Arduino-Kompendium

Danny Schreiter

1. Auflage: Mai 2019

ISBN: 978-3-96645-018-8

Herausgegeben durch:
BMU Media GmbH
Hornissenweg 4
84034 Landshut

www.bmu-verlag.de
info@bmu-verlag.de

Coverfoto: CAD Rendering, Andrew Whitham
Lektorat: Markus Neumann
Druck: Wydawnictwo Poligraf

Arduino-Kompendium

Inhaltsverzeichnis

Alle Programmcodes und Schaltpläne aus diesem Buch stehen kostenfrei zum Download bereit. Dadurch müssen Sie Code nicht abtippen.

Außerdem erhalten Sie die eBook Ausgabe zum Buch im PDF Format kostenlos auf unserer Website:

www.bmu-verlag.de/arduino-kompendium
Downloadcode: siehe Kapitel 19

Kapitel 1
Einleitung

Hätte man vor vierzig Jahren Passanten auf der Straße gefragt, was ihr Leben mit Mikrocontrollern zu tun hat, hätten die meisten vermutlich nur verwundert mit den Schultern gezuckt. Zu dieser Zeit waren sich wohl auch nur wenige Menschen überhaupt über den Begriff im Klaren. Die sichtbare Welt funktionierte analog, und zwar ein bisschen auch deshalb, weil kaum jemandem der Begriff "digital" überhaupt verständlich war.

Heute leben wir in einer anderen Welt. Die Digitalisierung veränderte unseren Alltag und tut es noch immer. In unseren Hosentaschen tragen wir Computer, die man noch vor wenigen Dekaden für unmöglich realisierbar gehalten hätte. Die rasante technische Entwicklung machte auch vor den Hobbykellern nicht halt. So entstand in vielen Bastlerköpfen der Wunsch, ein klein wenig die digitale Welt selbst mitzugestalten, etwas zu kreieren. Der Trend des *Physical Computing* war geboren, allerdings zunächst noch sehr beschwerlich – mit Handbüchern in Telefonbuchstärke und kaum verfügbaren Hardware-Komponenten.

Ein paar findigen Italienern gelang es, diese Misere zu beenden. Sie schufen mit Arduino eine *Physical Computing Plattform* für Jedermann. Wegen der dadurch entstandenen schier unzählbaren Möglichkeiten machten die kleinen Experimentierplatinen rasch Karriere und wuchsen über den Hobbykeller hinaus: Künstler bereichern dadurch heute ihre Installationen mit interaktiven Elementen, Forscher bauen Prototypen für Feldversuche, Schüler lernen an Arduinos das Programmieren.

Dieses Buch wird nun auch Ihnen einen Einstieg in diese faszinierende Welt vermitteln, Impulse geben und Möglichkeiten aufzeigen. Nach der Lektüre werden Sie in der Lage sein, eigene Projekte selbst zu planen und umzusetzen, Programmcode zu entwickeln und Fehler zu beseiti-

gen. Dabei wird stets Wert auf eine ausgewogene Mischung zwischen Theorie und Praxis gelegt. Sie werden notwendige technische Grundlagen kennenlernen ohne ermüdend tief in Formeln oder Syntaxregeln zu versinken. Zahlreiche Beispiele machen das vermittelte Wissen anschaulich und nachvollziehbar. Die Reihenfolge der Kapitel wurde so gewählt, dass Sie die besprochenen Sachverhalte Schritt für Schritt an Ihrem eigenen Arduino nachvollziehen können.

Das Buch erhebt somit auch keinerlei Anspruch auf Vollständigkeit. Stattdessen wurde der Fokus darauf gelegt, dass Sie die wichtigsten Zusammenhänge auch ohne technische Vorkenntnisse verstehen können. Sie werden einen breiten Überblick über Möglichkeiten und Vorgehensweisen in verschiedenen Gebieten erhalten – sei es Elektronik, Programmierung, Peripherie oder Vernetzung. Möchten Sie anschließend Ihr Wissen in einem bestimmten Teilbereich vertiefen, können Sie dann hierdurch bereits auf eine gute Grundlage zurückgreifen.

Kapitel 2
Geschichte

2.1 Die Geschichte von Mikrocontrollern

Die Geschichte von Mikrocontrollern beginnt mit der Geschichte der Mathematik. Mit dem Bestreben der Gelehrten, die Welt und ihre Mechanismen in Formeln und Verhältnissen zu beschreiben, entstand rasch auch der Wunsch, diese effizient berechnen zu können. Bereits in der Antike begannen die Menschen mit dem Gebrauch der Rechenmethoden und Zahlensysteme, welche die Grundlagen für unsere modernen digitalen Geräte bilden. Dabei entwickelte man schon Handlungsvorschriften zur schrittweisen Lösung eines Problems, wie zum Beispiel den *euklidischen Algorithmus* zur Ermittlung des größten gemeinsamen Teilers zweier Zahlen. Ein Algorithmus ist also quasi das Kochrezept, das Schritt für Schritt den Lösungsweg für ein bestimmtes mathematisches Problem beschreibt – egal, ob nun zur Berechnung der nächsten Sonnenfinsternis, einer Brückentragfähigkeit oder Ihres Nettolohns. Über Jahrhunderte führte man solche Rechenvorschriften von Hand durch.

Verständlicherweise entstand das Bestreben, derartige Problemlösungen zu automatisieren. Bereits ab dem 17. Jahrhundert kamen erste mechanische Rechenwerke auf; doch es sollte bis 1941 dauern, als mit der *Zuse Z3* der erste programmierbare Universalrechner, der beliebige Algorithmen ausführen konnte, in Betrieb genommen wurde. Die daraus entwickelten Großrechner in vielen Universitäten und Unternehmen füllten ganze Säle. Nun berechnete man Staudämme, Hängebrücken und Bilanzen mit der neuen Technik.

Diese Anfänge der elektronischen Rechenwerke nutzte man auch für die Einsatzerprobung verschiedener Zahlensysteme. Während die mechanischen Vorgängermaschinen fast immer das uns Anwen-

dern bestens vertraute Dezimalsystem verwendeten, erwies sich dies im bereits im Einsatz der *Z3* als eher unvorteilhaft und umständlich. Neben einigen mutigen Versuchen mit exotischen Zahlensystemen (so zum Beispiel der auf dem Ternärsystem[1] basierende *Setun*-Rechner, welcher 1958 in der Sowjetunion gebaut wurde), kristallisierte sich bald das Binärsystem als Favorit der Hersteller heraus. Seine Reduktion auf nur zwei komplementäre Zustände (*0* oder *1*; *ein* oder *aus*) lässt sich in der Elektronik besonders einfach und robust realisieren. Wir werden im Kapitel 4 noch einen genaueren Blick auf das Binärsystem werfen.

Die aufkommende Halbleitertechnik verkleinerte elektronische Komponenten um mehrere Größenordnungen. Die daraus resultierende Entwicklung des Mikrochips Mitte der 1960-er Jahre ermöglichte fünf Jahre später schließlich auch den Bau des ersten Mikroprozessors. Nun konnten alle elektronischen Komponenten eines Rechners, der in der Relais-Bauweise der *Z3* noch ganze Sporthallen gefüllt hätte, auf einem daumennagelgroßen Silizium-Chip untergebracht werden. Die Bahn war frei für die Entwicklung unserer heutigen Computer.

Die fortschreitende Miniaturisierung ermöglichte es nun, neben dem eigentlichen Prozessor, also dem Gehirn des Rechners, auch weitere Komponenten mit auf dem Chip unterzubringen. Durch das Anfügen von Arbeits- und Programmspeicher sowie Schnittstellen zur Ein- und Ausgabe von Daten sowie optionalen weiteren Funktionsblöcken entsteht ein kompletter Computer auf nur einem Chip – der Mikrocontroller. Ohne diesen wäre heute Ihr Smartphone nicht smart und Ihre Kreditkarte nicht fälschungssicher.

2.2 Entstehung der Arduino-Plattform

Die stetig fallenden Preise für Mikrochips sorgten dafür, dass Mikrocontroller nach der Jahrtausendwende auch mehr und mehr in

1 Ein Ternärsystem, auch Dreiersystem, arbeitet nur mit drei möglichen Ziffern; je nach Definition 0, 1 und 2 oder -1, 0 und 1.

den heimischen Bastelkellern ankamen. Intelligente Roboter, fernsteuerbare Autos, automatische Murmelsortiermaschinen – die Miniaturcomputer machten vieles möglich, was mit herkömmlicher Elektronik schlicht zu aufwändig gewesen wäre. *Physical Computing* war der neue Trend: Man verband also das Basteln an der Hardware mit dem Programmieren an der Software, um kleine "intelligente" Selbstbau-Projekte zu realisieren. Diese können auf Ereignisse in unserer realen, analogen Welt reagieren, sie verarbeiten und sogar auf sie einwirken. Manch Bastler baut sich einen Blumengieß-Automat, ein anderer eine Lichtsteuerung oder gar einen selbstkreierten Staubsaug-Roboter.

Anfangs schien all dies jedoch noch als eine ingenieurtechnische Herausforderung: Zwar gab es preiswerte Mikrocontroller, diese waren jedoch auf den industriellen Markt ausgelegt. Um ihnen Herr zu werden, galt es umfangreiche Dokumentationen zu studieren, teils teure Software anzuschaffen und passende Platinen zu bauen – für den Heimgebrauch undenkbar.

Das wollten die Italiener Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino und David Mellis ändern. Sie trafen sich regelmäßig in einer Bar in Ivrea, die ihren Namen "Arduino" von einem einstigen italienischen König übernommen hatte. 2005 wählten sie diesen Namen für ihre selbstentwickelte Musterplatine, kurz darauf stellten sie auch die erste Version einer Computersoftware zur Programmierung des Controllers fertig. Erstmals gab es nun eine einfach zu nutzende Platine, die zahlreiche Anschlüsse eines Mikrocontrollers für verschiedenste Anwendungen zu Verfügung stellt, ohne dabei teures Zubehör zu erfordern – denn die Entwickler veröffentlichten das Projekt unter einer Creative-Commons-Lizenz und ermöglichten somit die einfache Nutzung für Jedermann.

Die fünf Studenten gründeten die *Arduino LCC* als gemeinsame Firma, um das Projekt weiter voranzutreiben. Die eigentliche Herstellung der Hardware übernahmen dabei externe Dienstleister. Schnell stellte sich wirtschaftlicher Erfolg ein; man schätzt, dass allein bis 2013 rund 700.000 Arduinos verkauft wurden. Jedoch entbrann-

te ein Markenrechtsstreit zwischen den Gründern, der 2015 sogar dazu führte, dass unter dem Namen *Genuino* eine neue Marke eingetragen wurde. Auch wenn der Streit 2016 offiziell beigelegt wurde, herrscht in manchen Bereichen noch Unsicherheit über die rechtliche Situation. Wundern Sie sich also nicht, wenn Sie nach "Arduinos" suchen, aber "Genuinos" angeboten bekommen – der Inhalt ist identisch. Im Sprachgebrauch blieb der ursprüngliche Name aber stets standhaft.

Es sei erwähnt, dass etwa zeitgleich zur Entwicklung des Arduino auch der Raspberry Pi entstand. Falls Sie sich nun nach dem Unterschied fragen: Es handelt sich dabei um einen Einplatinencomputer, der ebenfalls bei Selbstbau-Projekten äußerst beliebt ist. Seine Fähigkeiten übersteigen die eines Arduino aber um mehrere Größenordnungen. Es handelt sich um einen vollwertigen Computer mit Betriebssystem, Dateisystem, Monitoranschlüssen, USB-Ports und vielem mehr. Dadurch steigt allerdings auch das Level der Abstraktion, das heißt, man kann nicht mehr so "nah an der Hardware" agieren. Die Programmierung erfolgt wie bei einem normalen PC; auf dem Chip läuft umfangreiche Software, die von einer einzelnen Person kaum zu überblicken ist. Nicht zuletzt ist ein Raspberry Pi auch den gleichen Gefahren ausgesetzt wie ein normaler PC: Programmfehler sind mitunter schwer zu lokalisieren, er kann – je nach Betriebssystem – anfällig sein für Viren und unbedachte Updates können die Funktionalität unter Umständen komplett lahmlegen.

Im Gegensatz dazu benötigt Arduino weder ein herkömmliches Betriebssystem noch ein Dateisystem auf dem Chip. Er erlaubt den unmittelbaren Zugriff auf die Hardware (Ein- und Ausgangspins) und ist sehr zuverlässig, wenn Sie auf die Echtzeitfähigkeit, also die garantierte Reaktion in einem gewissen Zeitfernster, Wert legen. Der Befall mit Viren oder Malware ist nahezu ausgeschlossen. Diese minimalisierten Fähigkeiten schlagen sich auch im Preis nieder: Ein Arduino ist deutlich günstiger zu haben als ein Raspberry Pi. Kaufen Sie nur den zu Grunde liegenden Mikrocontroller-Chip, ist die Ersparnis noch größer.

Beide Plattformen haben also ihre Vor- und Nachteile. Als Faustregel gilt: Ist Ihr Projekt eher Hardware-orientiert, also wollen Sie Lämpchen, Motoren und Sensoren steuern, ist Arduino sehr wahrscheinlich die beste Lösung für Sie. Erfordert Ihr Projekt jedoch umfangreiche Rechenleistung, zum Beispiel weil Sie ein Kamerabild auswerten möchten oder einen Computermonitor als grafische Anzeige benutzen, sollten Sie zu einem Raspberry Pi greifen. Natürlich gibt es Überschneidungen. So könnten Sie eine Wanduhr beispielsweise sehr elegant per Arduino mit angeschlossenen Schrittmotoren oder LED-Anzeige realisieren; aber Sie könnten auch einen Raspberry Pi nehmen und per Videobeamer eine animierte Uhr projizieren. Allein Ihre Fantasie entscheidet. Es gibt auch zahlreiche Projekte, die beide Plattformen zusammen einsetzen: So zum Beispiel ein Roboter, dessen "Gehirn" ein Raspberry Pi ist, welcher seine Sensordaten jedoch von mehreren verteilten Arduinos bezieht, die diese Daten zunächst aufbereiten. Beide Plattformen stehen also nicht in Konkurrenz zueinander, sie ergänzen sich vielmehr. Im Folgenden fokussieren wir uns auf Arduino-Plattform, zu Raspberry Pi ist aber ebenfalls entsprechende Literatur erhältlich.

2.3 Überblick über verfügbare Hardware

Was die Hardware betrifft, hat sich in den letzten Jahren der Arduino UNO als Standardmodell durchgesetzt. Alle wichtigen Anschlussmöglichkeiten sind direkt über kleine Buchsenleisten an der Platine abgreifbar, dadurch muss man beim Experimentieren nichts löten. Die Platine kann sowohl über USB als auch über ein Netzteil mit Niedervoltstecker mit Spannung versorgt werden. Die aktuelle überarbeitete Version "UNO R3" (Revision 3) unterscheidet sich nur in nebensächlichen Details von den Vorgängerversionen. Als Mikrocontroller kommt nach wie vor ein "ATmega328P" des Herstellers Microchip AVR (vormals Atmel) zum Einsatz. Für viele Anwendungen sind sogenannte *Shields* verfügbar. Das sind Erweiterungsplatinen, die dank passender Dimensionierung direkt auf den Arduino UNO aufgesteckt werden können und dadurch auch sofort mit Betriebsspannung und allen benötigten Signalen versorgt werden.

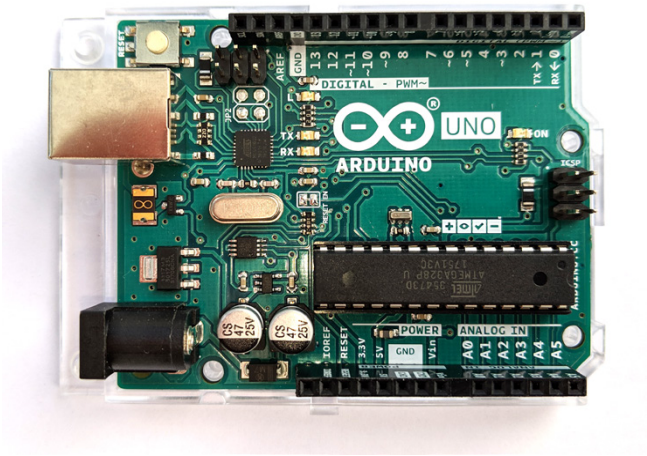

Abbildung 1 Arduino UNO R3

Neben dem Arduino UNO wurden noch viele weitere Bauformen entwickelt und getestet. Nachfolgend seien nur die genannt, die es zu einer gewissen Relevanz geschafft haben.

Arduino Nano ist die Miniaturisierung des Arduino UNO. Er besitzt weiterhin alle Anschlusspins und den gleichen Mikrocontroller, benötigt jedoch nur noch ein Viertel der Grundfläche seines großen Bruders. Möglich wurde dies, indem die Bauform der Mikrochips und des USB-Anschlusses verkleinert wurde, außerdem entfällt der separate Spannungseingang. Man kann ihn zum Experimentieren direkt auf ein Breadboard stecken. Er kann für alle Projekte genutzt werden, die auch mit Arduino UNO möglich sind, außer wenn Shields zum Einsatz kommen – durch die abweichende Bauform können sie nicht mehr direkt gesteckt werden.

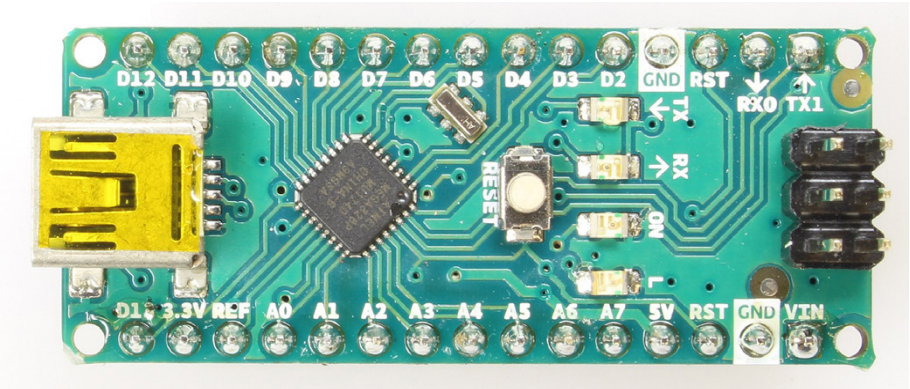

Abbildung 2 Arduino Nano

Der *Arduino Mini* wurde durch den Wegfall des USB-Ports nochmals verkleinert. Zum Programmieren ist deshalb ein spezieller Adapter nötig. Ansonsten entspricht er dem Arduino Nano.

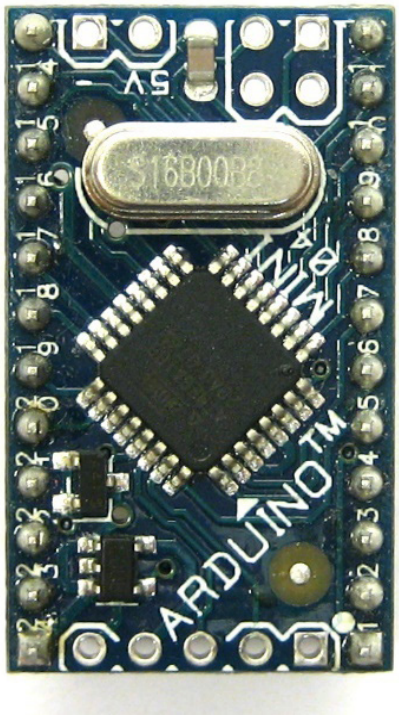

Abbildung 3 Arduino Mini

Arduino Micro ist ein etwas leistungsstärkerer Bruder des Arduino Nano. Trotz der etwas längeren Abmessungen passt er noch auf ein Breadboard. Er bietet mehr Ein- und Ausgabepins sowie geringfügig mehr Arbeitsspeicher, da nun ein ATmega32U4-Mikrocontroller zum Einsatz kommt.

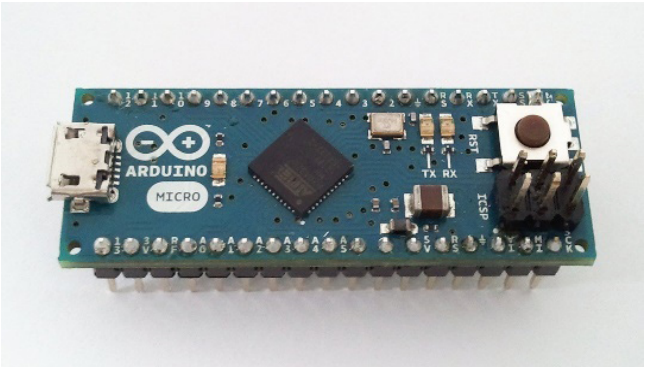

Abbildung 4 Arduino Micro

Der *Arduino Mega2560* bietet den 8-fachen Speicher des Arduino UNO, seine Anschlusszahl wurde mit 54 digitalen und 16 Analogen Pins mehr als verdreifacht. Entsprechend ist auch die Bauform der Platine etwas größer. Die meisten Shields sind dennoch kompatibel.

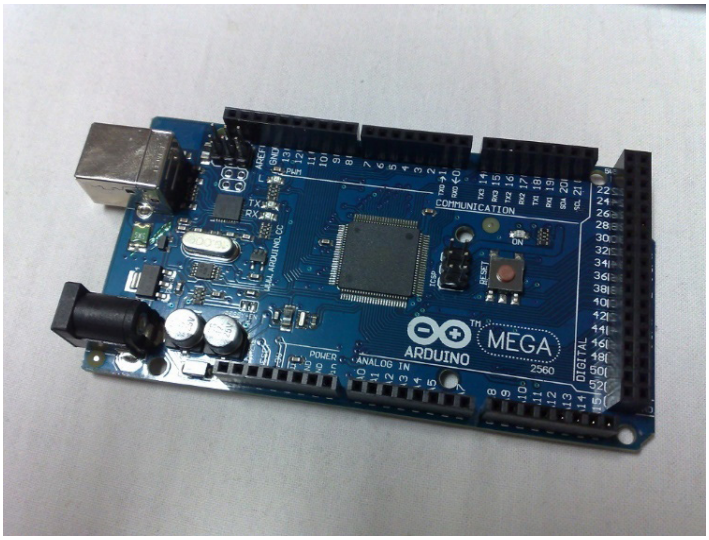

Abbildung 5 Arduino Mega2560

LillyPad Arduino ist der Name einer Reihe von Arduino-Boards, die durch ihre runde Bauform auffallen. Sie wurden dafür konzipiert in Textilien eingearbeitet zu werden. Ihr Durchmesser beträgt meist 5 cm, in den Anschlüssen orientieren sie sich weitgehend an den rechteckigen Boards, jedoch mit gewissen Einschränkungen.

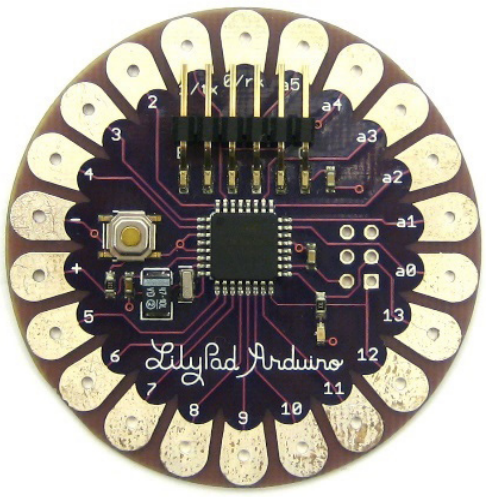

Abbildung 6 LillyPad Arduino

Die kleineste Miniaturisierung stellt der *Arduino Gemma* dar. Auf einer kreisrunden Scheibe von nur 2,8 cm Durchmesser bietet er allerdings auch nur 3 Datenpins.

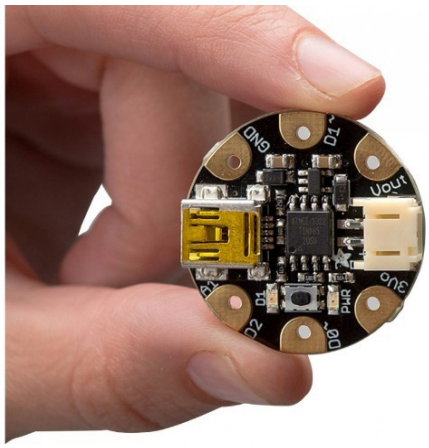

Abbildung 7 Arduino Gemma

Es existieren zahlreiche weitere Sonderbauformen, beispielhaft genannt sei *Arduino Robot*, dessen Form und Anschlüsse bereits an seinen angestrebten Einsatz als Roboter angepasst wurden oder Platinen, die bereits als Grundplatte für ein kleines Auto genutzt werden können.

Die Tatsache, dass das Arduino-Projekt im Wesentlichen unter freien Lizenzen veröffentlicht wurde, sorgt dafür, dass es auch zahlreiche Nachbauten und alternative Boards gibt. Die meisten von ihnen sind nahezu vollständig kompatibel zum Original, allerdings unterscheiden sie sich in einigen kleinen Details. So verfügen preiswertere Boards oft über den identischen Mikrocontroller und exakt die gleichen Anschlüsse, allerdings wird für die USB-Verbindung zum PC ein anderer USB-zu-Seriell-Konverter verwendet. Das führt dazu, dass Sie unter Umständen zunächst einen Treiber für diesen Konverter-Chip mit der Bezeichnung CH340 aus dem Internet laden müssen, bevor Ihr PC das Board erkennt. Näheres dazu erfahren Sie in Kapitel 16.1.

2.4 Shields

Die zahlreichen Anschlusspins der Arduino-Platinen ermöglichen die Verbindung mit einer Vielzahl von weiteren Komponenten, in späteren

Kapiteln werden wir noch viele Beispiele kennenlernen. Eine Besonderheit der Arduino-Plattform sind dabei die Shields. Dabei handelt es sich um Erweiterungs-Platinen, welche in ihren Abmessungen so gestaltet wurden, dass sie direkt auf den Arduino UNO aufgesteckt werden können. Die Anschlusspins dienen dabei zugleich als elektrische und mechanische Verbindung.

Je nach Shield-Typ liegt die Funktion beispielsweise in der Steuerung von Motoren, der Herstellung einer Netzwerkverbindung oder der Verbindung mit einer SD-Karte. Die dafür benötigten Pins werden direkt abgegriffen, alle weiteren Anschlüsse werden durch das Shield einfach weitergeleitet und sind genau wie auf der Hauptplatine abgreifbar. Gibt es keine Dopplungen unter den benötigten Signalpins, können Shields sogar kaskadiert werden.

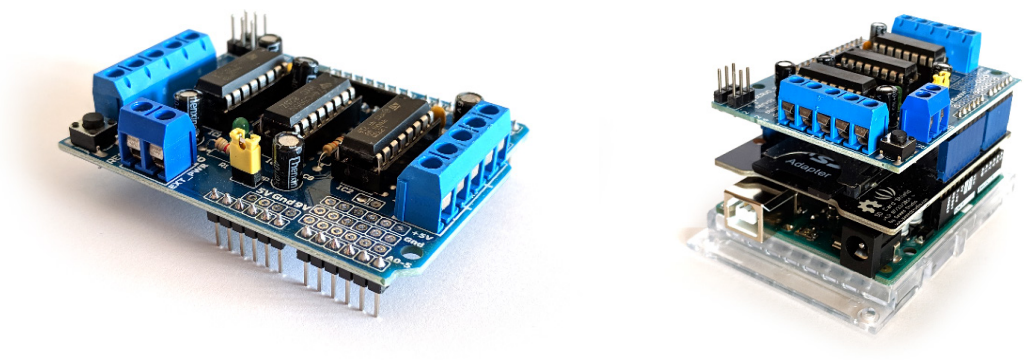

Abbildung 8 links ein Shield zur Steuerung von bis zu 6 Motoren, rechts eine Kaskade aus einem Arduino UNO mit aufgestecktem SD-Karten-Shield und zusätzlichem Motor-Shield

Der Vorteil von Shields liegt darin, dass die Anzahl der manuell zu steckenden Verbindungen sinkt. Damit verringert sich auch die Gefahr von Verdrahtungsfehlern und der Versuchsaufbau wirkt aufgeräumter.

2.5 Software-Überblick

Die heute meistgenutzte Software zur Arduino-Programmierung ist die offizielle Arduino IDE. Das Kürzel steht für “Integrated Develop-

ment Environment" (Integrierte Entwicklungsumgebung) – dieser etwas sperrige Begriff weist darauf hin, dass mit dieser Anwendung alle Schritte der Softwareentwicklung "an einem Ort" möglich sind.

Ursprünglich entstand die Arduino IDE aus dem Projekt *Wiring*. Hernando Barragán entwickelte es ab 2003 im Rahmen seiner Masterarbeit mit dem Ziel, Designern und Künstlern die Arbeit mit Elektronik zu erleichtern. Diese schreckten häufig vor den relativ komplizierten Details der Schaltungen und Prozessoren zurück – Barragán wollte diese "quälenden Details" für die Anwender ausblenden, so dass ein Fokus auf das eigentliche Kunstprojekt möglich wird. Es entstand eine Java-basierte Software-Plattform, aus der schließlich auch die heutige Arduino IDE abgeleitet wurde. Als Anwender profitieren Sie dabei heute sehr von Barragáns Bestrebungen zur Vereinfachung, deshalb werden wir uns in den folgenden Kapiteln stets auf die Arduino IDE beziehen.

Für den Überblick sei jedoch erwähnt, dass die Programmierung der Arduino-Boards auch mit zahlreichen anderen, teils umfangreicheren und komplexeren Anwendungen möglich ist. Die Plattform ist quelloffen und verwendet einen Standard-Mikrocontroller, somit gibt es kaum Schranken. Die folgenden Alternativen richten sich an erfahrenere Anwender mit komplexeren Projekten:

- *Atmel Studio* stammt direkt vom Hersteller des auf dem Board verwendeten Mikrocontrollers. Es läuft nur unter Windows und ermöglicht eine sehr hardwarenahe Entwicklung und kann dem Chip beispielsweise im Hinblick auf Rechengeschwindigkeit und Speicherplatzoptimierung noch Reserven entlocken.[2]
- *Microsoft Visual Studio* benötigt die *VisualMicro*-Erweiterung, um mit dem Arduino kompatibel zu sein. Dies empfiehlt sich, wenn man ohnehin schon anderen Projekte in Visual Studio bearbeitet – so muss man nicht die Plattform wechseln.[3]

2 https://www.microchip.com/mplab/avr-support/atmel-studio-7

3 https://www.visualmicro.com/page/Arduino-Visual-Studio-Downloads.aspx

- *MariaMole* ähnelt in der Handhabung sehr der Arduino IDE, lässt sich jedoch in einigen Punkten besser an persönliche Vorlieben anpassen.[4]
- *embedXcode* erweitert die in der Macintosh-Welt verbreitete Plattform *Xcode* um die Arduino-Unterstützung[5]

Außerdem besteht die Möglichkeit, den Web-Editor auf der offiziellen Arduino Website[6] zu verwenden. Da dieser jedoch, wie oft bei Web-Anwendungen, weniger stabil läuft als vergleichbare Desktop-Software, werden wir uns hier im Buch stets auf die Desktop-Version beziehen.

2.6 Installation der Arduino-IDE

Die Arduino IDE ist für Windows, Mac OS X und Linux verfügbar. Im Folgenden werden wir uns auf die Installation unter Windows fokussieren, da dies sicherlich die meisten Leser betrifft. Die Installation in den anderen Betriebssystemumgebungen stellt jedoch ebenso keine größere Herausforderung dar, wenn Sie den Umgang mit Ihrem Betriebssystem gewohnt sind.

Zunächst laden wir die Installationsdatei von der offiziellen Arduino-Website:

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

Die Software wird dort sowohl als ausführbare Exe-Datei als auch als Zip-Archiv angeboten. Die erstgenannte ist die einfachere Variante für Einsteiger, da somit zeitgleich auch alle nötigen Treiber und Komponenten installiert werden. Allerdings sind dafür Administrator-Rechte unter Windows notwendig. Das Zip-Archiv ist etwas kleinteiliger in der Installation, benötigt jedoch keine Admin-Rechte. Als weitere Alternative besteht seit kurzem die Möglichkeit, die Software als App aus dem

4 http://dalpix.com/mariamole
5 http://playground.arduino.cc/Main/EmbedXcode
6 https://create.arduino.cc/editor

Microsoft Store zu laden, ein entsprechender Link findet sich ebenfalls auf der Website. Unabhängig vom gewählten Weg wird ein Spendenhinweis angezeigt. Da die Software kostenlos ist, finanziert sich die Entwicklung zu großen Teilen aus Spenden. Ob Sie dazu beitragen möchten, bleibt selbstverständlich Ihnen überlassen. Auf den Download hat dies keinen Einfluss.

Wir beziehen uns im Folgenden auf die Installation per Windows-Installer (Exe-Datei). Laden Sie die Datei auf Ihren Computer und öffnen Sie diese.

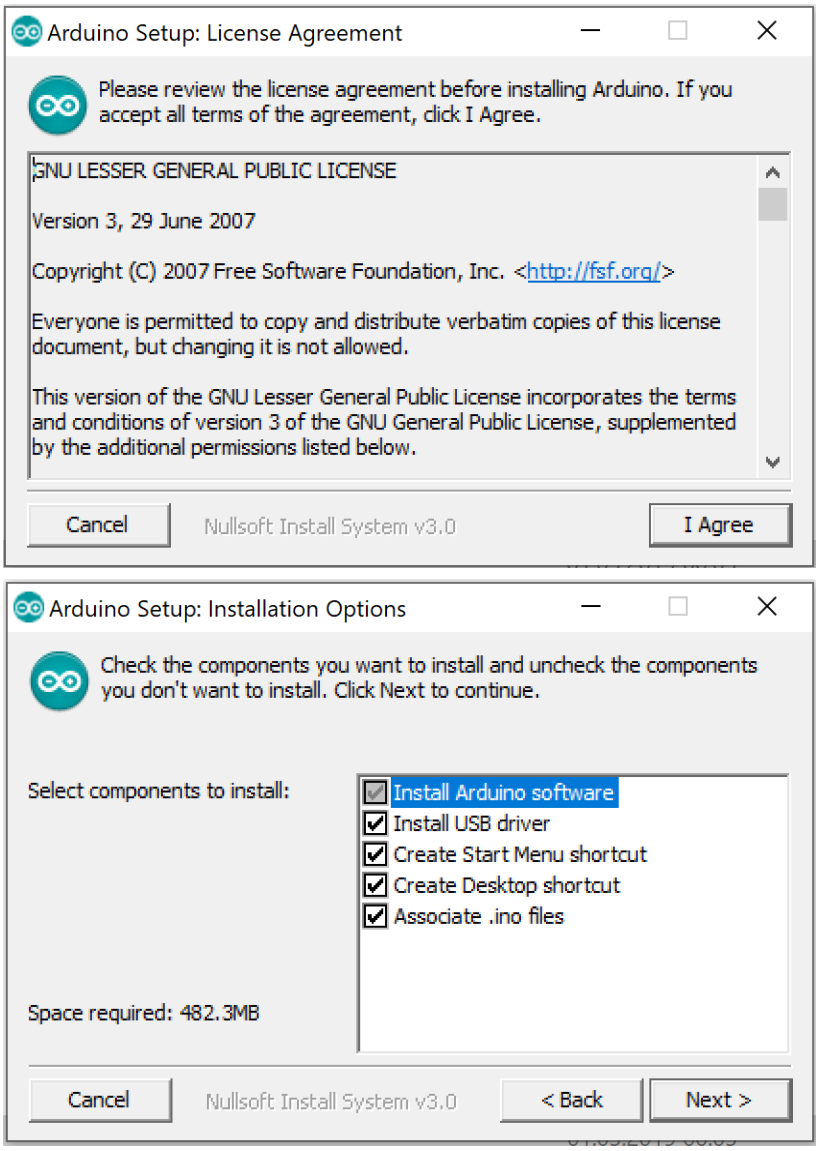

Abbildung 9 Installation der Arduino IDE

Zunächst werden Sie gebeten, den Lizenzvertrag zu lesen. Anschließend könne Sie wählen, welche Komponenten Sie installieren möchten. Die Arduino-Software und der USB-Treiber sollten auf jeden Fall angewählt werden. Ob Sie Desktop-, Startmenü- und Dateiverknüpfungen nutzen möchten, ist Ihren Vorlieben überlassen.

Anschließend startet die Installation, bei einem modernen Computer sollte sie nicht länger als 5 Minuten dauern.

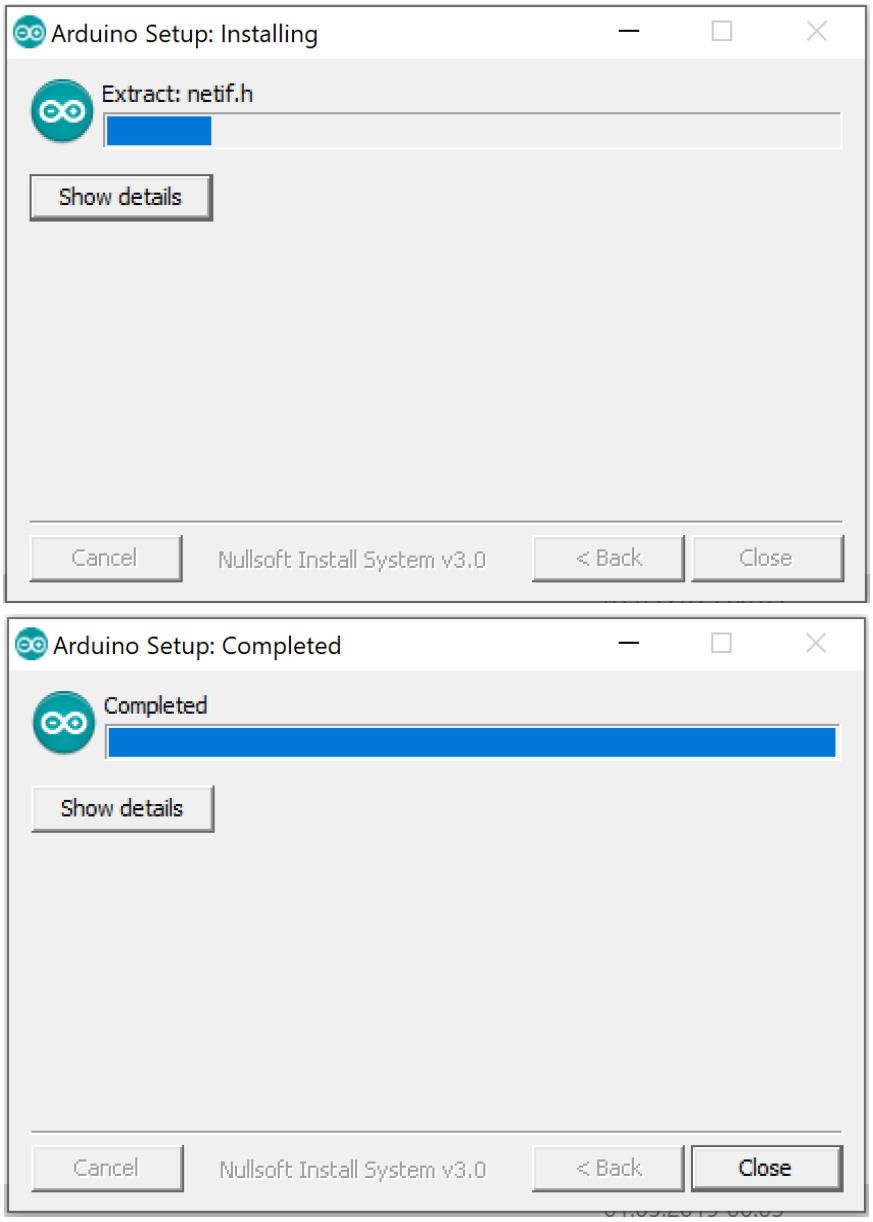

Abbildung 10 Nach wenigen Minuten ist die Installation durchlaufen.

Ist die Installation vollständig abgeschlossen, lässt sich dieses Fenster schließen und Sie können die Arduino IDE nun über den Desktop oder das Startmenü öffnen.

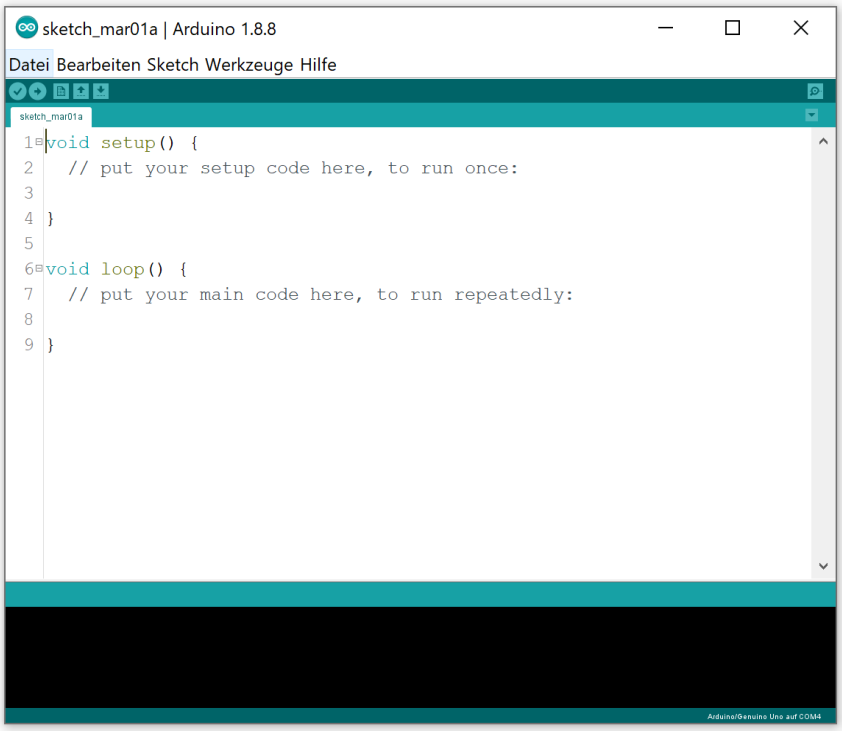

Abbildung 11 Die Arduino IDE – der mittlere Bereich mit weißem Hintergrund ist für den Programmcode vorgesehen; im schwarzen Balken am unteren Rand erscheinen etwaige Fehlermeldungen oder Hinweise.

Zunächst wird Ihnen eine nahezu leere Seite angezeigt. In diesem Feld erstellen Sie später den Programmcode, bei Arduino auch *Sketch* genannt. Im Kapitel 4 werden wir uns ausführlich damit beschäftigen. Im unteren schwarzen Feld werden Sie auf Probleme aufmerksam gemacht, falls der Sketch aus irgendeinem Grund nicht auf den Mikrocontroller übertragen werden kann.

Bevor Sie mit dem Programmieren loslegen, sollten Sie noch kurz zwei wichtige Einstellungen prüfen. Stecken Sie dazu Ihre Arduino-Platine per USB an den Computer an und schauen Sie nach, ob unter *Werkzeuge -> Board* das korrekte Board ausgewählt ist, welches Sie benutzen

möchten. Im Regelfall sollte das “Arduino/Genuino Uno” sein. Im Menüpunkt darunter muss ein COM-Port mit einem Häkchen versehen sein, die Zahl kann jedoch von diesem Beispiel abweichen.

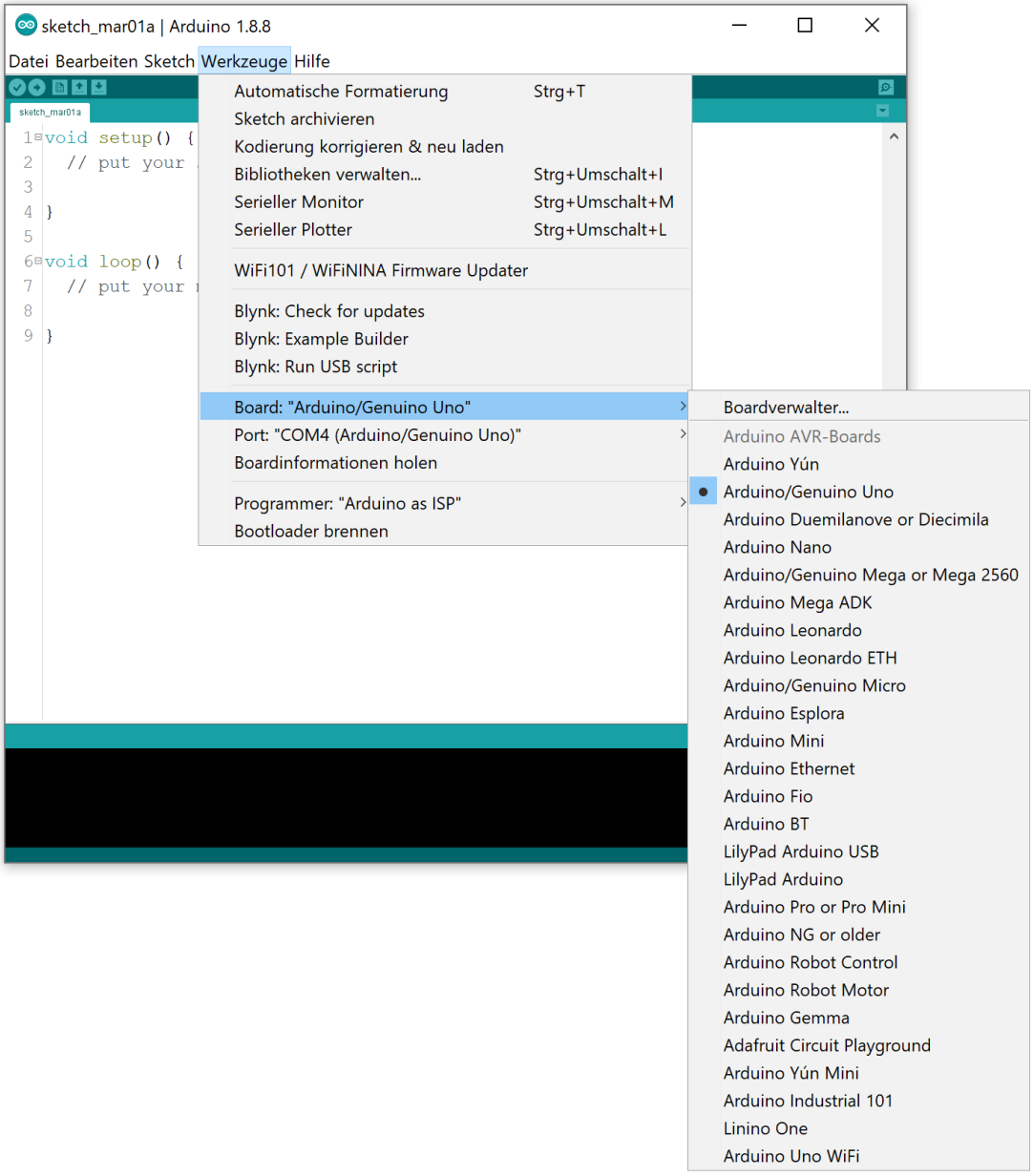

Abbildung 12 Die korrekte Auswahl des verwendeten Arduino-Boards ist wichtig.

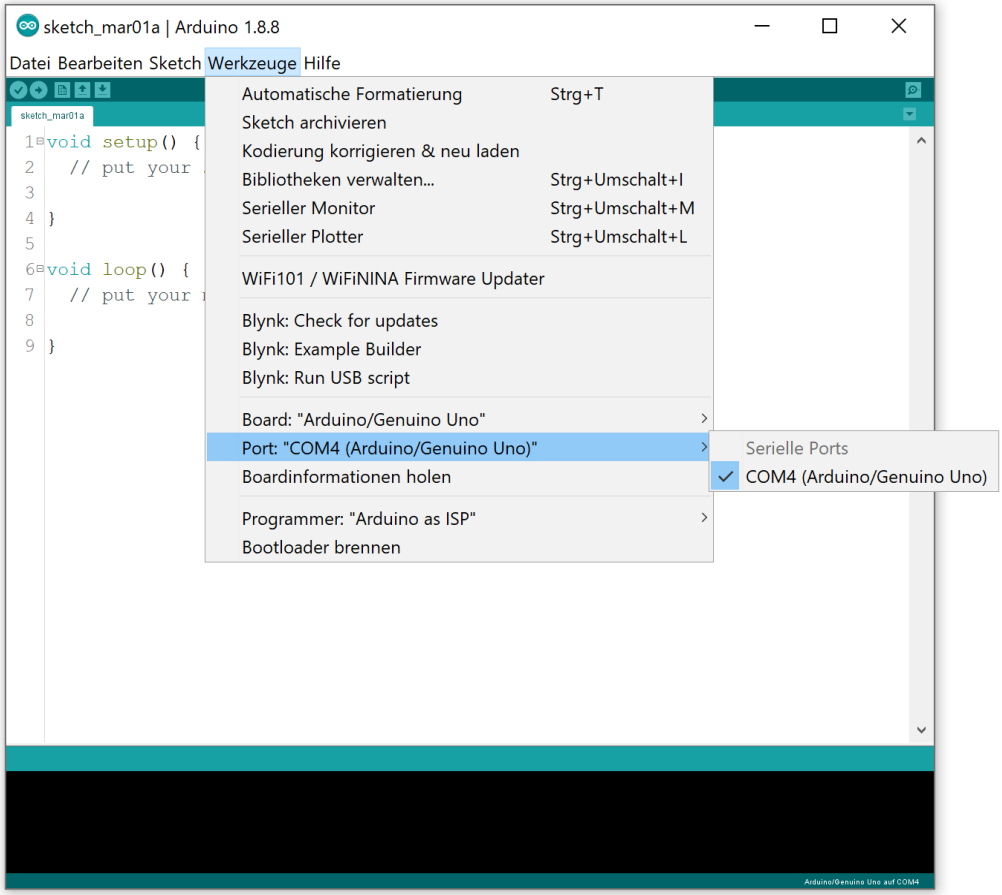

Abbildung 13 Trotz der USB-Verbindung muss als Anschluss ein serieller Port ausgewählt werden, da die Verbindung zum eigentlichen Arduino-Mikrocontroller letztlich seriell (über einen Wandler auf dem Board) erfolgt.

2.7 Bibliotheken

Die Arduino-Gemeinde lebt vom gegenseitigen Austausch. Daher ist es nicht selten, dass Nutzer ihren selbstgeschriebenen Programmcode für andere zur Verfügung stellen. Das geschieht entweder durch die Veröffentlichung des Sketches selbst oder einer Library-Datei, einer sogenannten Bibliothek. Diese können Sie sich wie ein Plug-In für Ihren Webbrowser vorstellen. Dementsprechend vielfältig sind die Einsatzzwecke: Einige dienen der Kommunikation mit angeschlossenen Sensoren oder Displays, andere wiederum erleichtern die Auswertung bestimmter Messdaten.

Über den Menüpunkt *Sketch -> Bibliothek einbinden -> Bibliotheken verwalten* kann in der Arduino IDE eine Übersicht der verfügbaren Bibliotheken angezeigt werden. Eine Suchfunktion erleichtert das Finden nach Stichworten. Fährt man mit der Maus über einen Eintrag, kann die entsprechende Bibliothek installiert oder aktualisiert werden.

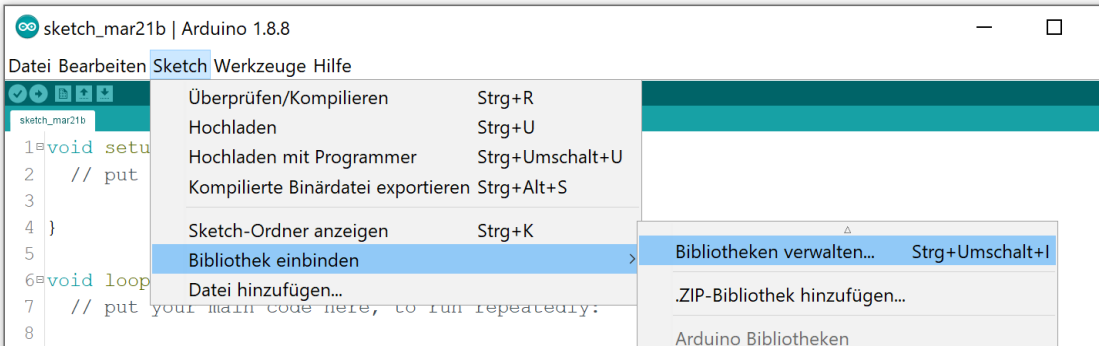

Abbildung 14 Der Bibliotheksverwalter kann einfach über das Menü geöffnet werden.

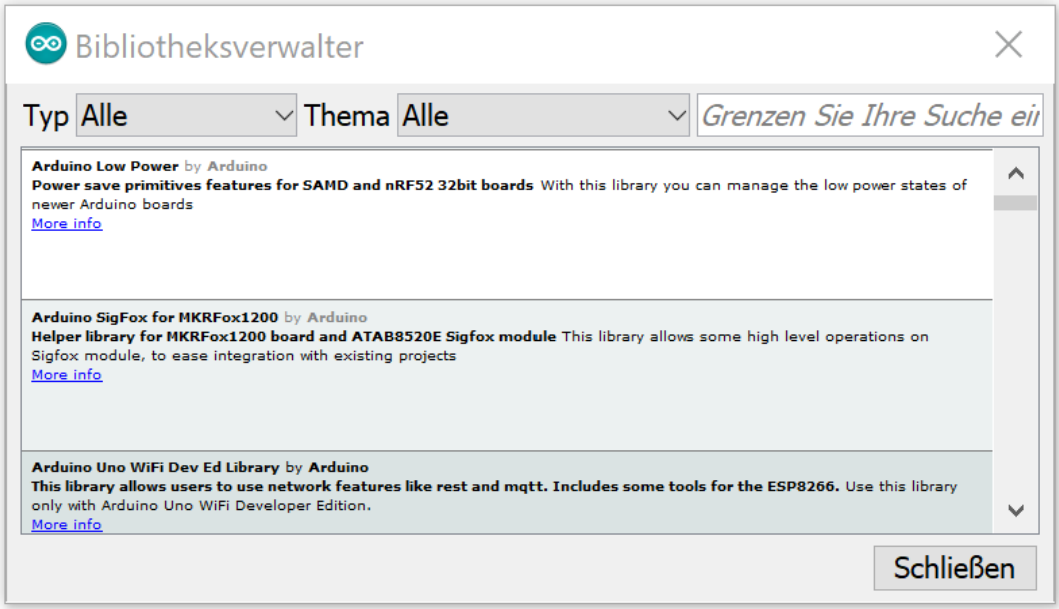

Abbildung 15 Der Bibliotheksverwalter ähnelt einer Suchmaske, welche den Server der Arduino-Website durchsucht.

Der Bibliotheksmanager durchsucht jedoch nur den Server des Arduino-Projektes. Es sind auf anderen Websites noch weitere Bibliotheken verfügbar, welche dann separat heruntergeladen und über den benach-

barten Menübefehl *.ZIP-Bibliothek hinzufügen* installiert werden können.

Grundsätzlich verhalten sich Bibliotheken passiv: Sie treten erst in Aktion, wenn sie explizit ins Programm eingebunden werden. Den dafür vorgesehenen Befehl `#include` lernen Sie in Kapitel 5.5.2 kennen, wenn er das erste Mal benötigt wird. Zusätzlich zum eigentlichen Programmcode sind oft noch Beispielsketche enthalten, welche automatisch dem Menü *Datei -> Beispiele* angefügt werden.

Die meisten Bibliotheken verfügen zudem über online-Dokumentationen, welche über die Handhabung informieren. Der Link dazu findet sich entweder in den Beispielsketchen oder kann über eine Internet-Suchmaschine gefunden werden. In selteneren Fällen enthalten die Bibliotheksdateien selbst eine Dokumentation, welche dann im Unterordner *\libraries* des Arduino-Verzeichnisses gefunden werden kann. Wir werden in späteren Kapiteln noch detaillierter auf bestimme Bibliotheken und deren Funktionen eingehen. Für den Anfang genügt es, wenn Sie den Begriff deuten können.

Nun sind Sie in Bezug auf die Software schon startklar. Das nächste Kapitel wird Ihnen nun noch ein paar elektronische Grundlagen vermitteln, bevor wir den ersten Sketch auf den Arduino laden.