Benz, Samuel

Ein Trinkwasserspender für das Klassenzimmer. Realisierung mit dem Arduino Mikrocontroller

technik-education (tedu). Fachzeitschrift für Unterrichtspraxis und Unterrichtsforschung im allgemeinbildenden Technikunterricht 1 (2021) 2, S. 32-39

technik – education

1. Jahrgang

Fachzeitschrift für Unterrichtspraxis und Unterrichtsforschung im allgemeinbildenden Technikunterricht

2|2021

www.tec-edu.net

tedu

tedu technik - education

1. Jahrgang
2|2021

Fachzeitschrift für Unterrichtspraxis und Unterrichtsforschung im allgemeinbildenden Technikunterricht

https://tec-edu.net/tedu

Mail
herausgeber@tec-edu.net

Anschrift
Pädagogische Hochschule Schw. Gmünd
Institut für Bildung, Beruf und Technik
Abteilung Technik
Oberbettringer Straße 200
73525 Schwäbisch Gmünd
www.tec-edu.net

Inhalt

ISSN: 2748-2022

tedu technik – education

Unterrichtspraxis

Ein Trinkwasserspender für das Klassenzimmer

Realisierung mit dem Arduino Mikrocontroller

Samuel Benz

SCHLAGWORTE

Programmieren
Mikrocontroller
Wasserspender
Elektronik
Messen-Steuern-Regeln

ABSTRACT

Im Folgenden wird die Umsetzung eines Trinkwasserspenders für das Klassenzimmer, der mit Hilfe eines Arduino Microcontroller realisiert wird, beschrieben. Der Grundgedanke, der diesem Projekt zugrunde liegt, ist der, dass ein vorhandener Wasserspender oftmals nur mit einem Wasserhahn gestartet und gestoppt wird. Die Betätigung des Wasserhahns ist einerseits unhygienisch, als auch unpraktisch, wenn lediglich ein Trinkbehälter gefüllt werden soll. Das hier vorgestellte Projekt wird noch über einen Taster betätigt. Wenn die Hygiene mit in die Konstruktion einbezogen wird, kann das Projekt einfach um eine kontaktlose Betätigung mittels eines Annäherungssensors erweitert werden. Das hier vorgestellte Projekt ist dafür kostengünstig und wird von Schüler*innen als sinnvoll wahrgenommen.

Die Aufgabenstellung

Im Folgenden wird zunächst vorgestellt, welche Anforderungen der Wasserspender erfüllen sollte.

- Der Wasserspender soll mittels Drucktaster ein Magnetventil öffnen.
- In Abhängigkeit vom gedrückten Knopf und der im Programmcode hinterlegten Durchflussmenge öffnet und schließt sich ein Magnetventil zur Wasserflussregulation.
- Der Wasserfluss kann unabhängig vom gewählten „Programm" über einen zusätzlichen Taster abgebrochen werden.
- Die Durchflussmenge kann gemessen werden.

Die gesamte Steuerung soll über einen Arduino Mikrocontroller geregelt werden. Im Sinne des EVA-Prinzips erfolgt die Eingabe mittels Schaltern/Tastern und das später beschriebene Flowmeter. Die Verarbeitung erfolgt durch einen Arduino Mikrocontroller und die Ausgabe durch das Magnetventil, sowie durch LEDs.

Die Regulierung des Wasserflusses mit Hilfe eines Magnetventils kann über zwei Ansätze realisiert werden: Entweder wird der Wasserbehälter oberhalb des Auslasses platziert. Damit verfügt das Wasser durch seine Lage über so viel potentielle Energie, dass es mit ausreichend Druck in den Behälter fließen kann. Oder der Wasserspender hat einen Anschluss an eine Wasserleitung, die in der Regel mit über einem Bar Druck arbeitet. Der Druck ist wichtig, damit das Wasser überhaupt fließen kann und der Durchflussmesser zuverlässig zählt.

Damit die Durchflussmenge gemessen werden kann, wird ein Durchflussmesser (Flowmeter) benötigt. In diesem Fall wird ein Flügelrad-Durchflussmesser verwendet. Das Flügelrad kommt durch das vorbeiströmende Wasser in Bewegung. An dem Flügelrad befindet sich ein Magnet, der sich durch die Drehung des Flügelrads mit dreht. Der rotierende Magnet passiert einen Hall-Sensor, der wiederum den Magneten registriert. So kann der Wasserfluss über die Drehung des Magneten am Flügelrad erfasst werden. Der Hall-Sensor sitzt auf dem Gehäuse des Flowmeters, er verwandelt die Drehbewegungen durchflussproportional in ein Frequenzsignal, dass dann wiederum vom Mikrocontroller ausgewertet werden kann.

Optisches Feedback für den Nutzer kann bspw. über in die Taster integrierte Leuchtdioden (LEDs) erzeugt werden. Diese leuchten jeweils dann auf, wenn ein Taster gedrückt wird.

Weitere Möglichkeiten, für eine Differenzierung oder Anpassung an den Einsatzort, können durch den Einsatz eines (Touch-)Displays oder externer LEDs, die ebenfalls mit dem Programm angesteuert werden, realisiert werden. Insbesondere mit Erfahrung im Bereich der Einsatzmöglichkeiten von Mikrocontrollern eröffnen sich zahlreiche didaktische Differenzierungoptionen, auf die hier nicht eingegangen werden kann.

Benötigtes Material

Für die Realisierung des Projekts wird folgendes Material benötigt. Zum Teil weichen die Preise, je nach Anbieter, deutlich voneinander ab. Insbesondere beim Mikrocontroller, beim Magntventil und beim Flowmeter sind deutliche Abweichungen beim Preis in Anhängigkeit der Anbieter möglich.

- 1x Arduino Uno ca. 20,00€

- 1x Magnetventil NC („normal closed“[1]) ca. 50,00€ (www.magnetventile-shop.de)
- 1x Flowmeter ca. 40,00€ (www.magnetventile-shop.de)
- 6x Drucktaster je ca. 3,00€ 1x Relais Schließer ca. 1,50€
- 1x Lochrasterplatine 40mmx10mm ca .2,00€ 1x Netzteil 12V DC 0,5A ca. 5,00€
- 1x USB A zu B Kabel ca. 1,00€
- Schaltlitzen in verschiedenen Farben und Längen ca. 3,00€
- 16x Jumper Kabel männlich (Alternativ 8x und dann halbieren) ca. 3,00€

Die Gesamtkosten belaufen sich für das Projekt auf ca. 140€. Bei einer späteren realen Umsetzung und Nutzung als Trinkwasserspender muss darauf geachtet werden, dass das Flowmeter und das Magnetventil für Trinkwasser (bspw. DIN-DVGW) ausgelegt sein müssen. Eine mögliche Umsetzung mit der Technikklasse für „Trockenübungen“ kann auch deutlich günstiger ausfallen, da bereits Flowmeter für 5€ und Magnetventile ab 6€ erhältlich sind. Diese sind dann jedoch nicht für den Einsatz in Trinkwassersystemen zugelassen.

Bei der Umsetzung eines Systems, das später nicht verwendet werden soll, kann in diesem Zusammenhang auch das in der Industrie übliche Arbeiten mit Prototypen thematisiert werden. Prototypen werden ebenfalls kostengünstig hergestellt, um die Funktion zu testen. Das eigentliche Design und die finale technische Umsetzung erfolgt dann erst später. Dazu passt auch die Verwendung einer Lochrasterplatine, die für eine finale Umsetzung durch eine gefräste oder gedruckte Platine ersetzt werden könnte.

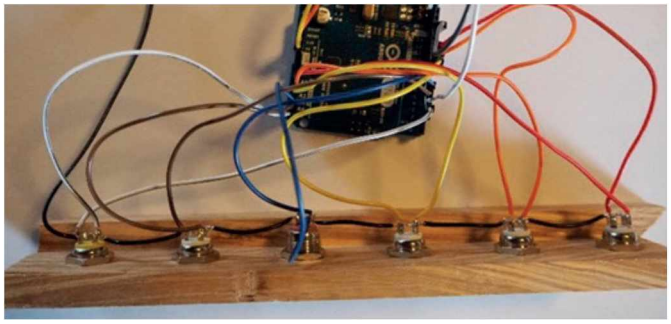

Abbildung 1: Verbindung der Taster und LEDs mit dem Arduino Mikrocontroller

Aufbau des Trinkwasserspenders

Im folgenden Teil wird der Aufbau des Trinkwasserspenders erklärt und dabei auf die Auswahl und Funktion der Hardware eingegangen.

Schalter, Taster und LEDs

Im ersten Schritt werden die Taster verbaut. Das anschließende Verdrahten ist durch die feste Lage leichter, da die Taster nicht fixiert oder später mit angelötetem Kabel weiter verarbeitet werden müssen. Die Taster, die hier verwendet werden, sind: LED-Druckschalter/-taster mit Ringbeleuchtung rot/weiß/orange 12 V, Ø12 mm, 2 A/48 V. Die weißen und die orangen Bauteile sind Drucktaster. Der rote Schalter ist ein Druckschalter.

Die hier verwendete Bauart eignen sich gut, da LED und Schalter/Taster in einem Gehäuse verbaut sind und bereits einen Vorwiderstand für die LED integriert ist. Da sowohl Schalter/Taster, als auch die LED eine GND-Verbindung brauchen, wird diese mit einer Litze verlötet. Diese wird dann auf der Lochrasterplatine zusammen mit den anderen GND-Anschlüssen verlötet. Die restlichen Anschlüsse werden zur einfachen Verbindung mit dem Arduino Mikrocontroller mittels der acht halbierten Jumperkabel auf Seiten der Schalter/Taster verlötet und beim Arduino eingesteckt.

Da die LEDs einen „fade in[2]“ und „fade out[3]“ Effekt haben, muss der Anschluss am Arduino an den „PWM“ (Pulsweitenmodulation) Pins angeschlossen werden. Diese können einen Analogen Ausgang simulieren und somit verschiedene Spannungen „erzeugen“. Das geschieht durch schnelles An- und Ausschalten des Pins. Durch das Ein- und Ausschalten mit einer hohen Frequenz kann in Summe über die Zeit ein analoger Wert simuliert werden. Um diese Funktion nutzen zu können, muss sie im Programmcode aufgerufen und spezifiziert werden. Die Pins, die zu PWM am Arduino Uno in der Lage sind, sind die Pins 3, 5, 6, 9, 10, 11.

Es bietet sich an, wenn die Taster die gleiche Farbe des Jumperkabels bekommen. Sie werden dann an die Analog Inputs angeschlossen. Dies dient lediglich der Übersichtlichkeit der Kabelführung auf dem Arduino Board, da sich die Analog Pins auf der gegenüberliegenden Seite der anderen Pins befinden (siehe Abb. 8). Über die Definition im Programm werden die Analog Input Pins zu digitalen Inputpins umgestellt. Das schließt die Verkablung der Taster ab. Sie sind nun so angeschlossen um programmiert zu werden (siehe hierzu „Zur Programmierung des Trinkwasserspenders“).

Flowmeter

Im nächsten Schritt muss festgestellt werden, wie viele Pulse der Flowmeter auf einen Liter Wasser ausgibt. Der Hersteller gibt dafür meist einen entsprechenden Wert an. Im vorliegenden Fall beträgt die Angabe 2750 pulses/liter. Dieser Wert kann allerdings durch den später anliegenden Wasserdruck abweichen und sollten daher unter den realen Bedingungen verifiziert werden. Die Pulse werden (wie oben

1 „normal closed“ bedeutet, dass im Normalzustand das Magnetventil geschlossen ist. Dies dient der Sicherheit, damit bei Stromausfall oder anderen Szenarien das Magnetventil auch im stromlosen Zustand sich nicht öffnet und ungewollt Wasser austritt.

2 „fade in“ kommt aus dem Englischen und bedeutet wörtlich „einblenden“, in der Technik wird diese Bezeichnung genutzt, wenn z.B. eine LED ihren Zustand von „AUS“ zu „AN“ schrittweise ändert.

3 Umgekehrt bei „fade out“, also „ausblenden“, wird der Zustand von „AN“ zu „AUS“ schrittweise geändert.

```
const int flow = 2; //Input für Signal vom Flowmeter
int pulse = 0;      //Pulszähler auf 0 gesetzt
int var = 0;        //Zählvariable für Pulse auf 0 gesetzt

void setup()
{
pinMode(input, INPUT_PULLUP);
Serial.begin(9600);
}

void loop()
{
  if(digitalRead(flow) > var) { var = 1; pulse++;}    //Wenn der Durchfluss größer als die Variable ist, zähle Variable eins hoch
  Serial.print(pulse);                                //Ausgabe der Pulse am seriellen Monitor
  if(digitalRead(flow) == 0) {var = 0;}               //Wenn kein Durchfluss herrscht, setze Variable auf 0
  delay(1);                                           //Delay für stabilität
}
```

Abbildung 2: Programm zum Auslesen der Pulse

beschrieben) erzeugt, indem ein Schaufelrad mit einem Magneten im Innern des Sensors mit jeder Umdrehung einen Hall-Sensor triggert, der dann über den Signal Ausgang ein Signal liefert.

Einstellen des Flowmeters

Um die tatsächlichen Angaben des Flowmeters zu verifizieren, wird das folgende Vorgehen empfohlen. Ein Programm liest das Signal vom Flowmeter und gibt es über den seriellen Monitor des Arduinos aus. Der Messapparat sieht so aus, dass am Auslass des Durchflussmessers ein Messbecher platziert wird, in den das durchgeflossene Wasser fließt. Wenn nun das Wasser durchfließt und der serielle Monitor die Pulse des Durchflussmessers anzeigt, können die Pulse direkt mit dem Füllstand verglichen werden. Sobald ein Liter im Messbecher ist, wird die Wasserzufuhr gestoppt und die Pulse können abgelesen werden. Um ein zuverlässiges Messergebnis zu bekommen, wird dieser Vorgang wiederholt. Der ermittelte Wert kann dann hoch- und runtergerechnet werden (für den vorliegenden Fall sind dies 0,4l und 1,5l). Als Alternative zu einem Messbecher kann auch eine Waage verwendet werden, um die durchgeflossene Menge Wasser zu ermitteln.

Bei der Ermittlung der Werte für den Flowmeter konnten im vorliegenden Fall exakt die vom Hersteller angegeben 2750pulses/liter bestätigt werden. Somit können die Pulse für 0,4l = 1100 Pulse und 1,5l = 4125 Pulse berechnet werden. Diese Werte werden später für die Programmierung des Hauptprogramms benötigt.

Das Magnetventil

Das Magnetventil wird mit 12V DC und 0,5A beschaltet. Es öffnet und schließt später die Wasserzufuhr. Da der Arduino Microcontroller maximal 5V Spannung ausgeben kann, wird hier auf ein externes Netzteil (12V DC, 0,5A) zurückgegriffen. Die Schaltung über den Arduino Microcontroller erfolgt später mittels eines Relais, das den Steuerstromkreis bei einem HIGH Signal schließt und so das Magnetventil öffnet. Da die Anwendung nicht zeitkritisch im Millisekundenbereich ist, sowie aufgrund der einfachen Beschaltung, kann ein Relais verwendet werden und es muss kein MOSFET verbaut werden.

Bei der Beschaffung des Magnetventils ist darauf zu achten, dass das Magnetventil im Normalzustand keinen Strom leitet und geschlossen ist. So wird verhindert, dass bei einem Stromausfall oder einer Störung Wasser unkontrolliert austritt.

Auch Relais gibt es in Ausführungen die im Normalzustand entweder einen Stromkreis schließen oder unterbrechen. Das hier verwendete Relais muss im Normalzustand unterbrechen. Wie schon beim Magnetventil wird so sichergestellt, dass auch bei einer Störung kein Wasser austritt.

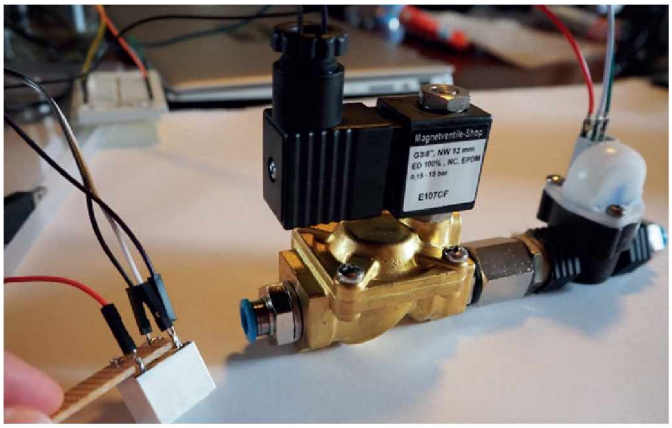

Abbildung 3: Baugruppe Magnetventil und Flowmeter

Die Angabe um welche Art von Relais es sich handelt, ist meist in der Typ Bezeichnung mit „NC“ oder „NO“ kenntlich gemacht. Die Bezeichnung für das vorliegende Magnetventil: Elektro-Magnetventil IG3/8 Zoll NC vorgesteuert. „IG3/8 Zoll“ gibt an, dass hier ein Innengewinde mit der Größe 3/8 Zoll vorliegt. Auch der Flowmeter hat ein Innengewinde mit 3/8 Zoll.

Der Anschluss für das Magnetventil ist wie folgt: Am Arduino Microcontroller wird die Spule des Relais mit Pin 4 und GND verbunden. Bei einem HIGH Signal des Steuerstroms schließt das Relais und schaltet den Laststrom. Der Laststromkreis verbindet einen Ausgang des Relais am Magnetventil mit dem Netzteil. Die GND-Anschlüsse von Magnetventil und Netzteil werden miteinander verbunden. Der

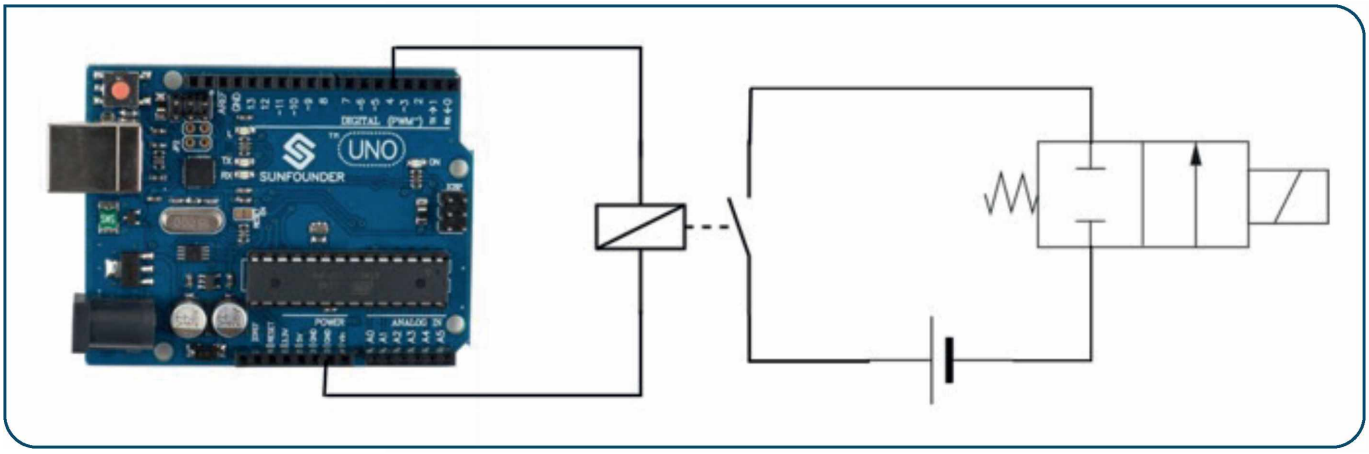

Abbildung 4: Schaltplan Anschluss Magnetventil und Relais mit Spannungsversorgung

Laststromkreis kann ebenfalls auf der Lochrasterplatine untergebracht werden.

Verbindung der Wasserleitungen

Wie oben erwähnt, wurden beim Flowmeter und beim Elektroventil ein 3/8 Zoll Innengewinde verwendet. Das Mittelstück zwischen den beiden Bauteile hat ebenfalls ein Innengewinde mit 3/8 Zoll. Magnetventil, Flowmeter und Mittelstück werden mit zwei 3/8 Zoll Außengewindeverbindern verbunden. Das hat den Vorteil, dass beide Seiten unabhängig voneinander fest angezogen werden können.

Abbildung 5: Verbindungsstück zwischen Flowmeter und Magnetventil

Um die Dichtigkeit gewährleisten zu können, wird bei jeder Verbindung sowohl eine Gummidichtung verwendet als auch ein Dichtungshanf, das auf das Gewinde gelegt und dann mit verschraubt wird.

Um die Schläuche anzuschließen, werden Schnellverbinder verwendet. Die Schnellverbinder dichten gleichzeitig den Schlauch an den Anschlussstellen ab. Die Verschraubung selbst hat an der Verbindungsstelle zum Bauteil ebenfalls noch eine Gummidichtung. Es ist wichtig, dass bei der Wahl des Schlauchs oder Steckverbinders beachtet wird, dass der Außendurchmesser des Schlauchs die relevante Bezugsgröße ist, da der Steckverbinder von außen her abdichtet. Für einen Wasserspender, der tatsächlich eingesetzt werden soll, muss außerdem auf die Trinkwassereignung aller Bauteile, also auch der Schläuche geachtet werden.

Abbildung 6: Anschluss Steckverbinder Schlauch

Damit sind alle Komponenten für dieses Gesamtprojekt erklärt und bereit zum Zusammenbau (vgl. Abb. 7 und Abb. 8).

Programmierung des Trinkwasserspenders

Die Hardware für die Eingabe und die Ausgabe sind soweit fertiggestellt. Um die Verarbeitung zu ermöglichen, folgt nun die Vorstellung des Quellcodes für dieses Projekt. Beim Quellcode für das „faden" der LEDs und zum Zählen der Pulse kann auf schon vorhandene Beispielsketche im Arduino Programm (Arduino CC öffnen - „Datei" - „Beispiele") zurückgegriffen werden. Diese beispielhaften Programme liefern allerdings nur das Grundgerüst und müssen angepasst werden.

Die Programmiersprache für den Arduino basiert auf C, C++ und Java. Um Funktionen und Abläufe nachschlagen zu können, hat Arduino eine Sprachreferenz auf der Webseite (https://www.arduino.cc/reference/de/), die sich bisher als sehr nützlich erwiesen hat. Ebenso ist das Forum (https://forum.arduino.cc/) ein gut nutzbares Tool, um Probleme nachschlagen zu können.

Didaktische Aspekte der Programmplanung

Bevor der Quellcode vorgestellt wird, soll hier zunächst noch einmal in verständlicher Textform beschrieben werden, was das Programm zu tun hat. Dieses Vorgehen ist auch im Technikunterricht sehr sinnvoll, da sich hier bereits zeigt, ob die Schüler*innen verstanden haben, was das Programm leisten soll, ohne dieses Verständnis durch Fehler in der Anwendung der komplizierten Programmiersprache zu verdecken. Die Übertragung der verständlich formulierten Anweisungen ist dann mit den oben aufgeführten Quellen für den Arduinocode zwar immer noch komplex, Fehler können dann aber auf die Programmierung zurückgeführt wer-

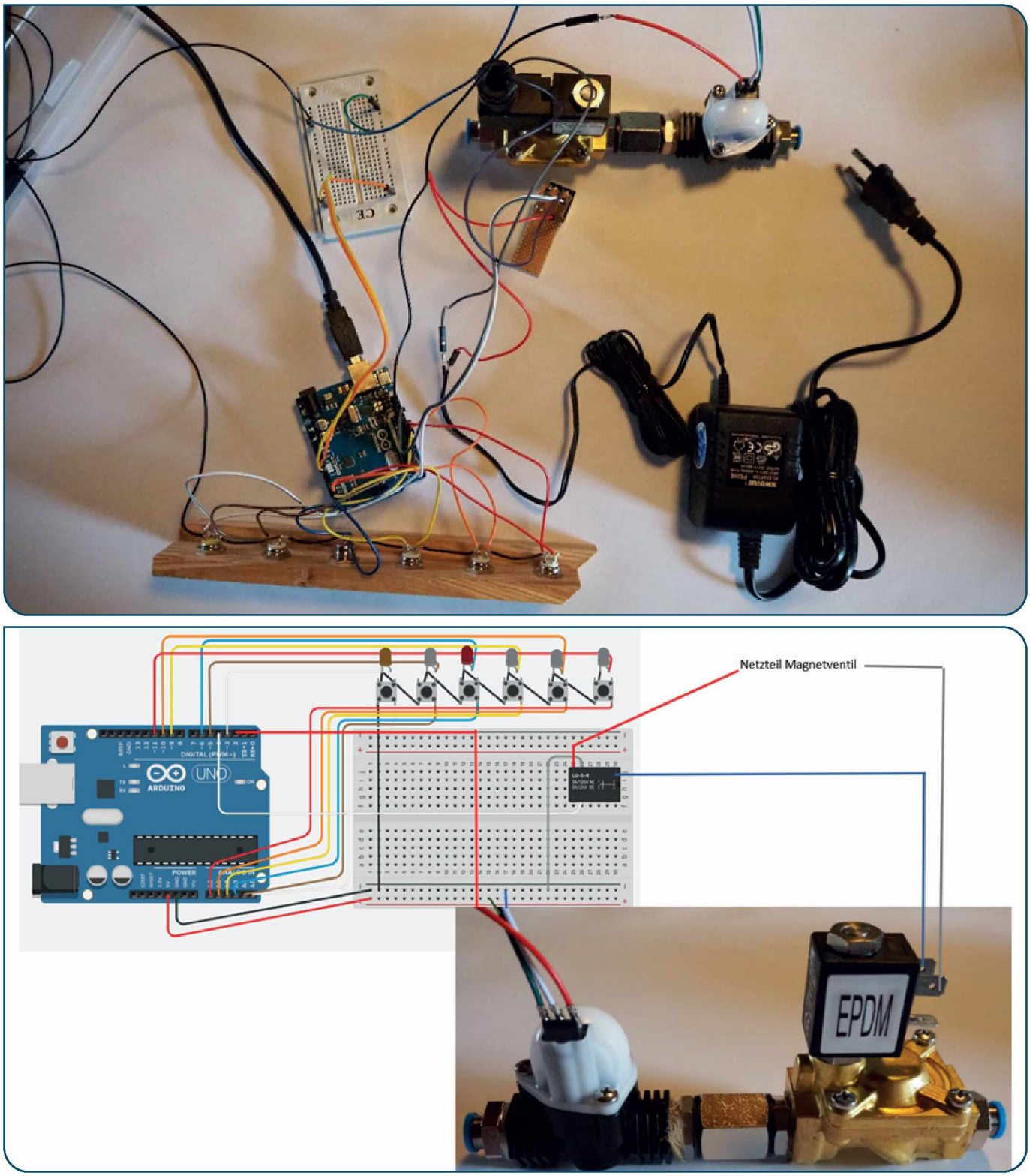

Abbildung 7: Tatsächliche und schematische Verbindung der Komponenten

den. Fehler bei der verständlichen Programmierung deuten auf Defizite beim technischen Verständnis hin.

Im Programm müssen die Eingänge und Ausgänge festgelegt werden. Als Eingänge gibt es die Taster und das Signal vom Flowmeter. Ausgänge sind die LEDs und die Steuerung des Relais. Das Programm muss nun ständig abfragen, welcher Taster gedrückt wird. Dabei wird unterschieden:

- 0,4l: Abbruchtaster und der aktuelle Taster leuchten auf, es werden 0,4l ausgelassen
- 1,0l: Ebenfalls leuchten die beiden Taster auf und es werden 1,0l ausgelassen
- 1,5l: Ebenfalls leuchten die beiden Taster auf und es werden 1,5l ausgelassen
- Hold: Schalter wird gedrückt, rote LED geht an und das Wasser fließt, bis der Schalter erneut betätigt wird
- Push: Solange der Taster gedrückt wird, leuchtet die

LED und es fließt das Wasser
- Abbruch: LED leuchtet bei Bereitschaft, bei Betätigen wird das Magnetventil sofort geschlossen

Entwicklung des Programms in vier Schritten

Das Programm wird nun in vier Teilen beschrieben. Diese vier Teile orientieren sich an der Struktur des Arduino-Programmaufbaus. Zunächst wird auf das Deklarieren der Variablen eingegangen. Danach wird der Setup beschrieben. Es folgt die Beschreibung des Hauptprogramms. Schließlich wird noch gesondert auf Funktionen eingegangen.

Deklarieren der Variablen

„int" ist der Datentyp der Variablen. Dieser kann ganze Zahlen speichern. Die Variablen können im Programm logisch mit Pins verbunden werden, oder sie können Werte speichern. Bei den Variablen ist es sinnvoll, wenn kurze und gleichzeitig aussagekräftige Bezeichnungen gewählt werden. Im Code ist es dann einfacher von „led3" zu sprechen als von „Pin3". Dieses Vorgehen wird besonders bei komplexeren Programmen notwendig und sollte daher auch schon bei übersichtlichen Programmen beherzigt werden. Nach jeder Anweisung wird mit einem Semikolon die Zeile für das Programm beendet. Alles was nach „//" steht, ist ein Kommentar, der vom Programmablauf ignoriert wird. Diese Kommentare sind erneut für den Technikunterricht sehr sinnvoll, da die Schüler*innen hier in verständlicher Sprache schreiben können, was sie in der Programmzeile ausdrücken wollen. Das erleichtert erneut die Diagnose und die Bewertung der Leistung der Schüler*innen und hilft bei der Fehlerbeseitigung.

```
int led3 = 3;            // LED Taster Abbruch digital pin 3  (weiß)
int led5 = 5;            // LED Taster Push digital pin 5     (braun)
int led6 = 6;            // LED Taster Hold digital pin 6     (blau)
int led9 = 9;            // LED Taster 1,5l digital pin 9     (gelb)
int led10 = 10;          // LED Taster 1,0l digital pin 10    (orange
int led11 = 11;          // LED Taster 0,4l digital pin 11    (rot)
int taster0 = A0;        // Taster 0,4l analog pin 0
int taster1 = A1;        // Taster 1,0l analog pin 1
int taster2 = A2;        // Taster 1,5l analog pin 2
int taster3 = A3;        // Taster Hold analog pin 3
int taster4 = A4;        // Taster Push analog pin 4
int taster5 = A5;        // Taster Abbruch analog pin 5
int ts0;                 //Tasterstatus 1
int ts1;                 //Tasterstatus 2
int ts2;                 //Tasterstatus 3
int ts3;                 //Tasterstatus 4
int ts4;                 //Tasterstatus 5
int ts5;                 //Tasterstatus 6
const int flow = 2;      //Signal von Durchflussmesser
int pulse = 0;           //Pulszähler auf 0 gesetzt
int var = 0;             //Zählvariable für Pulse auf 0 gesetzt
int maxDurchfluss = 0;   //Durchflussmenge auf 0 gesetzt
int relais = 4;          //Relais für Magnetventil digital pin 4
```

Abbildung 8: Beginn des Programms; Deklarieren der Variablen

Setup

Mit dem Befehl „pinMode" wird festgelegt, welche Funktion ein Pin besitzt. Pins können entweder als Input oder als Output definiert werden. Die Definition steht dann in der Klammer. So steht hier in Zeile 1, dass der zuvor festgelegte Pin 3 (led3) ein Output ist. Das ist schlüssig, wenn klar ist, dass damit eine LED geschaltet werden soll. Eine Besonderheit ist der letzte pinMode- Befehl „pinMode (flow, INPUT_PULLUP). Dieser Pin wurde als Input festgelegt. „PULLUP" ist eine Sonderform der Inputs. Der Arduino verfügt über integrierte Pullup-Widerstände, die mit diesem Befehl aktiviert werden können. Diese sorgen dafür, dass es bei den empfindlichen Pins zu keinen Messfehlern kommt, wenn das Eingangssignal im Schwellenwertbereich zwischen High und Low liegt. Das Schalten erfolgt dann nicht über ein HIGH-Signal, sondern über das Verbinden des Pins mit dem GND, also über ein LOW-Signal.

```
void setup() {
  pinMode(led3, OUTPUT);        //pin als Output festgelegt
  pinMode(led5, OUTPUT);        // - " -
  pinMode(led6, OUTPUT);        // - " -
  pinMode(led9, OUTPUT);        // - " -
  pinMode(led10, OUTPUT);       // - " -
  pinMode(led11, OUTPUT);       // - " -
  pinMode(taster0, INPUT);      //pin als Input festgelegt
  pinMode(taster1, INPUT);      // - " -
  pinMode(taster2, INPUT);      // - " -
  pinMode(taster3, INPUT);      // - " -
  pinMode(taster4, INPUT);      // - " -
  pinMode(taster5, INPUT);      // - " -
  pinMode(relais, OUTPUT);      // - " -
  pinMode(flow, INPUT_PULLUP); //pin als Input festgelegt, interner Pullup Widerstand aktiviert
  digitalWrite(taster0, HIGH); //wegen des Pullup Widerstands muss der pin auf HIGH
  digitalWrite(taster1, HIGH); //gesetzt werden, damit ein eindeutiges Signal anliegt
  digitalWrite(taster2, HIGH);
  digitalWrite(taster3, HIGH);
  digitalWrite(taster4, HIGH);
  digitalWrite(taster5, HIGH);
  Serial.begin(9600);           //Serielle Kommunikation mit PC starten

}
```

Abbildung 9: Das Setup des Programms

Pullup- und Pulldownwiderstände können auch außerhalb des Mikrocontrollers über Hardwareschaltungen realsiert werden. Die Verwendung innerhalb des Arduinos ist allerdings deutlich effizienter und sollte, wenn keine didaktischen Gründe dagegen sprechen, vorgezogen werden. Es gibt auch andere Möglichkeiten für ein stabiles Eingangssignal zu sorgen. Auf diese soll hier jedoch nicht eingegangen werden.

Für den Fall, dass man nachträglich die Pulse anpassen muss, bietet es sich an, die Befehle zur Ausgabe der Pulse über den Serial Monitor im Code zu lassen. Das betrifft im Setup die letzte Zeile. Der Befehl „Serial.begin(9600)" startet die Kommunikation mit dem PC. Das bedeutet, dass bei dem Befehlsaufruf der serielle Monitor des Programms gestartet wird, an dem dann die Pulse angezeigt werden, wenn der Arduino mit einem Computer verbunden wird. Ohne Anschluss an einen Computer hat der Befehl keine Auswirkung auf den Ablauf des Programms. Die Nutzung des Serial Monitors wurde schon im ersten Schritt angewandt, als die Pulse für den Flowmeter ausgelesen wurden.

Hauptprogramm

Mit „void loop()" beginnt das Hauptprogramm. Dieses läuft als Endlosschleife, solange der Arduino Microcontroller mit Strom versorgt wird. Zu Beginn wird nochmals der Wert der aller Taster auf HIGH gesetzt, da wie oben beschrieben über LOW geschaltet werden soll. Zusätzlich wird das Relais offengehalten, damit das Magnetventil nicht durch ein unklares Signal geöffnet wird.

Die Variablen, die mit „ts..." starten, werden verwendet, um den Status der Taster abzufragen. Dies erfolgt über den Befehl „digitalRead(taster x)", wobei „x" durch die jeweilige

```
void loop() {
   digitalWrite(taster0, HIGH);
   digitalWrite(taster1, HIGH);
   digitalWrite(taster2, HIGH);
   digitalWrite(taster3, HIGH);
   digitalWrite(taster4, HIGH);
   digitalWrite(taster5, HIGH);
   digitalWrite(relais, LOW);    //Relais offen halten
   ts0 = digitalRead(taster0);   //Abfrage des Tasterstatus
   ts1 = digitalRead(taster1);   // - " -
   ts2 = digitalRead(taster2);   // - " -
   ts3 = digitalRead(taster3);   // - " -
   ts4 = digitalRead(taster4);   // - " -
   ts5 = digitalRead(taster5);   // - " -


   if(ts0 == LOW && ts1 == HIGH && ts2 == HIGH && ts3 == HIGH && ts4 == HIGH)  //Wenn Taster 1 gedrückt, springe in die Schleife
   {
     pulse = 0;                                          //Puls auf 0 gesetzt
     maxDurchfluss = 1100;                               //1100 Pulse entsprechen 0,4l
     fadingIn(led11);                                    //Rufe Funktion fadingIn auf (siehe unten), übergebe aktuelle LED
     while(pulse < maxDurchfluss)                        //Beginn der while Schleife, sie läuft solange die Durchflussmenge
       {                                                 //von 1100 Pulse noch nicht erreicht ist
         digitalWrite(relais, HIGH);                     //Schließe das Relais -> Magnetventil öffnet
         if(digitalRead(taster5) == HIGH)                //Programm wird fortgesetzt, wenn Abbruchtaster nicht gedrückt
         {
           if(digitalRead(flow) > var) { var = 1; pulse++;}  //Wenn der Durchfluss größer als die Variable ist, zähle Variable eins hoch
           Serial.print(pulse);                          //Ausgabe der Pulse am seriellen Monitor
           if(digitalRead(flow) == 0) {var = 0;}         //Wenn kein Durchfluss herrscht, setze Variable auf 0
           delay(1);                                     //Delay für stabilität
         }
         else                                            //Wenn Abbruchtaster gedrückt, springe aus Schleife
         {
              break;                                     //Abbruch der Schleife
         }
       }
     digitalWrite(relais, LOW);                          //Öffne das Relais --> Magnetventil schließt
     fadingOut(led11);                                   //Rufe Funktion fadingOut auf (siehe unten)
   }
```

Abbildung 10: Der Beginn des Hauptprogramms

Zahl ersetzt wird. Es folgt die erste bedingte Verzweigung (if): Wenn taster0 ein LOW Signal liefert UND (&&) alle anderen Taster nicht gedrückt sind, dann wird zunächst der maximale Durchfluss bestimmt und dann die Funktion „fadingIn()“ aufgerufen. Diese Funktion wird später beschrieben.

Danach wechselt das Programm in eine While-Schleife. Damit beginnt der Vorgang das Wasser fließen zu lassen und den Durchfluss zu zählen. Die While-Schleife läuft so lange, bis die Bedingung für die Schleife nicht mehr erfüllt ist.

Die Bedingung lautet, dass die gezählten Pulse kleiner sind als die maximale Durchflussmenge. Anschließend wird das Relais geschlossen. Zusätzlich wird bei jedem Schleifendurchlauf abgefragt, ob der Abbruchtaster gedrückt wurde. Bei der Rechengeschwindigkeit des Arduino erfolgt dies mehrere Tausend mal pro Sekunde. Falls der Abbruchtaster gedrückt wird, springt das Programm zu „else“ und beendet die While-Schleife mit „break“.

Wird nicht abgebrochen, endet die While-Schleife, wenn die Durchflussmenge dem SOLL-Wert entspricht. Dies ist bei der aktuellen Einstellung bei 0.4 l Durchfluss der Fall. Ist die Bedingung der While-Schleife nicht mehr gegeben, wird die While-Schleife beendet. Dadurch wird das Relais wieder Stromlos und die Funktion „fadingOut()“ aufgerufen.

```
while(ts3 == LOW)                                          //Solange der Hold Taster gedrückt ist, mache folgendes
{
  digitalWrite(relais, HIGH);                              //Magnetventil geöffnet
  for (int fadeValue = 0 ; fadeValue <= 255; fadeValue += 5) //Fade von minimum zu maximum in 5er Schritten
  {
    analogWrite(led6, fadeValue);                          //Schreibe den Wert für Fade an den Pin der LED
    delay(30);                                             //Delay für sichtbares faden
  }
  for (int fadeValue = 255 ; fadeValue >= 0; fadeValue -= 5) //Fade von maximum zu minimum in 5er Schritten
  {
    // sets the value (range from 0 to 255):
    analogWrite(led6, fadeValue);                          //Schreibe den Wert für Fade an den Pin der LED
    delay(30);                                             //Delay für sichtbares faden
  }
  digitalWrite(taster3, HIGH);                             //Schreibe an den Taster Hold den Wert HIGH
  ts3 = digitalRead(taster3);                              //Lese den Status von Taster Hold
}                                                          //--> sonst hängt das Programm in Endlosschleife fest
```

Abbildung 11: Taster „Hold“ - ein Teil des Hauptprogramms

Die Taster Push und Hold funktionieren nach dem oben beschriebenen Code. Durch das Betätigen des Tasters Push wird direkt in die While-Schleife gesprungen. Dadurch wird das Relais geschlossen und somit das Magnetventil geöffnet.

Die for-Schleife wurde in das Programm integriert, damit die LEDs der Taster einfaden und ausfaden. Dies passiert solange der Taster gedrückt wird.

Abschließend muss in jedem Schleifendurchlauf der Wert des Tasters erneut ausgelesen werden, da sich das Programm sonst in einer Endlosschleife gefangen hält, ohne eine Möglichkeit einzugreifen. Alternativ kann am Ende der Schleife die Eingangsbedingung abgefragt werden, damit das Programm prüfen kann, ob es noch einmal die Schleife durchlaufen muss. Ohne diese Abfrage würde diese Überprüfung nie aktualisiert und das Programm würde die Schleife nie verlassen. Es würde lediglich ein Reset auf dem Arduino Microcontroller Board als letzte Möglichkeit verbleiben.

Die Funktionen „fadingIn()" und „fadingOut()"

Oben wurde bereits auf den Funktionsaufruf verwiesen.

```
void fadingIn(int led)                                    //Funktion für einfaden der LED, übernimmt LED von Funktionsaufruf
{
      for (int fadeValue = 0 ; fadeValue <= 255; fadeValue += 5) //Fade von minimum zu maximum in 5er Schritten
      {
        analogWrite(led, fadeValue);                      //Schreibe den Wert für Fade an die aktuelle LED
        analogWrite(led3, fadeValue);                     //Schreibe den Wert für Fade an die Abbruch LED
        delay(30);                                        //Delay für sichtbares faden
      }
}

void fadingOut(int led)                                   //Funktion für ausfaden der LED, übernimmt LED von Funktionsaufruf
{
      for (int fadeValue = 255 ; fadeValue >= 0; fadeValue -= 5)  //Fade von maximum zu minimum in 5er Schritten
      {
        analogWrite(led, fadeValue);                      //Schreibe den Wert für Fade an die aktuelle LED
        analogWrite(led3, fadeValue);                     //Schreibe den Wert für Fade an die Abbruch LED
        delay(30);                                        //Delay für sichtbares faden
      }
}
```

Abbildung 12: Funktionen als zusätzlicher Programmteil

Bei den Funktionen handelt es sich um einen Programmabschnitt, der erst dann abgearbeitet wird, wenn die besagte Funktion aufgerufen wird. Dieses Vorgehen empfiehlt sich für Programmteile, die an unterschiedlichen Stellen im Programm benötigt werden. Es erspart den Programmierer*innen, den Code immer wieder in das Programm zu integrieren. Hier wird das faden der LEDs aufgezeigt. Da dies an mehreren Programmstellen vorkommt, aber immer unverändert bleibt, wird durch die Funktion Text im Hauptprogramm gespart.

Da allerdings die LEDs unterschiedliche „Namen" haben, muss der jeweilige Name der angesprochenen LED im Funktionsaufruf mit an die Funktion übergeben werden, zum Beispiel: „fadingIn(led11);" Dann springt das Programm in die Funktion und übergibt den Wert „led11" an den Platzhalter „led" in der Funktion. Beim Faden wird der analoge Wert von 0 bis 255 in 5er Schritten erhöht und mit einem Delay (Verzögerung) von 30ms der Effekt des Hellerwerdens oder des Dunklerwerdens erzeugt.

Lernpotentiale für den Technikunterricht

Die Fertigung des Trinkwasserspenders ermöglicht, verschiedene Problemstellungen der technischen Umwelt in einem Artefakt zu vereinen. Die grundsätzlichen Fertigkeiten können wiederholt und gefestigt werden (bspw. bohren, löten, schleifen). Das Projekt fördert die Problemlösekompetenz durch das Programmieren, da hier nach dem Prinzip „Trial and Error" in mehreren Schritten auf die richtige Lösung (in dem Fall das funktionierende Programm) hingearbeitet wird. Außerdem kann viel über die Beschaffenheit der einzelnen Werkstoffe und deren sinnvoller Einsatz gesprochen werden, zum Beispiel welche Stoffe sich für Schläuche eignen. Ein weiterer Aspekt, der bearbeitet werden kann, ist die Gestaltung eines Gehäuses für den Arduino und für die Taster. Dies ist abhängig von dem Einbauort. Aus der Umweltperspektive ist dieses Projekt geeignet, um den Schülerinnen und Schülern den Blick zu schärfen können für nachhaltigen Umgang mit den natürlichen Ressourcen.

Autoreninformation

Samuel Benz
studiert an der PH Schwäbisch Gmünd Lehramt für die Sekundarstufe 1 (B.A./M.Ed.) mit dem Fach Technik. In seiner Bachelorarbeit beschäftigt er sich momentan mit der Konzeption und Entwicklung einer mobilen Unterrichtseinheit für den Einsatz von Mikrocontrollern im Technikunterricht.

tedu

2|2021