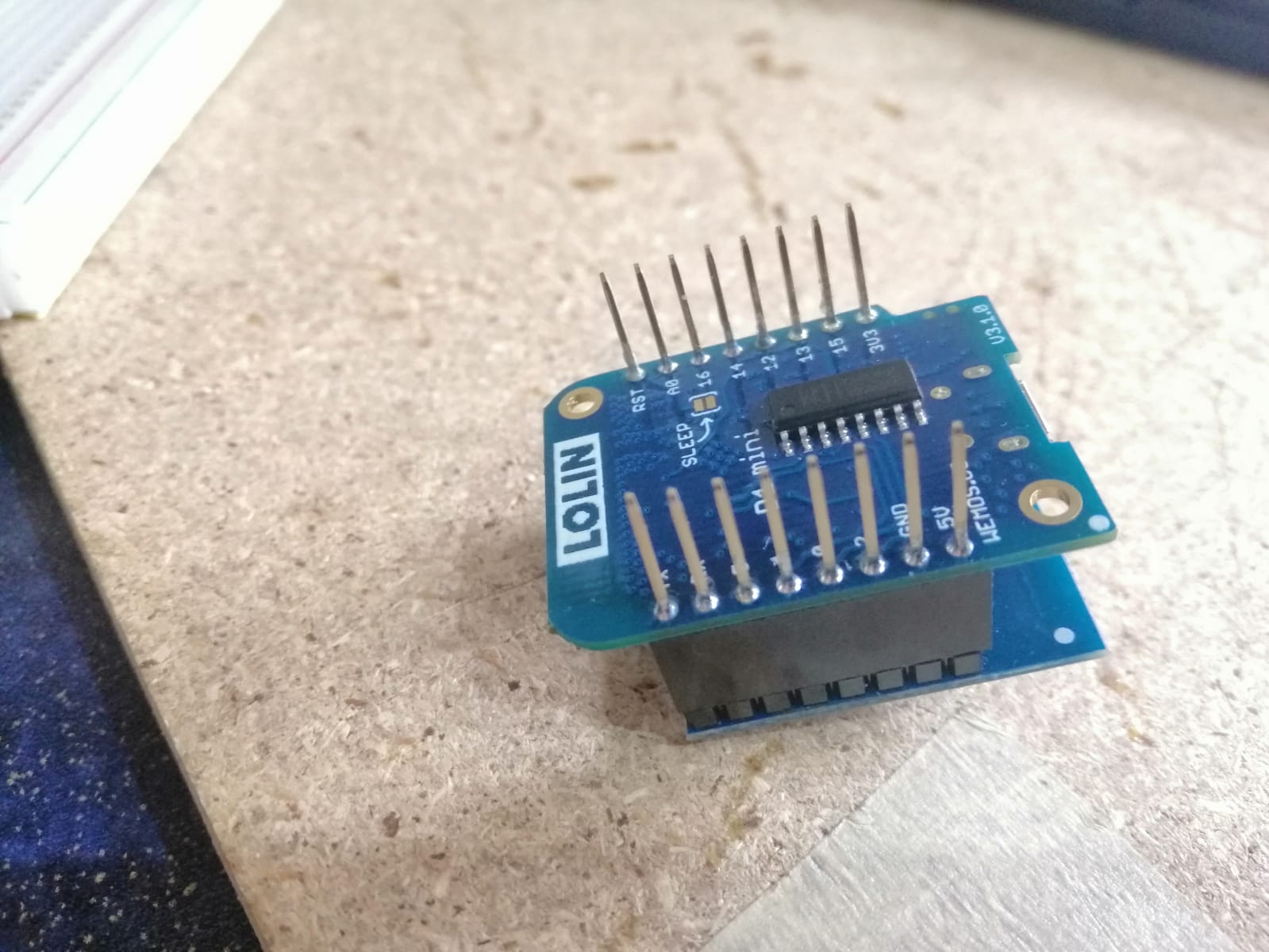
Übungsblatt 6: ESP8266 & M5Stack

Für Code und Videos siehe Ordner „Code“ und „Anhang“

# Aufgabe 1.1



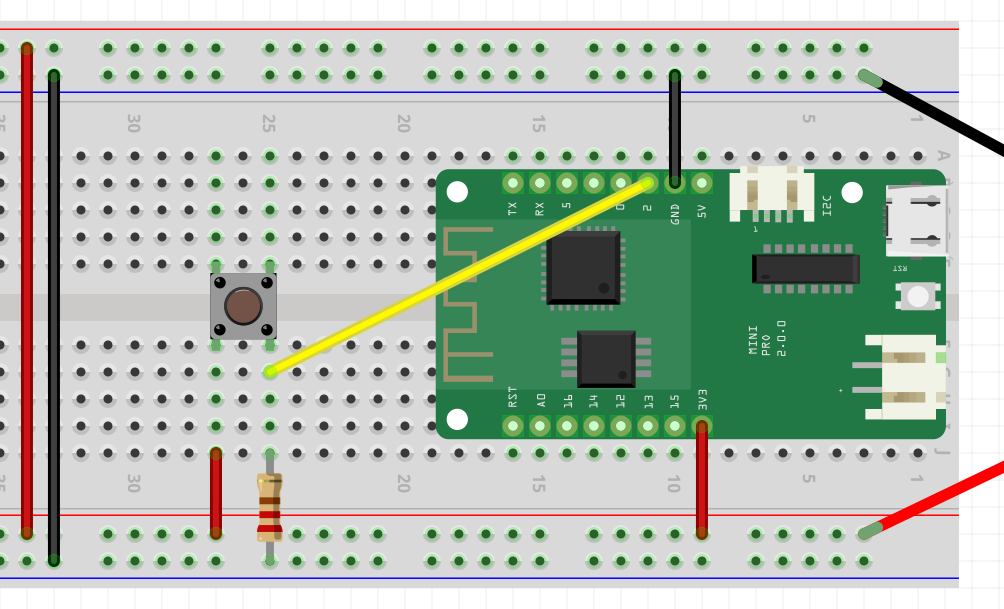
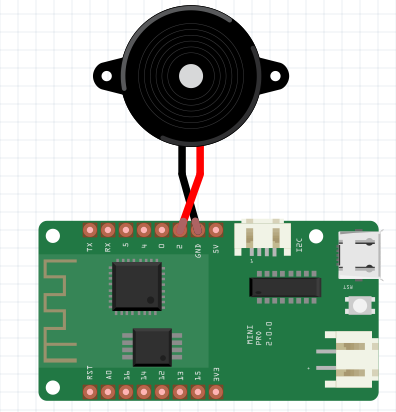
### Anmerkungen

* In PlatformIO muss „WEMOS D1 mini pro“ oder „WEMOSE D1 mini light“ ausgewählt werden („WEMOS D1 MINI ESP32“ ist inkorrekt, Mikrocontroller kann dann nicht erkannt werden)
* Im Code kann per DHT dht(Dht\_type, Dht\_PIN) ein DHT Objekt erzeugt werden
* Dht\_type in unserem Fall: DHT **11**
* Dht\_PIN: in unserem Fall: PIN **D4** (kann **nicht** geändert werden: <http://www.esp8266learning.com/wemos-dh11-shield-example.php>)

### Aufgabe 1.2

* Passwort, SSID wurden in einer extra Datei „secret.h“ angelegt (außerdem habe ich die IP-Adresse vor dem Pushen auf Github gelöscht)
* Code für Client und Server wurde größtenteils aus den gegebenen Codebeispielen hergenommen
* Wichtig: für den WEMOS D1 Mini muss *<ESP8266WiFi.h>* inkludiert werden
* Temperatur und Luftfeuchtigkeit werden alle fünf Sekunden von dem Sensor gemessen und dann per Wifi an den PC gesendet
* Gesendet werden die Daten per print() Methode des Clients, welcher man einen String/ die Message übergeben kann. Der String, den ich als Message übergebe, besteht dabei aus dem Float-Wert der Temperatur (in °C) und dem Float-Wert der Luftfeuchtigkeit (in %), wobei beide durch ein Leerzeichen getrennt sind, um sie auf Server-Seite wieder korrekt auslesen zu können.
* Auf der Server-Seite empfangen wir die Daten als String. Indem ich einfach die Daten am Leerzeichen teile, erhalte ich meine zwei einzelnen Messwerte
* Zusätzlich habe ich die **win10toast** Python-Library installiert, um mir Toast-Messages am PC ausgeben zu lassen, wenn …
* … Luftfeuchtigkeit zu hoch ist (über 60%): die Meldung informiert mich dann, dass mal wieder gelüftet werden sollte
* … Temperatur zu hoch ist (über 25°C): die Meldung informiert mich dann, dass ich bei der Hitze genügend trinken sollte
* Win10toast Library: <https://pypi.org/project/win10toast/>

### Aufgabe 2: WiFi Türklingel

* Der WEMOS, bei dem sich der Button für die Klingel befindet, dient als Client, der WEMOS, der mit dem Lautsprecher verbunden ist, als Server
* Einen WEMOS (der mit Lautsprecher) versorge ich mit Strom über USB an meinem Laptop, den anderen mit dem SBC-POW-BB Breadboard Power-supplier (Wichtig: WEMOS mit 3.3V versorgen!)
* Die zwei Schaltungen (siehe Bilder weiter unten) waren schnell und einfach umgesetzt: (links der Wemos mit Button, rechts der Wemos mit Lautsprecher)
* Anmerkung: in den Bildern unten wurde ein WEMOS d1 mini Pro verwendet (da ich den WEMOS d1 mini light nicht finden konnte). Die digitalen PINS sind hier etwas anders beschriftet.
* Code für die Wifi Connection größtenteils aus dem gegebenen Sample Code übernommen
* Passwort, SSID und IP sind geheim und in die extra Datei „secret.h“ ausgelagert, welche nicht mit auf Github gepushed wird
* Ich habe die Klingel im Code so implementiert, dass immer ein gleich langer Ton (per tone(PIN, frequency, duration) Methode) abgespielt wird, unabhängig wie lange die Klingel gedrückt wird. D.h. drückt jemand auf die Klingel und hält diese gedrückt, so ertönt trotzdem nur einmal das Klingelgeräusch.

Aufgabe 3: WiFi Telegraph

* Schaltung für beide WEMOS identisch (siehe Bilder unten)
* Auch der Code für beide ist nahezu identisch. Einziger Unterschied: ein WEMOS dient als Client (verbindet sich mit Server), der andere als Server (initiiert Verbindung, wartet auf Clients, die sich verbinden). Welcher WEMOS welche Funktion übernimmt ist egal, da beide sowohl Daten senden, als auch empfangen können.
* Es werden immer nur Daten an den jeweils anderen WEMOS geschickt, wenn sich der Status des Buttons ändert. D.h. sobald man auf den Button klickt wird einmal ein HIGH Signal an den anderen WEMOS geschickt. Hält man den Button weiterhin gedrückt, wird nichts gesendet. Lässt man dann den Button los, wird einmalig LOW gesendet. Dann wird wieder solange nichts gesendet, bis der Button wieder gedrückt werden würde.
* Auf diese Weise kann verhindert werden, dass in jeder Loop Iteration Daten gesendet werden, die den Zustand der LED aber eigentlich nicht verändern

