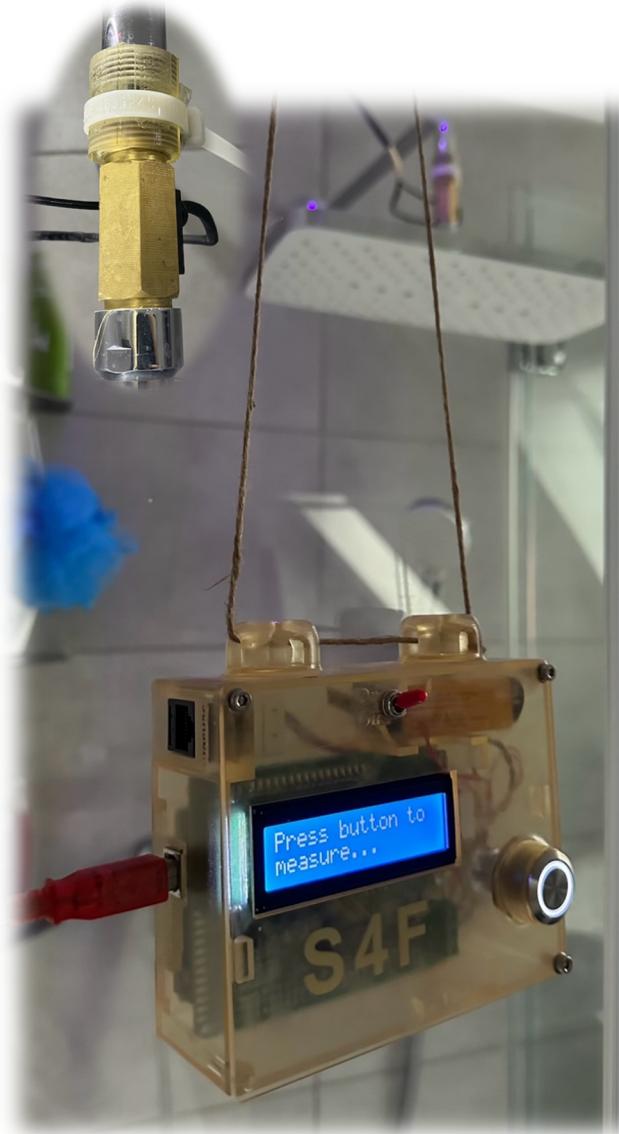


S 4 F

(Showers4Future)

Ein **Jugend Forscht** Projekt von
Oliver R. Zinzen (15)



Fachgebiet: **Physik**

Jahr: 2024/25

Bundesland: Brandenburg
(West)

Barnim Gymnasium Bernau

Klasse 9M

Betreuung:

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



Projektüberblick

In meinem Projekt geht es darum, Wasser und Energie bewusster und sparsamer zu nutzen.

Ich habe einen Adapter für die Dusche entwickelt, mit dem die verbrauchte Wassermenge und -temperatur gemessen werden können. Ziel ist es, Verbrauchsinformationen leicht zugänglich zu machen, da ich überzeugt bin, dass solche Daten unser Verhalten positiv beeinflussen können, um Ressourcen und Kosten zu sparen.

Auf die Idee kam ich aus zwei Gründen: Zum einen bereiten uns die steigenden Energiepreise Sorgen, und zum anderen bedroht der Klimawandel unsere Zukunft. Ich glaube, dass intelligente technologische Ansätze uns dabei unterstützen können, umweltbewusster zu handeln.

Der von mir entwickelte Adapter wird beispielsweise an einer Duscharmatur installiert. Mithilfe eines Arduino-Mikrocontrollers werden die Sensoren für Temperatur und Wassergeschwindigkeit ausgelesen und die Daten dem Nutzer auf einem LCD-Display angezeigt. Ich habe den Adapter erfolgreich gebaut und bei uns zu Hause getestet, um erste Erfahrungen zu sammeln. Ich nenne ihn S4F („Showers for Future“).

Der Adapter hat gut funktioniert, aber es gibt noch viele Möglichkeiten zur Verbesserung. Zum Beispiel könnte er kleiner und kompakter gebaut werden, und es wäre sinnvoll, die erfassten Daten kabellos zu übertragen. Gerne würde ich S4F auch bei anderen Familien testen, um herauszufinden, ob die bereitgestellten Informationen nicht nur bei uns, sondern auch bei anderen Nutzern ein bewussteres und sparsameres Verhalten im Umgang mit Wasser und Energie fördern.

(Für Bild: siehe Titelseite)

Inhaltsverzeichnis

PROJEKTÜBERBLICK	I
INHALTSVERZEICHNIS	II
1 FACHLICHE KURZFASSUNG	1
2 MOTIVATION UND FRAGESTELLUNG	1
3 HINTERGRUND UND THEORETISCHE GRUNDLAGEN	1
3.1 TEMPERATURMESSUNG	2
3.2 WASSERVERBRAUCH	2
3.3 MICROCONTROLLER	2
4 VORGEHENSWEISE, MATERIALIEN UND METHODEN	3
4.1 ELEKTRONIK UND PROTOTYP	3
4.2 PROGRAMMIEREN	4
4.3 LÖTEN	4
4.4 3D-DESIGN & 3D-DRUCK	4
4.5 IMPLEMENTIERUNG ZU HAUSE	4
5 ERGEBNISSE	5
5.1 DER PROTOTYP	5
5.2 IMPLEMENTIERUNG DES THERMISTORS ZUR TEMPERATURMESSUNG	7
5.2.1 Berechnung des Thermistorwiderstands	7
5.2.2 Temperaturberechnung	7
5.3 ERMITTlung DES CPL-WERTS	8
5.3.1 Versuchsaufbau	8
5.3.2 Datenanalyse	9
5.3.3 Validierung des CPL-Werts	9
5.4 DIE SOFTWARE	10
5.5 3D-DESIGN UND DRUCK	12
6 ERGEBNISDISKUSSION, FAZIT UND AUSBLICK	13
6.1 ERSTE EINDRÜCKE	13
6.2 ZUKUNFT	14
6.3 EIN SOZIALWISSENSCHAFTLICHES EXPERIMENT	15
7 QUELEN- UND LITERATURVERZEICHNIS	16
8 UNTERSTÜTZUNGSLEISTUNGEN	17
9 ANHANG	18
9.1 TEILELISTE	18
9.2 MESSWERTE ZUR CPL-WERT BESTIMMUNG	19

1 Fachliche Kurzfassung

In meinem Projekt „S4F“ habe ich ein Gerät entwickelt, das Nutzer über ihren Wasser- und Energieverbrauch informiert.

Ein 3D-gedruckter Adapter, der zwischen Wasserarmatur (z.B. Duscharmatur) und Schlauch geschraubt wird, enthält Fluss- und Temperatursensoren und ist mit einem Arduino-Mikrocontroller verbunden. Die von mir entwickelte Software misst die Wassergeschwindigkeit und Temperatur und zeigt die ermittelten Werte auf einem LCD-Display an. Nach der Dusche berechnet das Gerät Verbrauchsdaten wie Duschzeit, Wassermenge, Durchschnittstemperatur, Energieverbrauch und Kosten und informiert den Nutzer darüber.

Das Ziel des Projekts ist es, den Nutzern Daten zur Verfügung zu stellen, die ein bewussteres und umweltfreundlicheres Verhalten ermöglichen.

2 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 hilft, unseren Wasser- und Energieverbrauch besser zu verstehen und diese wertvollen Ressourcen 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.

3 Hintergrund und theoretische Grundlagen

Die Idee für S4F kam mir im Winter 2022. Meine Familie war besorgt über die steigenden Energiekosten nach Russlands Invasion der Ukraine. Ich lebe mit meinen Eltern und drei Brüdern zusammen und 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 zu reduzieren.

3.1 Temperaturmessung

Um die Temperatur zu messen, dachte ich sofort an einen Thermistor. Dieses elektronische Bauteil war mir zuvor in einem Arduino-Starter-Kit¹ begegnet. Ein Thermistor ist ein elektrischer Widerstand mit der besonderen Eigenschaft, dass sein Widerstandswert von der Temperatur abhängt.

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 berechnen². 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-Lernsets 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.

3.2 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 werden³.

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.

3.3 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 durch 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 Energieeffizienz sehr wichtig.

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 geschrieben und getestet werden. Zudem bietet die Arduino-Community eine ausgezeichnete Unterstützung: Es gibt zahlreiche Anleitungen, Foren und Codebeispiele, die den Einstieg erleichtern⁵. Darüber hinaus ist es relativ einfach, ein Arduino-Board mit einem LCD-Display zu verbinden, um dem Nutzer Anweisungen anzuzeigen oder Daten zu präsentieren.

4 Vorgehensweise, Materialien und Methoden

4.1 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 Einsticken von Jumper-Kabeln zu verbinden. Eine Liste der verwendeten Komponenten befindet sich im Anhang (9.1). Die wichtigsten Komponenten sind:

- **Arduino Uno R3 Dev. Kit:** Ein Arduino-Entwicklerboard als Steuereinheit.
- **Durchflusssensor⁶:** Kann mithilfe des Hall-Effekts Durchflussmengen zwischen 1 und 30 L/min messen.
- **Thermistor:** Ein NTC-Temperatursensor (TRU Components, Modell MJSTS-103-3950-1-600-3D⁷) mit einem $10\text{k}\Omega$ Referenzwiderstand bei 25°C .
- **Steckplatine und Elektronikkomponenten:** Die Steckplatine sowie verschiedene Bauteile (Widerstände, Taster, Potentiometer, Jumper-Kabel, LCD-Display) stammen aus mehreren Elektronik-Lernsets^{1,8,9}.

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

4.2 Programmieren

Für die Programmierung habe ich die Software Arduino IDE verwendet. Mit dieser Entwicklungsumgebung kann man C++ Code schreiben, auf Syntaxfehler prüfen und anschließend auf ein 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.

4.3 Löten

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

Zunächst habe ich mit der CAD-Software Autodesk Eagle¹⁰ (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 verlötet, um eine robuste und dauerhafte Verbindung zu gewährleisten.

4.4 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 360¹¹ (kostenlos für Schüler*innen) entworfen.

Die 3D-Modelle habe ich anschließend als STL-Dateien exportiert und mit einem 3D-Drucker (Keyence Agilsta 3200W¹²) gedruckt. Der Agilsta ist ein professioneller 3D-Drucker, der anstelle des herkömmlichen PLA-Filaments einen fotoaktivierbaren Druck-Harz verwendet. Dadurch kann der Agilsta eine höhere Auflösung sowie bessere Materialfestigkeit und Hitzebeständigkeit erreichen.

4.5 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.

5 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.

5.1 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.
- **16x2 LCD-Display Modul:** Das Display kann 16 Zeichen in 2 Reihen darstellen. Wie im Arduino Wiki beschrieben¹³, ist das LCD-Display mit 5V+ und Gnd verbunden. Der V0-Pin des Displays ist mit einem 10kΩ 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, offen) und 5V (HIGH, geschlossen) erkennt, womit der Arduino 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 (R_5) und der

temperaturabhängige Thermistor in Reihe zwischen 5V+ und Gnd geschaltet sind. Da der Thermistor einen Referenzwert von $10\text{k}\Omega$ bei 25°C hat, habe ich einen $10\text{k}\Omega$ 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 5.2 beschrieben auslesen kann.

Nachdem der funktionierende Schaltplan stand (Abbildung 1A, B), um den Taster, den Thermistor und den Hall-Effekt-Sensor auszulesen, habe ich mir einen Lötplan erstellt (Abbildung 1C). Anschließend habe ich die Schaltung auf einer Lochrasterplatine umgesetzt (Abbildung 1D). Da ich eine Platine mit einem

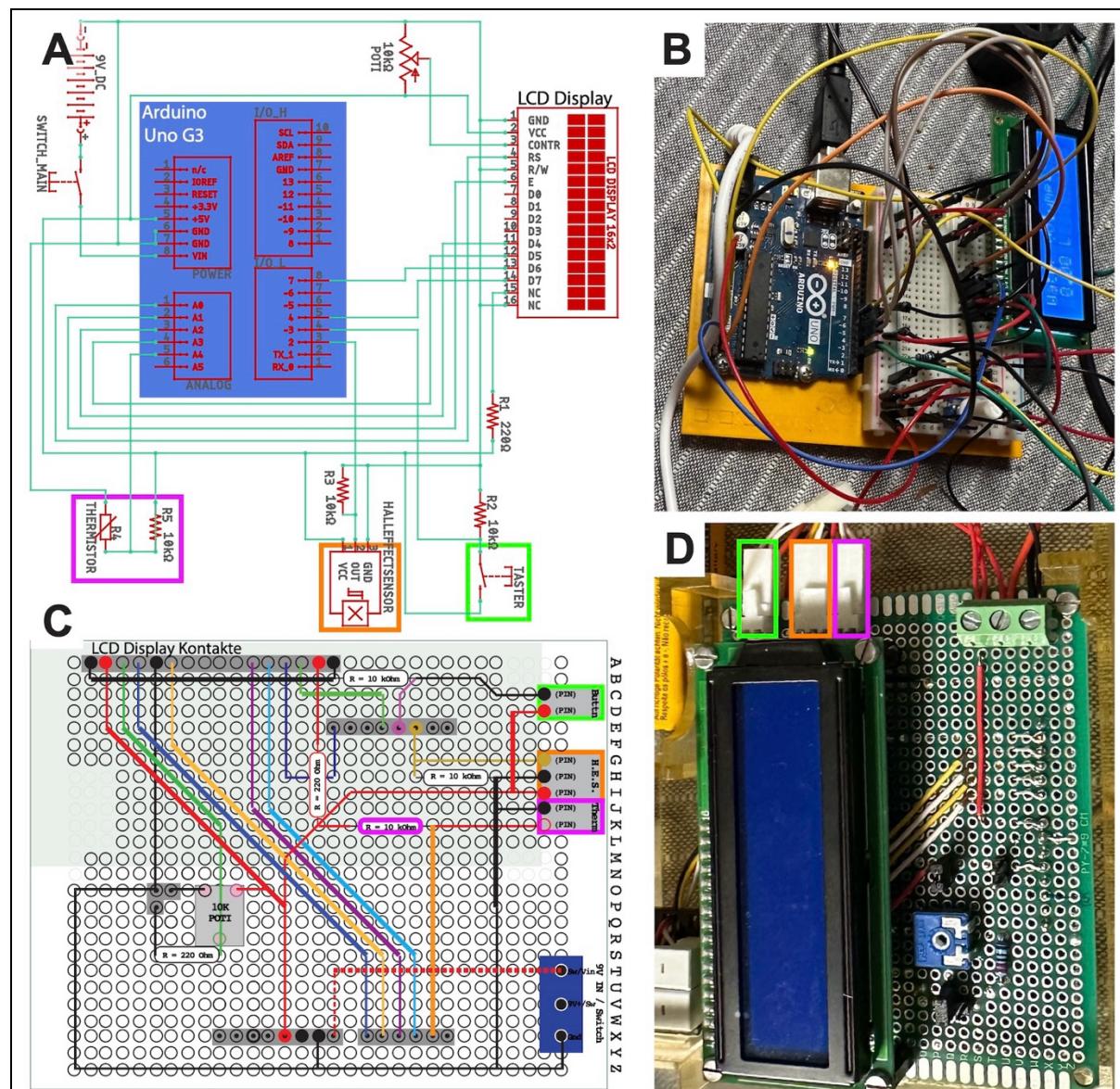


Abbildung 1: Prototyp Schaltpanelen Shield Entwicklung

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**). Einige Komponenten sind umrahmt (Thermistor in rosa, HES in orange, Input Taster in hellgrün). Detaillierte Pläne stehen zum Download auf GitHub bereit^{[14](#)}.

Lochrastermaß von 2,54 mm verwendete, konnte ich die Lötstifteleisten so anordnen, dass die Platine direkt auf den Arduino aufgesteckt werden konnte, um eine stabile Verbindung zu gewährleisten. Zusätzlich habe ich eine Kontaktleiste eingelötet, sodass das LCD-Display mit seinen 16 Stiften direkt auf die Platine aufgesetzt werden kann. Dies ermöglichte es, das Display und den Arduino bei Bedarf wieder zu trennen, falls ich Änderungen an der Platine vornehmen muss.

5.2 Implementierung des Thermistors zur Temperaturmessung

5.2.1 Berechnung des Thermistorwiderstands

Da der Thermistorwiderstand R_{Ther} mit einem Arduino nicht direkt ausgelesen werden kann, werde ich ihn mit Hilfe des Ohm'schen Gesetzes¹⁵ anhand der gemessenen Spannung berechnen ($R_{Ther} = \frac{U_{Ther}}{I}$). Die Spannung über den Thermistor U_{Ther} 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 ($U_{In} = 5V$) in einen Wert zwischen 0 – 1023 (10-Bit-Auflösung) um.

Ich benutze einen Spannungsteiler, der aus zwei Widerständen in Reihe besteht: R_1 (ein konstanter Widerstand von $10k\Omega$) und R_{Ther} (der Widerstand des Thermistors).

Der Gesamtwiderstand des Spannungsteilers ist: $R_{Ges} = R_1 + R_{Ther}$

Nach dem Ohm'schen Gesetz¹⁵ gilt: $U_{In} = R_{Ges} \cdot I_{Ges} = (R_1 + R_{Ther}) \cdot I$

Daraus folgt:

$$I_{Ges} = \frac{U_{In}}{(R_1 + R_{Ther})}$$

Die 5V Eingangsspannung teilt sich über Widerstände in Reihe auf: $U_{In} = U_1 + U_{Ther}$

Die Spannung über den Thermistor ist nach dem Ohm'schen Gesetz definiert als:

$$U_{Ther} = R_{Ther} \cdot I_{Ther} \quad \Leftrightarrow \quad I_{Ther} = \frac{U_{Ther}}{R_{Ther}}$$

Da die Stromstärke überall im Stromkreis gleich ist gilt $I_{Ges} = I_{Ther}$ und der Widerstand des Thermistors R_{Ther} kann berechnet werden:

$$\frac{U_{Ther}}{R_{Ther}} = \frac{U_{In}}{(R_1 + R_{Ther})}$$

Nach Umstellung: $R_{Ther} = \frac{R_1}{\left(\frac{U_{In}}{U_{Ther}} - 1\right)}$

Mit $R_1 = 10k\Omega$, $U_{In} = 5V$, und dem Wert für U_{Ther} , den der Arduino über den ADC ausliest kann der Thermistorwiderstand R_{Ther} berechnet werden.

5.2.2 Temperaturberechnung

Mit Hilfe der Steinhart-Hart-Gleichung lässt sich die Temperatur T in Kelvin aus dem Widerstand R_{Ther} berechnen:

$$T_{Kelvin} = \frac{1}{(A + B \cdot \ln(R_{Ther}) + C \cdot (\ln(R_{Ther}))^3)}$$

Die Werte A , B , und C Thermistor-spezifische Konstanten, die ich mit einem Online-Thermistor Rechner¹⁶ und den Widerstandswerten aus dem Thermistor Datenblatt⁷ ermittelt habe:

$$A = 1,105646556 \cdot 10^{-3} \quad B = 2,369707092 \cdot 10^{-4} \quad C = 8,420273017 \cdot 10^{-8}$$

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

$$T_{Celsius} = T_{Kelvin} - 273,15$$

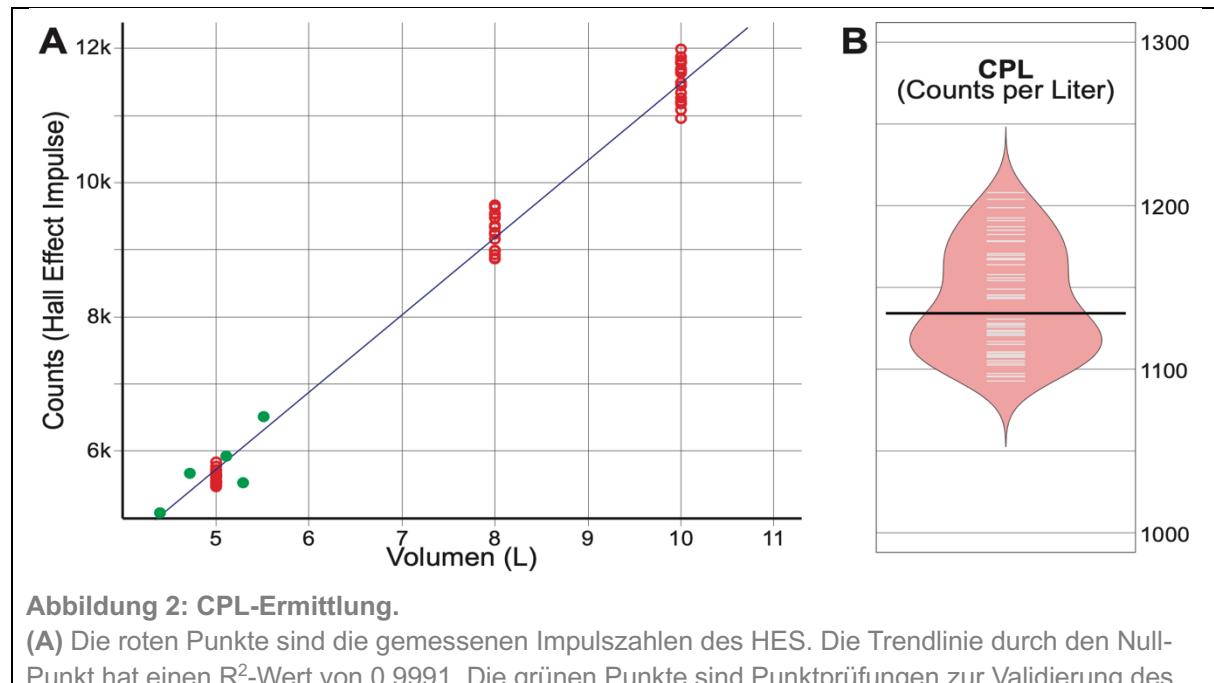
5.3 Ermittlung des CPL-Werts

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.

5.3.1 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 9.1 aufgeführt.

5.3.2 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 $y = 1148,3 \cdot x$ beschrieben, wobei die Steigung der CPL-Wert ist:

$$CPL_{lin} = 1148,3$$

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 *Median* = 1134,2 CPL. Dies ist ein CPL-Wert, der auf einer statistischen Vereilung der Datenpunkte beruht.

$$CPL_{stat} = 1134,2$$

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

$$CPL_{Schnitt} = 1141,25$$

5.3.3 Validierung des CPL-Werts

Zur Validierung des CPL-Werts habe ich Test durchgeführt. Ich habe den Eimer mehrmals für eine unbestimmte Zeit (10 – 30 Sekunden) gefüllt während der Arduino die Impulse zählte. Die exakte Wassermenge habe ich anschließend mit einer Küchenwaage bestimmt (1g Wasser = 1mL). Die Ergebnisse sind als grüne Punkte im Scatterplot (Abbildung 2A) eingezeichnet und Tabelle 1 aufgeführt. Das gemessene Volumen habe ich mit den für alle 3 CPL-Werte ermittelt, wobei ich die Genauigkeit der 3 CPL-Werte vergleichen wollte. Die Ergebnisse sind in Tabelle 1 aufgeführt.

Tabelle 1: Validierung des bestimmten CPL-Werts

			CPL _{Lin} = 1148.3		CPL _{Stat} = 1134.2		CPL _{Schnitt} = 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, denn die berechneten Werte wichen im Schnitt um weniger als $\pm 2\%$ von der gemessenen Wassermenge ab. Dies zeigt die die Genauigkeit der CPL-Konstante und ich werde den Wert $CPL = 1141,25$ in meinem Arduino Sketch verwenden.

5.4 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 thermistorspezifischen 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 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 angeschlossen sind. Falls der Thermistor-Schaltkreis offen ist (keine Sensoren angeschlossen), ist der Widerstand unendlich groß, was zu einem Temperaturwert von $-273,15^{\circ}\text{C}$ 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. Die Wassertemperatur wird alle zwei Sekunden mit der Funktion `MeasureTemp()` gemessen. Beide Werte werden im Zwei-Sekunden-Takt mit `ReportInfo()` auf dem Display aktualisiert.

Die Funktion `MeasureFlow()` berechnet die aktuelle Durchflussrate (`FlowRate`) in L/Minute, indem sie die Anzahl der gezählten Hall-Impulse (`HallCount`) durch die Konstante `CPL` teilt und das Ergebnis mit 10 multipliziert (da die Funktion alle 6 Sekunden 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 5.2 beschrieben. Auch hier wird der Durchschnittswert der Temperatur berechnet.

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: $\text{Duration(min)} = \text{TimeStart} - \text{TimeStop}$
- Wasserverbrauch: $\text{Volume(L)} = \text{FlowRateAvg(L/min)} \cdot \text{Duration(min)}$
- Schnitttemperatur: $\text{TempAvg}({}^{\circ}\text{C})$
- Energieverbrauch: $\text{Energy(Wh)} = \frac{\text{Volume(L)} \cdot \text{c_Water(kWh)} \cdot (\text{TempAvg}({}^{\circ}\text{C}) - \text{TempBase}({}^{\circ}\text{C}))}{1000 (\text{L} \cdot {}^{\circ}\text{C}) \cdot 0.7}$
- Kosten: $\text{Cost(Euro)} = (\text{Energy} \cdot \text{EnergyPrice}) + (\text{Volume} \cdot \text{WaterPrice})$

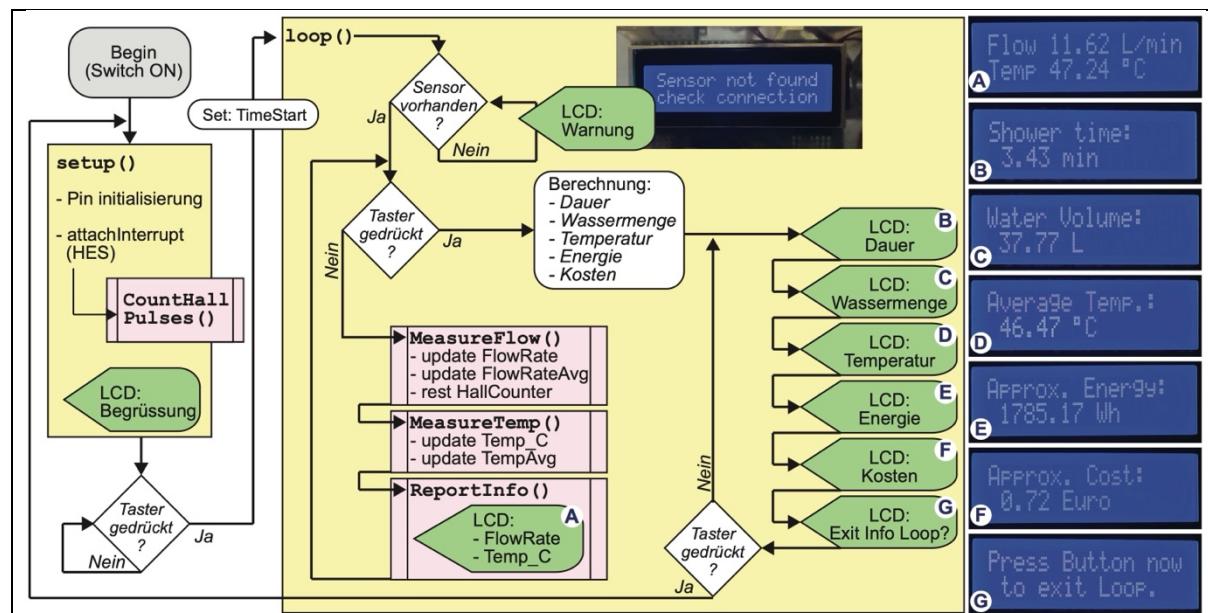


Abbildung 3: Diagramm der Software-Logik mit LCD-Display Beispielen

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 wird variieren, je nachdem, ob das Wasser z.B. mit Strom, Gas oder Öl erhitzt wird. Für die Berechnung habe ich Werte basierend auf realistischen Preisen im Landkreis Barnim verwendet¹⁸: 0,28 Euro/kWh Energie und 5,86 Euro/m³ Wasser (inkl. Abwasser).

5.5 3D-Design und Druck

Um den Temperatursensor zu integrieren, habe ich ein Gehäuse für den Thermistor entworfen und mit einem 3D-Drucker hergestellt (Abbildung 4A). Da Wasserarmaturen in der Regel ein ½-Zoll-Gewinde besitzen und der

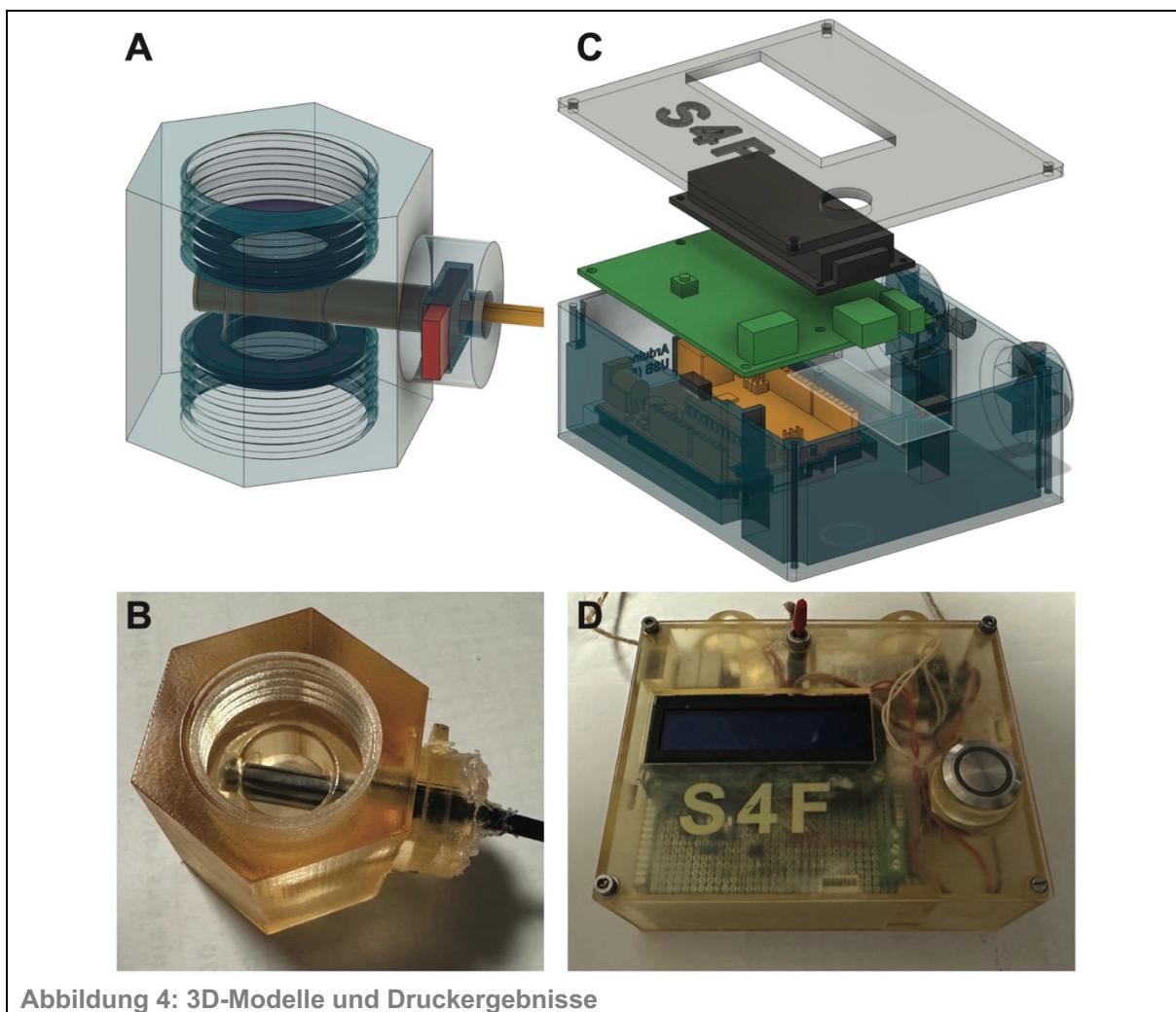


Abbildung 4: 3D-Modelle und Druckergebnisse

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 Komponenten.

Durchflusssensor auf beiden Seiten mit einem ½-Zoll-Außengewinde ausgestattet ist, entschied ich mich, das Thermistorgehäuse mit ½-Zoll-Innengewinden zu entwerfen. Wenn der Fluss- und der Temperatursensor miteinander verschraubt werden, ergibt sich an einem Ende ein Innen- und am anderen Ende ein Außengewinde. Dadurch kann der Adapter problemlos zwischen Armatur und Schlauch montiert werden. Der Thermistor wird mittig in das Gehäuse eingesetzt und mit Silikon wasserdicht versiegelt. Im Inneren des Gehäuses befinden sich zwei Ablagen, die Platz für Gummidichtungen bieten (Abbildung 4B).

Das Gehäuse für die Elektronik wurde exakt auf die Maße des Arduinos, der Platine und des LCD-Displays abgestimmt (Abbildung 4C, D). Der Arduino wird eingelegt, so dass der USB-B-Anschluss zugänglich bleibt. Anschließend wird das verlöste Elektronik-Shield auf den Arduino gesteckt und mit dem Gehäuse verschraubt. Danach wird das LCD-Display auf das Shield aufgesteckt und mit Schrauben befestigt. Eine 9V-Batterie wird in die vorgesehene Aussparung eingelegt und mit dem Shield verbunden. Die Sensoren werden über einen RJ45-Keystone-Stecker¹⁹ angeschlossen, der in das Gehäuse integriert wird und mit den entsprechenden Ports des Shields verbunden ist. Im Gehäusedeckel befinden sich ein Schalter, mit dem die Batterie vom Stromkreis getrennt werden kann, sowie ein Taster, der mit dem Knopf-Port des Shields verbunden ist. Schließlich wird der Deckel auf das Gehäuse geschraubt, sodass das LCD-Display durch eine Aussparung sichtbar ist. Das Gerät ist damit einsatzbereit. Am Gehäuse befindet sich eine Halterung, die es ermöglicht, das S4F-System außen an der Dusche zu befestigen. So können die Daten während des Duschens abgelesen werden (siehe Bild auf Titelseite).

6 Ergebnisdiskussion, Fazit und Ausblick

In meinem Projekt „S4F“ habe ich einen Adapter entwickelt, der auf gängige Wasserarmaturen mit ½-Zoll-Gewinde passt. Dieser Adapter enthält Sensoren, die Wassermenge und -temperatur messen. Die Sensoren sind über eine eigens entwickelte Elektronik mit einem Arduino-Mikrocontroller verbunden. Arduino, Elektronikplatine und ein LCD-Display sind in einem 3D-gedruckten Gehäuse untergebracht.

Das S4F Arduino Betriebssystem misst kontinuierlich Wassermenge und -temperatur und zeigt die Werte auf dem LCD-Display an. Durch erneuten Knopfdruck beendet der Nutzer das Programm. Anschließend berechnet und zeigt der Arduino 5 Werte: (i) Duschzeit, (ii) Wasserverbrauch, (iii) Durchschnittstemperatur, (iv) Energieverbrauch und (v) ungefähre Kosten.

6.1 Erste Eindrücke

Nach der Installation des ersten S4F-Geräts in unserer Dusche haben wir interessante Einblicke in unser Duschverhalten gewonnen. Die gesammelten Daten führten zu spannenden

Gesprächen in der Familie. Besonders aufschlussreich war es zu sehen, wer am längsten und wer am heißesten duscht. Unsere Durchschnittswerte sind:

- Duschdauer: 6,92 Minuten
- Wasserverbrauch: 60,02 Liter
- Durchschnittstemperatur: 42,96 °C
- Energieverbrauch: 2510,63 Wh
- Ungefähr Kosten: € 1,07

Die teuerste Dusche, die wir bisher gemessen haben, dauerte 13.61 Minuten bei 46,38 °C, verbrauchte 98,90 L und kostete etwa € 1,88. Diese Zahlen waren für uns ein Weckruf. Bei sechs Familienmitgliedern könnten sich unsere jährlichen Duschkosten auf etwa €2343,30 belaufen ($\text{€}1,07 \cdot 365 \text{ Tage} \cdot 6 \text{ Personen}$).

Die Informationen, die wir jetzt mit Hilfe der S4F haben, machen uns unseren Ressourcenverbrauch erst richtig bewusst und motivieren uns, nachhaltiger zu handeln. Wir haben uns vorgenommen, unsere Duschgewohnheiten zu ändern, um Wasser, Energie und Kosten zu sparen. Das Material für S4F hat uns ca. €90 gekostet (siehe 9.1), eine Summe, die wir in wenigen Monaten einsparen könnten.

6.2 Zukunft

Es gibt viele Möglichkeiten, S4F weiterzuentwickeln und noch nützlicher zu machen. Meine Mutter wünscht sich z.B., dass das Display die Duschzeit anzeigt, so dass man den Verbrauch besser steuern kann. Sollte ich das Projekt für *Jugend forscht 2025/26* weiterverfolgen, würde ich zuerst daran arbeiten, die erfassten Daten nach der Dusche per E-Mail an die Nutzer zu senden. Dafür wäre jedoch ein anderer Mikrocontroller nötig, da der von mir verwendete Arduino Uno G3 nicht kabellos kommunizieren kann. Eine mögliche Alternative wäre der Arduino Uno G4 mit WLAN oder ein ESP32-Mikrocontroller, der sowohl WLAN als auch Bluetooth unterstützt.

Zwar weiß ich noch nicht wie kompliziert es ist mit einem Microcontroller E-Mails zu versenden, aber wenn mir das Gelingt könnte ich das System auch noch verbessern indem vor jeder Dusche ein Benutzer ausgewählt wird. So könnten die Daten personenspezifisch gespeichert und versendet werden. Über einen längeren Zeitraum ließen sich dann individuelle Zeitprofile erstellen.

Die Nutzung von WLAN würde jedoch den Energieverbrauch erhöhen. Derzeit verwende ich eine 9V-Batterie, die über einen Spannungsregler auf 5V heruntergeregt wird. Dabei geht jedoch viel Energie als Wärme verloren. Eine effizientere Lösung wäre ein wiederaufladbarer Akku, der über einen Boost-Converter den Arduino mit stabilen 5V versorgt. Dies wäre nicht nur energieeffizienter, sondern auch umweltfreundlicher.

Ich kann mir vorstellen, dass ein Unternehmen den S4F Prototyp professionell entwickelt und herstellt. Die Elektronik könnte so miniaturisiert werden, dass der Adapter nur ein wenig größer als der Durchflusssensor selbst wäre, denn der Mikrokontroller-Chip ist nur so groß wie ein Fingernagel. Der Thermistor kann auch

wesentlich kleiner sein und direkt beim Propeller angebracht sein. Statt LCD Display könnte das Modul direkt mit einer App (z.B. über Bluetooth) Informationen geben und Daten loggen. Eine Knopfbatterie würde wahrscheinlich genug Strom liefern, aber vielleicht wäre es sogar möglich den Propeller als Turbine zu nutzen um Spannung zu generieren und einen kleinen Akku zu laden*.

6.3 Ein sozialwissenschaftliches Experiment

Eine weitere interessante Anwendung für S4F wäre ein kleines sozialwissenschaftliches Experiment. Ich könnte ein verbessertes Gerät für einige Wochen an verschiedene Familien meiner Klassenkameraden verleihen. Mit deren Einwilligung könnten die gesammelten Daten über WLAN auch an mich gesendet werden. Den Familien würde ich die ausgewerteten Daten selbstverständlich zur Verfügung stellen. Ich würde die anonymisierten Daten für mein nächstes *Jugend forscht*-Projekt verwenden, um zu untersuchen, ob und wie sich das Duschverhalten durch den Einsatz von S4F verändert. Das Projekt könnte Aufschluss darüber geben, wie technische Lösungen Menschen zu einem nachhaltigeren Verhalten motivieren können.

* Den Durchflusssensor als Turbine zu verwenden, ist eine komplexe Herausforderung, da dies den Einsatz des Propellers zum Antrieb eines Motors erfordert. Wenn der Motor eine Batterie lädt, entsteht ein erhöhter (wahrscheinlich variabler) Widerstand und der CPL-Wert wäre variabel.

7 Quellen- und Literaturverzeichnis

- 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].
- ² Wikipedia.org (2025): Steinhart-Hart Gleichung, [online] <https://de.wikipedia.org/wiki/Steinhart-Hart-Gleichung> [abgerufen am 12.01.2025].
- ³ Wikipedia.org (2025): Hall-Effekt, [online] <https://de.wikipedia.org/wiki/Hall-Effekt> [abgerufen am 12.01.2025].
- ⁴ Arduino.cc (2025): Software | Arduino, [online] <https://www.arduino.cc/en/software> [abgerufen am 12.01.2025].
- ⁵ Arduino.cc (2025): Arduino Reference, [online] <https://www.arduino.cc/reference/cs/> [abgerufen am 12.01.2025].
- 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].
- 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-1570951.html> [abgerufen am 12.01.2025].
- 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].
- 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=AfmBOorcQefaeVhu9hzNdY7BtcV7Ft-uDLdNtTredJ6e7MYfBe4t1id> [abgerufen am 12.01.2025].
- ¹⁰ Autodesk.com (2025): Autodesk Eagle, [online] <https://www.autodesk.com/products/eagle/> [abgerufen am 12.01.2025].
- ¹¹ Autodesk.com (2025): Autodesk Fusion, [online] <https://www.autodesk.com/products/fusion-360> [abgerufen am 12.01.2025].
- 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].
- 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/>
- ¹⁴ Github (2025): S4F, [online] <https://github.com/rzinzen/S4F> [abgerufen am 12.01.2025].
- 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].
- ¹⁶ SRS (2025): Thermistor Calculator v1.1, [online] <https://www.thinksrs.com/downloads/programs/therm%20calc/ntccalibrator/ntccalculator.html> [abgerufen am 12.01.2025].
- ¹⁷ Arduino.cc (2025): attachInterrupt(), [online] <https://www.arduino.cc/reference/cs/language/functions/external-interrupts/attachinterrupt/> [abgerufen am 12.01.2025].
- ¹⁸ Stadtwerke Bernau (2025): Stadtwerke Neukunden – NK-Portal, [online] <https://kundenportal.stadtwerke-berbau.de/Neukunden/Neukunden/Tarifuebersicht> [abgerufen am 12.01.2025].
- ¹⁹ Wikipedia.org (2025): Keystone Module, [online] https://en.wikipedia.org/wiki/Keystone_module [abgerufen am 14.01.2025].

8 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.

Ich habe KI in Form von ChatGPT benutzt um meinen selbst-geschriebenen Text auf Rechtschreib- und Grammatikfehler zu prüfen. Die KI hat auch einige Unklarheiten gefunden, die ich überarbeiten konnte. Ich habe alles auf Richtigkeit geprüft.

9 Anhang

9.1 Teileliste

Nr.	Bauteil	Beschreibung	Kosten (€, ca.)
1	Arduino Uno G3	Microcontroller	14,00
2	LCD 1602	LCD Display mit 16 Zeichen in 2 Reihen	6,00
3	Steckplatine	In einem Elektronik-Lern-Set vorhanden	4,00
4	Jumper Kabel	In einem Elektronik-Lern-Set vorhanden	1,50
5	Thermistor	TRU Components MJSTS-103-3950-1-600-3D Temperatursensor, bestellt bei Conrad Elektronik	2,49
6	FlowSensor	Messing Gehäuse mit Hall Effekt Sensor Bestellt bei Conrad Amazon	12,93
7	Widerstände	In einem Elektronik-Lern-Set vorhanden	2,00
8	Knöpfe	In einem Elektronik-Lern-Set vorhanden	2,00
9	Lötgeräte	z.B. Lötzinn, Flux, Lötstation, Schrumpfschlauch, Heißluftpistole, Silberdraht, 3. Hand, usw. (Grundausrüstung - hatten wir zu Hause)	0,00
10	Potentiometer	10kΩ lineares Potentiometer	2,00
11	Lochrasterplatine	70mm x 90mm, Rastermaß 2,54mm, Teil eines Sets, bestellt bei Reichelt Elektronik	2,60
12	Taster	TRU Components GQ 19H-S bestellt bei Conrad Elektronik	6,99
13	Kippschalter	TRU Components TC-MK245 bestellt bei Conrad Elektronik	1,99
14	Harz für 3D-Drucker	Wurde mir gestellt, kosten ungefähr 21,50 + 5,25	26,70
15	RJ45 Keystone	Delock 8703, Buchse zum Anschließen der Sensoren, im Gehäuse mit der Platine verdrahtet, bestellt bei Reichelt	3,95
16	LAN Internet Kabel	Hatten wir zuhause; eine Seite abgeschnitten und mit den Sensorkabeln verlötet und verschrumpft.	0,00
17	Software	Ich habe nur (für Schüler) kostenlose Software benutzt.	0,00

Summe (ca.): € 89,15

9.2 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