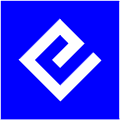
|  |  |  |
| --- | --- | --- |
| Josef-Effner-Gymnasium  Dachau | Abiturjahrgang  2018 | |
| **S E M I N A R A R B E I T** | | |
| **Rahmenthema des Wissenschaftspropädeutischen Seminars:**  Wissenschaftspropädeutisches Arbeiten als pädagogisch-didaktische Aufgabe in der Oberstufe des bayerischen Gymnasiums  **Leitfach:** Informatik | | |
| **Thema der Arbeit:**  **Entwicklung von Hard- und Software für die Messung und Auswertung des (Raum-)Klimas** | | |
| **Verfasser/in**:  Lukas Aldersley | | **Kursleiter/in:**  StR Sebastian Paintner |
| Abgabetermin: | | 07. November 2017 |



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bewertung** | Note | Notenstufe in Worten | Punkte |  | Punkte |
| schriftliche Arbeit |  | befriedigend | 7 | x 3 | 21 |
| Abschlusspräsentation |  | gut | 10 | x 1 | 10 |
| Summe: | | | | |  |
| Gesamtleistung nach § 61 (7) GSO = Summe : 2 (gerundet) | | | | | 16 |

|  |
| --- |
|  |
| Datum und Unterschrift der Kursleiterin / des Kursleiters |

Entwicklung von Hard- und Software für die Messung und Auswertung des (Raum-)Klimas

Inhaltsverzeichnis

[1. Vorwort 3](#_Toc497773248)

[2. Wahl der Komponenten und Programmiersprachen 4](#_Toc497773249)

[2.1. Wahl der Hardwarebasis 4](#_Toc497773250)

[2.2. Wahl der Programmiersprache 5](#_Toc497773251)

[3. Ursprüngliche Anforderungen und erste Pläne für die Umsetzung 6](#_Toc497773252)

[3.1. Anforderungen der Fachschaften Physik und Geographie 6](#_Toc497773253)

[3.2. Umsetzungspläne 6](#_Toc497773254)

[4. Probleme / Technische Schwierigkeiten bei der Umsetzung 8](#_Toc497773255)

[4.1. Probleme beim Bau der Hardware 8](#_Toc497773256)

[4.2. Probleme bei der Software 9](#_Toc497773257)

[5. Erweiterte Anforderungen und neues Konzept für die Umsetzung 11](#_Toc497773258)

[5.1. Hinzugefügte Anforderung 11](#_Toc497773259)

[5.2. Überarbeitetes Konzept 11](#_Toc497773260)

[5.3. Probleme bei der Fertigstellung 11](#_Toc497773261)

[6. Fertiges Projekt 13](#_Toc497773262)

[7. Nachwort 15](#_Toc497773263)

[8. Betriebsanleitung 16](#_Toc497773264)

[8.1. Java Programm 16](#_Toc497773265)

[8.2. Gerät 18](#_Toc497773266)

[9. Quellenverzeichnis 19](#_Toc497773267)

[9.1. In der Arbeit verwendete Quellen 19](#_Toc497773268)

[9.2. Im Code verwendete Quellen 19](#_Toc497773269)

[9.2.1. Java 19](#_Toc497773270)

[9.2.2. Arduino 20](#_Toc497773271)

[10. Anhang 21](#_Toc497773272)

[11. Erklärung 24](#_Toc497773273)

# Vorwort

Wenn man einen Raum, besonders in einem öffentlichen Gebäude (beispielsweise eine Schule), betritt, ist es anhand der Luftqualität leicht erkennbar, ob dieser zuvor benutzt worden ist. Ein stickiges Raumklima hat üblicherweise einen merklichen Einfluss auf die Leistungs- und Konzentrationsfähigkeit sich darin befindlicher Personen. Als Schüler werde ich selbst täglich mit jener Situation konfrontiert, woraus die Idee der Entwicklung einer Hard- und Software zur Messung und Auswertung von Raumklima entstand. Im Folgenden beabsichtige ich, die genaueren Details und Schritte zu erläutern.

# 2. Wahl der Komponenten und Programmiersprachen

## 2.1. Wahl der Hardwarebasis

Für den Bau eines Gerätes, das Daten aufzeichnen soll, gibt es mehrere mögliche Plattformen, auf denen so ein Projekt realisiert werden kann. Zum einen gibt es diverse Mikrocontroller, zum anderen (Linux) Einplatinencomputer. Ferner gibt es noch einige weitere Optionen, auf die hier aber nicht eingegangen wird, da diese nur für sehr spezielle Anwendungsfälle sinnvoll sind.

Für dieses Projekt muss die Möglichkeit gegeben sein, relativ viele Sensoren anzuschließen und die Sensordaten aufzuzeichnen; Auswertung oder größere Berechnungen am Gerät sind nicht notwendig.

Um zu entscheiden, welche Plattform besser geeignet ist, werden nachfolgend die wichtigsten Eigenschaften gegeneinander abgewogen. Beispielhaft werden in diesem Vergleich der Raspberry PI 3 Model B und der Arduino MEGA2560 R3.0 verwendet.

Ein Micro Controller bietet zwar deutlich weniger Rechenleistung, hat jedoch mehr I/O Pins, die in einem solchen Projekt mehr wert sind als die Fähigkeit komplexere Berechnungen durchzuführen.

Der Raspberry hat PI 40 Pins, von denen prinzipiell 26 als GPIO (General Purpose Input/Output) verwendet werden können, jedoch können z.B. Pins GPIO 14 und GPIO 15 nicht beliebig verwendet werden, da der Raspberry PI eine Konsole über diese beiden Pins ausgibt. Auch wenn diese deaktiviert werden kann, ist dies nur eine der Schwierigkeiten, die bei Einplatinencomputern auftreten können. Darüber hinaus ist die Ansteuerung der Pins äußerst aufwendig. Mikrocontroller wie der Arduino MEGA 2560 haben 70 I/O Pins, von denen 16 analoge Signale messen können (mit einem Integrierten ADC (=Analog-Digital-Converter)) und 15 PWM-Signale senden können (PWM=Puls Weiten Modulation; pseudo Analog). Viele Einplatinencomputer wie z.B. der Raspberry PI haben keinen ADC, der aber für einige Sensoren zwingend notwendig ist. Das wichtigste Argument gegen einen Einplatinencomputer ist die Beschränkung der Eingangsspannung auf maximal 3.3 V. Viele Sensoren arbeiten mit 5 V, weswegen diese auch ihr Ausgangssignal mit 5 V senden, was die meisten Einplatinencomputer zerstören würde. Einige Mikrocontroller wie z.B. der Arduino Due arbeiten ebenfalls mit maximal 3.3 V und scheiden deswegen aus.

Die wichtigsten Argumente sind im Folgenden nochmals als Tabelle zusammengefasst.

|  |  |  |
| --- | --- | --- |
|  | Einplatinencomputer  (Raspberry PI 3 Model B) | Mikrocontroller  (Arduino MEGA 2560) |
| Anzahl der nutzbaren Pins | 24 (26) | 70 |
| ADC Pins | 0 | 16 |
| PWM | 1 | 15 |
| Maximale Spannung | 3.3V | 5V |
| Ansteuerung der Pins | Meistens Kompliziert | Tendenziell simpler zu verwenden |

Nach diesen Überlegungen bleibt nur eine Option: Arduino.

Während der Entwicklung und möglicherweise im finalen Produkt wird ein Arduino MEGA 2560 verwendet werden, da dieser mehr Pins und mehr Speicher als andere Arduino-Boards hat. Dies hat den Vorteil, dass während der Entwicklung mehr Sensoren angeschlossen werden können, da mehr freie Pins vorhanden sind, und nicht genauestens auf den Speicherbedarf des Programms geachtet werden muss, da der Arduino MEGA 2560 achtmal mehr Speicher als ein Arduino Uno hat.

Im finalen Produkt soll aber, wenn möglich, ein Arduino UNO oder Arduino Pro Mini verwendet werden, da diese kleiner und kostengünstiger sind.

## 2.2. Wahl der Programmiersprache

Aufgrund der Wahl der Arduino Plattform ergibt sich als sinnvolle Programmiersprache die üblicherweise verwendete, angepasste Mischform aus C und C++, mit der Arduinos üblicherweise programmiert werden.

Für die Auswertungssoftware gibt es im Grunde zwei Möglichkeiten: Java oder C#. Java hat eine sehr große Community, in der man für die meisten Probleme Hilfe erhalten kann. Außerdem bietet Java gegenüber C# den Vorteil plattformunabhängig zu sein. C# ist zunächst nur auf einigermaßen aktuelle Windows-Versionen beschränkt. Es ist zwar möglich C# Code auf Linux auszuführen, dies bedarf allerdings einer speziellen Software und der Code muss gesondert für Linux kompiliert werden. C# bietet für die meisten Anwendungsfälle zwar weit überlegene APIs, jedoch war mir keine Library bekannt, mit der das Zeichnen von Graphen ohne weiteres möglich ist. Wohingegen ein Java-Projekt, das Sensordaten live zeichnen konnte, bekannt war. Das bedeutet, das mindestens eine Java-Library, mit der Graphen gezeichnet werden können existiert.

Aufgrund der Plattformunabhängigkeit und dem fehlenden Wissen von passenden C# Libraries fiel die Wahl auf Java, obwohl C# sehr viel angenehmer gewesen wäre.

# 3. Ursprüngliche Anforderungen und erste Pläne für die Umsetzung

## 3.1. Anforderungen der Fachschaften Physik und Geographie

Die Anforderungen der Fachschaft Physik halten sich in Grenzen, vor allem was die Hardware betrifft. Für die Fachschaft Physik ist es wichtig, die Luftfeuchtigkeit und die Temperatur zu messen und auf Knopfdruck auf eine (micro) SD-Karte aufzeichnen zu können. Die Messintervalle für die Aufzeichnung sind mit etwa einer Messung alle zehn Sekunden festgesetzt. Die Software muss, um für die Fachschaft Physik ausreichend zu sein, in der Lage sein die Messwerte als Kurvenverlauf zu zeichnen und bei Bedarf genaue Messwerte für einen bestimmten Zeitpunkt auszugeben. Außerdem ist eine integrierte Betriebsanweisung gefordert.

Ferner wollte die Fachschaft Geographie die Möglichkeit haben, die Temperatur und die Luftfeuchtigkeit zu messen. Darüber hinaus möchte die Fachschaft Geographie, falls möglich, in der Lage sein, die Niederschlagsmenge aufzuzeichnen. Die Messintervalle für den Geographiemodus sind auf vier Messungen pro Tag (00:00 Uhr, 06:00 Uhr, 12:00 Uhr und 18:00 Uhr) festgelegt. Wie schon von der Fachschaft Physik angefordert, soll auch hier die Aufzeichnung auf eine SD-Karte erfolgen. Die Software muss nur in der Lage sein, die Werte in Kurvenform anzuzeigen.

## 3.2. Umsetzungspläne

Um die gerade beschriebenen Anforderungen zu erfüllen, wird eine RTC d.h. eine Echtzeituhr, zur Festlegung der Dateinamen benötigt. Deren Einsatz ergibt folgende Vorteile: Der Benutzer muss sich nicht auf umständliche Weise um die Festlegung und Eingabe eines Dateinamens kümmern. Es wird auch vermieden, dass mehrere Aufzeichnungen in derselben Datei gespeichert werden, was zu einer Fehlinterpretation von Messwerten führen könnte, die zeitlich weit auseinanderliegen.

Die micro SD-Karten Anbindung soll über die SPI-Schnittstelle (Serial Peripheral Interface) mit einem Arduino Ethernet Shield als Adapter realisiert werden. Shields sind Aufsteckmodule, die von oben auf den Arduino montiert werden und zusätzliche Schnittstellen bzw. Funktionen ermöglichen können. Das Ethernet Shield stellt die Möglichkeit bereit, einen Arduino über einen RJ45 Stecker per Ethernet mit dem Internet zu verbinden, und hat zusätzlich einen micro SD Steckplatz. Die Ethernet-Fähigkeiten sind für diesen Anwendungsfall irrelevant und werden nicht angeschlossen oder benutzt.

Des Weiteren werden Sensoren benötigt, um den eigentlichen Hauptzweck des Geräts zu erfüllen.

Für die Temperatur und die Luftfeuchtigkeit fiel die Wahl auf den AM2302/DHT22 Sensor, der im Arduino Umfeld sehr weit verbreitet ist. Neben dem hohen Verbreitungsgrad spricht auch die Fähigkeit sowohl Temperatur als auch Luftfeuchtigkeit zu messen für den Sensor.

Der Niederschlagsmengensensor für die Fachschaft Geographie stellte eine sehr viel größere Herausforderung dar. Da kein fertiges Modul, das für dieses Projekt geeignet schien, vorhanden war, musste ein Sensor selbst entwickelt werden. Da dieser für die Auftraggeber von geringerer Priorität ist, wurde die Entwicklung dieses Sensors auf einen späteren Zeitpunkt verschoben, da die Entwicklung der allgemeinen Funktionalität Vorrang hatte.

# 4. Probleme / Technische Schwierigkeiten bei der Umsetzung

## 4.1. Probleme beim Bau der Hardware

Beim ersten Test des Geräts schien etwas mit der SD-Karte oder der SD-Library nicht in Ordnung zu sein, da nach einer Test-Aufzeichnung immer noch keine Datei auf der SD-Karte zu finden war. Es stellte sich heraus, dass das Problem darin lag, dass die SD-Library nur die Dateisysteme FAT16 oder FAT32, jedoch ohne die VFAT Erweiterung und damit nur Dateinamen nach dem 8.3 Standard unterstützt.[[1]](#footnote-1)[[2]](#footnote-2) Die Lösung für dieses Problem war einfach: Den Dateinamen kürzen. Nachdem nun die Messwerte aufgezeichnet werden konnten, kristallisierte sich ein weiteres Problem heraus. In der Datei mit den Messwerten fand sich häufig „NaN“ (= Not a Number) anstatt eines Messwertes. Bei genauerer Untersuchung stellte sich heraus, dass sich „NaN“ etwa alle 5-10 Minuten in den Messreihen des DHT22-Sensors fand. Diese Beobachtung führte zu dem Entschluss zwei andere Sensoren zusätzlich anzuschließen und über den Zeitraum von einigen Wochen zu testen, um zu entscheiden welcher Sensor am besten geeignet ist. Die beiden zusätzlich angeschlossenen Sensoren waren beide von Bosch: der BMP180 und der BME280. Beide Sensoren können die Temperatur und den Luftdruck messen, der BME280 kann außerdem auch, wie der DHT22, die Luftfeuchtigkeit messen. Aus den so gewonnenen Vergleichsdaten wurde der Verdacht, dass der DHT22 abgesehen von seinem niedrigen Preis keinerlei Vorteile bietet, als unbrauchbar gilt und sofern möglich vermieden werden sollte, bestätigt.

Es wird zunehmend ersichtlich, dass es Probleme beim Anschließen der Komponenten geben muss, da BMP180, BME280, die RTC (Real-Time Clock/Uhr) und zwei LCD Displays über das I²C-Protokoll (Inter-Integrated Circuit) gesteuert werden. Der Arduino kann jedoch nur über die Pins 21 (Taktsignal) und 22 (Datenleitung) mit I²C-Geräten kommunizieren. Glücklicherweise ist I²C ein Bus-Protokoll d.h. es können mehrere Geräte an diesen Bus angeschlossen werden, die nur dann Daten senden, wenn der Master z.B. vom Gerät „LCD-Display 1“ mit der Adresse 0x27 Daten anfordert. Das bedeutet, dass der Arduino in regelmäßigen Intervallen ein einzelnes Gerät ansprechen muss, welches daraufhin seine Messwerte zurücksendet und danach wartet bis wieder Daten angefordert werden.

Während der Testphase wurde noch ein weiteres signifikantes Problem bemerkt: Die Stromversorgung des Geräts. Um zumindest zeitweise batteriebetrieben arbeiten zu können musste die Leistungsaufnahme, die zu diesem Zeitpunkt noch bei etwa 10W lag, drastisch reduziert, oder auf kurze Leistungsphasen beschränkt werden. Die Möglichkeit das Gerät Netzunabhängig zu betreiben ist hauptsächlich für die Fachschaft Geographie, die nur 4-mal Täglich Messwerte benötigt, relevant. Deshalb kann in den etwa 6 Stunden zwischen den Messzeiten für Geographie die Stromzufuhr für die Sensoren getrennt werden und erst kurz vor der Messung wieder verbunden werden. Dies kann die Leistungsaufnahme in den inaktiven Phasen auf 0.01W reduzieren. Das Blockieren bzw. Freigeben der Spannung wird mithilfe des TIP122 NPN Transistors realisiert.

## 4.2. Probleme bei der Software

Das erste Problem, das auftrat, war, dass es keine geeigneten Beispiele bzw. keine Dokumentation zu JFreeChart, also der Library, mit der die Wertverläufe gezeichnet werden sollten, gab. Weitergeholfen hat hier ein Video auf YouTube, in dem jemand einen Photowiderstand an einen Mikrocontroller angeschlossen hatte und die Messwerte in Echtzeit als Graph darstellen lassen konnte[[3]](#footnote-3). Leider konnte das aus dem Quellcode gewonnene Wissen nicht direkt für dieses Projekt verwendet werden, da in dem gefundenen Quellcode nur eine Messreihe gezeichnet werden konnte und es keine Option gab, wie von der Fachschaft Physik angefordert, die Messwerte an einer bestimmten Stelle auszugeben, nachdem auf einen Punkt in der Zeichenebene geklickt wurde. Aus der Demo-Datei, die beim Download der Library mitgeliefert wird, kann abgelesen werden wie einzelne Features heißen, jedoch nicht wie sie Implementiert werden sollen. Anhand der Namen der gesuchten Klassen konnte mithilfe der Code-Vorschläge von Eclipse erraten werden wie auch die von der Fachschaft Physik angeforderte Funktion implementiert werden konnte.

Nachdem nun eine grundlegende Funktionalität gegeben war, musste das Layout geordnet werden. Dies war Notwendig, da manches so seltsam angeordnet wurde, dass Teile der Bedienoberfläche gar nicht zu sehen waren oder einfach nicht lesbar waren (siehe Bilder 1-4). Möglicherweise sind die Zuordnungen welches Element an welcher Stelle stehen soll redundant (z.B. übergeordnetes Element und untergeordnetes Element werden beide angewiesen sich am linken Rand anzuordnen). Teilweise traten hier auch sehr merkwürdige Phänomene wie das völlige Ignorieren von Anweisungen oder das Ausführen der gegenteiligen Reaktion. So kam es vor, dass zum Beispiel, sich manche Elemente, nachdem sie mit der Anweisung „element.setAlignmentX(Component.LEFT\_ALIGNMENT);“ an den linken Bildschirmrand angeheftet wurden, sich so weit nach rechts bewegten, dass sie zu etwa 75% aus dem Sichtbereich verschwanden. Stattdessen hatte die Anweisung „element.setAlignmentX(Component.RIGHT\_ALIGNMENT);“ den Effekt, dass sich das Element am linken Bildschirmrand anordnete.

Um mehr Platz für die Graphen zu haben kam die Idee auf die Java Full-Screen Exclusive Mode API zu verwenden auf. Wie in den Programmierhinweisen[[4]](#footnote-4) erwähnt, wird mit dem Methodenaufruf *isFullScreenSupported()* überprüft, ob die Funktion unterstützt wird. Das Ergebnis dieser Abfrage war stets *true,* obwohl der Aufruf der Methode *setFullscreenWindow(JFrame)* zwei unterschiedliche Ergebnisse hervorbrachte. (Siehe Bild 5 und Bild 6) Die Ursache für die unterschiedlichen Ergebnisse war, dass beim ersten Test eine AMD Radeon HD5450 benutzt wurde und beim zweiten Test diese dedizierte GPU ausgebaut war, weshalb die Bildausgabe über die integrierte Intel HD 610 erfolgte. Aufgrund der Tatsache, dass die Software, auch während die Intel Grafikeinheit für die Bildausgabe verantwortlich war, meldete, dass der Vollbildmodus möglich sei, wurden weitere Tests auf anderen Computern ausgeführt. Als Ergebnis konnte festgehalten werden, dass auf allen Computern die eine Integrierte Grafikeinheit von Intel verwenden das Problem wie in Bild 6 zu erkennen auftrat. Es gab zwei mögliche Lösungen für dieses Problem. Die erste war, die Vollbildfunktion gänzlich zu entfernen und die zweite Möglichkeit war, eine Fallunterscheidung einzufügen, die zunächst überprüft auf welchem Betriebssystem das Programm ausgeführt wird und je nach System per Befehlszeile den Namen der verwendeten Grafikeinheit abfragt. Wenn der Name der Grafikeinheit das Wort „Intel“ enthält wird das Fenster nur maximiert, und kein Vollbildmodus verwendet. Wenn der Name jedoch das Wort „Intel“ nicht enthält wird der Vollbildmodus verwendet. Das Funktionieren dieser Unterscheidung wurde sowohl auf Windows (Windows 7, 8 und 10) als auch unter Linux (Debian und Ubuntu) mit ATI/AMD Grafikkarten (sowohl Integriert als auch dediziert), Nvidia Grafikkarten und Intel Grafikeinheiten getestet.

Nachdem die Software weitgehend stabil lief, wurden auch Aufzeichnungen langer Messperioden mit etwa 1.000.000 Datensätzen zu je 16 Messwerten durchgeführt. Bereits beim ersten Test wurde klar, dass Anpassungen zwingend notwendig waren: Die Ladezeit für diese etwa 16.000.000 Messwerte betrug ungefähr 4 Stunden. Als Lösung für dieses Problem wurde die Möglichkeit die Werte zu Interpolieren eingefügt. D.h. dem Benutzer wurde die Möglichkeit gegeben auszuwählen wie viele Werte zwischen den zu zeichnenden Werten übersprungen werden sollten. Außerdem werden in der finalen Version nicht mehr 16 Werte pro Datensatz gespeichert, sondern voraussichtlich nur noch 5 Werte. Dies sollte dazu führen, dass die Ladezeiten im akzeptablen Bereich bleiben werden.

# 5. Erweiterte Anforderungen und neues Konzept für die Umsetzung

## 5.1. Hinzugefügte Anforderung

Bei den Rücksprachen mit den Auftraggebern wurden im Verlauf des Projekts noch einige weitere Anforderungen hinzugefügt. Die Fachschaft Physik wünschte sich die Möglichkeit CO2 Werte, die Umgebungshelligkeit und, falls möglich bzw. aussagekräftig, die Lautstärke zu messen. Ferner wurde der Wunsch, per Knopfdruck die Auswertungssoftware am PC starten zu können geäußert.

Die Fachschaft Geographie fügte die Anforderung, dass der Luftdruck und die Helligkeit zu messen seien, hinzu. Ferner sollte ein Sondermodus in der Auswertungssoftware geschaffen werden, der es erlaubt, den Tagesdurchschnitt der Temperatur und die Summe der Niederschläge des Tages anzuzeigen.

## 5.2. Überarbeitetes Konzept

Wie bereits in Kapitel 4.1 erwähnt, wurde die ursprüngliche Entscheidung den DHT22 Sensor zu verwenden, verworfen. Da BMP180 und BME280 immer die Gleichen werte lieferten und nun auch noch der Luftdruck gemessen werden soll, fiel die Endgültige Entscheidung, welcher Sensor am besten geeignet ist, zugunsten des BME280 aus, da nun alle drei Funktionen, die dieser bietet benötigt werden und keine Probleme bezüglich der Genauigkeit festgestellt werden konnten.

Die Messung des CO2-Wertes wird mithilfe eines MQ135 Gas Sensors durchgeführt. In einem Blog[[5]](#footnote-5) wurde erläutert, wie ein solcher Sensor zu verwenden ist.

Die Helligkeit wird mit zwei Photowiderständen, die an den ADC des Arduino angeschlossen werden, gemessen.

Der Rest des Konzeptes blieb gleich.

## 5.3. Probleme bei der Fertigstellung

Das erste Problem, das eigentlich die ganze Zeit über bestand, ist das ein Gehäuse benötigt wird. Dieses muss Wasserdicht und trotzdem Luftdurchlässig sein, da die Elektronik vor Regen geschützt sein muss, und trotzdem die Sensoren reelle Messwerte liefern sollen. Gelöst wurde das Problem durch die Idee, ein Vogelhäuschen zu verwenden.

Ein weiteres Problem wurde während der späteren, längeren Testphasen immer offensichtlicher: Die Temperaturen wurden immer ungenauer. Dies fällt nur auf, wenn man die gemessene Temperatur mit der Ausgabe eines Thermometers vergleicht. Der Grund für die Differenz war der MQ135-Sensor, der für die Messung des CO2-Wertes verwendet wird, da dieser sich aufheizen muss um zu funktionieren. Die Lösung hierfür, ist die Temperaturempfindlichen Sensoren weiter vom MQ135 weg zu Platzieren und eine Art „Hitzeschild“ in Form eines Stücks Platine, das Senkrecht zwischen dem MQ135 und den anderen Sensoren steht, einzubauen.

Die Fachschaft Physik wollte die Möglichkeit haben per Knopfdruck am Gerät die Auswertungssoftware zu starten. Dies ist nicht möglich, zumindest nicht mit der schon weit fortgeschrittenen Zeit, als der Wunsch geäußert wurde. Um die Auswertungssoftware per Knopfdruck zu starten, müsste der Arduino sich als Tastatur/Maus ausgeben. Der Arduino MEGA 2560, der für dieses Projekt verwendet wird kann sich nicht als ein anderer Gerätetyp ausgeben. Ein Arduino Leonardo kann das tun, hat jedoch nicht genügend Speicherplatz um die Firmware des Geräts aufzunehmen. Um genügen Speicher für die Firmware zu haben müsste man mehrere Controller zusammenschalten und das Programm aufteilen. Das ist technisch jedoch äußerst schwierig, da eine Fehlerfreie Kommunikation zwischen den beiden Controllern Voraussetzung für ein solches Konstrukt ist. Die Kommunikation zwischen den Controller müsste über SoftwareSerial, einem Seriellen Protokoll, das rein Softwarebasiert ist, erfolgen, da HardwareSerial ausschließlich zur Kommunikation mit einem Computer über USB verwendet werden kann. Da die Kommunikation hier ausschließlich über Software erfolgt, müssen beide Controller zu jedem beliebigen Zeitpunkt in der Lage sein auf eingehende Anweisungen zu reagieren. Dies ist grundlegend anders als bei HardwareSerial (oder nur Serial) wo eingehende Nachrichten in einem Puffer zwischengespeichert werden und von dort aus eingelesen werden können. Der eben erwähnte Pufferspeicher ist bei SoftwareSerial nicht vorhanden. Das Problem liegt darin, dass Interrupts verwendet werden um auf eingaben des Benutzers zu reagieren. Interrupts werden bevorzugt behandelt, weswegen manche Anweisungen des einen Controllers an den anderen verloren gehen können. Das Verlieren von internen Anweisungen führt zwangsläufig zu Fehlern, die den Reibungslosen Betrieb des Geräts verhindern würden.

# 6. Fertiges Projekt

Die Hardware des Projekts basiert auf dem Arduino MEGA 2560, auf den alle Sensoren und sonstigen Geräte aufgesteckt werden. Folgende Geräte werden verwendet:

* DS3231 RTC
* 2x LDR (Photowiderstand)
* BME280
* MQ135
* 2004 I²C-LCD
* 1602 I²C-LCD
* 2x TIP122 NPN-Transistor
* SD-Karte
* Lautstärkensensor
* Servo
* HX711 Wägezellensteuerung
* 5kg Wägezelle

Die Positionen, in denen die Komponenten auf die Platine gesteckt werden können sind Eindeutig bis auf die Positionen der Stecker für die LCD-displays und die Helligkeitssensoren. Diese dürfen in alle Steckplätze, in die sie passen gesteckt werden. Beim Zusammenstecken der Teile soll mit den größten Bauteilen angefangen werden. Dadurch wird sichergestellt, dass Teile nur in Steckplätze gesteckt werden können, in die sie auch dürfen. Es muss allerdings darauf geachtet werden, dass alle Pins die ein Bauteil hat, irgendwo eingesteckt sind. Um die Elektronik vor Regen zu schützen, befindet sich der Großteil der empfindlichen Teile in einem Vogelhäuschen welches als Gehäuse dient. Es besteht die Möglichkeit, einen Niederschlagssensor anzuschließen. Dieser befindet sich außerhalb des Gehäuses und wird nur verwendet, wenn der Geographie-Modus aktiviert ist. Der eben erwähnte Geographie-Modus ist einer von zwei Betriebsmodi. Im Standard-Modus erfolgt etwa alle zehn Sekunden eine Messung. Im Geographiemodus dagegen nur viermal täglich. Wie zwischen den Modi umzuschalten ist, wird in Kapitel 8 beschrieben.

Die Messung des CO2-Gehaltes ist mangels genauer Referenzmessungen nicht eine absolute Messung, sondern nur eine relative. Das heißt, dass die ppm-Angaben dieses Geräts nicht mit anderen vergleichbar sind.

Der zweite Teil ist die Auswertungssoftware. Diese kann die aufgezeichneten Klimadaten als Graphen anzeigen, bei Bedarf ein oder mehrere Wertereihen ausblenden sowie die Daten Interpolieren. D.h. nur z.B. jeden zehnten Wert zeichnen, um die Ladezeiten zu verbessern. Wie auch das Gerät, hat die Software zwei Modi. Einen Regulären Modus, in dem einfach nur die Werte gezeichnet werden und einen Geographie-Modus, in dem nur die Summen der Niederschläge und die Durchschnitte der Temperaturen eines Tages (d.h. 4 Messungen) angezeigt werden

# 7. Nachwort

Vorrangegangenem Text sind die detaillierten Schritte und Einzelheiten der Entwicklung und Funktionsweise der Hard- und Software zur Messung und Auswertung von Raumklima zu entnehmen. Ich würde hiermit gerne meiner Hoffnung Ausdruck verleihen, dass die Ergebnisse meiner Arbeit im Schulalltag Anwendung finden und von Nutzen sein können.

# 8. Betriebsanleitung

## 8.1. Java Programm

Beim Starten des Programms öffnet sich zunächst eine Dateiauswahl, in der die anzuzeigende Datei ausgewählt werden muss. Die anzeigbaren Dateien sind die CSV Dateien, die vom Gerät aufgezeichnet werden, aber alle anderen CSV Dateien, die nur Zahlen enthalten, können ebenfalls angezeigt werden (möglicherweise mit eingeschränktem Funktionsumfang, vor allem bei Verwendung des Geographie-Anzeigemodus).

Nach dem Klick auf "Öffnen" wird die ausgewählte Datei eingelesen und alle Werte gezeichnet. Bei einem Klick auf "Abbrechen" in der Dateiauswahl wird das Programm beendet. Nachdem eine Datei ausgewählt wurde, wird diese automatisch gelesen und versucht die enthaltenen Werte zu zeichnen. Abhängig davon, was in der Datei RaumklimaConfig.txt bzw. in den Einstellungen des Programms festgelegt wurde, werden entweder alle Wertereihen gezeichnet (Standard-Modus) oder nur die Wertereihen mit dem Titel Temperatur oder Niederschlagsmenge gezeichnet, auch wenn noch andere Werte vorhanden sind (Geographie-Modus). Hierbei werden immer 4 Werte eingelesen, der Durchschnitt der Temperatur und die Summe der Niederschläge gebildet und gezeichnet.

Nun sollten Sie in der Mitte des Fensters die Graphen der Messwerte sehen können. Direkt darunter befindet sich eine Legende, der Sie entnehmen können welche Farbe welchem Messwert zugeordnet ist. Am oberen Rand des Fensters befindet sich eine Menüleiste. Im Untermenü "Datei" finden Sie drei Optionen: "Datei öffnen", "neues Fenster öffnen" und "Fenster schließen". Wenn Sie "Datei öffnen" anklicken öffnet sich wieder die Dateiauswahl und Sie können sich eine andere Datei anzeigen lassen. "neues Fenster öffnen" bewirkt fast das gleiche, abgesehen davon, dass in diesem Fall ein weiteres Fenster geöffnet wird und Sie so zwei (oder mehr) Dateien gleichzeitig betrachten können. "Fenster schließen" schließt einfach nur das momentan aktive Fenster.

Neben dem Dateimenü finden Sie das Menü "Einstellungen", worauf später noch eingegangen wird. Rechts daneben befindet sich ein Knopf um das Programm im Vollbildmodus anzeigen zu lassen, oder diesen wieder zu verlassen. Abhängig davon welche Grafikkarte im Computer eingebaut ist, kann es sein, dass der Vollbildmodus nicht unterstützt wird. In diesem Fall wird das Fenster maximiert.

Ein Klick auf "Hilfe" wird ein Fenster, in dem diese Betriebsanleitung angezeigt wird, öffnen.

Nachdem Sie auf "Graphen speichern" geklickt haben, wird je nach dem was Sie unter "Einstellungen" gewählt haben, ein Bild der Wertereihen gespeichert.

Wenn Sie "Detailansicht" wählen, wird sich unten ein weiterer Bereich einblenden, in dem die genauen Werte für einen bestimmten Punkt angezeigt werden können. Um sich hier die Werte anzuzeigen lassen, klicken Sie bitte auf den gewünschten Punkt im Graphenbereich. Um den Detailbereich wieder auszublenden klicken Sie einfach ein weiteres Mal auf "Detailbereich".

Nachdem sie in der Menüleiste auf "Einstellungen" geklickt haben, öffnet sich ein Fenster, in dem Sie die wichtigsten Einstellungen vornehmen können. Sie sehen oben drei Registerkarten: "Allgemeine Einstellungen", "Graphenbereich" und "Tastenkombinationen".

Unter "Allgemeine Einstellungen" haben Sie zunächst die Möglichkeit zu konfigurieren was beim Speichern des Bildes der Wertereihen geschehen soll. Als Dateitypen stehen "JPG" und "PNG" zur Auswahl. Wenn ein Haken vor einem Dateityp gesetzt ist, wird ein Bild mit diesem Dateityp gespeichert, unabhängig davon, ob noch ein anderer Dateityp gewählt ist oder nicht. Direkt darunter haben Sie die Möglichkeit die Auflösung des Bildes festzulegen. Wenn "Automatisch festlegen" gewählt ist, wird die Bildschirmauflösung Ihres Computers verwendet.

Die nächste Einstellmöglichkeit legt fest, was beim Klick auf "Vollbildmodus" geschehen soll. Falls Ihr Computer eine Integrierte Grafikeineheit von Intel verwendet, können Sie hier nichts verändern und die Option "Vollbildmodus" ist ausgegraut. Auf Computern mit Intel Grafik wurde diese Funktion deaktiviert, da der Vollbildmodus zwar offiziell unterstützt wird, in der Praxis das Programm jedoch unbenutzbar macht. Falls Ihr Computer irgendeine andere Grafiklösung verwendet, können Sie wählen ob das Fenster "nur" Maximiert werden soll, oder ob in den Vollbildmodus gewechselt werden soll. Die Möglichkeiten für den Anzeigemodus wurden eingangs schon erklärt.

Falls die Voraussetzungen für den Geographie-modus (Wertereihen mit dem Titel "Temperatur" oder "Niederschlagsmenge" müssen vorhanden sein) nicht gegeben sind, wird automatisch der Standard-Modus verwendet. Für diese Einstellung müssen sie durch klicken auf den Knopf "Übernehmen" bestätigen, dass Sie eine Änderung wünschen, da hierfür alle Werte erneut eingelesen werden müssen und es deshalb etwas dauern könnte bis die Aktion abgeschlossen ist.

Bei "Update Optionen" können Sie wählen, ob das Programm bei jedem Start automatisch prüfen soll, ob Softwareupdates vorhanden sind. Wenn Sie auf den Knopf "Jetzt auf Updates Prüfen" klicken wird das Programm überprüfen, ob es Updates gibt und gegebenenfalls die Updates Installieren und neu starten.

In der Registerkarte "Graphenbereich" haben Sie die Möglichkeit einzelne Wertereihen auszublenden, indem Sie links vor dem jeweiligen Namen das Häkchen entfernen.

Rechts haben Sie die Möglichkeit, die Interpolation (d.h. wie viele Aufgezeichnete Werte sollen zwischen gezeichneten Werten übersprungen werden) zu konfigurieren. Sie können zwischen einigen vordefinierten Werten wählen oder, wenn Sie "Benutzerdefiniert" auswählen, in das Textfeld eine positive Zahl zwischen 1 und 4.294.967.295 (der Obergrenze für Integer (=einfache zahlen) in Java) eingeben. Wenn Sie die Interpolationsstufe gewählt haben, klicken Sie bitte "OK (nur Interpolation)". Dies wird wieder alle in der Datei vorhandenen Messwerte einlesen und sichtbar machen. Dies kann, abhängig vom Aufzeichnungszeitzraum bis zu einigen Minuten dauern.

In der Registerkarte "Tastenkombinationen" können Sie die Tastenkombinationen ändern, indem Sie auf den Knopf "Ändern" hinter der jeweiligen Funktion klicken und den Anweisungen folgen.

Hinweis: Immer wenn hinter einem Knopf eine Tastenkombination in Klammern steht, können Sie diese Funktion auch durch drücken dieser Tastenkombination aufrufen.

Darüber hinaus besteht die Möglichkeit zu Zoomen. Diese Funktion kann verwendet werden, indem Sie den gewünschten Bereich, beginnend von links oben, markieren. Um heraus zu zoomen, beginnen Sie das markieren von rechts unten.

## 8.2. Gerät

Das Gerät hat zwei Displays: ein größeres Haupt-Display und ein kleineres für Einstellungen. Ferner hat das Gerät zwei Knöpfe, um die Einstellungen aufzurufen bzw. zu verändern. Wenn Sie den linken Knopf drücken, wird sich das kleinere Display aktivieren und die erste Menüseite anzeigen. Das erneute drücken des linken Knopfes wird solange auf die nächste Menüseite weiterblättern, bis alle Menüseiten angezeigt wurden. Dann wird sich das kleine Display wieder deaktivieren. Wenn Sie den linken Knopf nun nochmals betätigen, aktiviert sich das kleinere Display wieder und zeigt die erste Menüseite an. Der rechte Knopf könnte als „Ändern“ bezeichnet werden. Auf den Menüseiten wird der Knopf die jeweilige Funktion umschalten. Wenn z.B. die Menüseite „Haupt-LCD Status“ angezeigt wird, wird das drücken des Rechten Knopfes des Status des Hauptbildschirms von „AN“ nach „AUS“, oder umgekehrt, ändern.

Um Werte aufzuzeichnen, stellen Sie bitte sicher, dass Sie den gewünschten Modus (Physik/Geographie) in den Einstellungen des Geräts gewählt haben und navigieren Sie zur Menüseite „Aufzeichnung“ und starten die Aufzeichnung durch drücken des rechten Knopfes.

# 9. Quellenverzeichnis

## 9.1. In der Arbeit verwendete Quellen

Wikimedia Foundation Inc. (Hrsg.): „File Allocation Table“ in <https://de.wikipedia.org/w/index.php?title=File_Allocation_Table&oldid=168585102>, abgerufen am 16.10.2017

Arduino AG (Hrsg.): „SD Library“ in <https://www.arduino.cc/en/Reference/SD>, abgerufen am 22.10.2017

YouTube LLC (Hrsg.), upgrdman: „Graph Arduino Sensor Data with Java and JFreeChart” <https://www.youtube.com/watch?v=cw31L_OwX3A>, abgerufen am 06.11.2017

ORACLE Deutschland B.V. & Co. KG (Hrsg.):„Full-Screen Exclusive Mode” <http://docs.oracle.com/javase/tutorial/extra/fullscreen/exclusivemode.html>, abgerufen am 04.11.2017

Hackaday (Hrsg.), [Georg Krocker](https://hackaday.io/hacker/38722-georg-krocker): „MQ135 Arduino Library“ in <https://hackaday.io/project/3475-sniffing-trinket/log/12363-mq135-arduino-library> abgerufen am 05.11.2017

## 9.2. Im Code verwendete Quellen

### 9.2.1. Java

ORACLE Deutschland B.V. & Co. KG (Hrsg.): „Java Tutorials Code Sample – BoxLayoutDemo.java“ in <https://docs.oracle.com/javase/tutorial/displayCode.html?code=https://docs.oracle.com/javase/tutorial/uiswing/examples/layout/BoxLayoutDemoProject/src/layout/BoxLayoutDemo.java>, abgerufen am 06.11.2017

Stack Exchange Inc. (Hrsg.), dfa, Attila: „[How to download and save a file from Internet using Java?](https://stackoverflow.com/questions/921262/how-to-download-and-save-a-file-from-internet-using-java)“ in <https://stackoverflow.com/questions/921262/how-to-download-and-save-a-file-from-internet-using-java>, abgerufen am 06.11.2017

Stack Exchange Inc. (Hrsg.), icza: „[Loading image resource](https://stackoverflow.com/questions/9864267/loading-image-resource)“ in <https://stackoverflow.com/questions/9864267/loading-image-resource/9866659#9866659>, abgerufen am 06.11.2017

Stack Exchange Inc. (Hrsg.), davmac: „[Encoding in exported JAR](https://stackoverflow.com/questions/12920039/encoding-in-exported-jar)“ in <https://stackoverflow.com/questions/12920039/encoding-in-exported-jar>, abgerufen am 06.11.2017

Stack Exchange Inc. (Hrsg.), Telmo Marques, er.vikas:“ [Number of lines in a file in Java](https://stackoverflow.com/questions/453018/number-of-lines-in-a-file-in-java)“ in <https://stackoverflow.com/questions/453018/number-of-lines-in-a-file-in-java>, abgerufen am 24.09.2017

Stack Exchange Inc. (Hrsg.), arshajii: „[Math round java](https://stackoverflow.com/questions/13210491/math-round-java)“ in <https://stackoverflow.com/questions/13210491/math-round-java>, abgerufen am 06.11.2017

Stack Exchange Inc. (Hrsg.), oche: „[Get the video card model via command line in Windows](https://superuser.com/questions/723506/get-the-video-card-model-via-command-line-in-windows)“ in <https://superuser.com/questions/723506/get-the-video-card-model-via-command-line-in-windows>, abgerufen am 06.11.2017

BinaryTides: „How to get information about graphics card (GPU) on Linux“ in <http://www.binarytides.com/linux-get-gpu-information/>, abgerufen am 06.11.2017

Java Examples (Hrsg.): „Java String Split Example“ in <http://www.java-examples.com/java-string-split-example> , abgerufen am 06.11.2017

### 9.2.2. Arduino

Hackaday (Hrsg.), [Georg Krocker](https://hackaday.io/hacker/38722-georg-krocker): „MQ135 Arduino Library“ in <https://hackaday.io/project/3475-sniffing-trinket/log/12363-mq135-arduino-library> abgerufen am 05.11.2017

Rocket Scream Electronics: „Lightweight Low Power Arduino Library“ in <http://www.rocketscream.com/blog/2011/07/04/lightweight-low-power-arduino-library/> ,abgerufen am 06.11.2017

GitHub Inc. (Hrsg.), rocketscream, AnoxySoftware: „ATMega2560 - power\_usart3\_disable' was not declared in this scope #30“ in <https://github.com/rocketscream/Low-Power/issues/30>, abgerufen am 17.10.2017

GitHub Inc. (Hrsg.), aguegu: „readme.md“ in <https://github.com/aguegu/ardulibs/tree/master/hx711>, abgerufen am 06.11.2017

Weiterhin wurden fast alle mitgelieferten Beispiele von Arduino und den diversen Libraries verwendet, allerdings nur zum Kennenlernen der Sprache und nicht direkt im eigenen Programm

# 10. Anhang

Bild 1

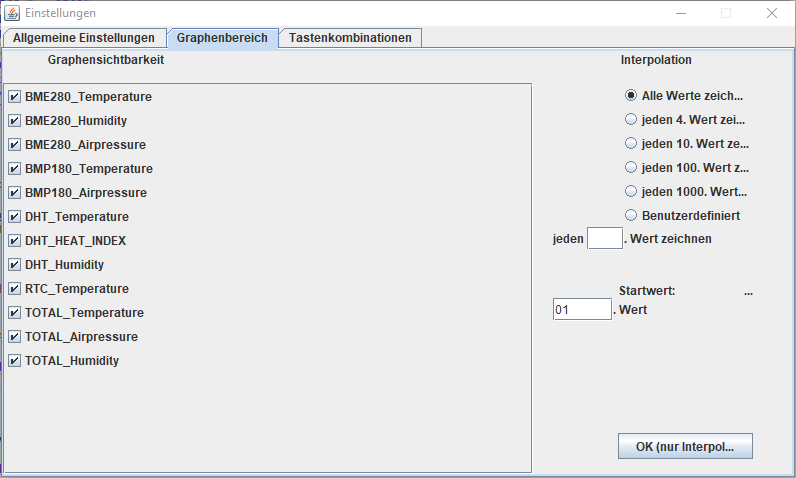


Bild 2

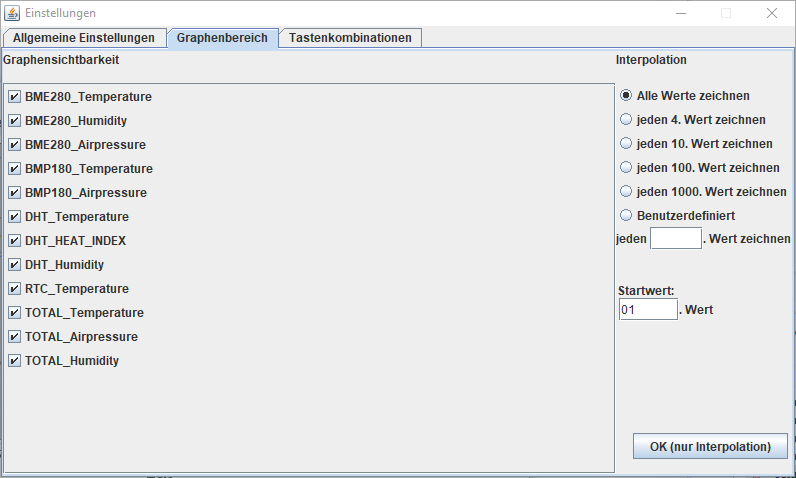


Bild 3

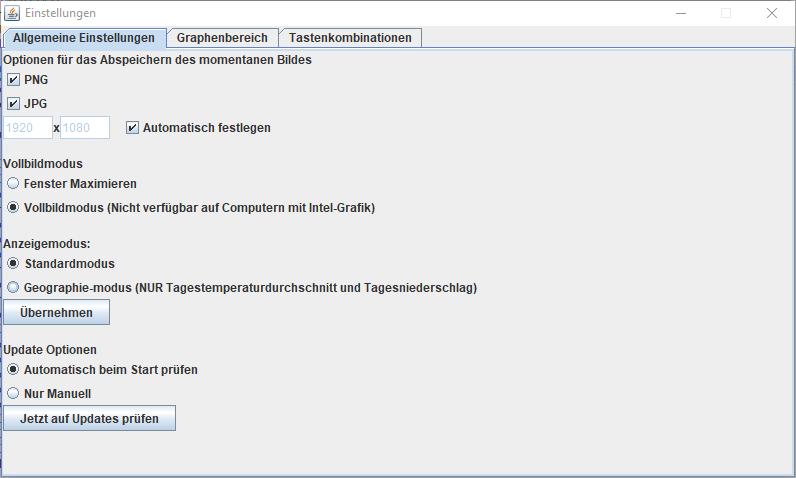


Bild 4

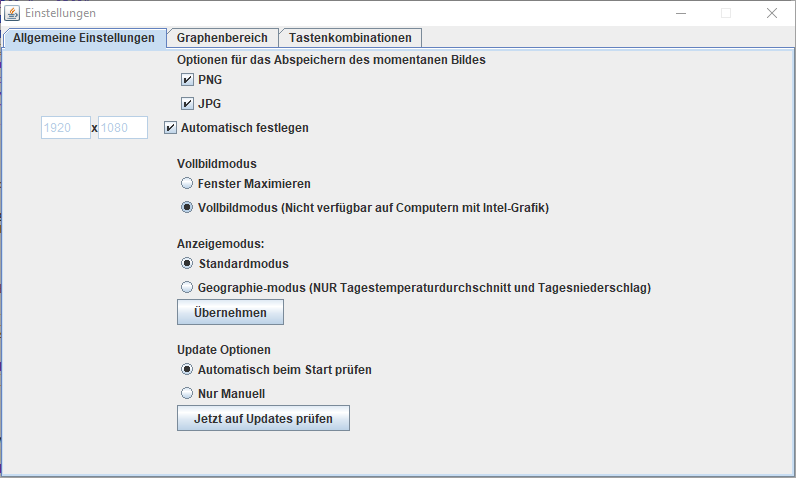


Bild 5

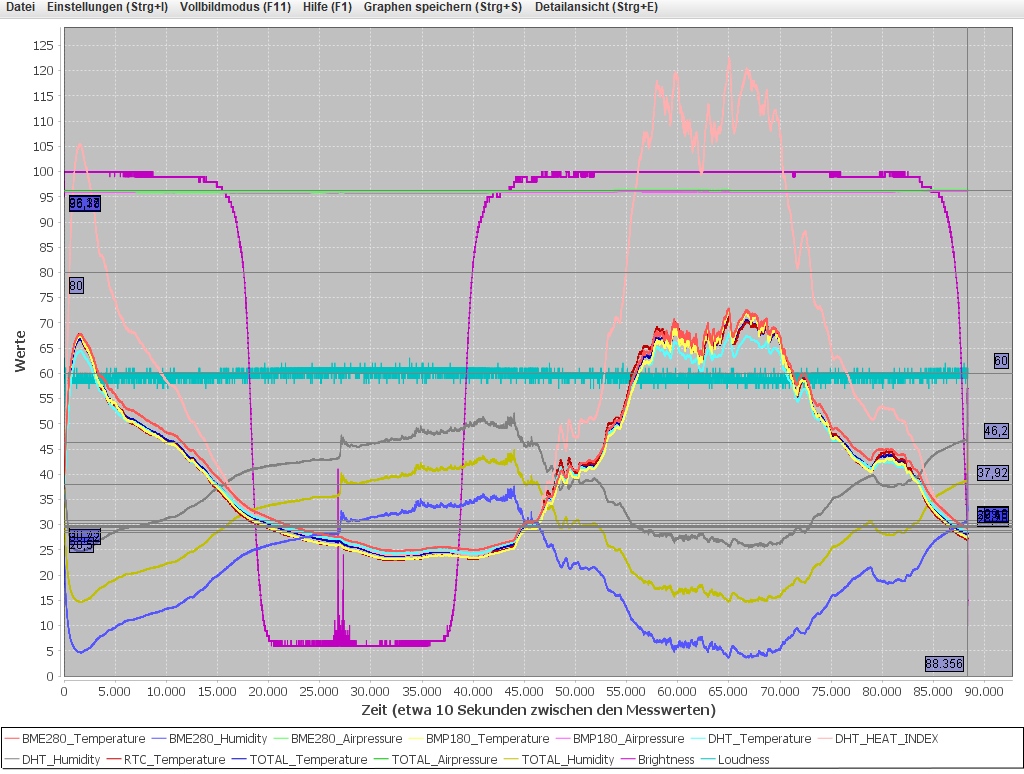
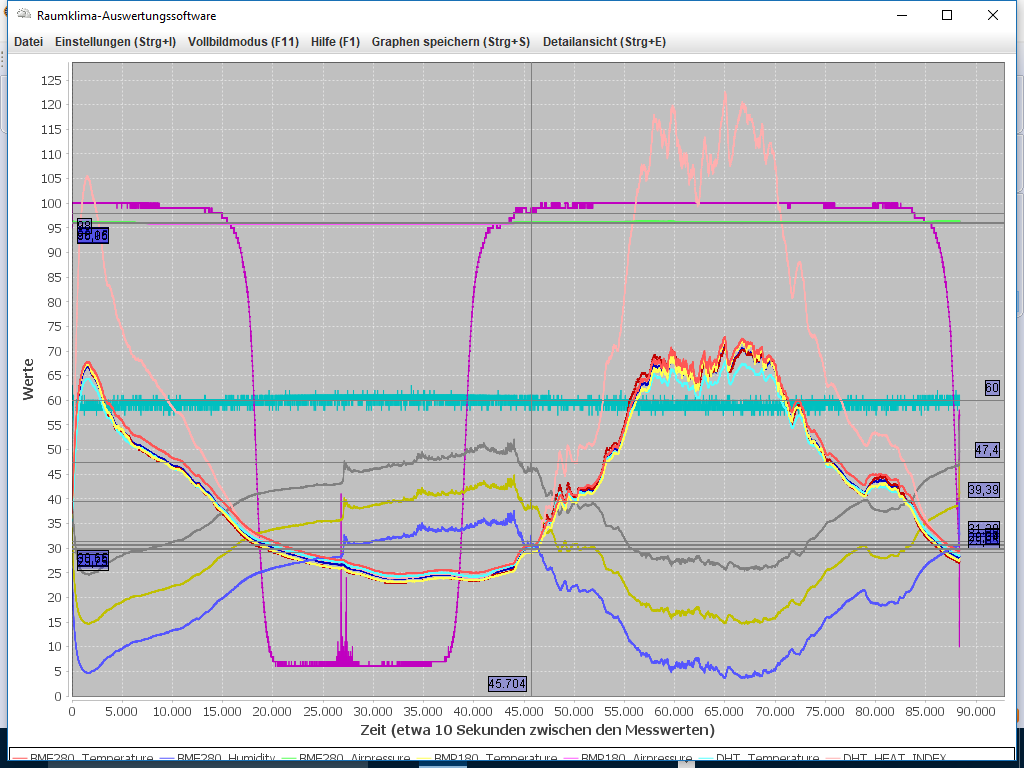


Bild 6



# 11. Erklärung

Ich erkläre hiermit, dass ich die Seminararbeit ohne fremde Hilfe angefertigt habe und nur die im Literaturverzeichnis angeführten Quellen und Hilfsmittel verwendet habe.

Dachau, den 07.11.2017

Lukas Aldersley

1. Wikimedia Foundation Inc. (Hrsg): „File Allocation Table“ in <https://de.wikipedia.org/w/index.php?title=File_Allocation_Table&oldid=168585102>, abgerufen am 16.10.2017 [↑](#footnote-ref-1)
2. Arduino AG (Hrsg.): „SD Library“ in <https://www.arduino.cc/en/Reference/SD>, abgerufen am 22.10.2017 [↑](#footnote-ref-2)
3. YouTube LLC (Hrsg.), upgrdman: „Graph Arduino Sensor Data with Java and JFreeChart” in <https://www.youtube.com/watch?v=cw31L_OwX3A>, abgerufen am 06.11.2017 [↑](#footnote-ref-3)
4. ORACLE Deutschland B.V. & Co. KG (Hrsg.):„Full-Screen Exclusive Mode” in <http://docs.oracle.com/javase/tutorial/extra/fullscreen/exclusivemode.html>, abgerufen am 04.11.2017 [↑](#footnote-ref-4)
5. Hackaday (Hrsg.) [Georg Krocker](https://hackaday.io/hacker/38722-georg-krocker): „MQ135 Arduino Library“ in <https://hackaday.io/project/3475-sniffing-trinket/log/12363-mq135-arduino-library> abgerufen am 05.11.2017 [↑](#footnote-ref-5)