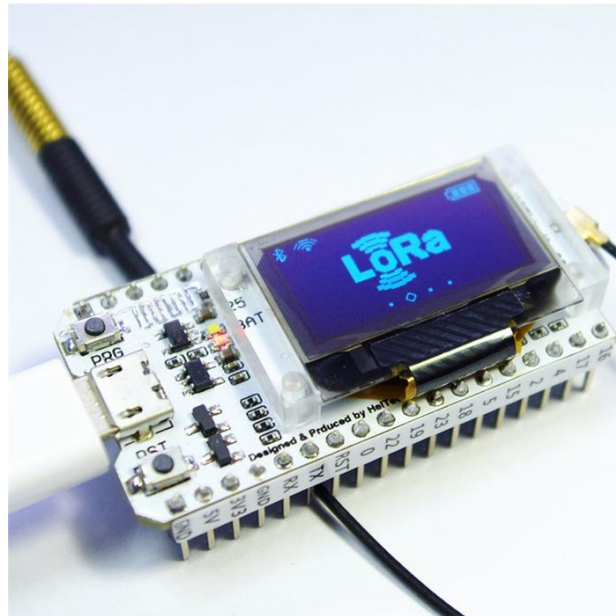


## Willkommen!

Und herzlichen Dank für den Kauf unseres AZ-Delivery NodeMCU ESP32 mit integrierten OLED Display. Auf den folgenden Seiten gehen wir mit dir gemeinsam die ersten Schritte des Programmierens durch.

Viel Spaß!



Der Leistungsstarke ESP32 hat in diesem Modul viele Möglichkeiten. Es werden WLAN, Bluetooth, Display und ein 433 MHz Transceiver onBoard mitgeliefert. Die Leistung reicht von einfach Sensorauswertung bis hin zu Voice Encoding und Musik Streaming. Die Frequenz ist variabel einstellbar von 80MHz bis 240 MHz. Viele IO Pins lassen einem die Möglichkeit weitere Sensoren und Schnittstellen anzusteuern. Je nach Anwendungszweck kann die Stromaufnahme auf unter 5µA abgesenkt werden und dein Modul kann somit perfekt über die Integrierte Akkuschnittstelle mit einer Batterie versorgt werden.

## 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.

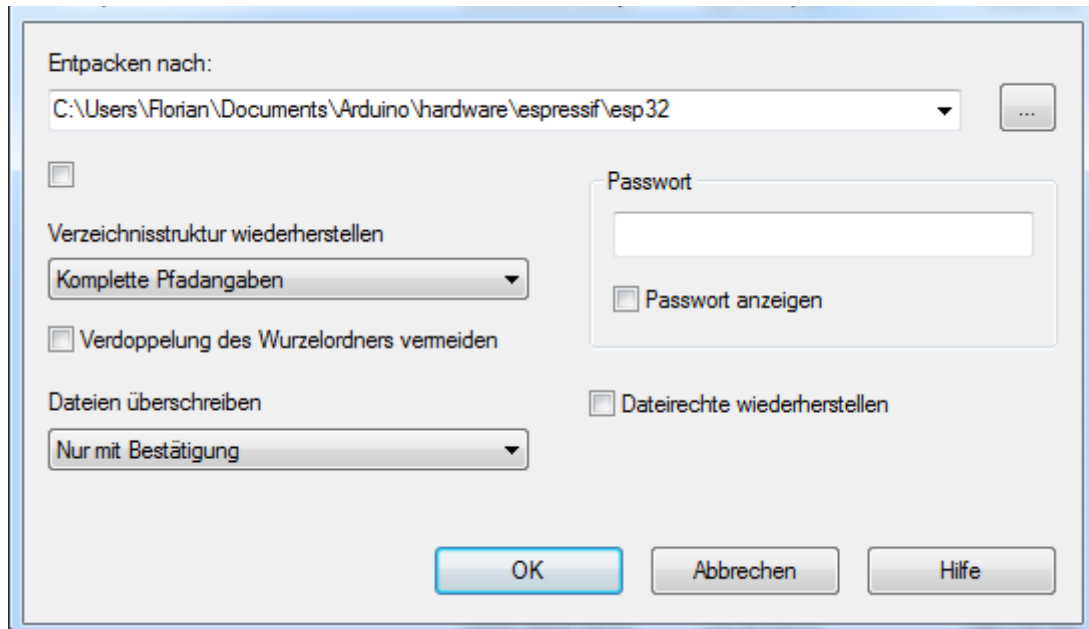
Die Treiber für den CP210x Serial Chip werden beim anstecken automatisch (von der Arduino Software mitgeliefert) installiert.

Nachdem alle Grundvoraussetzungen getätigt wurden, müssen wir uns noch die benötigten Pakete für den ESP32 manuell herunterladen und in die Arduino Software einbinden. Laden wir dazu von GIT die aktuellen Daten herunter:

<https://github.com/Alictronix/LoRa-ESP32-OLED/archive/master.zip>

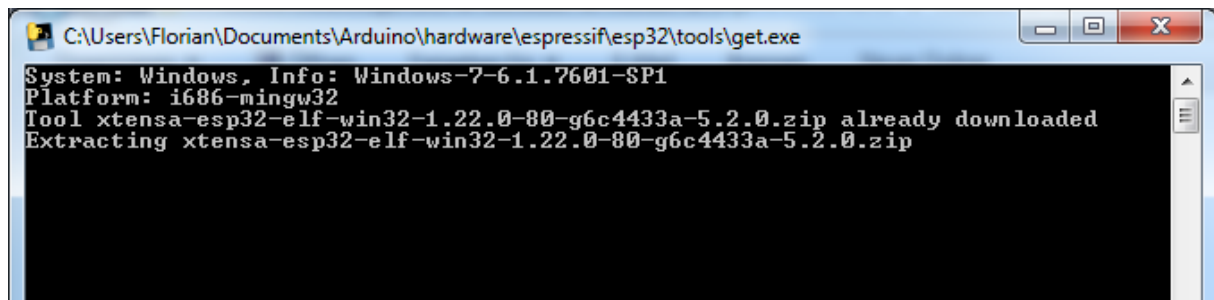
Diese Zip Datei entpacken (mit 7zip) wir in den Ordner: [Eigenes Userverzeichnis (C:\Benutzer\Florian)\ \ Eigene Dokumente \ Arduino \ hardware \ espressif \ esp32

Hinweis: Sollten diese Ordner nicht existieren, dann lege diese einfach neu an.



Name	Änderungsdatum	Typ	Größe
arduino-esp32-master	23.01.2018 12:08	Dateiordner	
cores	23.01.2018 12:08	Dateiordner	
docs	23.01.2018 12:08	Dateiordner	
libraries	23.01.2018 12:08	Dateiordner	
package	23.01.2018 12:08	Dateiordner	
tools	28.01.2018 17:38	Dateiordner	
variants	23.01.2018 12:08	Dateiordner	
.gitignore	23.01.2018 12:08	GITIGNORE-Datei	1 KB
.gitmodules	23.01.2018 12:08	GITMODULES-Datei	1 KB
.travis.yml	23.01.2018 12:08	YML-Datei	3 KB
appveyor.yml	23.01.2018 12:08	YML-Datei	1 KB
boards.txt	23.01.2018 12:08	TXT-Datei	51 KB
component.mk	23.01.2018 12:08	MK-Datei	1 KB
Kconfig	23.01.2018 12:08	Datei	3 KB
Makefile.projbuild	23.01.2018 12:08	PROJBUILD-Datei	1 KB
package.json	23.01.2018 12:08	JSON-Datei	1 KB
platform.txt	23.01.2018 12:08	TXT-Datei	9 KB
programmers.txt	23.01.2018 12:08	TXT-Datei	0 KB
README.md	23.01.2018 12:08	MD-Datei	3 KB

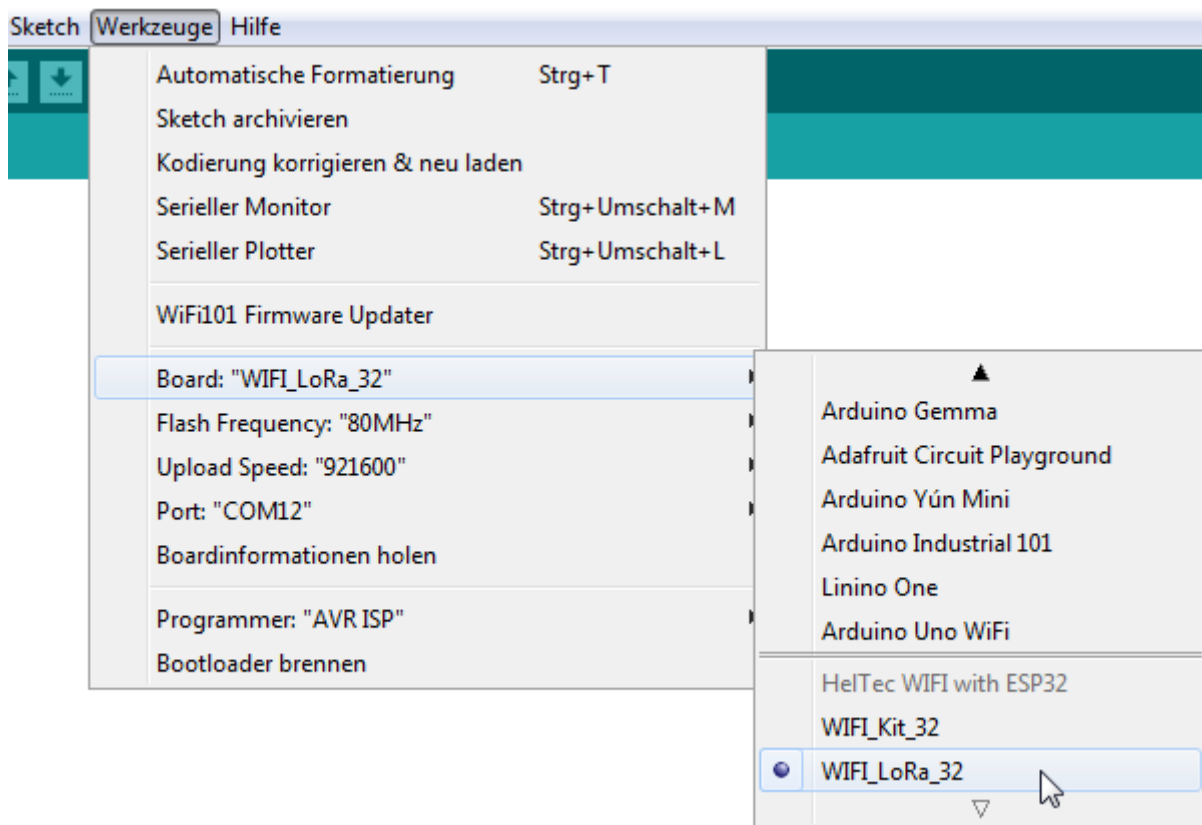
In dem Ordner tools, gibt es eine „get.exe“. Diese müssen wir einmal ausführen und alle benötigten Softwarepakete installieren und herunterladen lassen.



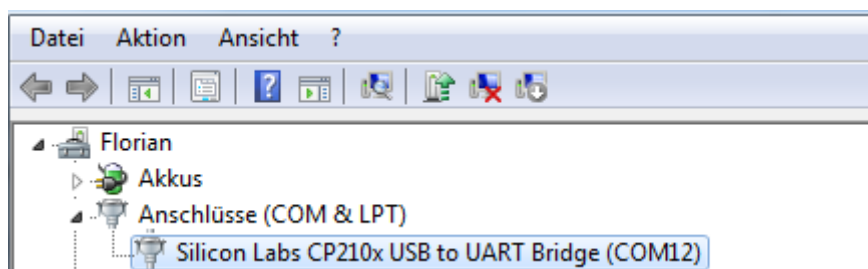
```
C:\Users\Florian\Documents\Arduino\hardware\espressif\esp32\tools\get.exe
System: Windows, Info: Windows-7-6.1.7601-SP1
Platform: i686-mingw32
Tool xtensa-esp32-elf-win32-1.22.0-80-g6c4433a-5.2.0.zip already downloaded
Extracting xtensa-esp32-elf-win32-1.22.0-80-g6c4433a-5.2.0.zip
```

Dies dauert einen kurzen Moment, wenn alles abgeschlossen wurde, schließt das schwarze Fenster von selbst wieder.

Anschließend starten wir die Arduino Software und gehen unter Werkzeuge > Board und suchen uns das ESP32 Dev Module heraus.



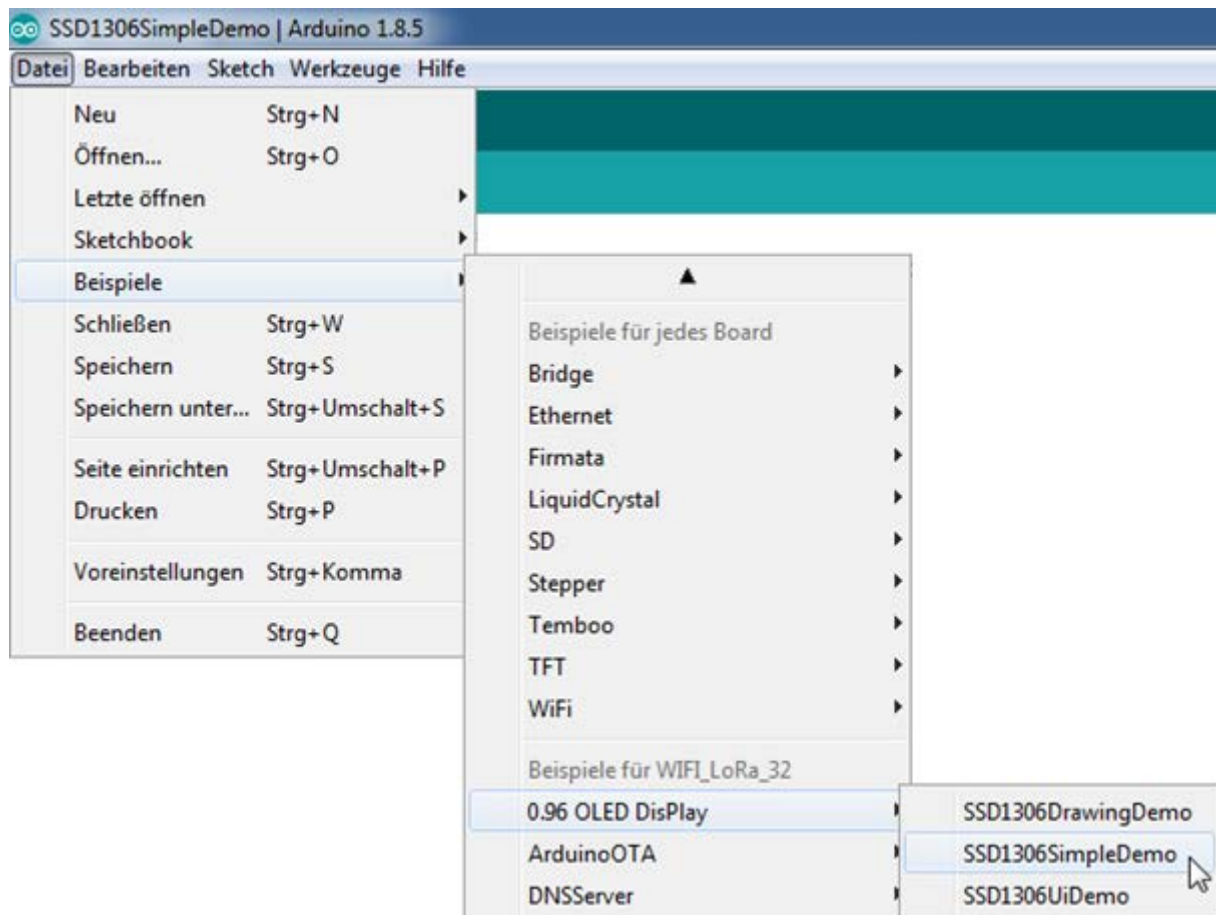
Bei Port musst du nur noch den Com-Port deines Serial Adapters eintragen, diesen kannst du beim Gerätemanager auslesen und ggf. auch abändern.




## Der Arduino Code:

Nachdem nun alle Vorbereitungen erledigt wurden, schreiben wir unseren ersten Code.

Wähle unter Datei > Beispiele > 0.96 OLED Display > SSD1306SimpleDemo aus:



Nun wird uns ein langer Code angezeigt. Diesen Code lassen wir unverändert und können nun oben auf  klicken und somit unser Programm Verifizieren Programm.

Wenn alles stimmt und unser Programm keine Fehler enthält

```
Kompilieren abgeschlossen.  
Archiving built core (caching) in: C:\Users\Florian\AppData\Local\Temp\arduino_cache_759261\core\core_espressif_esp32_WiFi_LoRa_32_FlashFreq_80,UploadSpeed_115200  
Der Sketch verwendet 213127 Bytes (20%) des Programmspeicherplatzes. Das Maximum sind 1044464 Bytes.  
Globale Variablen verwenden 22072 Bytes (7%) des dynamischen Speichers, 272840 Bytes für lokale Variablen verbleiben. Das Maximum sind 294912 Bytes.
```

können wir es auf den ESP32 hochladen. Dazu klicken wir oben auf 

Nach kurzer Zeit sollten diese Zeilen angezeigt werden:

```
Hochladen abgeschlossen.
Archiving built core (caching) in: C:\Users\Florian\AppData\Local\Temp\arduino_cache_759261\core\core_espressif_esp32_
Der Sketch verwendet 231603 Bytes (22%) des Programmspeicherplatzes. Das Maximum sind 1044464 Bytes.
Globale Variablen verwenden 22188 Bytes (7%) des dynamischen Speichers, 272724 Bytes für lokale Variablen verbleiben.
esptool.py v2.0-dev
Connecting...
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 921600
Changed.
Attaching SPI flash...
Configuring flash size...
Compressed 8880 bytes to 5533...

Writing at 0x00001000... (100 %)
Wrote 8880 bytes (5533 compressed) at 0x00001000 in 0.1 seconds (effective 888.0 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 105...

Writing at 0x00008000... (100 %)
Wrote 3072 bytes (105 compressed) at 0x00008000 in 0.0 seconds (effective 2048.0 kbit/s)...
Hash of data verified.
Compressed 8192 bytes to 47...

Writing at 0x0000e000... (100 %)
Wrote 8192 bytes (47 compressed) at 0x0000e000 in 0.0 seconds (effective 3640.9 kbit/s)...
Hash of data verified.
Compressed 301824 bytes to 140016...

Writing at 0x00010000... (11 %)
Writing at 0x00014000... (22 %)
Writing at 0x00018000... (33 %)
Writing at 0x0001c000... (44 %)
Writing at 0x00020000... (55 %)
Writing at 0x00024000... (66 %)
Writing at 0x00028000... (77 %)
Writing at 0x0002c000... (88 %)
Writing at 0x00030000... (100 %)
Wrote 301824 bytes (140016 compressed) at 0x00010000 in 3.3 seconds (effective 726.4 kbit/s)...
Hash of data verified.

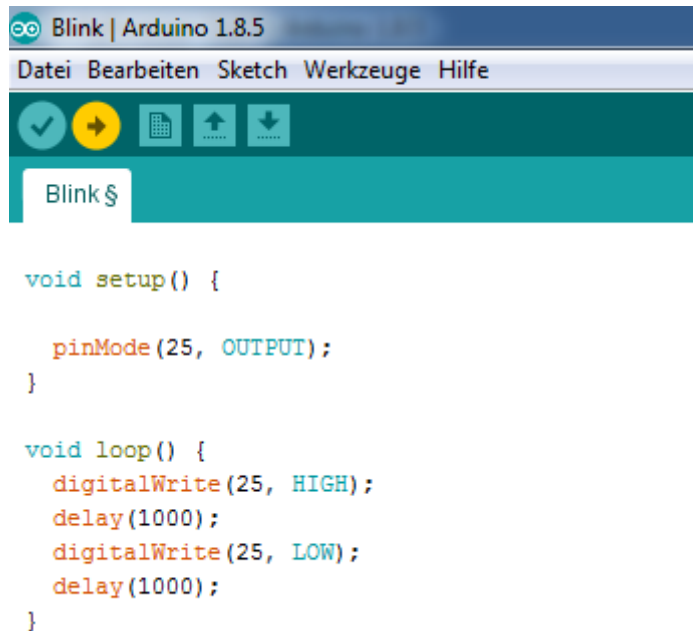
Leaving...
Hard resetting...
```

Sollte das Flashen nicht automatisch funktionieren, dann drücken wir beide Taster auf dem Board (PRG und RST) und versetzen den ESP32 in den Programmiermodus. Dies sollte aber nur in Ausnahmefällen nötig sein.

Auf dem Display werden nun verschiedene Text und Bild-Demos angezeigt.

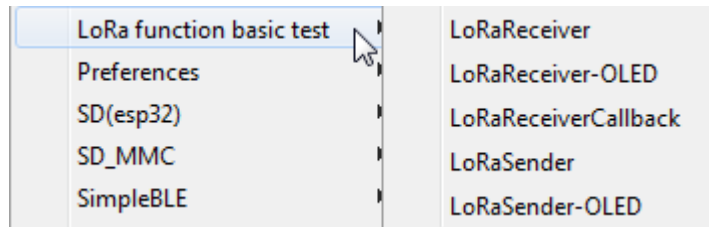
Du kannst auch nur eine LED blinken lassen und die Blink Beispieldatei ausprobieren.

Die OnBoard LED hat die Nummer 25. Der Code fürs blinken sieht so aus:



Es sind viele Beispiele enthalten, probiere alle einmal aus und passe die Skripte an deine Projekte an und verwirkliche deine Ideen.

Sehr interessant ist es auch, wenn du 2 Boards hast kannst du diese mit LoRa (433MHz) miteinander kommunizieren lassen.



Viel Spaß!

**Du hast es geschafft, jetzt kannst du deine eigenen Projekte programmieren.**

Ab jetzt heißt es eigene Projekte verwirklichen.

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

<https://az-delivery.de>

Viel Spaß!  
Impressum

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