# Smartphone-gesteuerter LED-Scheinwerfer

*Idee, techn. Leitung, Programmierung: Prof. Dr.-Ing. Tobias Ellermeyer; Konstruktion: M.Sc. Moritz Schulte*


**Erlebe das Internet der Dinge (*IoT* = Internet of Things) live, indem du einen über das Internet bzw. ein Smartphone steuerbaren farbigen LED-Scheinwerfer baust. Diese Anleitung erklärt Schritt für Schritt den Zusammenbau und das Aufspielen der Software. Zusätzlich erfährst du, wie du diesen Scheinwerfer z.B. auch in dein WLAN-Netz zu Hause einbinden kannst. Und es gibt reichlich Hintergrund-Infos, die für den Zusammenbau zwar nicht benötigt werden, aber hoffentlich dein Interesse wecken, wie die Dinge funktionieren.**

## Einleitung

Die gesamte Industrie entwickelt sich immer weiter hin zu vernetzten Herstellungsprozessen mit individuellen Produkten. Was du vom Pkw schon kennst, nämlich dass man sich Farbe, Motoren und zahlreiches Sonderzubehör auswählen kann, wird auch bald bei einfachen Konsumgütern Realität. So hat z.B. der Sportschuh-Hersteller Adidas eine Fabrik in Betrieb genommen, wo für den Kunden individuell angefertigte Schuhe (vom Aussehen, Applikationen bis hin zur an das Fußbett angepassten Sohle) bestellen kann. Diese neue Art, Dinge individuell und automatisiert zu fertigen wird als die 4. Industrielle Revolution oder kurz **Industrie 4.0** bezeichnet.

Eng damit verbunden ist das sogenannte **Internet der Dinge (IoT)**, welches Alltagsgegenstände internettauglich macht. Beispiele hierfür sind Kühlschränke mit Internetanschluss, IP-Kameras, Smart-Home Steuerungen, usw.

Gerade die **Mechatronik**, welche aus den Teilgebieten Maschinenbau, Elektrotechnik und Informatik besteht, profitiert von diesem Trend, da sie alle erforderlichen Teilgebiete abdeckt.

Um dir einen Einblick in diese Teilgebiete zu geben, haben wir ein Projekt erstellt, welches sowohl aus Mechanik, als auch aus Elektronik und Programmieren (Informatik) besteht. Natürlich sind schon viele Dinge entsprechend vorbereitet, damit die Zeit reicht.

Aber am Ende eines erfolgreichen Studiums der Mechatronik kannst du derartige Aufbauten selbst entwickeln und natürlich auch vielfältige andere Problemstellungen aus der Industrie lösen.

**Do not look into Pixie LED with the remaining eye!**

**ACHTUNG:
Nicht direkt aus kurzer Entfernung in die LED schauen!**

## Bauteile

Rechts sind die eingesetzten Bauteile abgebildet, die im Folgenden kurz vorgestellt werden. Eine genauere Auflistung der Bauteile findest du am Ende dieses Dokuments.

Die Intelligenz des gesamten Aufbaus steckt in einem Mini-Computer namens **NodeMCU** V1.0 „Amica“ mit einem ESP8266 Prozessor. Dieser Computer sieht sicherlich nicht so aus, wie du es erwartest, vor allem da er keinen Bildschirm und keine Tastatur hat. Auch ist der Prozessor längst nicht so leistungsfähig wie du es von deinem Smartphone kennst und hat auch wesentlich weniger Speicher.

Der ESP8266-Prozessor läuft mit einer Taktfrequenz von 0,08 GHz, während dein Smartphone vermutlich mehrere Rechenkerne hat, die je nach Modell mit über 2 GHz Taktfrequenz arbeiten; das heißt, die Rechenleistung beträgt nur ungefähr 1/50 der deines Smartphones.

Auch der Speicher ist recht knapp: Arbeitsspeicher (RAM) 0,064 MB und Datenspeicher (Flash) 4 MB im Vergleich zum Smartphone mit 1024 … 8092 MB = 1 … 8 GB Arbeitsspeicher und 8 … 256 GB Datenspeicher.

Wesentlicher Vorteil dieses kleinen Computers ist sein Preis von ca. 7 € und die Tatsache, dass er sich durch seinen Aufbau hervorragend für „Basteleien“ eignet. **WLAN** ist bei diesem Preis auch schon an Board, und die NodeMCU kann sich wahlweise in ein bestehendes Netz einbuchen, oder aber einen eigenen Accesspoint aufmachen. Mit Spannung versorgt und auch programmiert wird die NodeMCU durch einen Micro-USB-Stecker.

Als LED kommt eine **Pixie-LED** von Adafruit zum Einsatz. Diese LED kann aus den Grundfarben Rot, Grün und Blau beliebige Farben mischen. Mehr dazu, wie dies funktioniert, findest du im Anhang. Die LED hat eine maximale Leistung von 3 Watt. ***Wichtig: Nicht direkt aus kurzer Entfernung in die eingeschaltete LED schauen.***

Im Prinzip handelt es sich bei einem **Servo** um einen kleinen Motor mit Getriebe, welcher eine Welle antreibt.

Zusätzlich enthält der Servo jedoch eine kleine Elektronik, so dass man von außen eine Position (genauer einen Winkel) vorgeben kann, auf die die Welle dann gestellt wird.

Die hier verwendeten Servos haben einen Einstellbereich von etwa 180°.

Die Schwenkfunktion wird durch zwei Miniatur-**Servos** vom Typ SG90 realisiert. Diese stammen eigentlich aus dem Modellbau und werden dort z.B. für die Steuerung von Leitwerken in ferngesteuerten Flugzeugen eingesetzt.

Weiterhin werden noch ein **Halter** für die Pixie-LED und den zweiten Servo benötigt. Diese wurden an der Fachhochschule mit einem 3D-Drucker hergestellt.

Damit das ganze schön aussieht, brauchen wir noch ein **Gehäuse**. Hier wurde Sperrholz als Baumaterial gewählt, welches mit einem Laser in die passende Form geschnitten und auch beschriftet wurde. Auch dieser Laser ist an der Fachhochschule vorhanden. Befestigt werden die einzelnen Holzteile durch sogenannte Schwalbenschwänze, die durch Löcher gesteckt werden, und auf die anschließend zur Sicherung ein Gummiring gespannt wird.

Um die Kabel von der LED und den zwei Servos vernünftig an die NodeMCU anschließen zu können, hat eine fleißige Studentische Hilfskraft noch eine **Verbindungsplatine** gelötet. Zu guter Letzt: für solch ein Projekt wird natürlich eine gehörige Portion **Ingenieurgeist** benötigt☺.

*NodeMCU Board*

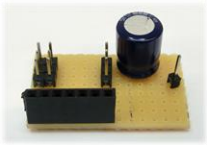
*Pixie 3W RGB LED*

*Servos SG90*

*Halter Pixie LED*

*Servo Halter*

*Gehäuse*



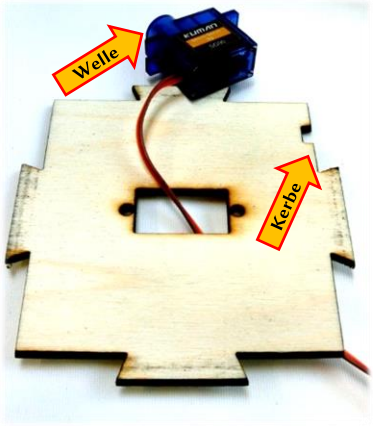
*Verbindungsplatine*


*Ingenieurgeist*

## Mechanischer Zusammenbau

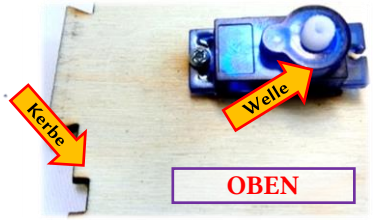
Gestartet wird mit dem mechanischen Zusammenbau des LED-Scheinwerfers. Hierzu brauchst du an einigen Stellen etwas Geschick und Geduld, insbesondere bei den kleinen Schrauben.

Als erstes montierst du den Servo in der Grundplatte. Stecke dafür das Kabel durch die Öffnung. Die Welle des Servos muss von der Auskerbung weg zeigen (Pfeile).

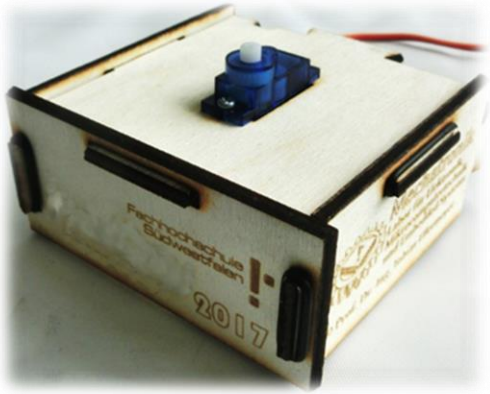



Dann wird der Servo mit den gezeigten Gewindeschrauben M2 befestigt (s. Bilder unten). Die Schraube wird von oben durchgesteckt, und von unten wird erst eine Unterlegscheibe und anschließend die Mutter aufgesetzt. Bei Bedarf kannst du eine Pinzette oder Zange zur Hilfe nehmen.

Erst eine Schraube befestigen, nur leicht anziehen und dann die zweite Schraube ebenfalls montieren. Zum Schluss beide Schrauben anziehen (nicht zu fest).







Jetzt montierst du zuerst die beiden Seitenteile links und rechts (Beschriftung „Mechatronik“). Jeweils einen großen Gummiring über den „Schwalbenschwanz“ ziehen, um die Bauteile zu fixieren.

Danach baust du die vordere Seitenplatte (Beschriftung „Labortag“) an. Diese Platte muss nur mit zwei Ringen links und rechts befestigt werden.

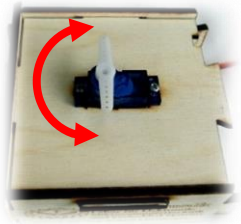
Die hintere Platte wird zunächst nicht montiert.

Bevor der Halter für den zweiten Servo montiert werden kann, muss der Servo zunächst in die richtige Position gedreht werden. Die hier verwendeten Servos haben einen Stellbereich von etwas mehr als 180°, d.h. die Welle kann maximal um ca. eine halbe Umdrehung gedreht werden.

Um die Welle drehen zu können, benutzt du den (ansonsten nicht benötigter) Servoarm und steckst ihn auf die Welle, ohne ihn festzuschrauben und führst die aufgeführten Einstell-Schritte aus:

**1. Schritt**: Servo im Uhrzeigersinn in den Anschlag drehen

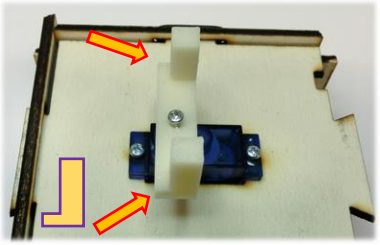
**2. Schritt**: Nun etwas (ca. 3 Minuten auf einem Ziffernblatt) nach links zurückdrehen.

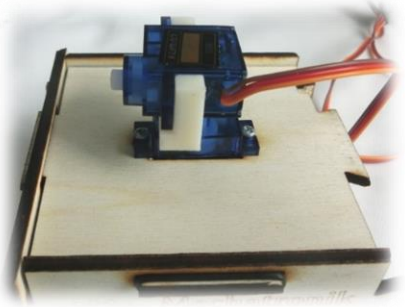


**3. Schritt**: Servoarm parallel zur Vorderseite aufstecken, prüfen, ob dieser sich um 180° bewegen lässt. Zuletzt wieder in die Ausgangsposition (d.h. 12 Uhr) drehen.

Nun nimmst du den Servoarm wieder ab (ohne die Achse zu verdrehen) und steckst stattdessen den Halter für den zweiten wie rechts gezeigt auf. Halte den Servo beim Aufdrücken des Halters von unten fest.

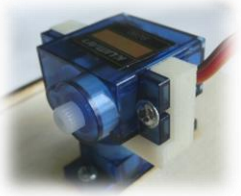
Achte auf die Ausrichtung (s. Pfeile). Anschließend mit der kleinen Schraube festschrauben.





Jetzt kann der zweite Servo montiert werden. Achte genau darauf, wie er im Bild platziert ist, vor allem, auf welcher Seite das Kabel ist.

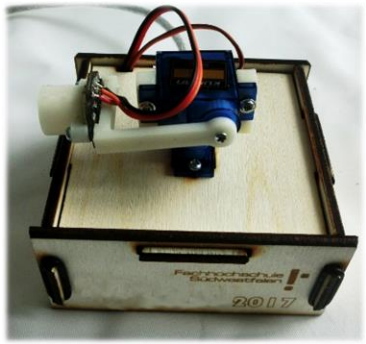
Schraube den Servo dann mit den beiden Schrauben mit dem breiteren Kopf fest.



Als nächstes muss noch die LED an dem zweiten Servo befestigt werden.

Dazu baust du zuerst die LED in den Halter ein. Die benötigten Teile und wie sie zusammengehören siehst du auf der rechten Seite.

Setze zunächst eine Schraube ein und ziehe diese nur leicht fest (ca. 4 Umdrehungen, bis die Schraube „gepackt“ hat). Dann die zweite Schraube ansetzten und beide festziehen.

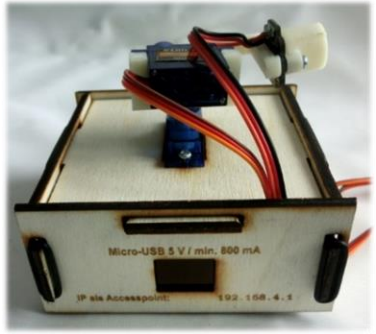


Nun wird der LED-Halter ebenfalls mit einer kleinen Schraube am Servo befestigt. Vorher bringst du den Servo wieder mit dem Servoarm in den linken Endanschlag (linkes Bild; hier ist die Rückwand bereits montiert, das kommt aber erst später…).

Prüfe, ob die LED sich um 180° bewegen lässt.

Damit das Kabel sich nicht von der LED löst, wird es jetzt noch mit einem Kabelbinder gesichert. Ziehe diesen erst wie rechts gezeigt fest.

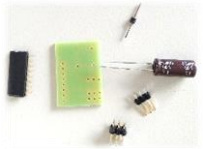
Anschließend knippst du den überstehenden Rest mit einem Seitenschneider ab.

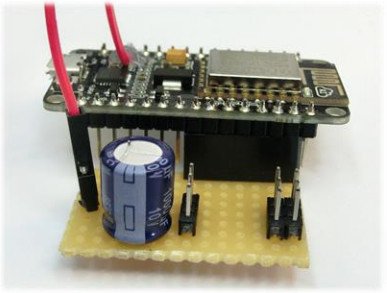



Jetzt erst wird das letzte Seitenteil aufgesetzt und mit zwei Gummiringen fixiert. Der obere Schwalbenschwanz kann wieder ohne Gummiring verbleiben.

Nun geht es an das Innenleben. Du benötigst die NodeMCU und die Anschlussplatine.

**WICHTIG: Die Anschlussplatine muss erst zusammengelötet werden. Hierbei hilft dir ein Mitarbeiter!** In der Box findest du nur die Einzelteile!





Stecke die Anschlussplatine so an die NodeMCU, dass der erste Kontakt auf der Buchsenleiste der Anschlussplatine mit dem **Pin D0 der NodeMCU** verbunden ist.

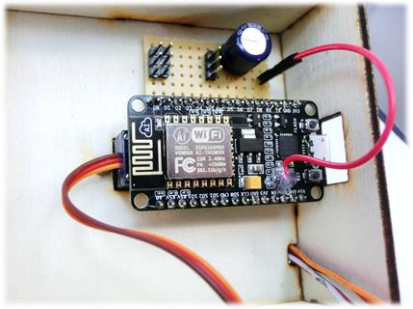
Weiterhin muss das rote Kabel wie gezeigt auf die Anschlussplatine gesteckt werden.

*(Durch dieses Kabel werden die +5 V vom USB-Port direkt auf die Servos und die LED gegeben. Da die NodeMCU keinen Pin hat, um diese Spannung abzugreifen, wurde ein zusätzliches Kabel angelötet)*

Jetzt klebst du ein Stück doppelseitiges Klebeband (die Rolle liegt aus) von unten auf den Servo.

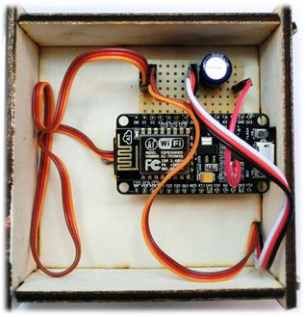
Entferne anschließend die Schutzfolie von dem Klebeband

Klebe dann die NodeMCU auf den Servo. Achte darauf, dass der Micro-USB-Stecker mittig im Loch sitzt. Die NodeMCU-Platine sollte auch auf der Steckerseite bündig am Holz anliegen.



Zum Abschluss werden die Servos und die LED mit der NodeMCU verbunden.

Beginne mit dem unteren Servo (der in der Holzplatte) und stecke dessen Anschlusskabel auf die Pins ganz am Rand (NodeMCU-Pin D1). Achte darauf, dass das orange Kabel zu der NodeMCU zeigt.



Als nächstes wird der zweite Servo (D2) und zuletzt die LED (D5) aufgesteckt. Hier muss das weiße Kabel zur NodeMCU zeigen.

Zuletzt verstaust du alle Kabel und drehst den Aufbau wieder um.

**Jetzt kann die Programmierung beginnen!**

*Anmerkung: Über ds rote und schwarze Kabel wird die Versorgungsspannung (+5V / Masse) an die Bauteile geführt. Das orange bzw. weiße Kabel wird für Steuersignale benutzt. Mehr dazu bei den Hintergrund-Infos.*

## Programmierung

Ein paar Worte vorweg: Es gibt verschiedene Programmierumgebungen für die NodeMCU bzw. den ESP8266. Diese Umgebungen werden oft auch als **IDE** (Integrated Development Environment) bezeichnet. Hier wird die Arduino-IDE verwendet, da diese kostenlos und weit verbreitet ist und zu einem schnellen Erfolg führt. Es soll jedoch nicht unerwähnt bleiben, dass es bessere IDEs gibt, wenn der volle Funktionsumfang genutzt werden soll, oder wenn größere Projekte anstehen. Für unsere Zwecke reicht aber die Arduino-IDE.

Wenn du die Umgebung zu Hause verwenden möchtest, findest du am Ende einen Link, wo du dir das Paket herunterladen kannst.

### Arduino starten und Sketch öffnen

Auf unseren Labor-PCs läuft Linux als Betriebssystem, daher sehen einige Menüs vielleicht etwas anders aus, als du es gewohnt bist. Einige Absätze in dieser Anleitung geben dir auch Tipps, wie du bestimmte Dinge unter Windows einstellen müsstest, falls du später einen anderen Rechner benutzt.

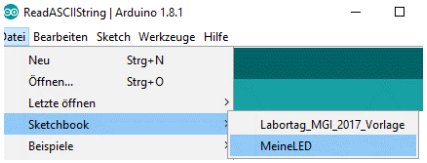
Starte nun den PC, die Login-Daten für den Rechner bekommst Du von einem Mitarbeiter genannt.

Zum Starten klickst Du doppelt auf das Arduino-Symbol rechts unten im Desktop.

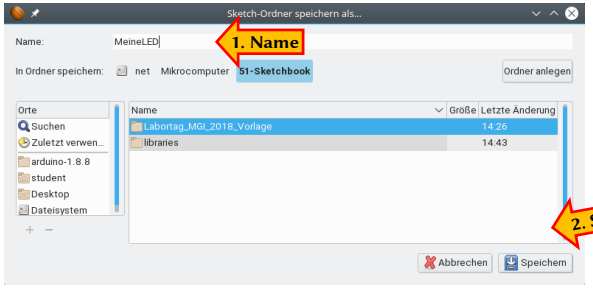
Ein Projekt heißt bei Arduino „**Sketch**“ und besteht aus mehreren Dateien. Als Programmiersprache verwendet Arduino C++, jedoch haben die Dateien nicht wie sonst in C++ üblich die Endung `*.cpp`, sondern `*.ino`. Weiterhin gibt es sog. Header-Dateien mit der Dateiendung `*.h`.

Da Arduino immer beim Start den letzten Sketch öffnet, sollte mit etwas Glück nun schon der richtige Sketch geöffnet sein. Dies erkennst du daran, in der Titelzeile „`Labortag_2020_Vorlage | Arduino` ...“ steht (Der Screenshot rechts ist etwas veraltet).

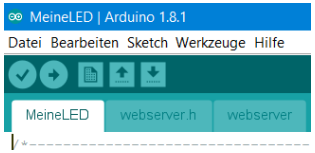

Wenn dies nicht der Fall ist, musst du erst den Sketch öffnen, indem du auf *Datei->Sketchbook* gehst und dort „`Labortag_2020_Vorlage`“ auswählst. Der Sketch wird dann in einem neuen Fenster geöffnet; das Fenster mit dem falschen Sketch kannst du dann schließen.




Als nächstes speicherst du die Vorlage unter einem neuen Namen ab, damit du bei Änderungen immer auf den Original-Code zurückgreifen kannst. Hierzu kannst du dir einen eigenen Namen für den Sketch ausdenken, vermeide aber Leer- und Sonderzeichen! In dieser Anleitung verwenden wir jetzt „`MeineLED`“.




Dazu gehst du auf *Datei->Speichern unter* und gibst als Namen den von dir gewählten Sketchnamen ein. Der Name in der Titelzeile und der Name der „Hauptdatei“ ändern sich automatisch entsprechend:



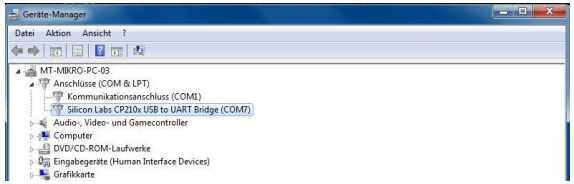

**Wichtig**: Wenn du nun Arduino neu starten solltest, musst du natürlich diesen Sketch öffnen, um weiterzuarbeiten.

### NodeMCU an den Computer anschließen

Ziehe zunächst das einzelne Kabel (die +5V) von der Anschlussplatine wieder ab, so dass LED und Servos keine Spannung haben. Der Aufbau brauch manchmal mehr Strom, als der USB-Port liefern kann, so dass der PC diesen abschaltet. Außerdem wird so verhindert, dass die Servos oder die LED bei Programmierung unvorhersehbare Dinge tun).

Bevor es jetzt mit den Einstellungen weitergeht, verbindest du die NodeMCU mit einem Micro-USB-Kabel mit dem PC. Es sollte unten rechts eine kurze Meldung erscheinen, dass etwas verbunden wurde...

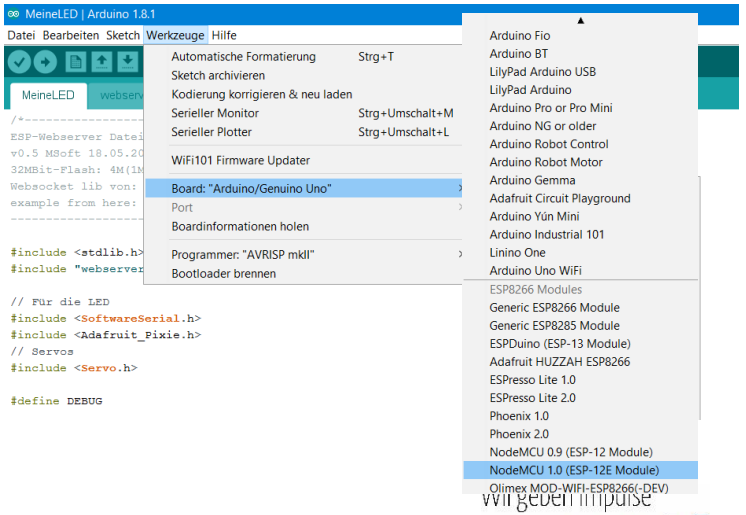
**In unserem Labor ist das folgende nicht nötig**, aber wenn du mit Windows arbeitest: *Zur Sicherheit prüfst du, welchen Anschluss die NodeMCU zugewiesen bekommen hat. Dazu startest du den Gerätemanager über* ⊞ x *(Windows-X) und den Gerätemanager auswählst. Die nach dem Start erscheinende Meldung bezüglich Standardnutzer bestätigst du. Im Gerätemanager klappst du den Eintrag „Anschlüsse (COM & LPT) auf; dort sollte ein Eintrag „Silicon Labs CP210x...“ erscheinen. Merke dir den dahinter angegebenen Anschluss (hier COM7).*



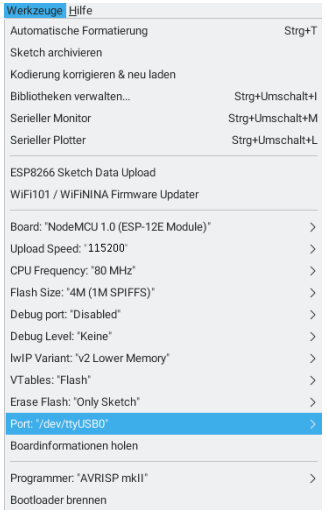


### Board-Einstellungen prüfen und ggf. anpassen

Die Arduino-Umgebung unterstützt viele verschiedene Entwicklungsplatinen. Daher musst du erst einstellen, welche Platine hier verwendet wird. Wähle den Menüpunkt *Werkzeuge* und schaue, ob dort bei *Board* bereits „NodeMCU 1.0“ steht. Wenn nicht, gehst du auf Board und wählst aus der Liste das richtige Board aus.



Die Kommunikation zwischen PC und NodeMCU erfolgt über die USB-Verbindung. Du musst hierfür die richtigen Parameter einstellen. Diese findest du ebenfalls unter dem Menüpunkt *Werkzeuge*. Setze die Upload-Speed auf `115200` und den Port auf „`/dev/ttyUSB0`“ *(zu Hause verwendest Du den wie oben beschrieben ermittelten COM-Port, z.B. COM7).*




## Programm kompilieren

Programme in C bzw. C++ werden nicht von dem Computer direkt ausgeführt. Diese Programmiersprachen dienen nur dazu, um in menschen-lesbarer Form Anweisungen an den Computer zu geben. Damit dieser den Code möglichst schnell ausführen kann, muss er erst in eine für den Computer optimierte Form übersetzt werden. Diesen Vorgang nennt man **Kompilieren**, das dazugehörige Programm Compiler.

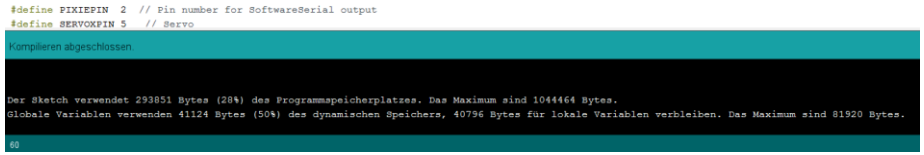
**Wichtig**: Jedes Mal, wenn der Code geändert wird, muss er neu kompiliert werden.

Den C-Code nennt man **Quelltext** (bzw. Source-Code), da er die Quelle für das ist, was später auf dem Computer ausgeführt wird.


In der Arduino-Welt wird dieser Vorgang mit „Überprüfen“ bezeichnet, da durch das Kompilieren gleichzeitig geprüft wird, ob der Quelltext Fehler enthält. Hierbei werden jedoch nur syntaktische Fehler (d.h. z.B. Tippfehler, Befehle falsch geschrieben usw.) entdeckt, nicht jedoch Fehler im Programmentwurf.

Klicke auf den Haken oben links in dem Fenster und beobachte den unteren schwarzen Fensterbereich (Konsole) und die Statuszeile darüber. Dort steht zunächst für eine Weile „Sketch wird kompiliert“.

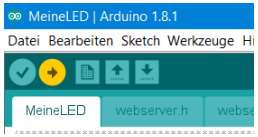
Wenn alles funktioniert, sollte danach folgendes erscheinen (die Zahlen können variieren):

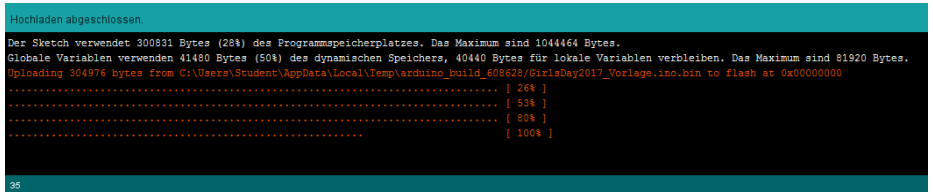



Falls Fehler im Programm sind, sieht der untere Bereich wie unten dargestellt aus. Hier wurde als Beispiel ein Semikolon hinter einem Befehl vergessen, die Zeile danach ist rot markiert (da der Compiler erst beim folgenden Befehl merkt, dass zwischen den Befehlen das Semikolon fehlte).


## Programm übertragen

Nun muss das übersetzte Programm noch auf die NodeMCU übertragen werden. Dazu dient der Pfeil rechts neben dem Haken, ebenfalls oben Links im Fenster. Hierbei erscheint zunächst wieder „Sketch wird kompiliert“ in der Zeile Statuszeile. Anschließend startet der sogenannte Upload (d.h. das Hochladen auf die NodeMCU).




## Dateisystem (SPIFFS) erzeugen

Für unser Projekt müssen einige Dateien auf der NodeMCU abgelegt werden (dies geschieht ähnlich wie auf einem USB-Stick, jedoch mit anderen Werkzeugen). Hierzu muss dort einmalig ein Dateisystem erzeugt werden; dieses bleibt auch bei Änderungen und erneutem Übertragen des Programms erhalten. Das Tool, welches hierzu verwendet wird, erwartet, dass die Dateien innerhalb des Sketches im Unterverzeichnis „`data`“ liegen. Bei älteren Arduino-Versionen wird dieses Verzeichnis beim Befehl „Speichern unter“ unglücklicherweise nicht mit kopiert, so dass du dies von Hand erledigen musst.

Auf den Labor-PCs gibt es hier aber keine Probleme, auch unter Windows sollte die aktuelle Arduino-Version dies automatisch erledigen.

Um den Ordner auf den Prozessor zu laden, wählst du in Arduino *Werkzeuge->ESP8266 Sketch Data Upload*

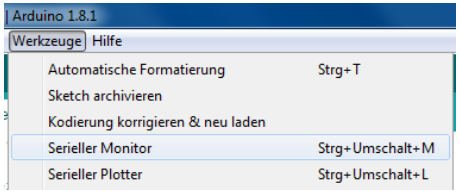
Wenn alles planmäßig läuft (d.h. NodeMCU angeschlossen, richtiger Port gewählt), sollte nun die Datenübertragung starten. Dies ist im unteren Statusfenster zu erkennen und dauert eine Weile.

Falls dort eine Fehlermeldung erscheint, versuche es noch einmal; ggf. auch einmal die NodeMCU vom USB-Bus trennen und neu einstecken. Wenn alles nicht hilft, sage kurz Bescheid…

### *Serieller Monitor*

Wenn dieses Hochladen fehlerfrei durchgelaufen ist, stellt sich natürlich die Frage, wie man ggf. Infos darüber bekommen kann, was die NodeMCU gerade tut. Da es nicht möglich ist, einen Bildschirm anzuschließen, muss ein anderer Weg gegangen werden (der bei eingebetteten Systemen üblich ist).

Über die USB-Verbindung kann über die serielle Schnittstelle (s.o.) Text ausgegeben werden (dies geschieht im Programm z.B. über `Serial.println`). Um diesen Text nun zu sehen, muss man ein Terminal auf der seriellen Schnittstelle des PCs starten. Hierzu hat Arduino bereits den seriellen Monitor eingebaut, der genau diese Aufgabe erledigt. Auch diesen findest du unter den Werkzeugen.



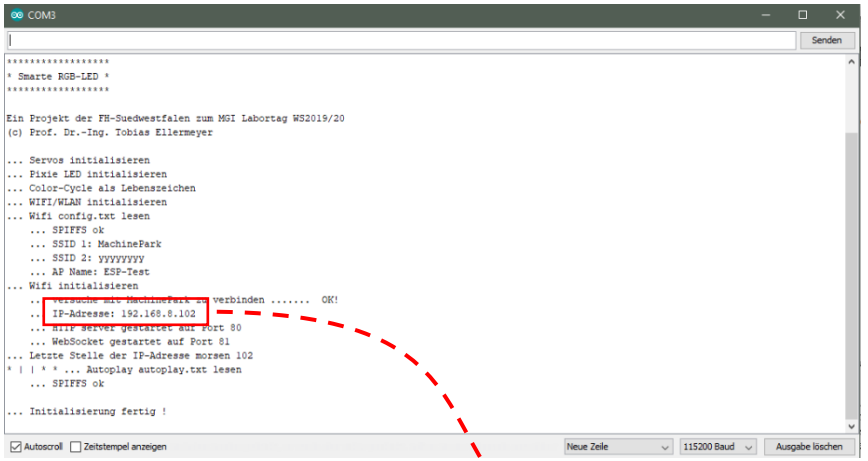

Wenn du den seriellen Monitor startest, öffnet sich ein zusätzliches Fenster, in dem Ausgaben zu sehen sind (auch hier muss die Geschwindigkeit auf 115200 Baud eingestellt sein).

## *Über WLAN verbinden*

Standardmäßig ist das Programm auf der NodeMCU so eingestellt, dass es sich mit dem WLAN *„Machine-Park“* des Labors verbindet. So ist diese zwar über die PCs im Labor erreichbar, jedoch nicht über das Smartphone.

Schaue zunächst in den seriellen Monitor, dort sollte eine Info stehen, welche IP-Adresse die NodeMCU zugewiesen bekommen hat. Falls hier nur kryptische Zeichen stehen, prüfe erst, ob die Baudrate auf 115200 eingestellt ist (rechts unten im seriellen Monitor). Drücke dann den Reset-Knopf auf der NodeMCU (kleiner Knopf „RST“ neben der Micro-USB-Buchse).

Im folgenden Bild ist die entsprechende Ausgabe markiert:




Starte nun auf dem PC den Internet-Browser Firefox und gebe oben in die Adresszeile diese IP-Adresse ein:


Nun sollte eine Seite erscheinen, die von der NodeMCU zur Verfügung gestellt wird. Stecke testweise das einzelne rote Kabel wieder auf die NodeMCU. Wenn du auf den Farbverlauf klickst, sollte sich die Farbe ändern (denk bitte daran, nicht direkt aus kurzer Entfernung in die LED zu schauen). Durch das untere Steuerkreuz kannst du die Position der Servos ändern. Später auf dem Smartphone kannst du auch den Finger über diesen Feldern bewegen (statt zu klicken)

Teste zunächst, ob alles wie gewünscht funktioniert. Es kann sein, dass sich bei allzu schnellen aufeinanderfolgenden Klicks die NodeMCU zurücksetzt und neu startet. Dies liegt daran, dass der USB-Port des Computers nicht genügend Strom liefern kann. Mit einem Smartphone-Netzteil mit mindestens 800 mA Ausgangsstrom sollte dies nicht mehr auftreten. Ggf. ziehst du das rote Kabel wieder ab, um diese Probleme zu vermeiden.

### *WLAN-Einstellungen*

Unterhalb dieses Steuerkreuzes findest du noch den Link „Konfiguration“. Hierdurch öffnet sich der SPIFFS-Explorer, welcher die Dateien anzeigt, die auf der NodeMCU liegen (das sind genau die, die wir ganz zu Beginn der Programmierung eingespielt haben).

Im oberen Teil der Seite kannst du die WLAN-Einstellungen ändern. Hier gibt es zwei WLAN Einträge mit jeweils einem `ssid` bzw. `bssid` und `pass` bzw. `bpass` Eintrag. Die zweiten Einträge (`bssid/bpass`) sind backup-Einträge, falls das erste WLAN nicht gefunden wird. Der Name des WLANs, mit dem sich die NodeMCU verbinden soll, wird in `ssid` eingetragen, dass dazugehörige Password in `pass`. Wie gesagt kann ein zweites WLAN durch die `bssid/bpass` Einträge hinzugefügt werden.

**Wichtig: Wie du vielleicht schon gemerkt hast, kennst du jetzt das Passwort des WLANs *MachinePark*. Dasselbe gilt natürlich auch, wenn du die Zugangsdaten deines Netzes zu Hause hier einträgst und jemand den Aufbau in die Finger bekommt.**

**Sei dir also dieser Sicherheitslücke bewusst, bevor du hier zu Hause deine Zugangsdaten einträgst. Natürlich hätten wir hier noch eine zusätzliche Sicherung einbauen können, dass würde das Programm und die Konfiguration aber unübersichtlicher machen.**

Weiterhin findest du noch die Einträge `ap_name` und `ap_pass` sowie `ap_ip`. Wenn die NodeMCU keines der beiden WLAN-Netze findet, erzeugt sie selbst ein eigenes WLAN-Netz mit dem Namen `ap_name`.

An dieser Stelle wird vielleicht deutlich, warum die NodeMCU nicht direkt so konfiguriert war, dass sie ein eigenes WLAN startet, da dann alle Aufbauten zunächst ein WLAN-Netz mit demselben Namen erzeugen, und du dann nicht weißt, welches „deine“ Smart-LED ist.

Um jetzt die **NodeMCU mit deinem Smartphone zu verbinden**, musst du zwei Dinge tun:

1. Verhindern, dass sich die NodeMCU mit dem Netz *MachinePark* verbindet. Das erreichst du am einfachsten, indem du dort ***None*** einträgst. Dann wird das Einbuchen in ein bestehendes Netz unterbunden.
2. Einen **eigenen Netzwerk-Namen vergeben**. Ändere dazu den Namen bei `ap_name` in einen frei gewählten Namen um (maximal 32 Buchstaben, keine Leerstellen, keine Sonderzeichen außer Bindestrich und keine Umlaute). Ein Passwort solltest du zunächst nicht setzen.

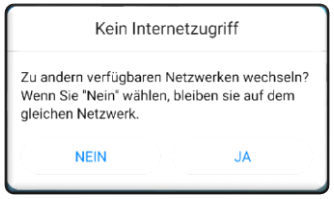
Wenn du diese beiden Schritte ausgeführt hast, klickst du auf „Speichern“ und anschließend auf „ESP Neustart“. Nun sollte die NodeMCU neu starten, was du im seriellen Monitor beobachten kannst. Falls dies nicht funktioniert, trennst du die NodeMCU einmal kurz vom USB und steckst sie wieder ein. Dann musst du aber auch den seriellen Monitor schließen und neu starten.

Wie du jetzt siehst, werden die WLAN-Netze nicht gefunden und ein eigener Accesspoint aufgemacht.

## Jetzt kommt endlich dein Smartphone zum Einsatz!

Gehe in die WLAN-Einstellungen deines Smartphones und suche nach deiner NodeMCU (d.h. dem Namen, den du gerade bei `ap_name` eingetragen hast). Verbinde dich mit diesem WLAN.

**Hinweis**: Bei einigen Smartphones erscheint noch eine Warnung, dass dieses WLAN keinen Internet-Zugang bietet. Hier musst du „Trotzdem verwenden“ bzw. „Nein“ o.ä. auswählen.



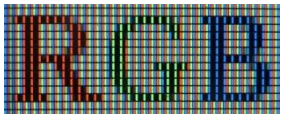

Wenn dein Smartphone verbunden ist, öffnest du den Internet-Browser und gibst als Adresse `http:/2.2.1.1` ein. Es sollte wieder die dir bekannte Seite mit Farb- und Servosteuerung erscheinen.

**Wichtig**: Nun ist der Aufbau ausschließlich als Accesspoint erreichbar. Wenn du ihn zu Hause in dein WLAN einbinden willst, musst du dich zunächst mit dem Smartphone verbinden (das geht natürlich auch mit einem Laptop/Tablet, indem du dort das entsprechende WLAN auswählst) und dann auf der Konfigurationsseite die Zugangsdaten für dein WLAN zu Hause eingibst. Denke aber an den Sicherheitshinweis weiter oben.

## Hintergrund-Infos

### Farbige LEDs

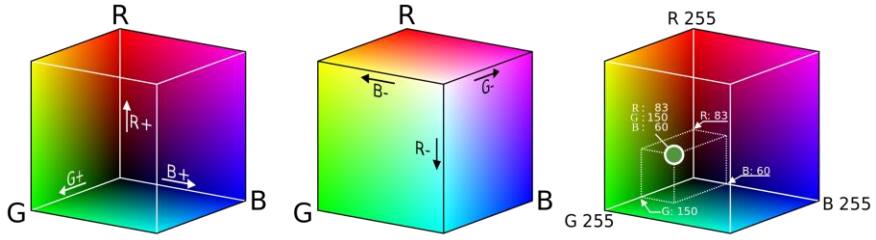
Wie du vielleicht schon bemerkt hast, hat die LED drei verschiedene Grundfarben, nämlich Rot, Grün und Blau. Aus diesen Grundfarben werden alle anderen Farben zusammengemischt.




Vielleicht hast du in der Schule schon einmal den Unterschied zwischen additiver und subtraktiver Farbmischung gelernt. Sicher weißt du jedoch, dass beim Wasserfarbkasten ein Gemisch von Rot/Grün und Blau nicht Weiß ergibt, sondern eher ein hässliches Braun. Das liegt daran, dass beim Malen oder Drucken die subtraktive Farbmischung entsteht, da bestimmte Farben aus dem weißen Licht (welches vom Papier reflektiert wird) herausgefiltert werden. Hier verwendet man üblicherweise die Farben Cyan (eine Art Türkis), Gelb und Magenta. Da diese Farben kein sattes Schwarz ergeben, wird dies meist getrennt aufgetragen. Abgekürzt wird diese Farbdarstellung mit CYMK (Cyan, Yellow, Magenta, Key=Schwarzanteil).

Anders sieht es bei der additiven Farbmischung aus. Hier wird im Prinzip eine weiße Fläche durch verschiedenfarbiges Licht angestrahlt und die Farben überlagern sich. Hierzu benötigt man dann die Farben Rot, Grün und Blau bzw. die sogenannte RGB-Darstellung. Diese Variante ist seit dem 19. Jahrhundert bekannt und wird in allen Monitoren, Beamern und Displays angewendet.

Indem man die Intensität der einzelnen drei Primärfarben ändert, können verschiedene Farben dargestellt werden. Dies lässt sich gut als Farbwürfel darstellen:




*By User:Maklaan - Own workbased on:RGB_farbwuerfel.jpg by Horst Frank, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=37705057*

Ganz links siehst du die Hinterseite, also von Schwarz kommend die einzelnen Farbkomponenten zunehmend. In der Mitte ist die Vorderseite dargestellt. Hier ergibt sich Weiß, wenn alle drei Farben maximal hell sind. Ganz rechts sind die einzelnen Achsen beschriftet; üblicherweise verwendet man für die Intensität der einzelnen Primärfarben den Wertebereich 0 (aus) bis 255 (maximal). Der dargestellte grüne Punkt ergibt sich dann, indem man den Rotanteil auf 83, den Grünanteil auf 150 und den Blauanteil auf 60 setzt. Schwarz wäre (0,0,0), Weiß (255,255,255) und Lila (255,0,255).

P.S.: Die Farbwürfel sehen auf einem Bildschirm deutlich besser aus als ausgedruckt; das liegt daran, dass der Drucker die Farben Cyan, Magenta und Gelb verwendet, um diese Farben darzustellen.

***Warum Wertebereich 0...255***: In der Computertechnik werden Daten als Bits bzw. Bytes gespeichert. Der Wertebereich 0…255 entspricht hierbei genau einem Byte.

***Wie viele Farben kann man so darstellen***: Für jede Farbe gibt es 256 verschiedene Möglichkeiten (0…255). Somit haben wir 256 verschiedene Rotwerte, die mit 256 verschiedene Grünwerten und dann nochmal mit 256 Blauwerten kombiniert werden können. Es gibt also $256 \cdot 256 \cdot 256 = 16{,}7$ Millionen verschiedene Farben, die ein Computer oder Smartphone üblicherweise darstellen kann.

Einen kleinen Nachteil dieser Farberzeugung wirst du beobachten, wenn du deinen LED-Scheinwerfer im Dunkeln betreibst: Da die einzelnen Farben räumlich recht weit auseinanderliegen, funktioniert die Farbmischung am Rand des Lichtkegels nicht so gut.

***Weiße LEDs***: Wie beschrieben kann weißes Licht durch die drei Grundfarben erzeugt werden. In den handelsüblichen LEDs für Beleuchtungszwecke wird jedoch ein anderer Weg gegangen. Da LEDs immer nur eine Farbe erzeugen, weißes Licht aber aus „allen“ Farben besteht, muss ein Trick angewandt werden: Die LED leuchtet in diesen Leuchtmitteln im UV-Bereich. Durch eine Phosphorschicht wird das UV-Licht dann in weißes Licht gewandelt. Dieses Verfahren hat einen besseren Wirkungsgrad (Lichtausbeute pro Watt), ist kostengünstiger und liefert einen besseren Weißton (bei RGB wären teilweise Farbsäume zu sehen).

### Ein-/Ausgänge der NodeMCU

| 0 / AUS<br>Falsch / Low | 1 / AN<br>Wahr / High |
|---|---|
| ca. 0 Volt | ca. 3,3 Volt |

Typisch für die Digitaltechnik ist, dass es nur zwei Zustände gibt, nämlich 0 und 1 bzw. Falsch/Wahr oder auch Aus/An. Die NodeMCU besitzt mehrere Pins, die entweder als Ausgang oder als Eingang verwendet werden können. Da diese universell verwendbar sind, werden derartige Pins als ***GPIO*** (General Purpose Input Output) bezeichnet. Eine Null bzw. aus entspricht dann einer Spannung von 0 V, wohingegen eine Eins bzw. an einer Spannung von 3,3 V entspricht (dies ist die Betriebsspannung der NodeMCU).

### Helligkeit einstellen

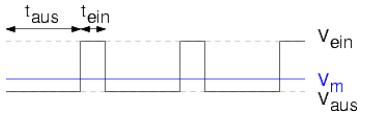
Es bleiben jetzt bei unserem LED-Scheinwerfer noch zwei Fragen:

Warum gehen nur drei Kabel zur LED, wenn dort doch 3 einzelne Farben enthalten sind, und man daher mindestens 4 Leitungen benötigen würde (eine für jede LED und eine gemeinsame Rückleitung).

Wie stellt ein Computer (der eigentlich nur Nullen und Einsen kennt) die Helligkeit möglichst einfach einstellen.

Zu a): Hier ist der Trick, dass auf der Rückseite der LED auch noch ein kleiner Minicomputer (Mikrocontroller) sitzt, der die einzelnen Farben ansteuert. Dieser Mikrocontroller (und die LED) werden einerseits mit Spannung versorgt (rot = +5 V und schwarz = Masse). Über die verbleibende Leitung werden Daten übertragen, die dem Mikrocontroller sagen, welche LED er wie hell einstellen soll.

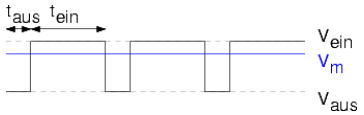
Zu b): Die Helligkeit wird mit Hilfe der ***Pulsweitenmodulation*** (PWM) eingestellt. Hierzu wird die LED immer eine bestimmte Zeit voll eingeschaltet und die übrige Zeit komplett ausgeschaltet. Ist die LED z.B. immer 0,2 ms eingeschaltet (ms = Millisekunden = Tausendstel-Sekunden) und 0,8 ms ausgeschaltet, so ist sie im Mittel 20 % der Zeit eingeschaltet. Da dieses Ein-/Ausschalten sehr schnell geschieht, nämlich z.B. tausendmal pro Sekunde, bemerkt das Auge dies nicht (dieses kann Bildänderungen nur bis ca. 30 … 60 mal pro Sekunde folgen; deswegen erkennst du z.B. bei einem schnell drehenden Rad eines Fahrrades auch die Speichen nicht mehr). Für das Auge sieht es also so aus, als würde die LED mit 20 % der maximal möglichen Helligkeit leuchten. Ändert der Mikrocontroller (bzw. Computer) nun das Verhältnis Ein- zu Auszeit, so ändert sich die Helligkeit. Für die LED hingegen sieht es so aus, als würde im Mittel eine entsprechend geringere Spannung anliegen:




$$V_{ein} = 5\ \text{V}; V_{aus} = 0\ \text{V}$$

$$t_{ein} = 0{,}25\ \text{ms}; t_{aus} = 0{,}75\ \text{ms}$$

$$V_m = \frac{t_{ein}}{t_{ein} + t_{aus}} \cdot V_{ein} = 1{,}25\ \text{V}$$




$$V_{ein} = 5\ \text{V}; V_{aus} = 0\ \text{V}$$

$$t_{ein} = 0{,}75\ \text{ms}; t_{aus} = 0{,}25\ \text{ms}$$

$$V_m = \frac{t_{ein}}{t_{ein} + t_{aus}} \cdot V_{ein} = 3{,}75\ \text{V}$$

Diese Methode zur Einstellung von Helligkeiten oder auch Motorleistungen ist heutzutage üblich, da sie sich mit einfachen Mitteln realisieren lässt und auch wenig Verlustwärme produziert.

Wenn z.B. eine Straßen- oder U-Bahn anfährt, hörst du auch oft ein Summen oder Pfeifen. Dies liegt daran, dass die Spannung für den Motor auch über Pulsweitenmodulation immer wieder kurz ein- und ausgeschaltet wird, um den Motor langsam (bzw. mit geringer Leistung) anlaufen zu lassen.

***LEDs und Dimmer***: Auch für den Hausgebrauch gibt es LED-Leuchtmittel, die sich dimmen lassen. Hierbei entsteht jedoch oft nicht der gewünschte Effekt. Während bei Glühlampen die Lichtfarbe ins orange/rote wechselt (da der Glühfaden weniger heiß ist), wird die LED einfach nur dunkler, die Lichtfarbe bleibt jedoch gleich. Teurere LED-Leuchtmittel versuchen daher den „Gemütlichkeits-Effekt" von Glühlampen nachzuahmen, indem durch zusätzliche gelb/orange LEDs die Lichtfarbe beim Dimmen geändert wird.

### *Servos ansteuern*

Auch bei den Servos wird zu einem Trick gegriffen, um die Position möglichst einfach über eine Leitung zu übertragen (die anderen beiden Kabel sind wieder für Versorgungsspannung und Masse reserviert). Die hier eingesetzte Technik stammt noch aus den frühen 80er-Jahren, als die Modellbau-Fernsteuerungen entwickelt wurden.

Auch hier wird wieder nur mit An/Aus-Signalen gearbeitet, wobei die Dauer des An-Signals die Position bestimmt (die Aus-Zeit ist hierbei nicht wichtig). Eine An-Dauer von 1,5 ms entspricht der Mittellage, bei einer Dauer von 1,0 ms bzw. 2,0 ms ist der linke bzw. rechte Anschlag gemeint.

Diese Signale werden ähnlich wie die Pulsweitensignale vom Mikrocontroller bzw. der NodeMCU erzeugt und typischerweise alle 20 ms, d.h. 50-mal pro Sekunde, wiederholt.

Da Servos häufig bei derartigen Bastelprojekten eingesetzt werden, ist eine entsprechende Bibliothek bereits vorhanden. Leider kann die NodeMCU die benötigten Signale nur durch Software erzeugen (andere Mikrocontroller machen dies in Hardware), so dass es zu Ungenauigkeiten im Timing kommt. Daher werden die Servo-Signale in der vorliegenden Software nach einer definierten Zeit abgeschaltet, um ein „Zucken" der Servos zu verhindern.

### *Software, Download-Link und Stückliste*

**Download-Link**

Die eingesetzte Arduino-Version kannst du dir hier herunterladen. Entpacke die Datei einfach in ein Verzeichnis auf deinem Laufwerk (ca. 1,2 GB), vermeide Dabei aber Leerzeichen im Dateipfad.

`http://t1p.de/Labortag` (Das zweite Zeichen ist eine Eins, kein Buchstabe i)

Unter diesem Download findest du auch die 3D-Dateien und diese Anleitung als PDF.

**Verwendete Arduino-Installation / Bibliotheken (für Setup ohne obige ZIP-Datei)**

Zip-File von arduino.cc heruntergeladen
Entpackt in Ordner C:\arduino-1.8.11
Dort einen Ordner portable erzeugt
Arduino durch arduino.exe gestartet
Unter Voreinstellungen: Boardverwalter: http://arduino.esp8266.com/stable/package_esp8266com_index.json
Werkzeuge->Boards->Boardverwalter: ESP8266 installieren (v2.63)
Bibliotheken: WebSockets (Markus Sattler v2.1.4); Adafruit-Pixie (v1.1.1); ESP8266_ISR_Servo (v1.0.1)
ESP8266FS Plugin: https://github.com/esp8266/arduino-esp8266fs-plugin (V0.5.0)
??Pixie LED library: https://learn.adafruit.com/pixie-3-watt-smart-chainable-led-pixels/wiring-and-test

**Stückliste**

| Anzahl | Bezeichnung | ca. Preis | Bezugsquelle z.B. |
|---|---|---|---|
| 1 | NodeMCU „Amica" | 7,95 € | exp-tech.de (NodeMCU v2 lua based ESP8266) |
| 1 | Pixie LED | 15,95 € | exp-tech.de (Adafruit Pixie LED 3W) |
| 2 | Servos SG-90 | 2,30 € / Stk | Amazon Marketplace |
| 1 | Anschluss-Platine | s.u. | s.u. |
| 1 | Bausatz Holzteile | 1,95 € | Baumarkt + Lasercutter |
| 1 | 3D-Druckteile-Satz | ??? | 3D-Druck-Service o.ä. |
| 2 | Muttern M2 | 2,09 € / 100 Stk. | conrad.de (216364 – 62) |
| 2 | Unterlegscheiben M2 | 1,79 € / 100 Stk. | conrad.de (521659 – 62) |
| 4 | Schrauben M2 x 10 | 2,99 € / 100 Stk. | conrad.de (889752 - 62) |
| 2 | Schrauben Servo-Arm | ./. | liegen den Servos bei |
| 2 | Schrauben oberer Servo | ./. | liegen den Servos bei |
| 1 | Kabel JR-Stecker 3-pol | | Amazon |

Stückliste Anschluss-Platine

| Anzahl | Bezeichnung | ca. Preis | Bezugsquelle z.B. |
|---|---|---|---|
| 3 | Stiftleisten 1x3 2,54 mm | 0,96 € gesamt | conrad.de (1088187 – 62; zurecht schneiden) |
| 1 | Stiftleiste 1x1 2,54 mm | ./. | conrad.de (Rest von 1088187-62) |
| 1 | Buchsenleiste 1x10 2,54 mm | 0,71 € | conrad.de (1088188 – 62) |
| 1 | Kabel rot | | conrad.de |
| 1 | Elektrolyt-Kondensator 220 µF / 10 V | 0,26 € | conrad.de (460710 – 62) |
| 1 | Lochrasterplatine | 2,89 € | conrad.de (529580 – 62) |

USB-Netzteil + Micro-USB-Kabel aus Alt-Beständen…

## *Kontakt*

**Prof. Dr.-Ing. Tobias Ellermeyer**
ellermeyer.tobias@fh-swf.de
**Fachhochschule Südwestfalen**
Frauenstuhlweg 31 – 58644 Iserlohn
http://www.fh-swf.de

Fachhochschule Südwestfalen
University of Applied Sciences
Wir geben Impulse