

Eintwicklung eines CO₂-Messers

Mikrocomputertechnik - Bericht

Studiengang Elektrotechnik

Studienrichtung Fahrzeugelektronik

Duale Hochschule Baden-Württemberg Ravensburg, Campus Friedrichshafen

von

Alexander Herrmann Johannes Ruffer Serkant Soylu

Abgabedatum:	19.04.2020
Bearbeitungszeitraum:	01.10.2019 - 19.04.2020
Matrikelnummer:	9859538 x 1011921 x 9964027
Kurs:	TFE18-2
Gutachter der Dualen Hochschule:	Hans Jürgen Herpel

Eidesstattliche Erklärung

Gemäß Ziffer 1.1.13 der Anlage 1 zu §§ 3, 4 und 5 der Studien- und Prüfungsordnung für die Bachelorstudiengänge im Studienbereich Technik der Dualen Hochschule Baden-Württemberg vom 29.09.2015.

Wir versichern hiermit, dass wir unsere Projektarbeit mit dem Thema:

Entwicklung eines CO₂-Messers

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben. Wir versichern zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Friedrichshafen, den 10. März 2020

Alexander Herrmann

Johannes Ruffer

Serkant Soylu

AUTOREN

Kurzfassung

Der Bericht wurde von drei Studierenden an der Duale Hochschule Baden-Württemberg (DHBW)-Ravensburg Campus Friedrichshafen im Rahmen der Mikrocomputertechnik Vorlesung eigenständig von der Projektplanung, über die Durchführung, bis hin zum Projektabschluss mit Dokumentation durchgeführt.

Ziel der Arbeit ist es, durch die Praxiserfahrung mit Mikrocomputern, Fähigkeiten und Wissen in diesem Bereich zu erwerben. Durch das Vergleichen von anfänglichen Kosten- und Komplexitätsschätzungen mit den späteren Ergebnissen in der Umsetzung können die Studierenden ein Fazit ziehen, inwiefern diese übereinstimmen. So werden auch Fähigkeiten im Bereich des Projektmanagements weiterentwickelt.

Inhaltsverzeichnis

1. Einleitung	1
2. Anforderungen	3
3. Kosten und Arbeitsplan	5
4. Entwurf	7
4.1. Schaltungslayout	7
4.2. Gehäuse	8
5. Softwareimplementation	9
5.1. CCS811	10
5.2. Mikro-SD	12
6. Hardware	13
7. Testing	15
8. Handbuch	17
9. Installationsanleitung	19
10. Fazit	21
Verzeichnis verwendeter Abkürzungen und Formelzeichen	23
Literaturverzeichnis	25
Sachwortverzeichnis	26
Abbildungsverzeichnis	27
Tabellenverzeichnis	29

A. Anhang	31
A.1. Weitere Abbildungen	31

1. Einleitung

Mithilfe eines Arduinos wurde in diesem Projekt ein CO₂-gesteuerter Fensterheber simuliert, welcher dazu dienen soll die Räumlichkeiten bei schlechter Luftqualität automatisch zu lüften. Auch das automatische schließen des Fensters nach Wiederherstellung von guter Luftgüte wird simuliert.

Das Öffnen und Schließen der Fenster wird anhand von einer LED simuliert, welche nach der Überschreitung eines bestimmten Grenzwertes aufleuchtet. Sobald nach ausreichendem Lüften die Luftgüte unter einen Schwellwert gerät, soll die LED wieder aus gehen.

Damit dies möglich ist, wurde die Entwicklung von Software-Code, sowie ein Schaltungslayout und der 3D-Druck des Gehäuses selbständig vorgenommen.

Der folgende Bericht dokumentiert die Vorgehensweisen und Umsetzung von der Projektplanung, über die Implementierung von Hardware und Software, bis hin zu durchgeführten Tests und den Projektabschluss.

2. Anforderungen

Damit die Bewertung des Projektes erfolgreich wird, müssen zu Projektbeginn Anforderungen erarbeitet und festgelegt werden. Diese sind unveränderbar, da das Ergebnis sonst verfälschen würde.

Nummer	Anforderungen	Verifikationsmethode
1	Echtzeitmessung der Luftgüte	Measurement
2	Mindestmessbereich von 400 ppm bis 5000 ppm	Review
3	Visualisierung der Luftgüte mithilfe von LEDs (gut, mittel, schlecht)	Test
4	Ausgabe der Luftgüte mithilfe von LiquidCrystal Display (LCD)-Display	Test
5	Ansteuern eines Fensterscheibenmotors mithilfe einer LED simulieren	Test
6	Bei schlechter Luftgüte: Fenster öffnet sich (LED an)	Test
7	Bei guter Luftgüte: Fenster schließt sich (LED aus)	Test
8	Speichern im CSV-Format	Test, Analysis
9	Zugriff auf Messdaten über SD-Karte	Test
10	Benutzer kann zwischen drei Messprofilen auswählen (Messprofil: Abtastrate)	Test

Tabelle 2.1.: Anforderungen an das Projekt

Eine Anforderung muss messbar und/oder überprüfbar sein, um deren Umsetzung später objektiv bewerten zu können. Somit wird auch eine sogenannte Verifikationsmethode festgelegt, mit der die Anforderungen später überprüft werden kann.

Es gibt sechs verschiedene Verifikationsmethoden:

- Similarity: Suche und Abgleich mit bereits vorhandenen Lösungen [Hel, S. 122]

- Insparity: Soll- und Ist-Abgleich von Ein- und Ausgängen nach einem formalen Ablauf (Ein- und Ausgangskriterien sind vorher definiert) [Pet09, vgl. S. 308]
- Review: Soll- und Ist-Abgleich von Ein- und Ausgängen ohne formalen Ablauf (Ein- und Ausgangskriterien sind vorher nicht definiert) [Pet09, vgl. S. 317]
- Measurement: Durchführung einer Reihe von Operationen, die das Ziel haben, einen Wert einer quantitativen oder kategorialen Darstellung eines oder mehrerer Attribute zu bestimmen [Dep, vgl. S. 395]
- Analysis: Basiert auf Heuristiken und Statistiken [...] [, die man] sich als starke Compiler-Typisierung [...] im Rahmen einer ausführlichen Datenfluss-Analyse vorstellen [kann] [Jay, vgl. S. 4]
- Test: Prüft und bewertet Software auf Erfüllung der für ihren Einsatz definierten Anforderungen und misst ihre Qualität [Wik20]

3. Kosten und Arbeitsplan

Die Kosten des Projektes belaufen sich, wie in 3.1 nachzuvollziehen ist, auf 73,14€. An dieser Stelle muss jedoch angefügt werden, dass es sich hierbei nicht um die reinen Materialkosten für das Projekt handelt, da beispielsweise das Arduinoset einiges mehr, als nur die benötigten Materialien bietet. Somit kann man sagen, dass sich die Kosten bei einer Bestellung mit Einschränkung auf die benötigten Komponenten senken würden.

Einkauf	Kosten
Arduinoset	27,95€
CO2-Sensor (CCS811)	8,50€
MikroSD-Modul	3,50€
LED-Fassung	4,59€
Mechanische Bauteile	24,99€
Sonstiges	4,56€
Gesamt	73,14€

Tabelle 3.1.: Aufgewendete Kosten für das Projekt

Damit wir zu jedem Zeitpunkt einschätzen können, ob wir im Zeitplan sind, haben wir zu Beginn des Jahres den in Abbildung 3.1 zu sehenden Projektplan erstellt. An diesen Ablauf inklusive den Daten, hat sich jedes Gruppenmitglied zu halten. Eine Verzögerung in der Entwicklung, bedeutet eine Verschiebung der Abwicklung des Projektes. Davon hängt somit die Fertigstellung, als auch der Erfolg des Projektes ab.

Auch, wenn die Abgabefrist erst vier Wochen nach Semesterende abläuft, haben wir uns, wie in Abbildung 3.1 zu sehen ist, den 12. April als Meilenstein für die Abgabe des Projektes gesetzt. Die Begründung für diese Entscheidung liegt zum einen darin, dass wir bei Problemen bezüglich der Abgabe einen gewissen Puffer haben. Zum anderen beginnt ab Anfang April die dritte Praxisphase der dualen Studierenden, in welcher sie ab Mitte des Monats schon in ihrem neuen Projekt vertieft sind und somit weniger Zeit für das Mikrocomputertechnik-Projekt aufwenden können.

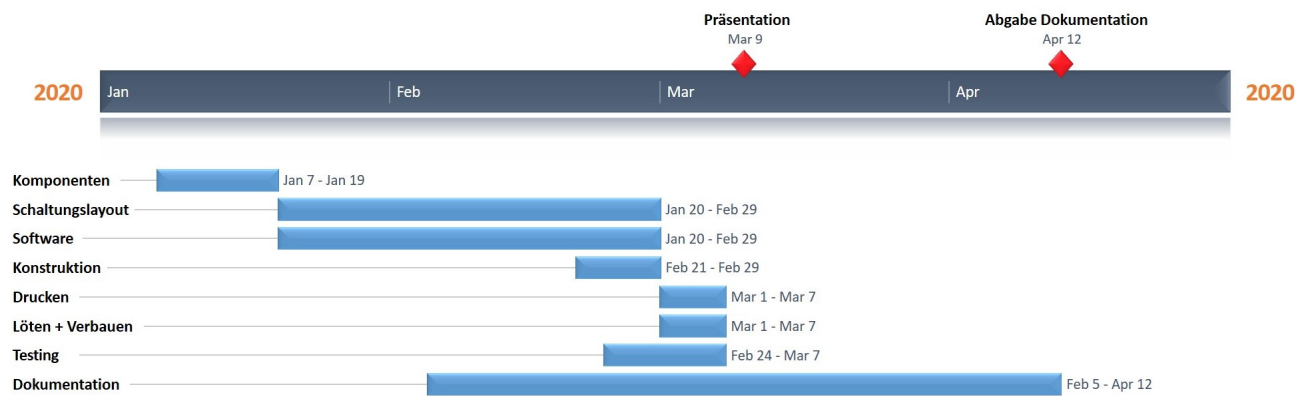


Abbildung 3.1.: Ablauf des Projektes von Januar bis April

4. Entwurf

Bevor es an die Umsetzung von Soft- und Hardware ging, wurden für die verschiedenen Gebiete Zeichnungen und Diagramme angefertigt. Zum einen diente es dazu, das Vergessen von Anforderungen zu vermeiden. Zum anderen konnten die bevor stehenden Aufgaben so besser verteilt werden. Auch die Vorstellungen der einzelnen Gruppenmitgliedern in Bezug auf die Umsetzung des Projektes wurden mit dieser Methode zusammengeführt, um spätere Differenzen zu vermeiden.

4.1. Schaltungslayout

In Abbildung 4.1 ist das Schaltungslayout des Projektes zu sehen, welches wir mithilfe von Fritzing angefertigt haben.

Das LCD-Display ist an sechs Pins mit dem Arduino verbunden. Zudem benötigt das LCD eine Versorgungsspannung von ca. 5 Volt. Der Eingang V0 ist dabei über einen Widerstand auf Masse geschaltet. Je nach Höhe des Widerstandes ändert sich der Kontrast des LCD-Displays. Darunter befindet sich in Abbildung 4.1 der CO₂-Sensor CCS811, welcher über zwei Pins an den Arduino angeschlossen ist. Einen extra Anschluss an die Versorgungsspannung benötigt dieser nicht, da er die maximal benötigten 3.6 Volt über den I²C-Anschluss bekommt.

Unterhalb des Co₂-Sensors ist das SD-Karten-Modul zu sehen, welches eine Versorgungsspannung von 5 Volt auf Masse benötigt. Die restlichen vier Anschlüsse sind mit den Pins 8 bis 11 am Arduino verbunden.

Zwei von den oben zu sehenden Taster dienen für die Menüauswahl auf dem LCD-Display. Einer soll dabei für das Wechseln innerhalb des Menüs dienen, der andere ist für die Bestätigung der Auswahl zuständig. Damit die Schaltung auch ein- und ausgeschaltet werden kann, ohne die Versorgungsspannung zu kappen, haben wir einen dritten Taster eingebaut. Die Vorwiderstände aller Taster belaufen sich auf 2k Ω .

Die vier LEDs sind, wie in den Anforderungen definiert, für die Visualisierung der Luftgüte

zuständig. Dabei dienen die Farben Grün, Gelb und Rot für gute, mittlere und schlechte Luftgüte. Die blaue simuliert den angeschlossenen Fensterscheibenmotor. Für die Vorwiderstände der vier LEDs haben wir jeweils 330Ω gewählt.

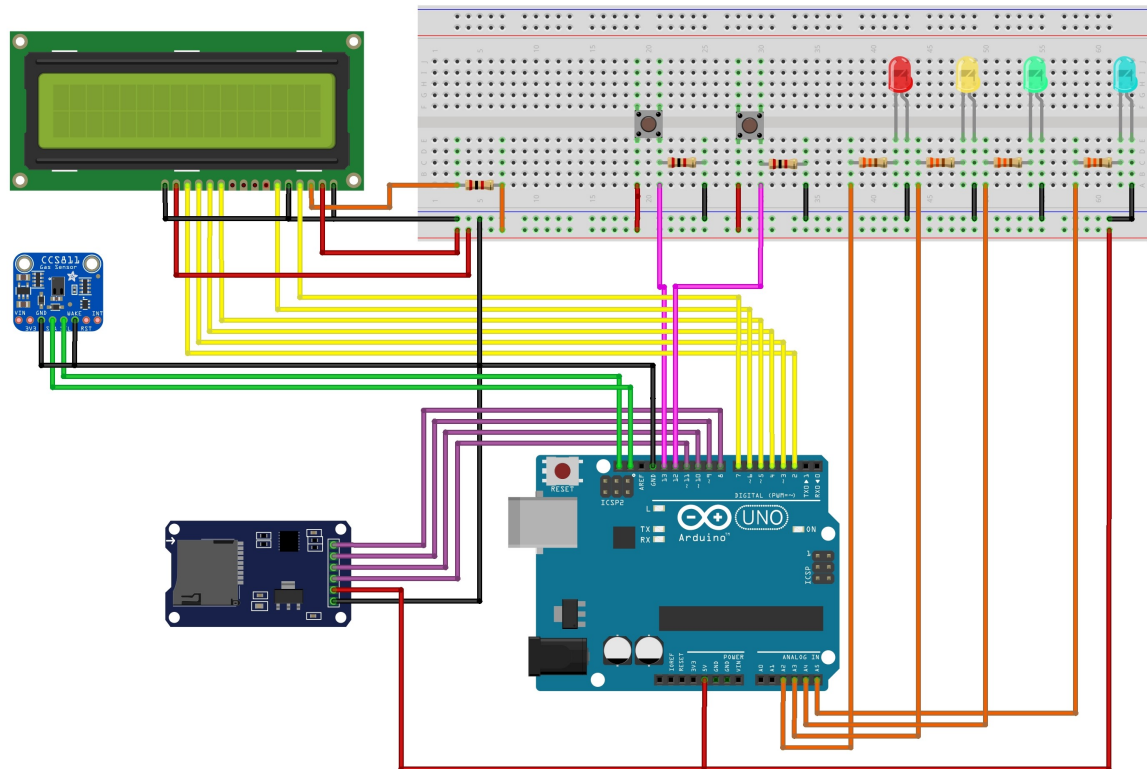


Abbildung 4.1.: Schaltungslayout vom 27.02.2020

4.2. Gehäuse

5. Softwareimplementation

Für die entgültige Struktur der Software waren mehrere Anforderungen ausschlaggebend.

Zunächst wurden drei verschiedene Messprofile definiert und implementiert, sodass der Benutzer zu Beginn zwischen einer Echtzeit-, Stunden- und Tagesmessung wählen kann. In den Messprofilen ist definiert, in welchen Abständen und wie lange Messungen durchgeführt werden sollen. Somit konnten Anforderung Nummer 1 und 10 aus der Tabelle 2.1 gemeinsam umgesetzt werden.

Nach der Wahl des Messprofils muss der Arduino den CO₂-Sensor ansteuern und richtig konfigurieren. Es muss getestet werden, ob er funktionstüchtig und bereit ist eine Messung zu starten. Zudem kann der verwendete Sensor nicht nur CO₂-Werte, sondern beispielsweise auch Temperaturen messen, sodass der Arduino die richtigen Werte anfordern muss. Damit dies möglich ist, mussten wir die Bibliothek `<Adafruit_CCS811.h>` einbinden. Genauerer zu diesem Algorithmus wird in Kapitel 5.1 erläutert.

Zunächst wird im Setup geprüft, ob der Sensor gestartet werden kann. Falls dies nicht der Fall sein sollte, wird eine Fehlermeldung ausgegeben. Nach erfolgreichem Start ist der Arduino angehalten so lange mit dem Programm zu warten, bis der Sensor zurückmeldet, dass er bereit ist, die Messung zu beginnen.

Auch während dem Programmdurchlauf wird bei jeder neuen Messung kontrolliert, ob der CO₂-Sensor funktionstüchtig ist. Danach wird mithilfe der oben genannten Bibliothek der CO₂-Wert gemessen und an den Arduino weitergegeben.

Nach Einlesen der Daten, vergleicht der Mikrocomputer diese mit den gegebenen Grenzwerten. Je nach Bewertung des gemessenen Werte wird eine der grün/gelb/roten LEDs eingeschaltet. Auch die blaue LED, welche die Ansteuerung des automatisierten Fensterscheibenmotors simulieren soll, wird je nach Messwert an- oder ausgeschaltet.

Zudem werden dem Anwender in Echtzeit die jeweiligen Daten im LCD Display aufgegeben, was durch die Bibliothek `<LiquidCrystal.h>` möglich ist.

Damit der Verlauf der Messung später auf Excel geplottet werden kann, wird auf einer Mikro-SD-Karte der gemessene Wert im .csv-Format als .txt-Datei abgespeichert.

Der Anwender nach nun entscheiden, ob er eine weitere Messung durchführen möchte oder nicht. Bei positiver Eingabe wird das Programm von vorne durchgeführt, während beim Ab-

lehnen einer weiteren Messung der Mikrocomputer in einen sogenannten Sleep-Mode geht.

5.1. CCS811

Wir haben den in Abbildung 5.1 zu sehenden CCS811-Sensor verwendet da er einige Vorteile mit sich bringt. Er ist ein digitaler Metalloxid (MOX)-Gassensor mit einem extrem geringen Stromverbrauch. Wie in Abbildung 5.2 zu sehen ist, besitzt er einen Analog-Digital (AD)-Wandler und eine I²C-Schnittstelle, was dem Entwickler eine leichte Soft- und Hardware Integration bietet. Zudem soll er eine Lebenszeit von über 5 Jahren nachweisen. [ams16, vgl. S. 1]

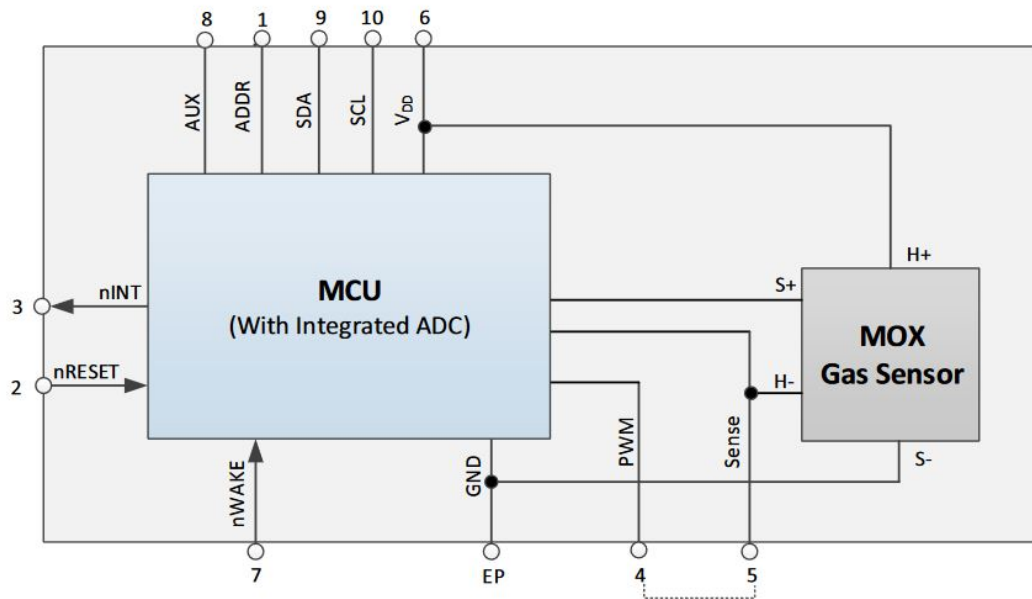


Abbildung 5.1.: Verwendeter CCS811-MOX-Gassensor

Der CO₂-Sensor ist über den Pins SDA und SCL mit dem Arduino verbunden. Pin 7 ist dabei auf Masse kurzgeschlossen. Die restlichen Pins werden in unserem Projekt nicht benötigt.

Der Abruf von Messdaten auf dem I²C erfolgt über das Master-Slave Prinzip. Dabei nimmt der Arduino die Rolle des Masters, und der CO₂-Sensor die des Slaves ein. Das bedeutet, der CCS811 darf nur Informationen senden, wenn er vom Arduino die Aufforderung bekommen hat.

Um an die richtigen Daten des Sensors zu gelangen, muss der Arduino in unserem Projekt die ersten beiden Bytes auslesen, da diese die nötigen Informationen über den CO₂-Gehalt der Luft zu Verfügung stellen. Die restlichen Bytes werden in unserem Projekt nicht ausgelesen, da wir für diese keine Verwendung haben.



Quelle: [ams16, S. 3]

Abbildung 5.2.: CCS811 Blockdiagramm

Byte 0	Byte 1	Byte 2	Byte 3
eCO ₂ High Byte	eCO ₂ Low Byte	TVOC High Byte	TVOC Low Byte

Byte 4	Byte 5	Byte 6	Byte 7
STATUS	ERROR_ID	See RAW_DATA	See RAW_DATA

Quelle: [ams16, vgl. S. 14]

Abbildung 5.3.: Inhalt der 8-Byte-Übertragung des CCS811-MOX-Sensors

In der Softwareimplementierung wird der CO₂-Sensor mithilfe der Bibliothek <Ardufruit_CCS811.h> wie in Abbildung 5.4 gezeigt wird aufgerufen. Zunächst werden Setups für das I²C-Interface und die Hardware durchgeführt. Danach wird geprüft, ob die Kommunikation mit dem Sensor aufgebaut werden kann. Wenn das der Fall ist, wird solange gewartet, bis der Sensor bereit ist, Daten zu lesen. Falls jedoch ein Problem auftritt, wird eine Fehlermeldung ausgegeben und der Arduino geht in eine sogenannte Endlosschleife, bis das Programm neu geladen wird oder der Arduino ausgeschaltet wird.

Die Abfrage nach den CO₂-Werten erfolgt über eine weitere Prüfung auf Fehler und einer anschließenden Messung. Danach werden die Daten auf einem Array gespeichert.

```
// sensor controll
if(!CCS.begin()){
    Serial.println("Failed to start sensor! Please check your wiring.");
    while(1);
}
// wait for the sensor to be ready
while(!CCS.available());
```

Abbildung 5.4.: Codeausschnitt für das Setup aus dem Softwareprogramm

```
// checks if data is available to be read
if(CCS.available()){
    // read and store the sensor data
    // data can be accessed with geteCO2
    if(!CCS.readData()){
        // measure co2 value
        measurement[i] = CCS.geteCO2();
    }
}
```

Abbildung 5.5.: Codeausschnitt für die Messung aus dem Softwareprogramm

5.2. Mikro-SD

6. Hardware

@Serkant: Bitte auch noch das CAD-Modell einbinden!

7. Testing

Projekt: CO2-Sensor	Datum:
ID: CO201	Version: 1.0
Titel: Visualisierung der Luftqualität auf Basis von CO2-Grenzen	
Items: void ask(int)	TestKfg: 01
Zielsetzung: Der Test soll zeigen, dass die Software die gemessenen CO2-Werte richtig interpretieren kann.	
Anforderungen: R01	
Erforderliche Inputs zu Testbeginn: CO2 Werte	

Tabelle 7.1.: Testbeschreibung Test 1

Tester:	Beobachter:
Protokolldatei:	
Status:	Problembericht:

Tabelle 7.2.: Testprotokoll Test 1

Projekt: CO2-Sensor	Datum:
ID: CO202	Version: 1.0
Titel: Auswahl von verschiedenen Messprofilen	
Items: void ask(int)	TestKfg: 01
Zielsetzung: Der Test soll zeigen, dass der Anwender zwischen drei verschiedenen Messprofilen wählen kann.	
Anforderungen: R03	
Erforderliche Inputs zu Testbeginn: Anwender	

Tabelle 7.3.: Testbeschreibung Test 2

Tester: Alexander Herrmann	Beobachter: Johannes Ruffer
<p>Protokoll:</p> <p>Zunächst wurde der Arduino an den Laptop angeschlossen und somit das aktuelle Programm gestartet. Nach Auswahl des Schreibmodus, wurde das Messprofilmenü angezeigt. Nach drei UP-Button-Klicks, wurde der Messmodus 1 (Echtzeitmodus) mit dem Enter-Button bestätigt.</p> <p>Die anschließende Messung wurde in Echtzeit auf dem LCD-Display ausgegeben. Die abgespeicherten Messungen wurden ebenfalls in Echtzeit gemessen. Anschließend wurde das Programm neu gestartet, um den Modus 2 auszuwählen. Auch dieser wurde erfolgreich mit den richtigen Zeitintervallen durchgeführt.</p> <p>Der dritte Start des Programms diente dazu, die Tagesmessung auszuprobieren. Auch hier hat die Auswahl und Durchführung in Bezug auf die Testanforderungen erfolgreich funktioniert.</p>	
Status: Erfolgreich	Problembericht: Nicht vorhanden

Tabelle 7.4.: Testprotokoll Test 2

8. Handbuch

9. Installationsanleitung

Die folgenden Bibliotheken müssen heruntergeladen sein:

- Adafruit_CCS811.h (Arduino driver für den CCS811 Gas-Sensor)
- SPI.h (Ermöglicht die Kommunikation mit dem Serial Peripheral Interface)
- Wire.h (Ermöglicht die Kommunikation über I^2C)
- LiquidCrystal.h (Zur Integration des LCD-Displays)

In der folgenden Tabelle sind alle Verbindungen zwischen den einzelnen Komponenten mit dem Arduino aufgelistet. In 4.1 ist das entsprechende Layout als Zeichnung zu finden.

Nummer	Bauteil	Spezifikation	Anschlusspin Arduino
1	LCD-Display	RS RW E D4 D5 D6 D7 A K	D7 GND D6 D5 D4 D3 D2 5V GND
2	LEDs	Rot Gelb Grün Blau	A3 A2 A1 A0
3	Taster	Up-Button Enter-Button	12 13
4	CCS811	WAK SDA SCL VCC	GND SDA SCL 5V
5	MicroSD-Slot	GND MISO MOSI SCK CS VCC	GND D11 D10 9 8 5V

Tabelle 9.1.: Zuordnung der Pins

10. Fazit

Verzeichnis verwendeter Abkürzungen und Formelzeichen

DHBW Duale Hochschule Baden-Württemberg

MOX Metalloxid

AD Analog-Digital

LCD LiquidCrystal Display

Literaturverzeichnis

- [ams16] AMS AG ; SPARKFUN (Hrsg.): *CCS811: Ultra-Low Power Digital Gas Sensor for Monitoring Indoor Air Quality*. https://cdn.sparkfun.com/assets/learn_tutorials/1/4/3/CCS811_Datasheet-DS000459.pdf. Version: 2016
- [Dep] DEPARTMENT OF RESEARCH & DEVELOPMENT, DEPARTMENT OF INFORMATION TECHNOLOGIES AND SYSTEMS (Hrsg.): *A Data Quality Measurement Information Model Based On ISO/IEC15939*. https://s3.amazonaws.com/academia.edu.documents/36881777/ICIQ2007-ADataQualityMeasurementInformationModelBasedOnISOIEC15939.pdf?response-content-disposition=inline%3B%20filename%3DA_Data_Quality_Measurement_Information_M.pdf&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWOWYYGZ2Y53UL3A%2F20200310%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200310T081440Z&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&X-Amz-Signature=68ac8145994178d50baf90ab0a7557af4a341fa239938ed408aba2cd8a3c8e95
- [Hel] HELGA MEYER, Heinz-Josef R.: *Projektmanagement: Von der Definition über die Projektplanung zum erfolgreichen Abschluss*. Bremen : Springer Gabler <https://books.google.de/books?id=1ho3CwAAQBAJ&printsec=frontcover&dq=projektmanagement+von+der+definition+%C3%BCber+die+projektplanung+zum+erfolgreichen+abschluss&hl=de&sa=X&ved=0ahUKEwi66ufKpY3oAhUF06YKHSFvB7wQ6AEIMjAB#v=onepage&q=projektmanagement%20von%20der%20definition%20%C3%BCber%20die%20projektplanung%20zum%20erfolgreichen%20abschluss&f=false>. – ISBN 978-3-658-07569-9
- [Jay] JAY ABRAHAM, PAUL JONES, RAOUL JETLEY: *Formale Verifikationsmethoden für die Entwicklung von High-Integrity-Software für Medizinische Geräte*

- [Pet09] PETER LIGGESMEYER: *Software-Qualität: Testen, Analysieren und Verifizieren von Software: Prof. Dr.-Ing. 2.* Heidelberg : Spektrum Akademischer Verlag, 2009 <https://link.springer.com/content/pdf/10.1007%2F978-3-8274-2203-3.pdf>. – ISBN 978-3-8274-2056-5
- [Wik20] WIKIPEDIA (Hrsg.): *Softwaretest*. <https://de.wikipedia.org/wiki/Softwaretest#Literatur>. Version: 01.03.2020

Abbildungsverzeichnis

3.1. Ablauf des Projektes von Januar bis April	6
4.1. Schaltungslayout vom 27.02.2020	8
5.1. Verwendeter CCS811-MOX-Gassensor	10
5.2. CCS811 Blockdiagramm	11
5.3. Inhalt der 8-Byte-Übertragung des CCS811-MOX-Sensors	11
5.4. Codeausschnitt für das Setup aus dem Softwareprogramm	12
5.5. Codeausschnitt für die Messung aus dem Softwareprogramm	12

Tabellenverzeichnis

2.1. Anforderungen an das Projekt	3
3.1. Aufgewendete Kosten für das Projekt	5
7.1. Testbeschreibung Test 1	15
7.2. Testprotokoll Test 1	15
7.3. Testbeschreibung Test 2	15
7.4. Testprotokoll Test 2	16
9.1. Zuordnung der Pins	20

A. Anhang

A.1. Weitere Abbildungen