S 4 F

(Showers4Future)

Ein Jugend Forscht Projekt von

Oliver R. Zinzen (15)

Fachgebiet: **Physik**

Jahr: 2024/25

Bundesland: Brandenburg (Ost)

**Barnim Gymnasium Bernau**

Klasse 9M

Betreuung:

* Fr. Hänsel   
  (Mathematik & Physik)
* Fr. Gust   
  (Physik)

# Projektüberblick

# Inhaltsverzeichnis

[1 Projektüberblick 2](#_Toc187650211)

[2 Inhaltsverzeichnis 3](#_Toc187650212)

[3 Fachliche Kurzfassung 4](#_Toc187650213)

[4 Motivation und Fragestellung 5](#_Toc187650214)

[5 Hintergrund und theoretische Grundlagen 5](#_Toc187650215)

[5.1 Temperaturmessung. 6](#_Toc187650216)

[5.2 Wasserverbrauch 6](#_Toc187650217)

[5.3 Microcontroller 7](#_Toc187650218)

[6 Vorgehensweise, Materialien und Methoden 8](#_Toc187650219)

[6.1 Elektronik und Prototyp 8](#_Toc187650220)

[6.2 Programmieren 8](#_Toc187650221)

[6.3 Löten 8](#_Toc187650222)

[6.4 3D-Design & 3D-Druck 9](#_Toc187650223)

[6.5 Implementierung zu Hause 9](#_Toc187650224)

[7 Ergebnisse 9](#_Toc187650225)

[7.1 Der Prototyp 9](#_Toc187650226)

[7.2 Implementierung des Thermistors zur Temperaturmessung 12](#_Toc187650227)

[7.3 Ermittlung des CPL-Werts 13](#_Toc187650228)

[7.4 Die Software 15](#_Toc187650229)

[7.5 3D-Design und Druck 18](#_Toc187650230)

[8 Ergebnisdiskussion, Fazit und Ausblick 21](#_Toc187650231)

[8.1 Erste Eindrücke: 21](#_Toc187650232)

[8.2 Zukunft 21](#_Toc187650233)

[8.3 Ein sozialwissenschaftliches Experiment 22](#_Toc187650234)

[9 Quellen- und Literaturverzeichnis 23](#_Toc187650235)

[10 Unterstützungsleistungen 25](#_Toc187650236)

[11 Anhang 26](#_Toc187650237)

[11.1 Teileliste 26](#_Toc187650238)

[11.2 Messwerte zur CPL-Wert bestimmung 27](#_Toc187650239)

# Fachliche Kurzfassung

# Motivation und Fragestellung

Wasser und Energie sind essentielle Ressourcen, die wir täglich nutzen – sei es beim Duschen, Baden oder Händewaschen. Doch meist wissen wir nicht, wie viel Wasser und Energie wir dabei tatsächlich verbrauchen. In meiner Facharbeit möchte ich ein Gerät entwickeln, das uns hilft, unseren Wasser- und Energieverbrauch besser zu verstehen und diese wertvollen Ressourcen bewusster einzusparen. Hierzu plane ich, einen Adapter zu entwickeln, der an eine Wasserarmatur angeschlossen werden kann. Dieser Adapter misst Wasserfluss und Temperatur und informiert den Nutzer über den Verbrauch.

Die zentrale Fragestellung meiner Arbeit lautet: "*Wie entwickle ich eine Elektronik, die Wasserfluss und Temperatur misst, die gesammelten Daten an den Nutzer übermittelt und wie integriere ich die Sensoren in einen Armatur-Adapter?*" Mein Adapter soll "Showers4Future" (kurz: "S4F") heissen.

Dieses Thema fasziniert mich, weil es Technik und Umweltschutz miteinander verbindet. Ich bin überzeugt, dass wir genaue Informationen über unseren Verbrauch benötigen, um unser Verhalten gezielt und nachhaltig zu ändern. S4F soll uns dabei unterstützen, achtsamer mit Wasser und Energie umzugehen.

# Hintergrund und theoretische Grundlagen

Die Idee für S4F kam mir im Winter 2022. Meine Familie und ich waren besorgt über die steigenden Energiekosten nach der Invasion der Ukraine durch Russland. Ich lebe mit meinen Eltern und drei Brüdern zusammen. Mir wurde klar, dass wir viel Energie für Duschen verwenden. Aber niemand konnte mir genau sagen, wie viel Wasser und Energie wir tatsächlich beim Duschen verbrauchen. Also habe ich beschlossen, diesen Verbrauch zu messen.

Ich bin überzeugt, dass wir, wenn wir genaue Informationen über unseren Wasser- und Energieverbrauch hätten, gezielt darauf reagieren könnten. S4F soll nicht nur helfen wichtige Ressourcen zum Wohl der Umwelt zu schonen, sondern auch unsere Ausgaben deutlich zu reduzieren.

## Temperaturmessung.

Um die Temperatur zu messen, dachte ich sofort an einen Thermistor. Dieses elektronische Bauteil war mir ein paar Monate zuvor in einem Arduino-Starter-Kit[[1]](#endnote-1) begegnet. Ein Thermistor ist ein elektrischer Widerstand mit der besonderen Eigenschaft, dass sein Widerstandswert von der Temperatur abhängt. Es gibt zwei Arten von Thermistoren: NTC (Negative Temperature Coefficient) und PTC (Positive Temperature Coefficient). Bei steigender Temperatur sinkt der Widerstand eines NTCs, während er bei einem PTC steigt.

Bei meinen Recherchen im Internet stieß ich auf die sogenannte Steinhart-Hart-Gleichung. Mit dieser Gleichung lässt sich die Temperatur anhand des Widerstandswerts eines Thermistors und einiger spezifischer Konstanten des Bauteils berechnen[[2]](#endnote-2). Die Herausforderung bestand jedoch darin, den Widerstand des Thermistors auszulesen, da dies nicht direkt mit einem Mikrocontroller wie z.B. einem Arduino möglich ist.

Dank meines Elektronik-Lernsets1 wusste ich, dass der Widerstand eines Thermistors indirekt gemessen werden kann, indem man ihn in einem Spannungsteiler verwendet und die Spannung über den Thermistor misst. Wenn die Spannung über den Thermistor und die Gesamtspannung des Spannungsteilers bekannt sind, lässt sich mithilfe des Ohm’schen Gesetzes der Widerstand des Thermistors berechnen. Den so ermittelten Widerstandswert kann ich schließlich in die Steinhart-Hart-Gleichung einsetzen, um die Temperatur zu bestimmen.

## Wasserverbrauch

Ich habe mich im Internet nach Sensoren umgesehen, mit denen ich den Wasserverbrauch messen kann und bin dabei auf sogenannte Durchflusssensoren (Flow Sensors) gestoßen, die meist auf dem Hall-Effekt basieren.

Der Hall-Effekt beschreibt, dass Elektronen in einer geraden Linie durch eine leitfähige Platte wandern, wenn ein elektrischer Strom durch die Platte fließt. Befindet sich jedoch ein Magnet in der Nähe, werden die Elektronen zur Seite abgelenkt, wodurch eine Spannung über der Platte entsteht. Diese Spannung kann gemessen werden[[3]](#endnote-3).

Ein Durchflusssensor (Hall-Effekt-Sensor, HES) nutzt diesen Effekt: Er besitzt einen Propeller, der sich durch den Wasserfluss dreht. Die Propellerflügel sind dabei mit kleinen Magneten ausgestattet. Wenn sich der Propeller dreht, passieren die Magneten wiederholt den Hall-Effekt-Sensor und erzeugen Spannungsimpulse. Ich gehe davon aus, dass die Frequenz dieser Impulse proportional zur Wassergeschwindigkeit ist.

Diese Spannungsimpulse und ihre Frequenz sollte ich mit einem Mikrocontroller erfassen und zählen können, um so den Wasserverbrauch zu bestimmen.

## Microcontroller

Um den Thermistor und den Hall-Effekt-Sensor (HES) auszulesen, plane ich, einen Mikrocontroller zu verwenden. Es gibt derzeit mehrere beliebte Plattformen, wie den Raspberry Pi und Arduino. Mikrocontroller sind Chips, die programmierbar sind und Signale verarbeiten und ausgeben können. Solche Mikrocontroller sind als sogenannte Entwicklungsboards (Dev-Kits) erhältlich, bei denen der Mikrocontroller-Chip über General Purpose Input/Output (GPIO)-Pins zugänglich ist. An diese Pins können Sensoren, Motoren oder andere Geräte angeschlossen werden, die dann durch passende Programmierung ausgelesen oder gesteuert werden können.

In einem vorherigen Projekt habe ich zusammen mit meinem Bruder einen Raspberry Pi verwendet, um in Python eine selbstoptimierende Wind- und Solaranlage (SoWiSo) zu entwickeln. Dabei fiel mir jedoch auf, dass der Raspberry Pi ein vollwertiger Unix-Computer ist und vergleichsweise viel Energie verbraucht. Da das S4F-Gerät batteriebetrieben sein soll, ist ein energieeffizienter Mikrocontroller erforderlich.

Aus diesem Grund habe ich mich für die Arduino-Plattform entschieden. Arduino-Boards sind deutlich energiesparsamer als der Raspberry Pi und bieten dennoch ausreichend Leistung für mein Vorhaben. Das Arduino Uno R3 Dev-Kit kann entweder über eine 5V-USB-Verbindung oder eine andere Gleichstromquelle (bis zu 12V) betrieben werden. Es verfügt über 14 digitale und 6 analoge GPIO-Pins, die sich flexibel als Eingänge zum Einlesen von Signalen oder als Ausgänge zum Steuern von Geräten programmieren lassen.

Die Arduino-Plattform verwendet C++ als Programmiersprache. Der Code kann mit der kostenlosen Entwicklungsumgebung Arduino IDE (Integrated Development Environment) [[4]](#endnote-4) geschrieben, getestet und auf den Mikrocontroller geladen werden. Zudem bietet die Arduino-Community eine ausgezeichnete Unterstützung: Es gibt zahlreiche Anleitungen, Foren und fertige Codebeispiele, die den Einstieg erleichtern[[5]](#endnote-5). Darüber hinaus ist es relativ einfach, ein Arduino-Board mit einem LCD-Display zu verbinden, um dem Nutzer Anweisungen anzuzeigen oder Daten zu präsentieren.

# Vorgehensweise, Materialien und Methoden

## Elektronik und Prototyp

Den ersten Prototyp zur Auslesung der Sensoren mit einem Arduino habe ich mithilfe einer Steckplatine entwickelt. Diese ermöglicht es, elektronische Komponenten durch einfaches Einstecken von Jumper-Kabeln zu verbinden. Eine Liste der verwendeten Komponenten befindet sich im Anhang (siehe 11.1). Die wichtigsten Komponenten sind:

* **Arduino Uno R3 DevKit:** Ein Arduino-Entwicklerboard mit 20 GPIO-Pins, das als zentrale Steuereinheit dient.
* **Durchflusssensor:** Ein Wasserdurchflusssensor[[6]](#endnote-6) (bestellt bei Amazon), der mithilfe des Hall-Effekts Durchflussmengen zwischen 1 und 30 L/min messen kann. Der Sensor hat ein ½-Zoll-Gewinde.
* **Thermistor:** Ein NTC-Temperatursensor (TRU Components, Modell MJSTS-103-3950-1-600-3D[[7]](#endnote-7), bestellt bei Conrad Electronic) mit einem 10 kΩ Referenzwiderstand bei 25°C. Ein verfügbares Datenblatt enthält alle technischen Details.
* **Steckplatine und Elektronikkomponenten:** Die Steckplatine sowie verschiedene Bauteile wie Widerstände, Taster, Potentiometer, Jumper-Kabel und ein 16x2 LCD-Display stammen aus mehreren Elektronik-Lernsets1,[[8]](#endnote-8),[[9]](#endnote-9).

Mit dieser Hardware konnte ich die grundlegenden Funktionen des Prototyps testen und die Sensoren erfolgreich in den Arduino integrieren.

## Programmieren

Für die Programmierung habe ich die Software Arduino IDE verwendet. Mit dieser Entwicklungsumgebung kann man Code direkt in C++ schreiben, auf Syntaxfehler prüfen und anschließend auf ein angeschlossenes Arduino-Board übertragen. Ein besonders nützliches Feature der Arduino IDE ist der Serial Monitor. Dieser ermöglicht es, dass der Arduino während des Programmablaufs Nachrichten sendet. So können potenzielle Probleme im Code einfach diagnostiziert und behoben werden.

Die Grundlagen des Arduino-Programmings habe ich mithilfe eines Arduino-Übungsbuchs8 gelernt. Bei Fragen oder Herausforderungen konnte ich meist hilfreiche Antworten im Arduino-Wiki5 finden.

## Löten

Nachdem ich einen funktionierenden Prototypen auf einer Steckplatine gebaut hatte, plante ich, ein sogenanntes „Shield“ zu erstellen. Ein Shield ist eine Platine, die direkt auf den Arduino aufgesteckt werden kann und auf der alle Schaltkreise fest verlötet sind.

Zunächst habe ich mit der CAD-Software Autodesk Eagle[[10]](#endnote-10) (kostenlos für Schüler\*innen) die Schaltkreise aufgezeichnet und dokumentiert, wie diese mit dem Arduino und dem LCD-Display verdrahtet waren. Anschließend entwickelte ich einen detaillierten Lötplan.

Die Umsetzung erfolgte auf einer Lochrasterplatine (90mm x 70mm, Rastermaß 2,54mm). Alle benötigten Komponenten wurden gemäß des Lötplans auf die Platine aufgebracht, mit Silberdraht verbunden und sauber verlötet, um eine robuste und dauerhafte Verbindung zu gewährleisten.

## 3D-Design & 3D-Druck

Für das S4F-Gerät habe ich zwei Bauteile selbst gedruckt: die Thermistor-Halterung, die mit der Armatur und dem Flusssensor verschraubt wird, sowie das Gehäuse für die Elektronik. Beide Teile habe ich mit der Software Autodesk Fusion 360[[11]](#endnote-11) (kostenlos für Schüler\*innen) entworfen.

Die 3D-Modelle habe ich anschließend als STL-Dateien exportiert und mit einem 3D-Drucker (Keyence Agilsta 3200W[[12]](#endnote-12)) gedruckt. Der Agilsta ist ein professioneller 3D-Drucker, der anstelle des herkömmlichen PLA-Filaments ein foto-aktivierbares Resin verwendet. Dadurch kann der Agilsta eine höhere Auflösung sowie bessere Materialfestigkeit und Hitzebeständigkeit erreichen.

## Implementierung zu Hause

Das erste S4F Gerät habe ich in unserer Dusche zu Hause installiert, und wir haben es circa eine Woche lang benutzt. Von unseren ersten Erfahrungen werde ich berichten.

# Ergebnisse

Die Entwicklung des S4F-Geräts hat längere Zeit in Anspruch genommen. Besonders beim ersten Prototyp auf der Steckplatine musste ich regelmäßig Komponenten ändern und umstecken, was parallel zur Softwareentwicklung lief. Zum Beispiel, als ich auf die Idee kam, ein LCD-Display zu verwenden, musste ich die Verkabelung komplett überarbeiten, um die benötigten GPIO-Pins zu belegen und die Betriebssoftware darauf abzustimmen. Ebenso, als ich mehr über die Thermistor-Temperatur-Berechnung recherchierte, musste ich einen neuen Spannungsteiler implementieren. In den folgenden Ergebnissen werde ich deshalb hauptsächlich die Endergebnisse darstellen, aber keine detaillierte Beschreibung der Umwege und Anpassungen während der Entwicklung geben.

## Der Prototyp

Den ersten Prototypen habe ich auf einer Steckplatine verkabelt, wobei der Arduino über USB mit 5V versorgt wurde. Das Steckbrett ist in Abbildung 1B dargestellt, und der schematische Schaltplan ist in Abbildung 1A zu sehen. Die wichtigsten Komponenten des Prototyps sind:

* **Arduino Uno R3** Mikrocontroller mit 14 digitalen GPIOs und 6 analogen GPIOs (die auch als digitale Pins genutzt werden können). Zusätzlich verfügt der Arduino über mehrere Power-Pins, von denen ich den „5V“-Pin als positiven Pol (5V+) und den „Gnd“-Pin als negativen Pol (Gnd) für meine Schaltung verwendet habe. Zur Verkabelung nutzte ich schwarze Kabel für Gnd und rote Kabel für 5V+ (Abbildung 1B).
  + **16x2 LCD-Display Modul:** Das Display kann 16 Zeichen in 2 Reihen darstellen. Wie im Arduino Wiki beschrieben[[13]](#endnote-13), ist das LCD-Display mit 5V+ und Gnd verbunden. Der V0-Pin des Displays ist mit einem 10 kΩ Potentiometer verbunden, um den Displaykontrast durch Regulierung des elektrischen Potentials am V0-Pin einzustellen. Um dem Arduino das Senden von Text an das Display zu ermöglichen, ist der Arduino mit den Pins En, RS sowie D4, D5, D6 und D7 des Displays verbunden. Dadurch werden insgesamt 6 digitale GPIO-Pins des Arduinos benötigt.
  + **Taster:** Ein Taster ermöglicht es dem Arduino, auf Nutzereingaben zu reagieren. Wenn der Taster betätigt wird, schließt er einen Stromkreis zwischen 5V+ und Gnd über einen 10kΩ Widerstand, wodurch der digitale Pin auf 5V gesetzt wird. Ein 7. Digitaler Pin wird so programmiert, dass er den Unterschied zwischen 0V (LOW) und 5V (HIGH) erkennt, womit der Ardino auf die Knopfbetätigung reagieren kann.
  + **Hall-Effekt-Sensor:** Der Hall-Effekt-Sensor hat drei Kabel. Zwei (rot und schwarz) dienen der Stromversorgung und sind an 5V+ und Gnd angeschlossen. Über das dritte Kabel (grün) wird der Hall-Impuls an einen 8. digitalen GPIO-Pin übertragen, die vom Arduino dann gezählt werden können
  + **Thermistor:** Um den Thermistor auszulesen, verwende ich einen Spannungsteiler, bei dem ein konstant bleibender Widerstand und der temperaturabhängige Thermistor in Reihe zwischen 5V+ und Gnd geschaltet sind. Da der Thermistor einen Referenzwert von 10 kΩ bei 25°C hat, habe ich einen 10 kΩ Widerstand als konstante Komponente verwendet. So wird die Spannung bei 25°C halbiert und Widerstandsunterschiede im Thermistor sollten gut erfassbar sein. Die Spannung über den Thermistor wird an einen 9. GPIO-Pin des Arduinos übertragen. Dies muss ein analoger Pin sein, damit der Arduino die Größe der Spannung wie in 7.2 beschrieben auslesen kann.

Nachdem der funktionierende Schaltplan stand, um den Taster, den Thermistor und den Hall-Effekt-Sensor auszulesen, habe ich mir einen Lötplan erstellt (Abbildung 1B). Anschließend habe ich die Schaltung auf einer **Lochrasterplatine** umgesetzt. Da ich eine Platine mit einem Lochrastermaß von 2,54 mm verwendete, konnte ich die Lötstiftleisten so anordnen, dass die Platine direkt auf den Arduino aufgesteckt werden konnte, um eine stabile Verbindung zu gewährleisten. Zusätzlich habe ich eine Buchsenleiste eingelötet, sodass das LCD-Display mit seinen 16 Stiften (12 davon belegt) direkt auf die Platine aufgesetzt werden konnte. Dies ermöglichte es, das Display und den Arduino bei Bedarf wieder zu trennen, falls ich Änderungen an der Platine vornehmen musste.

|  |
| --- |
| Abbildung 1: Prototyp-Schaltplan uns Shield-Entwicklung  Der Schaltplan (A) basiert auf dem Steckbrett Prototyp (B). Auf (A) basierend wurde ein Lötplan (C) entwickelt was mir erlaubte ein Arduino-Shield mit fest verlötet Elektronikkomponenten zu bauen (D). Detaillierte Pläne (A, C) stehen zum Download auf GitHub bereit[[14]](#endnote-14). |

## Implementierung des Thermistors zur Temperaturmessung

### Berechnung des Thermistorwiderstands

Da der Thermistrorwiderstand mit einem Arduino nicht direkt ausgelesen werden kann, werde ich ihn mit Hilfe des Ohm’schen Gesetzes[[15]](#endnote-15) anhand der gemessenen Spannung berechnen (). Die Spannung über den Thermistor kann über die analogen GPIO-Pins eingelesen werden, da diese mit einem 10-Bit-ADC (Analog-to-Digital Converter) ausgestattet sind. Der ADC wandelt die Spannung am Pin relativ zur internen Referenzspannung () in einen Wert zwischen 0 – 1023 (10-Bit-Auflösung) um.

Ich benutze einen Spannungsteiler, der aus zwei Widerständen in Reihe besteht: R1 (ein konstanter Widerstand von 10kΩ) und RTher (der Widerstand des Thermistors). Der Gesamtwiderstand des Spannungsteilers ist: .

Nach dem Ohm‘schen Gesetz15 gilt:

Daraus folgt:

Die 5V Eingangsspannung teilt sich über Widerstände in Reihe auf:

Die Spannung über den Thermistor ist nach dem Ohmschen Gesetz definiert als:

Da die Stromstärke überall im Stromkreis gleich ist gilt und der Widerstand des Thermistors RTher kann berechnet werden:

Mit , , und dem Wert für UTher, den der Arduino über den ADC ausliest kann der Thermistorwiderstand berechnet werden.

### Temperaturberechnung

Mit Hilfe der Steinhart-Hart-Gleichung lässt sich die Temperatur *T* in Kelvin aus dem Widerstand  berechnen:

Die Werte *A, B,* und *C* Thermistor-spezifische Konstanten, die ich mit einem Online-Thermistor Rechner[[16]](#endnote-16) und den Widerstandswerten aus dem Thermistor Datenblatt9 ermittelt habe:

Um die Temperatur in °C umzurechnen, ziehe ich anschließend noch 273,15 ab:

## Ermittlung des CPL-Werts

Um eine zuverlässige Konstante zur Umrechnung der Hall-Effekt-Impulse in Liter zu ermitteln, habe ich die CPL-Konstante (Counts per Liter) bestimmt. Diese Konstante ermöglicht es dem Arduino, das verbrauchte Wasservolumen durch einfache Division der gezählten Impulse durch den CPL-Wert zu berechnen.

### Versuchsaufbau

Ich habe ein Utility-Programm (Arduino Sketch verfügbar auf GitHub14) geschrieben, das die Hall-Effekt-Impulse zählt. Der Start und Stopp der Zählung erfolgt über einen Knopf. Der Versuch wurde folgendermaßen durchgeführt:

1. Durchflusssensor wurde direkt an der Duscharmatur installiert.
2. Ein Eimer wurde mit Markierungen bei 5L, 8L und 10L versehen.
3. Während das Wasser in den Eimer lief, zählte das Programm die Impulse. Nach Erreichen einer bestimmten Markierung wurde die Zählung gestoppt.
4. Variationen: Der Test wurde mit verschiedenen Wassermengen und Wassergeschwindigkeiten (Hahnöffnung) durchgeführt.

Ich habe diesen Vorgang 51 Mal wiederholt und dabei die gezählten Impulse sowie die jeweilige Wassermenge notiert. Alle Messwerte sind im Anhang 11.1 aufgeführt.

### Datenanalyse

Ein Scatterplot der Count-Werte für 5L, 8L und 10L (Abbildung 2A) zeigt eine nahezu lineare Beziehung zwischen den gezählten Impulsen und der Wassermenge. Eine Trendlinie, berechnet in Excel, liefert einen R²-Wert von 0,9991. Dies deutet darauf hin, dass die CPL-Konstante eine präzise Vorhersage der verbrauchten Wassermenge ermöglicht. Die Trendlinie wird durch die lineare Gleichung beschrieben, wobei die Steigung der CPL-Wert ist:

Eine Bean-Plot-Darstellung (Abbildung 2B) der auf 1L-normalisierten Counts (CPL) verdeutlicht die enge Verteilung der Messwerte im Bereich von ca. 1100 bis 1200 CPL, mit einem . Dies ist ein CPL-Wert, der auf einer statistischen Vereilung der Datenpunkte beruht.

*Auch den er Durchschnitt* der 51 CPL-Werte *könnte man nehmen um den Arduino Impulszählungen in Wasservolumen umrechnen zu lassen*:

### Validierung des CPL-Werts

Zur Validierung des CPL-Werts habe ich fünf zusätzliche Tests durchgeführt:

1. Der Eimer wurde für eine unbestimmte Zeit (10 – 30 Sekunden) gefüllt, während der Arduino die Impulse zählte.
2. Die exakte Wassermenge wurde anschließend mit einer Küchenwaage bestimmt (1g Wasser = 1mL).   
   Die Ergebnisse sind als grüne Punkte im Scatterplot (Abbildung 2A) eingezeichnet und in Tabelle 1 aufgeführt
3. Das gemessene Volumen wurde mit dem CPL-basierten berechneten Volumen verglichen, wobei ich die Genauigkeit der 3 CPL-Werte vergleichen möchte. Die 5 Messergebnisse sind in Tabelle 1 aufgeführt.

Table 1: Validierung des bestimmten CPL-Werts

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **CPLLin = 1148.3** | | **CPLStat = 1134.2** | | **CPLSchnitt = 1141.25** | |
| **Nr.** | **Counts** | **Gemessen (L)** | **Erwartet (L)** | **Fehler (%)** | **Erwartet (L)** | **Fehler (%)** | **Erwartet (L)** | **Fehler (%)** |
| 1 | **5098** | **4.385** | 4.440 | 1.245 | 4.495 | 2.504 | 4.467 | 1.871 |
| 2 | **5935** | **5.107** | 5.169 | 1.204 | 5.233 | 2.463 | 5.200 | 1.830 |
| 3 | **5549** | **5.289** | 4.832 | -8.634 | 4.892 | -7.498 | 4.862 | -8.069 |
| 4 | **5680** | **4.708** | 4.946 | 5.065 | 5.008 | 6.371 | 4.977 | 5.714 |
| 5 | **6522** | **5.509** | 5.680 | 3.099 | 5.750 | 4.380 | 5.715 | 3.735 |
|  |  |  |  |  |  |  |  |  |
| Fehler-Durchschnitt: | | | | **1.40** |  | **1.64** |  | **1.02** |

Obwohl alle 3 CPL-Werte gute vorhersagen trafen, war der Durchschnittswert etwas besser, den die berechneten Werte wichen im Schnitt um weniger als ±2% vom der gemessenen Wassermenge ab. Dies zeigt die die Genauigkeit der CPL-Konstante und ich werded den Wert in meinem Arduino Sketch verwenden.

|  |
| --- |
| Abbildung 2: CPL-Ermittlung.   1. Die roten Punkte sind die gemessenen Impulszahlen des HES für 5L, 8L und 10L. Die Trendlinie durch den Null-Punkt hat einen R2-Wert von 0,9991. Die grünen Punkte sind Punktprüfungen um zu schauen ob die mit dem CPL-Wert berechneten Volumenwerte die tatsächliche Wassermenge gut vorhersagt. 2. Die 51 Messwerte sind in einem Bean-Plot als Counts pro Liter dargestellt und zeigen eine enge Verteilung mit einem Median-Wert von 1134,2 CPL. |

## Die Software

Ein Arduino-Programm wird als “Sketch” bezeichnet. Der S4F-Sketch umfasst 354 Zeilen und ist auf GitHub verfügbar14. Der Code ist ausführlich kommentiert, um die einzelnen Schritte und die Funktionsweise des Programms zu verdeutlichen. Die Programmstruktur ist in Abbildung 3 dargestellt.

Das Programm ist in verschiedene sogenannte Funktionen unterteilt, wobei jede Funktion einen bestimmten logischen Ablauf umfasst. Diese Funktionen können miteinander kombiniert werden, um die gewünschte Programmlogik zu realisieren.

Wenn das S4F-Gerät gestartet wird, bootet der Arduino direkt in das Betriebsprogramm. Zunächst wird eine library eingebunden, die die Kommunikation mit dem LCD-Display ermöglicht. Anschließend werden innerhalb von Millisekunden verschiedene Variablen und Konstanten definiert, darunter die GPIO-Pins für den Taster, das LCD-Display und den Hall-Effekt-Sensor. Auch thermistorspezifischn Konstanten (A, B, C) sowie CPL werden an dieser Stelle definiert. Ein Arduino-Sketch besteht immer aus zwei Hauptfunktionen: setup() und loop().

In der setup()-Funktion wird festgelegt, wie sich die einzelnen Pins verhalten sollen. Beispielsweise wird der Pin für den Taster („ButtonPin“) sowie der für den Hall-Effekt-Sensor („HallPin“) als „INPUT“ definiert, sodass der Arduino diese Pins kontinuierlich überwachen kann. Der HallPin wird zusätzlich mit einem attachInterrupt()[[17]](#endnote-17) versehen, wodurch der Arduino unabhängig vom restlichen Programm Spannungsänderungen an diesem Pin registriert. Jede Spannungsänderung – also jeder Impuls des Hall-Effekt-Sensors – ruft die Funktion CountHallPulses() auf, die eine Variable namens HallCount um 1 erhöht und damit die Impulse zählt.

Zusätzlich werden in der setup()-Funktion alle relevanten Variablen auf ihren Anfangswert gesetzt. Danach wird eine Willkommensnachricht auf dem LCD-Display angezeigt, die den Benutzer auffordert, den Taster zu drücken, um die Messungen zu starten. Das Programm wartet anschließend in einer Schleife (while-Loop), bis der Taster gedrückt wird und sich damit die Spannung am ButtonPin sich ändert.

Nach Betätigung des Tasters wird die Startzeit (TimeStart) festgelegt, und das Programm wechselt zur Hauptfunktion eines jeden Arduino-Sketches: der loop()-Funktion. Diese wird kontinuierlich wiederholt und bildet das Herzstück des Programms.

Zu Beginn der loop()-Funktion überprüft das Programm, ob die Sensoren korrekt angeschlossen sind. Falls der Thermistor-Schaltkreis offen ist (keine Sensoren angeschlossen), wird ein unendlich großer Widerstand gemessen, was zu einem Temperaturwert von -273,15°C führt. In diesem Fall zeigt das Display eine Fehlermeldung an und fordert den Benutzer auf, die Sensorverbindungen zu prüfen.

Wenn die Sensoren korrekt arbeiten, beginnt das Programm mit den Messungen. Alle sechs Sekunden wird die Funktion MeasureFlow() aufgerufen, um die aktuelle Durchflussrate zu berechnen, während die Wassertemperatur alle zwei Sekunden mithilfe der Funktion MeasureTemp() gemessen wird. Beide Werte werden im Zwei-Sekunden-Takt durch die Funktion ReportInfo() auf dem Display aktualisiert.

Die Funktion MeasureFlow() berechnet die aktuelle Durchflussrate (FlowRate) in Litern pro Minute, indem sie die Anzahl der gezählten Hall-Impulse (HallCount) durch die Konstante CPL teilt und das Ergebiss mit 10 multipliziert da die Funktion alle 6 Sekunden (10 mal pro Minute) aufgerufen wird. Zusätzlich wird der Durchschnitt der Durchflussrate berechnet. Abschließend setzt die Funktion HallCount auf 0 zurück, um die nächste Messung vorzubereiten.

Die Funktion MeasureTemp() berechnet die aktuelle Wassertemperatur (Temp\_C), wie in Abschnitt 7.2 beschrieben. Auch hier wird der Durchschnittswert der Temperatur berechnet.

|  |
| --- |
| Abbildung 3: Software-Logik  Der Ablauf des Betriebprogramms ist schematisch dargestellt. |

Wenn der Taster nach der Dusche erneut gedrückt wird, beendet das Programm den Messzyklus und definiert die Stopp-Zeit. Anschließend werden die ermittelten Werte nacheinander auf dem Display angezeigt:

* Duschzeit:
* Wasserverbrauch:
* Schnittemperatur:
* Energieverbrauch:
* Kosten:

Die letzten beiden Werte (Energieverbrauch und Kosten) dienen lediglich der Orientierung, da sie von mehreren Faktoren beeinflusst werden und daher weniger präzise sind. Der Energieverbrauch hängt insbesondere von der spezifischen Wärmekapazität des Wassers (c\_Water), der Temperatur des kalten Leitungswassers (TempBase) sowie von der Effizienz des Warmwasserbereitungs- und Speichersystems ab. Da die Effizienz solcher Systeme typischerweise zwischen 60 % und 80 % liegt, habe ich einen realistischen Wert von 70 % verwendet und den Energiewert entsprechend nach oben angepasst, indem ich durch 0,7 teile.

Die Berechnung der Kosten ist ebenfalls ungenau, da sie von den lokalen Energie- und Wasserpreisen abhängt. Der Energiepreis kann zusätzlich stark variieren, je nachdem, ob das Wasser z.B. mit Strom, Gas oder Öl erhitzt wird. Für die Berechnung habe ich Werte basierend auf den Preisen im Landkreis Barnim verwendet[[18]](#endnote-18), die realistisch scheinen: 0,28 Euro/kWh Energie und 5,86 Euro/m3 Wasser (inkl. Abwasser).

## 3D-Design und Druck

Um den Temperatursensor zu verbauen habe ich ein Gehäuse für den Thermistor entworfen und gedruckt (Abbildung 4A). Da Wasserarmaturen ½-Zoll Gewinde haben und der Flowsensor auf beiden Seiten ein ½-Zoll Aussengewinde hat habe ich mich entschlossen das Thermistorgehäuse mit ½-Zoll Innengewinden zu drucken. Wenn ich Fluss- und Temperatursensor miteinander verschraube, habe wir an einem Ende ein Innen- und am anderen Ende ein Außengewinde und deshalb kann der Adapter einfach zwischen Armatur und Schlauch gesetzt werden. Der Thermistor wird mittig in das Gehäuse eingeschoben und soll dann mit Silikon versiegelt werden. Im inneren des Gehäuses sind 2 Ablagen für Gummidichtungen (Abbildung 4B).

Das Gehäuse für die Elektronik habe ich genau auf die Maße des Arduinos, der Platine, und des LCD Displays angepasst (Abbildung 4C, D). Der Arduino wird so eingelegt, dass der USB-B Port zugänglich ist. Dann wird das verlötete Elektronik Shield auf den Arduino gesteckt und mit dem Gehäuse verschraubt. Dann wird das LCD-Display aufgesteckt und mit dem Shield verschraubt. Eine 9V Batterie wird in die vorhandene Aussparung gelegt und mit dem Shield verkabelt. Die Sensoren werden über einen RJ45 Keystone Stecker angeschlossen, der in das Gehäuse eingeklickt wird und mit den Sensor Ports am Schield verbunden wird. In den Deckel des Gehäuses wird ein Schalter eingeschraubt, mit dem die Batterie vom Stromkreis getrennt werden kann. Ein Taster wird auch in den Deckel geschraubt und mit dem Knopf-Port am Schield verbunden. Dann muss nur noch der Deckel aufgeschraubt werden, so dass das LCD Display durch das vorhandene Loch ragt und die S4F ist bereit. Am Gehäuse ist auch eine Halterung, so dass die S4F an der Dusche außen aufgehangen werden kann und man unter der Dusche die Daten leicht ablesen kann.

|  |
| --- |
| Abbildung 4: 3D-Modelle und Druckergebnisse  Zu sehen sind die 3D-Modelle für (A) die Thermistorhalterung und (C) das Gehäuse für die Elektronik. B und D zeigen die gedruckten und zusammengebauten Teile. |

# Ergebnisdiskussion, Fazit und Ausblick

In meinem Projekt „S4F“ habe ich einen Adapter entworfen der auf alle gängigen Wasserarmaturen mit ½-Zoll Gewinde passen sollte. Ich habe in diesem Adapter Sensoren verbaut um Wassermenge und Wassertemperatur zu bestimmen. Die Sensoren habe ich mit einem Arduino Microcontroller über eine selbst-entwickelte Elektronik verbunden. Arduino und die Elektronikplatine sowie ein kleines LCD-Display habe ich in einem 3D-gedrucktem Gehäuse verbaut.

Für den Arduino habe ich ein Betriebsprogramm geschrieben: Wenn der Nutzer das Programm startet misst der Arduino kontinuierlich Wassergeschwindigkeit und Wassertemperatur und zeigt die Werte auf dem LCD-Display. Wenn der Nutzer das Programm durch erneuten Knopfdruck beendet, berechnet und zeigt der Arduino 5 Werte: (i) Duschzeit, (ii) Gesamtwasserverbrauch, (iii) Durchschnittstemperatur, (iv) Energieverbrauch und (v) ungefähre Kosten.

## Erste Eindrücke:

Wir haben das erste S4F Gerät bei uns zu Hause in der Dusche installiert und konnten erste Eindrücke sammeln. Ich und meine ganze Familie waren von den Daten überrascht und es wurde zu einem interessanten Gesprächsthema. Besonders interessant war es natürlich wer am längsten und wer am heißesten duscht. Die Wassermenge hängt größtenteils von der Länge der Dusche ab, da alle den Wasserhahn immer voll aufmachen. Im Schnitt dauert eine Dusche bei uns xxx min, verbraucht xxx L Wasser und xxx kWh Energie bei einer Durchschnittstemperatur von xxx °C und kostet deshalb ca. xxx Euro. Die teuerste Dusche die wir bislang gemessen haben war xxx min lang und kostete ca. xxx Euro. Das war etwas erschreckend, denn wenn man pro Dusche 1 Euro annimmt, dann kann es sein dass sich bei unserer 6-köpfigen Familie die Jahreskosten 1 Euro • 365 Tage/Jahr • 6 Duschen/Tag = 2190€ betragen. Deshalb haben wir uns vorgenommen in Zukunft kürzer und weniger zu duschen.

Diesen Entschluss haben wir gefasst, weil wir jetzt die Informationen über unseren Verbrauch und dessen Kosten haben. Deshalb glaube ich das ein Gerät wie S4F wirklich potential hat das Verhalten von Nutzern zu verändern. Uns hat das Gerät ca. xxx Euro an Material gekostet und selbst dass könnte innerhalb von ein paar Monaten eingespaart werden, wenn wir unser Duschverhalten ändern.

Ich kann mir also vorstellen, dass eine Firma ein S4F-Adaptermodul vertreibt. Mit professioneller Entwicklung könnte man die gesamte Elektronik schrumpfen, so dass der Adapter nur ein wenig grösser als der Durchflusssensor selbst wäre.

## Zukunft

Es bestehen einige Verbesserungsmöglichkeiten um S4F noch nützlicher zu machen. Wenn ich S4F für Jugend Forscht weiterentwickle, würde ich als aller erstes daran arbeiten das Daten nach der Dusche per E-Mail an den Nutzer gesendet werden. Zwar weiß ich noch nicht genau wie schwer das hinsichtlich der Programmierung wird, ich weiß aber schon das es technisch mit dem Microcontroller den ich verwendet habe nicht möglich ist. Der Arduino Uno G3 besitzt nämlich kein WiFi oder andere Kommunikationsmöglichkeiten. Dafür könnte ich evtl. entweder ein Arduino Uno G4 mit WiFi oder einen ESP32 Microcontroller mit WiFi und Bluetooth verwenden.

Sollte mir das gelingen, würde ich das S4F Gerät auch dahingehend verbessern das man vor jeder Dusche einen von mehreren Nutzer auswählt, sodass die Daten nach Person aufgeschlüsselt und versendet werden. Es wäre auch interessant die Daten über geraume Zeit zu speichern, so dass Zeitprofile erstellt werden können.

Wifi-Nutzung bedeutet aber natürlich auch einen höheren Energieverbrauch und es wäre interessant die Spannungsquelle zu verbessern. Derzeit benutze ich eine 9V Batterie die im Arduino über einen Spannungsregulator auf 5V herunter reguliert wird, dabei geht aber wohl viel Energie als Wärme verloren. Besser wäre es vielleicht einen wiederaufladbaren Akku zu benutzen (was auch besser für die Umwelt ist) der über einen effizienten Boost-Converter den Arduino mit 5V versorgt.

## Ein sozialwissenschaftliches Experiment

Ich kann mir vorstellen ein verbessertes S4F Gerät für jeweils ein paar Wochen einigen Familien von Klassenkameraden zur Verfügung zu stellen, die dann S4F in ihren Duschen verwenden. Ich würde die Einwilligung einholen, dass die Daten über Wifi auch an mich versendet werden. Ich würde den Familien die ausgewerteten Daten natürlich zur Verfügung stellen. Ich würde aber auch die anonymisierten Daten für mein nächstes Jugend Forscht Projekt auswerten wollen, um herauszufinden in wie weit sich das Duschverhalten anderer Familien mit Hilfe der S4F verändert.

# Quellen- und Literaturverzeichnis

# Unterstützungsleistungen

Ich möchte mich besonders bei Fr. Hänsel für ihre Betreuung meiner Jugend Forscht Arbeit bedanken. Ich möchte mich auch bei Frau Gust bedanken, die meine Facharbeit betreut hat und die mir erlaubt hat die Facharbeit mit dem Jugend Forscht Projekt zu kombinieren. Ich will mich bei meinem Vater für seine Hilfe beim Löten und Programmieren bedanken. Schließlich möchte ich mich noch beim BIMSB (Berlin Institute for Medical Systems Biology) bedanken, weil ich dort den 3D-Drucker benutzen durfte.

# Anhang

## Teileliste

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nr. | Volumen (L) | Wasserhahn | Füllzeit (sec) | HES Counts | CPL |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## Messwerte zur CPL-Wert bestimmung

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nr. | Volumen (L) | Wasserhahn | Füllzeit (sec) | HES Counts | CPL |
| 1 | 5 | ganz offen | 25 | 5716 | 1143,20 |
| 2 | 5 | ganz offen | 25 | 5726 | 1145,20 |
| 3 | 5 | ganz offen | 24 | 5636 | 1127,20 |
| 4 | 5 | ganz offen | 24 | 5519 | 1103,80 |
| 5 | 5 | ganz offen | 29 | 5605 | 1121,00 |
| 6 | 5 | ganz offen | 23 | 5464 | 1092,80 |
| 7 | 5 | ganz offen | 24 | 5540 | 1108,00 |
| 8 | 5 | ganz offen | 24 | 5513 | 1102,60 |
| 9 | 5 | ganz offen | 25 | 5779 | 1155,80 |
| 10 | 5 | tlw. offen | 45 | 5673 | 1134,60 |
| 11 | 5 | tlw. offen | 36 | 5544 | 1108,80 |
| 12 | 5 | tlw. offen | 43 | 5610 | 1122,00 |
| 13 | 5 | tlw. offen | 45 | 5653 | 1130,60 |
| 14 | 5 | tlw. offen | 47 | 5553 | 1110,60 |
| 15 | 5 | tlw. offen | 47 | 5526 | 1105,20 |
| 16 | 5 | tlw. offen | 64 | 5487 | 1097,40 |
| 17 | 5 | tlw. offen | 42 | 5640 | 1128,00 |
| 18 | 5 | tlw. offen | 44 | 5839 | 1167,80 |
| 19 | 5 | tlw. offen | 50 | 5635 | 1127,00 |
| 20 | 5 | tlw. offen | 56 | 5477 | 1095,40 |
| 21 | 8 | ganz offen | 40 | 9366 | 1170,75 |
| 22 | 8 | ganz offen | 39 | 9234 | 1154,25 |
| 23 | 8 | ganz offen | 40 | 9263 | 1157,88 |
| 24 | 8 | ganz offen | 42 | 9540 | 1192,50 |
| 25 | 8 | ganz offen | 43 | 9664 | 1208,00 |
| 26 | 8 | ganz offen | 55 | 9631 | 1203,88 |
| 27 | 8 | ganz offen | 61 | 9527 | 1190,88 |
| 28 | 8 | tlw. offen | 79 | 9340 | 1167,50 |
| 29 | 8 | tlw. offen | 73 | 9163 | 1145,38 |
| 30 | 8 | tlw. offen | 88 | 8882 | 1110,25 |
| 31 | 8 | tlw. offen | 89 | 8859 | 1107,38 |
| 32 | 8 | tlw. offen | 104 | 8922 | 1115,25 |
| 33 | 8 | tlw. offen | 60 | 9480 | 1185,00 |
| 34 | 8 | tlw. offen | 102 | 8985 | 1123,13 |
| 35 | 8 | tlw. offen | 79 | 8990 | 1123,75 |
| 36 | 10 | ganz offen | 52 | 11988 | 1198,80 |
| 37 | 10 | ganz offen | 48 | 11169 | 1116,90 |
| 38 | 10 | ganz offen | 52 | 11824 | 1182,40 |
| 39 | 10 | ganz offen | 51 | 11670 | 1167,00 |
| 40 | 10 | ganz offen | 51 | 11781 | 1178,10 |
| 41 | 10 | ganz offen | 50 | 11697 | 1169,70 |
| 42 | 10 | ganz offen | 47 | 11207 | 1120,70 |
| 43 | 10 | ganz offen | 48 | 11871 | 1187,10 |
| 44 | 10 | tlw. offen | 94 | 10958 | 1095,80 |
| 45 | 10 | tlw. offen | 72 | 11440 | 1144,00 |
| 46 | 10 | tlw. offen | 105 | 11091 | 1109,10 |
| 47 | 10 | tlw. offen | 84 | 11258 | 1125,80 |
| 48 | 10 | tlw. offen | 99 | 11342 | 1134,20 |
| 49 | 10 | tlw. offen | 101 | 11490 | 1149,00 |
| 50 | 10 | tlw. offen | 98 | 11639 | 1163,90 |
| 51 | 10 | tlw. offen | 102 | 11785 | 1178,50 |

1. Elegoo.com (2025): ELEGOO UNO R3 Project The Most Complete Starter Kit Tutorial, [online] <https://eu.elegoo.com/blogs/arduino-projects/elegoo-uno-r3-project-the-most-complete-starter-kit-tutorial> [abgerufen am 12.01.2025]. [↑](#endnote-ref-1)
2. Wikipedia.org (2025): Steinhart-Hart Gleichung, [online] <https://de.wikipedia.org/wiki/Steinhart-Hart-Gleichung> [abgerufen am 12.01.2025]. [↑](#endnote-ref-2)
3. Wikipedia.org (2025): Hall-Effekt, [online] <https://de.wikipedia.org/wiki/Hall-Effekt> [abgerufen am 12.01.2025]. [↑](#endnote-ref-3)
4. Arduino.cc (2025): Software | Arduino, [online] <https://www.arduino.cc/en/software> [abgerufen am 12.01.2025]. [↑](#endnote-ref-4)
5. Arduino.cc (2025): Arduino Reference, [online] <https://www.arduino.cc/reference/cs/> [abgerufen am 12.01.2025]. [↑](#endnote-ref-5)
6. Amazon.de (2025): 1-30L / min Wasserdurchflussschalter, [online] <https://www.amazon.de/gp/product/B07XDZ25SY/ref=ppx_yo_dt_b_search_asin_image?ie=UTF8&psc=1> [abgerufen am 12.01.2025]. [↑](#endnote-ref-6)
7. Conrad.de (2025): TRU COMPONENTS MJSTS-103-3950-1-600-3D, [online] <https://www.conrad.de/de/p/tru-components-mjsts-103-3950-1-600-3d-temperatursensor-30-bis-105-c-10-k-3950-k-1570951.html> [abgerufen am 12.01.2025]. [↑](#endnote-ref-7)
8. Arduino.cc (2025): Arduino Starter Kit Multi-language, [online] <https://store.arduino.cc/products/arduino-starter-kit-multi-language?srsltid=AfmBOoo0SrVGEwepGFm_RW5cQv4Eu3bNgyMQxEinp2FHh5t-q_M4DR6r> [abgerufen am 12.01.2025]. [↑](#endnote-ref-8)
9. Elegoo.com (2025): ELEGOO 37 in 1 Sensor Modules Kit Tutorial, [online] <https://www.elegoo.com/blogs/arduino-projects/elegoo-37-in-1-sensor-modules-kit-tutorial?srsltid=AfmBOorcQefaeVhuj9hzNdY7BtcV7Ft-uDLdNtTredJ6e7MYfBe4t1id> [abgerufen am 12.01.2025]. [↑](#endnote-ref-9)
10. Autodesk.com (2025): Autodesk Eagle, [online] <https://www.autodesk.com/products/eagle/> [abgerufen am 12.01.2025]. [↑](#endnote-ref-10)
11. Autodesk.com (2025): Autodesk Fusion, [online] <https://www.autodesk.com/products/fusion-360> [abgerufen am 12.01.2025]. [↑](#endnote-ref-11)
12. Keyence.de (2025): AGILISTA-3200W, [online] <https://www.keyence.de/products/3d-printers/3d-printers/agilista-3100/models/agilista-3200w/> [abgerufen am 12.01.2025]. [↑](#endnote-ref-12)
13. Arduino.cc (2025): Liquid Crystal Displays (LCD) with Arduino, [online] https://docs.arduino.cc/learn/electronics/lcd-displays/ [abgerufen am 13.01.2025].

    https://docs.arduino.cc/learn/electronics/lcd-displays/ [↑](#endnote-ref-13)
14. Github (2025): S4F, [online] <https://github.com/rzinzen/S4F> [abgerufen am 12.01.2025]. [↑](#endnote-ref-14)
15. Fluke (2025): What is Ohm’s Law? (Electrical, Fundamentals), [online] <https://www.fluke.com/en/learn/blog/electrical/what-is-ohms-law> [abgerufen am 13.01.2025]. [↑](#endnote-ref-15)
16. SRS (2025): Thermistor Calculator v1.1, [online] <https://www.thinksrs.com/downloads/programs/therm%20calc/ntccalibrator/ntccalculator.html> [abgerufen am 12.01.2025]. [↑](#endnote-ref-16)
17. Arduino.cc (2025): attachInterrupt(), [online] <https://www.arduino.cc/reference/cs/language/functions/external-interrupts/attachinterrupt/> [abgerufen am 12.01.2025]. [↑](#endnote-ref-17)
18. Stadtwerke Bernau (2025): Stadtwerke Neukunden – NK-Portal, [online] <https://kundenportal.stadtwerke-bernau.de/Neukunden/Neukunden/Tarifuebersicht> [abgerufen am 12.01.2025]. [↑](#endnote-ref-18)