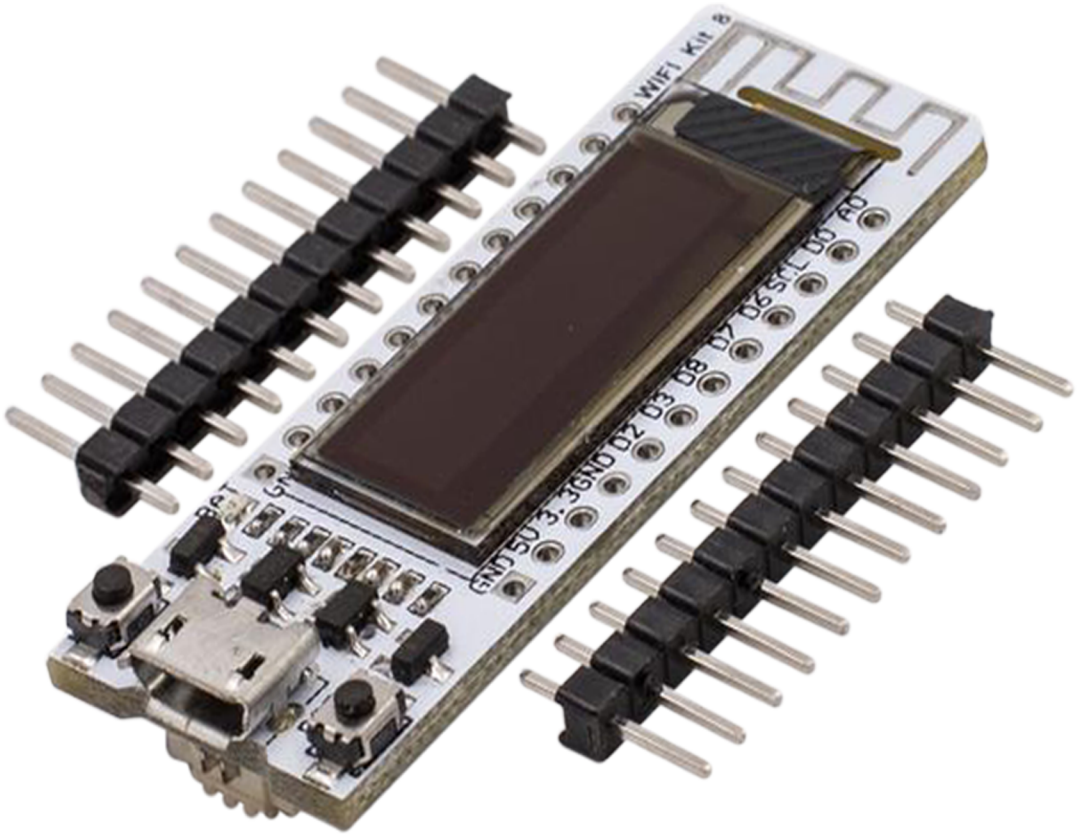
ESP8266 Mikrocontroller mit integrierten 0.91 OLED Display



Contents:

1. Indtroduction

Welcome!

And thank you for purchasing our AZ-Delivery ESP8266 microcontroller
with integrated 0.91" OLED display.
On the following pages, we will take you through the first programming steps.
We wish you a lot of fun!
This board with an ESP8266 chip and an integrated charging interface,
for one battery pack
can be programmed under NodeMCU and Arduino.
We cover the programming process in Arduino in this eBook.
The board has 32MB flash memory,
WLAN and 12 I/O pins (i^2c, SPI, PWM and Analog Input).
The display is 0,91" and has a resolution of 128 x 32 pixels.

2. Programming the ESP8266 with OLED

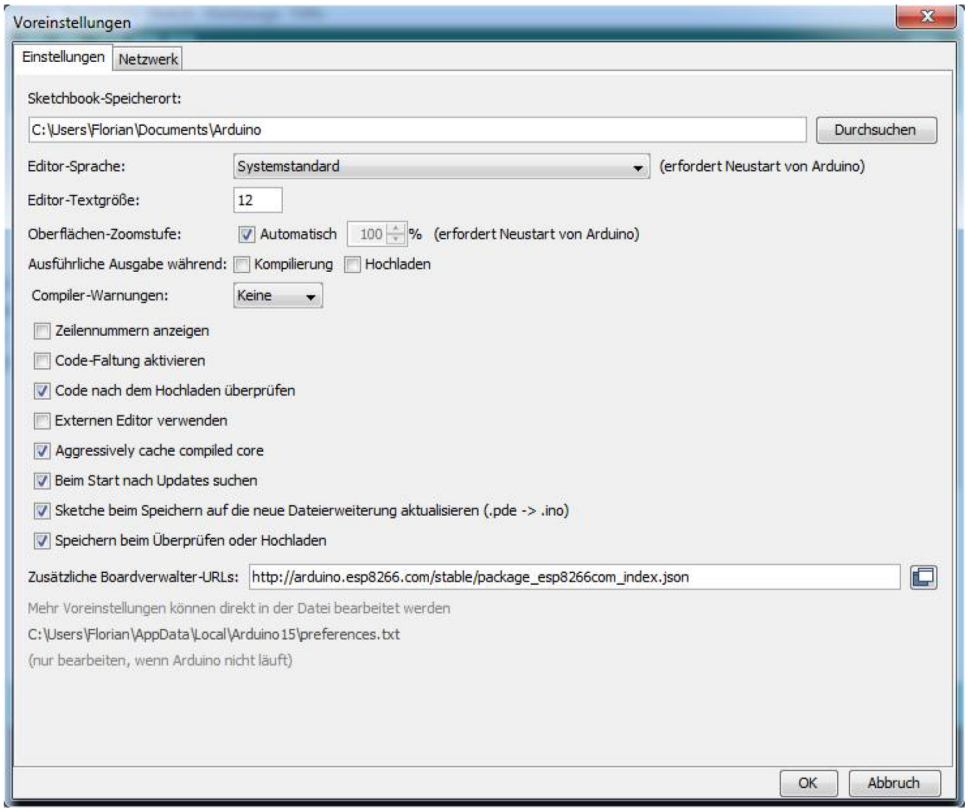
This ESP8622 compared to ESP8266-01, has more flash memory and more GPIO ports.

2. 1 Preparing the software

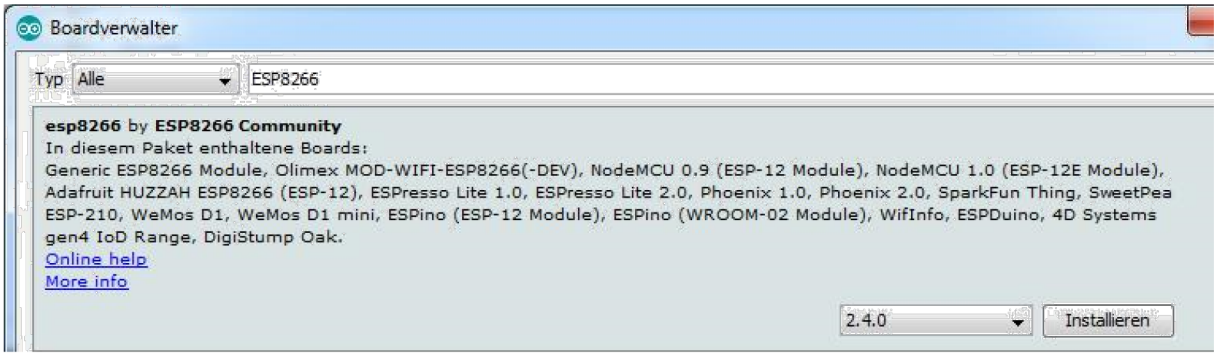
The Arduino software seen and used in this step has already been installed. If, however, you do not have it, then please download and install it on your PC from this web address: https://www.arduino.cc/en/Main/-Software#. Additionally, the driver for the CP210x should also have been installed. If not, then you should know that it comes with the Arduino software.

After all of the basic requirements have been met, we can now start with the installation of the software. The Arduino software will firstly need all of the information about the ESP8266, which we can do by entering the following web address, under the „Preferences" > „Additional Board Administrators-URLs":

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

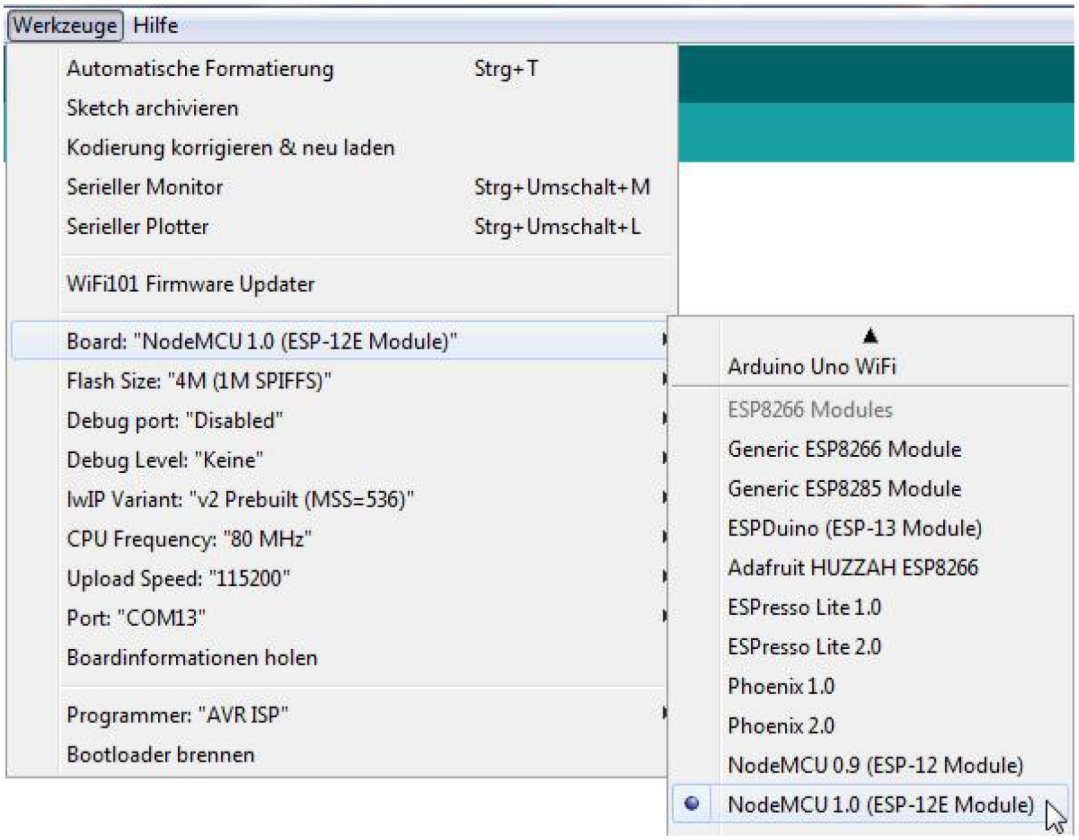


If you have already entered the link, click on the button and add a new line in the window. Confirm the entry by clicking on „OK“. Once this is completed, go to „Tools“ > „Board“ > „Board Administrator“ and install the ESP8266 library. Search for „ESP8266“, in the board administrator, in
the search engine, located in the top right. As a result, you will get the package from ESP8266 Community. Select this one and click on install.

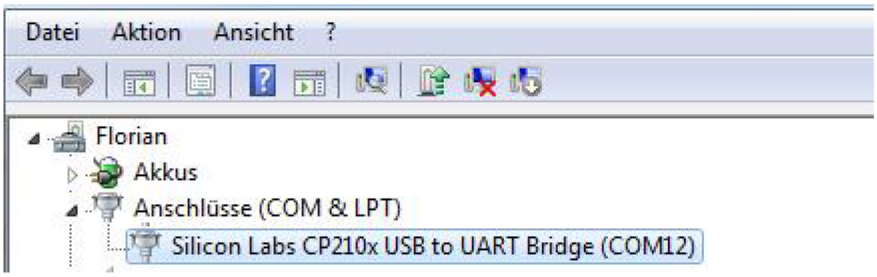


After a successful installation, next to the package INSTALLED will be now shown.

The next step is to select the correct board:
Under Tools > Board: „Generic ESP8266 Module“
Port: “COMxx” (here is your port of the serial adapter)

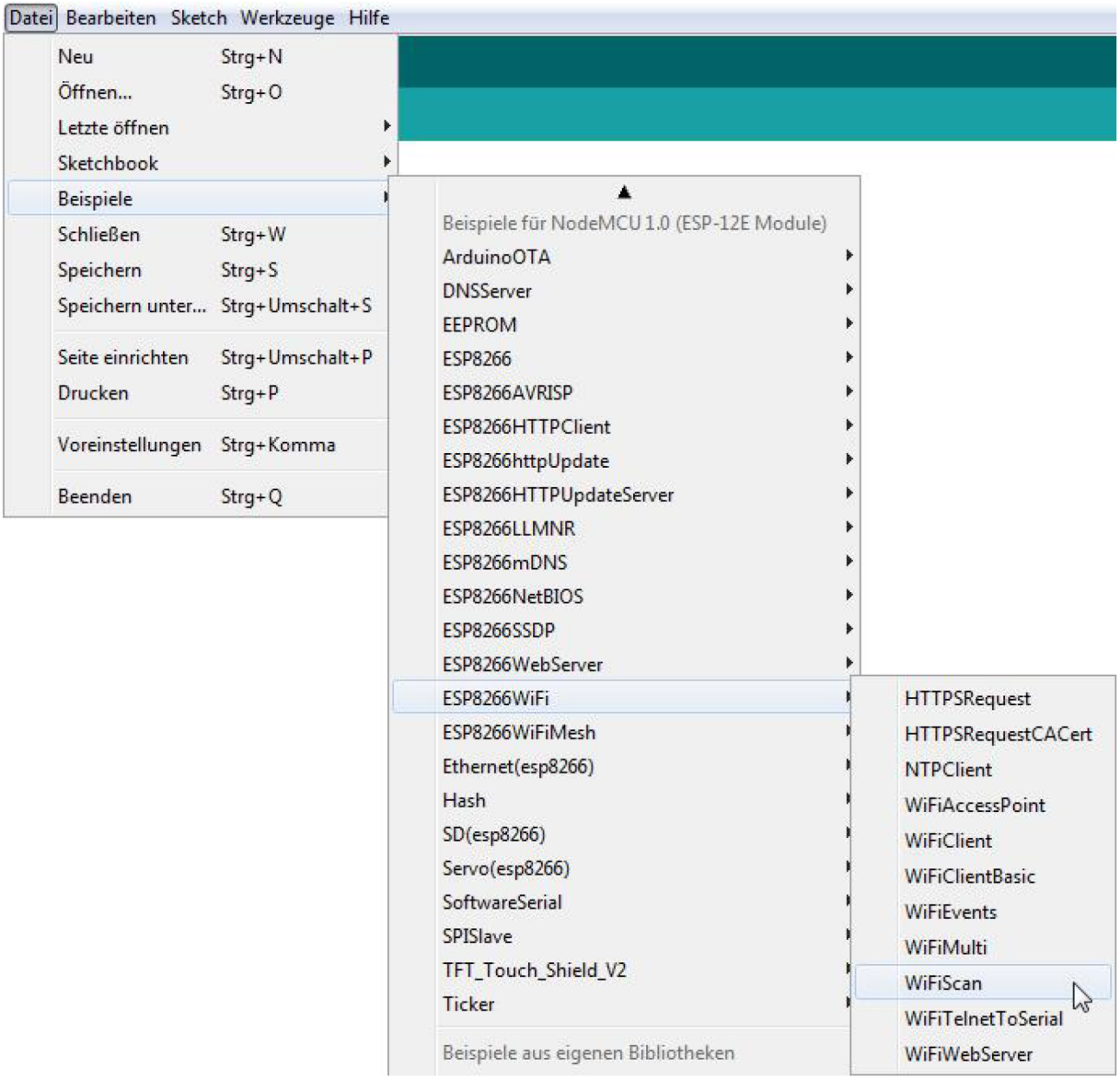


The COMPort can be accessed via the device manager and can also be changed:



2.2 The Arduino Code

Now that the wiring has been completed, we can start writing our first code.
We let all Wi-Fi networks in our surroundings to be shown.
To do this, select under File > Examples > ESP8266Wifi > WifiScan.



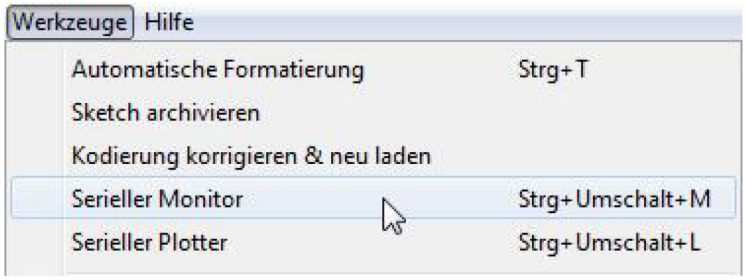
After the code has been displayed, we click on and verify our program:
If everything is correct and our program contains no errors,

```
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
```

We can upload it onto the ESP8622. To do that, we should click on .
After a short period of time, the program will be loaded onto the chip:

```
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
.....................................................................
```

Once the 100% has been reached, then the program has been completely transferred.
Open the Serial Monitor in the Arduino software:
Tools > Serial Monito



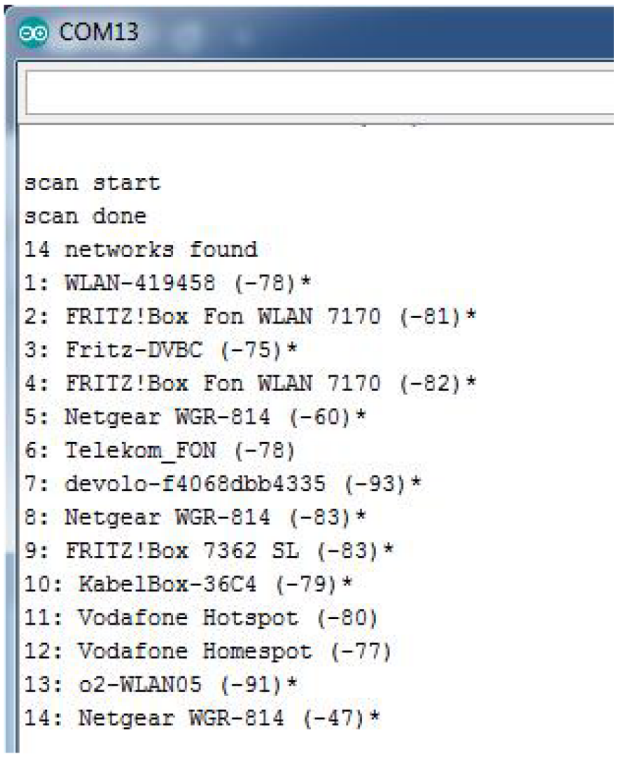
A new scan is automatically started every 5 seconds and the result is distributed via the serial interface.
At the beginning of each and every scan

scan start

is displayed, and it ends with

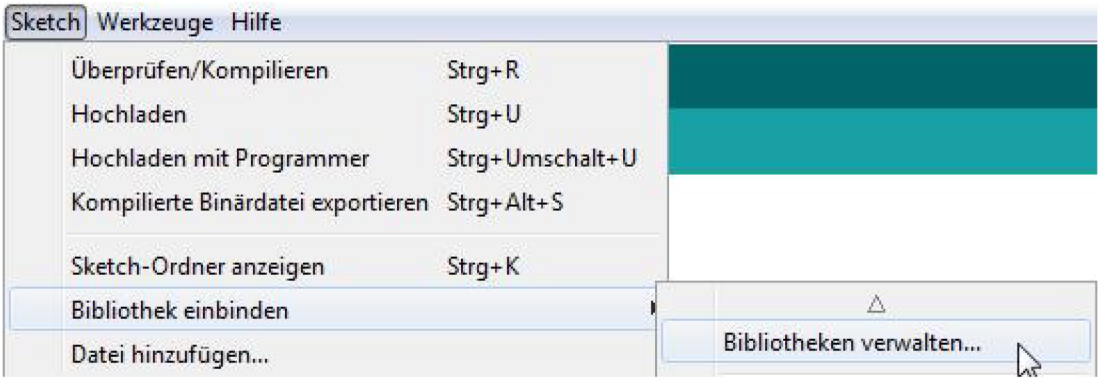
scan done.

Finally, all available Wi-Fi networks will be listed. It will look similar to the image on the right.

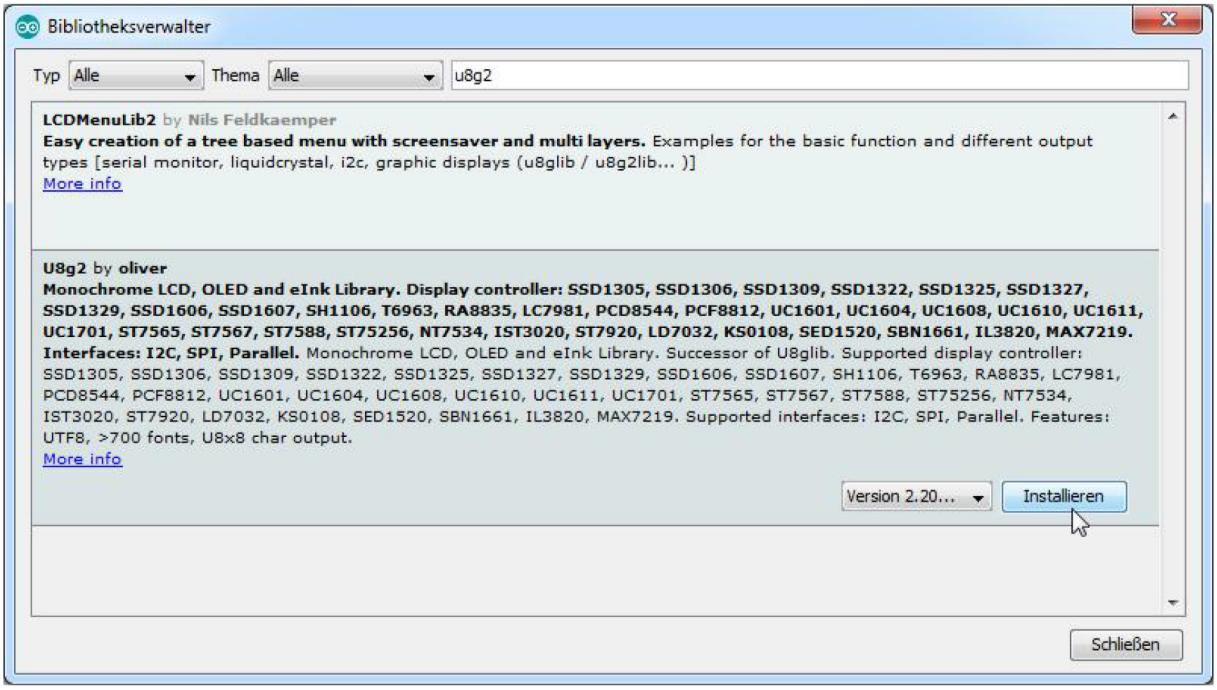


2.3 Controlling the OLED display

In order to be able to control the display, we would need the corresponding libraries (information) in the Arduino software.
Begin under Sketch > Include Library > Manage Libraries ...

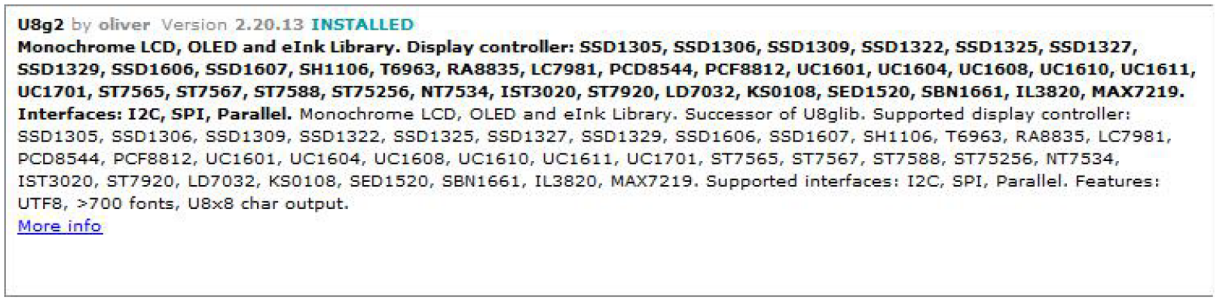


The library administrator and search there for „u8g2"

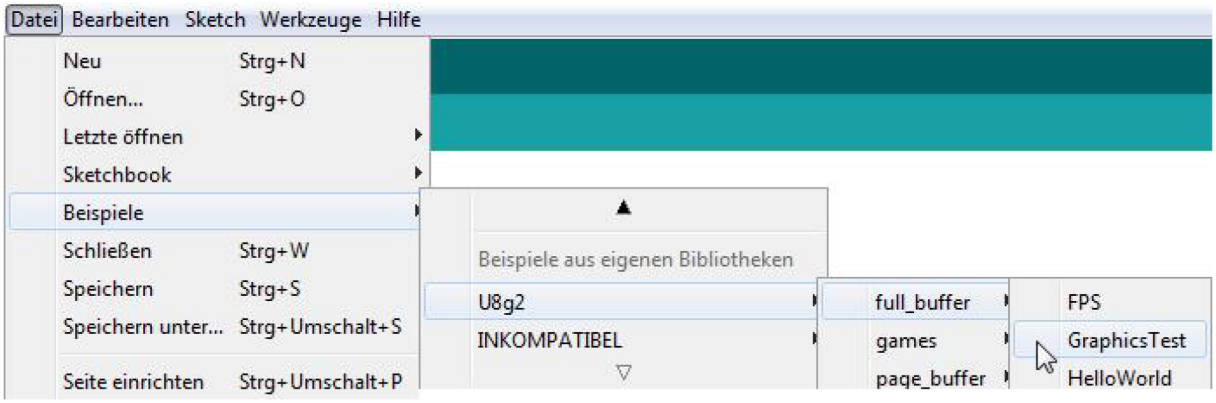


After the package has been selected, click on install, located on the bottom right.

After waiting for a few seconds, the „INSTALLED" will appear.



Now you can close the window, and commence with the programming process.
Select under Examples > U8g2 > full_buffer > GraphicsTest:



Now a longer code will be opened. In the first lines, a lot of display types are registered. These, however, have the "//" characters at the beginning of the line. For our display, we now have to search for and activate this line, by removing the "//" characters, placed at the beginning of the line:

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

After the transmission the display will now show demo texts and images.

Based on this demonstration, we can also program a scrolling text.
For a smooth playback, we should change the CPU frequency to 160MHz and the SPI memory to 3MB (3M SPIFFS).

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

Note: if you would like to make a longer scrolling text, then you have to activate the 16-bit support in the u8g2.h file. The file can be found in the Arduino directory:

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

In line 72 there is //#define U8G2_16BIT

It should be changed to #define U8G2_16BIT

Subsequently, save the file and then re-transmit the code.

The code is as follows:

```
#include <U8g2lib.h>
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();
    }
}
```