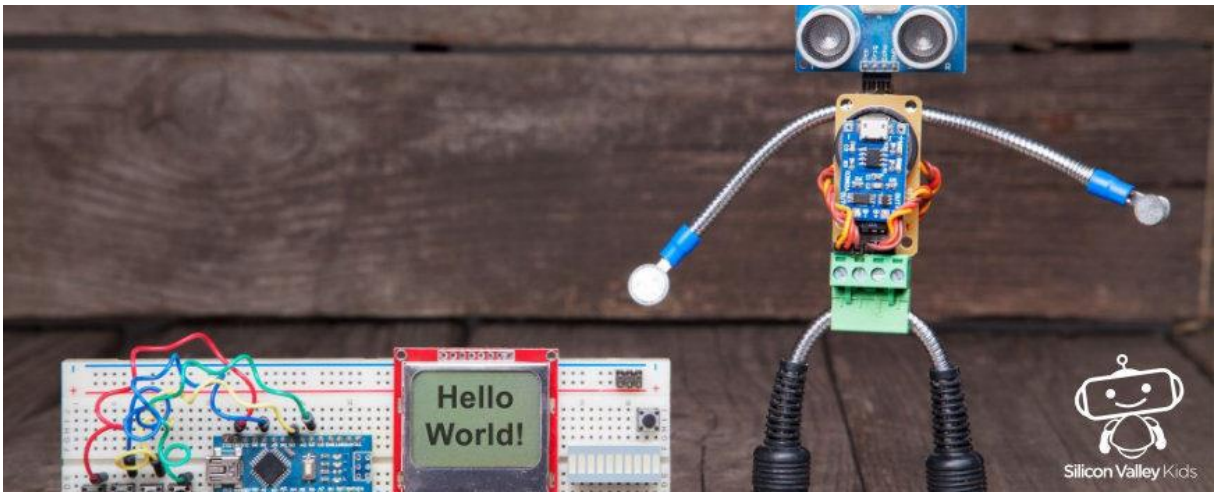
# Leitfaden für das Arbeiten mit dem Arduino

Willkommen beim Einstieg in die programmierbare Welt mit ARDUINO

Erstellt von Noah Gautschi, 12.02.2024




(Abbildung 1: Silicon Valley Kids, 2022)

# Inhalt

1 Was ist Arduino ..... 4
2 Was ist Arduino IDE ..... 5
2.1 Texteditor / Sketchbook ..... 5
2.2 Kompilierer ..... 5
2.3 Uploader ..... 5
2.4 Bibliotheksverwaltung / Library Manager ..... 5
2.5 Serielles Monitorfenster ..... 5
2.6 Board & Port ..... 6
2.7 Debugger ..... 6
3 Aufbau eines Arduino Programmes ..... 7
3.1 setup() Funktion ..... 7
3.2 loop() Funktion ..... 7
3.3 Zusätzliche Funktionen oder Deklarationen ..... 7
3.4 Variablendeklarationen ..... 7
3.5 Kommentare ..... 8
3.6 Beispiel ..... 8
4 Die wichtigsten Begriffe ..... 9
4.1 Grundlegendes ..... 9
4.2 Ausgaben / Eingaben ..... 9
4.3 Serielle Schnittstelle ..... 9
4.4 Abfragen/Schleifen ..... 9
4.5 Mathematisches ..... 10
4.6 Vergleiche ..... 10
4.7 Boolsche Operatoren ..... 10
4.8 Interrupts ..... 10
4.9 Sonstiges ..... 10
5 Arduino UNO Board Eigenschaften ..... 11
5.1 Mikrocontroller ..... 11
5.2 Digitale und analoge Pins ..... 11
5.3 USB-Anschluss ..... 11
5.4 Stromversorgung ..... 11
5.5 Reset-Taste ..... 11
5.6 ICSP-Header ..... 11
5.7 Board-LEDs ..... 11
5.8 Kurzschluss ..... 11

6 Arduino UNO Pinout ..... 12
7 Übungen ..... 13
7.1 Übung 01 Modi ..... 13
7.2 Übung 02 Serial Monitor ..... 14
7.3 Übung 03 Servomotor ..... 15
7.4 Übung 04 Ultraschallsensor ..... 16
7.5 Übung 05 Schrittmotor ..... 17
Abbildungsverzeichnis ..... 18
Literaturverzeichnis ..... 18

# 1 Was ist Arduino

Arduino ist eine Open-Source-Plattform, die entwickelt wurde, um das Prototyping von Elektronikprojekten zu erleichtern. Sie besteht aus einer Reihe von Hardwarekomponenten, darunter Mikrocontroller-Boards und eine Entwicklungsumgebung namens Arduino Integrated Development Environment (IDE).

Die Arduino IDE ist eine Softwareanwendung, die speziell für die Programmierung von Arduino-Mikrocontrollern entwickelt wurde.

Die Hardwareplattform von Arduino besteht aus verschiedenen Mikrocontroller-Boards, die jeweils unterschiedliche Funktionen und Spezifikationen bieten. Zu den beliebtesten Boards gehören Arduino Uno, Arduino Nano, Arduino Mega, Arduino Due und Arduino Leonardo. Diese Boards verfügen über verschiedene Ein- und Ausgangsanschlüsse, die es ermöglichen, Sensoren, Aktoren und andere elektronische Komponenten anzuschliessen und zu steuern.

Die Arduino-Plattform bietet eine breite Palette von Anwendungen und Projekten, darunter Robotik, Home Automation, Wearables, Kunstinstallationen und mehr. Dank der umfangreichen Community und des grossen Ökosystems von Arduino-Zubehör und -Bibliotheken gibt es nahezu unbegrenzte Möglichkeiten für kreative Projekte und Experimente.

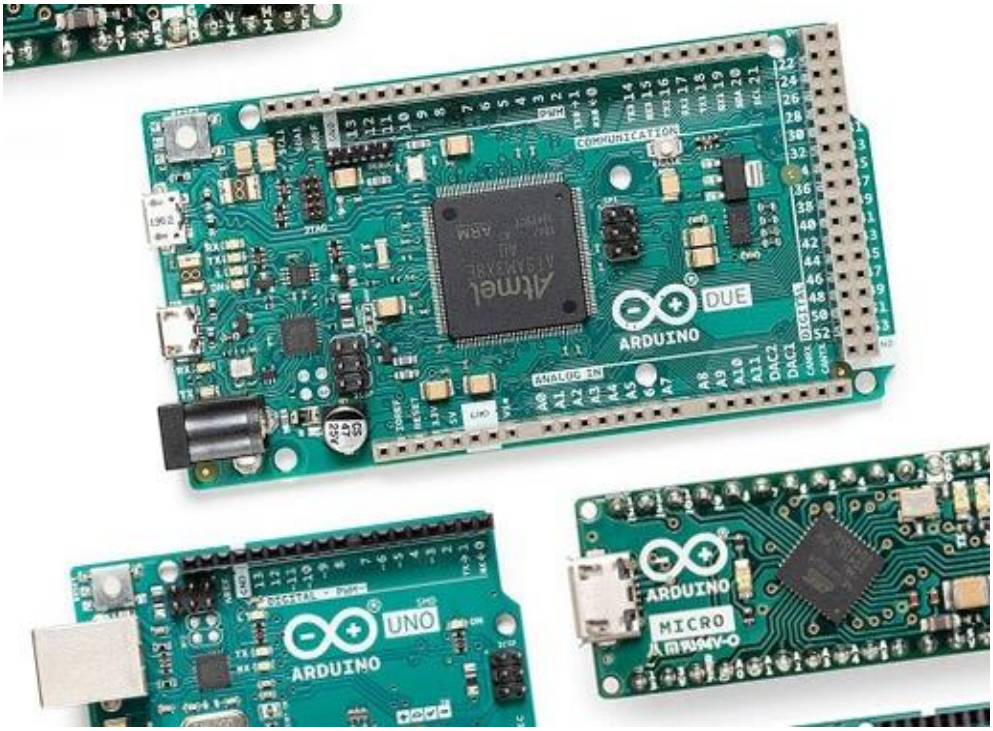



Abbildung 1: Die bekanntesten Arduino Boards (Arduino DUE oder auch MEGA, UNO und MICRO) (BerryBase, 2024)

# 2 Was ist Arduino IDE

Die Arduino Integrated Development Environment (IDE) ist eine Softwareanwendung, die speziell für die Programmierung von Arduino-Mikrocontrollern entwickelt wurde. Es handelt sich um eine plattformübergreifende Entwicklungsumgebung, die auf Windows, Mac OS X und Linux läuft und eine benutzerfreundliche Schnittstelle für die Entwicklung von Arduino-Projekten bietet.

Die Arduino IDE ermöglicht es Benutzern, Code für Arduino-Boards zu schreiben, zu kompilieren und auf diese hochzuladen. Sie enthält einen Texteditor, der das Schreiben von Programmcode in einer modifizierten Version von C/C++ erleichtert, sowie eine Vielzahl von Funktionen und Tools, die den Entwicklungsprozess unterstützen. Dazu gehören:

## 2.1 Texteditor / Sketchbook

Die IDE verfügt über einen integrierten Texteditor, der speziell für die Arduino-Programmierung optimiert ist. Er bietet Syntaxhervorhebung, automatische Vervollständigung und andere Funktionen, die das Schreiben von Code erleichtern.

## 2.2 Kompilierer

Die Arduino IDE enthält einen integrierten Compiler, der den geschriebenen Code in ausführbaren Maschinencode umwandelt. Dieser Schritt wird als Kompilierung bezeichnet und ist notwendig, um sicherzustellen, dass der Code auf dem Arduino-Board ausgeführt werden kann.

## 2.3 Uploader

Nach der Kompilierung kann der Benutzer den Code mithilfe der IDE auf das Arduino-Board hochladen. Die IDE enthält einen integrierten Uploader, der den Code über die USB-Schnittstelle des Computers auf das Board überträgt.

## 2.4 Bibliotheksverwaltung / Library Manager

Die Arduino IDE verfügt über eine Bibliotheksverwaltung, die es Benutzern ermöglicht, zusätzliche Bibliotheken von Drittanbietern zu installieren und zu verwalten. Diese Bibliotheken enthalten vorgefertigte Funktionen und Codebeispiele, die in eigenen Projekten verwendet werden können.

## 2.5 Serielles Monitorfenster

Die IDE enthält ein Serielles Monitorfenster (unter „open serial monitor/ plotter“ abrufbar), das es Benutzern ermöglicht, serielle Daten zu überwachen, die vom Arduino-Board an den Computer gesendet werden. Dies ist besonders nützlich für das Debugging von Code und die Überwachung von Sensorwerten. Ebenso kann dieses Fenster verwendet werden, um Inputs einzugeben.

## 2.6 Board & Port

In diesem Fenster wird das angeschlossene Board ausgewählt. Es sind eine Vielzahl an verschiedenen Modellen vorhanden, aus denen ausgewählt werden kann. Dazu muss der Port, respektive der verwendete Eingang am Pc, ausgewählt werden. Sollte der Port nicht bereits ersichtlich sein, kann er im Geräte-Manager (Windows) eingesehen werden.

## 2.7 Debugger

Er ermöglicht es den Code zu überprüfen, zu analysieren und zu debuggen. Dabei kann gibt es verschiedene Varianten ihn einzusetzen: Serial Monitor (Ausgabe von Variablenwerten, Debugging-Informationen und andere wichtige Informationen überwachen), Breakpoints (an bestimmten Stellen anhalten -> auf die Zeilennummern klicken), Step-by-Step-Ausführung (Verstehen des Programmflusses), Variablenüberwachung und Stack Trace (aktuellen Aufruflistenpfad und die Reihenfolge der Funktionen).


Abbildung 2: Ausschnitt der Benutzeroberfläche (GUI) der Arduino IDE auf einem Mac mit verschiedenen Beschreibungen (Söderby & Hylen, 2024)

# 3 Aufbau eines Arduino Programmes

Ein Programm in der Arduino IDE, auch als "Sketch" bezeichnet, ist in der Regel in zwei grundsätzliche Hauptteile (setup()-Funktion und die loop()-Funktion) unterteilt und noch mögliche Erweiterungen (zusätzliche Funktionen oder Deklarationen). Diese Struktur ermöglicht eine übersichtliche und organisierte Programmierung in der Arduino IDE. Der Code wird bei Zeile 1 begonnen zu lesen, nach unten hin mit aufsteigender Zeilenzahl wird damit fortgefahren. Die setup()-Funktion wird einmal beim Start des Programms ausgeführt, während der Code in der loop()-Funktion kontinuierlich wiederholt wird, solange das Programm läuft. Zusätzliche Funktionen und Variablen können je nach Bedarf hinzugefügt werden, um den Sketch zu erweitern und zu verbessern. „Startvariablen“ sollten daher vor die setup()-Funktion kommen, um zum einen eine Übersicht der verwendeten Variablen/Konstanten zu erhalten und alle Variablen/Konstanten bereits definiert zu haben.

## 3.1 setup() Funktion

Die setup()-Funktion wird einmal beim Start des Programms ausgeführt. Sie wird verwendet, um Initialisierungen und Konfigurationen vorzunehmen, die nur einmal ausgeführt werden müssen, z. B. das Festlegen von Pin-Modi oder das Konfigurieren von Variablen.

```
void setup() {pinMode(LED_BUILTIN, OUTPUT); } // Beispiel: LED als Ausgang definieren
```

## 3.2 loop() Funktion

Die loop()-Funktion wird kontinuierlich ausgeführt, nachdem die setup()-Funktion abgeschlossen ist. Hier befindet sich der Hauptteil des Programms, der wiederholt ausgeführt wird. In dieser Funktion werden typischerweise die meisten Aktionen und Anweisungen ausgeführt.

```
void loop() {                          // loop()-Funktion mit Schaltung einer LED
digitalWrite(LED_BUILTIN, HIGH);       // LED einschalten
delay(1000);                           // Eine Sekunde warten
digitalWrite(LED_BUILTIN, LOW);        // LED ausschalten
delay(1000); }                         // Eine Sekunde warten
```

## 3.3 Zusätzliche Funktionen oder Deklarationen

In diesem Abschnitt können zusätzliche Funktionen definiert werden, die im Hauptteil des Programms verwendet werden sollen. Dies kann dazu beitragen, den Code zu strukturieren und zu organisieren, insbesondere wenn der Sketch umfangreicher wird.

```
void myFunction() {}                   // Funktionscode
```

## 3.4 Variablendeklarationen

Hier werden Variablen definiert und initialisiert, die im Programm verwendet werden.

```
int sensorValue = 0;                   // Variable zur Speicherung eines Sensorwerts
```

## 3.5 Kommentare

Kommentare dienen dazu, den Code zu dokumentieren und zu erklären, was bestimmte Abschnitte des Codes tun. Sie sind hilfreich für andere Personen, die den Code lesen, oder für den Programmierer selbst, wenn er den Code zu einem späteren Zeitpunkt überarbeitet.

```
                                        // Dies ist ein Kommentar
```

## 3.6 Beispiel

```
const int ledPin = 13;                  // Pin für die LED
const int buttonPin = 2;                // Pin für den Taster
int previousButtonState = LOW;          // Variable, Speicherung des Taster-Zustands

void setup() {
pinMode(ledPin, OUTPUT);                // Definiere den Pin für die LED als Ausgang
pinMode(buttonPin, INPUT);              // Definiere den Pin für den Taster als Eingang}

void loop() {
int buttonState = digitalRead(buttonPin);           // Lese den aktuellen Zustand des Tasters
if (buttonState == HIGH && previousButtonState == LOW) {
                                        // Überprüfe, ob der Taster gedrückt wurde und
der vorherige Zustand LOW ist
digitalWrite(ledPin, !digitalRead(ledPin));         // Wechsle den Zustand der LED}
previousButtonState = buttonState;      // Aktualisiere den vorherigen Zustand des Tas-
ters
}
```

# 4 Die wichtigsten Begriffe

## 4.1 Grundlegendes

| | |
|---|---|
| Grundstruktur | void setup() { }void loop() { } |
| Variable deklarieren (Typ Integer, Wert = 0) | int led = 0; |
| Konstante deklarieren (Typ Integer, Wert = A0) | const int analogInPin = A0; |
| Array deklarieren (Typ Integer, 6 Werte) | int Arrayname[6] = {2, 4, -8, 3, 2}; |
| Pin-Funktion festlegen | pinMode(led, OUTPUT); |
| Library einbinden | #include <libX.h> |

## 4.2 Ausgaben / Eingaben

| | |
|---|---|
| Pin auf HIGH schalten | digitalWrite(led,HIGH); |
| PWM-Pin auf maximal Wert setzen | analogWrite(led, 255); |
| Digitalen Zustand von Pin schalter lesen | digitalRead(schalter); |
| Funktion: Analog Pin A0 auslesen | analogRead(A0); |
| Ton Ausgabe an Pin8, Frequenz x, länge y | tone(8, x, y); |
| Ton Ausgabe an Pin8 stoppen | noTone(8); |

## 4.3 Serielle Schnittstelle

| | |
|---|---|
| Serielle Schnittstelle initialisieren | Serial.begin(9600); |
| Text seriell Ausgeben | Serial.print(“Hallo”); |
| Text seriell Ausgeben + neue Zeile | Serial.println(“Hallo”); |
| Byte seriell Ausgeben | Serial.write(val); |
| Funktion: Anzahl von auslesbaren Bytes | Serial.available(); |
| Funktion: seriell Einlesen | Serial.read(); |
| Funktion: Text im Buffer finden (true/false) | Serial.find(„TEXT“); |
| Funktion: Text i. Buffer finden bis Endetext | Serial.findUntil(“TEXT“,“ENDE TEXT“); |
| Speichert Anzahl Bytes in Buffer | Serial.readBytes(buffer, length); |
| Speichert Bytes in Buffern bis lenght oder Ende-chracter erreicht | Serial.readBytesUntil(character, buffer, length); |
| Funktion: Findet nächste Integerzahl | Serial.parseInt(); |
| Prozedur aufgerufen wenn Daten vorhanden | void serialEvent(); |

## 4.4 Abfragen/Schleifen

| | |
|---|---|
| If-Abfrage mit Else | if (button == HIGH){ }else{ } |
| Switch-Abfrage | switch (x) { case 1: case 2: default: } |
| Wartezeit (500 ms) | delay(500); |
| For-Schleife | for (x = 2; x < 7; x++) { } |
| While-Schleife | while (schalter == HIGH) { } |
| oder | do{} while (schalter==HIGH); |
| Schleife vorzeitig verlassen | break; |
| Überspringt ausführbaren Quelltext in Schleife | continue; |

## 4.5 Mathematisches

| | |
|---|---|
| Sinus, Cosinus, Tangens | sin(),cos(),tan() |
| Wurzel aus X | sqrt(x) |
| X hoch Y : | xy pow(x, y) |
| Absolut Wert einer Zahl (Betrag von x) | abs(x) |
| Zufällige Zahl | random() |

## 4.6 Vergleiche

| | |
|---|---|
| gleich | == |
| ungleich | != |
| grösser, kleiner | <, > |
| Grösser oder gleich, kleiner oder gleich | <= , >= |
| Funktion: kleinere Zahl von x und y | min(x, y) |
| Funktion: grössere Zahl von x und y | max(x, y) |

## 4.7 Boolsche Operatoren

| | |
|---|---|
| Und | && |
| Oder | \|\| |
| Nicht | ! |
| Datentypen | Boolean; char; unsigned char; byte; int; unsigned int; word; long; unsigned long; short; float; double; string; String; |
| Datentypkonvertierungen | char(); byte(); int(); word(); long(); float() |

## 4.8 Interrupts

| | |
|---|---|
| Interrupts verbieten | noInterrupts(); |
| Interrupts erlauben | nterrupts(); |
| Interrupt an Pin mit Funktion verknüpfen | attachInterrupt(interrupt, function, mode); |
| Interrupt Verbindung aufheben | detachInterrupt(interrupt) |

## 4.9 Sonstiges

| | |
|---|---|
| Kommentar eine Zeile | // |
| Kommentar längerer Bereich | /* */ |
| Array definieren, Funktion: Wertebereich transformieren (z.B. 10bit Sensor auf 8bit PWM) | map(sensorWert, 0, 1024, 0, 255); |
| Funktion Wertgrenzen bestimmen (alles darüber/darunter wird abgeschnitten) | constrain(sensorWert, 0, 255); |
| Bein Funktionen: Wert zurückgeben | return x; |
| Zum Sprungpunk hier springen | goto hier; |
| Prozedur | Void Prozedurname() { } |
| Funktion mit Integer-Rückgabewert und Byte-Parameter | int Funktionsname (byte Parameter) {return 13;} |

Quelle: https://fkainka.de/befehlsliste-arduino/

# 5 Arduino UNO Board Eigenschaften

## 5.1 Mikrocontroller

Der Arduino Uno verwendet den ATmega328P-Mikrocontroller von Atmel. Dieser Mikrocontroller verfügt über 32 KB Flash-Speicher für Programme, 2 KB SRAM und 1 KB EEPROM. Er ist der Hauptprozessor des Arduino Uno und führt den von Benutzern hochgeladenen Code aus.

## 5.2 Digitale und analoge Pins

Der Arduino Uno verfügt über insgesamt 14 digitale Ein-/Ausgangspins, von denen 6 als PWM (Pulsweitenmodulation) -Pins fungieren. Diese Pins können zum Anschliessen von LEDs, Motoren, Sensoren und anderen digitalen Komponenten verwendet werden. Darüber hinaus hat der Arduino Uno 6 analoge Eingangspins, die zur Erfassung von analogen Signalen wie Temperatur, Licht oder Spannung genutzt werden können.

## 5.3 USB-Anschluss

Der Arduino Uno ist mit einem USB-Anschluss ausgestattet, der es ermöglicht, den Mikrocontroller mit einem Computer zu verbinden. Dieser Anschluss dient zum Hochladen von Sketches (Programmcode) auf den Arduino und zur seriellen Kommunikation zwischen dem Arduino und dem Computer. Der Anschluss benötigt man beim Hochladen von Sketches, in diesem Fall kann der Arduino sein Stromverbrauch direkt vom angehängten Pc erhalten. Er kann jedoch auch autonom betrieben werden. Einmal das Programm hochgeladen, kann er vom Pc getrennt werden und von einer externen Stromversorgung betrieben werden.

## 5.4 Stromversorgung

Der Arduino Uno kann über den USB-Anschluss oder über einen externen Gleichstromadapter mit Strom versorgt werden. Die Betriebsspannung beträgt 5 Volt, und der Mikrocontroller kann eine maximale Stromstärke von etwa 20 mA pro Pin liefern. Über den externen Gleichstromadapter kann er mit 5 – 12VDC Input umgehen.

## 5.5 Reset-Taste

Der Arduino Uno verfügt über eine Reset-Taste, die verwendet wird, um den Mikrocontroller neu zu starten und den aktuellen Sketch neu zu laden.

## 5.6 ICSP-Header

Der In-Circuit Serial Programming (ICSP) -Header ermöglicht das Programmieren des Mikrocontrollers mit einem ISP-Programmiergerät.

## 5.7 Board-LEDs

Der Arduino Uno hat zwei integrierte LEDs - eine grüne LED, die als Stromversorgungsanzeige dient, und eine rote LED, die mit Pin 13 verbunden ist und standardmässig in vielen Beispielsketches verwendet wird.

## 5.8 Kurzschluss

Auf der Unterseite gibt es hervorstehende Pins, die es nicht gerne haben, wenn man einen Kurzschluss herbeiführt (zwei oder mehrere Pins miteinander verbinden, Achtung, nicht auf metallene / elektrisch leitende Oberflächen legen!!). Ebenso ist es nicht ratsam die Stromversorgung auf derm Arduino kurzzuschliessen (3.3/5V mit GND verbinden ohne Verbraucher). Im Fall eines Kurzschlusses von der Stromversorgung trennen und die Reset-Taste betätigen. Falls nicht beschädigt sollte der Arduino wieder funktionieren. Falls nicht, bitte dem Dozenten melden.

# 6 Arduino UNO Pinout

Die Stromversorgung erhält der Arduino über den (schwarzen) Anschluss in der linken oberen Ecke (siehe Abbildung 2). Der USB-Anschluss sowie der Reset-Taster liegen auf der linken oberen Ecke. Auf der rechten Seite sind alle digitale I/O-Pins (Input und Output) im Gegensatz zur linken Seite liegen im unteren Bereich die analogen I/O-Pins. Ebenfalls auf der linken Seite in der Mitte ist die Stromversorgung mit Plus- und Minuspol (5V/3.3V und GND).

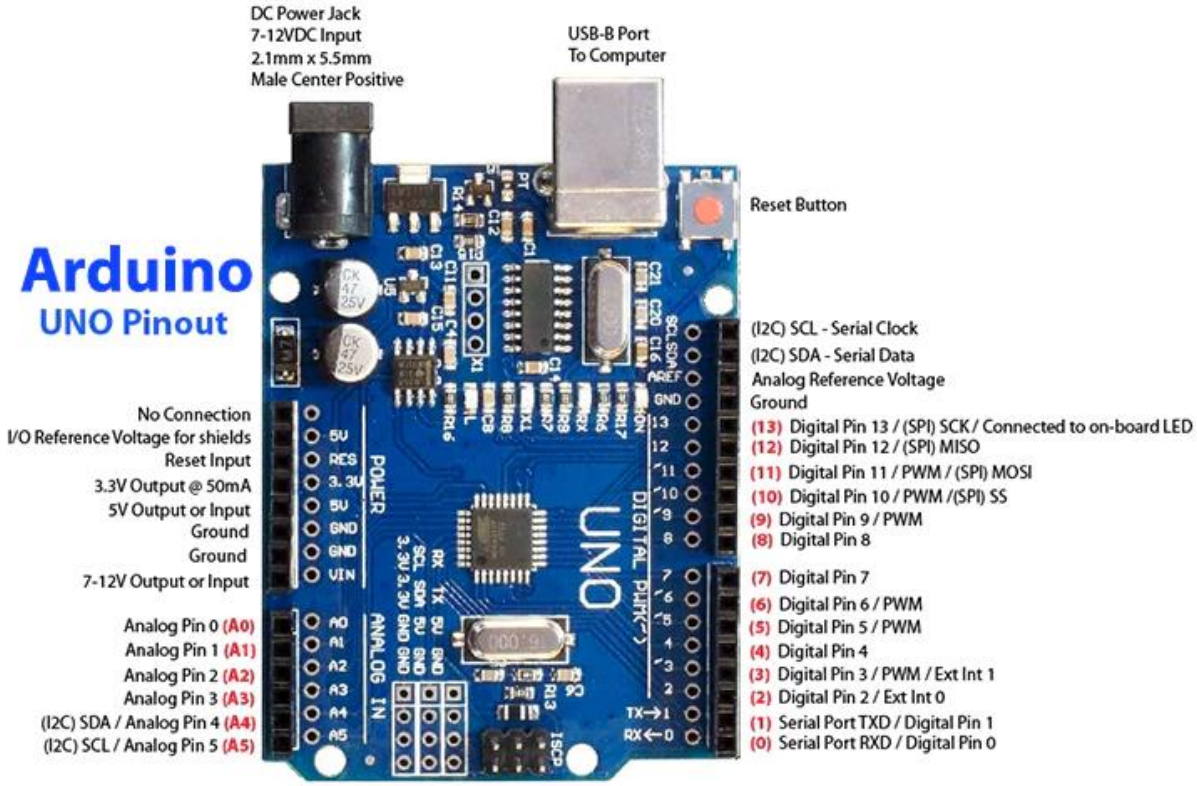



Abbildung 3: Der Arduino UNO in der Übersicht mit Pinbeschriftungungen (MakerBazar, 2024)

# 7 Übungen

## 7.1 Übung 01 Modi

Erstelle ein Sketch für einen Arduino UNO, um zwei LEDs über zwei Modi anzusteuern. Zwei Schalter sollen zwischen zwei Modi wechseln. Modus 1 und Modus 2 schalten jeweils eine LED ab und eine andere an und umgekehrt. Benutze dafür die folgenden Zuordnungen:

| | | | |
|---|---|---|---|
| LED1 auf Pin 10 | LED2 auf Pin 11 | Schalter1 auf Pin 1 | Schalter2 auf Pin 2 |
| digitalRead | digitalWrite | Modus 1 und 2 verwenden | Initialer Modus auf 0 |
| Initiale Schalterzustände auf LOW | | Aktuelle Schalterzustände | if und else if |

Projekt Hilfestellung:

1. Welche Komponenten wirst du brauchen?
2. Wie sind diese Komponenten miteinander verbunden?
3. Wie könnte eine solche Schaltung aussehen?
4. Welche Variablen und Konstanten könnten nützlich sein?

Code Hilfestellung:

5. Definiere die notwenigen Variablen und Konstanten
6. Aktiviere die Schnittstelle
7. Beginne mit dem Hauptteil des Codes
8. Lese die Zustände deiner Eingänge aus
9. Stelle den erwünschten Modus ein
10. Führe den erwünschten Modus aus
11. Achte auf deinen Zustand

## 7.2 Übung 02 Serial Monitor

Erstelle ein Sketch für einen Arduino UNO, um drei Schalter und ein Potentiometer auszulesen und im Serial Monitor darzustellen. Für jeden Schalter soll ein eigener Wert oder Wort ausgegeben werden. Für das Poti soll der Widerstandswert ausgeben werden. Benutze dafür die folgenden Zuordnungen:

| | | | |
|---|---|---|---|
| Schalter1 auf Pin 2 | Schalter2 auf Pin 3 | Schalter3 auf Pin 4 | Potentiometer auf Pin A0 |
| Initialer Zustand von Schalter 1, 2 und 3 auf LOW | | Schalter 1, 2 und 3 als Input-Pullup verwenden | |
| Serielle Schnittstelle und Monitor mit Baudrate 9600 | | | |

Projekt Hilfestellung:

1. Welche Komponenten wirst du brauchen?
2. Wie sind diese Komponenten miteinander verbunden?
3. Wie könnte eine solche Schaltung aussehen?
4. Welche Variablen und Konstanten könnten nützlich sein?

Code Hilfestellung:

5. Definiere die notwenigen Variablen und Konstanten
6. Aktiviere die Schnittstelle
7. Beginne mit dem Hauptteil des Codes
8. Lese die Zustände deiner Eingänge aus
9. Gib deine Zustände über den Serial Monitor aus
10. Füge eine Verzögerung ein zur Stabilisierung
11. Stelle den Serial Monitor auf dieselbe Baudrate wie die Schnittstelle

## 7.3 Übung 03 Servomotor

Erstelle ein Sketch für einen Arduino UNO, um einen Servomotor über das Drehen eines Potentiometers anzusteuern. Es soll dabei eine 1 zu 1 Verbindung von Poti auf die Drehung des Servomotors geben. Benutze dafür die folgenden Zuordnungen:

| | | |
|---|---|---|
| Servo.h als Library einbinden | analogRead | map(potState, 0, 1023, 0, 180) zur Normierung in Grad |
| Mit myservo.write den Servomotor ansteuern | | Initialer Potentiometer Zustand auf 0 |

Projekt Hilfestellung:

1. Welche Komponenten wirst du brauchen?
2. Wie sind diese Komponenten miteinander verbunden?
3. Wie könnte eine solche Schaltung aussehen?
4. Welche Variablen und Konstanten könnten nützlich sein?
5. Wie kann ich eine vorgefertigte Library einbinden?

Code Hilfestellung:

6. Benötigte Library installieren und einbinden
7. Definiere die notwendigen Variablen und Konstanten
8. Aktiviere die Schnittstelle
9. Beginne mit dem Hauptteil des Codes
10. Lese die Zustände deiner Eingänge aus
11. Gib deine Zustände über den Serial Monitor aus
12. Füge eine Verzögerung ein zur Stabilisierung

## 7.4 Übung 04 Ultraschallsensor

Erstelle ein Sketch für einen Arduino UNO, um ein Distanzsensor eines Autos auszulesen und eine Warnung auszugeben. Der Ultraschallsensor soll ausgelesen und die erhaltene Distanz soll in Zentimeter im Serial Monitor ausgegeben werden. Benutze dafür die folgenden Zuordnungen:

| | | |
|---|---|---|
| Trigger auf Pin 2 | Echo auf Pin 3 | Serielle Schnittstelle und Monitor mit Baudrate 9600 |
| digitalWrite(Trigger,LOW) oder HIGH | | if und else bei 10 Zentimeter Feedback: Stopp oder okay |
| Entfernung = (1/2)*Zeitdauer des Echo-Signals (in µs) * Schallgeschwindigkeit = t /29/2; | | |

Projekt Hilfestellung:

1. Welche Komponenten wirst du brauchen?
2. Wie sind diese Komponenten miteinander verbunden?
3. Wie könnte eine solche Schaltung aussehen?
4. Welche Variablen und Konstanten könnten nützlich sein?

Code Hilfestellung:

5. Definiere die notwenigen Variablen und Konstanten
6. Aktiviere die Schnittstelle
7. Beginne mit dem Hauptteil des Codes
8. Lese die Variable des Ultraschallsensors aus
9. Gib deine Variable über den Serial Monitor aus
10. Füge eine Verzögerung ein zur Stabilisierung

## 7.5 Übung 05 Schrittmotor

Erstelle ein Sketch für einen Arduino UNO, um einen Schrittmotor (Steppermotor) anzusteuern über Eingaben im Serial Monitor. Diese Eingaben sollen die Geschwindigkeit und Richtung des Schrittmotors steuern. Der Motor soll eine bestimmte Zeit mit den eingegebenen Parameter drehen. Nach dieser Zeit soll wieder eine neue Abfrage durchgeführt werden. Verwende zur Ansteuerung des Motors keinen externen Treiber. Benutze für diese Aufgabe die folgenden Zuordnungen:

| | | |
|---|---|---|
| Step für Geschwindigkeit auf Pin 3 | Dir für Richtung auf Pin 4 | Initiale Geschwindigkeit auf 0 |
| while (!Serial.available()) für den Puffer | Serielle Schnittstelle und Monitor mit Baudrate 9600 | |
| Serial.parseInt fürs Lesen von Integer | Serial.readString fürs Lesen von Strings | while |

Projekt Hilfestellung:

1. Welche Komponenten wirst du brauchen?
2. Wie sind diese Komponenten miteinander verbunden?
3. Wie könnte eine solche Schaltung aussehen?
4. Welche Variablen und Konstanten könnten nützlich sein?
5. Was für Berechnungen benötige ich?

Code Hilfestellung:

6. Definiere die notwenigen Variablen und Konstanten
7. Aktiviere die Schnittstelle
8. Beginne mit dem Hauptteil des Codes
9. Frage die Geschwindigkeit und Drehrichtung ab
10. Mit HIGHs und LOWs den Motor ansteuern

# Abbildungsverzeichnis

Abbildung 2: Die bekanntesten Arduino Boards (Arduino DUE oder auch MEGA, UNO und MICRO) (BerryBase, 2024) ..... 4
Abbildung 3: Ausschnitt der Benutzeroberfläche (GUI) der Arduino IDE auf einem Mac mit verschiedenen Beschreibungen (Söderby & Hylen, 2024)..... 6
Abbildung 4: Der Arduino UNO in der Übersicht mit Pinbeschriftungungen (MakerBazar, 2024)..... 12

# Literaturverzeichnis

MakerBazar. (2024, 8. Februar). *Arduino UNO R3 CH340G ATmega328(SMD)*. Verfügbar unter: https://makerbazar.in/products/arduino-uno-r3

Silicon Valley Kids (2022, 18. Februar). Arduino Projekte – 10 spannende Ideen. Verfügbar unter: https://www.sivakids.de/arduino-projekte/

Söderby, K. & Hylen, J. (2024). *Getting Started with Arduino IDE 2 | Arduino Documentation,* Arduino.cc. Verfügbar unter: https://docs.arduino.cc/software/ide-v2/tutorials/getting-started-ide-v2/