

Teil 1

# Lektion

# 5

*Blink und Add Libraries*

## Übersicht

In dieser Lektion lernen Sie, wie Sie Ihr UNO R3 Entwicklungsboard so programmieren, dass die eingebaute LED leuchtet. Außerdem erfahren Sie Schritt für Schritt, wie man ein Programm auf das Arduino Board hochlädt.

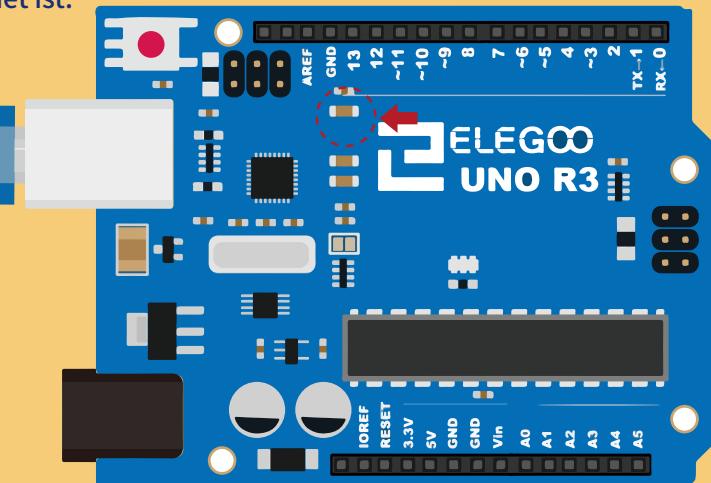
Darüber hinaus müssen wir lernen, wie man Bibliotheken hinzufügt, für die wir Bibliotheksfunktionen in zukünftigen Lernprozessen verwenden können, um Arduino-Funktionen einfacher zu erweitern.

### **Benötigte Bauteile:**

- (1) x Elegoo Uno R3

Aufbau

Das UNO R3 Board hat an beiden Seiten Steckplätze, die dazu benutzt werden können, externe elektrische Geräte und Module („Shields“) anzuschließen, um so die technischen Möglichkeiten zu erweitern. Das Board hat außerdem eine LED, die Sie in Ihren Sketches (Programmen) kontrollieren können. Diese LED ist fest im UNO R3 Board verbaut und wird oft „L“-LED genannt, da sie so auf dem Board gekennzeichnet ist.



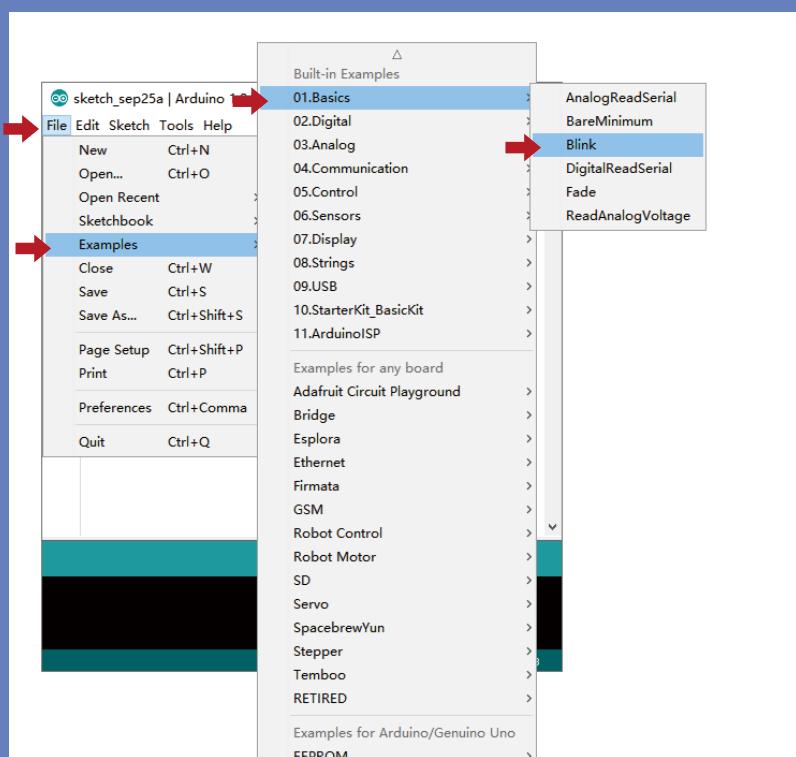
Es kann sein, dass Ihre „L“-LED bereits blinkt, wenn Sie Ihr UNO R3 Board mit einer Stromversorgung verbinden. Das liegt daran, dass die ausgelieferten Boards bereits mit dem „Blink“-Sketch vorinstalliert kommen, der die LED blinken lässt.

In dieser Lektion werden wir das UNO R3 Board mit unserem eigens kreierten Blink Sketch programmieren und anschließend die Blinkgeschwindigkeit ändern.

In Lektion 0 haben Sie die Arduino IDE (Entwicklungsumgebung) aufgesetzt und den richtigen Seriellen Port in der IDE eingestellt. Nun werden wir diese Verbindung testen, indem wir das UNO R3 Board erstmals selber programmieren.

Die Arduino IDE kommt mit einer großen Sammlung von Beispiel-Sketches, die Sie einfach öffnen und auf das Board hochladen können. In dieser Sammlung befindet sich auch bereits ein Sketch, der die L-LED zum leuchten bringt.

Laden Sie den Blink-Sketch, den Sie in der IDE unter Datei > Beispiele > 0.1Basics finden.



```
MyBlink | Arduino 1.8.9
File Edit Sketch Tools Help
MyBlink
/*
Blink

Turns an LED on for one second, then off for one second.

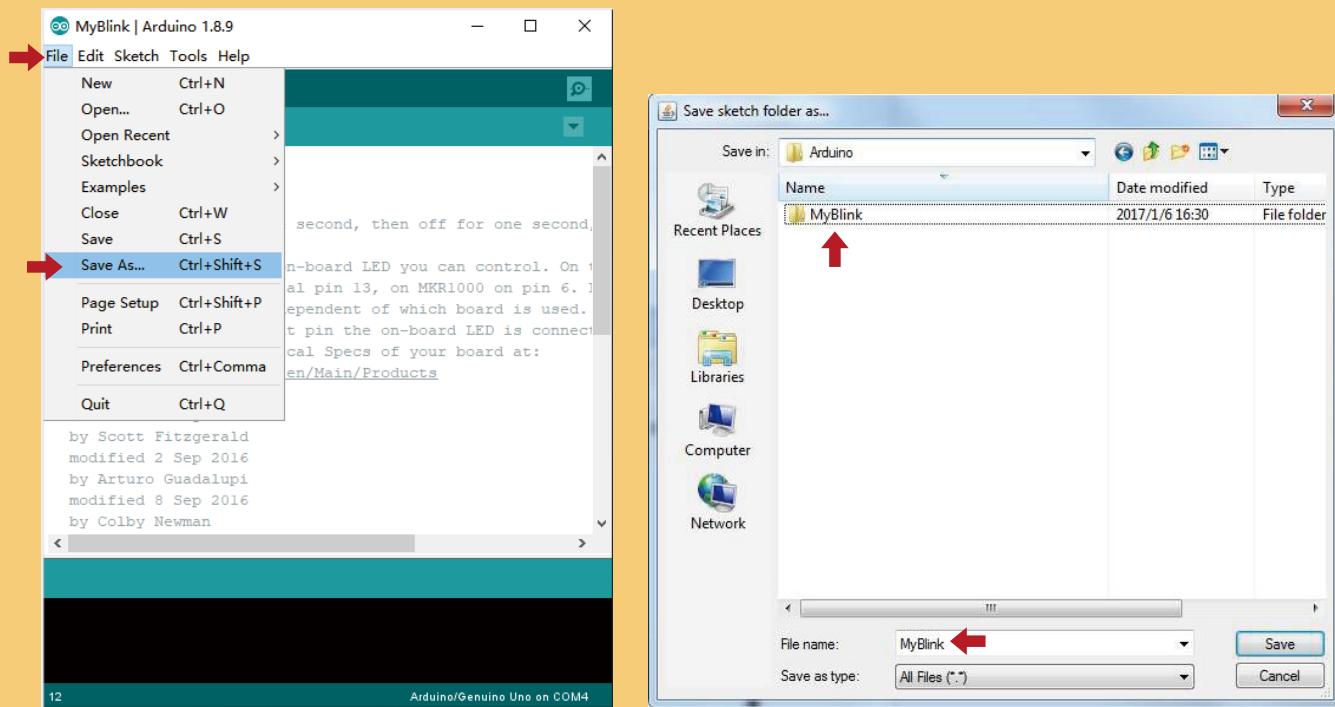
Most Arduinos have an on-board LED you can control. On 1
it is attached to digital pin 13, on MKR1000 on pin 6. 1
the correct LED pin independent of which board is used.
If you want to know what pin the on-board LED is connect
model, check the Technical Specs of your board at:
https://www.arduino.cc/en/Main/Products

modified 8 May 2014
by Scott Fitzgerald
modified 2 Sep 2016
by Arturo Guadalupi
modified 8 Sep 2016
by Colby Newman
Arduino/Genuino Uno on COM4
```

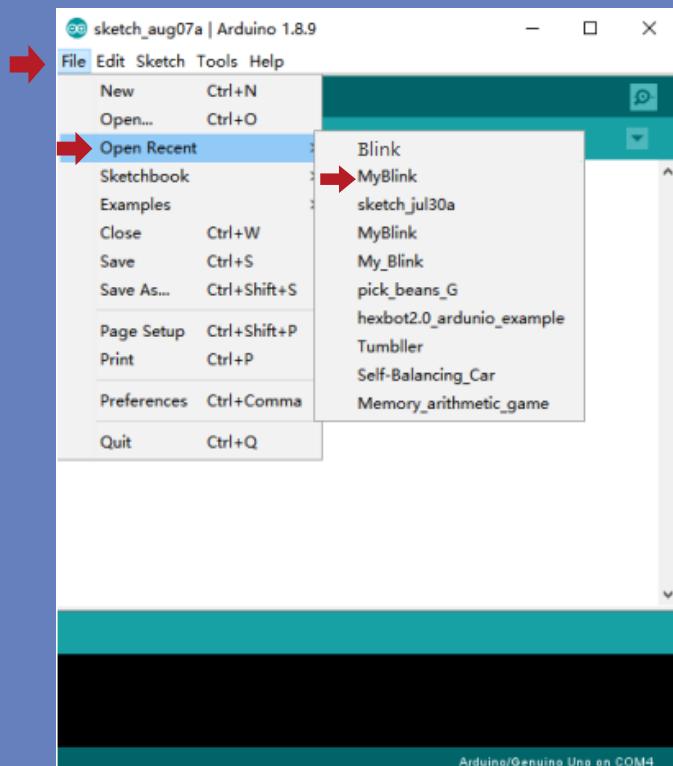
Wenn sich der Blink-Sketch geöffnet hat, sollten Sie das Fenster vergrößern, um sich einen Überblick über den gesamten Programm-Code schaffen zu können.

Die mitinstallierten Beispiel-Sketches sind nicht „schreibbar“. Das heißt, man kann Änderungen nicht abspeichern. Sie können den Sketch abändern und hochladen, aber nicht unter dem gleichen Dateinamen speichern. Da wir den Sketch ändern möchten, ist der erste Schritt den Sketch unter neuem Namen abzuspeichern. Im Arduino IDE Menü wählen Sie Datei > Speichern unter... aus und speichern den Sketch dann unter dem Namen „MyBlink“.

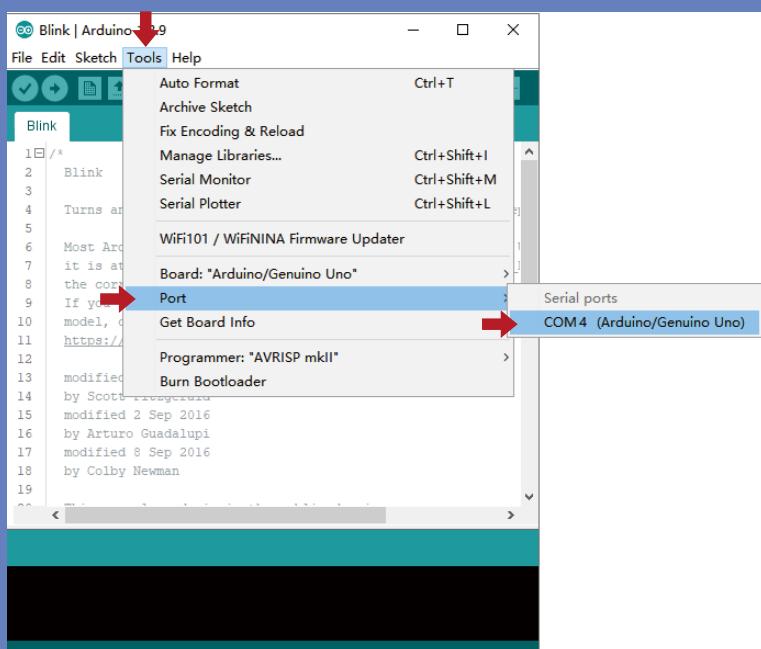
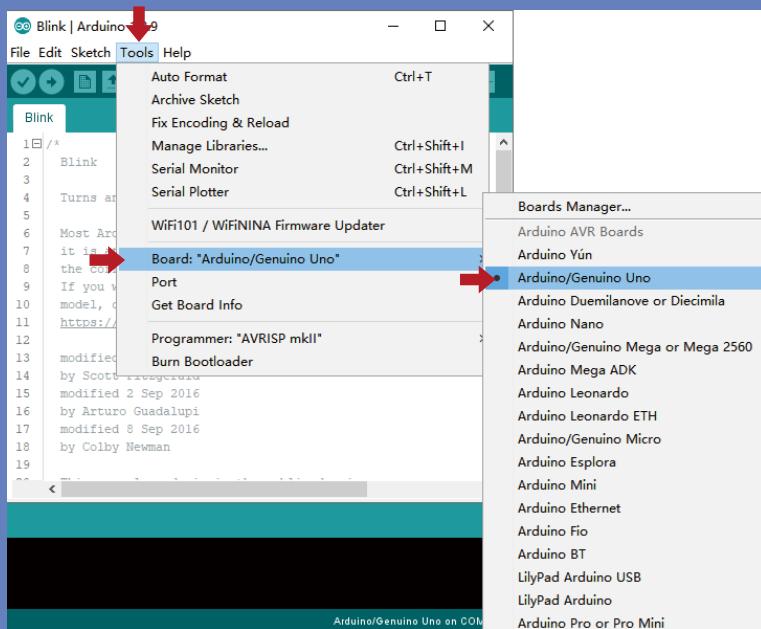
Sie speichern Ihre Kopie des Blink-Sketches in Ihrem Sketchbook (Projektordner) ab.  
Das bedeutet, dass Sie bei Bedarf den Sketch einfach über Datei > Sketchbook im Menü aufrufen können.



Verbinden Sie Ihr Arduino Board mit einem USB-Kabel mit dem Computer und überprüfen Sie, ob der „Board Typ“ und der „Serielle Port“ richtig eingestellt sind,



Verbinden Sie Ihr Arduino Board mit einem USB-Kabel mit dem Computer und überprüfen Sie, ob der „Board Typ“ und der „Serielle Port“ richtig eingestellt sind,



Achtung: Ihr Board-Typ und der Serielle Port kann sich von den in den Bildern zu sehenden Angaben unterscheiden. Wenn Sie das 2560-Board verwenden, dann wählen Sie „Mega 2560“ als Board-Typ, wenn Sie das UNO R3 Board verwenden, wählen Sie „Uno“ als Board-Typ, usw. Der Serielle Port ist bei jedem Gerät unterschiedlich und wird vom Computer zugewiesen. Im Gegensatz zum hier angegebenen Port COM26, könnte es bei Ihnen COM3 oder COM4 sein. Der richtige COM-Port ist durch ein „COMX“ (arduino XXX)“ gekennzeichnet.

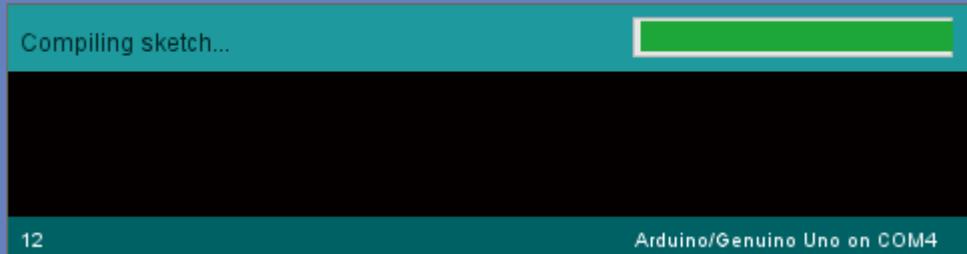
Die Arduino IDE zeigt die aktuellen Einstellungen für das Board im Fenster rechts unten an.



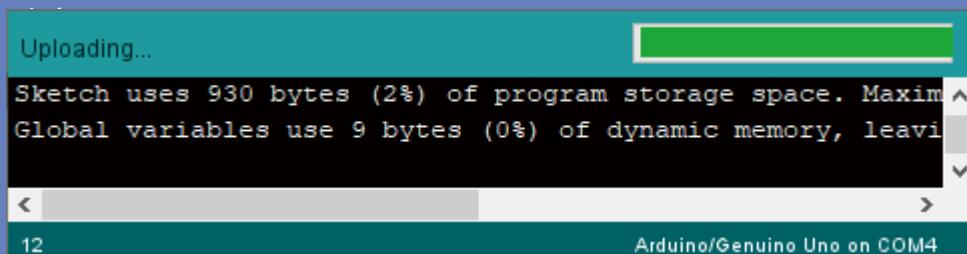
Klicken Sie nun auf die „Hochladen“-Schaltfläche. Dies ist das zweite Symbol von links.



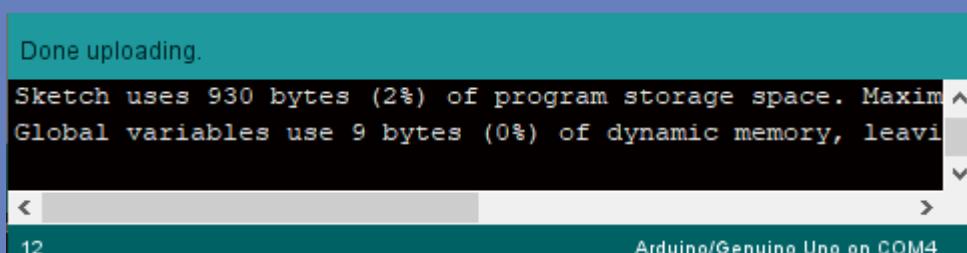
Wenn Sie den Statusbereich der IDE in der unteren Sektion des Fensters beobachten, sehen Sie, wie sich eine Fortschrittsleiste langsam füllt und Meldungen erscheinen. Zuerst erscheint die Meldung „Sketch wird kompiliert...“. Dabei wird der Sketch in ein für das Board passende Format umgewandelt.



Als nächstes ändert sich der Status zu „Hochladen“. An diesem Punkt sollten die LEDs des UNO R3 Boards anfangen zu blinken, da der Sketch auf das Board geladen



Zum Schluss ändert sich der Status zu „Hochladen abgeschlossen“.



Die Meldung darunter teilt uns mit, dass der Sketch 928 Bytes von 32.256 Bytes (der gesamte verfügbare Speicher) belegt. Es ist möglich, dass Sie stattdessen nach dem Schritt „Sketch wird kompiliert...“ eine orange Fehlermeldung erhalten:



Das kann zum einen bedeuten, dass Ihr Board gar nicht mit dem Computer verbunden ist oder aber die Treiber wurden (bei manueller Installation) nicht mitinstalliert. Zuletzt kann auch ein falsch ausgewählter Serieller Port für die Fehlermeldung verantwortlich sein. Wenn dieses Problem bei Ihnen auftritt, gehen Sie zurück zu Lektion 0 und überprüfen Sie Ihre Arduino IDE Installation. Hat dagegen alles geklappt, sollte das Board nach Abschluss neustarten und anfangen zu blinken. Öffnen Sie den Sketch.

### **Code :**

Beachten Sie, dass ein großer Teil dieses Sketches aus Kommentaren besteht. Dabei handelt es sich um keinen richtigen Code, also Anweisungen, was das Board zu tun hat, sondern nur um Kommentare, die dem Programmierer erklären, wie der Sketch funktioniert. Die Kommentare dienen Ihnen zur besseren Verständlichkeit. Alles zwischen den Zeichen /\* und \*/ am Anfang des Sketches ist ein Block Kommentar. Dort wird erklärt, wozu der Sketch dient

### **The Sketch start with**

```
/*
Blink
Turns an LED on for one second, then off for one second, repeatedly.

...
This example code is in the public domain. http://www.arduino.cc/en/Tutorial/Blink
*/
// the setup function runs once when you press reset or power the board
```

//

### **[Weitere Syntax]**

#### **Beschreibung**

Zeilenkommentare sind Zeilen im Programm, mit denen Sie sich oder andere über die Funktionsweise des Programms informieren. Sie werden vom Compiler ignoriert und nicht in den Prozessor exportiert, sodass sie keinen Speicherplatz im Flash-Speicher des Mikrocontrollers belegen. Der einzige Zweck von Kommentaren besteht darin, Ihnen das Verständnis (oder das Erinnern) zu erleichtern oder andere über die Funktionsweise Ihres Programms zu informieren.

Ein einzeiliger Kommentar beginnt mit // (zwei benachbarte Schrägstriche). Dieser Kommentar endet automatisch am Ende einer Zeile. Was auch immer // bis zum Ende einer Zeile folgt, wird vom Compiler ignoriert.

/\*\*/

### **[Weitere Syntax]**

#### **Beschreibung**

Der Anfang eines Block-Kommentars oder eines mehrzeiligen Kommentars wird durch das Symbol /\* eingeleitet und das Symbol \*/ sein Ende markiert. Diese Art von Kommentar wird aufgerufen, da er sich über mehr als eine Zeile erstrecken kann. Sobald der Compiler das /\* liest, ignoriert er alles, was folgt, bis er auf ein \*/ stößt.

### **The first block of code is :**

```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}
```

In diesem Fall gibt es nur einen Befehl, der, wie der Kommentar sagt, dem Arduino-Board dass wir den LED-Pin als Ausgang verwenden werden.

**setup()**

### **[Sketch]**

#### **Description**

Die Funktion **setup()** wird aufgerufen, wenn ein Sketch gestartet wird. Verwenden Sie diese Option, um Variablen, PIN-Modi, die Verwendung von Bibliotheken usw. zu initialisieren. Die Funktion **setup()** wird nach jedem Einschalten oder Zurücksetzen des Arduino nur einmal ausgeführt.

{ ... }

## [Weitere Syntax]

### Beschreibung

Geschweifte Klammern sind ein Hauptbestandteil der Programmiersprache C++. Sie werden in verschiedenen Konstrukten verwendet, die unten beschrieben werden. Dies kann für Anfänger manchmal verwirrend sein.

Auf eine öffnende geschweifte Klammer { muss immer eine schließende geschweifte Klammer folgen}. Dies ist eine Bedingung, die oft als “ausgeglichene Zahnpangen” bezeichnet wird. Die Arduino IDE (Integrated Development Environment) enthält eine praktische Funktion zum Überprüfen des Gleichgewichts von geschweiften Klammern. Wählen Sie einfach eine Klammer aus oder klicken Sie auf die Einfügemarkie unmittelbar nach einer Klammer und der logische Begleiter wird optisch hervorgehoben.

Unausgeglichene Klammern können häufig zu kryptischen, undurchdringlichen Compiler-Fehlern führen, die in einem großen Programm manchmal schwer zu finden sind. Aufgrund ihrer unterschiedlichen Verwendung sind geschweifte Klammern auch für die Syntax eines Programms unglaublich wichtig und das Verschieben einer Klammer um ein oder zwei Zeilen wirkt sich häufig dramatisch auf die Bedeutung eines Programms aus.

It is also mandatory for a sketch to have a 'loop' function.

```
// the loop function runs over and over again forever
void loop() {
    digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
    delay(1000); // wait for a second
    digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
    delay(1000); // wait for a second
}
```

**Loop :** Im Gegensatz zur 'Setup'-Funktion, die nur einmal ausgeführt wird, wird die loop-Schleife ständig ausgeführt und damit fortwährend wiederholt.

**digitalWrite(Pin num , HIGH/LOW):** Schreiben Sie einen HIGH- oder LOW-Wert in einen digitalen Pin.

Wenn der Pin mit pinMode () als OUTPUT konfiguriert wurde, wird seine Spannung auf den entsprechenden Wert eingestellt: 5 V für HIGH, 0 V (Masse) für LOW.

**delay(ms):** Hält das Programm für die als Parameter angegebene Zeit (in Millisekunden) an. Dabei stellen 1000 Millisekunden eine Sekunde dar.)

**ms :** Die Anzahl der Millisekunden, die angehalten werden soll (Bereich: 0 ~ 4394967295).

Innerhalb der loop-Funktion wird durch die Befehle der LED-Pin erstmal angeschaltet (HIGH). Darauf folgt eine 1000 Milisekunden (= 1 Sekunde) lange Pause. Dann wird der LED-Pin wieder ausgeschaltet (LOW) und es findet wieder eine 1-sekündige Pause statt. Jetzt werden Sie die LED dazu bringen schneller zu blinken. Wie Sie bereits geahnt haben könnten, liegt der Schlüssel der Blinkgeschwindigkeit darin, die Zeiten der Pausen anzupassen (die Zahlen innerhalb der Klammern der „delay“-Befehle).

```
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 volt
33   delay(500) ←                      // wait for a second
34   digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the vo
35   delay(500) ←                      // wait for a second
36 }
```

Die Zeitspanne der Pausen wird in Milisekunden angegeben (1s = 1000ms). Wenn Sie also die LED doppelt so schnell blinken lassen wollen, müssen Sie die Werte halbieren, also von 1000 auf 500 ms senken. Das sorgt dafür, dass zwischen Ein- und Ausschalten der LED nur noch jeweils eine halbe Sekunde Pause stattfindet. Laden Sie den Sketch erneut hoch und am Ende sollten Sie die LED schneller blinken sehen. Sie können die Werte nun beliebig anpassen und den Sketch erneut hochladen und beobachten, wie sich die Blinkgeschwindigkeit jedes mal ändert.

## Zusätzliche Arduino Bibliotheken einbinden

Wenn Sie mit der Arduino Entwicklungsumgebung und dessen Funktionen vertraut sind, möchten Sie möglicherweise die Möglichkeiten Ihres Arduinos mit zusätzlichen Bibliotheken erweitern.

### Was sind Bibliotheken?

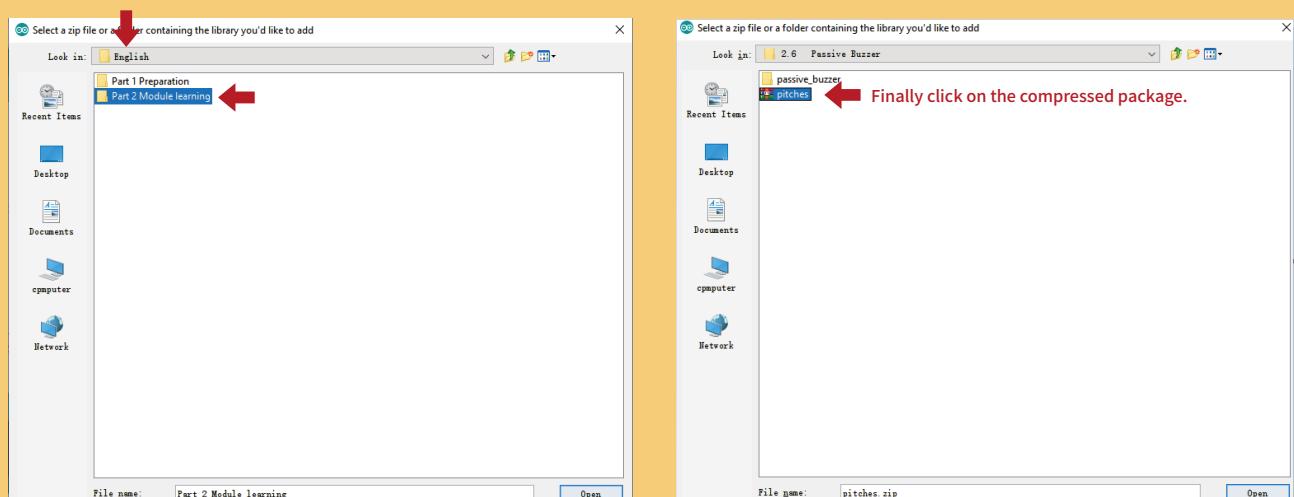
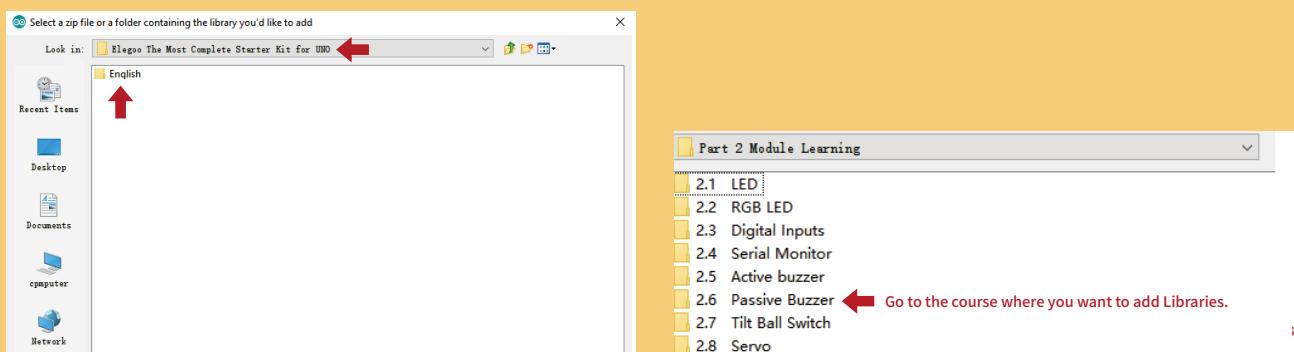
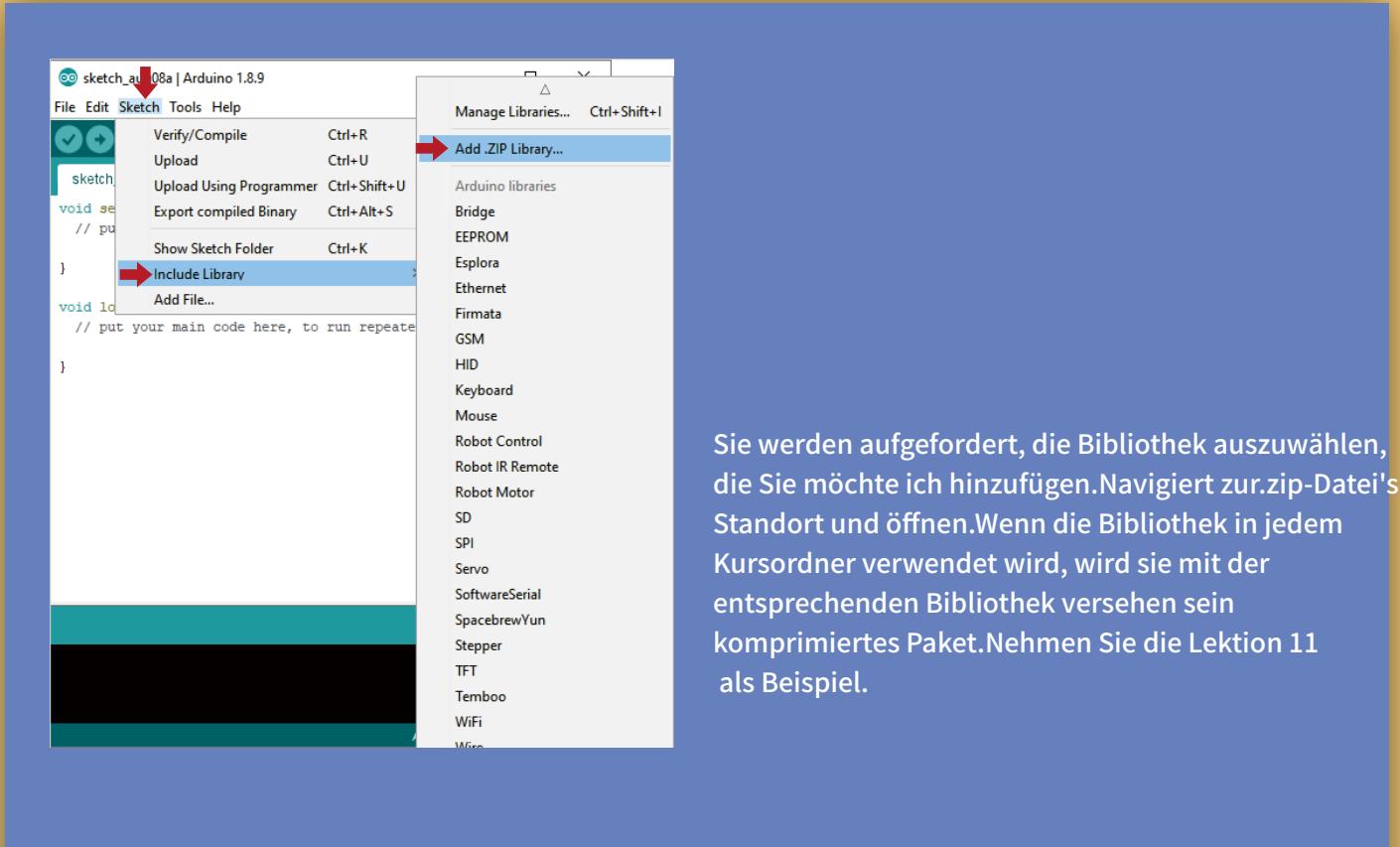
Bibliotheken sind Sammlungen von vorprogrammiertem Code, die es enorm einfach für Sie machen beispielsweise einen Sensor, ein Display oder ein Modul einzubinden. Zum Beispiel vereinfacht die vorinstallierte LiquidCrystal Bibliothek die Kommunikation zu LCD-Zeichenanzeigemodulen. Es gibt hunderte von zusätzlichen Bibliotheken für Arduino im Internet, die Sie sich einfach herunterladen können. Die mit der Arduino IDE mitkommenden Bibliotheken und einige zusätzliche Bibliotheken sehen Sie später zum Vergleich. Um zusätzliche Bibliotheken benutzen zu können, müssen diese erst installiert werden.

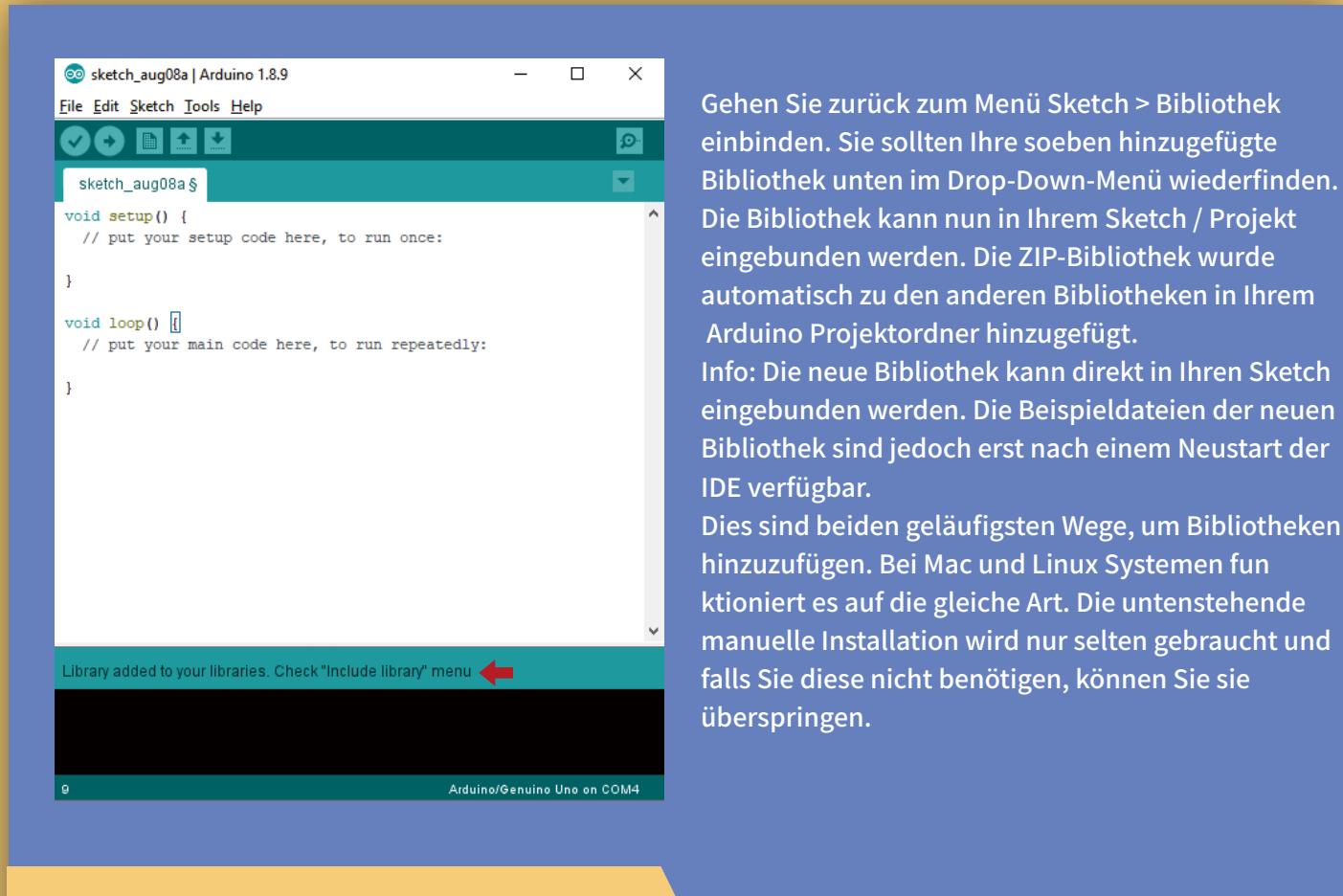
### Wie installiert man eine Bibliothek?

#### 1Eine .zip Bibliothek einbinden

Weitere Bibliotheken werden im Internet häufig als ZIP Dateien zum Download angeboten. Der Name des ZIP Archivs ist meist der Name der Bibliothek. Im Archiv befindet sich eine .cpp Datei, eine .h Datei und oft eine keywords.txt Datei und ein examples Ordner mit Code-Beispielen und andere Dateien, die von der Bibliothek benötigt werden. Seit Version 1.0.5 können Drittanbieter-Bibliotheken in der Arduino IDE eingebunden werden. Entpacken Sie die ZIP Dateien der Bibliotheken nicht selber, sondern belassen Sie sie so wie sie sind. Die IDE wird das Archiv automatisch entpacken.

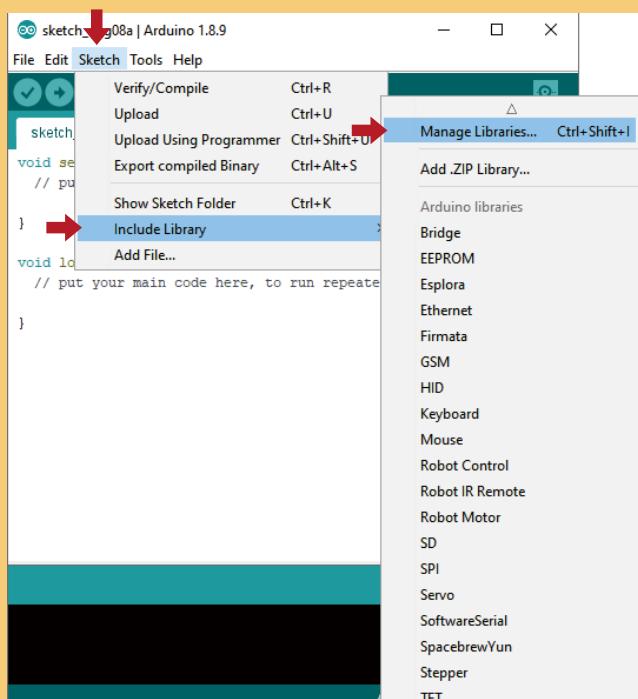
Navigieren Sie in der IDE zu Sketch > Bibliothek einbinden und wählen Sie „.ZIP Bibliothek hinzufügen“





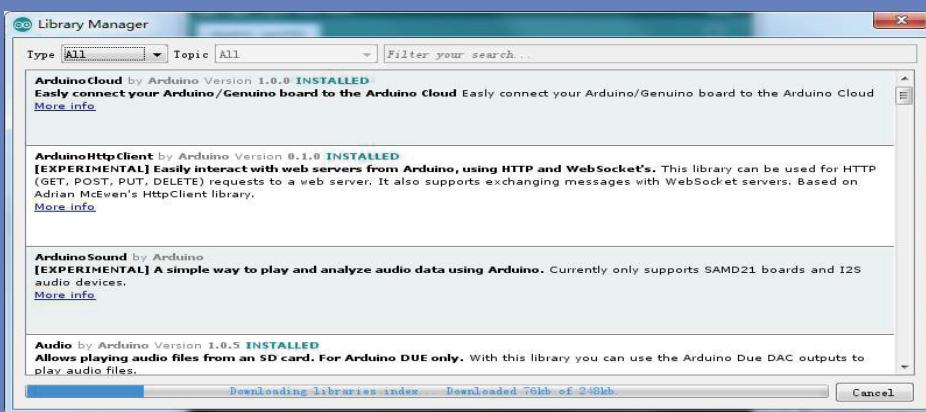
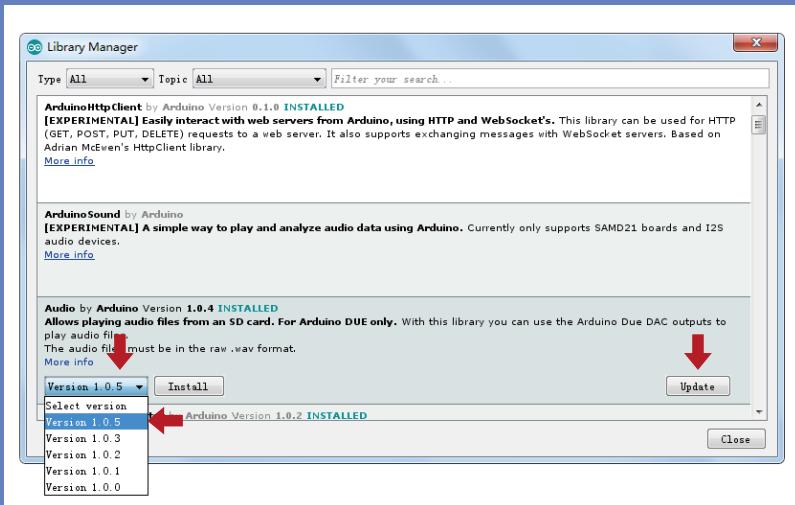
## 2) Mit Hilfe des Bibliotheksverwalters

Um eine neue Bibliothek in Ihre Arduino IDE einzubinden, können Sie den Bibliotheksverwalter nutzen (verfügbar seit Arduino IDE 1.8.9). Öffnen Sie die IDE und klicken unter Sketch > Bibliothek einbinden auf „Bibliotheken verwalten“.



Dann wird sich der Bibliotheksverwalter öffnen und Sie finden eine Liste voller Bibliotheken, die bereits installiert sind oder heruntergeladen werden können. Als Beispiel werden wir die Bibliothek Audio installieren. Scrollen Sie die Liste herunter und suchen Sie die Bibliothek. Dann wählen Sie die Version aus, die Sie installieren wollen. Manchmal gibt es nur eine verfügbare Version und es lässt sich keine Version auswählen. Keine Sorge: Das ist normal.

Manchmal müssen Sie ein bisschen geduldig sein, da es sehr viele Bibliotheken gibt. Lassen Sie die Software die Liste aktualisieren und suchen Sie dann den Eintrag



Klicken Sie zum Schluss auf Installieren und warten Sie, bis die Bibliothek installiert ist. Das Herunterladen kann abhängig von Ihrer Internetverbindungs geschwindigkeit etwas dauern. Sobald die Installation abgeschlossen ist, sollte die Bibliothek im Verwalter den Zusatz „Installiert“ haben. Danach können Sie den Bibliotheksverwalter wieder schließen.



Jetzt können Sie die Bibliothek auch schon über das Einbindungs menü hinzufügen. Wenn Sie Ihre eigene Bibliothek im Bibliotheksverwalter hinzufügen möchten, erstellen Sie einen Eintrag auf der Github-Seite von Arduino.

### 3) Manuelle Installation

Beenden Sie zuerst die Arduino Software, bevor Sie die Bibliothek manuell installieren. Dann entpacken Sie das ZIP-Archiv der Bibliothek. Wenn Sie zum Beispiel eine Bibliothek namens "ArduinoParty" heruntergeladen haben, entpacken Sie die ArduinoParty.zip Datei. Der entpackte Ordner sollte einen Ordner namens ArduinoParty enthalten, in welchem sich wiederum Dateien wie ArduinoParty.cpp und ArduinoParty.h befinden sollten. Wenn die .cpp und .h Dateien nicht in einem separaten Ordner liegen, müssen Sie einen erstellen. In diesem Fall würden Sie einen Ordner namens „ArduinoParty“ erstellen und verschieben alle Dateien, die im Zip-Archiv waren, in diesen Ordner (wie ArduinoParty.cpp und ArduinoParty.h). Verschieben Sie den ArduinoParty Ordner nun in Ihren Bibliotheksordner. Unter Windows und Mac ist dies standardmäßig unter Dokumente\Arduino\libraries . Unter Linux ist es ebenfalls libraries Ordner in Ihrem Projektordner.

Ihr Arduino Bibliotheks Ordner sollte nun so aussehen (unter Windows):

[My Documents\Arduino\libraries\ArduinoParty\ArduinoParty.cpp](#)

[My Documents\Arduino\libraries\ArduinoParty\ArduinoParty.h](#)

[My Documents\Arduino\libraries\ArduinoParty\examples](#)

oder so (unter Mac und Linux):

[Documents/Arduino/libraries/ArduinoParty/ArduinoParty.cpp](#)

[Documents/Arduino/libraries/ArduinoParty/ArduinoParty.h](#)

[Documents/Arduino/libraries/ArduinoParty/examples](#)

....

Es können mehr Dateien als nur die .cpp and .h Dateien vorhanden sein, gehen Sie nur sicher, dass sich alle Dateien in Ihrem Ordner befinden. Die Bibliothek wird nicht funktionieren, wenn sich die .cpp und .h Dateien direkt im libraries Ordner oder in einem weiteren Unterordner befinden. Zum Beispiel:

Dokumente\Arduino\libraries\ArduinoParty.cpp und

Dokumente\Arduino\libraries\ArduinoParty\ArduinoParty.cpp

würden nicht funktionieren.

Wenn alles richtig ist, starten Sie die Arduino IDE neu. Stellen Sie sicher, dass die neue Bibliothek unter Sketch > Bibliothek einbinden zu finden ist. Dann sind Sie fertig.