum Ausgeben von Systemzuständen
- Lösung: Serial.begin(BAUDRATE)
  - BAUDRATE = Symbole pro Sekunde
  - 115200 für ESP8266
- Serial.println(„Hier bin ich angekommen“)





Arduino Referenz

- https://www.arduino.cc/reference/en/
- https://www.arduino.cc/reference/de/
- https://github.com/przygodyzkodem/Arduino-Cheat-Sheet/blob/master/Arduino_Cheat_Sheet_1-en.pdf

Live-Demo

Hello World des Physical Computings!

Arbeitsaufgaben

– Für die Arbeitsaufgaben, bitte dem Link folgen

https://cs.uol.de/s/jRZW6oDkexwk3y4