Lektion 2: Blink

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.

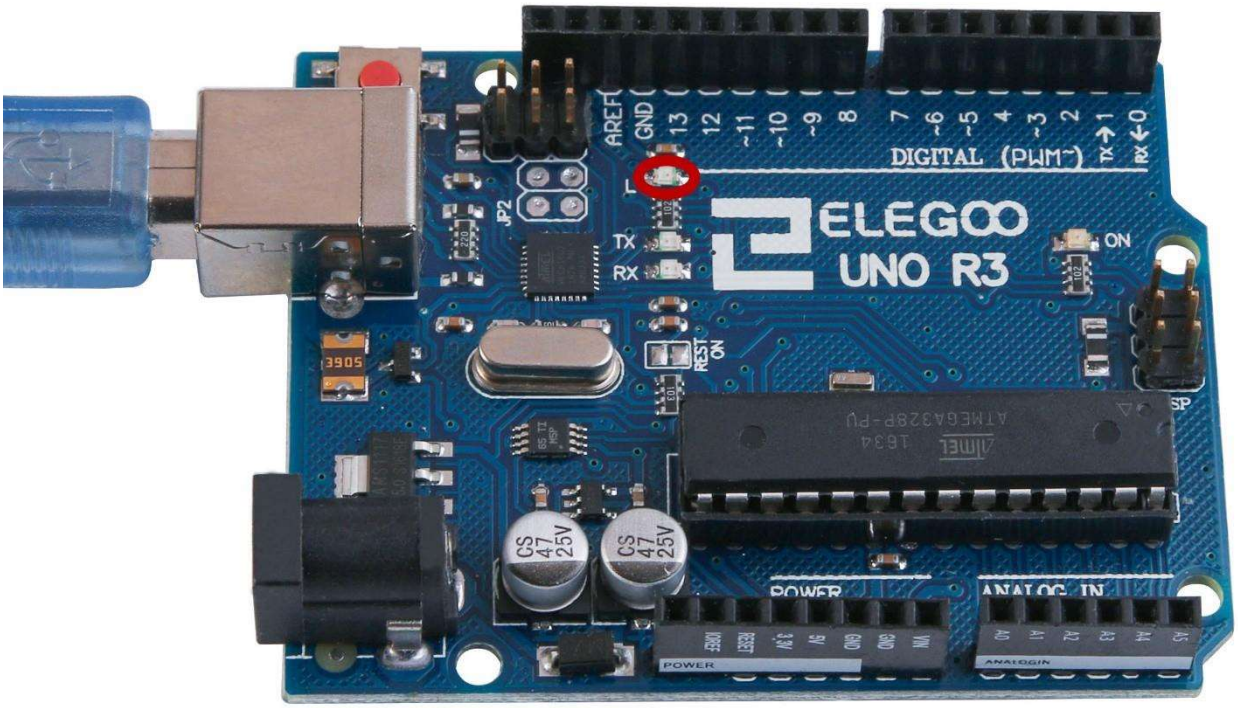
Benötigte Bauteile:

(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.



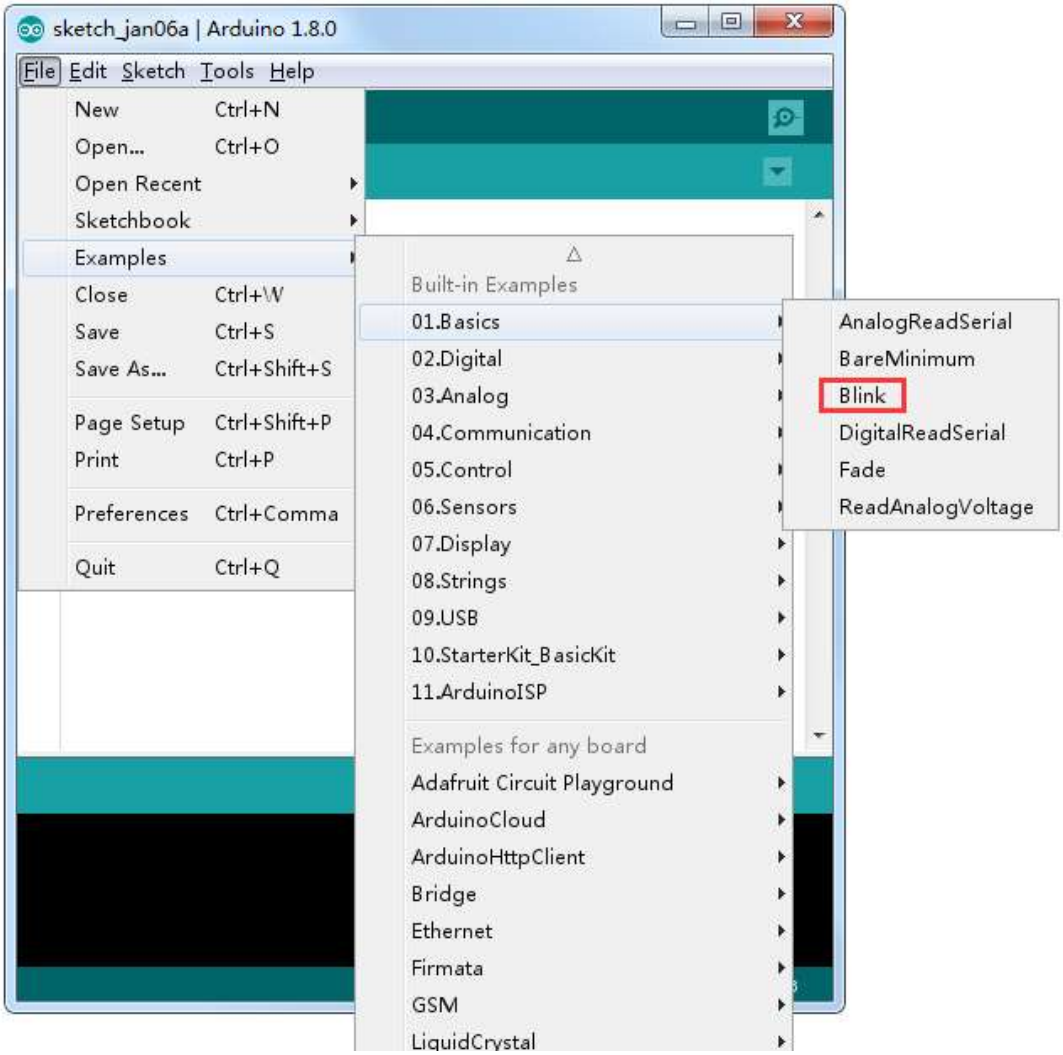
Es kann sein, dass Ihre „*L*“-LED bereits blinkt, wenn Sie Ihr UNO R3 Board mit einer Stromversorgung verbinden. Das liegt daran, dass die ausgelieferten Boards bereits mit dem „*Blink*“-Sketch vorinstalliert kommen, der die LED blinken lässt.

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

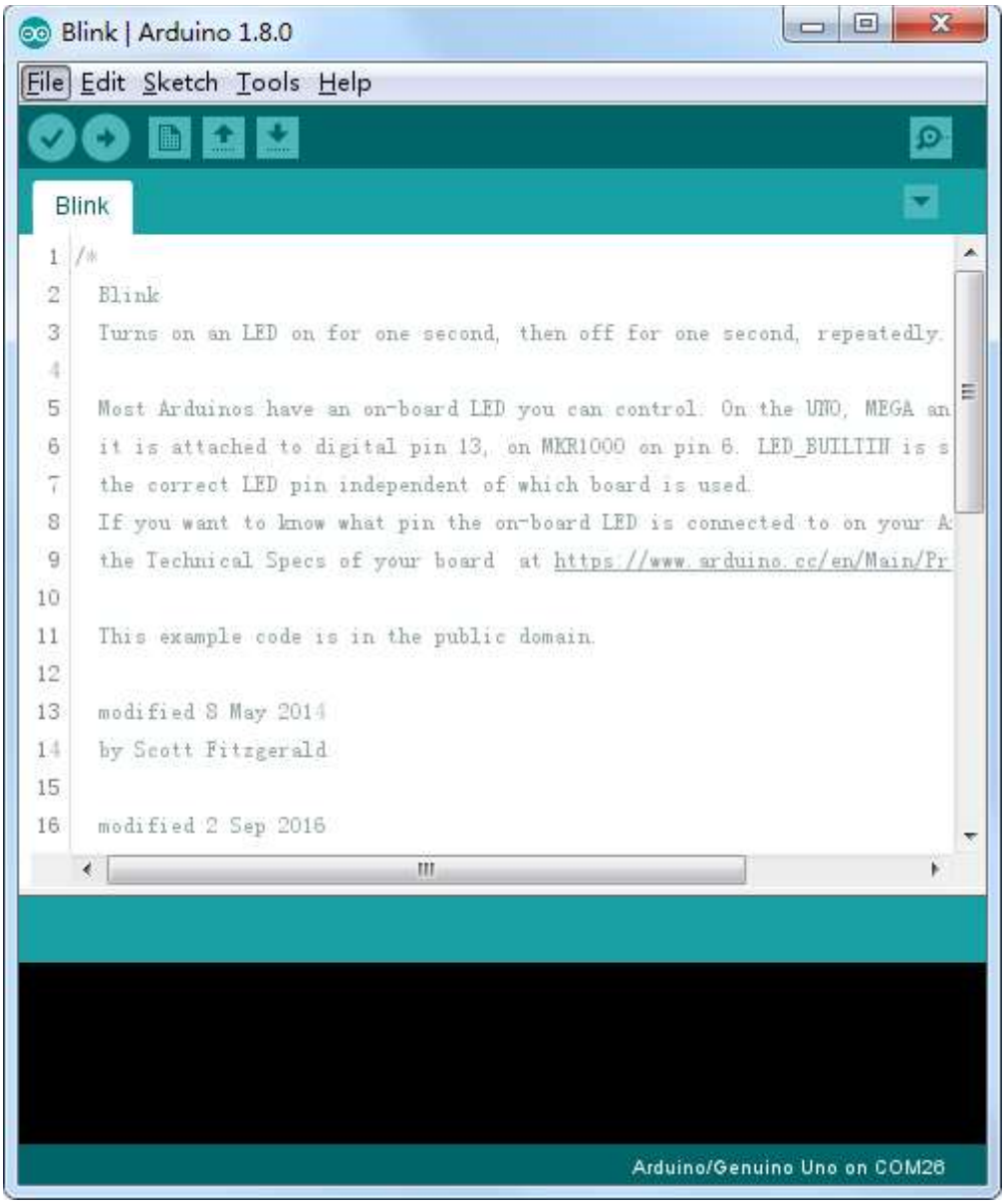
In Lektion 0 haben Sie die Arduino IDE (Entwicklungsumgebung) aufgesetzt und den richtigen Seriellen Port in der IDE eingestellt. Nun werden wir diese Verbindung testen, indem wir das UNO R3 Board erstmals selber programmieren.

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.



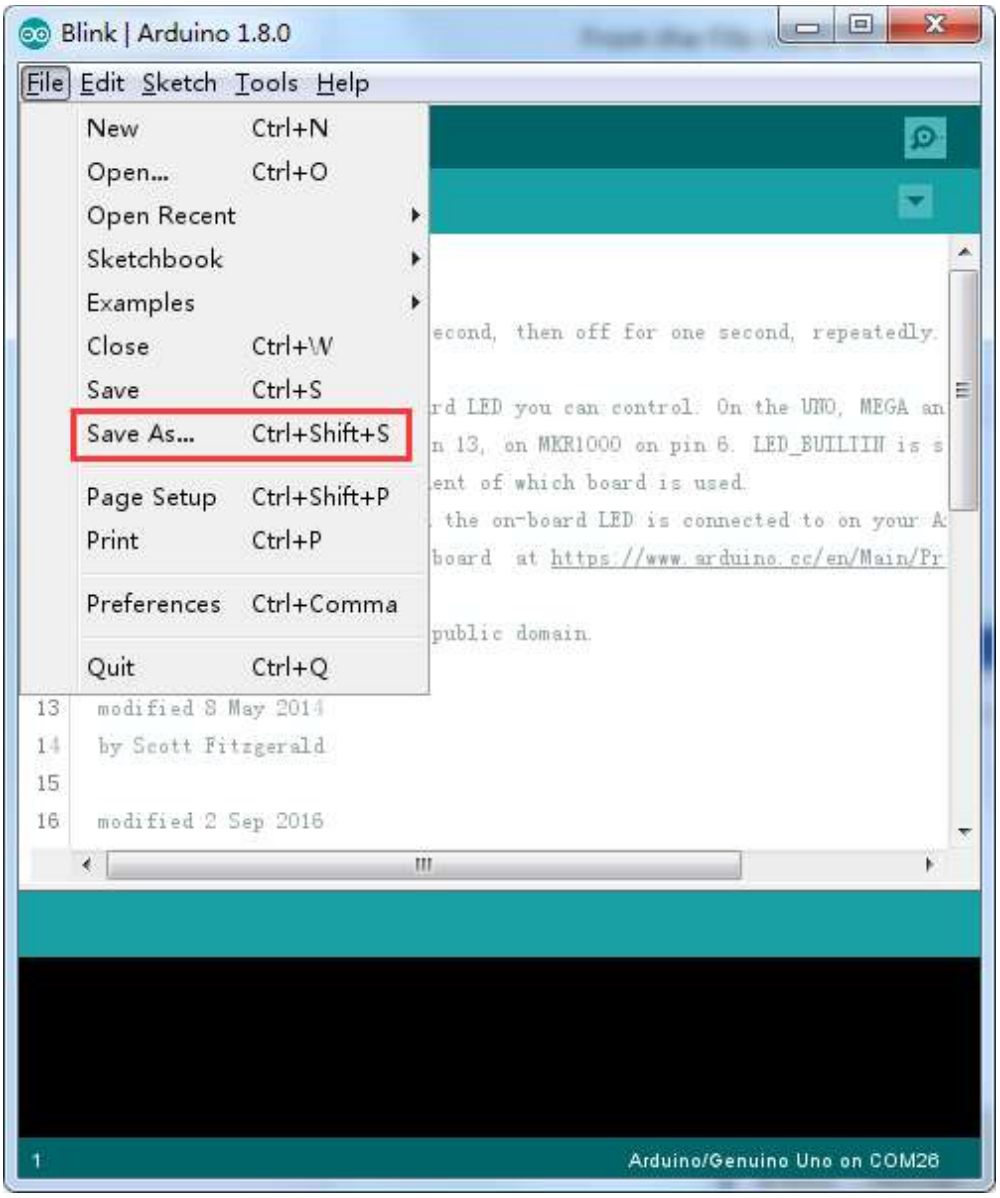
Wenn sich der Blink-Sketch geöffnet hat, sollten Sie das Fenster vergrößern, um sich einen Überblick über den gesamten Programm-Code schaffen zu können.



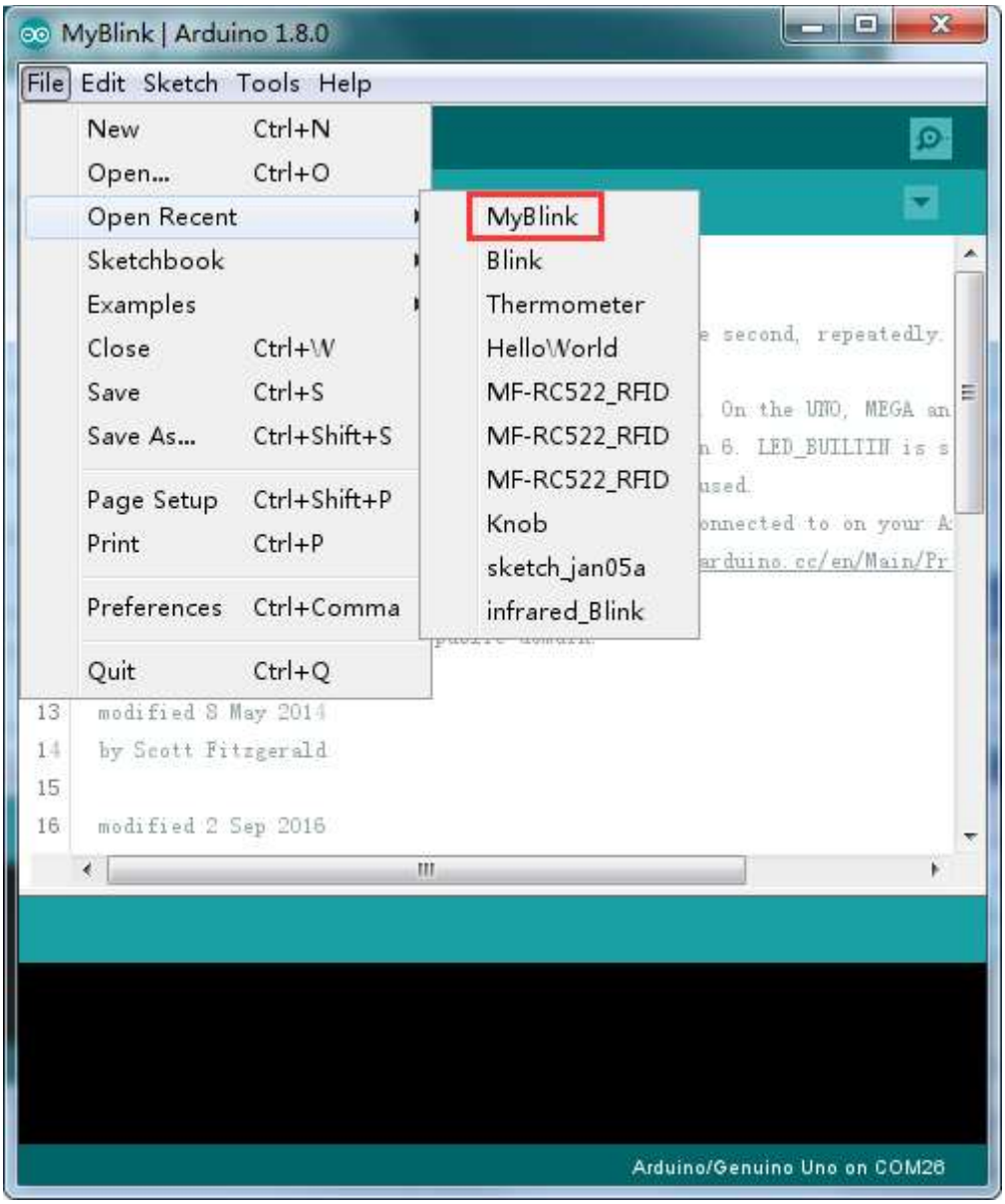
Die mitinstallierten Beispiel-Sketches sind nicht „schreibbar“. Das heißt, man kann Änderungen nicht abspeichern. Sie können den Sketch abändern und hochladen, aber nicht unter dem gleichen Dateinamen speichern.

Da wir den Sketch ändern möchten, ist der erste Schritt den Sketch unter neuem Namen abzuspeichern.

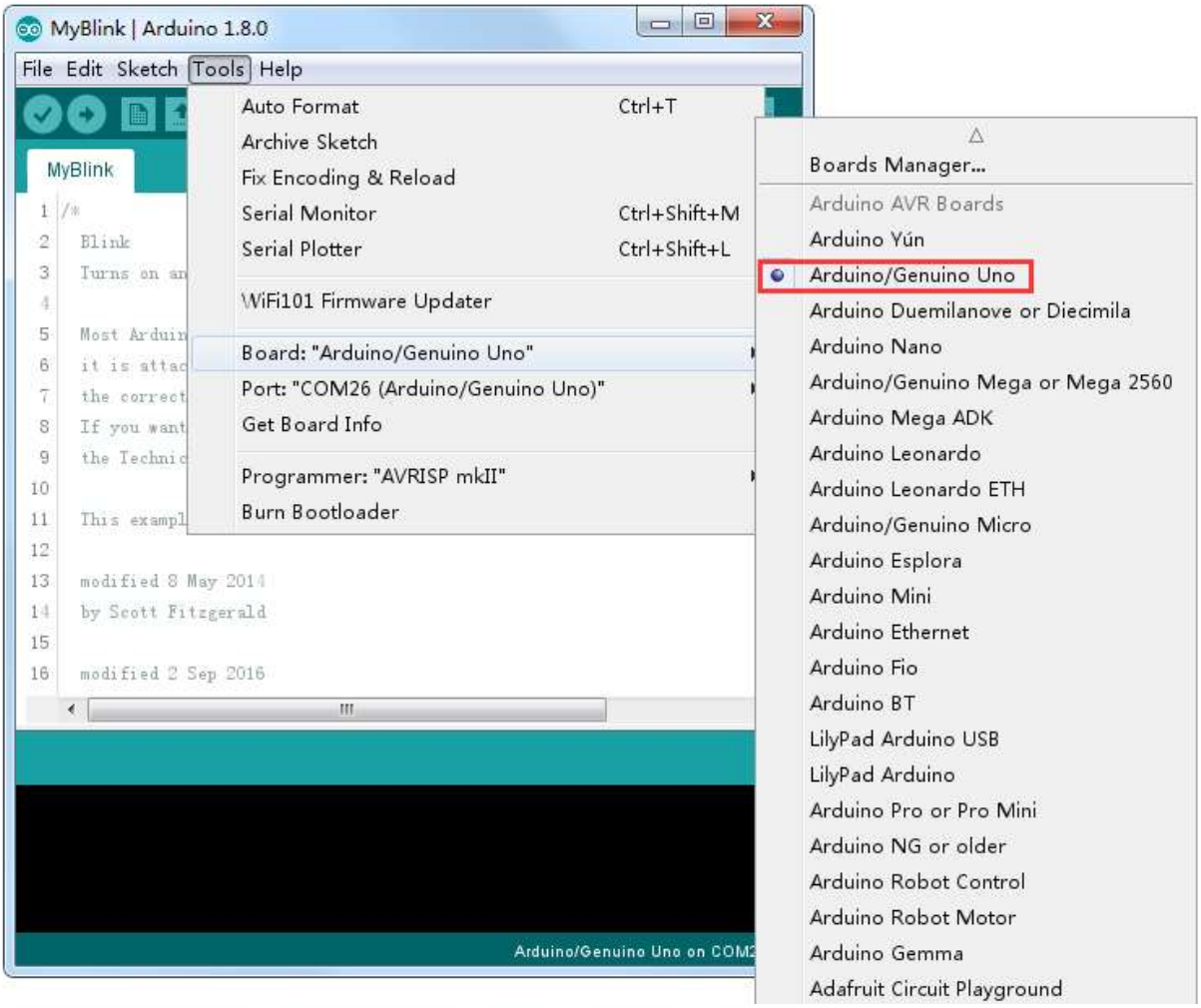
Im Arduino IDE Menü wählen Sie *Datei > Speichern unter…* aus und speichern den Sketch dann unter dem Namen „*MyBlink*“.

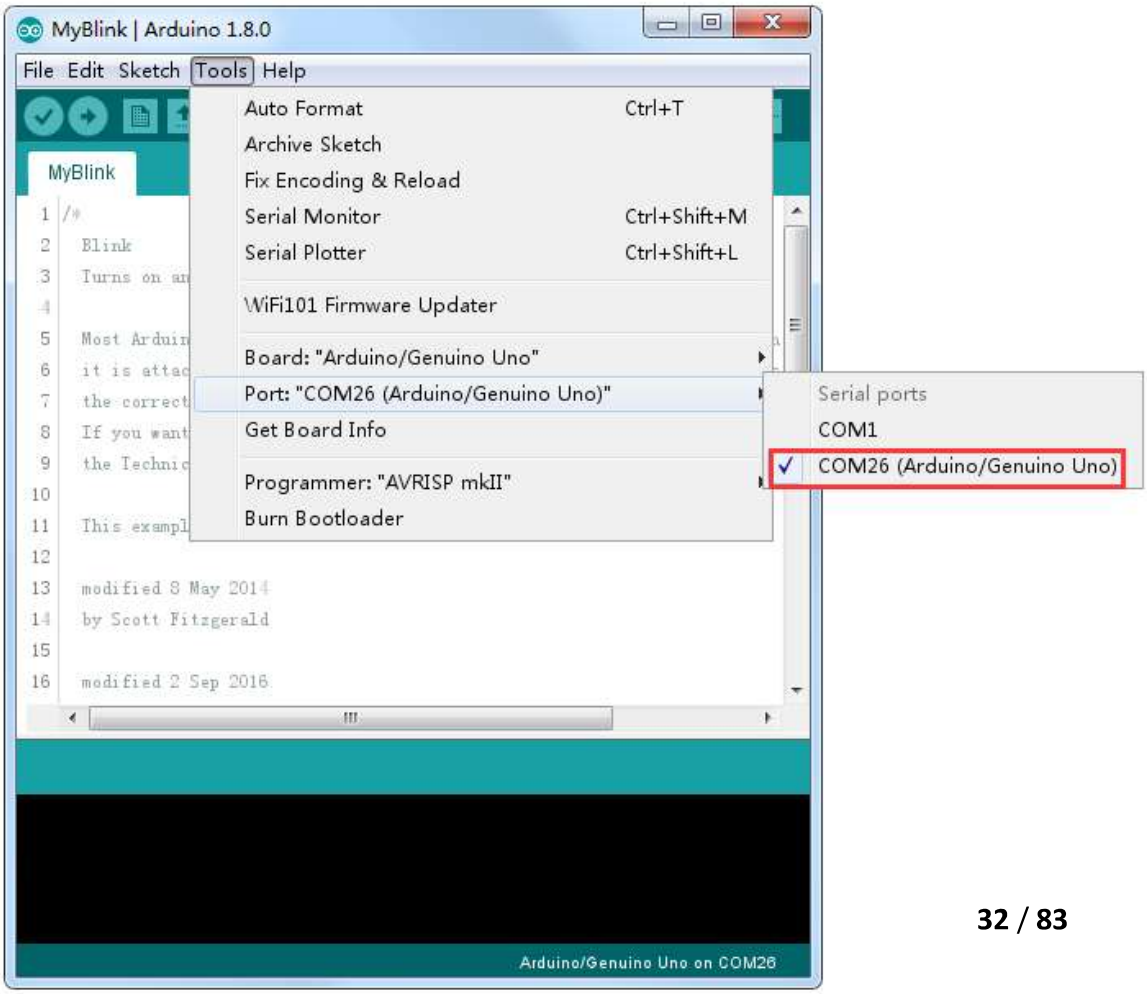

Blink | Arduino 1.8.0
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
modified 8 May 2014
by Scott Fitzgerald
modified 2 Sep 2016
Arduino/Genuino Uno on COM28

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 eigestellt sind,

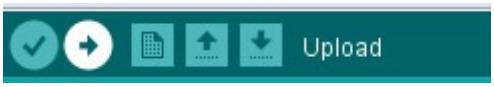

MyBlink | Arduino 1.8.0
File Edit Sketch Tools Help
MyBlink
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino/Genuino Uno"
Port: "COM26 (Arduino/Genuino Uno)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Boards Manager...
Arduino AVR Boards
Arduino Yún
Arduino/Genuino Uno
Arduino Duemilanove or Diecimila
Arduino Nano
Arduino/Genuino Mega or Mega 2560
Arduino Mega ADK
Arduino Leonardo
Arduino Leonardo ETH
Arduino/Genuino Micro
Arduino Esplora
Arduino Mini
Arduino Ethernet
Arduino Fio
Arduino BT
LilyPad Arduino USB
LilyPad Arduino
Arduino Pro or Pro Mini
Arduino NG or older
Arduino Robot Control
Arduino Robot Motor
Arduino Gemma
Adafruit Circuit Playground
modified 8 May 2014
by Scott Fitzgerald
modified 2 Sep 2016
Arduino/Genuino Uno on COM


MyBlink | Arduino 1.8.0
File Edit Sketch Tools Help
MyBlink
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino/Genuino Uno"
Port: "COM26 (Arduino/Genuino Uno)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM1
COM26 (Arduino/Genuino Uno)
modified 8 May 2014
by Scott Fitzgerald
modified 2 Sep 2016
Arduino/Genuino Uno on COM26

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.

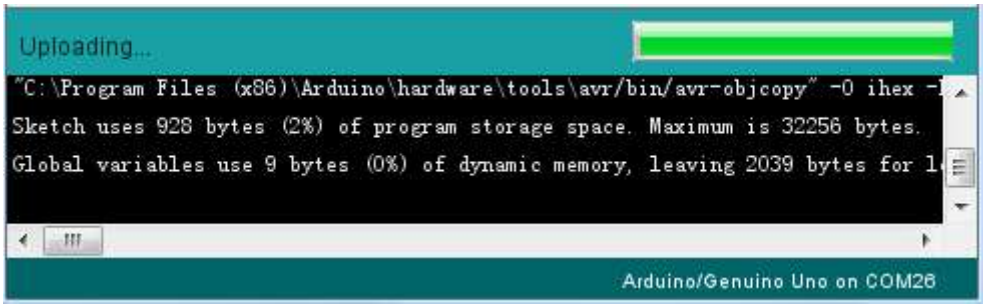
Die Arduino IDE zeigt die aktuellen Einstellungen für das Board im Fenster rechts unten an.

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

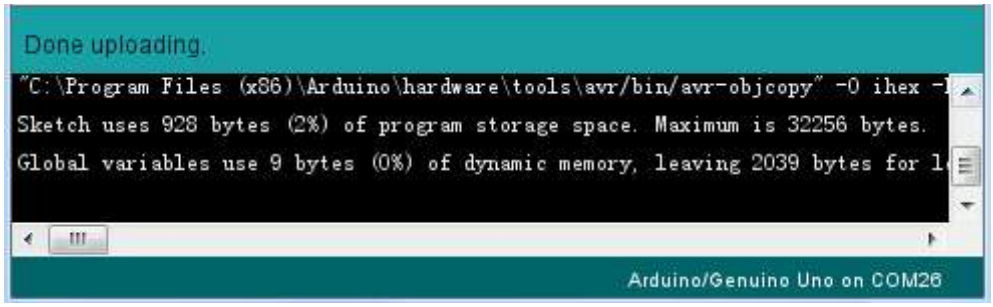


Wenn Sie den Statusbereich der IDE in der unteren Sektion des Fensters beobachten, sehen Sie, wie sich eine Fortschrittsleiste langsam füllt und Meldungen erscheinen. Zuerst erscheint die Meldung *„Sketch wird kompiliert...“.* Dabei wird der Sketch in ein für das Board passende Format umgewandelt.

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 wird.



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.

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.
Ein-Zeilen-Kommentare starten mit // und alles von diesem Zeichen an bis hin zum Ende der Zeile wird als Kommentar interpretiert.
Die erste Zeile des Codes lautet:

```
int led = 13;
```

Wie der Kommentar darüber erklärt, weist dies dem Pin der LED einen Namen zu (*led*). Dieser Pin ist der Pin 13 auf den meisten Arduinos, wie beispielsweise UNO und Leonardo.
Als nächstes haben wir die „*setup*"-Funktion. Der Kommentar erklärt, dass alles was sich in dieser Funktion befindet ausgeführt wird, wenn der *Reset*-Knopf gedrückt wird. Es wird außerdem ausgeführt, wenn das Board aus irgendeinem anderen Grund resettet wird, wie zum Beispiel nach dem Einstecken der Stromversorgung oder nach dem Hochladen eines Sketches.

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

Jeder Arduino Sketch muss eine setup-Funktion haben. Der Code für diese Funktion muss zwischen den geschwungenen Klammern { } platziert werden.
In diesem Fall gibt es in der setup-Funktion nur einen einzigen Befehl, welcher bestimmt, dass das Arduino Board den LED-Pin als einen Ausgang benutzen soll.
Außerdem grundlegend für jeden Sketch ist die „*loop*"-Funktion. Im Gegensatz zur setup-Funktion, die nur einmalig nach einem Reset läuft, wiederholt sich die loop-Funktion ständig, nachdem Sie alle ihre Befehle ausgeführt hat (loop = Schleife).

```
void loop() {
  digitalWrite(led, HIGH);    // turn the LED on (HIGH is the voltage level)
  delay(1000);                // wait for a second
  digitalWrite(led, LOW);     // turn the LED off by making the voltage LOW
  delay(1000);                // wait for a second
}
```

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 angegebn (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.