

Applied IoT Project

Luchtkwaliteitsmeting Hobokense Polder

Opleiding: 3ITIOT - 2VTITIOT

Academiejaar: 2023 - 2024

Gil Struyf, Joos Van Esbroeck, Jordy Van Mol, Jurgen Vanpeteghem, Niels Aarts

Dirk Van Merode, Maarten Luyts

Hardware | sensoren & PCB

Sensoren | SEN55 & SPG 41

SEN55

De SEN55 is een eenvoudig, alles-in-een platform voor sensoroplossingen voor het nauwkeurig meten van verschillende omgevingsparameters, zoals fijnstof, vluchtige organische stoffen (VOC's), oxiderende gassen, zoals stikstofoxiden (NOx), en als vochtigheid en temperatuur.

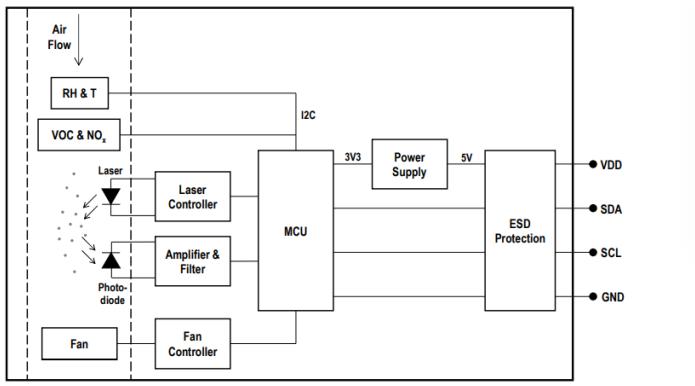
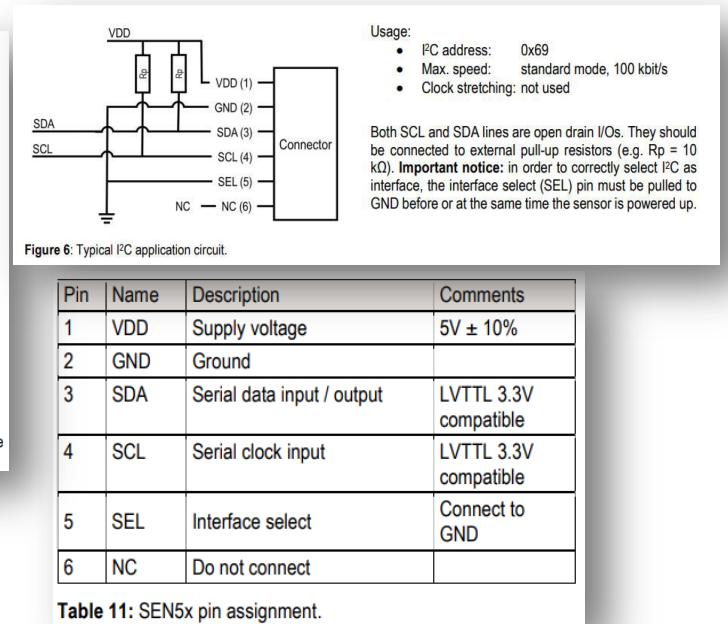


Figure 1 Functional block diagram of the SEN5x. The RH & T and VOC sensors are only contained in the SEN54 and SEN55, the NOx signal is only contained in the SEN55.



Over het algemeen kunnen de meeteenheden als volgt zijn:

- Fijnstof: Gemeten in microgram per kubieke meter ($\mu\text{g}/\text{m}^3$) voor de concentratie van fijnstofdeeltjes in de lucht.
- VOC's: Gemeten in ppb (parts per billion) of ppm (parts per million) voor de concentratie van vluchtige organische stoffen in de lucht.
- NOx: Gemeten in ppb of ppm voor de concentratie van stikstofoxiden (NOx) in de lucht.
- Vochtigheid: Gemeten in procent (%), wat de relatieve luchtvuchtigheid weergeeft.
- Temperatuur: Gemeten in graden Celsius ($^\circ\text{C}$) voor de temperatuur in de omgeving.

De exacte eenheden en meetbereik kunnen variëren afhankelijk van de specifieke sensoren en configuraties die worden gebruikt binnen het SEN55-platform van Sensirion.

SGP41

De SGP41 is een VOC- en NOx-sensor van Sensirion. Deze sensor is zeer geschikt voor het voortdurend monitoren van de VOC- en NOx-situatie, inclusief potentieel schadelijke gebeurtenissen die niet door mensen worden waargenomen.

De meeteenheid voor VOC kan in het algemeen worden uitgedrukt in parts per billion (ppb) of parts per million (ppm), afhankelijk van de concentratie van vluchtige organische stoffen. NOx-concentraties worden meestal uitgedrukt in ppb of ppm voor gassen zoals stikstofdioxide (NO_2) en stikstofmonoxide (NO). De precieze meeteenheden kunnen variëren afhankelijk van de specifieke configuratie en toepassing van de sensor.

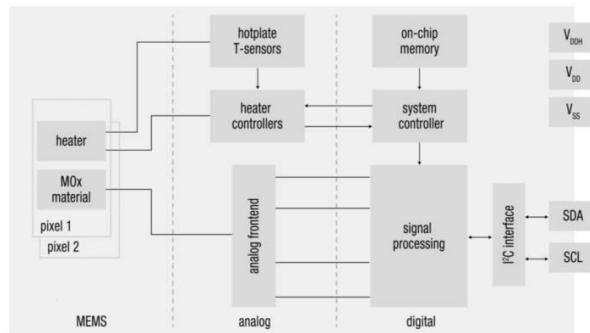


Figure 1 Functional block diagram of the SGP41.

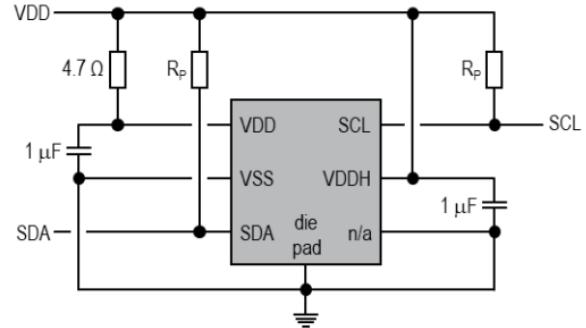
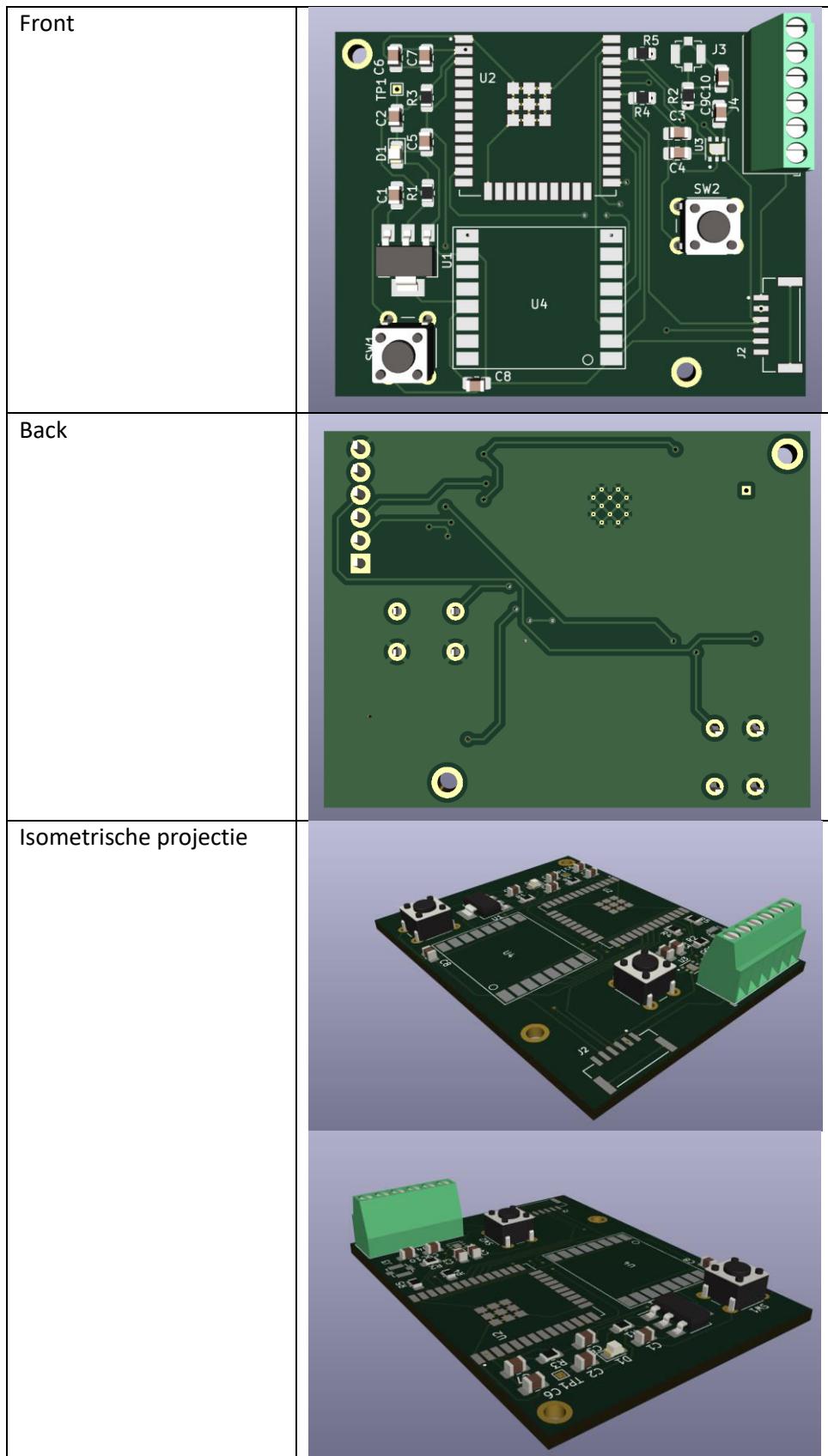
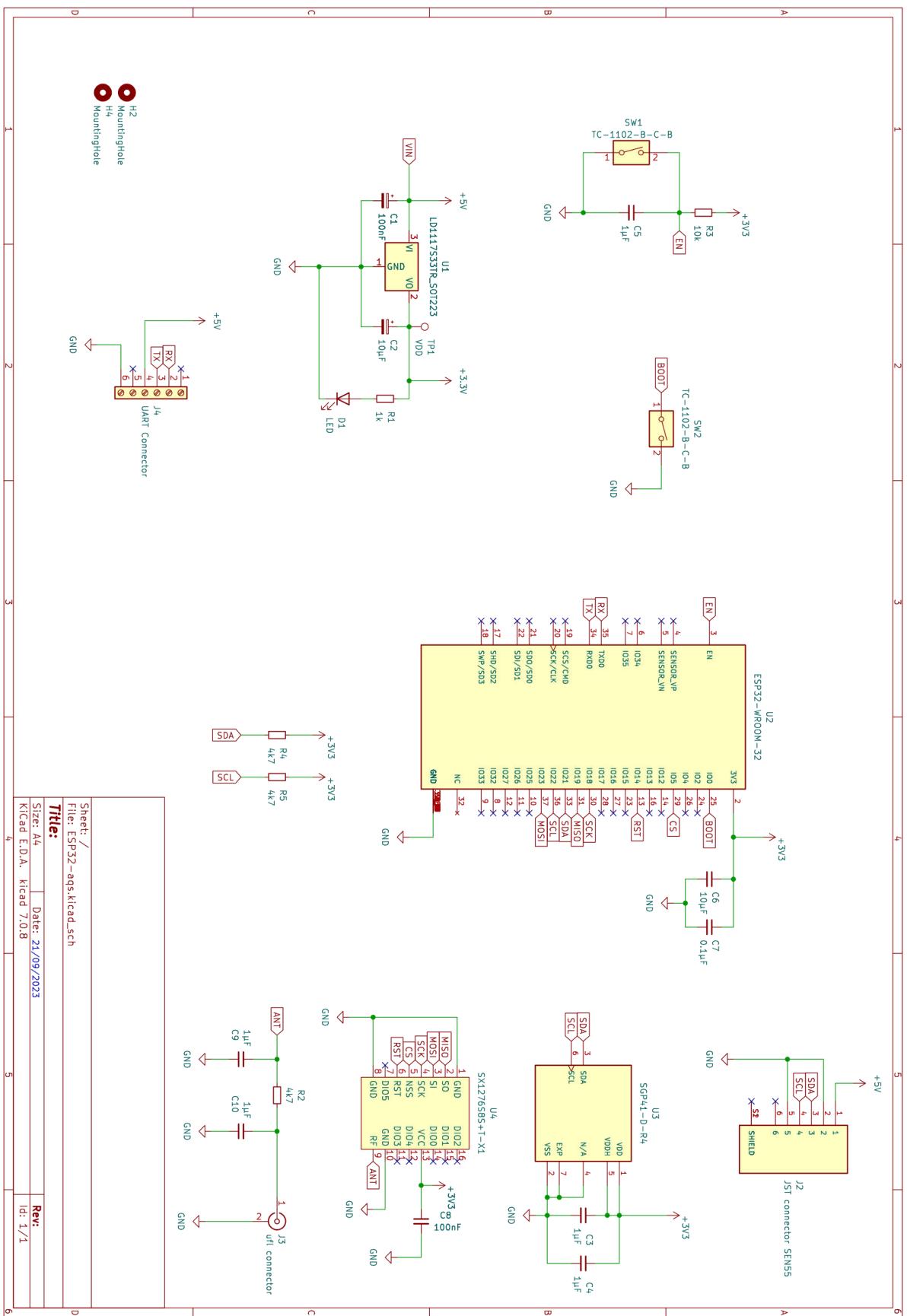


Figure 6 Typical application circuit.

PCB

1^{ste} versie





Uitleg PCB:

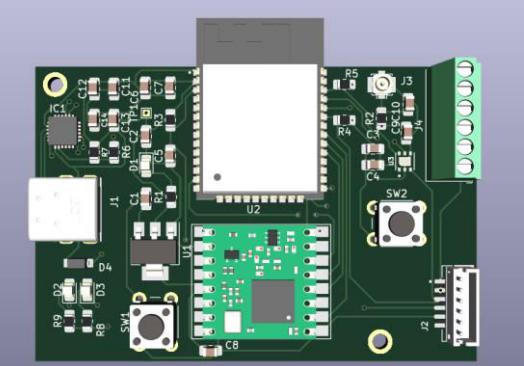
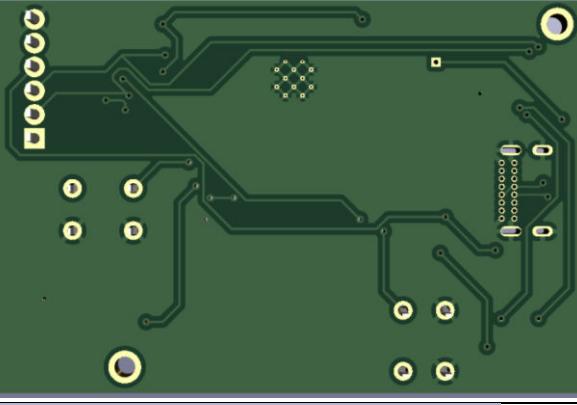
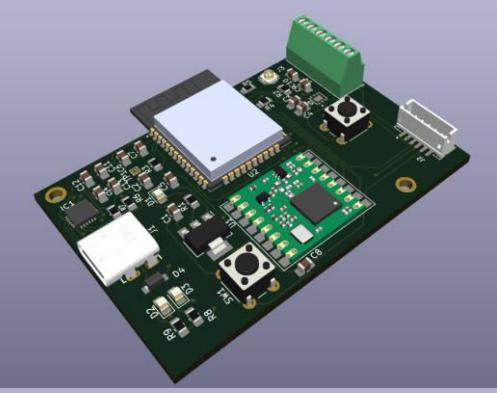
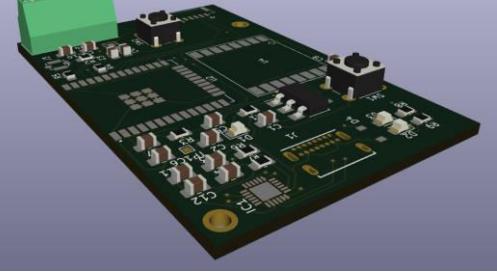
We dienden rekening te houden met verschillende aspecten, waaronder de dikte van onze PCB-lijnen in relatie tot het stroomverbruik. We hebben eerst het maximale stroomverbruik berekend en deze verdubbeld. Met behulp van een PCB Trace Width Calculator hebben we parameters zoals stroomverbruik, normale temperatuur, stijgende temperatuur en dikte ingevoerd. Uit de berekeningen bleek dat de lijndikte minimaal 0.03 mm moet zijn. De standaarddikte die we in KiCad gebruiken is 0.25 mm, wat ruim voldoende is.

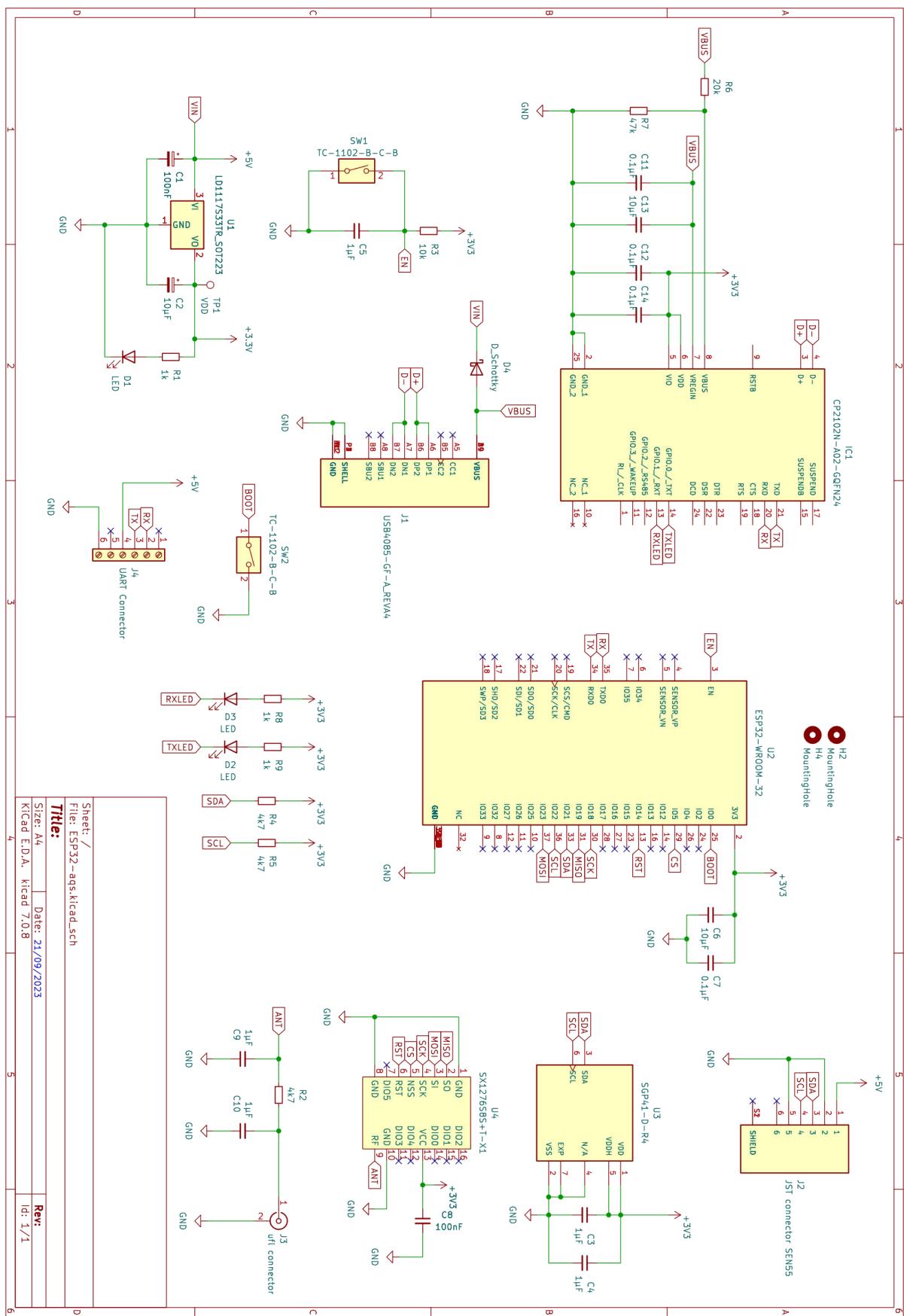
Bij de plaatsing van de ESP32-chip moesten we ervoor zorgen dat de antenne niet tussen andere componenten lag. Dit hebben we opgelost door de antenne iets buiten het bord te laten steken. Daarnaast was het belangrijk om ervoor te zorgen dat de connectoren aan de zijkant waren geplaatst om een optimale verbinding te garanderen.

BoM V1

Id	Designator	Footprint	Quantity	Designation	Supplier	Aanwezig
1	H2,H4	Mounting	2	MountingHole	x	
2	R5,R2,R4	R_0805_20	3	4k7		J
3	U3	SGP41	1	SGP41-D-R4		
4	C8,C1	C_0805_20	2	100nF		J
5	D1	LED_0805	1	LED		J
6	J3	U.FL_Molex	1	ufl connector		
7	C5,C4,C9,C	C_0805_20	5	1 μ F		J
8	R1	R_0805_20	1	1k Ω		J
9	SW1,SW2	SW_PUSH	2	TC-1102-B-C-B		
10	U2	MODULE	1	ESP32-WROOM-32	J	
11	C7	C_0805_20	1	0.1 μ F		
12	U4	SX1276S8S	1	SX1276S8S+T-X1	J	
13	TP1	TestPoint	1	VDD		J
14	J2	JST_BM06	1	JST connector SEN55		
15	J4	TerminalB	1	UART Connector	J	
16	C6,C2	C_0805_20	2	10 μ F		J
17	R3	R_0805_20	1	10k		
18	U1	SOT-223-3	1	LD1117S33TR_SOT22J		

2^{de} versie

Front	
Back	
Isometrische projectie	 



Uitleg veranderingen:

- UART-chip toegevoegd
- USB-C toegevoegd
- Schottky diode toegevoegd voor bescherming van de USB-C
- Verandering plaats JST-connector
- UFL-connector naar beneden geplaatst

De uitbreiding met UART-chip en USB-C connector werd toegevoegd als vervanging van de Adafruit CP2102N USB to Serial Converter die we voor het prototype gebruikten om de ESP32 te programmeren. Deze uitbreiding is dus niet noodzakelijk maar gewoon handig zodat we de PCB rechtstreeks via USB-C kunnen aansluiten en programmeren.

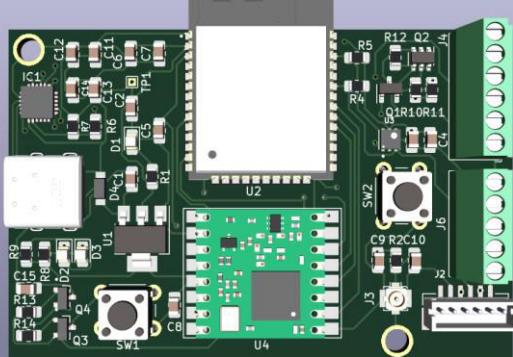
We gebruikten hiervoor het voorbeeldschema van deze converter om dit gelijkaardig te integreren op onze eigen PCB.

BoM V2

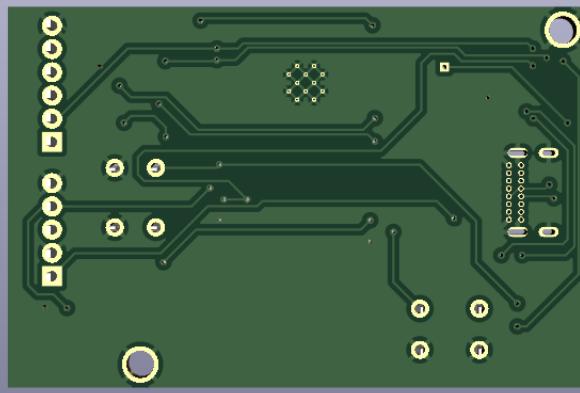
Id	Designator	Footprint	Quantity	Supplier and reference		Aanwezig in labo
				Designation	Supplier	
1	H2,H4	MountingHole_2.2mm_M2_ISO7380_Pad	2	MountingHole		
2	R5,R2,R4	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder	3	4k7	J	
3	IC1	QFN50P400X400X80-25N	1	CP2102N-A02-GQFN24	X	
4	U3	XDCR_SGP41-D-R4	1	SGP41-D-R4	J	
5	C8,C1	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder	2	100nF	J	
6	D1,D2,D3	LED_0805_2012Metric_Pad1.15x1.40mm_HandSolder	3	LED	J	
7	J3	UFL_Molex_MCRF_73412-0110_Vertical	1	ufl connector	J	
8	C5,C4,C9,C3,C10	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder	5	1ÂµF	J	
9	R1	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder	1	1kÎ©	J	
10	SW1,SW2	SW_PUSH_6mm	2	TC-1102-B-C-B	X	
11	R8,R9	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder	2	1k	J	
12	U2	MODULE_ESP32-WROOM-32	1	ESP32-WROOM-32	X	
13	C7,C14,C12,C11	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder	4	0.1ÂµF	J	
14	U4	SX1276S8S+T-X1	1	SX1276S8S+T-X1	X	
15	TP1	TestPoint_THTPad_1.0x1.0mm_Drill0.5mm	1	VDD		
16	J1	GCT_USB4085-GF-A_REV4	1	USB4085-GF-A_REV4	X	
17	J2	JST_BMO6B-GHS-TBT_LF_SN_N_	1	JST connector SEN55	X	
18	D4	SOD-123FL	1	D_Schottky	X	
19	J4	TerminalBlock_Phoenix_MPT-0.5-6-2.54_1x06_P2.54mm_Horizontal	1	UART Connector	X	
20	R7	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder	1	47k	J	
21	C6,C2,C13	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder	3	10ÂµF	J	
22	R3	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder	1	10k	J	
23	R6	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder	1	20k	J	
24	U1	SOT-223-3_TabPin2	1	LD1117S33TR_SOT223	X	

3de versie

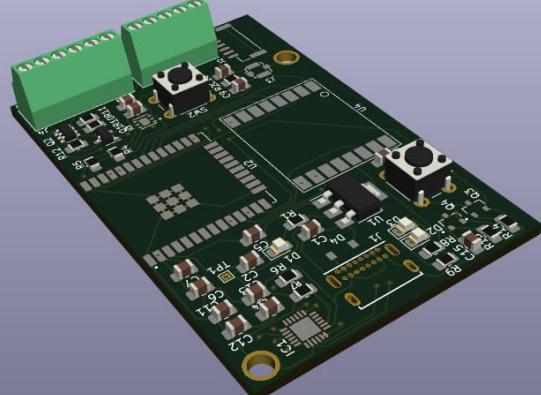
Front

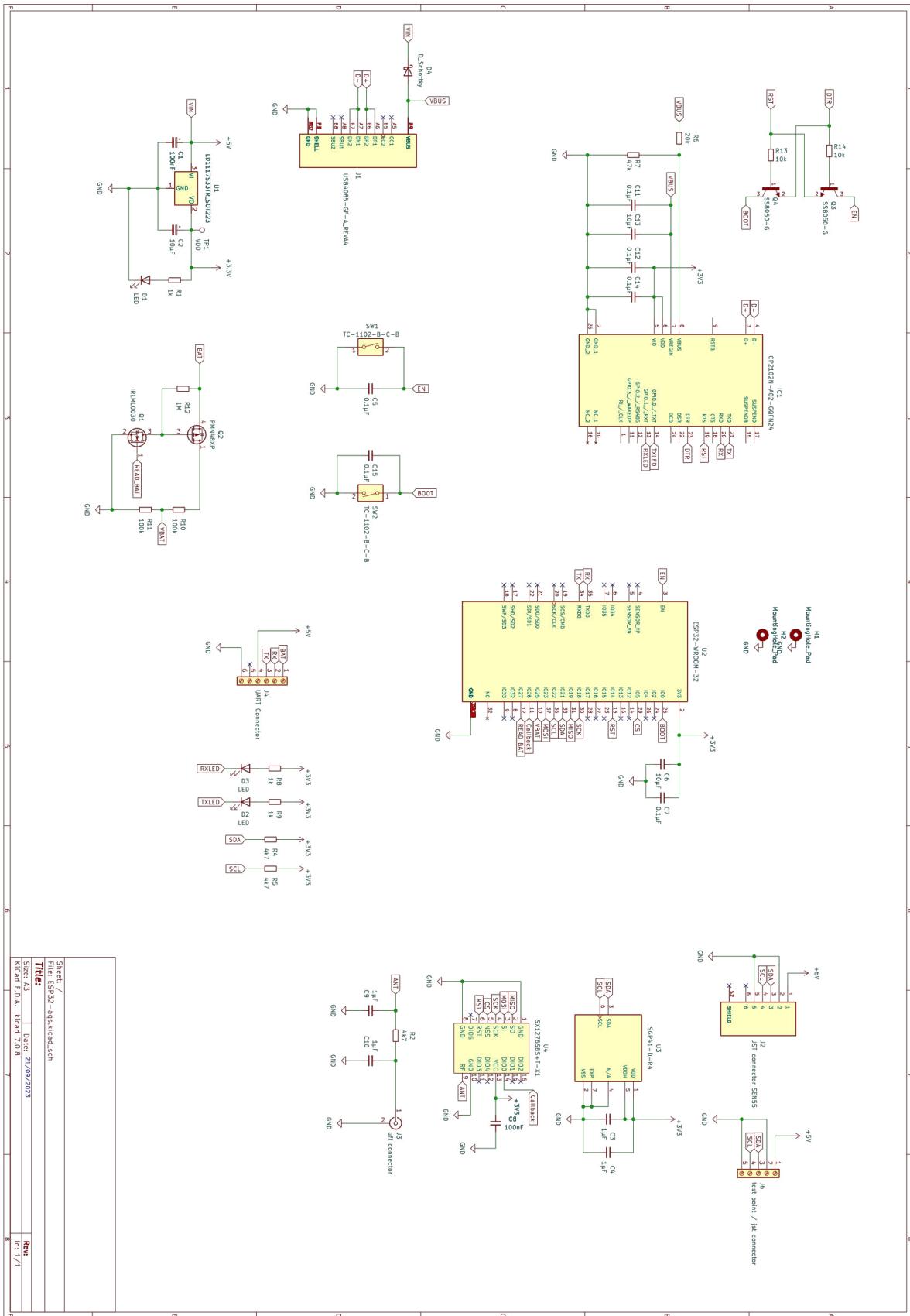


Back



Isometrische projectie





Uitleg veranderingen:

In de derde versie werden nog enkele revisies gemaakt ten opzichte van voorgaande versie.

- DIO0 van de Lora module werd verbonden met GPIO33 van de ESP32. Deze was niet verbonden in de vorige versie en is wel nodig om de Lora communicatie succesvol te laten verlopen.
- Er werd een circuit toegevoegd om de batterijspanning van de batterij aan de DFRobot te kunnen uitlezen. Dit circuit bestaat uit een spanningsdeler en een mosfet schakeling om de uitlezing te kunnen aan/uit schakelen om stroom te besparen.
- De mounting holes werden vergroot naar 3mm. Deze maat is gebruikelijker en handiger om de PCB vast te schroeven op de backplate in de behuizing.
- Er werd ook een klein circuit toegevoegd om de ESP32 automatisch te kunnen programmeren zonder de BOOT knop te moeten indrukken tijdens het uploaden van de code.
- Ten slotte werden er nog kleine aanpassingen gemaakt aan de layout van het PCB-design en er werd een extra screw terminal toegevoegd om de SEN55 aan te sluiten. Op die manier kunnen we kiezen tussen de JST-connector of de screw terminal.

BoM V3

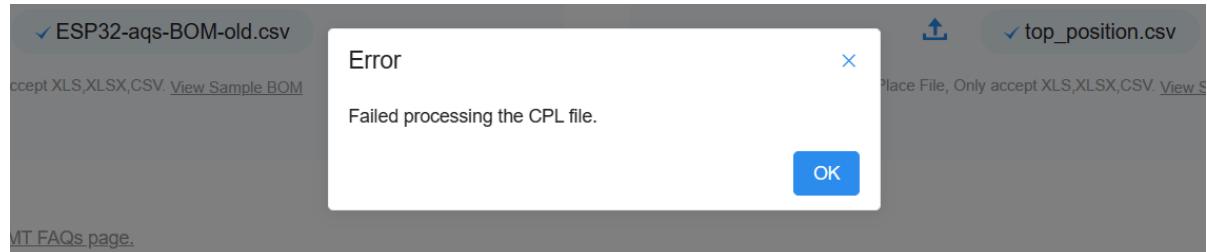
1	Comment	Designator	Footprint	LCSC Part Number			
2	100k	R10	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder				
3	100k	R11	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder				
4	4k7	R5	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder				
5	4k7	R2	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder				
6	4k7	R4	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder				
7	1k	R1	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder				
8	1k	R8	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder				
9	1k	R9	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder				
10	PMN48XP	Q2	TSOP-6_1_C552986				
11	CP2102N-	IC1	QFN50P40 C969913				
12	SGP41-D-F	U3	XDCR_SGP C3659325				
13	100nF	C1	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder				
14	100nF	C8	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder				
15	LED	D1	LED_0805_2012Metric_Pad1.15x1.40mm_HandSolder				
16	LED	D2	LED_0805_2012Metric_Pad1.15x1.40mm_HandSolder				
17	LED	D3	LED_0805_2012Metric_Pad1.15x1.40mm_HandSolder				
18	ufl connec	J3	U.FL_Molex_MCRF_73412-0110_Vertical				
19	0.1uF	C5	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder				
20	0.1uF	C7	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder				
21	0.1uF	C15	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder				
22	0.1uF	C14	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder				
23	0.1uF	C12	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder				
24	0.1uF	C11	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder				
25	1uF	C4	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder				
26	1uF	C9	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder				
27	1uF	C3	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder				
28	1uF	C10	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder				
29	TC-1102-B	SW1	SW_PUSH_6mm				
30	TC-1102-B	SW2	SW_PUSH_6mm				
31	ESP32-WR	U2	MODULE_C529582				
32	SX1276S8S	U4	SX1276S8S C2844472				
33	VDD	TP1	TestPoint_THTPad_1.0x1.0mm_Drill0.5mm				
34	~	J5	USB-TYPE-C2927038				
35	Mounting	H1	MountingHole_2.2mm_M2_ISO7380_Pad				
36	Mounting	H2	MountingHole_2.2mm_M2_ISO7380_Pad				
37	JST connec	J2	JST_BM06 C189892				
38	test point	J6	TerminalB 5-5-2.54_1C474923				
39	D_Schottk	D4	SOD-123F C489144				
40	UART Coni	J4	TerminalB 5-6-2.54_1C474924				
41	47k	R7	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder				
42	10uF	C2	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder				
43	10uF	C6	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder				
44	10uF	C13	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder				
45	10k	R13	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder				
46	10k	R14	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder				
47	SS8050-G	Q3	TRANS_SS C164885				
48	SS8050-G	Q4	TRANS_SS C164885				
49	IRLML003I	Q1	SOT-23				
50	1M	R12	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder				
51	20k	R6	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder				
52	LD1117S3	U1	SOT-223-3 C86781				

PCB laten bestukken via JLCPCB

Voor het genereren van de BoM en CPL-file hebben we volgende instructies van JLCPCB gevolgd:

<https://jlpcb.com/help/article/81-How-to-generate-the-BOM-and-Centroid-file-from-KiCAD>

Dit alleen is niet genoeg aangezien je dan volgende fout krijgt in JLCPCB:



Om dit op te lossen open je de BoM.csv met Excel en doet volgende aanpassingen:

Voor	1	Id	Designator	Footprint	Quantity	Designatio	Supplier and ref
	2		1 R11,R10	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder	2	100k	
	3		2 R5,R2,R4	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder	3	4k7	
	4		3 Q2	TSOP-6_1.65x3.05mm_P0.95mm	1	PMN48XP	
	5		4 IC1	QFN50P400X400X80-25N	1	CP2102N-A02-GQFN24	
	6		5 U3	XDCR_SGP41-D-R4	1	SGP41-D-R4	
	7		6 C8,C1	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder	2	100nF	
	8		7 D1,D2,D3	LED_0805_2012Metric_Pad1.15x1.40mm_HandSolder	3	LED	
	9		8 J3	U.FL_Molex_MCRF_73412-0110_Vertical	1	ufl connector	
Na	1	Comment	Designator	Footprint	LCSC Part Number		
	100k	R11,R10	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder				
	4k7	R5,R2,R4	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder				
	PMN48XP	Q2	TSOP-6_1.65x3.05mm_P0.95mm				
	CP2102N-A02	IC1	QFN50P400X400X80-25N				
	SGP41-D-R4	U3	XDCR_SGP41-D-R4				
	100nF	C8,C1	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder				
	LED	D1,D2,D3	LED_0805_2012Metric_Pad1.15x1.40mm_HandSolder				
	ufl connector	J3	U.FL_Molex_MCRF_73412-0110_Vertical				

Aanpassingen:

- Aanpassing headers
- Waardes veranderen van plek
- Verwijdering van kolom quantity en supplier and ref
- Toevoeging van LCSC Part Number (optioneel)

Hieraan open je het bestand in notepad en doe je volgende aanpassingen:

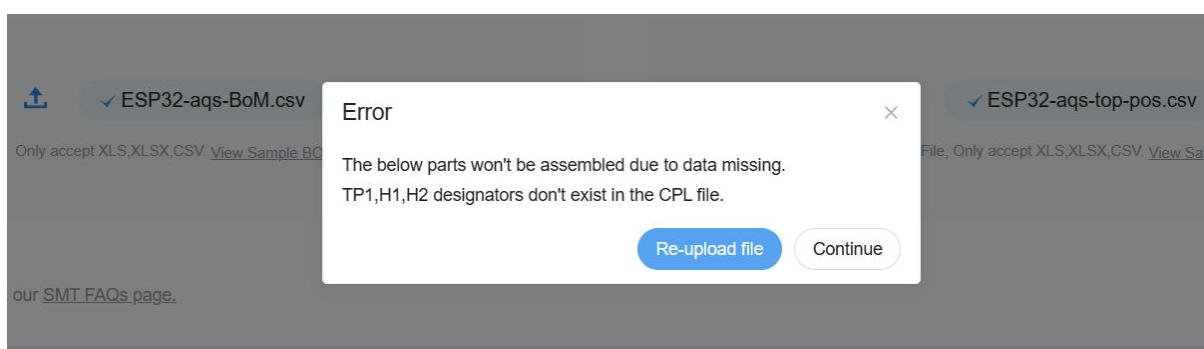
Voor	Comment;Designator;Footprint;LCSC Part Number 100k;R11,R10;R_0805_2012Metric_Pad1.20x1.40mm_HandSolder; 4k7;R5,R2,R4;R_0805_2012Metric_Pad1.20x1.40mm_HandSolder; PMN48XP;Q2;TSOP-6_1.65x3.05mm_P0.95mm; CP2102N-A02-GQFN24;IC1;QFN50P400X400X80-25N; SGP41-D-R4;U3;XDCR_SGP41-D-R4; 100nF;C8,C1;C_0805_2012Metric_Pad1.18x1.45mm_HandSolder; LED;D1,D2,D3;LED_0805_2012Metric_Pad1.15x1.40mm_HandSolder; ufl connector;J3;U.FL_Molex_MCRF_73412-0110_Vertical;
Na	Comment,Designator,Footprint,LCSC Part Number 100k,R10,R_0805_2012Metric_Pad1.20x1.40mm_HandSolder 100k,R11,R_0805_2012Metric_Pad1.20x1.40mm_HandSolder 4k7,R2,R_0805_2012Metric_Pad1.20x1.40mm_HandSolder 4k7,R4,R_0805_2012Metric_Pad1.20x1.40mm_HandSolder 4k7,R5,R_0805_2012Metric_Pad1.20x1.40mm_HandSolder PMN48XP,Q2,TSOP-6_1.65x3.05mm_P0.95mm CP2102N-A02-GQFN24,IC1,QFN50P400X400X80-25N SGP41-D-R4,U3,XDCR_SGP41-D-R4 100nF,C1,C_0805_2012Metric_Pad1.18x1.45mm_HandSolder 100nF,C8,C_0805_2012Metric_Pad1.18x1.45mm_HandSolder LED,D1,LED_0805_2012Metric_Pad1.15x1.40mm_HandSolder LED,D2,LED_0805_2012Metric_Pad1.15x1.40mm_HandSolder LED,D3,LED_0805_2012Metric_Pad1.15x1.40mm_HandSolder ufl connector,J3,U.FL_Molex_MCRF_73412-0110_Vertical

Aanpassingen:

- ; veranderd naar ,
- Alle componenten met dezelfde waardes opgesplitst

Dit zijn alle aanpassingen die nodig zijn voor de BoM file. De enige aanpassingen die je moet maken in de PCL-file zijn de aanpassingen die je al hebt gemaakt bij het volgen van de instructies van JLCPCB.

Als je dit hebt aangepast, ga je op volgende error botsen:



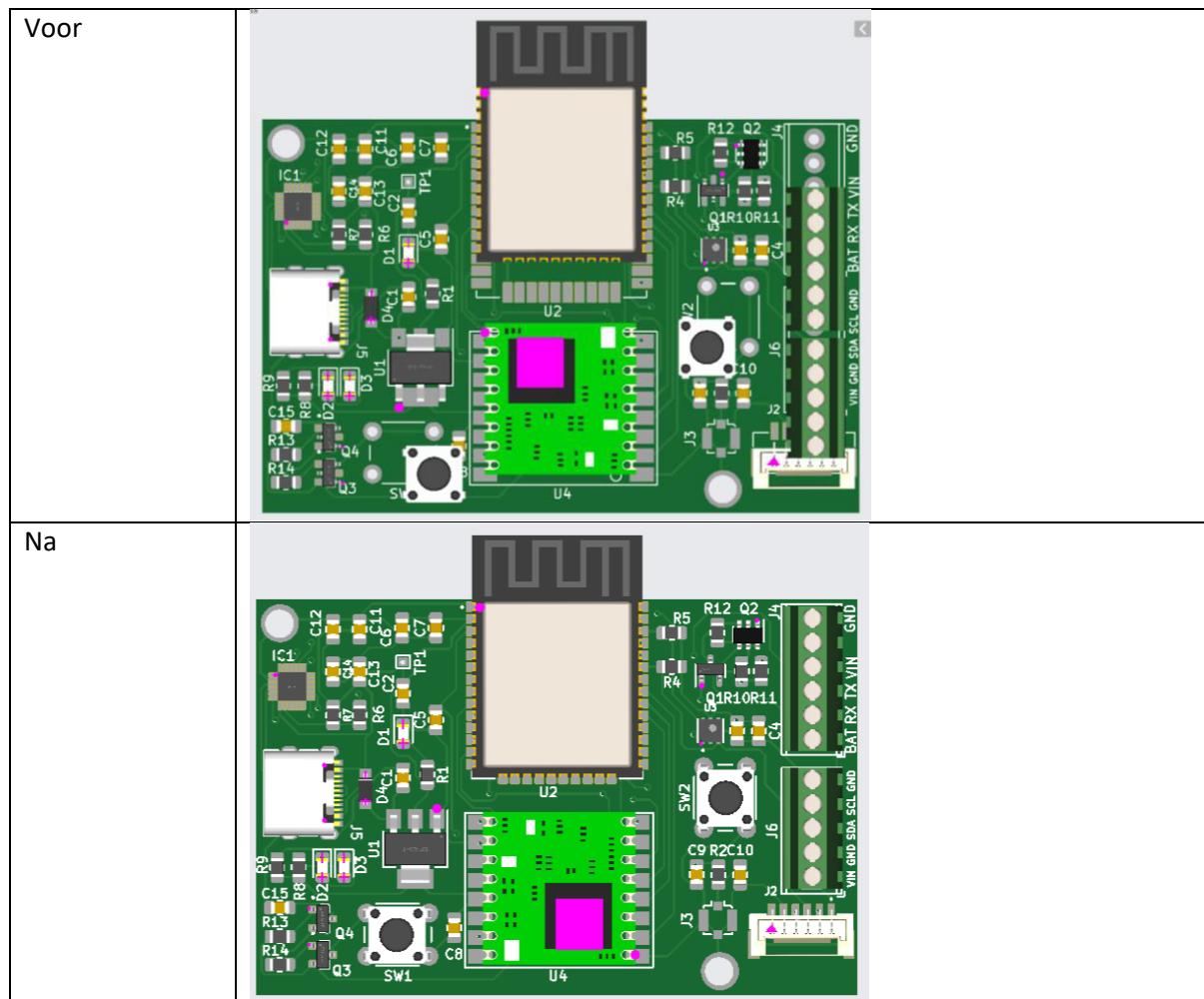
Deze error zegt dat er geen data is voor volgende designators, maar in dit geval zijn deze mounting holes of testing holes in de PCB en moet hier natuurlijk niets bestukt worden. Hier druk je dus nu op continue.

Nu zie je het volgende:

Top Designator	Comment	Footprint	Matched Part Detail	Qty	Source	Lib Type	Total Cost	Select
R11	100k	R_0805_2012...	RS-05K1003FT C118847 125mW ±100ppm/°C ±1% 100kΩ 0805 Chip Re...	20	JLCPCB	Extended	\$0.0340 ⓘ	<input checked="" type="checkbox"/>
R10	100k	R_0805_2012...	RS-05K1003FT C118847 125mW ±100ppm/°C ±1% 100kΩ 0805 Chip Re...	20	JLCPCB	Extended	\$0.0340 ⓘ	<input checked="" type="checkbox"/>
R2	4k7	R_0805_2012...	0805W8J0472T5E C26022 125mW Thick Film Resistors ±100ppm/°C ±5% ...	20	JLCPCB	Extended	\$0.0260 ⓘ	<input checked="" type="checkbox"/>
R5	4k7	R_0805_2012...	0805W8J0472T5E C26022 125mW Thick Film Resistors ±100ppm/°C ±5% ...	20	JLCPCB	Extended	\$0.0260 ⓘ	<input checked="" type="checkbox"/>
R4	4k7	R_0805_2012...	0805W8J0472T5E C26022 125mW Thick Film Resistors ±100ppm/°C ±5% ...	20	JLCPCB	Extended	\$0.0260 ⓘ	<input checked="" type="checkbox"/>
U3	SGP41-D-R4	XDCR_SGP41-...	SGP41-D-R4 C3659325 Detailed description is being updated	5	JLCPCB	Extended	\$34.5675 ⓘ	<input type="checkbox"/> Standard Only
C8	100nF	C_0805_2012...	CC0805KRX7R9BB104 C49678 50V 100nF X7R ±10% 0805 Multilayer Ceramic...	10	JLCPCB	Basic	\$0.0430 ⓘ	<input checked="" type="checkbox"/>
C1	100nF	C_0805_2012...	CC0805KRX7R9BB104 C49678 50V 100nF X7R ±10% 0805 Multilayer Ceramic...	10	JLCPCB	Basic	\$0.0430 ⓘ	<input checked="" type="checkbox"/>
D1	LED	LED_0805_201...	NCD0805R1 C84256 Colorless transparency 67mcd~195mcd -30°C~...	15	JLCPCB	Basic	\$0.1605 ⓘ	<input checked="" type="checkbox"/>
D2	LED	LED_0805_201...	NCD0805R1 C84256 Colorless transparency 67mcd~195mcd -30°C~...	15	JLCPCB	Basic	\$0.1605 ⓘ	<input checked="" type="checkbox"/>
D3	LED	LED_0805_201...	NCD0805R1 C84256 Colorless transparency 67mcd~195mcd -30°C~...	15	JLCPCB	Basic	\$0.1605 ⓘ	<input checked="" type="checkbox"/>
IC1	CP2102N-A0...	QFN50P400X4...	CP2102N-A02-GQFN24 C1550551 USB-to-UART UART QFN-24-EP(4x4) USB ICs ...	5		Extended	\$0.00 ⓘ	<input type="checkbox"/> 5 shortfall
J3	uflconnector	U.FL_Molex_M...	No part selected		Search	Pre-order		
Q2	PMN48XP	TSOP-6_1.65x...	No part selected		Search	Pre-order		

Bij de designators waarbij nog geen component aangewezen is kan je simpelweg op het vergrootglas klikken en de component zelf zoeken.

Als laatste stap krijg je je PCB te zien. Bij volgend voorbeeld zie je dat er een aantal zaken niet juist staan. Deze componenten moet je zelf nog goed zetten door oftewel de waardes aan te passen in de PCL-file oftewel in JLCPCB zelf:

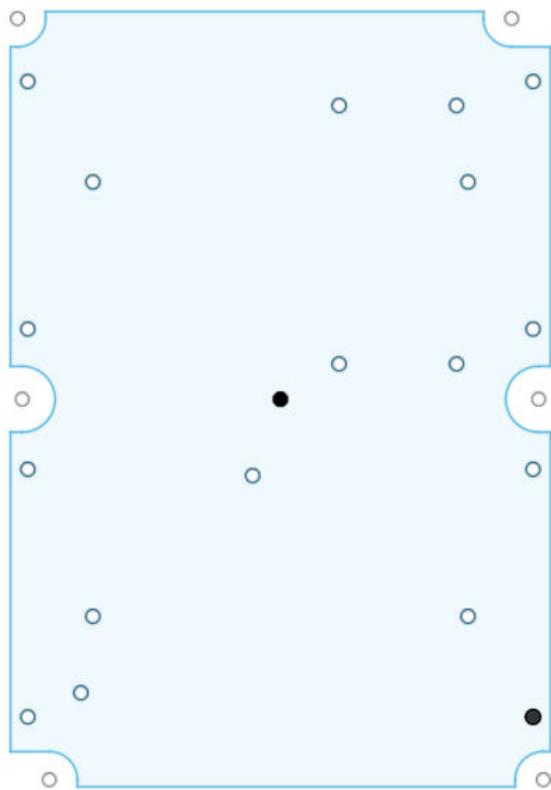


Layout en design (enclosure)

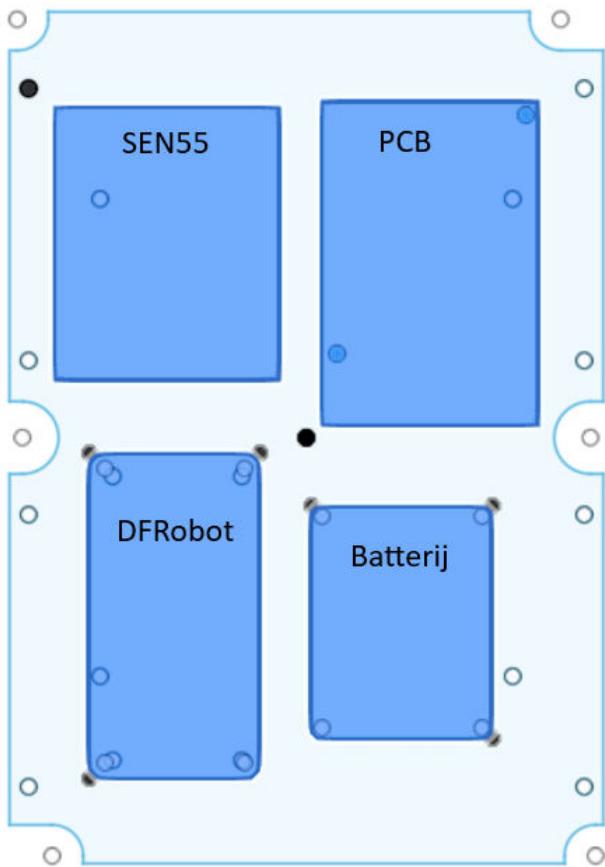
Backplate

Aangezien we alles gemakkelijk willen monteren hebben we gekozen om een backplate te monteren in onze installatie doos. Dit maakt zowel ontwikkeling als installatie handiger. Alle componenten worden individueel gemonteerd op deze backplate doormiddel van standoffs:

- PCB
- DFRobot
- Batterij
- SEN55 (temp, hum, VOC, NOx, fijnstof sensor)

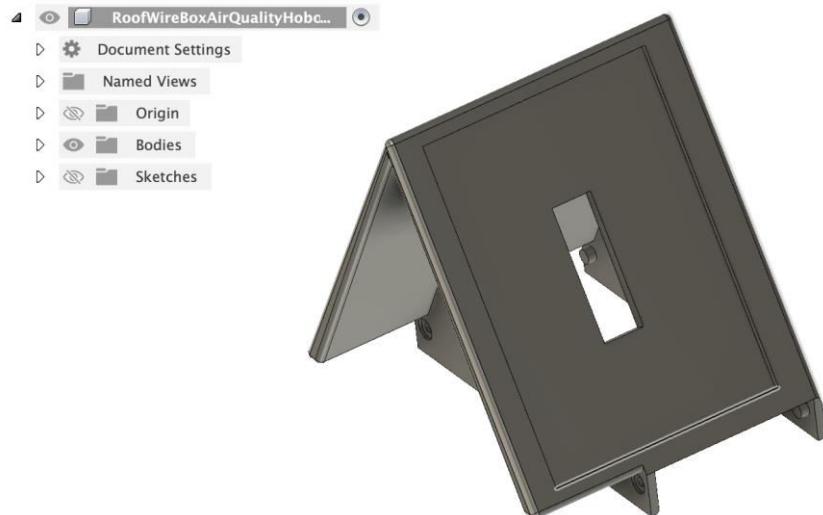


Section 1.01 Layout backplate



Dak

Aangezien vandalisme en het verstoren van de natuur een belangrijk topic is voor natuurnatuurpunt. Hebben we beslist om de sensoren op een bepaalde hoogte te hangen (bijvoorbeeld in een boom). Aangezien we niet een lelijke installatie doos in een boom willen hangen willen we deze dus camoufleren in de natuur, hierdoor hebben we gekozen om de installatie doos meer te doen lijken op een nestkastje. Dit gaan we doen door een 3D print te ontwikkelen die als puntdak zal gemonteerd worden op de bovenkant van de installatie doos. Voordeel hiervan is dat we ook een plaats hebben om onze zonnepanelen te monteren.



Hardware code

Setup

Zorg dat je PlatformIO hebt geïnstalleerd in je gekozen IDE: [PlatformIO installation](#)

Dit project is gemaakt met behulp van PIO in VSCode

Creëer een project met het *Arduino Framework* en het *Espressif ESP32 Dev Module* bord. Voeg volgende libraries toe aan het project via *PIO Home > Libraries*

- MCCI LoRaWAN LMIC library (by IBM)
- Sensirion Core (by Sensirion)
- Sensirion I2C SEN5X (by Sensirion)

Voeg volgende lijnen toe aan het `platformio.ini` bestand:

```
build_flags =  
    -D ARDUINO_LMIC_PROJECT_CONFIG_H_SUPPRESS  
    -D CFG_eu868=1  
    -D CFG_sx1276_radio=1  
monitor_speed = 115200  
monitor_filters =  
    time
```

Het volledige bestand hoort er uiteindelijk zo uit te zien (met onze versies van de lib_deps hieronder toegevoegd):

```
[env:esp32dev]  
platform = espressif32  
board = esp32dev  
framework = arduino  
lib_deps =  
    mcci-catena/MCCI LoRaWAN LMIC library@^4.1.1  
    sensirion/Sensirion Core@^0.6.0  
    sensirion/Sensirion I2C SEN5X@^0.3.0  
build_flags =  
    -D ARDUINO_LMIC_PROJECT_CONFIG_H_SUPPRESS  
    -D CFG_eu868=1  
    -D CFG_sx1276_radio=1  
monitor_speed = 115200  
monitor_filters =  
    time
```

Na de setup van het *platformio.ini* bestand, zouden alle libraries bij lib_deps geïnstalleerd moeten zijn.

WARNING! In versie 4.1.1 van de LMIC library is er een bug. De functie *hal_init()* is meerdere keren gedefinieerd. Navigeer naar *.pio/libdeps/esp32dev/MCCI LoRaWAN LMIC library/src/hal/hal.cpp* in je project directory. Vanaf lijn 416, comment heel de *hal_init()* functie weg.

De repository bevat een *configuration.h* en een *example_credentials.h* header file. Na het toevoegen van je OTAA-device aan The Things Network, plak de credentials in *example_credentials.h* en hernoem het bestand *credentials.h*.

Zorg dat alle `#define` instellingen correct zijn afgesteld in `configuration.h`. Wanneer je debugging aanzet, zal je uit deze repository manueel de library `lora_debug` moeten importeren. Deze vind je in de subdirectory `lib` van het PIO-project.

Het project moet nu builden zonder errors.

Creatie van de code

SEN55 air quality sensor

De eerste stap in het creëren van de source code was het maken van de SEN55 sensor code. De Sensirion I²C SEN5X library bevat het default I²C adres en voorziet een command interface voor de I²C commando's die naar de sensor moeten gestuurd worden.

De measurement loop ziet eruit als volgt:

- `sen55.begin(Wire)`

Dit initialiseert de sensor met het standaard I²C adres dat in de sen5x library zit

- `sen55.startMeasurement()`

Start de fan en de sensor op

- `sen55.readMeasuredValues(float &var1, float &var2, ...)`

Leest de gemeten waarden van de data registers van de sensor, en slaat ze op in de variabelen doorgegeven als argumenten aan de functie

- `sen55.stopMeasurement()`

Stop de sensor en de fan, de sensor keert terug naar idle

In het begin gaf de sensor geen geldige waarden terug voor VOC, NOx en alle PM. Na te debuggen zijn we erachter gekomen dat dit veroorzaakt werd door te snel de gemeten waarden te lezen. De sensor kan pas na ongeveer 2 minuten geldige metingen uitvoeren voor eerdergenoemde parameters. De status van de sensor en deze setup delay wordt gecontroleerd door de flag `senReady`.

De metingen worden returned in het datatype 'float'. Om over LoRa te sturen moet de data omgezet worden naar een byte array. Dit doen we door de float getallen om te zetten naar unsigned integers. De waarden worden maal 100 gedaan om de komma te laten wegvalLEN en bij de temperatuur wordt tien bijgeteld om negatieve waarden te vermijden. De rest van de waarden is altijd positief. De getallen worden dan gezet in `uint32_t` variabelen.

Daarna splitsen we de data op in bytes met behulp van deze for loop:

```
for (byte i = 0; i < 8; i++) {  
    txBuffer[0+i*2] = (data[i] >> 8) & 0xFF;  
    txBuffer[1+i*2] = (data[i]) & 0xFF;  
}
```

De twee most significant bytes (twee "linkse" bytes) worden weggelaten omdat de meetwaarden nooit hoger zijn dan 655,35 (dit is de hoogst mogelijke waarde die we kunnen versturen in 2 bytes)

In het TTN-dashboard moeten de waarden terug worden samengevoegd en gedeeld worden door 100, en bij de temperatuur moet 10 worden afgetrokken.

TODO: Voeg temperature & rel. humidity compensatie toe en VOC & NOx algorithm tuning parameters voor meer accurate metingen.

SX1276 LoRa module

De volgende stap was het toevoegen van LoRaWAN functionaliteit. We zijn vertrokken van een voorbeeldsketch van een van onze lectoren, die gemaakt was voor gebruik met de TTGO T-Beam. Dit bordje gebruikten we voor de tests doorheen het semester. Veel tijd is verloren gegaan aan het debuggen van dit voorbeeldproject. Er stond veel functionaliteit in die op project niet gebruikt, in verband met de power management chip op de T-Beam en de GPS die gebruikt werd in de les van de lector. Deze code moesten we eruit halen zonder TTN-functionaliteit eruit te halen. We hadden uiteindelijk een sketch die data verstuurde over LoRa met ABP, maar geen controle over *wanneer* we de data verstuurden. De interface library die de voorbeeldsketch gebruikt, verborg heel de werking van de achterliggende library, dus het was veel te moeilijk om uit te zoeken waarom het misliep.

Daarom is de TTN library van dit voorbeeld eruit gegooid en wordt enkel de achterliggende [MCCI Arduino LMIC library](#) gebruikt. De meeste code komt uit examples van de MCCI LMIC library zelf.

De LMIC-setup ziet eruit als volgt:

```
os_init();
LMIC_reset();

// Disable adaptive data rate/adaptive spread factor
LMIC_setAdrMode(false);

do_send(&sendjob);
```

os_init() initialiseert de SX-module, en *LMIC_reset()* zorgt ervoor dat alle instellingen gereset worden. *do_send()* is een callback functie waar de sen55 code staat voor het uitlezen en formatteren/converteren van de data naar de byte array, die dan vervolgens de byte array in de queue zet om verzonden te worden over LoRa

In de loop staat er vervolgens dit:

```
void loop() {
    ...
    os_runloop_once();
}
```

Deze functie wordt continu uitgevoerd (wanneer de flags juist staan) om te checken of er een *osjob* in de queue staat, wat zeggen dat er een byte array in de queue staat om verzonden te worden. Wanneer er data gequeued is, wordt het verstuurd over LoRa, wanneer het netwerk beschikbaar is

De LMIC library werkt aan de hand van ‘events’ die worden gelogd door de LoRa module. Volgende functie voert code uit op basis van het event dat getriggerd is.

```
void onEvent (ev_t ev) {...};
```

Het belangrijkste deel van deze functie is dit:

```
case EV_TXCOMPLETE:
    Serial.println(F("EV_TXCOMPLETE (includes waiting for RX windows)"));
    ...
    // Go to sleep on completion of TX
    gotosleep = true;
```

```
break;
```

Hier wordt de flag *gotosleep* op true gezet, wat de esp deep sleep triggert. De code execution steunt op het feit dat er een *EV_TXCOMPLETE* event wordt gelogd om verder te gaan, aangezien de code pas verder kan runnen door de ESP in slaap te brengen.

ESP32 deep sleep

Voor de sleep mode van de ESP32 wordt de *esp_sleep.h* library gebruikt die included is in *Arduino.h*. Om deep sleep te doen werken met LoRa moet heel wat code toegevoegd worden. De LoRa-instellingen worden opgeslagen op de ESP nadat de ESP is gejoined via OTAA. Deze gegevens zouden verloren gaan bij deep sleep, waardoor de ESP terug zou moeten joinen na wake-up. Om dit te vermijden moet de LMIC-configuratie worden opgeslagen in het RTC-geheugen van de ESP, dat de data niet verloren laat gaan in low-power modus. Dit gebeurt aan de hand van deze functie:

```
void saveLMICToRTC() {
    Serial.println("Saving LMIC to RTC memory");
    RTC_LMIC = LMIC;

    // Compensate availability timer values for each band; else it keeps going up and tx_start
    interval gets bigger
    unsigned long now = millis();
    for (int i = 0; i < MAX_BANDS; i++) {
        ostime_t correctedAvail = RTC_LMIC.bands[i].avail - ((now / 1000.0 +
    TX_INTERVAL) * OSTICKS_PER_SEC);
        if (correctedAvail < 0) {
            correctedAvail = 0;
        }
        RTC_LMIC.bands[i].avail = correctedAvail;
    }
    RTC_LMIC.globalDutyAvail = RTC_LMIC.globalDutyAvail - ((now / 1000.0 + TX_INTERVAL) *
    OSTICKS_PER_SEC);

    if (RTC_LMIC.globalDutyAvail < 0) {
        RTC_LMIC.globalDutyAvail = 0;
    }
}
```

RTC_LMIC is een variable van het type *RTC_DATA_ATTR* *lmic_t*. *lmic_t* is een custom type gedefinieerd in de LMIC library die de LoRa configuratie bevat, en *RTC_DATA_ATTR* geeft aan dat deze variabele is opgeslagen in het RTC-geheugen van de ESP32. De rest van de code is een berekening die bepaalde instellingen in de LMIC-configuratie compenseert, omdat in deep sleep de *os_time* (die gebruikt wordt door de LMIC library) niet correct geüpdateerde wordt.

De LMIC-configuratie wordt terug uit het RTC-geheugen geladen met *loadLMICFromRTC()*

De sleep mode wordt ingeschakeld in de functie `doDeepSleep()` die de timer set en vervolgens de ESP in deep sleep brengt.

De ESP sleep gaf nog een grote fout; de LoRa module bleef in de `OP_POLL` (polling) operation mode. Uit de documentatie: "send empty UP frame to ACK confirmed DN/fetch more DN data". We vermoeden dat de module geen antwoord krijgt hierop, waardoor het blijft vasthangen in de `OP_POLL` modus. Deze modus wordt dan mee opgeslagen in de `RTC_LMIC` variabele waardoor bij herstarten van de ESP deze opmode nog steeds actief is. Er was schijnbaar geen goede manier om

dit op te lossen, omdat de library de operation modes zelf aanpast en de verschillende benodigde functies uitvoert voor het correct verzenden van de data. De oplossing die we in onze code hebben gebruikt, is dit:

```
void removePollingState() {
    bit_t lmic_status = LMIC_queryTxReady();
    if (!lmic_status) {
        LMIC.opmode &= ~OP_POLL;
    }
}
```

Deze functie verwijdert *OP_POLL* uit de LMIC configuratie aan de hand van een bitwise operation. Dit lost effectief het probleem op. Dit is waarschijnlijk niet de correcte manier om dit op te lossen, maar omdat er geen hulp te vinden was over dit probleem en de library hier geen interface voor voorziet, gebruiken we dit. De code checkt of de module nog in een operating mode staat waarin gezonden wordt met behulp van *LMIC_queryTxReady()*. Door de vele tests hebben we gemerkt dat wanneer dit true geeft op het moment van uitvoeren, de module in *OP_POLL* mode zit. Vervolgens wordt de bit die correspondeert met de opmode *OP_POLL* op nul gezet.

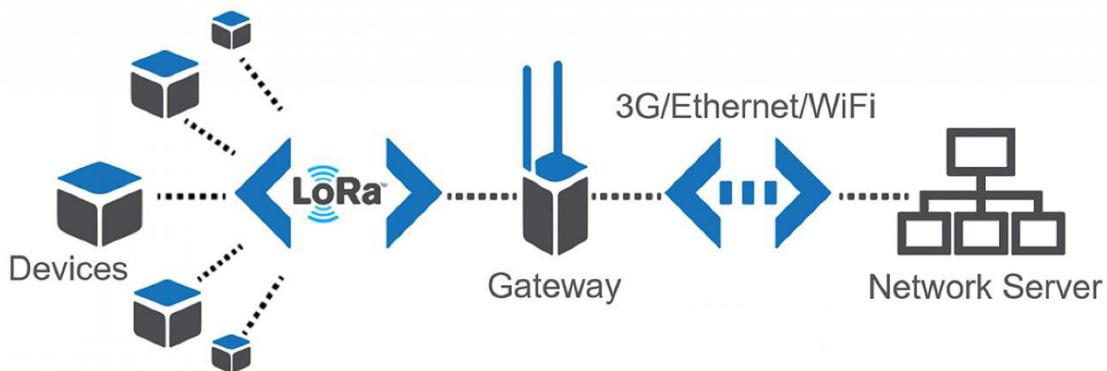
Deze functie wordt vlak voor het in deep sleep brengen van de ESP uitgevoerd om de opmode uit de RTC_LMIC-configuratie te houden.

TheThingsNetwork | LoRaWAN

TheThingsNetwork

Wat is het TheThingsNetwork

TheThingsNetwork (TTN) is een open source en door de gemeenschap gedreven project dat tot doel heeft een wereldwijd, crowd-sourced en gedecentraliseerd Internet of Things (IoT)-netwerk te creëren. Het biedt de infrastructuur en tools die nodig zijn om een LoRaWAN (Long Range Wide Area Network) voor IoT-apparaten te bouwen en te exploiteren. TTN is gebaseerd op LoRaWAN-technologie en wordt onderhouden door vrijwilligers en gemeenschappen. Het stelt bedrijven en ontwikkelaars in staat om IoT-oplossingen te bouwen boven op dit netwerk.



Figuur 1 Data overdracht sensor naar TTN

Hoe werkt TTN

Na het creëren van een end device en het genereren van de unieke sleutels en ID, volgt het volgende proces:

End Device Communicatie: Het end device, zoals een IoT-sensor of -apparaat, verzendt gegevens die het wil overdragen via het LoRaWAN-protocol.

Data Verpakt: De gegevens worden verpakt in LoRaWAN-pakketten, die de ID van het end device bevatten, evenals de beveiligingssleutels die nodig zijn voor de codering en beveiliging van de gegevens.

Verzending naar de Gateway: De LoRaWAN-pakketten worden via radiofrequentie verzonden naar een nabijgelegen gateway, die fungeert als een ontvangststation. Gateways kunnen overal in de omgeving zijn geïnstalleerd en kunnen de gegevens van meerdere end devices ontvangen.

Gateway naar Netwerk Server: De gateway ontvangt de pakketten, decodeert de gegevens en stuurt deze door naar een TTN-netwerkserver via een internetverbinding.

Netwerkserver: De TTN-netwerkserver ontvangt de gegevens van de gateway en is verantwoordelijk voor het routeren van de gegevens naar de juiste bestemming. Het verifieert ook de beveiligingssleutels om de integriteit en beveiliging van de gegevens te waarborgen.

Applicatie Server: De gegevens worden doorgestuurd naar een applicatieserver die specifiek is opgezet voor de toepassing of service waarmee de gegevens worden gebruikt. Deze server verwerkt en analyseert de gegevens en voert de nodige acties uit, zoals opslag, visualisatie of het triggeren van meldingen.

Terugkerende Communicatie: Als bi-directionele communicatie is vereist, kan de applicatieserver gegevens naar het end device sturen via dezelfde route, waarbij de server de gegevens naar de juiste gateway en uiteindelijk naar het end device routeert.

Beheer en Monitoring: TTN biedt beheer- en monitoringstools waarmee gebruikers de prestaties van hun netwerk kunnen volgen, apparaat registraties kunnen beheren en beveiligingsinstellingen kunnen configureren.

Stappenplan

- 1) Maak een applicatie aan.
- 2) Voeg een end device toe met volgende instellingen.
- 3) Genereer alle sleutels (keys).



End device type

Input method Enter end device specifics manually

Frequency plan * Europe 863-870 MHz (SF9 for RX2 - recommended)

LoRaWAN version * LoRaWAN Specification 1.0.0

Regional Parameters version * TS001 Technical Specification 1.0.0

Activation mode Activation by personalization (ABP)

Additional LoRaWAN class capabilities None (class A only)

Network defaults Use network's default MAC settings

Cluster settings Skip registration on Join Server

Provisioning information

DevEUI 70 B3 D5 7E D0 06 1E 33 2/50 used

Device address * 26 0B B7 EF

AppSKey * 2C AE C4 1F 4C F3 56 55 F1 92 71 C2 2F 7D 57 CE

NwkSKey * 5F 41 7C 5C 9B 81 D8 B8 76 12 51 A9 4D 9B 91 46

End device ID * sen55-meting
This value is automatically prefilled using the DevEUI

After registration

View registered end device

Register another end device of this type

Register end device

EXTRA! Bij ABP ga naar *general setting* < *Network layer* < *advanced MAC settings* < *Reset frame counters*

Dit zorgt ervoor dat dezelfde sensor herkend wordt nadat deze uit is getrokken en later terug gebruikt moet worden.

- 4) Voeg nu de nodige keys toe aan de credentials in de code voor de sensor.

```

/* Credentials file */
#pragma once

// Only one of these settings must be defined
#define USE_ABP
//#define USE_OTAA
#ifndef USE_ABP

    // LoRaWAN NwkSKey, network session key
    static const u1_t PROGMEM NWKSKEY[16] = { Voeg hier de NWSKEY toe };
    // LoRaWAN AppSKey, application session key
    static const u1_t PROGMEM APPSKEY[16] = { Voeg hier de APPSKEY toe };
    // LoRaWAN end-device address (DevAddr)
    // This has to be unique for every node
    static const u4_t DEVADDR = 0xVoeg hier de DEVADDR toe;

#endif

```

- 5) Upload de code met de gewenste sensor code.
- 6) *Payload Formatter < Uplink* voor het decoderen van bytes. (Voorbeeld voor SPG41)

```

SPG41
function Decoder(bytes, port) {
    var decoded = {};
    decoded.VOC = (bytes[0] << 8) + bytes[1];
    decoded.NOX= (bytes[2] << 8) + bytes[3];
    return decoded;
}

```

```

SEN55
function Decoder(bytes, port) {
    var decoded = {};
    decoded.pm1p0 = (bytes[0] << 8) + bytes[1] /100;
    decoded.pm2p5= (bytes[2] << 8) + bytes[3] /100;
    decoded.pm4p0= (bytes[4] << 8) + bytes[5] /100;
    decoded.pm10p0= (bytes[6] << 8) + bytes[7] /100;
    decoded.hum= (bytes[8] << 8) + bytes[9] /100;
    decoded.temp= (((bytes[10] << 8) + bytes[11]) /100) -10;
    decoded.voc= (bytes[12] << 8) + bytes[13] /100;
    decoded.nox= (bytes[14] << 8) + bytes[15] /100;
    return decoded;
}

```

LoRaWAN

Wat is LoRaWAN

LoRaWAN is een draadloos communicatieprotocol dat is ontworpen voor lange afstandsconnectiviteit van batterij gevoede IoT-apparaten. Het biedt een groot bereik, laag energieverbruik en wordt veel gebruikt in toepassingen zoals smart cities en industriële automatisering. Het is een open standaard en kan zowel publieke als private netwerken omvatten.

Specificaties

De specificaties van LoRaWAN (Long Range Wide Area Network) omvatten de volgende belangrijke kenmerken:

- **Frequentiebanden:** LoRaWAN werkt op de 868 MHz-band in Europa.
- **Lange Afstand:** Het biedt connectiviteit over lange afstanden, variërend van enkele kilometers tot tientallen kilometers, afhankelijk van de omgeving en de frequentieband.
- **Laag Energieverbruik:** LoRaWAN is ontworpen voor energieuinigheid, waardoor batterij gevoede IoT-apparaten maanden tot jaren op een enkele batterij kunnen werken.
- **Bij schaalbaarheid:** LoRaWAN-netwerken hebben meestal een gelaagde structuur die schaalbaarheid mogelijk maakt. Dit omvat eindapparaten (devices), gateways en een netwerkserver.
- **Adaptieve Modulatie:** LoRaWAN past de modulatie aan op basis van de afstand en omgevingsomstandigheden, waardoor het betrouwbaarheid en prestaties levert over variabele afstanden.
- **Bidirectionele Communicatie:** Het ondersteunt tweerichtingscommunicatie, waarbij apparaten gegevens kunnen verzenden en ontvangen via LoRaWAN-netwerken.
- **Beveiliging:** LoRaWAN bevat beveiligingsmaatregelen om gegevens te beschermen, waaronder encryptie van gegevens en apparaat authenticatie.
- **Open Standaard:** LoRaWAN is een open standaard, wat betekent dat het niet afhankelijk is van een specifieke fabrikant en interoperabiliteit tussen verschillende apparaten mogelijk maakt.
- **Join Procedure:** Apparaten moeten zich aanmelden bij een LoRaWAN-netwerk voordat ze gegevens kunnen verzenden of ontvangen. Dit wordt doorgaans gedaan via een join procedure.
- **Publieke en Private Netwerken:** LoRaWAN-netwerken kunnen zowel publiek als privaat zijn, afhankelijk van de toepassing en de beheerder.

Opzetten van de server (back-end)

Wat hebben we nodig

Het opzetten van onze back-end gebeurt op een Hetzner server. Op deze server maken we een nieuwe gebruiker "esp32aqs" aan. Om alles in te stellen en op te zetten gebruiken we het stappenplan uit de documentatie van het voorgaande project waarbij de back-end er gelijkaardig uitziet.

Zanzibar projectdocumentatie back-end

Voor de back-end van ons project maken we gebruik van vier essentiële services:

Node-Red: Deze service wordt ingezet om gegevens op te halen van TheThingsNetwork, deze te verwerken en vervolgens in de 'InfluxDB' database op te slaan.

InfluxDB: We gebruiken InfluxDB als onze database-oplossing. Hierin worden de gegevens opgeslagen die we verzamelen en verwerken met Node-Red.

Grafana: Om de verzamelde gegevens op een visueel aantrekkelijke manier te presenteren, vertrouwen we op Grafana. Met Grafana kunnen we de gegevens in onze Influx-database grafisch weergeven.

Caddy: Is een reverseproxy server die het toelaat om webservers aan de buitenkant open te zetten zonder extra poorten open te zetten. Ideaal om dus onze Grafana van het internet bereikbaar te maken

Deze combinatie van services stelt ons in staat om gegevens efficiënt te verzamelen, verwerken en presenteren in ons project

Docker

We hebben besloten om deze vier services binnen een Linux-omgeving op te zetten met behulp van containers. Het gebruik van containers biedt tal van voordelen, waaronder aanzienlijke flexibiliteit en een hoge mate van veiligheid. Containers zijn namelijk geïsoleerd van elkaar, wat betekent dat elke service in zijn eigen afzonderlijke omgeving opereert. Dit zorgt ervoor dat eventuele problemen of wijzigingen in één container de andere containers niet beïnvloeden. Bovendien maakt het opzetten van services in containers het proces van ontwikkelen, testen en implementeren aanzienlijk efficiënter en meer schaalbaar.

Docker-netwerk

Om effectieve communicatie tussen de Docker containers mogelijk te maken, is het essentieel dat ze zich binnen hetzelfde Docker-netwerk bevinden. Dit zorgt ervoor dat de containers naadloos met elkaar kunnen communiceren en gegevens kunnen uitwisselen. Door ze in hetzelfde netwerk te groeperen, kunnen we ervoor zorgen dat de services en applicaties die deze containers hosten, moeiteloos en efficiënt met elkaar kunnen samenwerken, waardoor de algehele systeemprestaties worden geoptimaliseerd.

Opzetten

1. Zet een Linux server op (bij voorkeur --> Debian)
2. Installeer zowel Docker als Docker Compose
3. Clone de repository van [gitlab](#)

4. Hier zie je de volgende docker-compose.yaml file in staan, deze gaat voor ons automatisch de Docker containers opzetten in de juiste volumes.

docker-compose.yaml:

```
version: "3.8"

services:
  node-red:
    container_name: node-red-air-quality-hoboken
    image: nodered/node-red:latest
    restart: unless-stopped
    environment:
      - TZ=Europe/Brussels
    ports:
      - '1880:1880'
    networks:
      - air-quality-hoboken
    volumes:
      - ./node-red-data:/data

  grafana:
    container_name: grafana-air-quality-hoboken
    image: grafana/grafana:latest
    restart: unless-stopped
    environment:
      - TZ=Europe/Brussels
    ports:
      - '3001:3000'
    networks:
      - air-quality-hoboken
    volumes:
      - ./grafana-data:/var/lib/grafana

  influxdb:
    container_name: influx-db-air-quality-hoboken
    image: influxdb:latest
    networks:
      - air-quality-hoboken
    security_opt:
      - no-new-privileges:true
    restart: unless-stopped
    ports:
      - '8086:8086'
    volumes:
      - ./influxdb2-data/config:/etc/influxdb2
      - ./influxdb2-data/appdata:/var/lib/influxdb2

  caddy:
    container_name: caddy-db-air-quality-hoboken
    image: caddy:latest
    restart: unless-stopped
    cap_add:
      - NET_ADMIN
    ports:
      - "80:80"
      - "443:443"
      - "443:443/udp"
    volumes:
```

```

- ./caddy-data/Caddyfile:/etc/caddy/Caddyfile # Make first this
file before running this compose "touch Caddyfile"
- ./caddy-data/site:/srv
- ./caddy-data/caddy_data:/data
- ./caddy-data/caddy_config:/config
networks:
- air-quality-hoboken

networks:
air-quality-hoboken:

```

- Bouw de Docker containers met behulp van Docker Compose: zorgt dat je in de folder bent waar de docker-compose file zich bevindt en voer het volgende commando uit:

docker-compose up -d

- (Bij errors kan het zijn dat de permissions van de volumes niet goed staan, pas deze aan)

Om de Docker containers te testen kan je gebruik maken van het volgende “ssh commando” om aan de web interface aan te kunnen zonder dat de poort op deze server aan de buitenkant werkelijk openstaat

7.

ssh -L <port-container>:localhost:<port-container> esp32aqs@195.201.26.105

Container	Port (web interface)
Node-red	1880
InfluxDB	8086
Grafana	3001

Node-Red

Wat is Node-Red

Node-RED is een visuele programmeeromgeving waarmee je Internet of Things (IoT) en automatiseringstaken kunt maken en beheren. Het is gebaseerd op knooppunten die je kunt slepen en neerzetten om workflows te bouwen. Deze knooppunten vertegenwoordigen verschillende functies en apparaten en kunnen worden verbonden om gegevens te sturen, verwerken en integreren.

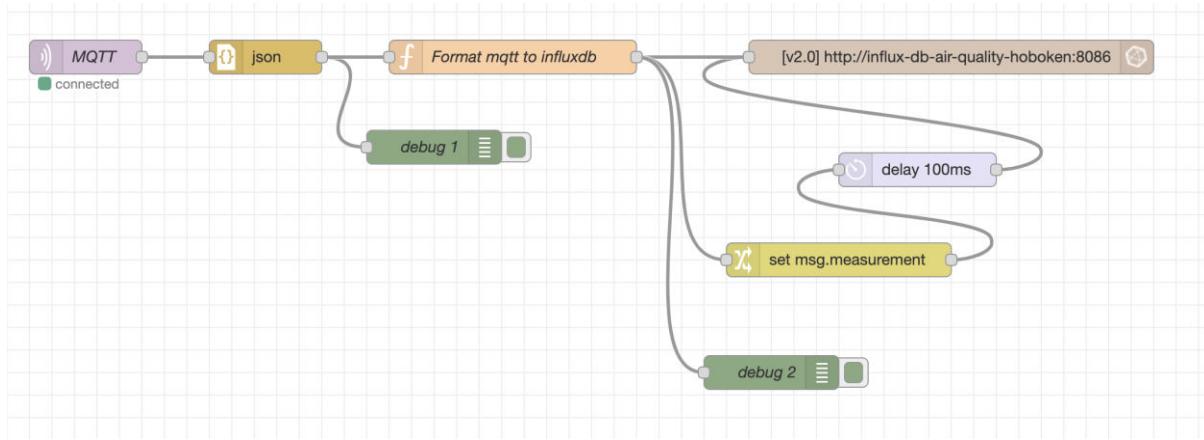
Node-RED is populair vanwege zijn gebruiksvriendelijke interface en flexibiliteit, waardoor het ideaal is voor taken zoals gegevensverwerking, het aansturen van hardware, en het automatiseren van taken in huis of op het werk. Het wordt vaak gebruikt in combinatie met Raspberry Pi en andere IoT-apparaten. Het biedt ondersteuning voor tal van protocollen en services, waardoor je gegevens van verschillende bronnen kunt verzamelen en ermee kunt interageren.

In essentie is Node-RED een tool waarmee je eenvoudig IoT- en automatisering workflows kunt ontwerpen zonder uitgebreide programmeervaardigheden te hebben.

Waarom gebruiken wij Node-Red

Node-RED is een uitstekende keuze voor het stroomlijnen van het proces om gegevens van TheThingsNetwork via MQTT binnen te halen en specifieke waarden uit de JSON-gegevens te filteren voordat deze worden opgeslagen in onze database, INFLUXDB. Node-RED biedt een intuïtieve visuele interface waarmee we gemakkelijk de MQTT-gegevensstroom kunnen vastleggen, de JSON-berichten kunnen analyseren en alleen de relevante informatie kunnen selecteren die we willen bewaren. Het bespaart ons tijd en vereenvoudigt het gegevensverwerkingsproces, waardoor we gegevens efficiënt kunnen opslaan en analyseren in INFLUXDB voor verdere inzichten en acties.

Schema



Blokken

Instellingen MQTT-blok → Bij het aanmaken van een MQTT integration een API-key genereren en deze opslaan (deze ga je later nodig hebben als wachtwoord in Node-Red).

MQTT

AirQualityHoboken

- Overview
- End devices
- Live data
- Payload formatters
- Integrations
 - MQTT**
 - Webhooks
 - Storage Integration
 - AWS IoT

Connection information

MQTT server host

- Public address: eu1.cloud.thethings.network:1883
- Public TLS address: eu1.cloud.thethings.network:8883

Connection credentials

Username	air-quality-hoboken@ttn
Password	Generate new API key Go to API keys

© 2023 The Things Stack by The Things Network and The Things Industries

EN v3.27.2 (7613ca53c)

MQTT node:

- Server: Hier vul je de naam of het IP-adres van de MQTT-broker in, samen met de poort (meestal 1883).
- Action:
 - Subscribe to Single Topic (Abonneren op enkelvoudig topic): Hiermee abonneer je de MQTT-node op een specifiek MQTT-topic dat je vooraf instelt.
 - Dynamic Subscription (Dynamisch abonnement): Hiermee kun je het MQTT-topic dynamisch instellen op basis van informatie in het inkomende bericht. Dit maakt je abonnement flexibel en afhankelijk van de gegevens die je ontvangt.

- Topic: Dit is een tekstlabel waarmee je berichten kunt categoriseren. Abonneer je op het gewenste MQTT-topic voor de juiste data te krijgen. In dit geval abonneren we op alle sensoren van air-quality-hoboken dit doen we zodat de back-end dynamisch is en dat je eenvoudig een extra sensor kan toevoegen zonder dat je aanpassingen moet doen aan de back-end of Grafana front-end.
- QoS: Kies de gewenste kwaliteit van service voor berichten (0, 1 of 2).

Edit mqtt in node

Properties	
Server	eu1.cloud.thethings.network:1883
Action	Subscribe to single topic
Topic	v3/air-quality-hoboken@ttn/devices/+/up
QoS	0
Output	auto-detect (string or buffer)
This option is deprecated. Please use the new auto-detect mode.	
Name	MQTT

Security: Vul de juiste username en wachtwoord in. Dit is de username en wachtwoord (API-key) die je in het begin hebt aangemaakt in TTN.

Edit mqtt in node > Edit mqtt-broker node

Properties		
Name	Name	
Connection	Security	Messages
Username	air-quality-hoboken@ttn	
Password	*****	

JSON node:

We plaatsen de JSON node voor de inkomende berichten naar een JSON-formaat te brengen.



Function node (Format MQTT to InfluxDB):

Deze node is noodzakelijk om de JSON-waardes om te zetten naar waardes die de InfluxDB out blok verwacht.

```
Bv: msg.payload = {  
    temp: 24.0,  
    hum: 12.1,  
    ...  
}
```

Hierbij verwacht influxDB-out blok 2 belangrijke zaken. Hij verwacht alle data fields in de msg.payload die in de database wilt hebben.

Ook verwacht hij een msg.measurement. Dit is de measurement die hij zijn fields values naar toe zal sturen in de database bucket. Elke lora sensor krijgt hier zijn measurement, hiervoor hebben we de device_id nodig uit te JSON en zetten we dit om naar de msg.measurement naam.

Bij deze node zullen we de ontvangen payload omzetten naar een zeer specifiek onderdeel (alleen het gene wat wij willen) van de payload en deze vervolgens doorsturen. Ook ga hij checken of de waardes niet verkeerd zijn, dit kan af en toe gebeuren omdat de sen55 eventjes nodig heeft om op te starten. Is deze niet opgestart zal deze valse informatie versturen, die willen we niet in onze database hebben natuurlijk.

Functie node (Format MQTT to InfluxDB)

```
let device_id = msg.payload.end_device_ids.device_id;  
let rssi = msg.payload.uplink_message.rx_metadata[0].rssi  
let longitude = msg.payload.uplink_message.locations.user.longitude  
let latitude = msg.payload.uplink_message.locations.user.latitude  
let pm1p0 = msg.payload.uplink_message.decoded_payload.pm1p0  
let pm2p5 = msg.payload.uplink_message.decoded_payload.pm2p5  
let pm4p0 = msg.payload.uplink_message.decoded_payload.pm4p0  
let pm10p0 = msg.payload.uplink_message.decoded_payload.pm10p0  
let hum = msg.payload.uplink_message.decoded_payload.hum  
let temp = msg.payload.uplink_message.decoded_payload.temp  
let voc = msg.payload.uplink_message.decoded_payload.voc  
let nox = msg.payload.uplink_message.decoded_payload.nox  
// let battery = msg.payload.uplink_message.decoded_payload.battery  
  
msg.measurement = device_id
```

```

msg.payload = {
  device_id: device_id,
  rssi: rssi,
  longitude: longitude,
  latitude: latitude,
  pm1p0: ((pm1p0 < 300) ? pm1p0 : null),
  pm2p5: ((pm2p5 < 300) ? pm2p5 : null),
  pm4p0: ((pm4p0 < 300) ? pm4p0 : null),
  pm10p0: ((pm10p0 < 300) ? pm10p0 : null),
  hum: ((hum < 300) ? hum : null),
  temp: ((temp < 300) ? temp : null),
  voc: ((voc < 300) ? voc : null),
  nox: ((nox < 300) ? nox : null),
  //battery: ((battery < 101) ? nox : battery),
}

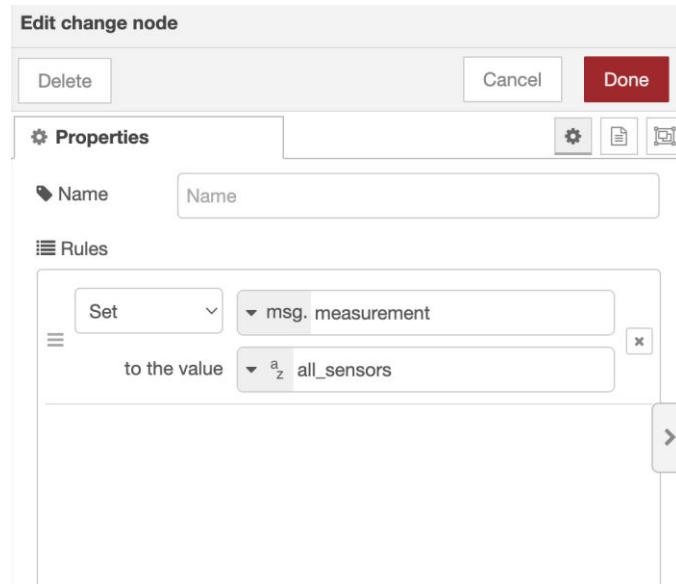
return msg;

```

Change-node (set msg.measurement)

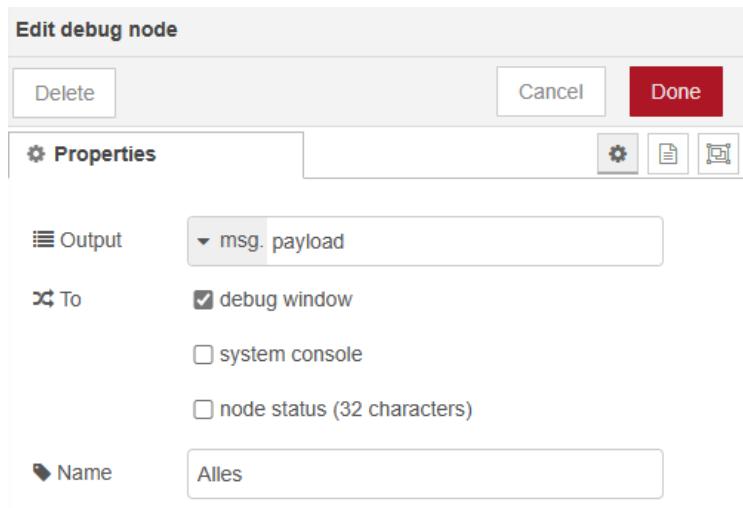
Zoals eerder hierboven al vermeld is geweest zal elke TTN-sensor zijn eigen measurement hebben in de bucket van de database. Ook hebben we gekozen om een measurement te hebben waar ook alle informatie van alle TTN-sensors inkomen, zodat je in de front-end (Grafana) een gemiddelde van alle sensoren kan bekijken. Hierdoor gaan we nadat we de data in de database hebben gedaan onder een bepaalde measurement diezelfde data nog eens in de database zal zetten onder een gezamenlijke measurement naam "all_sensors".

Hierdoor gaan we de measurement veranderen naar een vaste string "all_sensors".



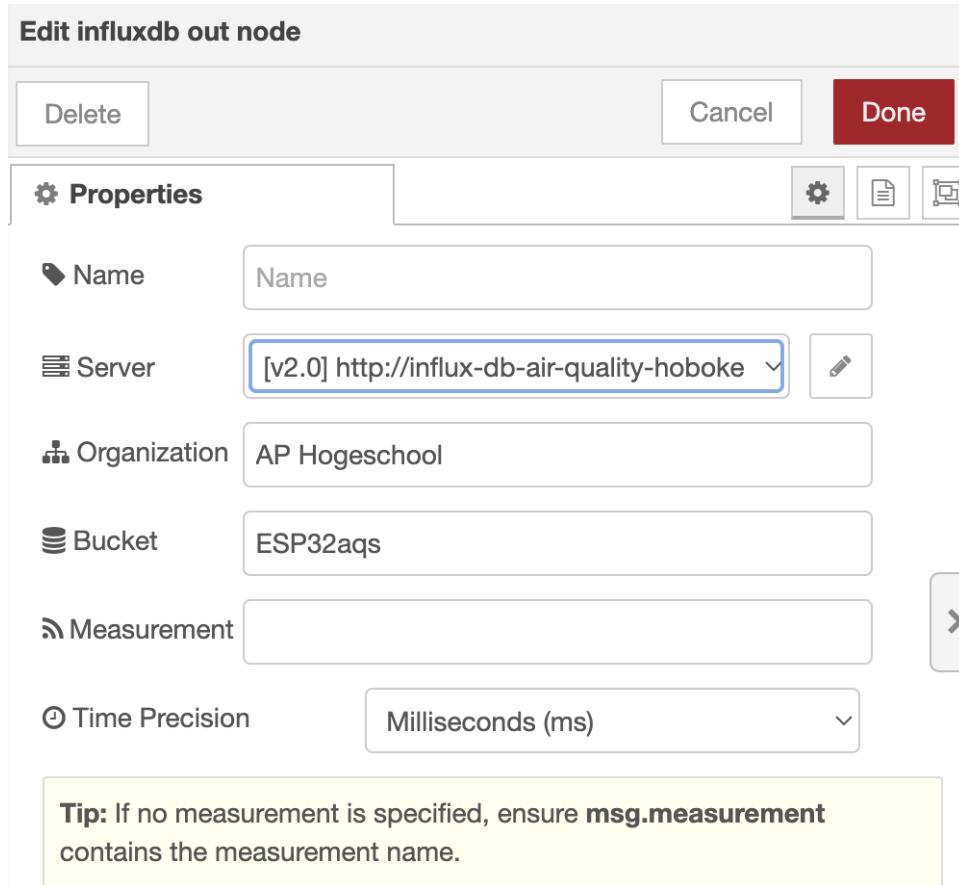
Debug node:

In deze node gaan we de data weergeven in de console.



InfluxDB out node:

Hierna gaan we een msg.payload informatie in de database zetten. In de bucket 'ESP32aqs', de measurement laten we leeg en geven het dus mee aan dit blok in de msg.measurement (kijk 'functie blok' beschrijving)



Het connecteren met de InfluxDB is eenvoudig!

Je geeft het IP-adres in van de InfuxDB Docker container of nog beter de domeinnaam van de Docker container (want het IP kan dus veranderen), de domeinnaam is hetzelfde naam als je Docker container name. En je zorgt dat je een API-token aanmaakt zoals in topic van InfluxDB is besproken. Als bucket kies je de bucket (database) die je hebt aangemaakt.

Edit influxdb out node > **Edit influxdb node**

<input type="button" value="Delete"/>	<input type="button" value="Cancel"/>	<input type="button" value="Update"/>
Properties		
	Name	<input type="text"/>
	Version	2.0
	URL	<input type="text" value="http://influx-db-air-quality-hoboken:8086"/>
	Token	<input type="text" value="....."/>
<input type="checkbox"/> Verify server certificate		<input type="button" value=">"/>

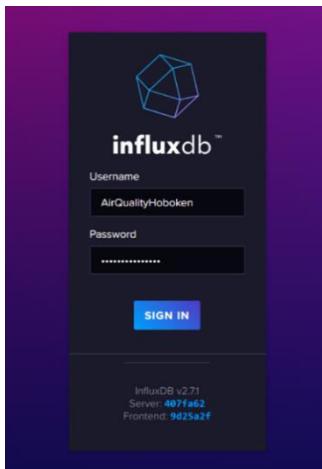
Database | InfluxDB & Docker

Als database voor ons project gebruiken we InfluxDB. InfluxDB is een tijdreeksdatabase die we kunnen gebruiken om IoT sensordata op te slaan.

We kunnen daarna de database beheren en data bekijken via de ingebouwde web interface. We krijgen toegang via de browser op <http://localhost:8086/> als we het volgende commando in de terminal typen

```
ssh -L 8086:localhost:8086 esp32aqs@195.201.26.105
```

In deze web interface moeten we vervolgens een nieuwe gebruiker en wachtwoord aanmaken. We kunnen dan een bucket aanmaken. Een bucket kan je vergelijken met de database waarin onze data zal worden opgeslagen. Wij maken de bucket “*ESP32aqs*” aan. We krijgen vervolgens ook een API-token te zien. Het is belangrijk dat je deze bewaart want je krijgt deze maar één keer te zien. Als we dit gedaan hebben kunnen we vervolgens inloggen in de database met de juist gemaakte gebruiker.



Als we ingelogd zijn kunnen we in de interface data toevoegen, buckets aanmaken en de data van onze buckets visualiseren.

Een "bucket" in InfluxDB is een container waarin tijdreeksgegevens worden opgeslagen. Buckets kunnen worden geconfigureerd met instellingen zoals hoe lang gegevens worden bewaard (tijdretentiebeleid) en welke verwerking erop wordt toegepast. Ze helpen bij het georganiseerd opslaan en beheren van gegevens, in ons geval hebben we gekozen om de gegevens voor altijd te bewaren in de bucket "*ESP32aqs*".

_start	_stop	_time	_value	_field	_measurement
2023-10-19T19:26:19Z	2023-10-19T19:26:19Z	2023-10-19T19:26:19Z	91	NOX	Sensordata
2023-10-19T19:26:19Z	2023-10-19T19:26:19Z	2023-10-19T19:26:19Z	282	NOX	Sensordata

Front-end | Grafana

Wat is Grafana

Grafana is een open-source datavisualisatie- en monitoringplatform dat wordt gebruikt voor het analyseren en weergeven van gegevens uit verschillende bronnen. Het stelt gebruikers in staat om grafieken, diagrammen en dashboards te maken om inzicht te krijgen in gegevens, zoals systeemprestaties, netwerkstatistieken, sensordata en meer. Grafana kan worden geïntegreerd met verschillende gegevensbronnen, zoals databases, Cloud services en monitoringstools, waardoor het een krachtig instrument is voor het bewaken en analyseren van gegevens in real-time. Wij gaan het gebruiken om gegevens te visualiseren en besluitvorming te ondersteunen.

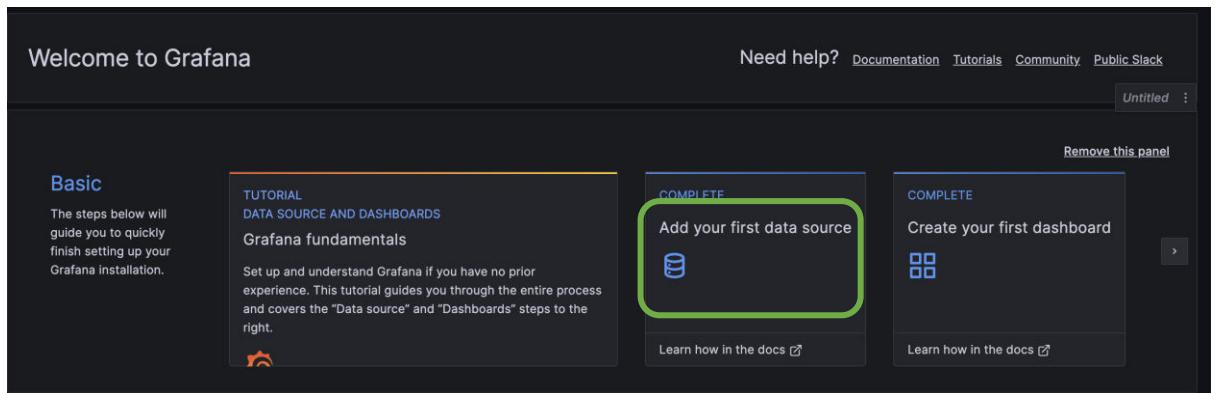
Opzetten grafana

Het belangrijkste om eerst te doen is het opzetten van een datasource, Grafana gaat natuurlijk ergens zijn data van heen moeten halen, hierbij gaan we de Grafana dus rechtstreeks laten communiceren met de InfluxDB. En gaan deze dus grafisch weergegeven worden doormiddel van Grafana dashboards.

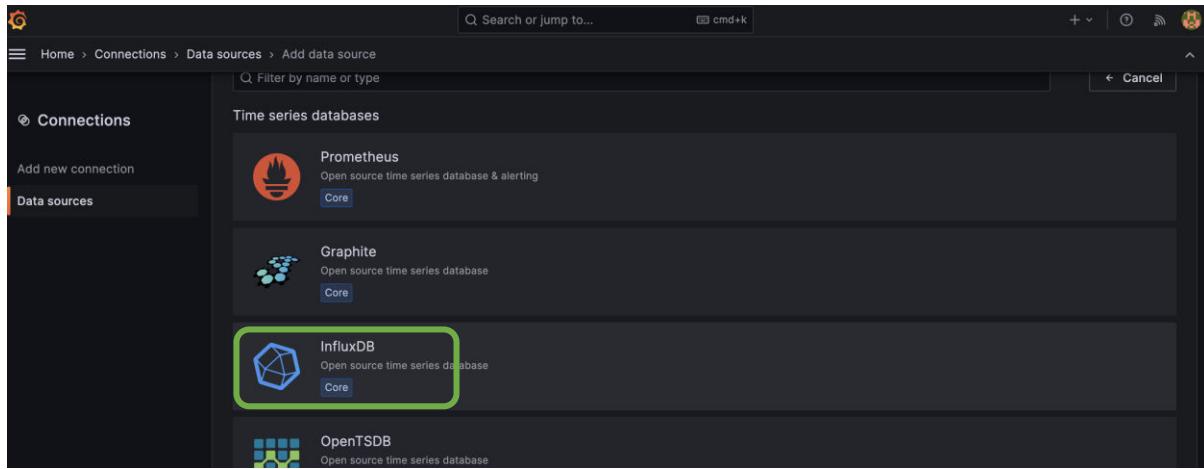
1. Ga naar de webserver van Grafana op port 3001. Dit kan door de volgende URL te gebruiken: <http://npunt.iot-ap.be/>. Indien de URL niet bereikbaar is kan je het volgende doen. Geef deze URL in: <http://localhost:3001> na dat je het volgende commando hebt gebruikt.

```
ssh -L 3001:localhost:3001 esp32aqs@195.201.26.105
```

2. Hierbij druk je op **data sources**



3. Selecteer dan uiteraard InfluxDB



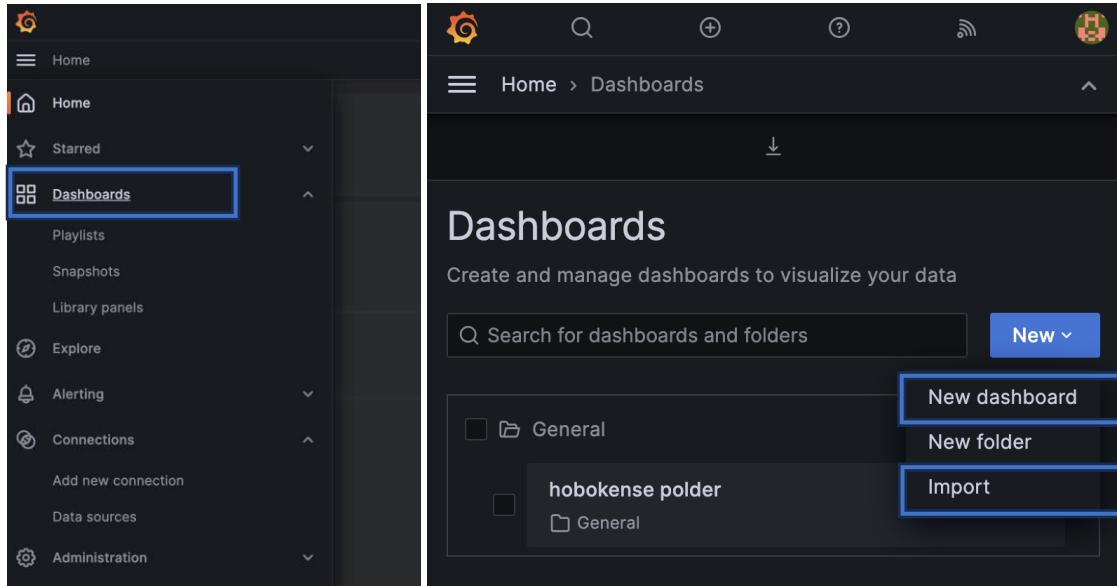
4. Vul nu de volgende waardes in

- Kies een naam
- Query Language: Flux
- URL: <http://influx-db-air-quality-hoboken/:8086>
- Basic Auth: off
- Organization: Voer hier de organisatie in die je hebt aangemaakt
- Token: Voer hier de API-token van InfluxDB
- Default bucket: vul hier de bucket die je hebt aangemaakt
- Save en test de connectie

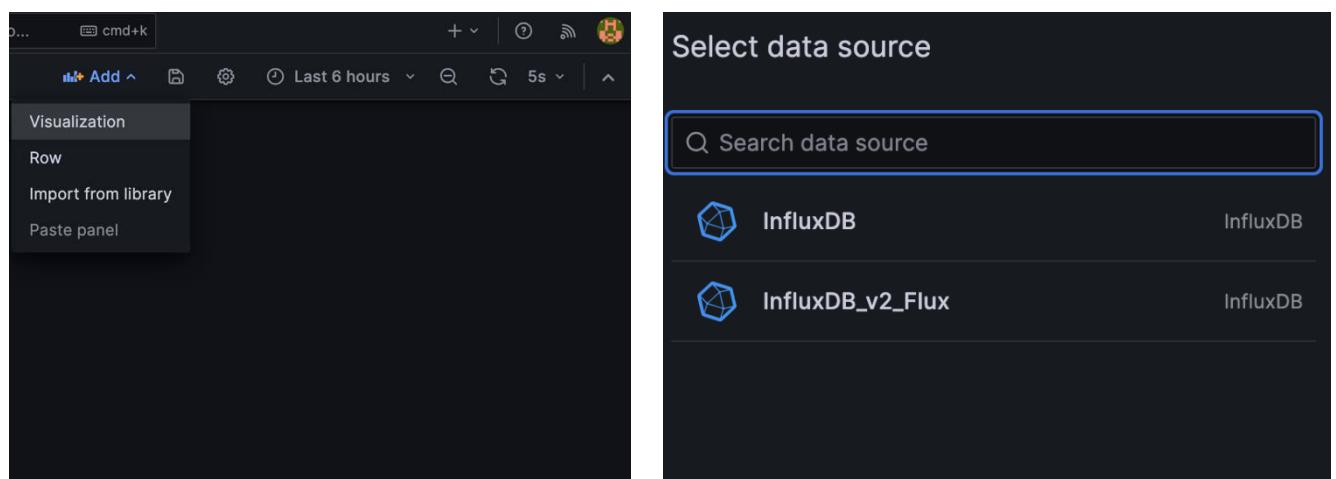
The screenshot shows the 'InfluxDB' configuration dialog in Grafana. It includes sections for 'Query Language' (set to 'Flux'), 'HTTP' (URL: http://172.20.0.2:8086, Allowed cookies: New tag (enter key to add), Timeout: 3), 'Auth' (Basic auth: With Credentials, TLS Client Auth: With CA Cert., Skip TLS Verify: off, Forward OAuth Identity: off), 'Custom HTTP Headers' (+ Add header), and 'InfluxDB Details' (Organization: AP Hogeschool, Token: configured, Default Bucket: ESP32aqs, Min time interval: 10s, Max series: 1000). At the bottom, a message says 'datasource is working. 3 buckets found'.

Opzetten dashboard en grafieken

1. Voor we een panel kunnen maken moeten we eerst een dashboard aanmaken. Dit kan je doen onder: Dashboard > New. Nu heb je de keuze om een nieuw leeg dashboard aan te maken of eentje te importen. Indien je die van ons wilt importen kan je dit doormiddel van dit json bestand -> [gitlab bestand](#) (in dien je deze optie kiest moet je volgende stappen niet per se meer volgen)



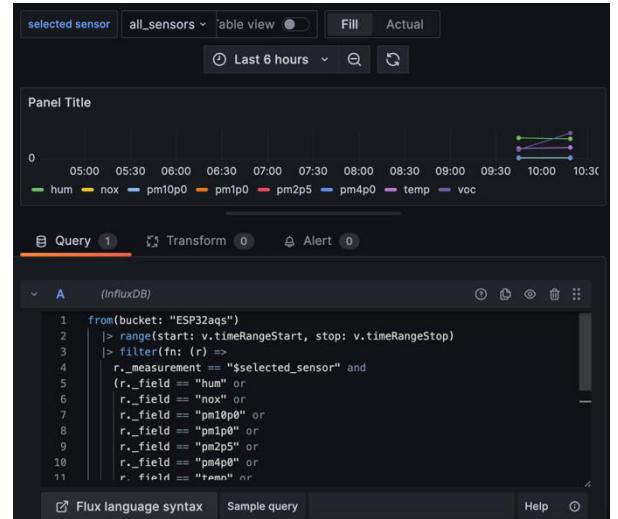
2. Nu kan je een panel/grafiek aanmaken. Door in je dashboard van boven in de balk "Add" aan te duiden en vervolgens de keuze "Visualization" te kiezen.
3. Selecteer dan jouw datasource die we eerder hebben aangemaakt.



4. Kies een type grafiek die je zou willen hebben

5. Maak een query naar de datasource: bijvoorbeeld deze

```
from(bucket: "ESP32aqs")
|> range(start: v.timeRangeStart, stop: v.timeRangeStop)
|> filter(fn: (r) =>
  r._measurement == "$selected_sensor"
and
  (r._field == "hum" or
  r._field == "nox" or
  r._field == "pm10p0" or
  r._field == "pm1p0" or
  r._field == "pm2p5" or
  r._field == "pm4p0" or
  r._field == "temp" or
  r._field == "voc")
)
```



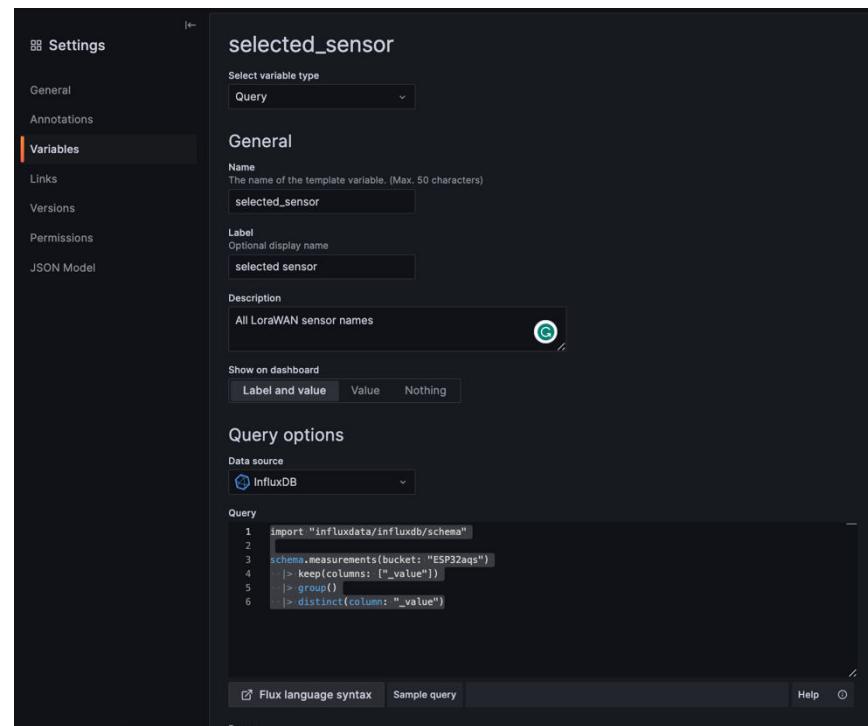
Aangezien ons dashboard voor meerdere sensors bevat, was er dus de nood om te wisselen tussen verschillende sensors. Dit kunnen we doen door de “r.measurement” in onze query dynamisch te maken (met een grafana variabele).

6. Maak een grafana variabele met een query naar al de measurement namen in de database. Dashboard Settings > Variables > + New Variable
7. Gebruik de volgende parameters: variable type = “Query”, name = “selected_sensor”, label “selected sensor”, show on dashboard = “label and value”, data source = jouw data source.

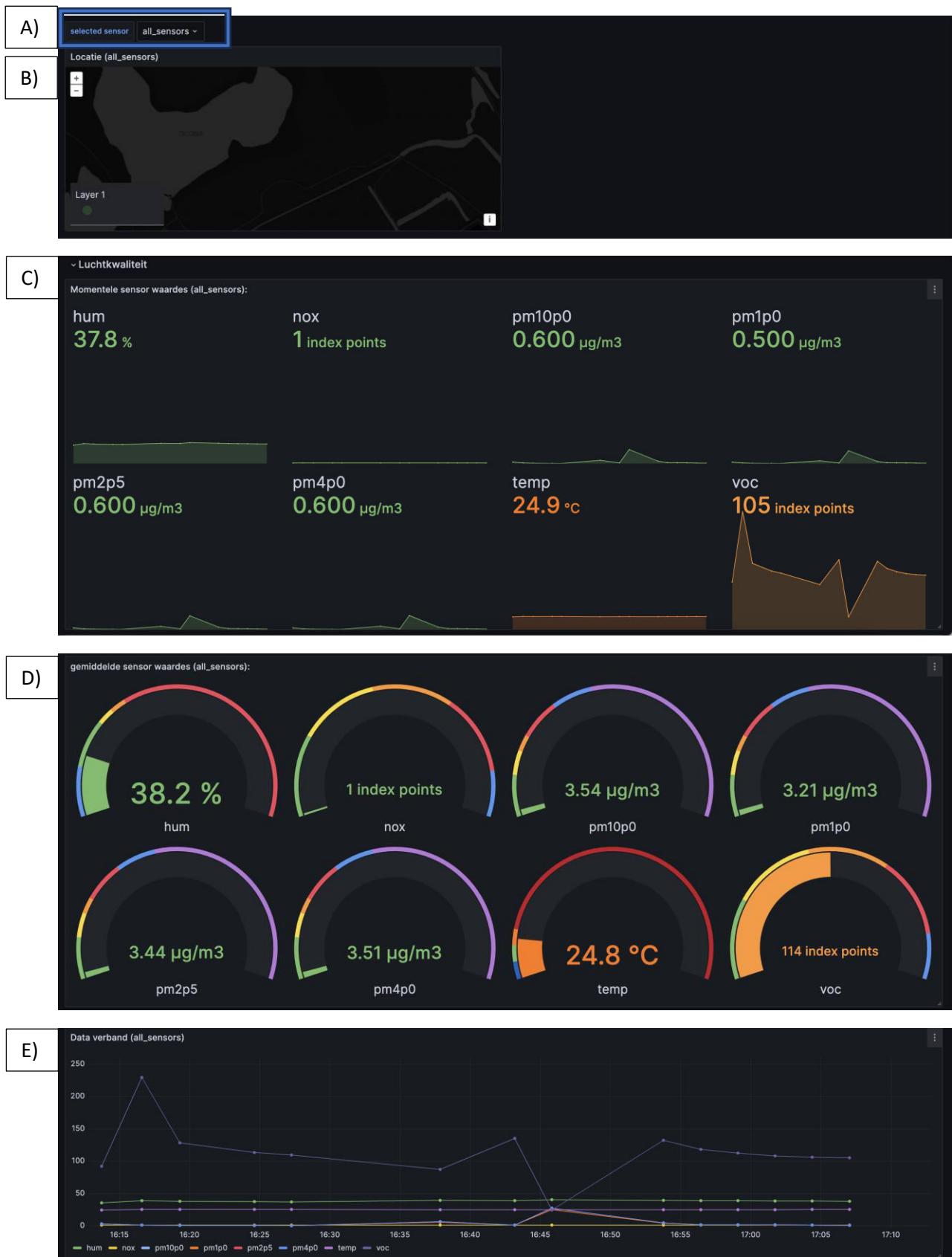
query:

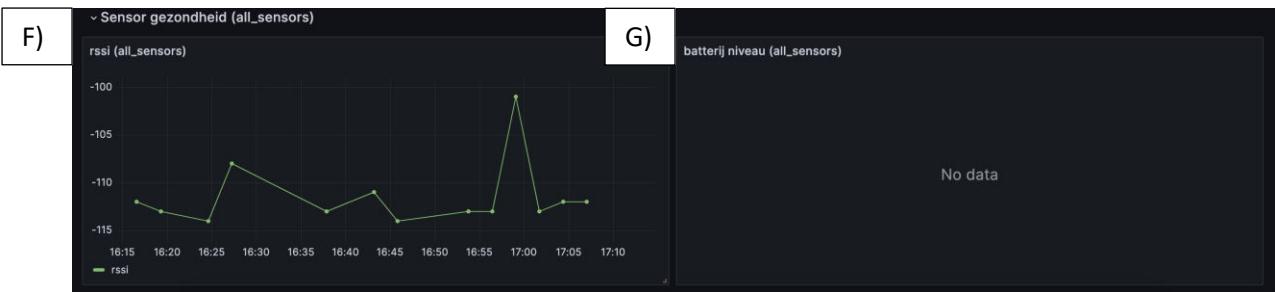
```
import "influxdata/influxdb/schema"

schema.measurements(bucket:
"ESP32aqs")
|> keep(columns: ["_value"])
|> group()
|> distinct(column: "_value")
```



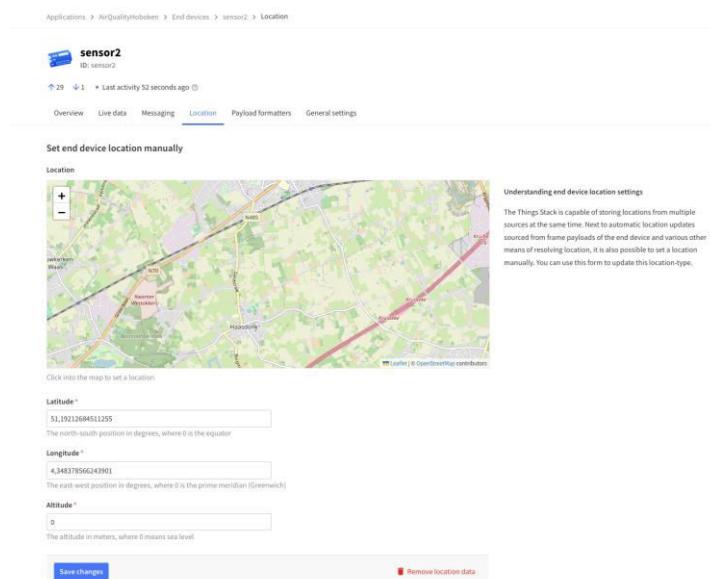
Dashboard





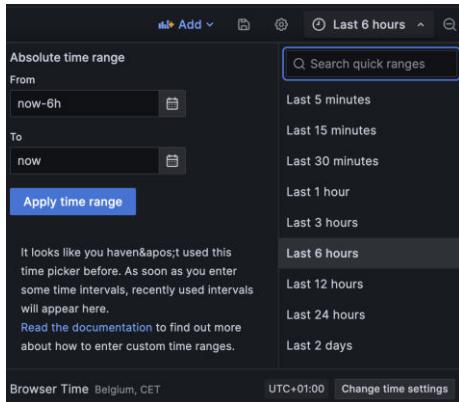
Dit is hoe het finale dashboard eruitziet:

- a) Linksboven zie je een drop down menu waaruit je kan kiezen welk sensor je wilt gaan bestuderen (dit zijn alle sensoren die zich in de database bevinden) ook kan je de optie “all_sensors” kiezen, het gehele dashboard zal aangepast worden aan de waarde van jouw keuze.
- b) Ook vind je van boven een kaart terug, deze geeft de locatie van de sensor weer (of in het geval van “all_sensor” keuze geeft deze alle locaties weer van alle sensoren). De locatie wordt gekozen afhankelijk van welke statisch locatie je hebt opgegeven in de TheThingsNetwork op de sensor. (Applications > AirQualityHoboken > End Device > (sensor) > location

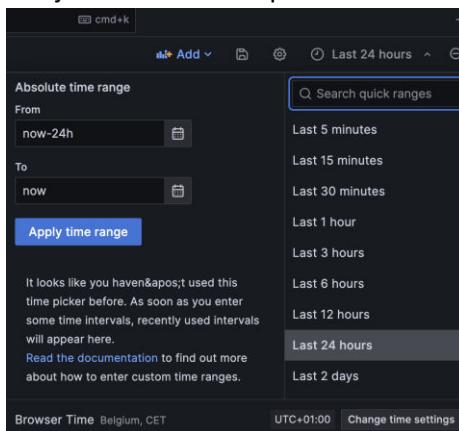


- c) Dit paneel met grafiek geeft de laatste actuele sensor waarde weer van elke sensor ook geeft deze een kleine history weer (deze history is afhankelijk van welke time range je hebt opgegeven).

De time range kan je helemaal boven aan aanpassen



- d) Dit paneel geeft de gemiddelde sensor waardes weer af alle data die zich in de timeline bevindt. Dit betekent dus als je gemiddelde tempratuur wilt weten van de afgelopen 24h stel je de timeline in op "last 24 hours".



- e) In dit paneel kan je de verbanden tussen verschillende sensor waardes weergeven. Ook hier is deze timeline afhankelijk
- f) Dit paneel geeft een grafiek weer over de rssi, met andere woorden hoe sterk de verbinding is tussen sensor kast en LoraWan-gateway. Hoe lager dit getal bij 0 zit hoe beter. Deze geeft dus geen informatie weer over de luchtkwaliteit. Ook hier is deze timeline afhankelijk.
- g) Dit paneel geeft de batterij capaciteit weer van de sensor kast. Maar wordt in de momentele versie van de sensor kast niet ondersteund.

Meetwaardes

Referentie NOx:

In de datasheet van de SEN55 sensor vinden we terug dat NOx wordt gemeten in “index points”. We kunnen waardes meten tussen 1 en 500.

2.4 Gas Specifications

Parameter	Comments	Values			Units
		Min.	Typ. ¹¹	Max.	
Output signals	VOC Index	1	–	500	VOC Index points
	NO _x Index	1	–	500	NO _x Index points

Bron: datasheet p.8

https://sensirion.com/media/documents/6791EFA0/62A1F68F/Sensirion_Datasheet_Environmental_Node_SEN5x.pdf

In de info notes kunnen we de betekenis van deze index terugvinden.

https://sensirion.com/media/documents/9F289B95/6294DFFC/Info_Note_NOx_Index.pdf

De referentie index waarde is 1. Een NOx waarde hoger dan 1 betekent dat er meer NOx gassen aanwezig zijn dan gemiddeld.

Een waarde dicht bij 1 betekent dat er (bijna) geen NOx gassen aanwezig zijn.

Referentie VOC:

De referentie index waarde is 100. Een VOC-waarde hoger dan 100 betekent dat er meer VOC aanwezig zijn dan gemiddeld.

Een waarde minder dan 100 betekent dat er minder VOC aanwezig zijn.

Info note:

https://sensirion.com/media/documents/02232963/6294E043/Info_Note_VOC_Index.pdf

Referentie Fijnstof:

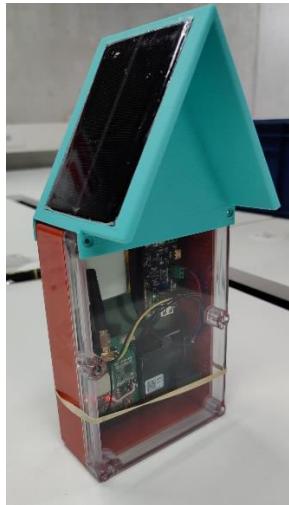
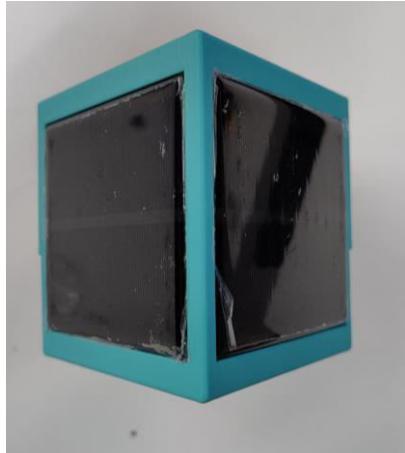
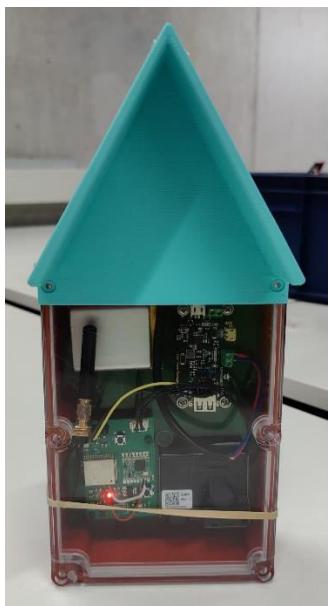
<https://www.vmm.be/data/fijn-stof>

- PM10: fijn stof met een diameter kleiner dan 10 µm (micrometer)
- PM2.5: fijn stof met een diameter kleiner dan 2,5 µm (micrometer)

Op de website van de Vlaamse milieumaatschappij vinden we terug dat volgens de Europese richtlijnen de jaargemiddelde PM2.5 concentratie niet hoger mag zijn dan 25 µg/m³.

Het jaargemiddelde PM10 concentratie mag niet hoger zijn dan 40 µg/m³.

Finale versie



Testresultaten



↑ 10:35:54	Forward uplink data message	DevAddr: 26 0B BF 31	<>		Payload: { hum: 41.79, nox: 1, pm10p0: 1.6, pm1p0: 1, pm2p5: 1.3,
↑ 10:35:54	Successfully processed dat...	DevAddr: 26 0B BF 31	<>		
↑ 10:18:53	Forward uplink data message	DevAddr: 26 0B BF 31	<>		Payload: { hum: 42.17, nox: 1, pm10p0: 2, pm1p0: 1.7, pm2p5: 1.9,
↑ 10:18:53	Successfully processed dat...	DevAddr: 26 0B BF 31	<>		
↑ 09:44:49	Forward uplink data message	DevAddr: 26 0B BF 31	<>		Payload: { hum: 44.53, nox: 1, pm10p0: 1.7, pm1p0: 1.6, pm2p5: 1.1,
↑ 09:44:49	Successfully processed dat...	DevAddr: 26 0B BF 31	<>		



Data Explorer

Table CUSTOMIZE Local SAVE AS

Filter tables... 

_start	_stop	_time	_value	_field	_measurement
2023-12-21 08:33:18	2023-12-21 09:45:00	2023-12-21 09:45:00	21,89	temp	sensor2
2023-12-21 08:33:18	2023-12-21 10:19:00	2023-12-21 10:19:00	23,03	temp	sensor2
2023-12-21 08:33:18	2023-12-21 10:36:00	2023-12-21 10:36:00	23,16	temp	sensor2

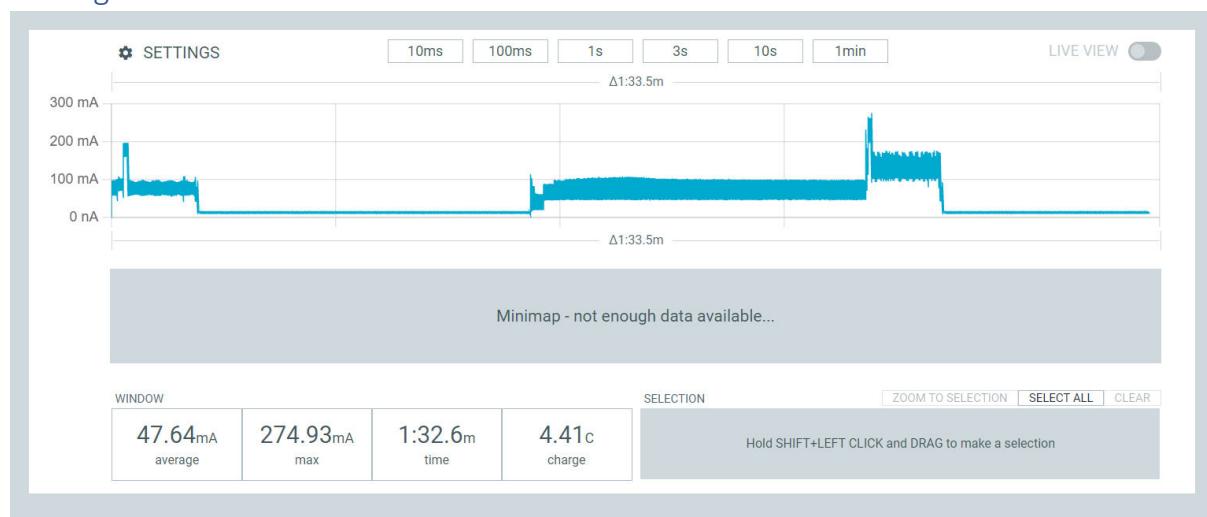


Energie verbruik

Verbruik componenten

COMPONENT	STATUS	TYPISCH VERBRUIK	MAX VERBRUIK
SEN55	Idle	2.6 mA	3 mA
	Measuring	70 mA	100 mA
SGP41	Idle	34 µA	105 µA
	Conditioning	4.2 mA	4.6 mA
ESP32	Regular	3 mA	3.4 mA
	Deep sleep	10 µA	/
SX1276	Active	40 mA	/
	Idle	1.5 µA	/
	Receive	12 mA	/
	Transmit	120 mA	/

Meting



Power budget berekening

Deze berekeningen zijn gemaakt met Sleep time gelijk aan 15 minuten en Idle time, waarin de sen55 opstart, 2 minuten. De avg current hebben we uit de metingen gehaald (zie hierboven).

De batterij heeft een capaciteit van 1250 mAh

State	Avg Current	Voltage	Time	% of total time	Current per measurement loop
Sleep	15 mA	3.3V	900s	87.6%	13.14 mA
Idle	105 mA	3.3V	120s	11.7%	12.285 mA
Transmit	171 mA	3.3V	7s	0.7%	1.197 mA

Average current gedurende de hele measurement loop: 26.622 mA

Uren van metingen op volle batterij (zonder opladen a.d.h.v. zonnepanelen meegerekend):

$$\frac{1250 \text{ mAh}}{26.622 \text{ mA}} = 46.95 \text{ h}$$

Mogelijke aanpassingen om nog stroom te besparen: Tijdens het zenden de sensor al uitschakelen voor een besparing tijdens de 'Transmit' state & de lengte van de deep sleep langer maken.

Bronnen

Algemeen project: <https://github.com/JokerIsMyBae/AirQualityHoboken>

SEN55 Doc:

https://www.sensirion.com/media/documents/6791EFA0/62A1F68F/Sensirion_Datasheet_Environmental_Node_SEN5x.pdf

SPG41 Doc:

https://sensirion.com/media/documents/5FE8673C/61E96F50/Sensirion_Gas_Sensors_Datasheet_SP41.pdf

Adafruit cp2102n UART to USB:

<https://cdn-learn.adafruit.com/downloads/pdf/adafruit-cp2102n-cp2104-friend-usb-to-serial-converter.pdf>

TheThingsNetwork (devices): <https://www.thethingsindustries.com/docs/devices/adding-devices/>

Node-Red: <https://nodered.org/>

InfluxDB: <https://www.influxdata.com/>

Aansluitschema SX1276: <https://www.thethingsnetwork.org/forum/t/programming-e>

MCCI Arduino LMIC library: <https://github.com/mcci-catena/arduino-lmic>

Sensirion Core library: <https://github.com/Sensirion/arduino-core>

Sensirion I2C Sen5x library: <https://github.com/Sensirion/arduino-i2c-sen5x>