Ein Feinstaub-Messgerät auf Arduino-Basis für die Feinstaubklassen PM10, PM2.5 und PM1.0

Bernd Laquai, 23.12.2016, Update 30.12.16

Die dramatische Zunahme der Luftverschmutzung in China, vor allem auch in den High-Tech Metropolen von Shanghai und Shenzen hat dazu geführt, dass die lokale Elektronik- und Halbleiter-Industrie relativ schnell den Markt für Luftqualitäts-Sensoren erkannt und mit Hochdruck daran gearbeitet hat, eine Technologie zu entwickeln, mit der man das Problem zumindest einmal elektronisch messbar machen konnte. Auch die lokalen Universitäten erkannten den Forschungsbedarf und halfen kräftig mit, so dass China heute wohl die Nummer Eins im Bereich der elektronischen Sensorik zur messtechnischen Erfassung der Feinstaubproblematik sein dürfte. Es entstanden Firmen wie Plantower und Nova Fitness, die in der Zwischenzeit wohl den größten Marktanteil an den sehr kostengünstigen Feinstaubsensoren, die in der Lage sind, die anspruchsvollen Partikelgrößen von PM2.5 und PM1.0 (PM=Particulate Matter= Feinstaub) zuverlässig zu messen, unter sich aufteilen. Eingesetzt werden diese Sensoren vor allem in Klimaanlagen und Luftreinigern. Der chinesische US-Markt bietet aber auch Einsatzmöglichkeiten als reines Messgerät, denn die Feinstaubanzeige in Wohnräumen ist in diesen Großstädten fast schon so populär wie die Anzeige der Temperatur. Ob die die Politik und die Wirtschaft allerdings aus diesen Messdaten Konsequenzen ziehen wird ist zwar noch fraglich aber die Bevölkerung hat schon mal ein Gefühl für die Einheit der Feinstaubkonzentration in ug/m^3 Luft und so wir in Deutschland über die Mitnahme eines Regenschirms nachdenken, wenn die elektronische Wetterstation Regen anzeigt, so wird der Bewohner in Shanghai eben über die Feinstaubmaske nachdenken, die er mitnehmen könnte, wenn die PM2.5 Anzeige Werte mehr als 100ug/m^3 signalisiert.

In Europa scheint man sich dagegen derzeit mehr mit den Folgen der Feinstaubproblematik zu beschäftigen und der Frage welche Grenzwerte die richtigen sind. Wie man das Problem messtechnisch flächendeckend und für die Bevölkerung erschwinglich erfassen kann, scheint weniger das Thema zu sein. Vielleicht will man ja auch gar nicht, dass die Bevölkerung so viel selbst misst, ähnlich wie beim Geigerzähler, das könnte ja schlafende Hunde wecken, so dass die Politik viel zu schnell unter Druck kommen könnte und der geliebten Automobil-Industrie oder den Flotteninhabern großvolumiger Diesel-Fahrzeuge unangenehme Auflagen machen müsste, weil zu viele Bürger entsprechende Maßnahmen einklagen. Von daher hat man es mit der Messtechnik vermutlich nicht allzu eilig und so wundert es auch nicht wie eine Feinstaub-geplagte Großstadt wie Stuttgart gerademal die Daten von zwei Messstationen öffentlich macht.

Dass dies nicht so sein müsste, merkt man schnell, wenn man sich einmal mit der in der Zwischenzeit zur Verfügung stehenden chinesischen Sensor-Technologie befasst. Fast ausnahmslos arbeiten die chinesischen Sensoren nach dem Laser-Streulicht-Verfahren, so dass anhand des Streuwinkels auch eine Klassifikation der Partikelgröße möglich ist. D.h. diese Sensoren liefern in der Regel mehrere Ergebnisse für die jeweiligen aerodynamischen Partikeldurchmesser. Hinsichtlich der gesundheitlichen Risiken ist ja mittlerweile bekannt, dass die Fraktion der Feinstaubteilchen mit weniger als 2.5um an aerodynamisch wirksamem Durchmesser die Entscheidendere ist, auch wenn die amtlichen Messstellen in der Regel

bisher nur PM10 (also Konzentration an Partikeln kleiner 10um) veröffentlichen. Von daher ist es wichtig, dass der Sensor auch einen Wert für PM2.5 liefert. Die meisten chinesischen Sensoren können das und liefern meist sogar noch einen Wert für PM1.0. Allerdings nimmt unter 2.5um die Detektionswahrscheinlichkeit ab und in der Regel hört sie dann bei 0.3um ganz auf. Das heißt die Ultra-feinen Partikel mit < 0.3um sind im Wert für PM2.5 nicht mehr enthalten. Aber damit lässt sich doch schon einiges anfangen. Und es ist eben nicht wahr, dass man zur Erkennung einer Situation mit hoher Feinstaubkonzentration ein tausende Euro teures Equipment benötigt. Gerichtsfeste quantitative Werte wird so ein Sensor allerdings nicht liefern, aber das wird auch von keinem Thermometer oder Barometer verlangt, den man im Kaufhaus kaufen kann.

Die chinesischen Sensoren haben meist auch etliche Intelligenz im Sensor, in der Regel immer einen Mikrokontroller. Dieser macht die Größenklassifikation, hat in der Regel gewisse Kalibrierdaten geladen und rechnet dann von Zählraten pro Partikelvolumen in Massenkonzentrationen in ug/m^3 um. Zudem bedient er die elektrische Schnittstelle, die in entweder eine UART (asynchrone serielle Schnittstelle, wie RS232) oder eine Pulsweitenmodulation je nach Konzentrationswert oder gar ein analoges Spannungssignal sein kann. Die UART läuft in der Regel mit dem Standard-Setup von 9600 Bit/s, 8 bits ohne Parity und mit einem Stoppbit. Gesendet wird in der Regel eine Sequenz von wenigen Bytes mit binärem Inhalt, die mit einem oder zwei Startbytes beginnen, also kein reiner ASCII Text. Diese Sequenz wird typischerweise einmal pro Sekunde gesendet. Bevorzugt man ein analoges Signal, dann kann man das PWM-Signal (verfügbar an einem Pin pro Größenklasse), dass die Pulsbreite eines Rechtecksignals entsprechend der PM-Konzentration variiert, auf einen Tiefpass geben und mit einem AD-Wandler oder Komparator auswerten. Die Grenzfrequenz des Tiefpasses sollte um zwei Größenordnungen tiefer liegen, als der Kehrwert der PWM-Periode. Dieser Tiefpass wandelt dann das PWM-Signal in ein Analogsignal um, welches den Momentanwert der Feinstaubkonzentration wiederspiegelt.

Die Sensoren sind prädestiniert dafür um mit einem weiteren Mikrocontroller kombiniert zu werden, der die weitere Signalauswertung, eventuell eine Steuerung oder auch eine Messwertanzeige macht. Den analogen Wert kann man direkt über einen Analog-Eingang mit minimalem Aufwand einlesen, die digitale Sequenz kann man über eine serielle Schnittstelle eines Mikrocontrollers mit nur wenig mehr Aufwand auswerten, muss dann aber die empfangenen Bytes zu einzelnen Messwerten in gängigen Zahlenformaten (float oder int) zusammensetzen, was aber auch keinen größeren Aufwand darstellt.

Da eine einfache Mikrocontrollerplatine von heute auch kein Problem hat, gleichzeitig ein Display anzusteuern hat man also mit einem Sensor, einem Arduino oder Raspberry Pi oder ähnlichem Mikrocontroller und einem alphanumerischen Display ruck zuck ein Feinstaubmessgerät zusammengesteckt, dass eine enorme Empfindlichkeit und brauchbare Genauigkeit erreicht. Es muss allerdings beachtet werden, dass diese Sensoren in der Regel für Klimaanlagen mit temperierter und entfeuchteter Luft bzw. für Innenraumanwendungen gedacht sind. Schaut man in das Datenblatt findet man die Angaben:

Operating temperature range: -20 … +50°C

Working humidity range: 0~80% rel. Feuchte

Während der Temperaturbereich noch für die meisten Außenanwendungen geeignet wäre, muss klar sein, dass die Messwerte sehr fraglich werden, wenn die relative Luftfeuchte über 80% steigt. Und genau das hat man vor allem bei Messungen im Freien schnell. Man muss also aufpassen, nicht, dass man die ebenfalls Mikrometer großen Nebeltröpfchen mit Feinstaub gleichsetzt. Aus diesem Grund empfiehlt es sich grundsätzlich mit dem Mikrocontroller gleichzeitig auch einen kombinierten Temperatur- und Feuchtesensor auszulesen, so dass man parallel feststellen kann, ob der Sensor innerhalb der Spezifikationen betrieben wird. Um den Verdrahtungsaufwand zu minimieren, empfiehlt es sich bei dem Temperatur-/Feuchte-Sensor genau wie beim Display eine Version mit I2C-Bus zu verwenden. Im Innenraum oder bei genauer Kenntnis von Temperatur und Feuchte im Freien kann man sich natürlich den Temperatur-/Feuchte-Sensor sparen.

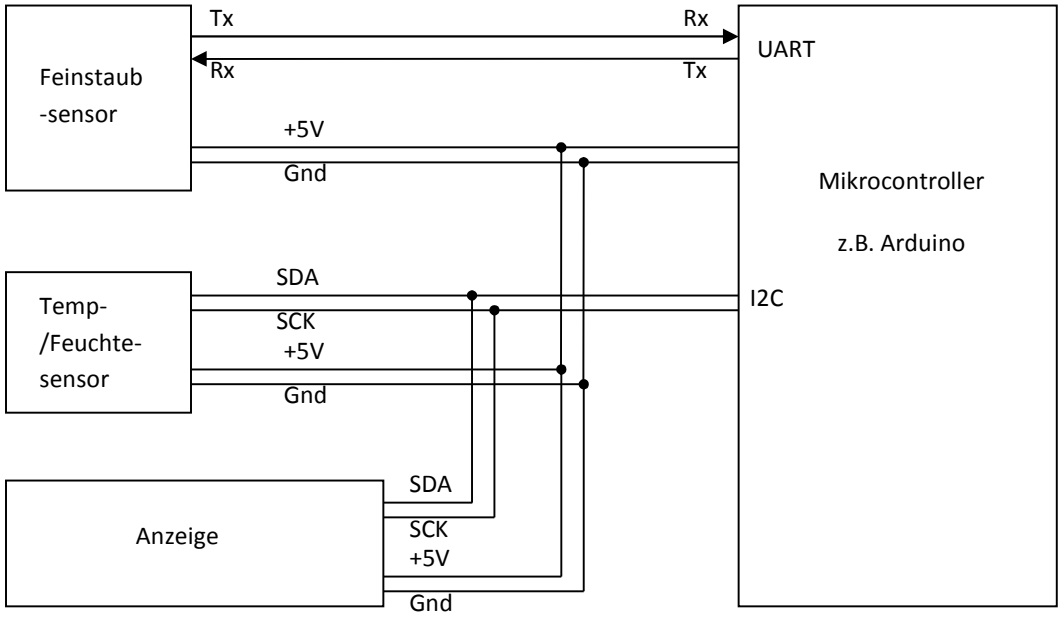

Abb. 1: Grundsätzliche Architektur eines einfachen Feinstaub-Messgeräts

Noch viel geringer wird der Aufwand ein Feinstaub-Messgerät zu bauen, wenn man bei einem Zubehör-Shop für Roboterzubehör einen Feinstaubsensor findet, der dann meist mit einer kleinen Break-Out Adapter-Platine und der passenden Library für einen Mikrocontroller geliefert wird. Der Wert der Break-Out Platine besteht in der Regel nur in der Anpassung der verschiedenen filigranen Stecker-Systeme und in einer kleinen Aktivitätsanzeige mit superkleinen SMD-LEDs. Der etwas größere Wert besteht in der Bereitstellung von fertigen Routinen, welche die Umwandlung der binären Information, welche auf der seriellen Schnittstelle empfangen wird, bewerkstelligen. Sie wandeln die Binärinformation in Werte um, welche die Feinstaubkonzentrationen der unterschiedlichen Größenklassen als Dezimalzahlen darstellen, mit denen man rechnen kann oder die man sehr einfach zur Anzeige bringen kann.

Ein Roboter-Zubehör-Shop, der einen solchen chinesischen Feinstaubsensor im Angebot hat ist DfRobot, der sein Headquarter ebenfalls in China hat. Im Gegensatz zum Einkauf über elektronische Marktplätze wie Aliexpress, Ebay oder Amazon, bei denen die Logistik und die Kaufabwicklung meist nur von ein-Mann-Firmen gemacht wird, mit der Folge von langen Lieferzeiten und teilweise auch Qualitätsproblemen, bekommt man bei DfRobot die Ware meist innerhalb von einer Woche mit DHL und kann sich auch sicher sein, dass es sich um unbenutzte Neuware handelt. Außerdem muss man nicht seine Kreditkartendaten übers Internet schicken, sondern kann mit Paypal bezahlen.

DfRobot bietet für seine Gravity Serie unter der Bestell-Nr. SEN0177 derzeit den Feinstaubsensor vom Typ „HK-A5 Laser PM2.5/10 Sensor“ an. Der auf dem Datenblatt im Wiki angegebene Hersteller ist bjhike.com, der auch noch andere Lösungen im Bereich Luftqualität liefert. Mit hoher Wahrscheinlichkeit stammen die Innereien des Sensors bzw. die Lizenz von Plantower, dem größten chinesischen Anbieter solcher Sensoren. Diese Annahme ist dadurch begründet, dass die technischen Daten und das Kommunikationsprotokoll identisch sind zu dem des kleinsten Sensors PMS1003 von Plantower. D.h. mit der von DfRobot zur Verfügung gestellten Beispielprogramm lässt sich auch ein Sensor vom Typ PMS1003 betreiben. Wenn man die Bildergalerie bzw. das Wiki bei DfRobot zu dem Produkt genauer anschaut, ist auch nicht der im Datenblatt gezeigte Sensor zu sehen, der heute auch geliefert wird, sondern das Vorgängermodell, der PMS1003, ganz wie er auf der Webseite von Plantower zu sehen ist. Und interessanterweise ist der PMS1003 auch genau derjenige Sensor, der in dem derzeit sehr stark beworbenen, kommerziellen Feinstaub-Messgerät für den Innenraum „LaserEgg“ (derzeit > 150 Euro) zu finden ist. Der HK-A5 ist dagegen etwas kleiner gebaut und ist aufgrund der etwas einfacheren rechteckigen Bauform auch leichter in ein Gehäuse zu integrieren als der PMS1003.

Um nun mit dem Feinstaubsensor von DfRobot ein Feinstaub-Messgerät aufzubauen ist denkbar einfach. Dazu braucht man lediglich noch eine Arduino-Mikrocontroller-Platine (ein Arduino-Uno reicht völlig). Den minimalsten Aufwand hat man dabei, wenn man direkt das Beispielprogramm, das man auf der DfRobot Produkt-Webseite herunterladen kann, verwendet. Man verkabelt den Sensor wie im Tutorial angegeben mit der Break-Out Adapter-Platine und der Arduino-Platine, kompiliert das Programm und lädt den Code auf die Arduino-Platine hoch. Dabei ist zu beachten, dass während des Hochladens das Kabel für Tx und Rx zum Break-Out Board nicht eingesteckt sein darf. Diese Verbindung darf erst dann am Arduino angeschlossen werden, wenn der Ladevorgang vollständig abgeschlossen ist. Hintergrund ist, dass die Ausleseroutine aus dem Beispielprogramm die Hardware Serial Schnittstelle verwendet, die auch zum Programmieren benötigt wird. Mit dem in der Entwicklungsumgebung vorhandenen seriellen Editor oder jedem anderen Monitor-Programm für eine serielle COM-Schnittstelle (z.B. HTerm) erhält man nach dem Herstellen der Verbindung nun periodisch im Sekundentakt die 3 Massenkonzentrationswerte für die Feinstaubklasse PM10, PM2.5 und PM1.0 auf dem Computer angezeigt.

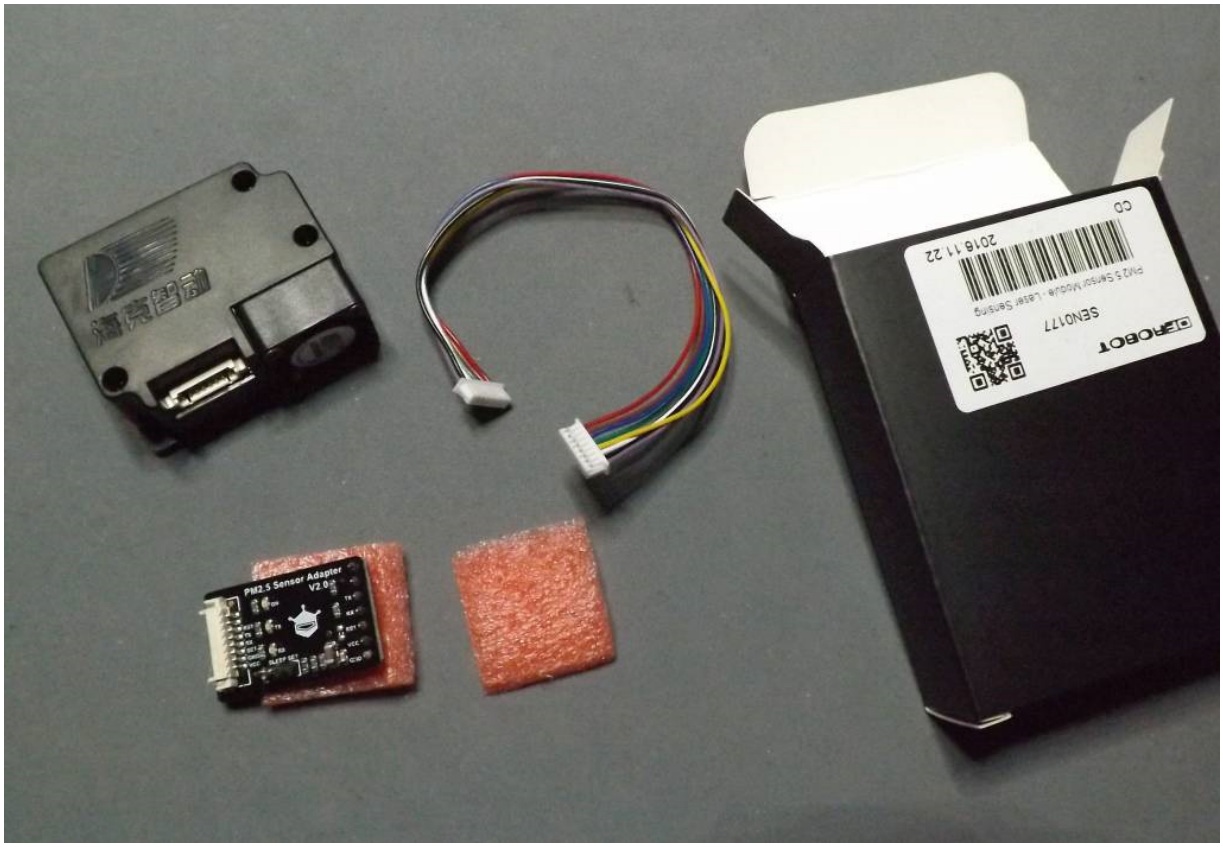

Abb. 2: Der Feinstaubsensor „HK-A5 Laser PM2.5/10“ von DfRobot mit Break-Out Adapter-Platine

Ein etwas handlicheres Messgerät, bei dem schließlich der Computer oder Laptop als Aufwand zur Anzeige wegfällt, erhält man, wenn man an die Arduino-Platine zusätzlich ein kleines Display anschließt. Hierzu eignet sich beispielsweise ein 4-zeiliges LCD-Display mit I2C Bus-Schnittstelle. Diese Schnittstelle benötigt lediglich eine Stromversorgung (5V und Masse) sowie die Datenleitung SDA und die Taktleitung SCL. Diese Art von LCD-Display gibt es von einigen Herstellern unter anderem ebenfalls von DfRobot (I2C TWI LCD2004) zusammen mit einer LCD-Display Bibliothek für den I2C-Bus.

Leider zwingt uns die Verwendung des I2C-LCD-Display derzeit noch zur Verwendung einer älteren Arduino IDE z.B. 1.5.2, denn bei den neueren IDE-Versionen 1.6.x und 1.8.x scheint es noch Probleme mit der mittlerweile doch schon betagten LCD-Bibliothek zu geben. Die älteren Varianten der Arduino Entwicklungsumgebung bekommt man aber auf der Arduino-Softwareseite unter „ältere Versionen“ ganz problemlos. Am günstigsten ist es dabei, die alte Entwicklungsumgebung nicht per Windows-Installer zu installieren, sondern nur das zip-Archiv an einer geeigneten Stelle auszupacken und dann das lokale Arduino-Executable auszuführen. So kann man leichter mehrere Versionen der Entwicklungsumgebung parallel halten. Bei der Popularität der I2C-LCD-Displays ist aber damit zu rechnen, dass es auch bald ein Update der alten I2C-LCD Bibliothek geben wird.

Ein wesentlicher Aspekt für das Auswerteprogramm ist, dass der Sensor selbst einen Mikrocontroller beinhaltet, der autonom Binärdaten im Sekundentakt über eine serielle Schnittstelle sendet. Von daher ist zunächst die wichtigste Aufgabe, die es zu lösen gilt, eine Synchronisation zu erreichen. Die Daten

werden nach einem bestimmten Protokoll gesendet, das eine Framestruktur erzeugt. Ein Frame beginnt immer mit dem Byte 0x42 gefolgt von 0x4d. Dann folgen zwei Bytes, welche die Länge des Frame-Inhalts von 28 Bytes kennzeichnen, und daher immer 0x0 und 0x1c sind. Zudem wird in den letzten zwei Bytes eine Quersumme aus allen vorigen Bytes, gebildet, so dass man auch die korrekte Übertragung und Auswertung der Framestruktur überprüfen kann. Auf diese Framestruktur kann daher auch synchronisiert werden, wenn man auf die fehlerfreie Auswertung prüft.

Die Synchronisation wird so gemacht, dass zunächst mit incomingByte = Serial.read() byteweise gelesen wird, was gerade auf der seriellen Schnittstelle des Arduino eintrifft. Wenn das Byte 0x42 im Datenstrom auftaucht, geht man davon aus, dass dies einen Frame-Beginn kennzeichnet, obwohl es ja auch ein Datenbyte sein könnte. Dann liest man die nächsten 31 Bytes in den Framepuffer der Länge 31 ein und testet an der Stelle Null des Pufferspeichers ob hier jetzt das zweite Framestart-Byte 0x4d steht und ob an der übernächsten Stelle die korrekte Framelänge von 28 (0x1C) auftaucht. Ist das der Fall, geht man davon aus, dass man den Frame richtig erwischt hat.

Von zwei Bytes, die einen Wert darstellen, wird immer das höherwertige Byte zuerst gesendet und dann das niederwertige. Um das höherwertige und das niederwertige Byte, das man aus einem Pufferspeicher liest, zu einer Zahl zusammen zu setzen, kann man sehr vorteilhaft vom Links-Schiebe-Befehl „<<“ Gebrauch machen. So schiebt z.B. der Befehl x = (buf[i]<<8) die Bits des Byte an der Stelle i in dem unsigned char Pufferspeicher buf um 8 Bits nach links (was einer Multiplikation mit 256 entspricht). Um nun die passende Integer Zahl x für einen zwei-Byte Wert zu erhalten, muss man das niederwertigere Byte, ohne es zu verschieben, noch zum verschobenen höherwertigen Byte dazu zählen, also x = ((buf[i]<<8) + buf(i+1)). Die Klammern sind zwingend nötig, sonst hat der Compiler mit der Interpretation der Verschieboperation Probleme. Diese Methode der Auswertung wird bei den Daten und der Quersumme eingesetzt. Danach prüft man den Pufferspeicher-Inhalt auf die Quersumme, und wenn die stimmt, werden die Daten als korrekt angesehen, ansonsten wird der Pufferspeicher-Inhalt verworfen und eine entsprechende Fehlermeldung ausgegeben, bevor der nächste Synchronisationsversuch beginnt.

Gelingt die Synchronisation, werden die Werte der verschiedenen PM-Konzentrationen in ug/m^3 als Zahlen zusammengesetzt und die 3 Anzeigestrings gebildet. Nach dem Löschen der vorherigen Anzeige und der entsprechenden Positionierung des Cursors in der Anzeige, werden die Werte zur Anzeige gebracht. Wenn keine Übertragungsfehler auftreten, dann bleibt der Lesevorgang beim nächsten Schleifen-Durchlauf in Sync, d.h. die Vergleiche liefern sofort logisch wahr und die Anzeige wird so im Sekunden-Takt aktualisiert.

Hat man die Software am Laufen, kann man Sensor, Anzeige und Arduino in ein passendes Gehäuse einbauen. Allerdings empfiehlt es sich darauf zu achten, dass entweder ein elektrisch leitfähiges Gehäuse (z.B. Alu) oder ein Gehäuse aus elektrisch leitfähigem Kunststoff verwendet wird. Als Notlösung ist es aber auch möglich, das Gehäuse mit einem leitfähigen Lack einzusprühen. Der Hintergrund ist, dass Feinstaub eine elektrische Ladung tragen kann und sich dann am eventuell statisch aufgeladenen Gehäuse ablagert und so dem Luftstrom entzogen wird (Plate-Out Effekt). Ferner sollte man den Sensor bündig mit der Lufteinlassseite ins Gehäuse montieren, so dass er nur Außenluft und keine falsche Luft von innen ansaugt. Zudem sollte man einige Luftaustrittsöffnungen auf der gegenüberliegenden Seite anbringen, die groß genug sind, dass alle angesaugte Luft auch ungehindert wieder austreten kann. Im

Innern des Gehäuses sollte die Luft einigermaßen wirbelfrei zum Austritt strömen können, so dass kein Staudruck entsteht, der die Strömungsgeschwindigkeit reduzieren könnte.

Für die Stromversorgung muss beachtet werden, dass der Sensor nicht gerade wenig Strom benötigt (laut Spezifikation sind es 120mA). Daher ist eine Batterie-Lösung nicht so sehr sinnvoll. Es kann aber sehr einfach ein Modellbau-LiPo-Akku (3S1P mit 11.1V) als Stromquelle verwendet werden. Der Vorteil dieser Lösung ist, dass diese Akkus klein und leicht sind, eine hohe spezifische Kapazität haben und es aus dem Modellbaubereich auch kleine Ladegeräte mit Balancer-Schaltung für ein schonendes Laden gibt.

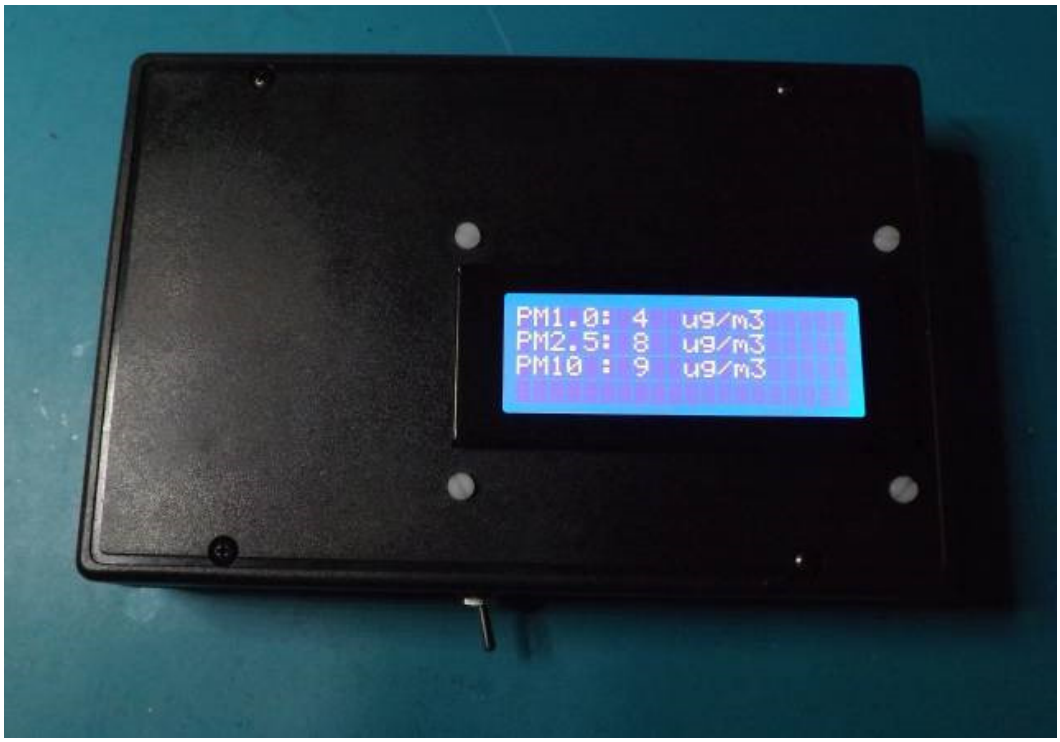

Abb. 3: Das Feinstaub-Messgerät in Betrieb, eingebaut im Gehäuse

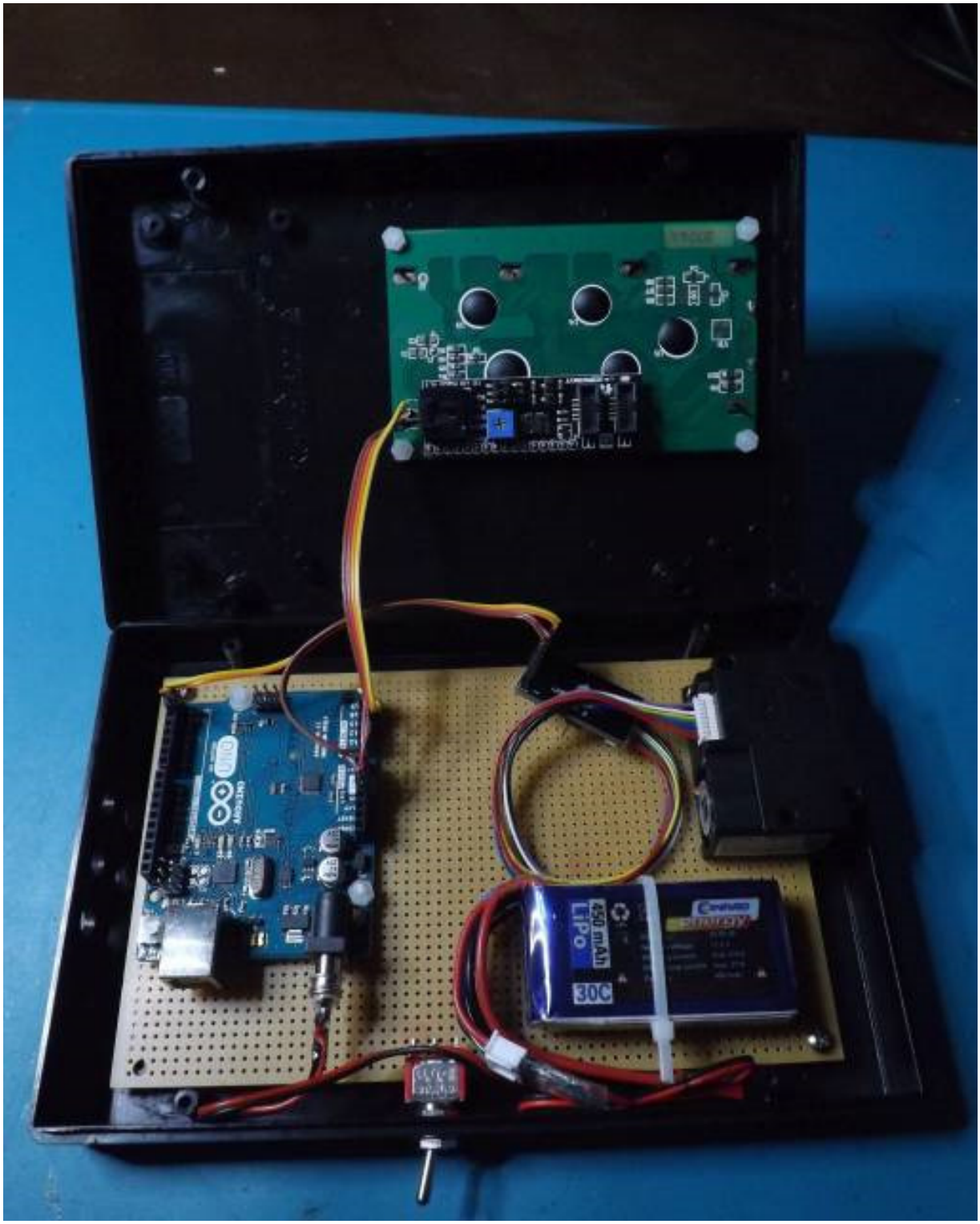

Abb. 4: Das Innere des Feinstaubmessgeräts

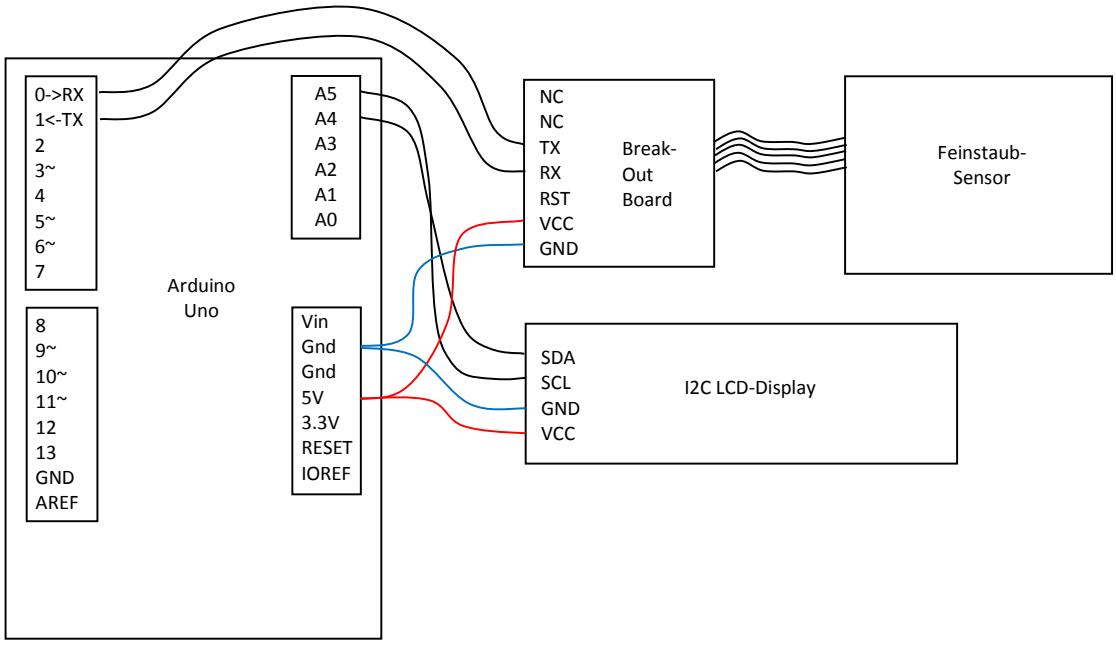

Abb. 5: Verkabelung zwischen Arduino, Break-Out Board und Display, das Kabel zwischen Break-Out Board und Sensor wird mitgeliefert.

Abb. 6a: Lufteinlass

Abb. 6b: Luftauslassöffnungen

Zum Testen bietet sich stark verdünnter Zigarettenrauch oder der Rauch eines Räucherstäbchens an, oder aber der Besuch einer entsprechenden Raucher-Kneipe. Auch in einem längeren Straßentunnel mit schlechter Lüftung werden recht hohe Werte erreicht. Werte, die nahe Null liegen, erreicht man dagegen in einem mit einer guten Klimaanlage ausgestattetes Bürogebäude. Die meisten guten Klimaanlagen haben nämlich Staubfilter, die den größten Teil des Feinstaubs abfangen. Oder aber man macht eine Wanderung auf einen Berggipfel eines Mittelgebirges oder gar auf die Zugspitze.

Zum Auslesen des Sensors kann man sich nun an das im Wiki der Produktseite von DfRobot halten. Das Beispielprogramm ist jedoch etwas überladen mit Funktionsaufrufen, was unnötig Speicherplatz verbraucht. Deswegen sei hier eine etwas einfachere Variante präsentiert.

Links:

DfRobot Webshop:

https://www.dfrobot.com

Ältere Arduino Entwicklungsumgebungen:

https://www.arduino.cc/en/Main/OldSoftwareReleases#previous

Umweltbundesamt:

http://www.umweltbundesamt.de/publikationen/feinstaubbelastung-in-deutschland

Anhang: Listing

```
//Anzeige auf 4-zeiligem LCD
//kompilieren z.B. mit Arduino_1_5_2 !!!
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

#define LEN 31

LiquidCrystal_I2C lcd(0x20,20,4);  // set the LCD address to 0x20 for a 20 chars and 4 line
display

unsigned char incomingByte = 0;   // for incoming serial data
unsigned char buf[LEN];
String str1, str2, str3;

void setup()
{
  Serial.begin(9600);   //use serial0
  Serial.setTimeout(1500);
  lcd.init();
  lcd.backlight();
  lcd.clear();
}
```

```
void loop()
{
      int i;
      int checksum, tcksum;
      int PM01Val, PM2_5Val, PM10Val;

      if (Serial.available() > 0) {
        incomingByte = Serial.read();
        if (incomingByte == 0x42) {
          Serial.readBytes((char *) buf, LEN); //(char *) is specific for old IDE < 1.6
          if ((buf[0] == 0x4d) && (buf[2] == 0x1C)) { //start character and frame length
            for (i=0; i<LEN-2; i++) {
              checksum = checksum + buf[i];
            }
            checksum = checksum + 0x42;
            tcksum = ((buf[LEN-2]<<8) + buf[LEN-1]);
            if (checksum == tcksum) { //valid data
              PM01Val=((buf[3]<<8) + buf[4]);
              PM2_5Val=((buf[5]<<8) + buf[6]);
              PM10Val=((buf[7]<<8) + buf[8]);

              lcd.clear();
              lcd.setCursor(0, 0);

              str1 = "PM1.0: ";
              str1 = String(str1 + PM01Val);
              str1 = String(str1 + "  ug/m3");

              str2 = "PM2.5: ";
              str2 = String(str2 + PM2_5Val);
              str2 = String(str2 + "  ug/m3");

              str3 = "PM10 : ";
              str3 = String(str3 + PM10Val);
              str3 = String(str3 + "  ug/m3");

              lcd.clear();
              lcd.setCursor(0, 0);
              Serial.print(str1);
              lcd.print(str1);
              lcd.setCursor(0, 1);
              Serial.print(str2);
              lcd.print(str2);
              lcd.setCursor(0, 2);
              Serial.print(str3);
              lcd.print(str3);



            }
            else {
              Serial.println("checksum Error");
              lcd.clear();
              lcd.print("checksum Error");
            }
          }
          else {
            Serial.println("frame error");
            lcd.clear();
            lcd.print("frame error");
          }
        }
      }
}
```