Raumklima Überwachung

Ein Projekt von Jakob Kaps

1.0 Kurzfassung

In Zeiten von Corona ist Luftqualität ein wichtiges Thema geworden. In meinem Projekt „Raumklima Überwachung“ habe ich ein System entwickelt, mit welchem Daten von Raumluftsensoren graphisch veranschaulicht werden. Dazu habe ich eine Sensoreinheit entwickelt, die die Daten per W-Lan an einen Server sendet. Dieser protokolliert die Daten in einer Datenbank und bereitet sie auf einer Weboberfläche auf. Mehrere Sensoreinheiten erlauben den Überblick über die Luftqualität in verschiedenen Räumen. Die Daten werden als Graphen oder anderen Darstellungsweisen angeboten. Meine Idee stellt eine sinnvolle Alternative zu einfachen Geräten, bei welchen die Luftdaten auf einem Display ausgegeben werden, dar. Die Vorteile meiner Idee liegen darin, dass theoretisch unendlich viele Räume zentral und digital überwacht und die Daten gespeichert werden können.

2.0 Inhaltsverzeichnis

3.0 Vorüberlegungen

3.1 Ideenfindung

Als im Frühjahr des Jahres 2020 die erste Corona-Welle die Welt stilllegte, wurde viel über Aerosole und Luftqualität nachgedacht. So ist bis heute dieses Thema wichtig. Ich hatte gleich die Idee, ein Raumklima-Messgerät mit einfachen Elektronikkomponenten für das Klassenzimmer zu entwickeln. Ein Display mit einem Microcontroller und einem CO_2-Sensor war meine erste Idee. Allerdings erschien mir beim Testen des Aufbaus das Display unpraktisch, da man immer zu Diesem hingehen musste, um die Werte abzulesen. Da aber kurze Zeit später schon ein „fertiges" Raumklimamessgerät in jedem Klassenzimmer stand, verwarf ich die Idee wieder. Nach längerem Überlegen kam ich zu dem Entschluss, den Raumluftsensor beizubehalten, das Display aber durch eine zentralisierte Lösung zu ersetzen. Die Luftdaten werden nun per W-Lan vom Messgerät an einen Server gesendet und anschließend graphisch aufbereitet. Dies macht es auch möglich, die Daten vieler Sensoreinheiten, die in verschiedenen Räumen stehen, theoretisch von überall aus schnell mit jedem internetfähigen Gerät abzurufen.

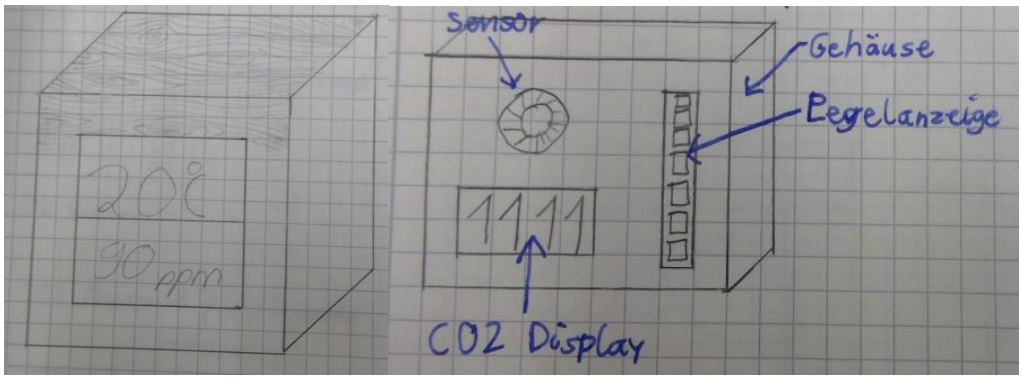

Abbildung 1 - Erste Ideen

3.2 Zielsetzung

Mein System ist dafür gedacht, die Messung der Raumluftqualität mit mehreren Sensoreinheiten zu vereinfachen. Das System könnte z.B. Anwendung in großen Gebäuden oder Veranstaltungshallen finden, da dadurch die Übersicht über die Luftqualität in mehreren Räumen gewährleistet ist. So könnte man auch eine Lüftungsanlage abhängig von der Luftqualität passend regulieren. Außerdem bietet es eine Grundlage für zukünftige Experimente unter anderem auch mit anderen Sensoren. Fragen wie „Besteht ein Zusammenhang einer Verschlechterung der Luftqualität mit bestimmten Schulfächern?" oder „Beeinflusst der CO_2-Gehalt im Klassenzimmer die Konzentrationsfähigkeit der Schüler?" könnten dadurch beantwortet werden.

4.0 Vorgehensweise, Materialien und Programmierung

4.1 Konzept und Aufbau

Das System besteht im Grunde aus vier Komponenten:

- Der Sensoreinheit, welche die Luftdaten CO_2-Gehalt, Temperatur und Luftfeuchtigkeit misst. Ein damit verbundener Microcontroller (Esp8266) sendet die Daten an den Server.
- Dem Server, bestehend aus einem Raspberry Pi mit einem Real Time Clock Modul.
- Dem Endgerät wie z.B. ein Laptop oder ein Handy zum Anzeigen der Weboberfläche, mit dem die Daten eingesehen werden können.
- Ein handelsüblicher W-Lan Router stellt die Verbindungen zwischen den Geräten her.

Die Luftdaten werden mit Hilfe des Kommunikationsprotokolls MQTT von der Messeinheit an den Raspberry Pi gesendet. Dieser speichert die Daten mit einem Python Programm in einem Datenbanksystem namens InfluxDB. Das Open-Source System Grafana veranschaulicht die gemessenen Daten auf einer Weboberfläche.

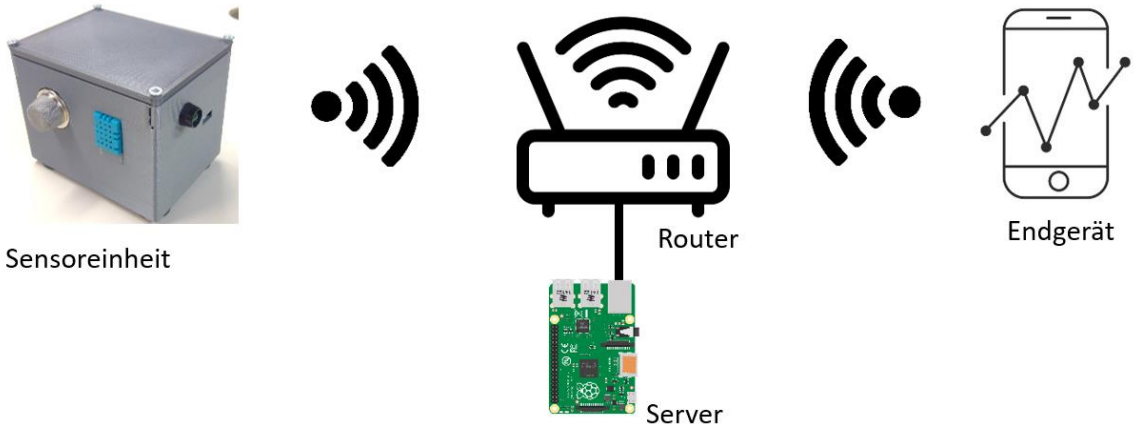

Abbildung 2 - Grundaufbau

4.2 Bau des Messgeräts

Der Bau des Messgeräts begann zuerst mit einigen Vorüberlegungen. Zum Messen der Luftdaten erschienen schließlich zwei günstige Sensoren als passend. Zum Erfassen des CO_2-Gehalts verwendete ich den Sensor MQ135, zum Messen der Lufttemperatur und Luftfeuchtigkeit den Sensor DHT11.

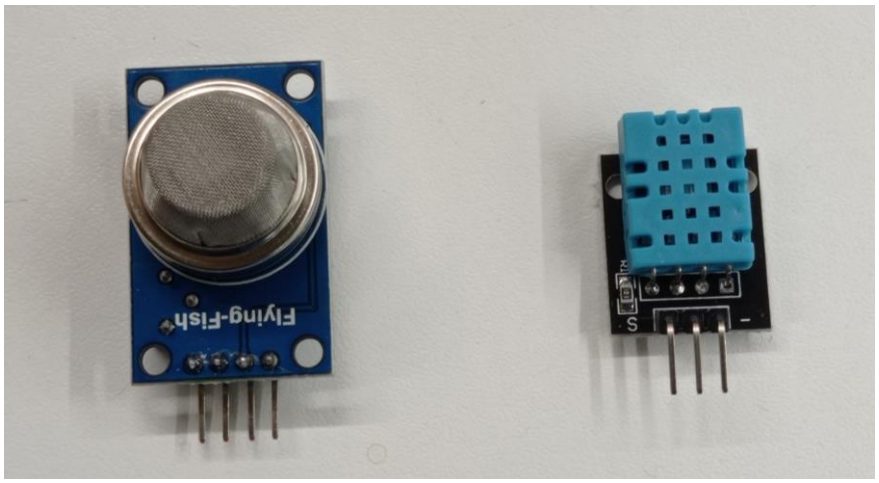

Abbildung 3 – Sensoren (links: MQ135, rechts: DHT11)

Zum Übertragen der Daten brauchte es einen W-Lan fähigen Microcontroller, um die Messwerte kabellos an den Server zu übertragen. Der Esp8266 der chinesischen Firma espressif erfüllt die-

se Anforderungen, weswegen ich diesen gewählt habe. Da die erste Version, die mir zur Verfügung stand, ein NodeMCU-Board, zu groß für mein Projekt war und ich nicht so viele Anschlüsse benötigte, entschied ich mich für den gleichen Controller, aber auf einer kleineren Platine, einem Wemos D1 mini.

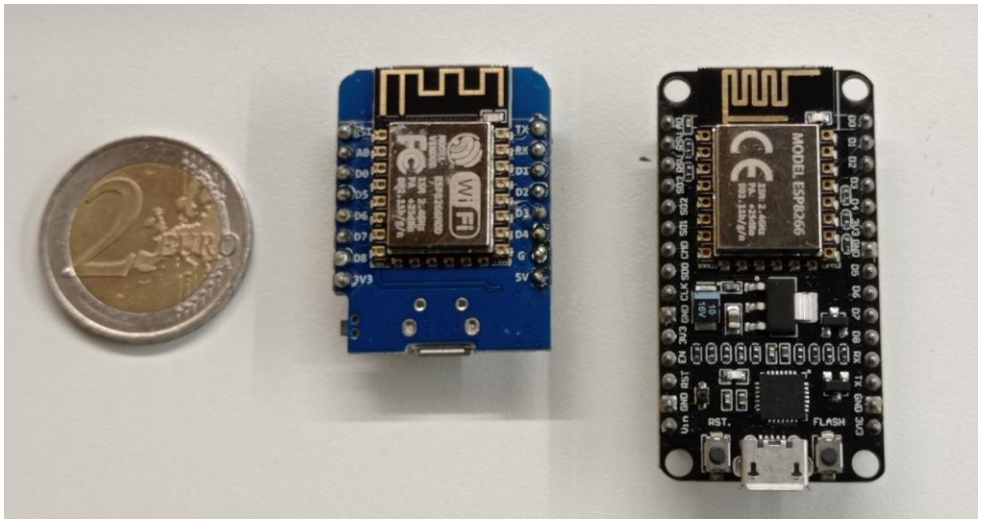

Abbildung 4 – Microcontroller im Größenvergleich mit einer Münze

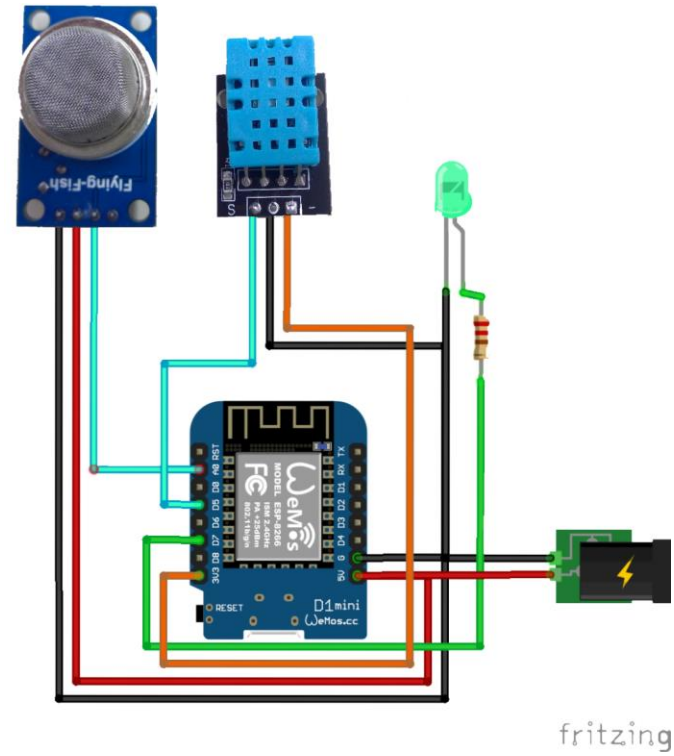

Abbildung 5 - Schaltplan

Schließlich wurde alles verkabelt und anschließend mit der Programmierungssoftware Arduino in der Arduino Programmiersprache, welche auf den Programmiersprachen C bzw. C++ basiert, programmiert. Das Programm für den Microcontroller besteht dabei aus 4 Teilen. Einem Startteil, in welchem verschiedene Softwareerweiterungen eingebunden werden und Variablen festgelegt werden. Einem Setup-Teil (Konstruktor), der den Microcontroller und die Schnittstellen für die Sensoren initialisiert. Der Hauptteil des Programms, der die Messung und Übertragung der Daten in kurzen Zeitabständen übernimmt. Außerdem gibt es am Ende noch Funktionen, um die Programmierung und die Ablaufsteuerung zu vereinfachen.

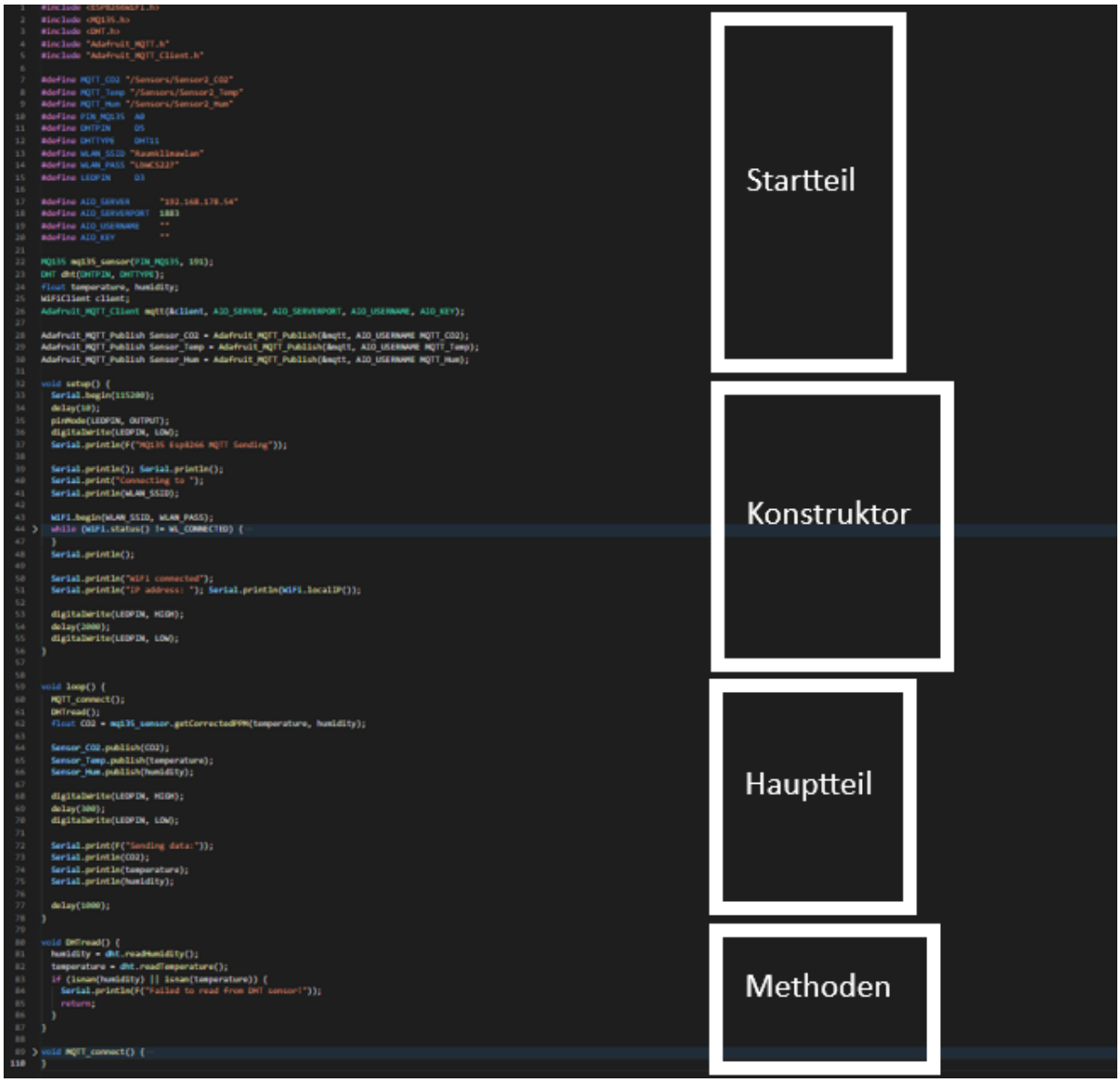

Abbildung 6 - Programmcode

Abbildung 7 - Programmierumgebung Arduino IDE

Anschließend wurde die provisorisch verkabelte Schaltung auf eine Lochrasterplatine gelötet, um am Ende nur die Kabel der Sensoren an die passenden Anschlüsse zu stecken. Dabei habe ich noch zusätzlich eine LED angeschlossen, um später den Status des Sensors am Gehäuse sehen zu können.

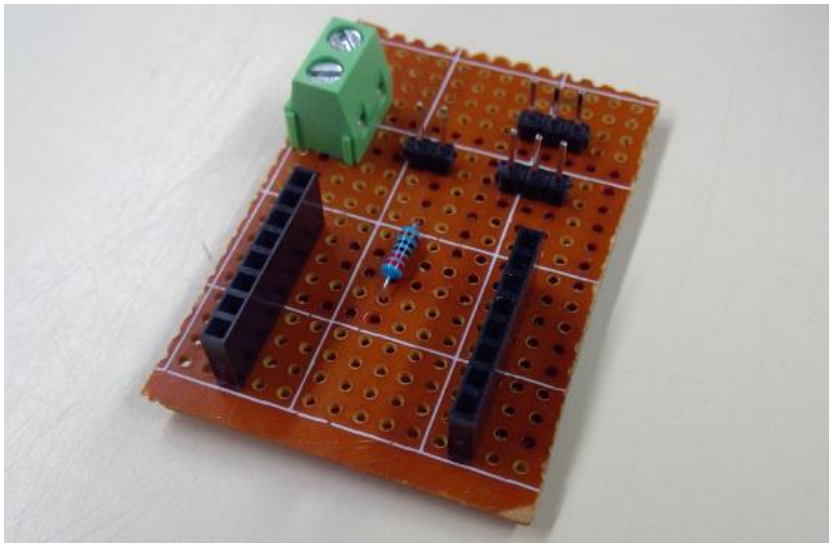

Abbildung 8 - Schaltung auf Lochrasterplatine

Die letzte Aufgabe war es, ein Gehäuse für die Sensoreinheit zu erstellen. Dafür designte ich mit der 3D-CAD-Software Freecad ein passendes Gehäuse, welches dann mit dem 3D-Drucker der Schule gedruckt wurde.

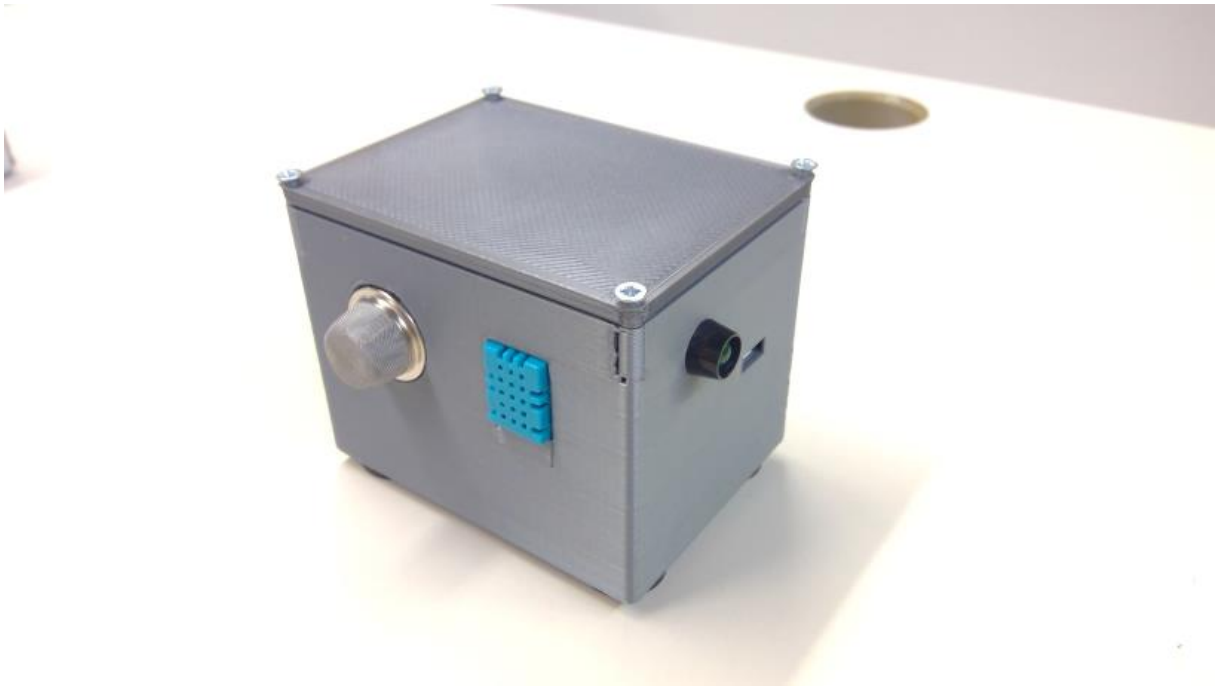

Abbildung 9 - 3D gedrucktes Gehäuse

Beim Design legte ich besonders Wert auf einige Details. Die Spannungsversorgung des Sensors sollte den heutigen Standards entsprechen, weswegen ich einen Micro-USB Anschluss verwendete. Damit kann mit jedem Handelsüblichen Micro-USB Kabel und z.B. einem Handynetzteil die Sensoreinheit betrieben werden. Außerdem wurden passende Löcher für Gummifüßchen in das Gehäuse eingearbeitet, damit bei mechanischer Belastung des Kabels das Messgerät nicht sofort vom Tisch rutscht.

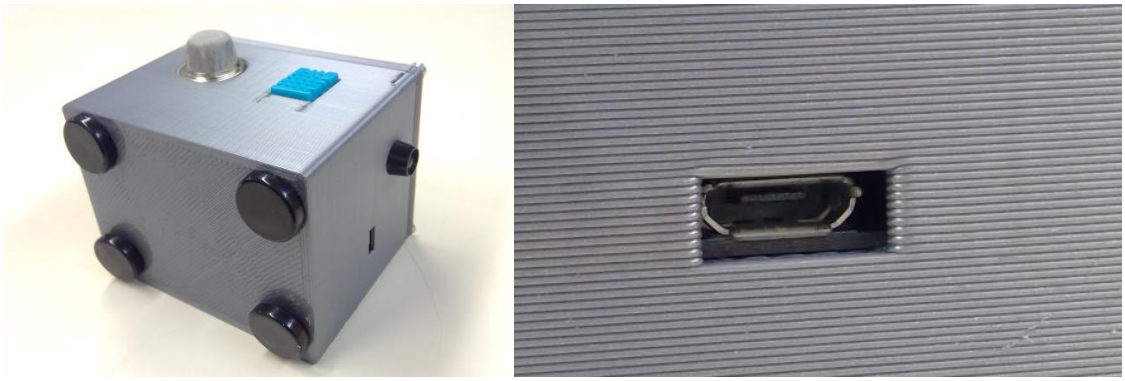

Abbildung 10 - Details am Gehäuse

4.3 Der Server

Der Server besteht aus einem Raspberry Pi 3, ein Einplatinencomputer, der per Lan-Kabel mit dem Router verbunden wird.

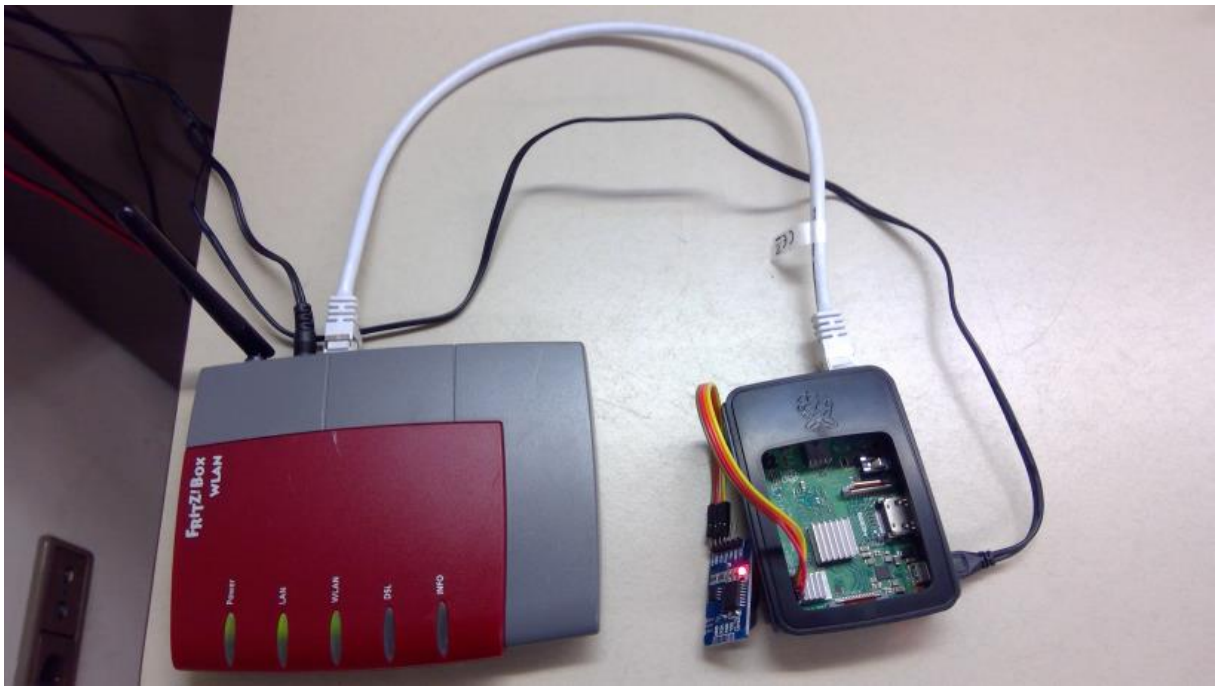

Abbildung 11 - Raspberry Pi und Router

Beim Testen des Systems ergab sich allerdings noch ein Problem. Da der Raspberry Pi kein internes Zeitmodul besitzt und auch nicht mit dem Internet verbunden war, um sich mit einem Zeitserver zu verbinden, um das aktuelle Datum und die Uhrzeit zu synchronisieren, wurden die Daten mit einem falschen Zeitstempel in der Datenbank gesichert. Die Lösung brachte ein Real-Time-Clock-Modul, welches mit einer Batterie dauerhaft als Uhr dient. Wenn der Raspberry Pi an den Strom angeschlossen wird, wird die aktuelle Uhrzeit des Moduls mit dem Raspberry Pi synchronisiert.

4.4 Programmierung

Um das System zum Laufen zu bekommen, mussten alle Teile programmiert werden. Um Daten über W-Lan an einen Minicomputer senden zu können, nutze ich das einfache Netzwerkprotokoll MQTT. MQTT (ursprünglich *MQ Telemetry Transport*) erlaubt es mir, kleine Datenmengen von einem Publisher (dt. Herausgeber bzw. Sender) über einen Broker (dt. Vermittler) an einen Subscriber (dt. Abonnent bzw. Empfänger) zu senden. Der Raspberry Pi dient als Broker. Ein selbstentwickeltes Python Programm, das auch auf diesem läuft, wirkt als Subscriber.

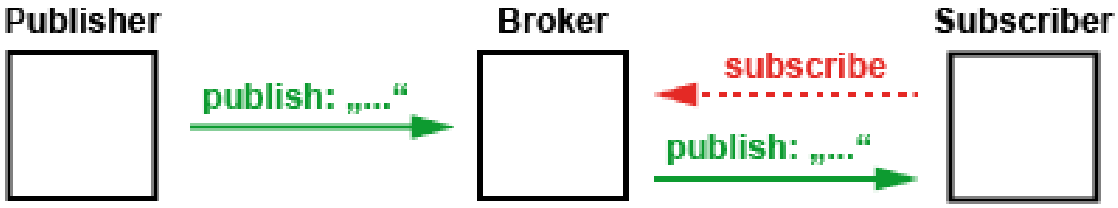

Abbildung 12 - MQTT Struktur

Für den Broker habe ich die Software Mosquitto verwendet, die mit wenigen Befehlen über das Terminal des Servers installierbar und steuerbar ist. Dazu installierte ich noch die Datenbanksoftware InfluxDB, ein Open Source Datenbankmanagementsystem, das die Messdaten der Luftsensoren in verschiedene Tabellen mit Zeitstempel abspeichert. Nun musste nur noch ein Programm, das die Daten in der Datenbank speichert, entworfen werden. Dies habe ich in der Programmiersprache Python geschrieben. Alle Programme sind hier https://github.com/arduinoms/Raumklima-Ueberwachung abrufbar.

4.5 Grafana

Grafana ist ein eine Open-Source Software, welche zum Veranschaulichen und Visualisieren von Daten entwickelt wurde. Für mein Projekt war dies also bestens geeignet, um die Luftdaten in unterschiedlichen Darstellungsweisen anzuzeigen. Die Daten konnte ich so z.B. in Form von Graphen oder auch die aktuelle Luftqualität in einfachen Zahlen veranschaulichen.

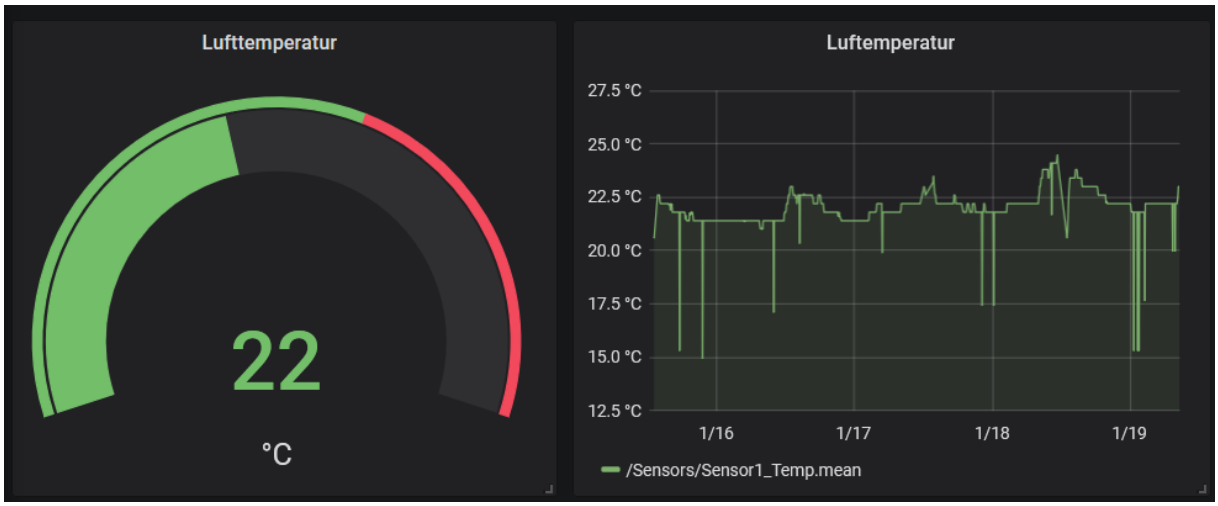

Abbildung 13 – Darstellungsweisen mit Grafana

5.0 Test des Aufbaus

Nach dem Bau des Systems folgte der erste Test. In einem Klassenzimmer, welches nicht über ein Fenster durchlüftet werden kann, stellte ich den Sensor auf, in einem Nebenraum den Router und Server. Anschließend startete ich alles und ließ das System drei Tage lang Daten aufzeichnen. An zwei Tagen fand im Klassenzimmer Unterricht statt, an einem nicht. Dieser unterrichtsfreie Tag dient dabei als Referenz.

Abbildung 14 – Raumluftdaten von drei Tagen

Abbildung 15 - Auswertung von drei Tagen

Hierbei lassen sich Zusammenhänge erkennen: So steigt während dem Unterricht der CO_2-Gehalt der Luft, die Luftfeuchtigkeit und die Lufttemperatur an. Außerdem lässt sich erkennen, dass die Werte außerhalb der Unterrichtszeiten im Klassenzimmer bis zum nächsten Schultag wieder sinken, da die Lüftungsanlage permanent läuft.

6.0 Zusammenfassung und Ausblick

Insgesamt funktioniert das System, hat aber noch Verbesserungspotential. So könnte ich noch mehrere Sensoreinheiten gleicher Bauart bauen und diese unter verschiedenen Bedingungen testen. Außerdem gäbe es noch viele interessante Möglichkeiten in der Visualisierungssoftware Grafana. Andere Darstellungsweisen oder auch Automatisierungen, z.B. Steuerung der Lüftungsanlage wären möglich. Auch eine konkrete Warnung des Systems bei zu hohen Luftwerten an die Lehrkraft wäre denkbar.

In diesem Projekt habe ich persönlich viel über Elektronik, Programmierung und 3D-Druck gelernt.

7.0 Quellen

https://wikipedia.org/wiki/MQTT

https://de.wikipedia.org/wiki/Raspberry_Pi

https://de.wikipedia.org/wiki/ESP8266

https://de.wikipedia.org/wiki/FreeCAD

https://www.elektronik-kompendium.de/sites/raspberry-pi/2709041.htm

https://de.wikipedia.org/wiki/Grafana

https://de.wikipedia.org/wiki/Arduino_(Plattform)

Bilder:

https://www.elektronik-kompendium.de/sites/net/2204051.htm

privat

8.0 Dank

Ein Dank geht an alle, die mich bei diesem Projekt unterstützt haben. Besonders möchte ich mich bei meinem Projektbetreuer Andreas Fromm bedanken, der mir bei vielen Problemen weiterhalf.