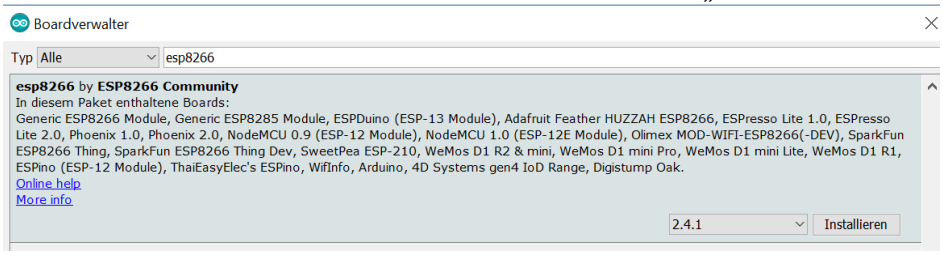
Einrichtung der Arduino IDE zur Programmierung der ESP8266 Module

Auf dem Computer muss das Programm „Arduino“, über das sich Mikrocontroller programmieren lassen, installiert sein. Es kann unter folgendem Link kostenlos heruntergeladen werden:

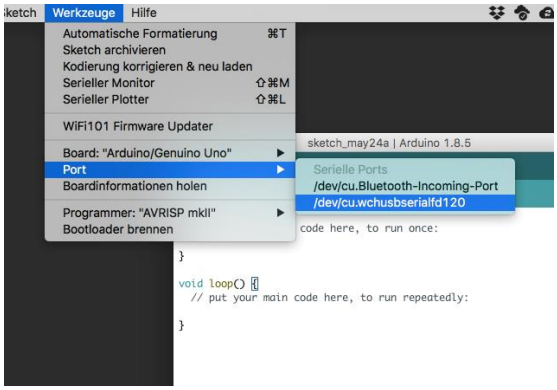
https://www.arduino.cc/en/Main/Software

Bevor Programme auf den Mikrocontroller gespielt werden können, müssen noch ein paar Schritte davor ausgeführt werden:

1) In dem Programm „Arduino“ unter Datei → Voreinstellungen in dem Feld „Zusätzliche Boardverwalter-URLs“ folgenden Link eingeben `http://arduino.esp8266.com/stable/package_esp8266com_index.json` und mit OK bestätigen
2) Unter Werkzeuge → Board → Boardverwalter nach „ESP8266“ suchen und das Paket mit einem Klick auf „Installieren“ installieren



3) Schließe nun den Mikrocontroller mit dem USB-Kabel an den Computer an
4) Unter Werkzeuge → Board nun „WeMos D1 R2 & Mini“ aus der Liste auswählen
5) Nun muss noch ein weiterer Treiber (nämlich der „CH340G Driver“) installiert werden: Auf der Internetseite `wiki.wemos.cc/downloads` findet ihr die entsprechenden Installationsdateien für euer Betriebssystem (Windows oder Mac OS X). Nachdem ihr diesen Treiber installiert habt muss der Computer neugestartet werden und anschließend das Programm „Arduino“ erneut geöffnet werden
6) Unter Werkzeuge → Port muss nun der USB-Anschluss, an dem der Mikrocontroller angeschlossen ist, ausgewählt werden. Im abgebildeten Beispiel heißt dieser Port „/dev/cu.wchusbserialfd120“ (Hinweis: bei euch kann diese Bezeichnung leicht abweichen)



Nun kann der Mikrocontroller programmiert werden.

Zur Info: Der Mikrocontroller führt nun – da er über den USB-Anschluss Strom bekommt – das Programm aus, welches als letztes darauf gespeichert wurde. Es kann also sein, dass die LED auf dem Mikrocontroller bereits blinkt, ohne dass etwas hochgeladen wurde.

Auf dem Board befindet sich eine Onboard-LED, die nun zum Blinken gebracht werden sollen, um euch mit der Programmierumgebung vertraut zu machen. In dem Programm „Arduino“ lassen sich viele Code-Beispiele finden.

1) Öffne über *Datei → Beispiele → 01.Basics* den Beispielcode „Blink“. In der folgenden Abbildung ist ein Ausschnitt dieses Programms dargestellt.

```
24 // the setup function runs once when you press reset or power the board
25 void setup() {
26   // initialize digital pin LED_BUILTIN as an output.
27   pinMode(LED_BUILTIN, OUTPUT);
28 }
29
30 // the loop function runs over and over again forever
31 void loop() {
32   digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
33   delay(1000);                       // wait for a second
34   digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
35   delay(1000);                       // wait for a second
36 }
```

   Kurze Erklärung des Codes:

   In dem Code stehen einige Kommentare, die dadurch gekennzeichnet sind, dass sie entweder durch `/*` begonnen und durch `*/` beendet werden oder hinter `//` stehen. Der Text von Kommentaren ist grau markiert und für die Funktionsweise des Codes nicht wichtig.

   In der `setup()` Funktion werden Voreinstellungen vorgenommen. Sie wird automatisch ausgeführt, sobald der Mikrocontroller Strom bekommt oder über die Reset Taste neugestartet wird.

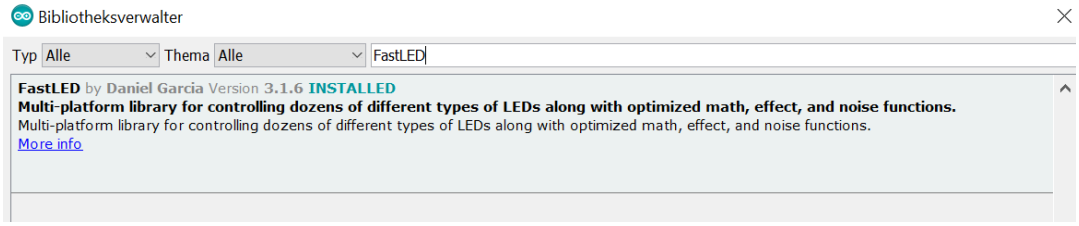
   Die `loop()` Funktion (Loop ist Englisch für Schleife) wird immer wieder hintereinander ausgeführt. In diesem Fall soll die Onboard-LED für 1000 Millisekunden (das entspricht einer Sekunde) leuchten („HIGH“) und dann wieder für 1000 Millisekunden nicht leuchten („LOW“). Dadurch kommt das Blinken zustande.

2) Um das Programm auf den Mikrocontroller zu laden, muss man auf „Hochladen“ klicken.

Nun wollen wir eine externe LED zum Leuchten bringen.

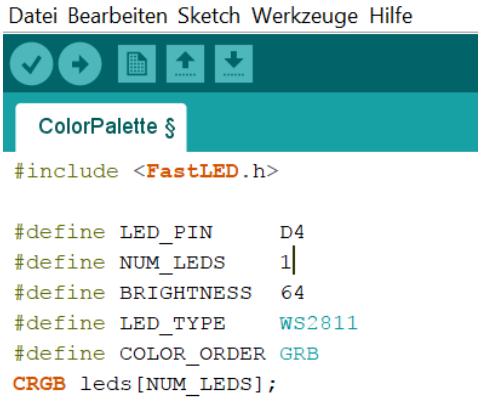
Um die externe LED mit dem Mikrocontroller anzusteuern, muss zunächst ein Paket nachinstalliert werden.

Hierfür geht man oben im Programm „Arduino“ in der Menüleiste auf Sketch → Bibliothek einbinden → Bibliotheken verwalten und sucht dann in dem sich öffnenden Fenster nach „FastLED“ und installiert die neueste Version dieser Bibliothek (siehe folgende Abbildung).



Nun wird die LED an den Mikrocontroller angeschlossen. Das schwarze Kabel steht für den Minus-Pol der LED, das rote Kabel für den Plus-Pol und das gelbe Kabel für den Daten-Pin. Diese müssen nun an die richtigen Stellen im Mikrocontroller gesteckt werden: Schwarz → 5V, Rot → G, Gelb → z.B. D4.
Für das Paket FastLED gibt es wieder verschiedene Beispielprogramme, von denen „FastLED ColorPalette“ nun genauer betrachtet wird:
Dieses wird unter Datei → Beispiele → FastLED → ColorPalette geöffnet.

Der Code ist ähnlich aufgebaut wie das erste Beispiel. Allerdings wird hier ganz am Anfang das Paket „FastLED.h“ eingefügt (Befehl: `#include <FastLED.h>`) und einige Dinge wie zum Beispiel die Anzahl der LEDs definiert. Hinter `#define NUM_LEDS` nun eine `1` eingeben, sofern nur eine LED verwendet wird. Mit dem Befehl `#define LED_PIN` wird dem Programm mitgeteilt, an welchem Pin des Mikrocontrollers der Daten-Pin der LED angeschlossen wird. Tragt an dieser Stelle im Programm beispielsweise `D4` ein.



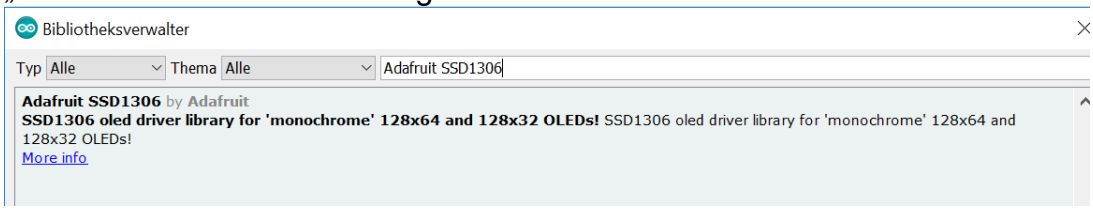
Dann folgt wie im obigen Beispiel die `setup()` Funktion für die Voreinstellungen.
Die `loop()` Funktion wird wieder immer wieder hintereinander ausgeführt und die LED wechselt stetig ihre Farbe.
Um das Programm auf den Mikrocontroller zu laden, muss wieder „Hochladen“ geklickt werden.

Um Displays mit dem Mikrocontroller zu verwenden, müssen entsprechende Bibliotheken ebenfalls nachinstalliert werden.

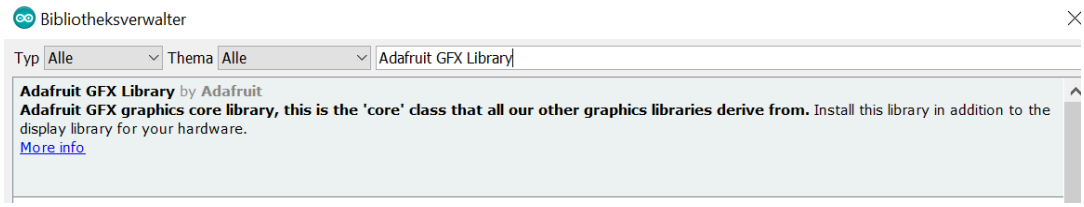
In unserem Workshop verwenden wir 0,66“ OLED Shields von WeMos für den D1 mini. Bevor das Display auf den Mikrocontroller gesetzt wird, sollte der Mikrocontroller zunächst vom Computer getrennt und die LED aus den Pins genommen werden. Beim Setzen des Displays auf den Mikrocontroller ist darauf zu achten, dass es richtig herum draufgesteckt wird. An dem Display finden sich wie auf dem Mikrocontroller ebenfalls die Bezeichnungen „5V“, „G bzw. GND“, „D4“ usw. Diese müssen jeweils passend auf den Mikrocontroller gesteckt werden. Nun kann dieser wieder an den Computer angeschlossen werden.

Um das von uns verwendete Display nutzen zu können, müssen zuerst zwei Pakete installiert werden. Hierfür geht man wie folgt vor:

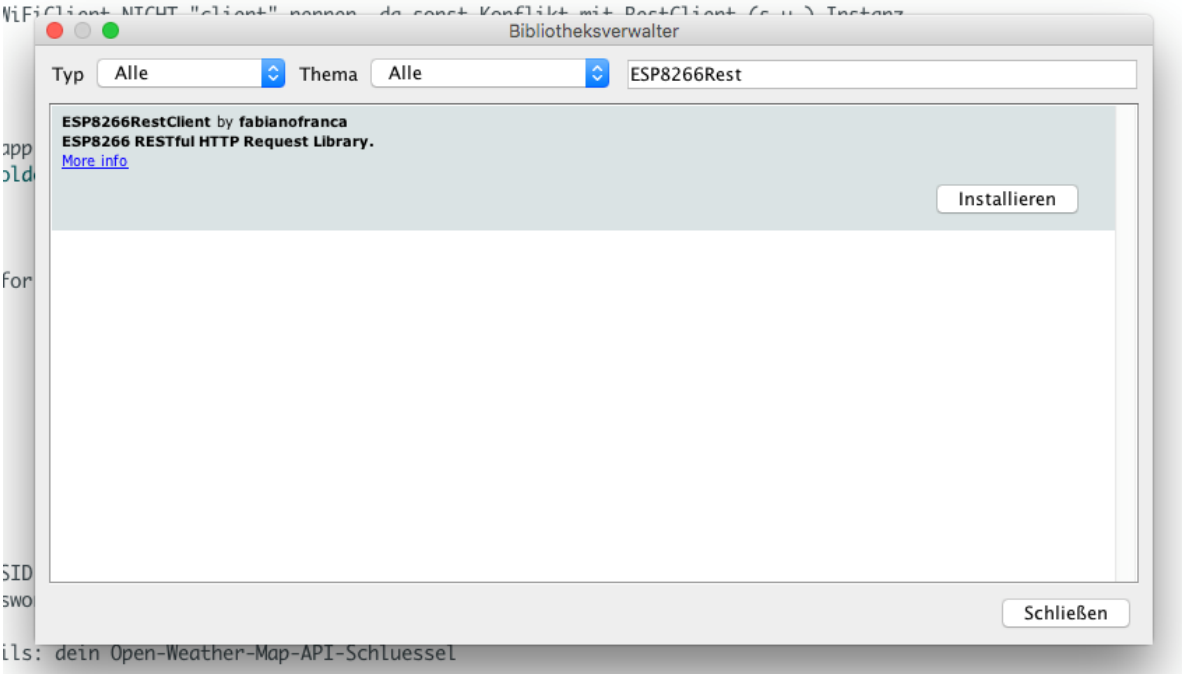
1) Unter Sketch → Bibliothek einbinden → Bibliotheken verwalten in der Suche „Adafruit SSD1306“ eingeben und dieses Paket installieren



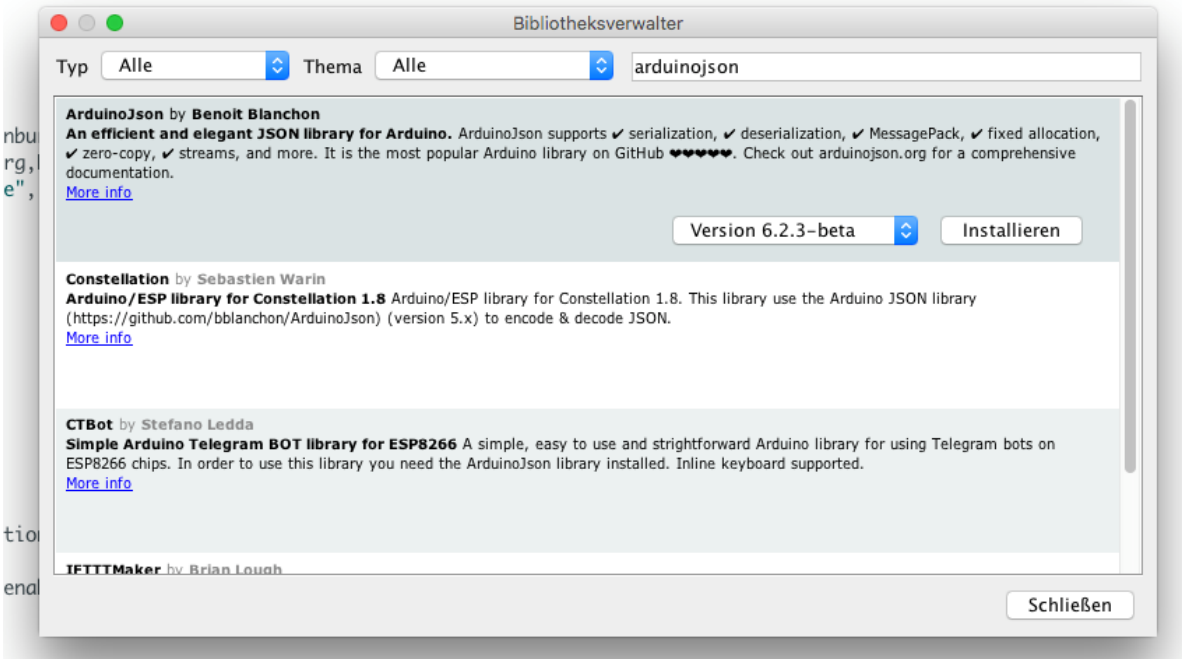
2) Anschließend unter Sketch → Bibliothek einbinden → Bibliotheken verwalten in der Suche „Adafruit GFX Library“ eingeben und dieses Paket installieren.



Um mit dem ESP8266 Anfragen über das Internet – wie z.B. an openweathermap.org – zu schicken, wird die Bibliothek „ESP8266RestClient“ benötigt, die über Sketch → Bibliothek einbinden → Bibliotheken verwalten installiert werden kann.



Um die Antworten, die so – häufig im JSON-Format – zurückgegeben werden, verarbeiten zu können, muss zusätzlich noch die Bibliothek „ArduinoJson“ installiert werden (erneut über Sketch → Bibliothek einbinden → Bibliotheken verwalten)



Der Code für die Blume liegt im digitalen Archiv. Die Effekte für die jeweiligen Wetterzustände können eingefügt werden:
https://github.com/projekt-smile/bloecke-blumen-mikrocontroller-und-das-internet-of-things/blob/master/Material_12_ArduinoSketch.ino