Weekly Reports

Luftqualität in Innenräumen - Gruppe 1

17. Januar 2022

Name	Matrikel Nr.	Arbeitsaufwand (h)
Friedrich Just	1326699	18,00
Stipe Knez	1269206	19,00
Lucas Merkert	1326709	20,00
Achim Glaesmann	1309221	17,50
Max-Rene Konieczka	1211092	18,00
Can Cihan Nazlier	1179244	14,00

Tabelle 1: Arbeitsaufwand dieser Woche

1 Überblick

1.1 Friedrich Just

Diese Woche wurden der Code für alle Sensoren in ein Programm zusammengeführt. Danach wurde dieser Mikrocontroller in unser Zigbee-Netzwerk integriert. Nun werden die aktuellen Sensordaten an den Koordinator gesendet wie in Abbildung 1 zu sehen ist. Eine Zeile steht für ein Paket mit Messungen aller 3 Sensoren. An erster Stelle steht die Nummer des Endgerätes. Es folgen der Sensor SHT21 mit den Messwerten der Temperatur und der relativen Luftfeuchtigkeit. Danach stehen die Werte des SCD41 mit der Temperatur, der relativen Luftfeuchtigkeit und dem CO2-Wert. Als Letztes kommt der CSS811 mit dem eCO2 und dem TVOC. Alle Temperatur- und Luftfeuchtigkeitswerte müssen durch 100 geteilt werden, damit der richtige Wert als Gleitkommazahl heraus kommt.

Bei dem CSS811 werden aus bisher noch nicht nachvollziehbaren Gründen falsche Werte an den Koordinator übertragen. Allerdings werden plausible Werte ausgegeben, wenn der Mikrocontroller direkt angeschlossen wird. (Siehe Abbildung 2) Außerdem ist bei der Berechnung des SHT21 etwas falsch, da dies zu hohe Werte sind. Der SCD41 gibt hingegen plausible Werte aus, denn ähnliche Werte habe ich mit einem separaten Sensor gemessen. Die Messdaten wurden erhoben, während Frischluft durch das geöffnete Fenster eindringen konnte.

```
1SHT;+3378;4056;SCD;+1957;4528;0934;CCS;0448;0007
1SHT;+3498;4084;SCD;+1913;4574;0880;CCS;6213;0885
1SHT;+3406;4052;SCD;+1893;4562;0852;CCS;5856;0831
1SHT;+3454;4024;SCD;+1880;4559;0826;CCS;6882;0987
1SHT;+3474;4016;SCD;+1881;4546;0806;CCS;7634;1102
1SHT;+3598;4020;SCD;+1885;4545;0786;CCS;7768;1122
1SHT;+3670;4052;SCD;+1893;4545;0775;CCS;7992;1156
1SHT;+3678;4072;SCD;+1897;4566;0768;CCS;7992;1156
1SHT;+3634;4012;SCD;+1904;4558;0763;CCS;7992;1156
1SHT;+3654;4120;SCD;+1911;4570;0760;CCS;7992;1156
1SHT;+3626;4168;SCD;+1923;4572;0756;CCS;7992;1156
1SHT;+3682;4168;SCD;+1932;4574;0755;CCS;7992;1156
1SHT;+3566;4116;SCD;+1943;4568;0752;CCS;7992;1156
1SHT;+3558;4184;SCD;+1947;4589;0751;CCS;7992;1156
1SHT;+3602;4252;SCD;+1955;4604;0755;CCS;4940;0691
1SHT;+3506;4208;SCD;+1967;4616;0790;CCS;3098;0411
```

Abbildung 1: Messdaten

```
-> CCS read data
->  00961 ppm CO2 CCS
->  00085 ppb TVOC CCS
```

Abbildung 2: Messdaten CCS811

1.2 Stipe Knez

Vergangene Woche wurde sich dazu entschieden, das Format, in dem die Daten der ZigBee Module an das Backend geschickt werden, zu überarbeiten. Anstatt die Daten in Form von JSON Strings an das Backend zu schicken, sollten diese nun in Form von regulären Strings, bei denen die Werte mit einem Semikolon getrennt werden, geschickt werden. Diese Funktionalität konnten wir im Verlauf der Woche mit Hilfe eines entsprechenden Parsers erfolgreich im Backend implementieren. Als nächstes wurde begonnen sich dem Übertragen der Daten vom Back- in das Frontend und der Erstellung eines Dashboards mit Hilfe der D3.js Library zuzuwenden.

1.3 Lucas Merkert

Diese Woche wurde der CCS811 Sensor[2] zum laufen gebracht. Hierbei haben wir uns mithilfe der Informationen von Dienstag einen geeigneten Zustandsautomaten wie in Abbildung 3 überlegt und diesen dann umgesetzt. Hierbei wurde der Sensor über die Adresse 0x5A angesprochen angesprochen. Im ersten Zustand wechselt der Sensor in den BOOT Modus damit ein Software-Reset durchgeführt werden kann und die Hardware ID überprüft werden kann. Danach wechselt der Sensor wieder in den APP Modus. In das MEAS_MODE Register wird dann der Befehl geschrieben, dass der Sensor jede Sekunde misst. Zuletzt wird noch im Status Register überprüft ob Daten bereit liegen und diese werden dann ausgelesen. Dies geschieht in einem Zyklus von einer Sekunde.

1.4 Achim Glaesmann

Diese Woche wurde dazu verwendet die von den Verschiedenen Teammitgliedern übernommenen Aufgaben bezüglich der Auslesung des Sensors und der Übertragung der Daten über das Zigbee Protokoll zusammenzuführen. Zunächst wurde sich persönlich getroffen, um in der Gruppe Probleme beim Auslesen des CCS811 und dem Aufbau des Funknetzes zu lösen. Nach testen des Sensors am Arduino und dem Übertragen der Programmlogik der anderen Sensoren auf den AVR-Chip, war es trotz mehrfachen Prüfen des Codes nicht möglich eine I2C-Kommunikation zum CCS aufzubauen. Selbst nach ausführlichem Testen und mehrmaligen umschreiben einzelner Softwarekomponenten, konnte das Problem nicht behoben werden. Es stellte sich heraus, das es kein softwareseitiges Problem war. Nach austauschen des Funkmoduls war es möglich eine Kommunikation zwischen Sensor und Mikrokontroller zu etablieren. Der Mikrokontroller wurde markiert und wird nochmal überprüft. Nachdem sowohl das Auslesen des CSS811, als auch Aufbau des Funknetzwerkes realisiert wurden, war es nötig die einzelnen Softwarelösungen für die Sensoren und das Netzwerk zu kombinieren. Dies wurde erfolgreich erledigt. Als nächste Aufgabe ist es nun nötig, die Übermittelten Daten an das Backend der Desktop-Applikation zu übermitteln.

1.5 Max-Rene Konieczka

Im Verlauf der letzten Woche wurde daran gearbeitet, Strings über Atmel Studio zu versenden und diese anschließend in unserer Datenbank anzeigen zu lassen. Nachdem dies gelungen ist, wurde angefangen sich mit dem Dashboard auseinanderzusetzen. Es wurde überlegt, die D3.js Library von Javascript [1] zu verwenden. Mit D3.js ist es möglich einfache Formen bis hin zu detaillierten Diagrammen zu erstellen. Die Daten der Sensoren sollen nun mithilfe der Axis API von D3 in einem Koordinatensystem dargestellt werden.

1.6 Can Cihan Nazlier

Durchgehen der Dokumentation von d3js. Websockets vorbereiten, sodass server side events gesendet werden können. Warum Websockets? Antwort: das frontend und das backend laufen als zwei seperate anwendungen und müssen sich untereinander austauschen. Das geschieht eigentlich über http. Das Problem jedoch ist, für http muss der Client dem Server request schicken um Daten zu empfangen. Da die Sensordaten im Backend ankommen und wenn die Kommunikation nur mittels http geschieht, so müsste der client alle 10 minuten eine Anfrage an den Server senden und es könnte durchaus dazu kommen , dass wir einzelne Messungen verpassen. Deshalb ist es wichtig, dass der server dem Client Daten sendet sobald welche da sind. Dazu wurde nun die Grundlage gelegt.

Literatur

[1] *D3 Data-Driven Documents*. URL: `https://d3js.org/`.

[2] *Datasheet CCS811*. URL: https://learn.adafruit.com/adafruit-ccs811-air-quality-sensor?view=all#documents.

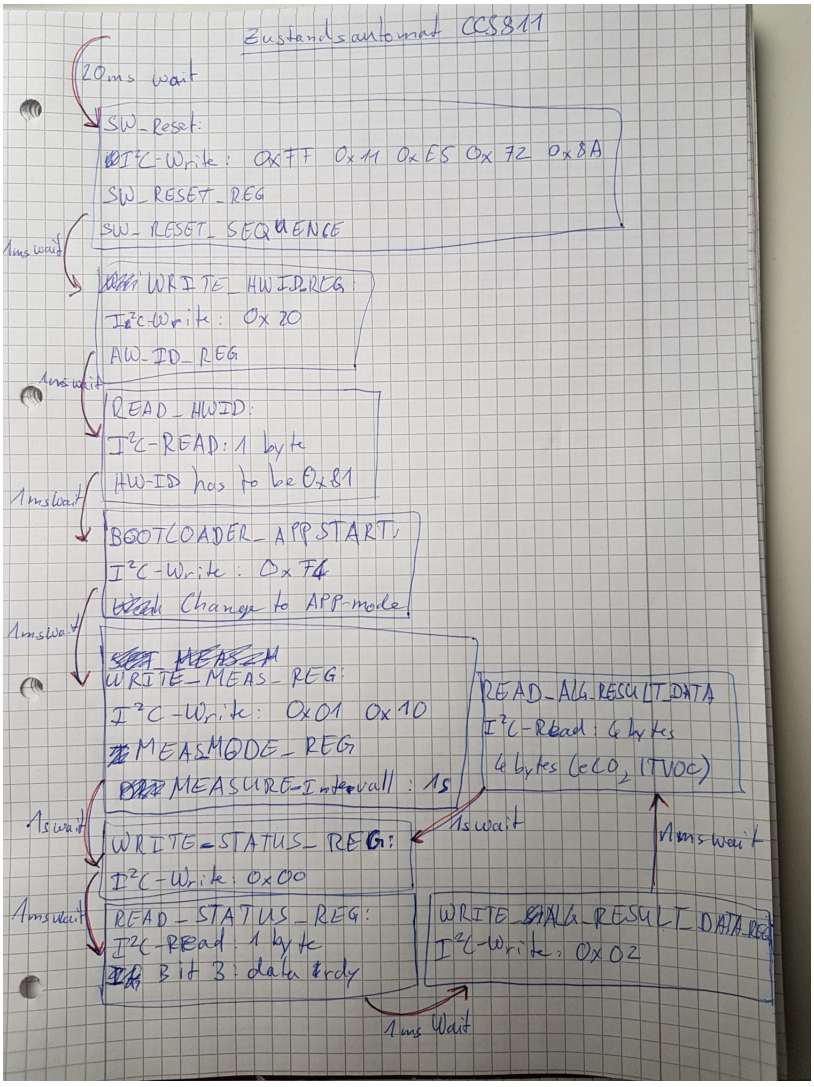

Abbildung 3: Zustandsautomat CCS811