

Willkommen!

Und herzlichen Dank für den Kauf unseres AZ-Delivery ESP8266 Mikrocontroller mit integrierten 0.91" OLED Display. Auf den folgenden Seiten gehen wir mit dir gemeinsam die ersten Programmierschritte durch.

Viel Spaß!



Dieses Board mit einem ESP8266 Chip und integriertem Ladeinterface, für ein Akkupack, kann unter NodeMCU und Arduino programmiert werden.

Wir beziehen uns in diesem eBook auf die Programmierung in Arduino.

Auf dem Board sind 32MByte Flash Speicher, WLAN und 12 I/O-Pins (i²c, SPI, PWM und Analog Input) vorhanden. Das Display mit 0,91" hat eine Auflösung von 128 x 32 Pixel.

Programmieren des ESP8266 mit OLED

Dieser ESP8622 hat im Vergleich zum ESP8266-01 mehr Flashspeicher und mehr GPIO Anschlüsse zur freien Verfügung.

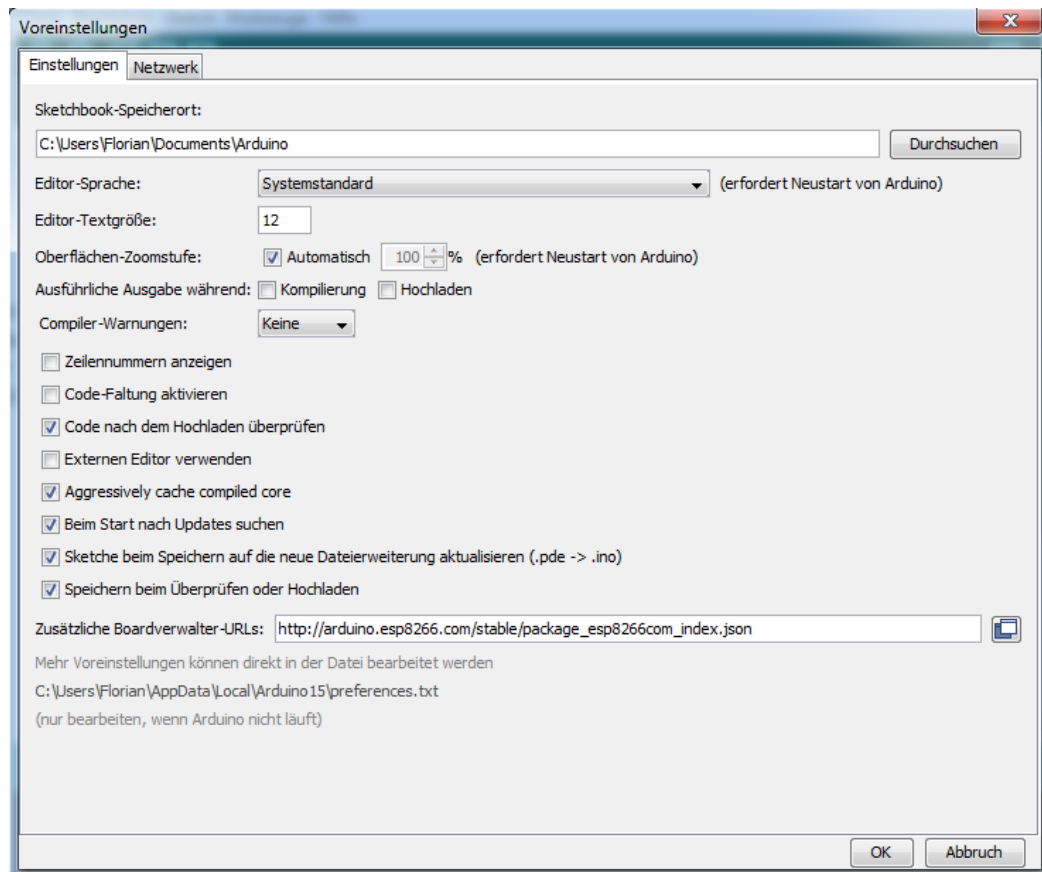
Vorbereiten der Software:


Die Arduino Software sehen wir in diesem Schritt als Installiert an, sollte diese bei dir noch fehlen, so kannst du diese unter <https://www.arduino.cc/en/Main/Software#> herunterladen und auf deinen PC installieren.

Außerdem die Treiber für den CP210x solltest du auch schon installiert haben, wenn nicht, wird dieser bei der Arduino-Software mitgeliefert.

Nachdem alle Grundvoraussetzungen getätigt wurden, starten wir nun mit der Einrichtung der Software. Die Arduino Software benötigt zunächst einmal alle Informationen zum ESP8266, dies können wir tun, indem wir unter dem „Voreinstellungen“ > „Zusätzliche Boardverwalter-URLs“ folgende Adresse eingeben:

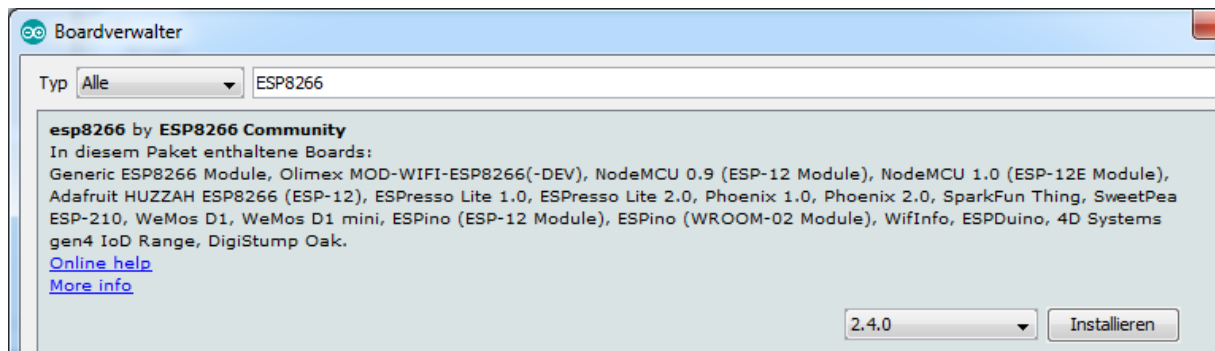
http://arduino.esp8266.com/stable/package_esp8266com_index.json



Evtl. wenn du schon einen Link eingetragen hast, auf den Button  klicken und in dem Fenster eine neue Zeile hinzufügen.

Bestätigen wir die Eingabe mit „OK“.

Ist das erledigt, gehen wir auf „Werkzeuge“ > „Board“ > „Boardverwalter“ und installieren die ESP8266 Bibliothek. In dem Boardverwalter geben wir in der Suchleiste oben rechts „ESP8266“ ein, es wird das Paket von ESP8266 Community angezeigt. Dieses wählen wir aus und klicken auf Installieren.



Nach erfolgter Installation steht neben dem Paket nun **INSTALLED**.

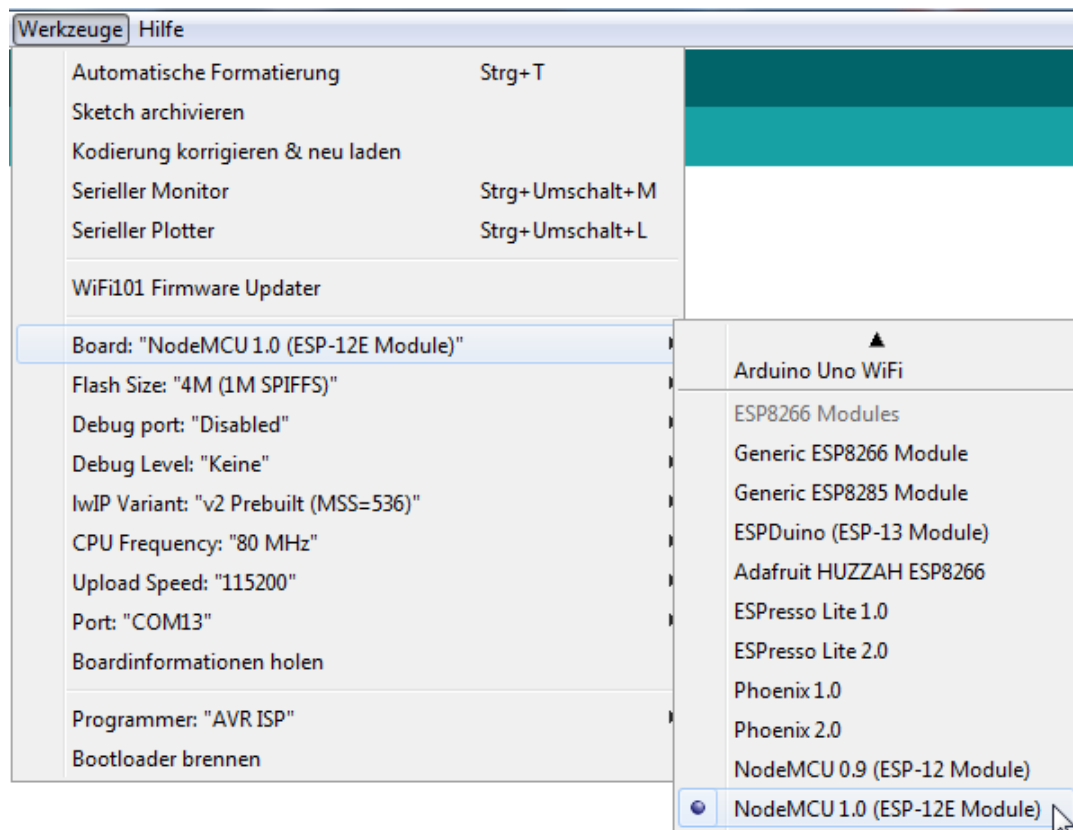


Als nächsten Schritt müssen wir das richtige Board auswählen:

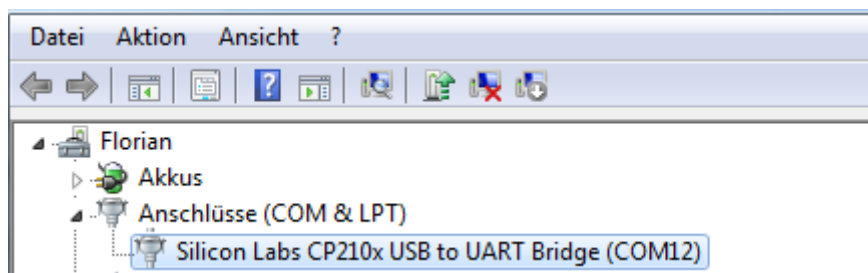
Unter Werkzeuge >

Board: „Generic ESP8266 Module“

Port: „COMxx“ (hier dein Port des Serial Adapters)



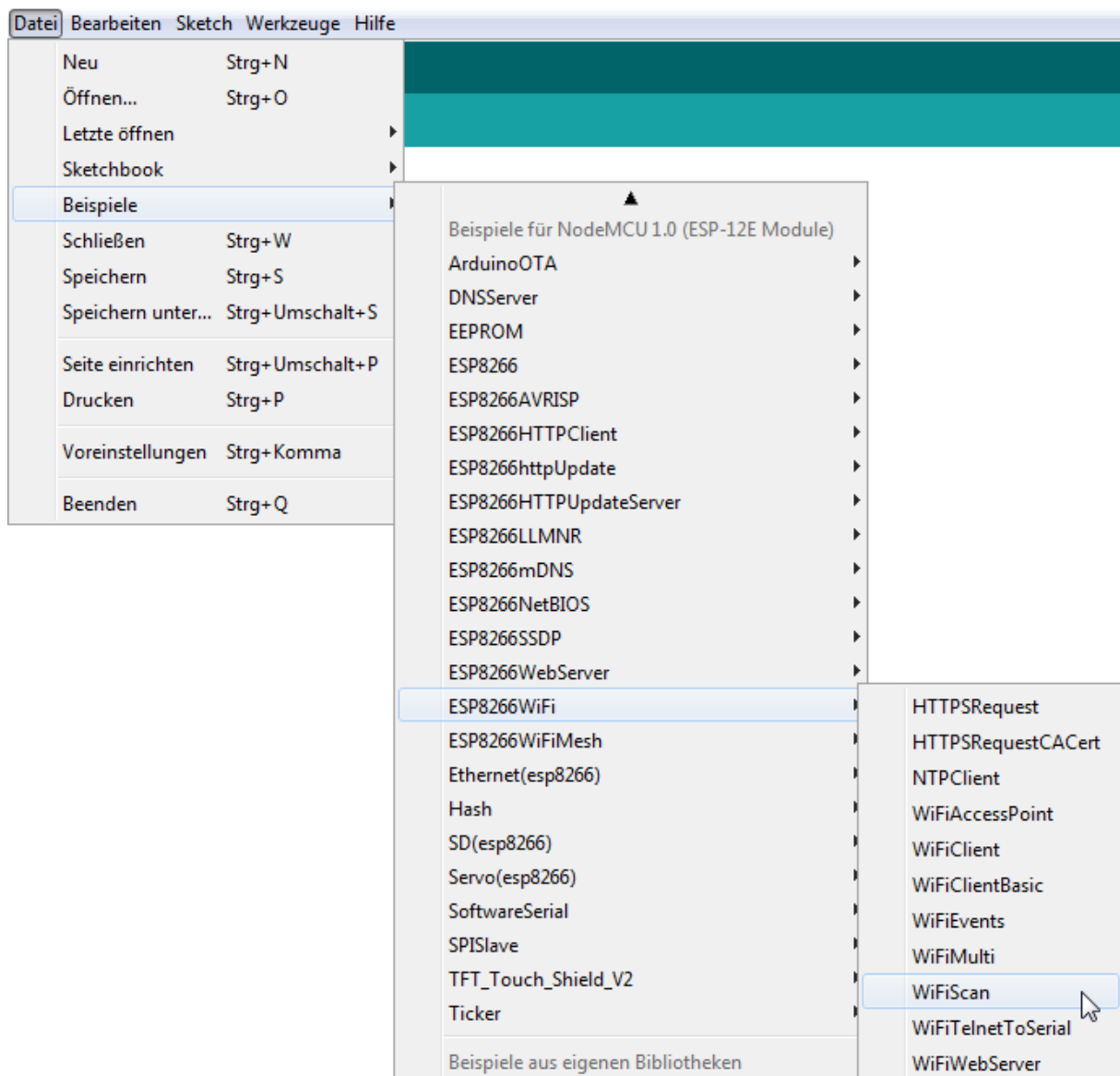
Der COMPort kann im Gerätemanager abgerufen und auch geändert werden:




Der Arduino Code:

Nachdem nun die Verdrahtung erledigt wurde, schreiben wir unseren ersten Code. Lassen wir alle WLAN-Netze in Reichweite anzeigen.

Wähle dazu unter Datei > Beispiele > ESP8266Wifi > WifiScan aus.



Nachdem der Code angezeigt wird, klicken wir oben auf  und Verifizieren unser Programm:

Wenn alles stimmt und unser Programm keine Fehler enthält

```
Kompilieren abgeschlossen.  
Der Sketch verwendet 252567 Bytes (24%) des Programmspeicherplatzes. Das Maximum sind 1044464 Bytes.  
Globale Variablen verwenden 33216 Bytes (40%) des dynamischen Speichers, 48704 Bytes für lokale Variablen verbleiben. Das Maximum si
```

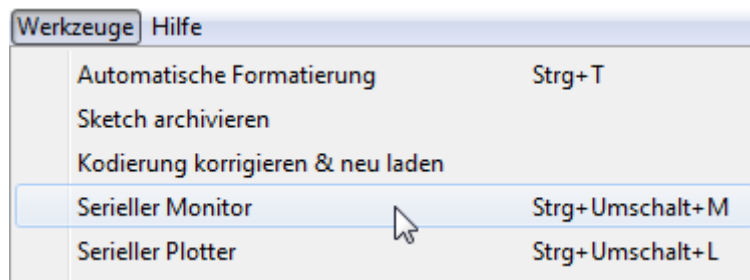
können wir es auf den ESP8622 hochladen. Dazu klicken wir oben auf 

Nach kurzer Zeit wird das Programm auf den Chip geladen:

```
Hochladen...  
Globale Variablen verwenden 33216 Bytes (40%) des dynamischen Speichers, 48704 Bytes für lokale Variablen verbleiben. Das Maximum si  
Uploading 256720 bytes from C:\Users\Florian\AppData\Local\Temp\arduino_build_493968\WiFiScan.ino.bin to flash at 0x00000000  
.....
```

Sobald dann 100% erreicht wurden, ist das Programm vollständig übertragen. Öffnen wir den Serial Monitor in der Arduino Software:

Werkzeuge > Serial Monitor



Es wird alle 5 Sekunden ein neuer Scan gestartet und das Ergebnis über die Serielle Schnittstelle ausgegeben.

Zu Beginn eines jeden Scans wird

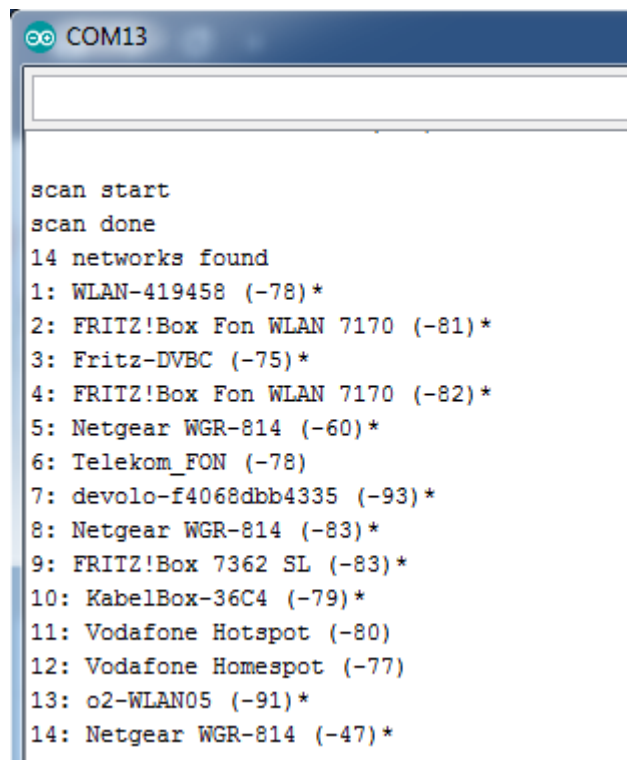
scan start

angezeigt, und mit

scan done

beendet.

Zuletzt werden alle erreichbaren WLAN-Netze aufgelistet. Dies sieht dann in etwa so aus wie es rechts zu sehen ist.

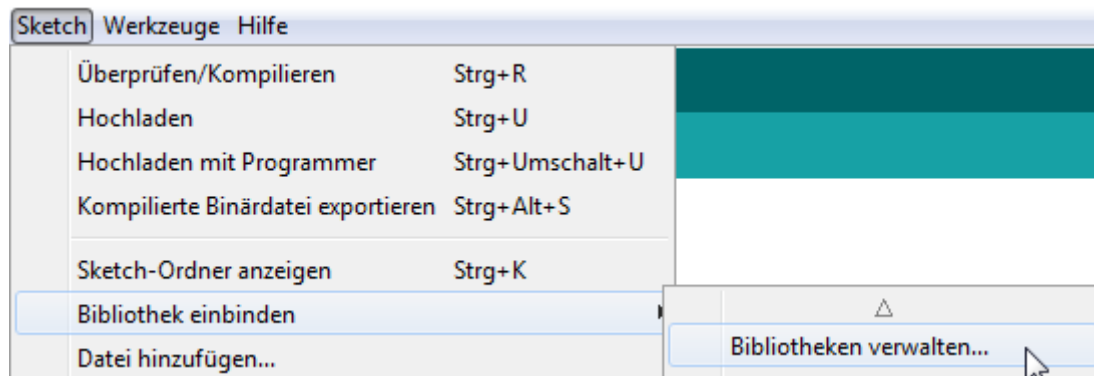


```
scan start  
scan done  
14 networks found  
1: WLAN-419458 (-78)*  
2: FRITZ!Box Fon WLAN 7170 (-81)*  
3: Fritz-DVBC (-75)*  
4: FRITZ!Box Fon WLAN 7170 (-82)*  
5: Netgear WGR-814 (-60)*  
6: Telekom_FON (-78)  
7: devolo-f4068dbb4335 (-93)*  
8: Netgear WGR-814 (-83)*  
9: FRITZ!Box 7362 SL (-83)*  
10: KabelBox-36C4 (-79)*  
11: Vodafone Hotspot (-80)  
12: Vodafone Homespot (-77)  
13: o2-WLAN05 (-91)*  
14: Netgear WGR-814 (-47)*
```

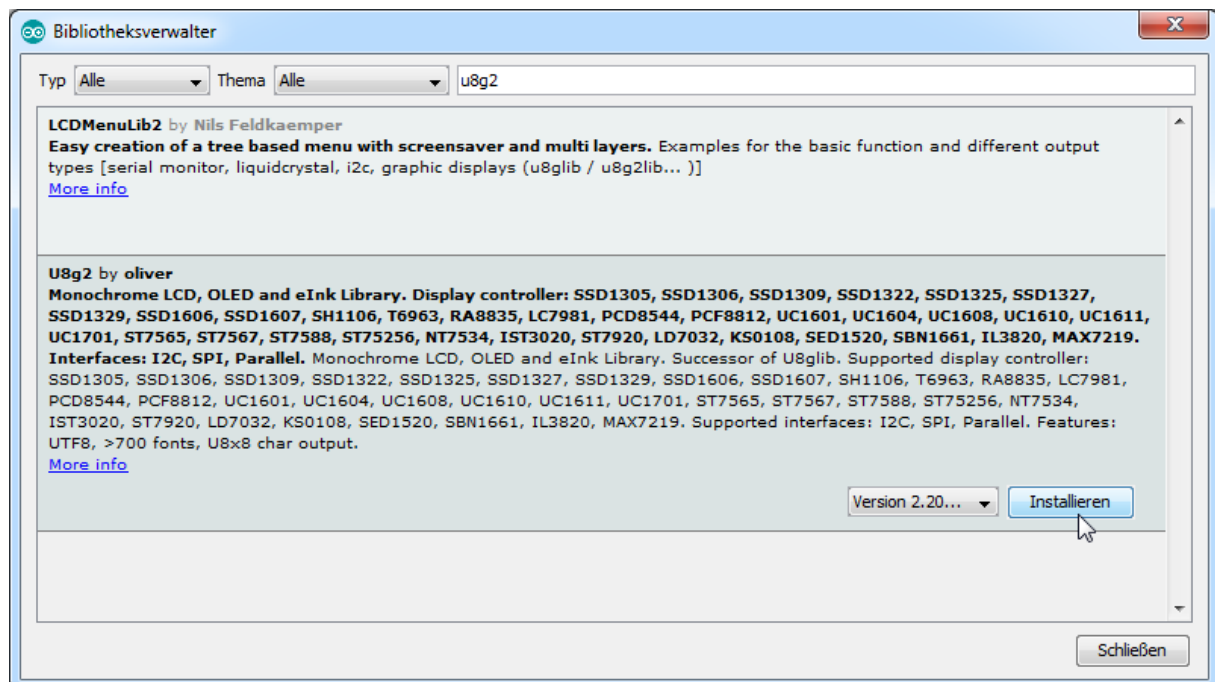
Ansteuern des OLED-Displays:

Für die Ansteuerung des Displays benötigen wir noch die entsprechenden Bibliotheken (Informationen) in der Arduino Software.

Starten wir unter Sketch > Bibliothek einbinden > Bibliotheken verwalten ...

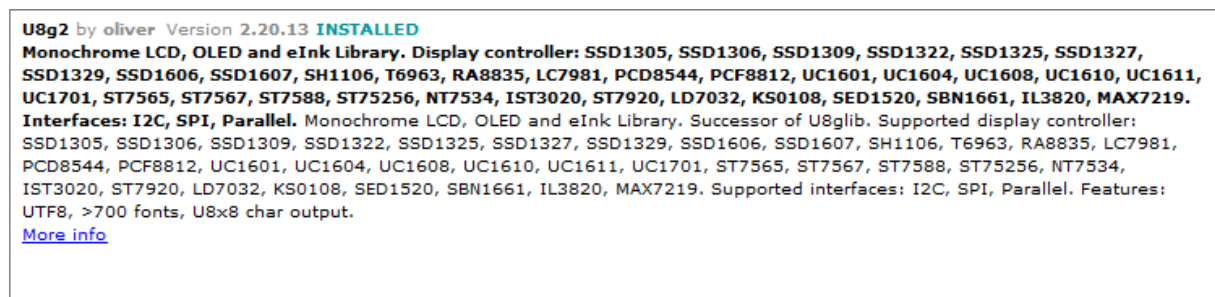


den Bibliotheksverwalter und suchen dort nach „u8g2“



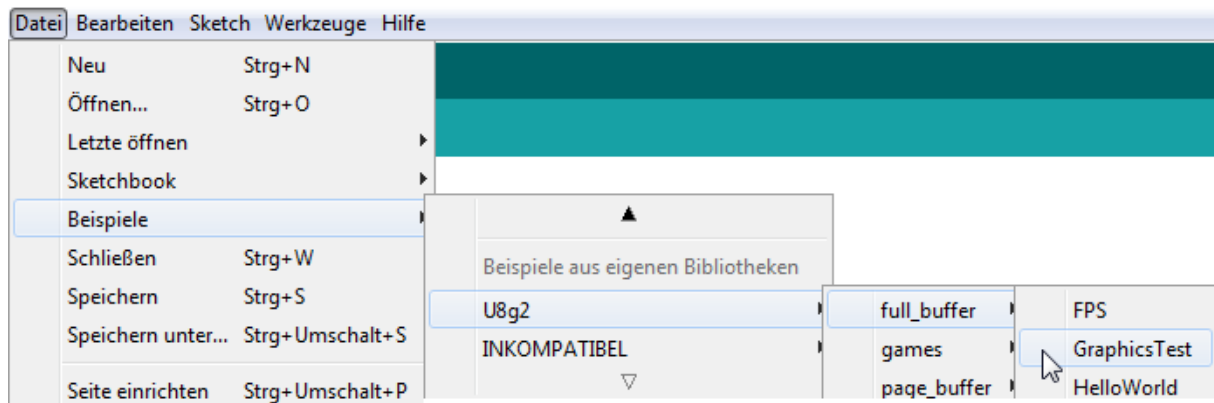
und klicken rechts unten auf Installieren, nachdem das Paket angewählt wurde.

Nach ein paar Sekunden Wartezeit erscheint „INSTALLED“



Nun Schließen wir das Fenster und können mit dem Programmieren loslegen.

Wählen wir unter Beispiele > U8g2 > full_buffer > GraphicsTest aus:



Es wird nun ein langer Code geöffnet, in den ersten Zeilen sind sehr viele Displaytypen eingetragen, diese sind aber mit den „//“ am Zeilenanfang auskommentiert. Für unser Display müssen wir nun diese Zeile suchen und aktivieren, indem wir die // am Anfang der Zeile entfernen:

```
U8G2_SSD1306_128X32_UNIVISION_F_HW_I2C u8g2(U8G2_R0, /* reset= */  
U8X8_PIN_NONE); // Adafruit ESP8266/32u4/ARM Boards + FeatherWing OLED
```

Nach dem Übertragen  zeigt das Display nun Demotexte und Bilder an.

Basierend auf dieser Demonstration können wir auch einen Lauftext programmieren.

Für eine flüssige Wiedergabe ändern wir die CPU-Frequenz auf 160MHz und den SPI-Speicher auf 3MB (3M SPIFFS)

```
Board: "NodeMCU 1.0 (ESP-12E Module)"  
Flash Size: "4M (3M SPIFFS)"  
Debug port: "Disabled"  
Debug Level: "Keine"  
lwIP Variant: "v2 Prebuilt (MSS=536)"  
CPU Frequency: "160 MHz"  
Upload Speed: "115200"  
Port: "COM13"
```

Hinweis möchte man einen längeren Lauftext machen, muss in der u8g2.h Datei die 16 Bit Unterstützung aktiviert werden. Die Datei findet ihr in euerem Arduino Verzeichnis unter:

Arduino\libraries\arduino_168079\src\clib\u8g2.h

In der Zeile 72 steht: `//#define U8G2_16BIT`

Dies wird geändert auf: `#define U8G2_16BIT`

Anschließend die Datei speichern und den Code neu Übertragen.

Hier folgt der Code:

```
#include <U8g2lib.h>
U8G2_SSD1306_128X32_UNIVISION_F_SW_I2C u8g2(U8G2_R0, /* clock=*/ 5, /*
data=*/ 4, /* reset=*/ 16);
//U8G2_SSD1306_128X32_UNIVISION_F_HW_I2C u8g2(U8G2_R0, /* reset=*/
U8X8_PIN_NONE);
u8g2_uint_t offset;
u8g2_uint_t width;
const char *text = "Florian ";
void setup(void) {
    u8g2.begin();
    u8g2.setFont(u8g2_font_logisoso32_tf);
    width = u8g2.getUTF8Width(text);
    u8g2.setFontMode(0);
}

void loop(void) {

    u8g2_uint_t x;
    u8g2.firstPage();
    do {
        x = offset;
        u8g2.setFont(u8g2_font_logisoso32_tf);
        do {
            u8g2.drawUTF8(x, 32, text);
            x += width;
        } while ( x < u8g2.getDisplayWidth() );
        u8g2.setFont(u8g2_font_logisoso32_tf);
        u8g2.setCursor(0, 64);
        u8g2.print(width);
    } while ( u8g2.nextPage() );
    offset -= 1;
    if ( (u8g2_uint_t)offset < (u8g2_uint_t) - width ){
        offset = 0;
        delay(1000);
        u8g2.clearBuffer();
    }
}
```

Du hast es geschafft, du kannst nun deine Projekte mit ESP8266-OLED programmieren und verwirklichen!

Ab jetzt heißt es Experimentieren.

Und für mehr Hardware sorgt natürlich dein Online-Shop auf:

<https://az-delivery.de>

Viel Spaß!
Impressum

<https://az-delivery.de/pages/about-us>