Der neue D1 mit ESP8266mod-12F im Uno-Format

Durch meine Mitarbeit am Blog und bei den eBooks komme ich sehr schnell in Berührung mit den neuen Produkten im Sortiment. Gerade wenn bei uns oder im Internet noch keine oder wenig Information zu finden ist, wird meine Neugier geweckt, so bei dem D1, einem ESP8266mod-12F in der Bauform des Arduino Uno. Was haben die beiden Boards gemeinsam? Sind sie sogar Pin-kompatibel? Oder gibt es Unterschiede, die man beachten muss?

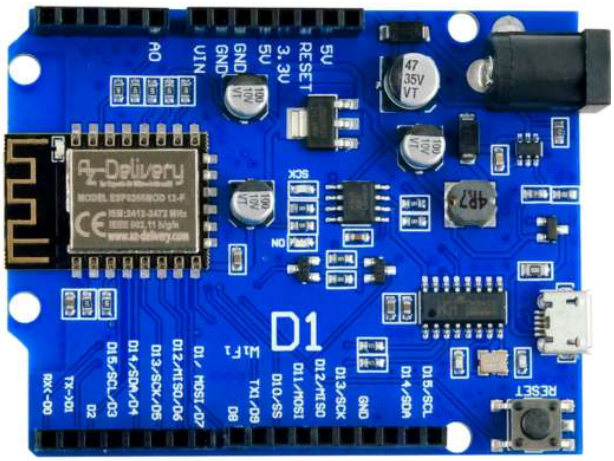

Bild 1: D1 mit ESP8266mod -12F in Baugröße Uno

Bei genauer Betrachtung der Beschriftung wird schnell klar, dass die Pinbelegung unterschiedlich ist und es gibt den **Warnhinweis**, dass die GPIOs nur 3,3V vertragen, im Gegensatz zum Arduino Uno. Außer der Beschriftung auf der Vorderseite gibt es eine sehr zielführende Beschriftung mit den GPIO-Nummern auf der Rückseite.

Meine ersten Versuche dienen der Überprüfung der GPIO-Bezeichnungen mit Hilfe des Programms Blink. Da ich die ESPs schon mit dem Bordverwalter in der Arduino IDE integriert hatte, suche ich unter ESP8266 den D1. Die erste Hürde genommen. Wer die ESPs noch nicht in die Arduino IDE integriert hat, sei auf die eBooks zum ESP8266 und ESP32 verwiesen.

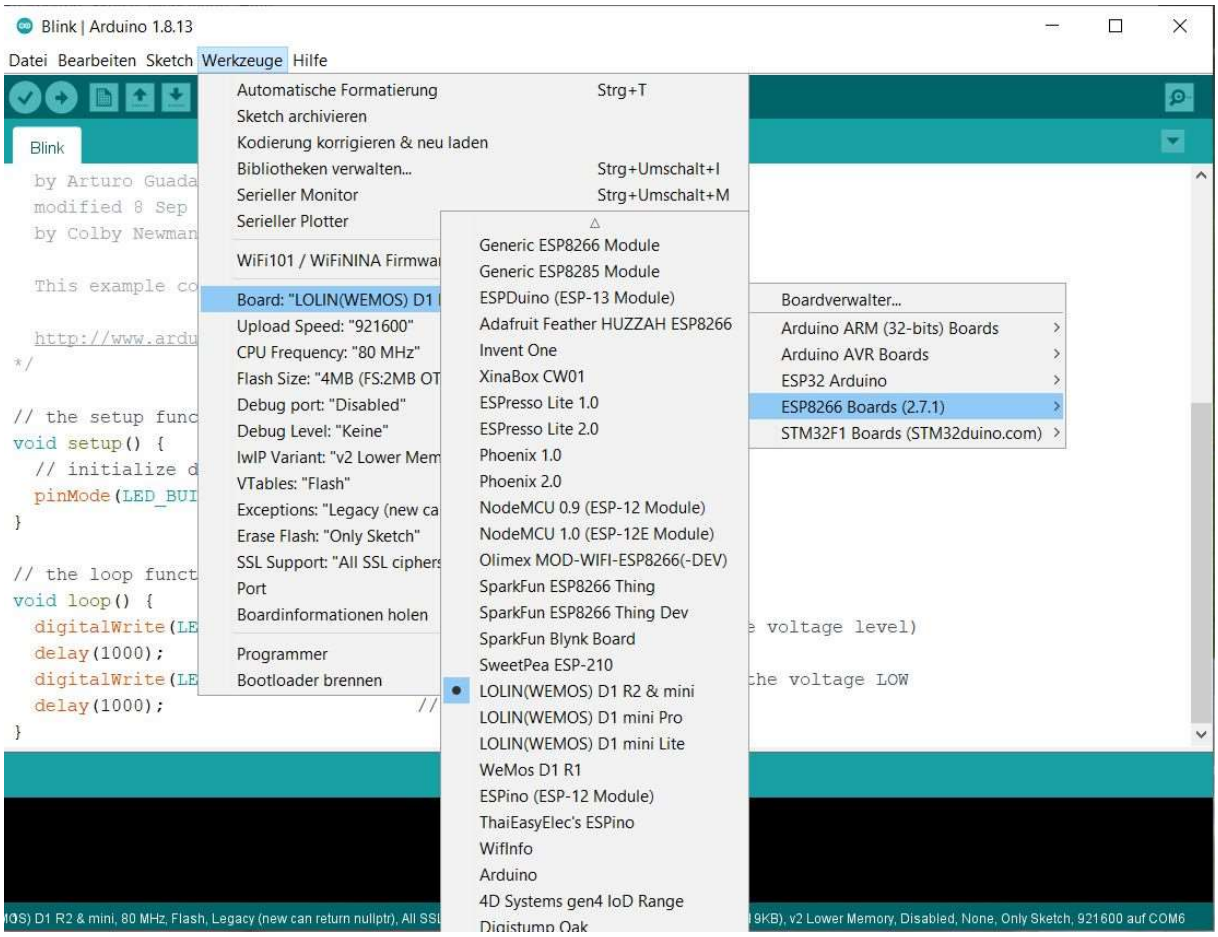

Bild 2: Bordverwalter der Arduino

Alle Bezeichnungen werden bestätigt, und es gibt die berühmt-berüchtigte LED_BUILTIN auf GPIO2 (beim Uno bekanntlich 13, bei anderen ESPs 0 oder 1, oder nicht bekannt). Darüber hinaus gibt es in der Reihe mit der LED ON noch eine blaue LED, die mit GPIO14/SCK verbunden ist. Wie auf dem Bild deutlich zu erkennen ist, gibt es nur einen analogen Eingang A0.

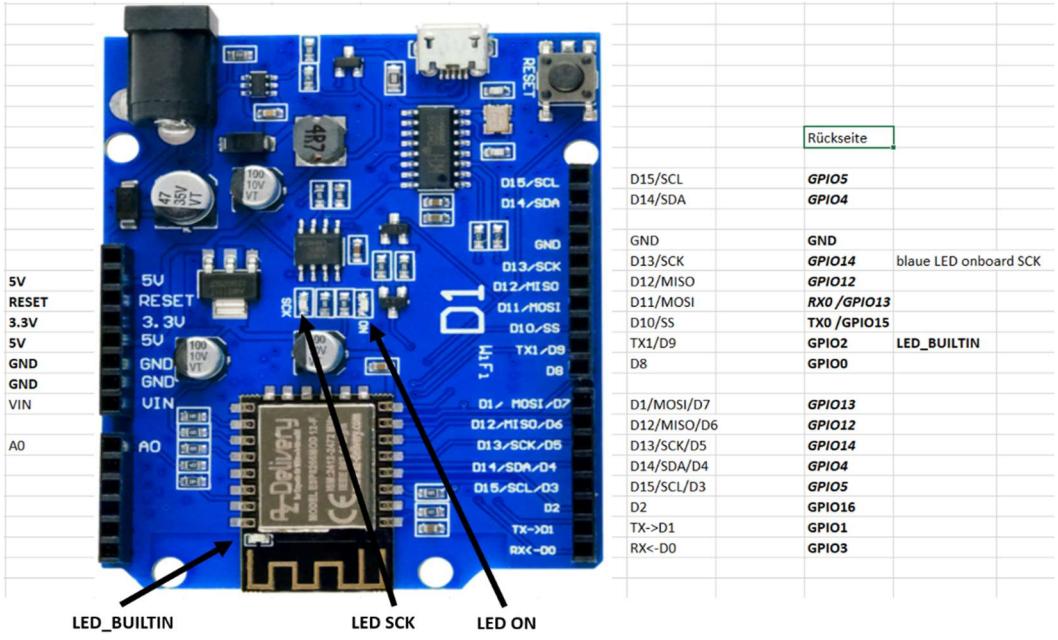

Bild 3: Pinout und LEDs

Sensoren und Schnittstellen

Als nächstes probiere ich die Programme für die Sensoren und Schnittstellen aus und beginne mit dem DHT22, einem Sensor für Temperatur und Relative Luftfeuchtigkeit. Dieser benötigt außer der Spannungsversorgung (3,3 V oder 5V) einen Pin für die Datenleitung. Der Daten-Pin wird über einen 10kOhm-Pullup-Widerstand mit VCC verbunden. Deshalb entscheide ich mich für VCC=3,3V. Wie im Arduino-Sketch vorgegeben definiere ich den Daten-Pin mit *#define DHTPIN 2*; aber Achtung: GPIO2 liegt anders als beim Uno, hier bei TX1/D9.

Die Anzeige erfolgt außer auf dem Serial Monitor auch auf einem LCD1602 mit I2C-Adapter. Das Display versorge ich wegen des deutlich besseren Kontrasts mit 5V. Der I2C-Anschluss ist auf dem Board doppelt ausgeführt, beide beschriftet mit D14/SDA und D15/SCL. Auch diese Pins liegen anders als beim Uno (A4 und A5).

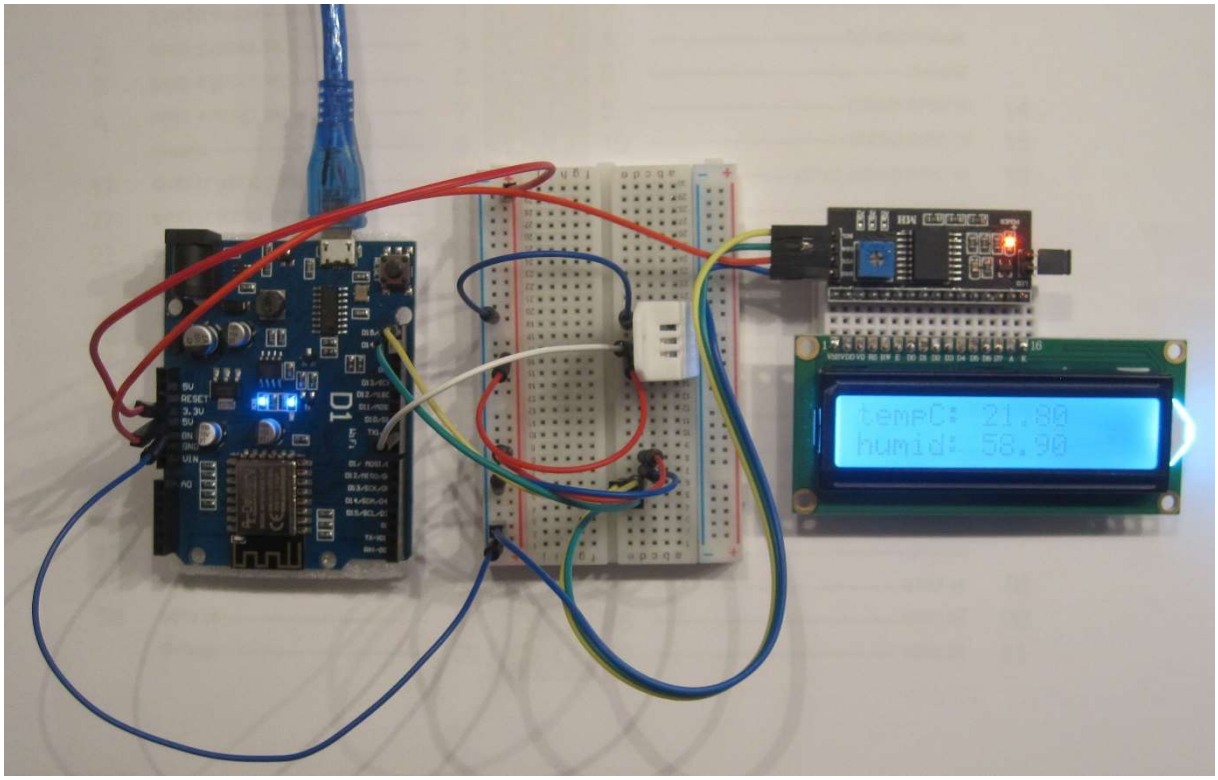

Bild 4: Versuchsaufbau mit DHT22 und LCD1602 mit I2C-Adapter

Der Sketch mit dem DHT22 funktioniert ohne Beanstandung, also weiter geht's mit dem BME280 von Bosch, einem kombinierten Temperatur-, Luftfeuchtigkeits- und Luftdrucksensor. Davon habe ich auch einen mit I2C- und SPI-Schnittstelle. Zunächst probiere ich die Seite mit der I2C-Schnittstelle parallel zum LCD1602 mit I2C-Adapter.

Bei dem Sketch aus dem Internet gab es zunächst zwei Schwierigkeiten, die jedoch schnell behoben waren. Erstens: Die Methode *.init()* zur Initialisierung des Objektes bme wurde nicht erkannt. Das hatte ich vorher schon einmal erlebt und stattdessen mit Erfolg *bme.begin()* verwendet. Und zweitens ist die Programmbibliothek von Adafruit auf die I2C-Adresse 0x77 voreingestellt. Wegen der Adresse 0x76 meines BME280 lautet die Zeile im Code deshalb *bme.begin(0x76).*

Bemerkenswert finde ich auch den Programmiertrick in dem Demo-Sketch. Die Initialisierung ist Teil des Sensor-Tests:

if (!bme.begin(0x76)) {

```
  Serial.println(F("Could not find a valid BME280 sensor, check wiring!"));
  while (1) delay(10);
}
```

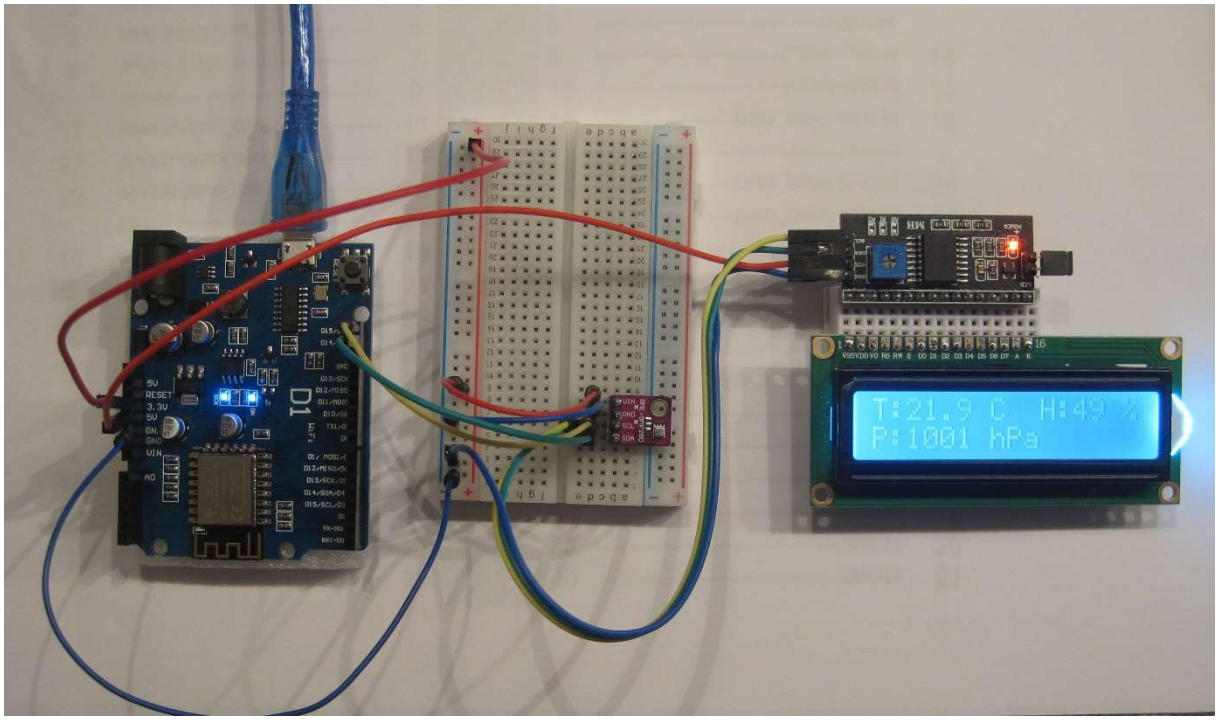

Bild 5: Versuchsaufbau mit BME280 und LCD1602 mit I2C-Adapter

Auch dieser Versuch verlief nach den anfänglichen Schwierigkeiten ohne weitere Probleme.

Auf der anderen Seite eines meiner BME280 (nicht aus dem AZ-Delivery Sortiment) befindet sich eine zweite Steckerleiste auf der gegenüberliegenden Seite mit der SPI-Schnittstelle. Dafür benutze ich das Demo-Programm bme280test von Lady Ada (Limor Fried). Es muss jedoch eine andere Zeile auskommentiert werden und für den D1 müssen andere Pin-Nummern definiert werden als für den Uno:

Also folgende Veränderungen:

```
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#define BME_SCK 14                  //Original 13
#define BME_MISO 12
#define BME_MOSI 13                 //Original 11
#define BME_CS 0                    //Original 10

//Adafruit_BME280 bme;       // die Zeile für I2C wird auskommentiert, stattdessen:
Adafruit_BME280 bme(BME_CS, BME_MOSI, BME_MISO, BME_SCK);    // software SPI
```

Der Pin D10/SS ist nicht geeignet, weil er den upload des Programms verhindert und nicht funktioniert. Gut geeignet für CS ist Pin D8 = GPIO0, andere Pins sind möglich.
Und **Wichtig**: Die Instanziierung *Adafruit_BME280 bme* für I2C muss auskommentiert werden // und stattdessen müssen bei software SPI die Kommentarzeichen // entfernt werden.

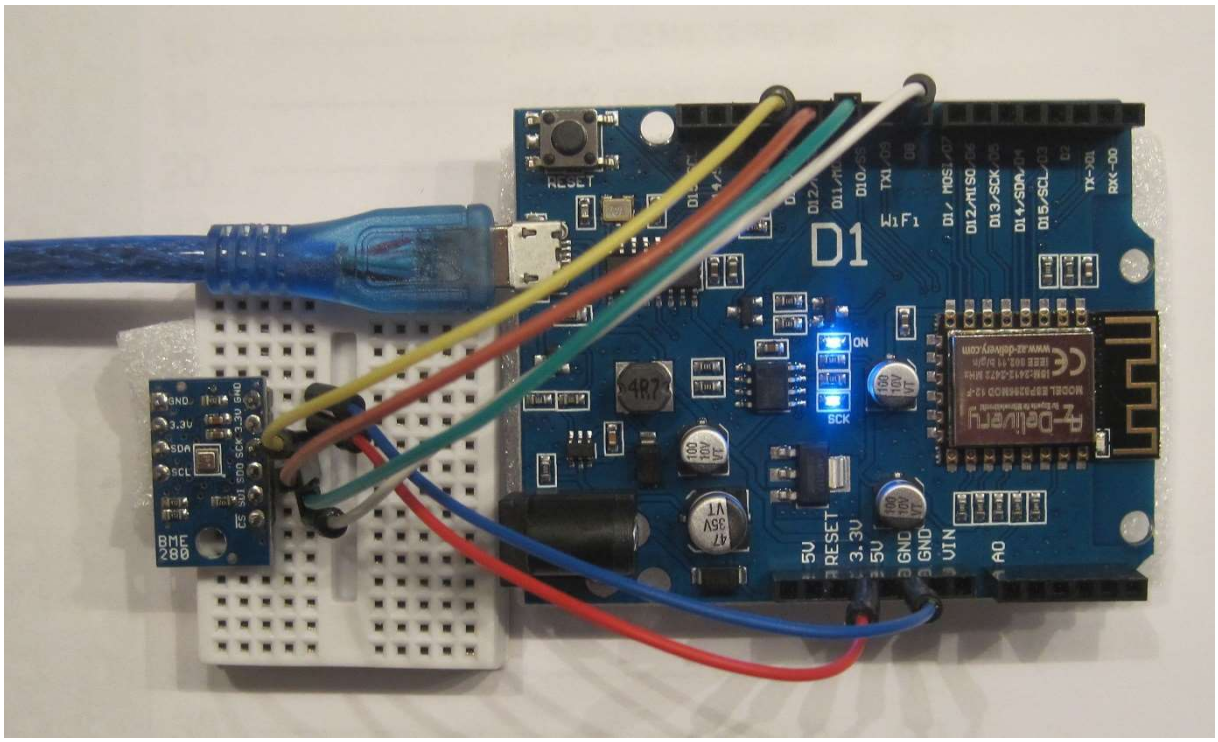

Bild 6: Versuchsaufbau mit BME280 an der SPI-Schnittstelle

Zwischenfazit:

Die auf der Rückseite angezeigten GPIOs sowie die Schnittstellen I2C und SPI funktionieren ohne Beanstandung.

Was zeichnet den D1 gegenüber dem Uno aus? Die bisher gezeigten Sketche waren ja mit kleinen Anpassungen aus der Beispielsammlung des Arduino Uno entnommen. Aber wenn mehr Rechenleistung und Speicher benötigt werden, ist der D1 sicherlich eine Empfehlung. Mich reizen die ESPs vor allem wegen der eingebauten Wifi-Schnittstelle, gut zu erkennen an den PCB-Antennen.

Also kurzerhand eine Programmerweiterung für den Versuch mit dem DHT22, um die Messdaten auch im heimischen WLAN bereit zu stellen. Bitte denken Sie daran, dass Sie hier Ihre persönliche SSID und Passwort eingeben müssen. Und dann klappt auch dieser Teil ohne Beanstandung.

Jetzt fehlt nur noch die Frage, welche Arduino Uno Shields auch beim D1 passen:

Damit sind wir wieder beim Thema Kompatibilität der Pins. Wir hatten ja gesehen, dass diese leider nicht gegeben ist. Also kommen nur Shields in Betracht, bei denen diese Unterschiede nicht ins Gewicht fallen, z.B. beim **Prototyping Shield**:

Da kann nicht viel schief gehen. Die Pins sind eins zu eins durchgeschleift. Man muss nur die Beschriftung ignorieren. Also das Pinout-Diagramm vom D1 daneben legen.

Wegen der eingebauten Wifi-Schnittstelle habe ich die **Wifi- und LAN Shields** nicht ausprobiert.

Bei dem **Display Shield** sind so viel bauliche Veränderungen vorzunehmen (siehe Gerald Lechners Blog), dass ich das nicht ausprobiert habe und auch nicht zuraten kann.

Aber beim **LCD1602 Keypad Shield** ist für die Benutzung der fünf Taster nur eine kleine bauliche Veränderung vorzunehmen: Einlöten eines 3,9 kOhm-Widerstands zwischen Eingang A0 und GND (siehe Bild). Das LCD funktioniert übrigens ohne I2C-Adapter. Deshalb ist eine andere Programmbibliothek zu inkludieren. Die verwendeten GPIOs sind dem Bild zu entnehmen:

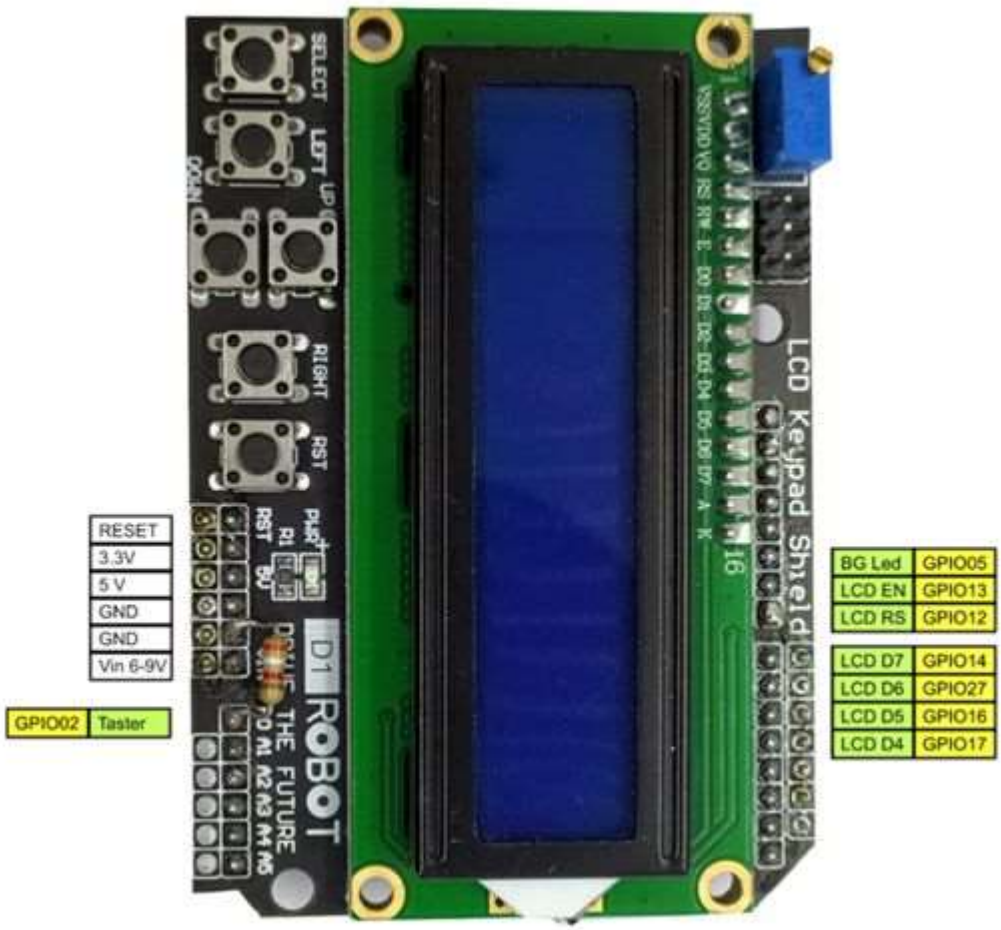

Bild 7: LCD Keypad Shield mit zus. 3,9 kOhm Widerstand

Alle Taster liegen über einen Spannungsteiler (s.u.) am analogen Eingang A0.

Die Auswertung der Tasten erfordert eine besondere Beachtung.
Der Spannungsteiler ist so ausgelegt, dass bei den anliegenden 5V der Unterschied zwischen den einzelnen Spannungen bei etwa 1V liegt.
Deshalb: Spannungsteiler mit 3,9 kOhm Widerstand zw. A0 und GND ergänzen (siehe Bild oben).

Wenn wir aber den 3900 Ohm Schutzwiderstand eingebaut haben, damit in keinem Fall mehr als 3.3 V an A0 anliegen, reduziert sich dieser Unterschied.

	Ohne Parallelwiderstand	3900 Ohm
Kein Taster	5,00 V	3,30 V
SELECT	3,60	2,60
LEFT	2,50	1,97
DOWN	1,60	1,40
UP	0,70	0,66

RIGHT	0,00	0,00

Diese Spannungswerte sind ungefähre Werte, die durch den 10bit-ADC (=Analog Digital Converter) auf Werte zwischen 0 und 1023 umgewandelt werden. Im Programm wird dann abgefragt, ob der Messwert z.B. zwischen zwei bestimmten Zahlen liegt, und das entspricht dann einer bestimmten Taste. Diese Zahlenwerte müssen ggf. für den D1 angepasst werden.

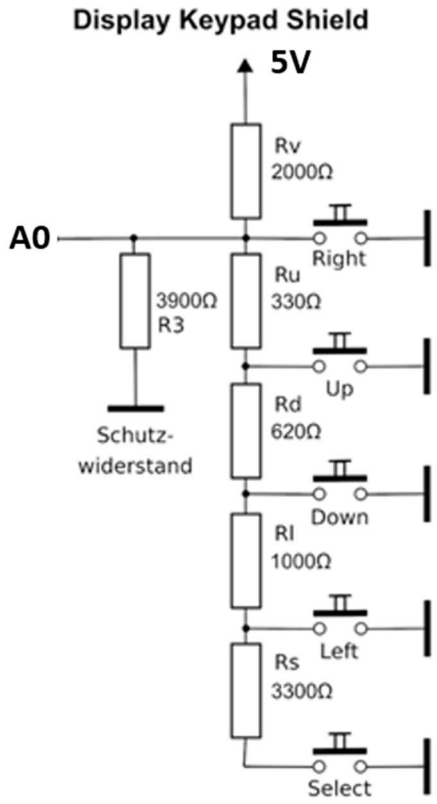

Bild 8: Spannungsteiler für die fünf Taster am Eingang A0

Fazit:

Alles in allem ist der D1 mit dem ESP8266mod-12F ein gelungenes Board, das die Vorteile des Arduino Uno in Bezug auf Bedienung und Spannungsversorgung mit den Vorteilen der ESP (mehr Speicher, Wifi) verbindet.

P.S. Es herrscht zum Zeitpunkt des Schreibens Unklarheit, ob der Name des neuen D1 mit ESP8266mod-12F nun D1 R1 oder D1 R2 ist. In der IDE war es egal, was ich vorgewählt habe. Und in der Beschriftung sehe ich keine weiteren Hinweise. Entscheidend ist für mich der verwendete Micro Controller, und der heißt ESP8266mod-12F.