Jugend forscht junior

Entwicklung eines Umweltdaten-Messgeräts für die Schmerzforschung

Technik

Samuel Komann, Leonard Schmidt

Schülerforschungszentrum Jena, Carl-Zeiss-Gymnasium Jena

Landeswettbewerb Thüringen 2026

Betreut von Frau Walther und Herrn Linhart

Projektüberblick

Das Ziel dieses Projekts war die Entwicklung eines Messgeräts zur Erfassung und Protokollierung der Beleuchtungsstärke und anderer Umweltdaten, um deren mögliche Korrelation mit dem postoperativen Schmerzempfinden, also der Stärke des Schmerzes nach Operationen, von Patienten verschiedener Kliniken zu untersuchen, die an der Forschung der QUIPS-Abteilung (Qualitätsverbesserung in der postoperativen Schmerztherapie) des Universitätsklinikums Jena beteiligt sind. Als technische Basis dient ein Raspberry Pi Pico 2, an den verschiedene Sensoren angeschlossen sind. Diese messen die Umweltdaten und der Mikrocontroller speichert diese auf einer microSD-Karte. Bei der Entwicklung des Geräts lag der Schwerpunkt auf Robustheit und einer besonders einfachen Bedienbarkeit. Eine zusätzliche Herausforderung stellten die zuverlässige Stromversorgung sowie das spezifische Einsatzumfeld - das Patientenzimmer - dar. Zum Projekt gehört außerdem die Erstellung einer leicht verständlichen Bedienungsanleitung. Die Projektidee entstand aus dem Wunsch der QUIPS-Gruppe, ein solches Gerät zu entwickeln. In Kürze soll ein multi-zentraler Einsatz im Rahmen einer Studie der QUIPS-Gruppe beginnen.

Inhaltsverzeichnis

1 Motivation und Fragestellung

Umweltfaktoren können Einfluss auf die Gesundheit haben. Bei einigen ist dies bekannt, beispielsweise Lärm[1] oder Feinstaub[2]. Bei anderen gibt es Hinweise, aber die Studienlage ist unklar. Dazu gehört zum Beispiel der Einfluss von Licht[3].

In einer früheren Studie der QUIPS-Gruppe des Universitätsklinikums Jena[4] zum Thema „Die Auswirkung der Umgebungslichtintensität auf postoperatives Schmerzempfinden“[5], wurde dies untersucht. Hier wurde keine direkte Korrelation zum Schmerzempfinden festgestellt, jedoch aber zur Übelkeit der Patienten.

In der Ergebnisdiskussion der vorangegangenen Studie wurde die Messung zu genau einem Zeitpunkt als eine der Schwächen erwähnt: „Für künftige Untersuchungen wäre es wünschenswert, anspruchsvollere Geräte zu verwenden, die in der Lage sind, die Schwankungen der Lichtintensität in den Patientenzimmern den ganzen Tag über zu beobachten.“[6] Da sich Lichtverhältnisse über die Zeit ändern können, stellt die einmalige Messung eine Fehlerquelle dar. So wurde der Wunsch nach einem Gerät, welches die Umweltdaten permanent aufnimmt und speichert und im klinischen Alltag einsetzbar ist, von Autor Dr. Marcus Komann an uns herangetragen, da es auf dem freien Markt kein Gerät gibt, das den Anforderung (siehe *2 Hintergrund und theoretische Grundlagen*) entspricht.

Ziel dieser Arbeit war also die Entwicklung eines den Anforderungen entsprechenden integrierten Geräts, mit dem weitere Studien in Krankenhäusern durchgeführt werden können. Außerdem sollte es zur einfachen Bedienung eine kurze Anleitung geben.

[1] siehe: Kvandová M, Rajlic S, Stamm P, Schmal I, Mihaliková D, Kuntic M, Jimenez MTB, Hahad O, Kollárová M, Ubbens H, Strohm L, Frenis K, Duerr GD, Foretz M, Viollet B, Ruan Y, Jiang S, Tang Q, Kleinert H, Rapp S, Gericke A, Schulz E, Oelze M, Keaney Jr. JF, Daiber A, Kröller-Schön S, Jansen T, and Münzel T (2023). “Mitigation of aircraft noise-induced vascular dysfunction and oxidative stress by exercise, fasting, and pharmacological a1AMPK activation: molecular proof of a protective key role of endothelial a1AMPK against environmental noise exposure”. In: *European Journal of Preventive Cardiology (2023) 30, 1554–1568* 30. DOI: {10.19224/ai2025.013}.

[2] siehe: Kvandová M, Rajlic S, Stamm P, Schmal I, Mihaliková D, Kuntic M, Jimenez MTB, Hahad O, Kollárová M, Ubbens H, Strohm L, Frenis K, Duerr GD, Foretz M, Viollet B, Ruan Y, Jiang S, Tang Q, Kleinert H, Rapp S, Gericke A, Schulz E, Oelze M, Keaney Jr. JF, Daiber A, Kröller-Schön S, Jansen T, and Münzel T (2023). “Mitigation of aircraft noise-induced vascular dysfunction and oxidative stress by exercise, fasting, and pharmacological a1AMPK activation: molecular proof of a protective key role of endothelial a1AMPK against environmental noise exposure”. In: *European Journal of Preventive Cardiology (2023) 30, 1554–1568* 30. DOI: {10.19224/ai2025.013}.

[3] siehe: Scheller JS, Komann M, Weinmann C, Weinmann J, Heitfeld S, Mielke Aet al: (2025). “Auf der Sonnenseite des Lebens – Die Auswirkung der Umgebungslichtintensität auf postoperatives Schmerzempfinden.” In: *Anästh Intensivme 2025;66:13–19*. DOI: {10.19224/ai2025.013}.

[4] *QUIPS* (o. D.). de. https://www.quips-projekt.de/. Besucht: 12.01.2026.

[5] siehe: Scheller JS, Komann M, Weinmann C, Weinmann J, Heitfeld S, Mielke Aet al: (2025). “Auf der Sonnenseite des Lebens – Die Auswirkung der Umgebungslichtintensität auf postoperatives Schmerzempfinden.” In: *Anästh Intensivme 2025;66:13–19*. DOI: {10.19224/ai2025.013}.

[6] Scheller JS, Komann M, Weinmann C, Weinmann J, Heitfeld S, Mielke Aet al: (2025). “Auf der Sonnenseite des Lebens – Die Auswirkung der Umgebungslichtintensität auf postoperatives Schmerzempfinden.” In: *Anästh Intensivme 2025;66:13–19*. DOI: {10.19224/ai2025.013}, S. 17-18.

2 Hintergrund und theoretische Grundlagen

Die QUIPS-Gruppe formulierte die folgenden Anforderungen. Das Gerät soll:

- über einen längeren Zeitraum die Beleuchtungsstärke jede Minute einmal messen
- weitere Umweltfaktoren aufnehmen
- alle Messwerte speichern
- im Patientenzimmer einsetzbar sein, d.h., klein, leise, robust, batteriebetrieben, leicht zu bedienen sein
- eine leicht verständliche Anleitung haben

Die Beleuchtungsstärke E_v ist der Lichtstrom Φ_v pro Fläche A:

$$E_\mathrm{v} = \frac{\Phi_\mathrm{v}}{A}$$

Sie wird in Lumen pro Quadratmeter ($\frac{lm}{m^2}$) oder Lux (lx) gemessen. $1\,\frac{lm}{m^2} = 1\,lx$.[7]

Bei Licht handelt es sich um eine Form der elektromagnetischen Strahlung, mit - je nach Definition - Wellenlängen zwischen 360 nm bis 400 nm und 760 nm bis 830 nm. Im physikalischen Sinne werden auch kürzere sowie längere Wellen mit einbezogen, wovon die Internationale Beleuchtungskommission[8] jedoch abrät.[9]

Die mit unserem Gerät vorgenommene Messung der Beleuchtungsstärke funktioniert durch einen Lichtsensor (siehe *Abbildung 1*). Dieser besteht aus einem Glasfilter, welcher die spektrale Empfindlichkeit an die Hellempfindlichkeitskurve anpasst, sowie einer Photodiode, welche einen zur Lichtmenge proportionalen elektrischen Strom erzeugt. Dieser wird in Lux umgerechnet.[10]

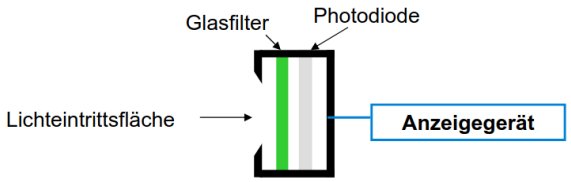

Abbildung 1: Schema eines Lichtsensors

Das finale Gerät soll über einen längeren Zeitraum die Beleuchtungsstärke messen und protokollieren, wobei durch mehrere Sensoren ein Bedienfehler erkannt und aus der Wertung genommen werden kann.

[7] Vgl. *Licht* (2025). de. `https://de.wikipedia.org/wiki/Licht`. Besucht: 12.01.2026.
[8] *International Commission on Illumination / Comission internationale de l'Eclairage / Internationale Beleuchtungskommission* (o. D.). en. `https://www.cie.co.at/`. Besucht: 12.01.2026.
[9] Vgl. *Licht* (2025). de. `https://de.wikipedia.org/wiki/Licht`. Besucht: 12.01.2026.
[10] Vgl. *Messung von Licht. Photometrie.* (o. D.). `https://www.ptb.de/cms/fileadmin/internet/fachabteilungen/abteilung_4/Messung_von_Licht_Photometrie.pdf`. Besucht: 12.01.2026, S. 12.

Des Weiteren soll das Gerät von dem vorhandenen Krankenhauspersonal leicht zu bedienen sein, was insbesondere auch das Erstellen einer leicht verständlichen Bedienungsanleitung fordert. Außerdem muss das Gerät unabhängig von anderen Geräten arbeiten, um den normalen Betriebsablauf nicht zu stören. Da auch andere Umweltfaktoren Einfluss auf das Schmerzempfinden haben könnten, sollen diese nach Möglichkeit mit aufgenommen werden.

3 Vorgehensweise, Materialien und Methoden

3.1 Materialien

Für die Steuerung der einzelnen Bauteile wird ein Mikrocontroller benötigt. Dieser aktualisiert das Display, spricht die einzelnen Sensoren an, erhält die Messwerte und speichert diese. Wir haben uns für einen Raspberry Pi Pico 2 (siehe *Abbildung 2*) entschieden, da dieser einfach zu programmieren ist und wenig Strom verbraucht, was für den Akkubetrieb wichtig ist.

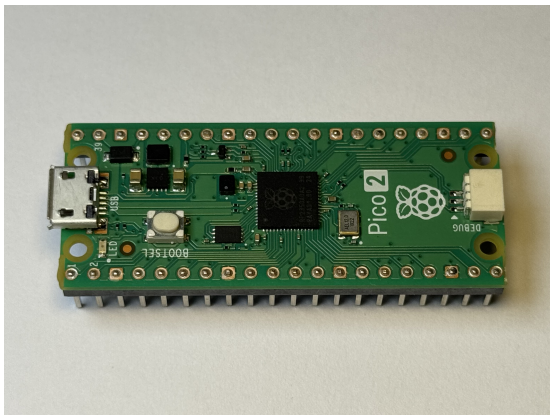

Abbildung 2: Raspberry Pi Pico 2

Zu den wichtigsten Teilen des Geräts gehören die Lichtsensoren. Es sollten drei davon in verschiedenen Winkeln eingebaut werden, damit man die Messungen von diesen vergleichen und so feststellen kann, ob der Raum gleichmäßig beleuchtet ist. Außerdem kann man so Bedienfehler, zum Beispiel wenn das Gerät an einer Wand steht, erkennen. Es werden Waveshare TSL2591 Umgebungslichtsensoren (siehe *Abbildung 3*) verwendet.

Um weitere Umweltfaktoren zu messen, wird ein Adafruit BME680 (siehe *Abbildung 4*) benutzt, weil dieser die Temperatur, die Luftfeuchtigkeit, den Luftdruck und flüchtige organische Verbindungen (Volatile Organic Compounds, kurz: VOC) messen kann. VOC sind gasförmige Stoffe in der Luft, die einen organischen Ursprung haben. Quellen für flüchtige organische Verbindungen sind zum Beispiel Möbel oder Reinigungsmittel.[11]

[11] Vgl. *Flüchtige organische Verbindungen* (Mai 2012). de. `https://www.umweltbundesamt.de/themen/gesundheit/umwelteinfluesse-auf-den-menschen/chemische-stoffe/fluechtige-organische-verbindungen`. Besucht: 08.01.2026.

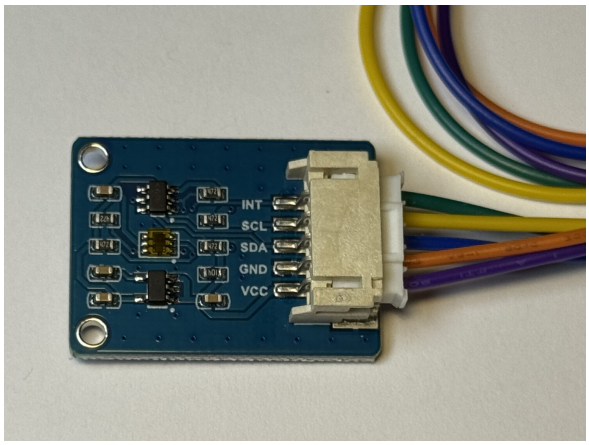

Abbildung 3: Waveshare TSL2591

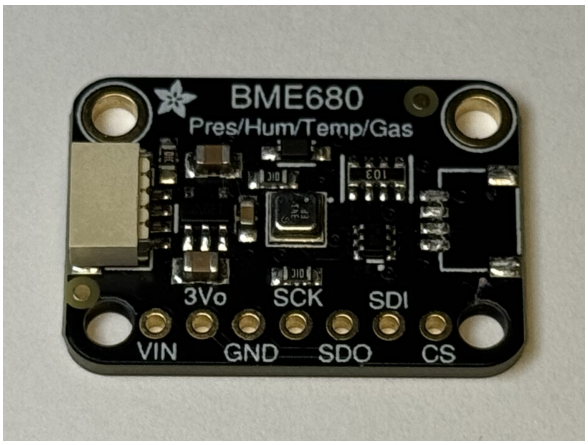

Abbildung 4: Adafruit BME680

Für das Display wurde als erstes ein alphanumerisches LCD (siehe *Abbildung 5*) ausprobiert. Dieses hat gut funktioniert, es benötigt aber 5 V Spannung und der Raspberry Pi Pico kann mit Akku nur 3,3 V liefern. Außerdem verbraucht es viel Strom, was für den Akkubetrieb nicht ideal ist. Deswegen sind wir auf ein Waveshare 2,9 Inch E-Paper-Modul (siehe *Abbildung 6*) mit 296x128 Pixeln umgestiegen, was des Weiteren den Vorteil der Lesbarkeit ohne Hintergrundbeleuchtung lieferte. Außerdem kann man mehr anzeigen, da es größer ist und man den Text überall hinsetzten kann. Dieses verbraucht nur sehr wenig Strom und läuft mit den gewünschten 3,3 V.

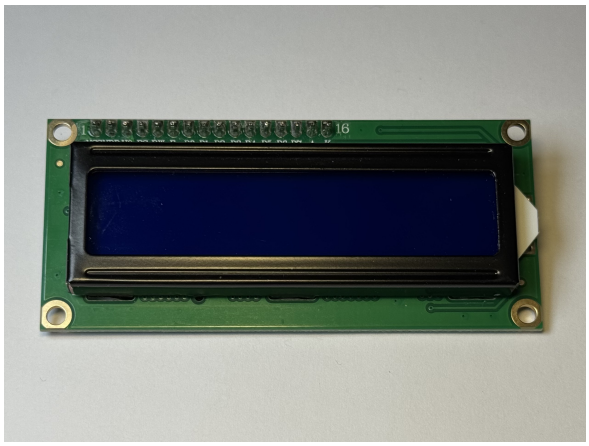

Abbildung 5: Alphanumerisches LCD

Abbildung 6: Waveshare E-Paper-Display

Wir haben uns dafür entschieden, einen Akku für die Stromversorgung zu benutzen, weil es unpraktisch wäre, wenn das Gerät immer ein Kabel braucht. Das Messgerät soll im Patientenzimmer einfach hingestellt werden können, ohne dass man es irgendwo anschließen muss. Als Akku wird ein Panasonic Lithium Ionen Akku (siehe *Abbildung 7*) mit 3450 mAh Kapazität und 3,6 V Nennspannung verwendet. Dieser wird in einem Batteriehalter von einem Lithium Battery Charger Module (siehe *Abbildung 8*) aufgeladen.

Der Raspberry Pi Pico verbraucht etwa 20 mA Strom[12]. Mit den angeschlossenen Modulen sind es geschätzt etwa 30 mA, vielleicht etwas weniger. Deswegen sollte der Akku etwa $\frac{3450\,mAh}{30\,mA} = 115\,h \approx 4$ Tage ohne Aufladen genutzt werden können. Das ist aber nur eine grobe Schätzung. Tatsächlich reicht der Akku im Dauerbetrieb länger (siehe *4 Ergebnisse*).

Um den Akkustand zu überwachen und anzuzeigen, wird ein Adafruit LC709203F Li-Poly / Li-Ion Spannungs- und Ladestandsensor (siehe *Abbildung 9*) verwendet.

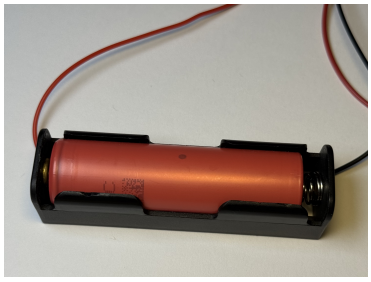

Abbildung 7: Li-Ion Akku

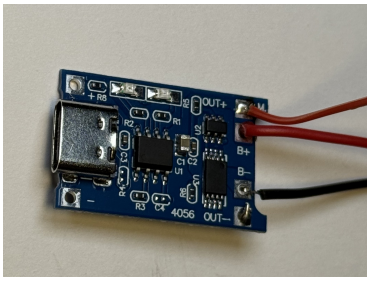

Abbildung 8: Ladegerät

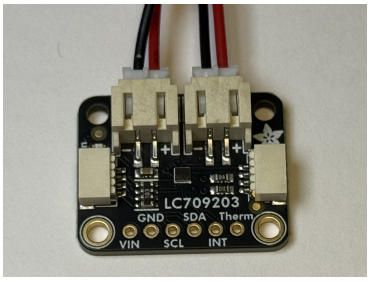

Abbildung 9: Akkustandsensor

Die gemessenen Daten sollen in einer CSV-Datei auf einer microSD-Karte gespeichert werden. Damit der Raspberry Pi Pico auf die microSD-Karte schreiben kann, wird ein microSD-Card Reader Modul (siehe *Abbildung 10*) verwendet. In der CSV-Datei stehen Datum, Uhrzeit und alle Messwerte in einer Zeile. Die Werte werden durch Kommas getrennt. Im Abstand von einer Minute werden neue Messwerte genommen und in eine neue Zeile geschrieben. Die generierte Datei kann man dann zum Beispiel in Excel öffnen.

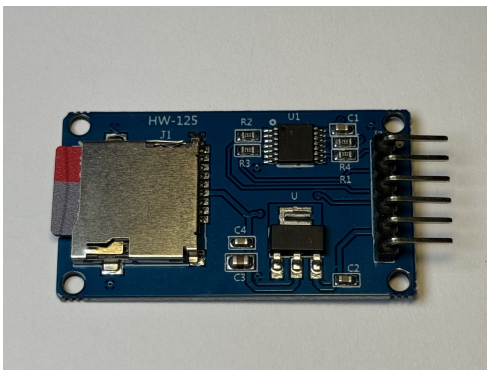

Abbildung 10: microSD-Kartenlesegerät

Damit alle Lichtsensoren zusammen an den Raspberry Pi Pico angeschlossen werden können, benötigt man einen I2C-Multiplexer. Die Lichtsensoren kommunizieren mit I2C über zwei Leitungen mit dem Pico und werden über Adressen angesprochen. Da dafür jedes Gerät eine andere Adresse benötigt,

[12]Vgl. *Raspberry Pi Pico: Stromverbrauch und Stromkosten* (o. D.). de. `https://www.elektronik-kompendium.de/sites/raspberry-pi/2707161.htm`. Besucht: 14.01.2026.

aber die Adressen der Lichtsensoren gleich sind, wird ein TCA9548A I2C-Multiplexer (siehe *Abbildung 11*) verwendet, der als Weiche dient und immer nur den Kanal zu dem Sensor öffnet, der gerade angesprochen werden soll.

Zum Bedienen des Geräts wurden zwei Taster (siehe *Abbildung 12*) eingebaut. Mit ihnen werden Messungen gestartet und gestoppt und man kann mit ihnen Messwerte ein- und ausblenden oder ins Menü gehen.

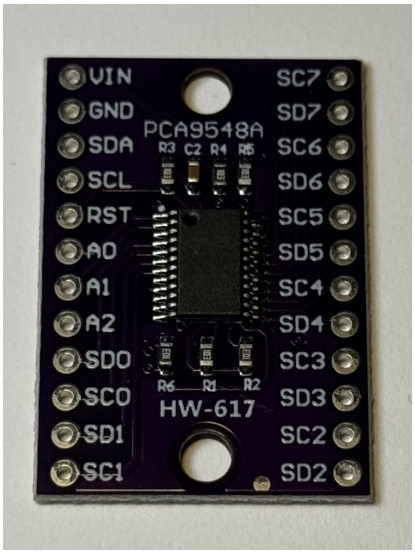

Abbildung 11: I2C-Multiplexer

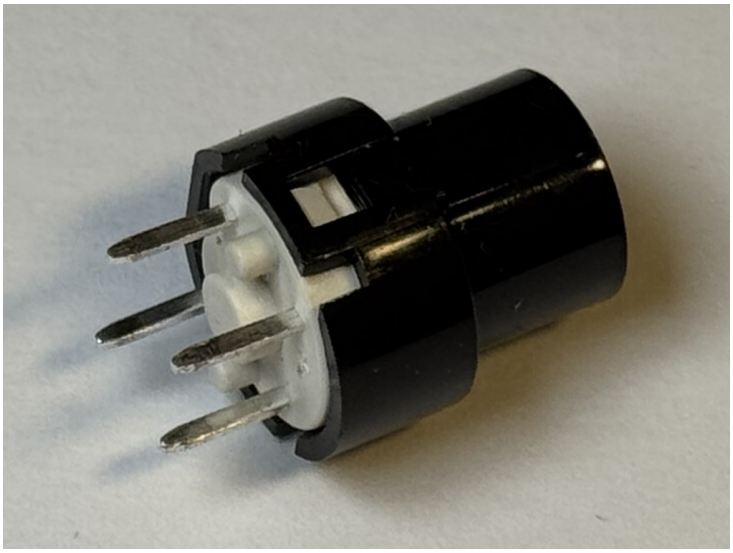

Abbildung 12: Taster

3.2 Programmierung

3.2.1 Bauteile

Auf dem Raspberry Pi Pico wurde Micropython installiert. Das ist eine Python-Version für Mikrocontroller. Um ihn zu programmieren, wurde Visual Studio Code mit der MicroPico Erweiterung benutzt. Diese ermöglicht das Hochladen von Dateien auf den Raspberry Pi Pico.

Um die Lichtsensoren auszulesen, wird eine Bibliothek für Micropython benötigt. Das ist eine separate Datei mit Funktionen, die man in dem eigentlichen ausgeführten Code aufruft. Mit den Funktionen kann man den Sensor ansprechen und einen Messwert erhalten. Das gesamte Licht und der Infrarotanteil des Lichts wird einzeln gemessen. Daraus wird dann die Beleuchtungsstärke in Lux berechnet.

Zunächst gab es Probleme, weil wir nur Python-Code zum Auslesen der Sensoren für einen richtigen Computer und nicht für Mikrocontroller gefunden haben. Dieser benutzt aber statt Micropython normales Python. Der Code konnte nicht ausgeführt werden, weil es dazwischen leichte Unterschiede gibt. Als wir das Problem festgestellt haben und eine Micropython-Bibliothek[13] gefunden haben,

[13] siehe: J. Fischer (2016). *micropython-tsl2591: Port of maxlklaxl/python-tsl2591for micropython-based devices like the ESP8266*. en. `https://github.com/jfischer/micropython-tsl2591`. Besucht: 08.01.2026.

hatte es zunächst immer noch nicht funktioniert, weil in dem Code standardmäßig andere I2C-Pins definiert waren. Nachdem wir das angepasst haben, hat es funktioniert.

Bei jeder Messung sollten je fünf Werte mit Abständen von 0,5 s von dem gesamten Licht und dem Infrarotanteil des Lichts genommen werden. Daraus wird bei beiden der Median benutzt, um die Beleuchtungsstärke zu berechnen. Die Mediane und die Beleuchtungsstärke werden dann in die CSV-Datei geschrieben. Es wird der Median genutzt, weil so durch zum Beispiel eine Hand, die sich kurz über dem Sensor befindet, möglichst keine Messfehler entstehen.

Außerdem kann man bei dem TSL2591 die Verstärkung (gain) und die Belichtungszeit (integrationtime) ändern. Die Verstärkung verstellt die Empfindlichkeit des Sensors. Für sie gibt es vier Möglichkeiten. Je höher die Verstärkung, desto niedrigere Beleuchtungsstärken, und je niedriger die Verstärkung, desto höhere Beleuchtungsstärken können gemessen werden. Bei der Belichtungszeit konnte keine Auswirkung auf die Ergebnisse festgestellt werden. Um für jede Helligkeit gute Messwerte zu bekommen, wird eine Messung mit Standardverstärkung gemacht, und je nach Wert dieser Messung wird die Verstärkung verstellt. Um herauszufinden, wie das eingestellt werden muss, haben wir bei verschiedenen Helligkeiten Werte mit allen Einstellungen genommen und danach beurteilt, was die besten Schwellenwerte sind.

Nach den Problemen bei dem TSL2591 haben wir zum Auslesen des BME680-Sensors direkt eine Micropython-Bibliothek[14] auf GitHub gefunden und benutzt. Dieser Sensor wird auch mit I2C verbunden. Mit der Bibliothek kann man einfach die Temperatur in °C, die Luftfeuchtigkeit in %, den Luftdruck in hPa oder den Gaswiederstand von flüchtigen organischen Verbindungen in Ω abrufen. Je höher der Anteil von VOC in der Luft ist, desto niedrigere Widerstände misst der Sensor. Je höher der Widerstand also ist, desto weniger VOC sind in der Luft, es gibt also eine bessere Luftqualität. Den Widerstandswert kann man dann zwischen den Messungen vergleichen, die Ohm sind aber nur der Messwert des Sensors und keine offizielle Einheit dafür. Auch hier wird wieder aus fünf Messungen der Median gespeichert.

Nach dem Umstieg auf das E-Paper-Display haben wir dafür der offizielle Micropython-Code von Waveshare[15] benutzt. Zu dieser Bibliothek wurden Funktionen hinzugefügt, die das Anzeigen von größerem Text und Symbolen ermöglichen. Das Display wird im Querformat benutzt. Man kann beliebigen Text an einer beliebigen Stelle einfügen. Im Gegensatz zu den meisten anderen Teilen wird das E-Paper-Display mit SPI gesteuert. SPI ist eine andere Möglichkeit, zwischen dem Mikrocontroller und einem anderen Gerät zu kommunizieren, benötigt aber mehr Kabel, unter anderem eines, über das die Bauteile unterschieden werden können (wie mit I2C-Adressen).

Damit man weiß, wann man das fertige Gerät aufladen muss, sollte der Akkustand angezeigt wer-

[14] siehe: R. Hammelrath (2024). *BME680-Micropython: Micropython driver for the BME680 sensor*. en. `https://github.com/robert-hh/BME680-Micropython`. Besucht: 08.01.2026.

[15] siehe: Wafeshare (2024). *Pico_ePaper_Code: Waveshrae Pico e-Paper driver code*. en. `https://github.com/waveshareteam/Pico_ePaper_Code`. Besucht: 08.01.2026.

den. Um den Akkustand herauszufinden, wird der Akkustandsensor mit einem Micropython-Code[16] verwendet. Er wird über I2C mit dem Pi verbunden.

Der microSD-Kartenleser wird auch über SPI angesprochen. Zum Schreiben und Lesen wird ebenfalls eine Micropython-Bibliothek[17] verwendet. Man kann dann eine Datei definieren, in die man dann Daten schreiben kann. Man kann aber auch Dateien von der microSD-Karte auslesen. Im fertigen Produkt sollen alle Daten in CSV-Dateien geschrieben werden.

Weil es viele Teile mit I2C gibt und die drei Lichtsensoren alle die gleiche Adresse haben, also nicht gleichzeitig an den Raspberry Pi Pico angeschlossen werden können, muss ein I2C-Multiplexer verwendet werden. Dieser benötigt keine Bibliothek, weil es nur einfache Befehle gibt, die einen oder mehrere Kanäle aktivieren.

Um dauerhaft auf ein Signal von einem Taster zu warten, wurden Interrupts eingerichtet. Diese führen, sobald ein Taster gedrückt ist, einen bestimmten Code aus. Das machen sie an einer beliebigen Stelle in dem Code. Dieser wird dabei unterbrochen, bis der Befehl von dem Taster ausgeführt wurde. Das unterscheidet Interrupts von Schleifen, die an nur einer Stelle im Code dauerhaft den Status des Tasters abrufen und nicht reagieren, wenn der Taster an einer anderen Stelle gedrückt wird.

Außer dem reinen Auslesen der Sensoren, dem Schreiben auf die microSD-Karte und der Aktualisierung des E-Paper-Displays haben wir alles andere selbst programmiert. Dazu gehören unter anderem die Auswertung von Tastendrücken zur Bedienung des Geräts, ein Menü (siehe *3.2.2 Menü*), eine Funktion zur korrekten Einstellung der Zeit sowie das Stromsparen (siehe *3.2.3 Strom und Zeit*). Außerdem müssen alle Sensoren mehrfach ausgelesen und die Werte in einer Datei gespeichert werden, wobei beachtet werden muss, dass die microSD-Karte nicht immer eingesteckt ist. Des Weiteren wird alles geloggt, um eine einfache Fehlerbehandlung (siehe *3.2.4 Fehlerbehandlung*) zu ermöglichen.

3.2.2 Menü

Damit man zum Beispiel das Messgerät neu starten, die Zeit mit einem Computer synchronisieren (siehe *3.2.3 Strom und Zeit*) oder Fehler behandeln (siehe *3.2.4 Fehlerbehandlung*) kann, haben wir ein Menü (siehe *Abbildung 13*) programmiert. In dieses gelangt man über einen langen Druck des rechten Tasters. Mit dem linken Taster kann man dann die Menüpunkte durchschalten und mit dem rechten bestätigen. Eine genauere Erklärung dazu gibt es in der Quick-Start-Anleitung.

[16] siehe: C. Reichl (2022). *MicroPython_LC709203F: MicroPython Library for I2C LC709203F battery status and fuel gauge.* en. `https://github.com/chris-reichl/MicroPython_LC709203F`. Besucht: 08.01.2026.

[17] siehe: B. Wynn (2025). *micropython-lib/micropython/drivers/storage/sdcard/sdcard.py at micropython-lib: Core Python libraries ported to MicroPython.* en. `https://github.com/micropython/micropython-lib/blob/master/micropython/drivers/storage/sdcard/sdcard.py`. Besucht: 08.01.2026.

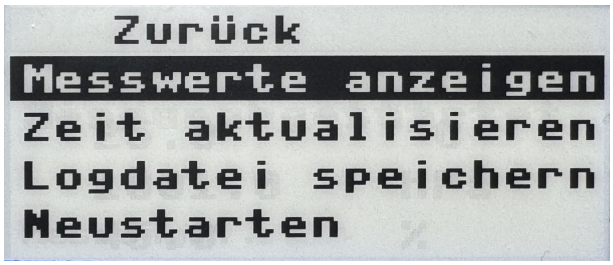

Abbildung 13: Menü

3.2.3 Strom und Zeit

Die Uhr des Raspberry Pi Pico 2 läuft nur, wenn er Strom bekommt. Ist der Akku leer, wird die Uhr zurückgesetzt. Damit der Akku nicht leer wird, wird der Akkustand so angezeigt, dass wenn auf dem Display 0 % steht, der Akku eigentlich noch 20 % hat. Der Code dazu ist in *Abbildung 14* zu sehen.

```
def battery():
    global measurement, battery_low, percent, last_percent
    battery_low = False
    i2c.writeto(0x70, b'\x10')
    percent = battery_sensor.cell_percent
    if round(percent) != last_percent:
        last_percent = round(percent)
        write_log("battery", percent)
    if percent >= 20:
        percent = 1.25 * percent - 25
    else:
        print("Akku leer")
        percent = 0
        measurement = False
        battery_low = True
        epd_BI(percent, battery_low, f"{t[2]:02d}.{t[1]:02d}.", f"{t[4]:02d}:{t[5]:02d}")
    print(percent)
    return percent
```

Abbildung 14: Akkustand auswerten

Die Schätzung, wie lange der Akku das Gerät mit Strom versorgen kann, waren 4 Tage. Dabei wurde aber mit der gesamte Akkukapazität von 3450 mAh gerechnet. Bei einem Test wurde der Code aus *Abbildung 14* benutzt. Dabei reichte der Akku 4 Tage lang. Damit der Akku noch länger hält, wird der Pi Pico zwischen den Messungen und wenn keine Messung läuft in einen Stromsparmodus geschaltet. Dadurch sollte das Gerät deutlich länger messen können.

Wenn sich der Akku trotzdem entleert, gibt es in der Quick-Start-Anleitung (Schnellstartanleitung zum Bedienen) eine Erklärung, wie man die Zeit des Messgeräts mit einem Computer synchronisiert. Dafür wird im Menü der Punkt **Zeit aktualisieren** ausgewählt, das Gerät an den Computer angeschlossen und ein Programm namens TimeSync.exe (siehe *Abbildung 15*) gestartet.

Abbildung 15: TimeSync.exe

3.2.4 Fehlerbehandlung

In einer Logdatei auf dem Mikrocontroller werden Tastendrücke, der Akkustand und Fehler mitgeschrieben. Unkritische Fehler werden im Code über try-except-Blöcke erkannt und abgefangen. Die Fehlermeldung wird dann mit Datum, Uhrzeit und Zeilenangabe gespeichert. Gibt es beim Benutzen des Geräts ein Problem, kann man die Logdatei über das Menü auf die microSD-Karte übertragen. Mit der Logdatei kann der Fehler eventuell erkannt und behoben werden.

3.3 Verbinden aller Teile

Alle I2C-Geräte, also die Lichtsensoren, der BME-Sensor und der Akkustandsensor, wurden mit dem I2C-Multiplexer verbunden. Dieser wurde über I2C an den Raspberry Pi Pico 2 angeschlossen. Die SPI-Geräte, also das E-Paper-Display und der microSD-Kartenleser wurden einzeln mit dem Pi verbunden. Die Taster wurden mit je zwei Kabeln direkt an den Pi angeschlossen.

Von dem Ladestandsensor gehen zwei Kabel (Plus und Minus) an das Ladegerät und weitere zwei Kabel an den Plus- und Minuspol von dem Akku. Von dem Ladegerät geht ein weiteres Kabel zu dem Raspberry Pi, um ihn mit Strom zu versorgen (siehe *Abbildung 16*).

Alle Verbundenen Bauteile ohne Gehäuse sieht man in *Abbildung 17*. Eine Übersicht der Verkabelung gibt es in *Abbildung 18*. Dabei wurden die beiden I2C Kabel, alle SPI Kabel und die beiden Stromleitungen (+ und -) zu je einer Linie zusammengefasst.

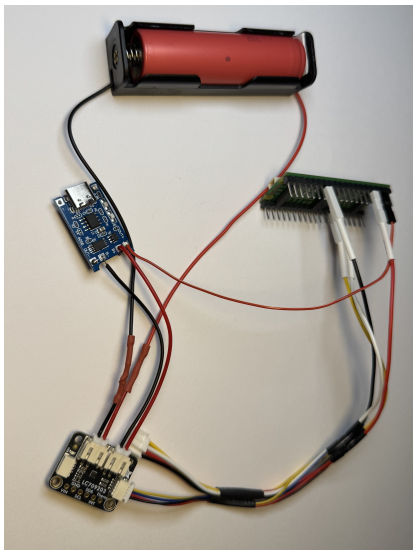

Abbildung 16: Verkabelung der Stromversorgung

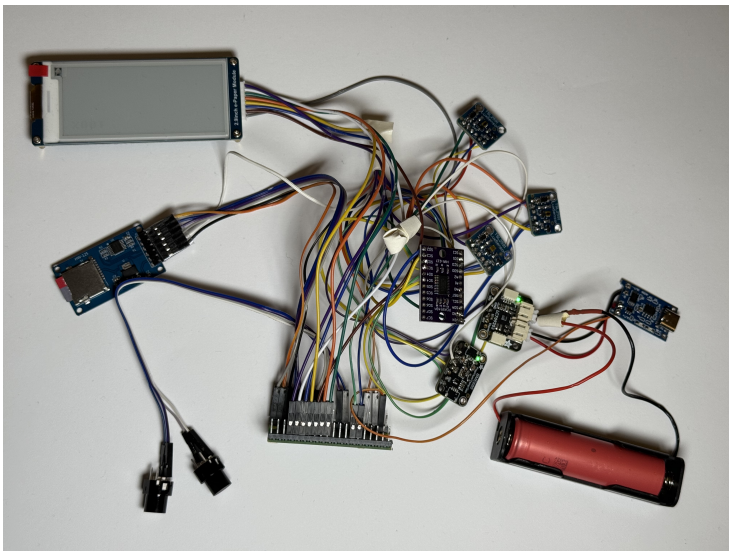

Abbildung 17: Verkabelung des Geräts

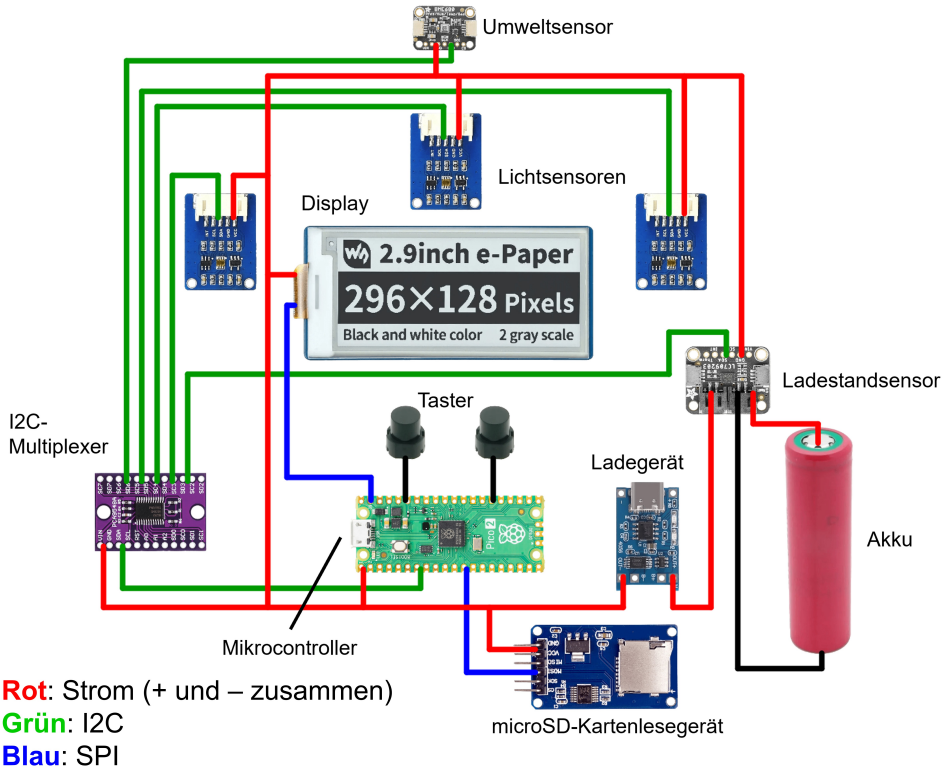

Abbildung 18: Vereinfachter Schaltplan

Nach der Verkabelung haben wir den Code fertig geschrieben, der ununterbrochen laufen soll, Messungen aufnimmt und diese speichert (siehe Ergebnisse). Alle Bibliotheken wurden als einzelne Dateien auf den Raspberry Pi Pico 2 hochgeladen. Der auszuführende Code wurde als *main.py* gespeichert.

main.py ist die Micropython-Datei, die bei Stromanschluss automatisch gestartet wird.

3.4 Gehäuse

Das Gehäuse sollte 3D-gedruckt werden, weil das einfach ist, schnell geht und so ein Gehäuse erstellt werden kann, in das alle Teile perfekt passen. Das 3D-Modell dafür haben wir mit FreeCAD erstellt (siehe *Abbildung 19*). In einen hohlen Quader von etwa $100\,mm \cdot 70\,mm \cdot 80\,mm$ wurden als erstes Löcher für das Display, die Taster, die microSD Karte, den Micro-USB Anschluss von dem Pi Pico und den USB-C Anschluss zum Aufladen gemacht. Außerdem wurden oben links und rechts die Kanten in einem 45° Winkel abgeschrägt. Zwei der Lichtsensoren sollen sich auf jeweils einer der dadurch entstandenen Flächen befinden, der dritte oben. Das sorgt dafür, dass sich die Messungen zum Teil überlappen, aber trotzdem in verschiedene Richtungen zeigen. Dadurch kann man beurteilen, ob ein Raum gleichmäßig beleuchtet ist oder ob das Gerät an einer Wand steht. Für die Sensoren gibt es kleine Löcher, damit sie nach draußen messen können, die Platinen aber in dem Gehäuse befestigt sind. Der Temperatur-, Luftfeuchtigkeit-, Luftdruck- und VOC-Sensor hat ebenfalls ein kleines Loch auf der Oberseite. Das Gehäuse sollte robust sein. Dafür hat es eine einfache Form und ist stabil. Außerdem befinden sich alle Teile, außer den Tastern, im Gehäuse oder schauen nur zum Teil heraus. Dadurch sollen Beschädigungen vermieden werden.

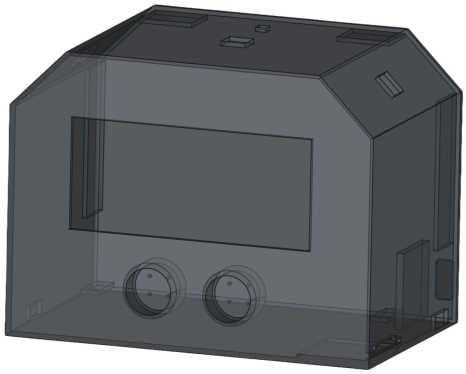

Abbildung 19: 3D-Modell des Gehäuses

Die hintere Wand haben wir separat gedruckt. So kann man die Elektronik einbauen und dann das Gehäuse verschließen.
Weil die Platinen der Lichtsensoren einen großen Steckplatz haben und dieser verhindert, dass die Lichtsensoren aus dem Gehäuse schauen können, während alles andere in dem Gehäuse ist, haben wir den Steckplatz entfernt und die Kabel einzeln angelötet.

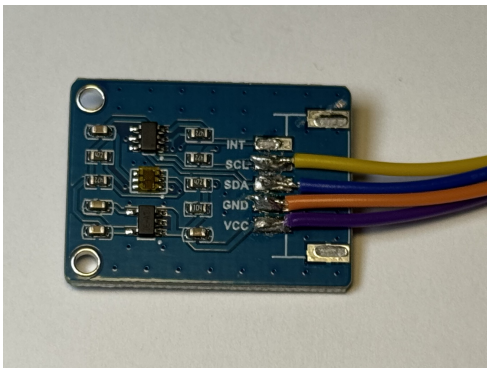

Abbildung 20: TSL2591 mit angelöteten Kabeln

Nach dem ersten Druck auf einem Anycubic Kobra 2 Pro wurden die Teile in dem Gehäuse befestigt. Nach ein paar Verbesserungen an dem 3D-Modell entstand das Gerät, welches in *Abbildung 21* zu sehen ist.

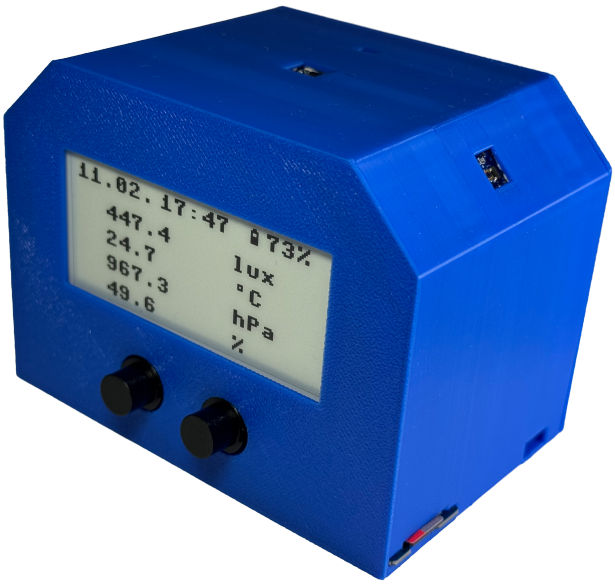

Abbildung 21: Fertiges Gerät

4 Ergebnisse

Das Gerät wird über einen Akku mit Strom versorgt. Wenn der linke Taster gedrückt wird, startet eine Messung. Es werden jede Minute Werte von den Sensoren erfasst. Man erhält die Beleuchtungsstärke von drei Sensoren, die Temperatur, die Luftfeuchtigkeit, den Luftdruck und den Anteil von VOC in der Luft. Diese Werte werden zusammen mit dem Datum und der Uhrzeit in einer CSV-Datei auf einer microSD-Karte gespeichert. Ist keine Speicherkarte eingesteckt, werden die Daten auf dem Raspberry

Pi Pico gespeichert. Sobald wieder eine microSD-Karte eingesteckt ist, werden die Daten auf diese übertragen. Das Display zeigt den Akkustand an. Außerdem kann man über den rechten Taster die Messwerte auf dem E-Paper Display anzeigen lassen. Ein erneuter Druck des linken Tasters stoppt die Messung. Über einen langen Druck des rechten Tasters kommt man in das Menü. Mit diesem kann man die Zeit mit einem Computer synchronisieren oder Fehler behandeln.

Der Pi Pico schreibt den Akkustand, Tastendrücke und Fehler, die der Code erkennt, in eine weitere CSV-Datei auf dem Pi. Das hilft bei der Fehlersuche, falls zum Beispiel die Messwerte komplett falsch sind oder ein Bauteil nicht mehr funktioniert.

Das Gerät ist einfach zu handhaben. Durch die Taster ist es leicht zu bedienen. Man kann eine microSD-Karte hineinstecken und diese auch während der Messung entfernen. Die Datei kann man dann zum Beispiel in Excel öffnen und auswerten. Wenn das Display anzeigt, dass der Akku fast leer ist, kann man ihn mit einem USB-C Kabel aufladen. Wenn er voll ist, wird das zusätzlich durch ein Licht von dem Ladegerät an der Seite signalisiert.

Damit das Messgerät von dem Krankenhauspersonal leicht bedient werden kann, gibt es eine Quick-Start-Anleitung.

Den Micropython-Code (Stand: 08.03.2026), die Quick-Start-Anleitung und eine CSV-Beispieldatei kann man auf dieser Website ansehen:
`https://cloud.uni-jena.de/s/ZGCNm3XsLmk3Tdk`

5 Ergebnisdiskussion

Das Gerät entspricht allen von der QUIPS-Gruppe angegebenen Anforderungen. Es kann über einen längeren Zeitraum die Beleuchtungsstärke messen, da es einen ausreichend großen Akku hat und auch während der Messung aufgeladen werden kann. Es misst jede Minute, was zu kontinuierlichen und genauen Messreihen führt.

Das Gerät eignet sich sehr gut für den Einsatz in Kliniken, da es klein, geräuschlos und akkubetrieben ist. Die einfache zwei-Taster-Bedienung ist, auch dank der Anleitung, schnell zu lernen und nicht überfordernd. Das Ziel, dass das Gerät im Klinikalltag kaum Mehraufwand bringt, für die Pflegenden leicht zu bedienen ist und trotzdem verwertbare Ergebnisse liefert, wurde auf jeden Fall erreicht.

Außer der Beleuchtungsstärke kann es auch einige weitere Umweltfaktoren messen. Diese sind Temperatur, Luftfeuchtigkeit, Luftdruck und flüchtige organische Verbindungen. Die Messwerte, besonders von der Beleuchtungsstärke, sind keine exakten Werte, aber sie sind unter den gleichen Messgeräten vergleichbar. Das reicht, um den Einfluss dieser Umweltfaktoren auf das postoperative Schmerzempfinden von Patienten zu erforschen. Die VOC werden in einem nur vergleichbaren Wert gemessen, der keine benutzbare Einheit hat.

Ein Problem ist, dass der Akku nicht leer werden darf, weil sonst die Uhr des Raspberry Pi Pico 2 unterbrochen wird, und, sobald er wieder Strom bekommt, von einem definierten Startpunkt wieder von vorne beginnt, die Zeit zu erfassen. Das sorgt natürlich für falsche Daten und Uhrzeiten. Um solche Fehler in der Auswertung zu erkennen, wird unter anderem der Verlauf des Akkustands gespeichert. Damit der Akku immer rechtzeitig aufgeladen wird, zeigt das Display schon bei 20 % an, dass er leer ist. Außerdem gibt es eine Möglichkeit, die Uhrzeit mit einem Computer zu synchronisieren.

6 Fazit und Ausblick

Am Anfang ging viel Zeit verloren, weil wir noch fast kein Wissen zu der Programmierung von Mikrocontrollern hatten. Das hat sich schnell gebessert. Dadurch konnte die erste, gut funktionierende Version des Geräts entwickelt werden. Diese erfüllt alle Anforderungen.

Sobald die Serienreife sichergestellt, ist wird auch der Einsatz des Geräts in einer Studie der QUIPS-Gruppe starten. Hierbei werden die Daten dezentral gesammelt und durch die Wissenschaftler der QUIPS-Gruppe ausgewertet. Für den Fall, dass eine Korrelation der Umweltfaktoren mit dem post-operativen Schmerzempfinden der Patienten festgestellt wird, kann diese Information genutzt werden, um zukünftigen Patienten den Aufenthalt angenehmer zu gestalten.

7 Quellen- und Literaturverzeichnis

Fischer, J. (2016). *micropython-tsl2591: Port of maxlklaxl/python-tsl2591for micropython-based devices like the ESP8266*. en. https://github.com/jfischer/micropython-tsl2591. Besucht: 08.01.2026.

Flüchtige organische Verbindungen (Mai 2012). de. https://www.umweltbundesamt.de/themen/gesundheit/umwelteinfluesse-auf-den-menschen/chemische-stoffe/fluechtige-organische-verbindungen. Besucht: 08.01.2026.

Hammelrath, R. (2024). *BME680-Micropython: Micropython driver for the BME680 sensor*. en. https://github.com/robert-hh/BME680-Micropython. Besucht: 08.01.2026.

International Commission on Illumination / Comission internationale de l'Eclairage / Internationale Beleuchtungskommission (o. D.). en. https://www.cie.co.at/. Besucht: 12.01.2026.

Kvandová M, Rajlic S, Stamm P, Schmal I, Mihaliková D, Kuntic M, Jimenez MTB, Hahad O, Kollárová M, Ubbens H, Strohm L, Frenis K, Duerr GD, Foretz M, Viollet B, Ruan Y, Jiang S, Tang Q, Kleinert H, Rapp S, Gericke A, Schulz E, Oelze M, Keaney Jr. JF, Daiber A, Kröller-Schön S, Jansen T, and Münzel T (2023). "Mitigation of aircraft noise-induced vascular dysfunction and oxidative stress by exercise, fasting, and pharmacological a1AMPK activation: molecular proof of a protective key role of endothelial a1AMPK against environmental noise exposure". In: *European Journal of Preventive Cardiology (2023) 30, 1554–1568* 30. DOI: {10.19224/ai2025.013}.

Licht (2025). de. https://de.wikipedia.org/wiki/Licht. Besucht: 12.01.2026.

Messung von Licht. Photometrie. (o. D.). https://www.ptb.de/cms/fileadmin/internet/fachabteilungen/abteilung_4/Messung_von_Licht_Photometrie.pdf. Besucht: 12.01.2026.

QUIPS (o. D.). de. https://www.quips-projekt.de/. Besucht: 12.01.2026.

Raspberry Pi Pico: Stromverbrauch und Stromkosten (o. D.). de. https://www.elektronik-kompendium.de/sites/raspberry-pi/2707161.htm. Besucht: 14.01.2026.

Reichl, C. (2022). *MicroPython_LC709203F: MicroPython Library for I2C LC709203F battery status and fuel gauge*. en. https://github.com/chris-reichl/MicroPython_LC709203F. Besucht: 08.01.2026.

Scheller JS, Komann M, Weinmann C, Weinmann J, Heitfeld S, Mielke Aet al: (2025). "Auf der Sonnenseite des Lebens – Die Auswirkung der Umgebungslichtintensität auf postoperatives Schmerzempfinden." In: *Anästh Intensivme 2025;66:13–19*. DOI: {10.19224/ai2025.013}.

Wafeshare (2024). *Pico_ePaper_Code: Waveshrae Pico e-Paper driver code*. en. https://github.com/waveshareteam/Pico_ePaper_Code. Besucht: 08.01.2026.

Wynn, B. (2025). *micropython-lib/micropython/drivers/storage/sdcard/sdcard.py at micropython-lib: Core Python libraries ported to MicroPython*. en. https://github.com/micropython/micropython-lib/blob/master/micropython/drivers/storage/sdcard/sdcard.py. Besucht: 08.01.2026.

Danksagung

Wir danken Christina Walther und Bernd Linhart für die tatkräftige Unterstützung und die Geduld sowie dem Schülerforschungszentrum Jena für das Bereitstellen der für dieses Projekt notwendigen Materialien und Räumlichkeiten. Des Weiteren danken wir Herrn Dr. Marcus Komann und der gesamten QUIPS-Gruppe für die Möglichkeit, dieses Gerät zu bauen.

Unterstützungsleistungen

Christina Walther, Projektleiterin, Schülerforschungszentrum Jena, hat uns Geräte, Materialien und Bauteile bereitgestellt und die schriftliche Arbeit korrekturgelesen.

Bernd Linhart, Ehrenamtlicher AG-Leiter, witelo e.V. hat uns inhaltlich beraten und die schriftliche Arbeit korrekturgelesen.

Unsere Eltern haben die schriftliche Arbeit korrekturgelesen.