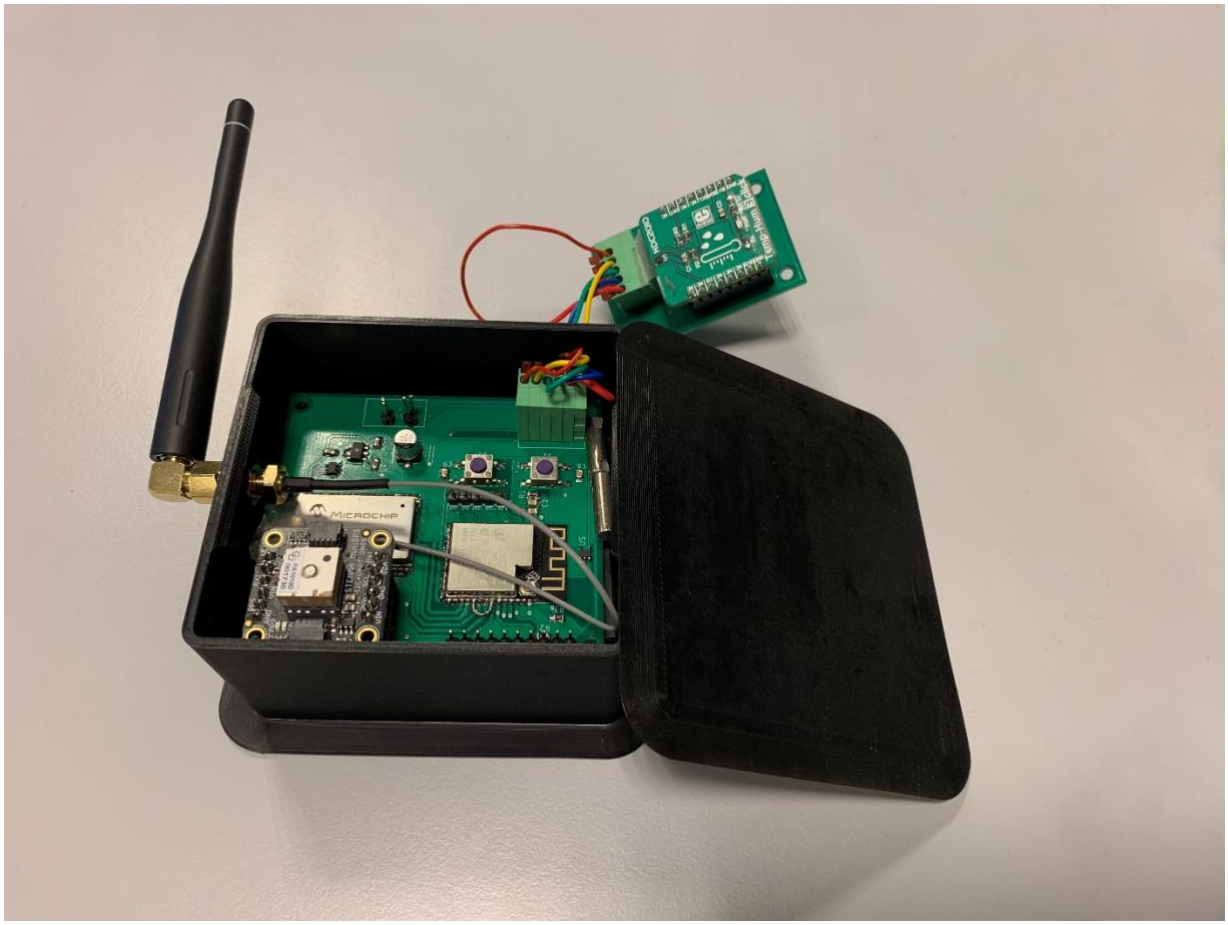
Labo IoT Nodes 2020-2021: The Cool Team



Basile Berckmoes

Niel Vandevelde

Manoëlle de Viron

18/05/2021

IIW KU Leuven

Technologiecampus Gent Elektronica-ICT

Inhoud

Inleiding

Voor het labo IoT Nodes moet er, in groep, een Embedded systeem ontworpen worden dat gebruik maakt van de concepten die geleerd werden tijdens de vakken “Sensors in Embedded Systems” en “IoT Nodes”.

Dit jaar is het thema “Logistiek”, en voor ons team is het de bedoeling om verschillende parameters van koel-containers te monitoren. Aan de ene kant moet de temperatuur gemeten worden, om een alarm te kunnen doorsturen wanneer de temperatuur binnen de container boven een bepaalde drempelwaarde stijgt. Bovendien moet het openen van de container gevolgd worden met het doorsturen van de lokalisatie van de container. Het doorsturen van data moet draadloos gebeuren, en het systeem wordt gevoed met een batterij. Het systeem moet over heel Europa werken.

Aanpak en structuur

Verwachtingen

- Temperatuur meten, alert doorsturen als het boven een bepaald drempel ligt
- Monitoren van het openen van de deur, locatie doorsturen als deur opent
- Draadloos doorsturen van data, moet vanuit heel Europa kunnen sturen
- Low-power aspect: werkt op een batterij, moet zo lang mogelijk meegaan

Specificaties van het systeem

Technisch

- Temperatuurmeting binnen een koel-container
- Alarm als temperatuur boven een bepaald drempel ligt
- Locatie kunnen bepalen
- Openen van de deur detecteren
- Moet onafhankelijk zijn op niveau van energie (werkt op (een) batterij(en))
- Draadloos versturen van data
    - Temperatuur als die boven drempel ligt
    - Locatie als deur geopend wordt

Niet-technisch

- Temperatuurwerking:
    - Binnen de container: -65°C tot 40 °C
    - Buiten de container: -15°C tot 40°C
- Grootte: moet op een container geplaatst worden (en niet storen): max 1dm^3
- Gewicht: maakt niet veel uit (<100g), moet aan de deur blijven plakken

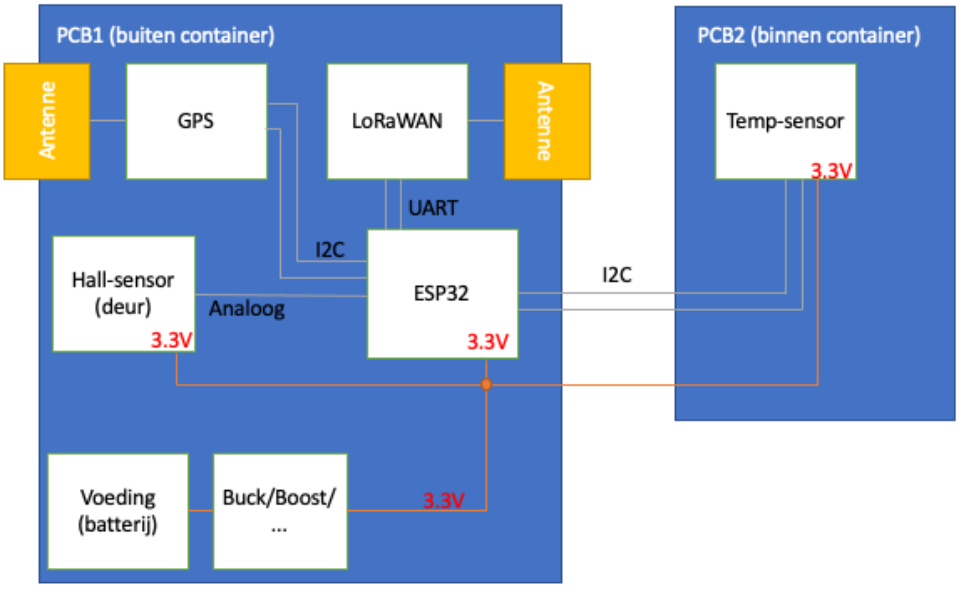
Niet-functioneel

- Tijd: 3 maand tot eerste prototype
- Kost: Geen budget gekregen (prijs componenten: zie excel op Git)

Legende

- IoT Node met behuizing (PCB met ESP32, GPS, LoRa + antenne en Hall)
- Binnenkant (gekoppeld met kabels): Temperatuursensor op PCB 2
- Magneet

Figuur 1: Plaatsing van het IoT node [bron afbeelding container: https://unithire.co.uk/wp-content/uploads/2019/02/Container-door.jpg]



Moeilijke aspecten:

- Vanuit binnenzijde van de container kan niet gestuurd worden
- Zo weinig mogelijk elektronische onderdelen binnen de container (elektronica gaat niet goed om met lage temperaturen)
- Oplading van een batterij gebeurt moeilijk

- Systeem wordt geplaatst op container, moet stevig vast zijn en behuizing moet ook stevig zijn om systeem te beschermen

Circuit

Tabel 1: Belangrijkste componenten van het circuit

	Microcontroller	LoRaWAN	GPS module	LDO	Hall sensor	Vochtigheid en temperatuur
Component	**ESP32**	**RN2483**	**Adafruit MiniGPS PA1010D**	**XC6231A332 PR-G**	**Si7217**	**HDC2010**
Rol	Beheer van heel circuit	Radio	GPS sensor	Voeding regulator	Deur detector	Temperatuur sensor
Voeding	3.0-3.6 V	2.1-3.6 V	3.0-4.3 V	2-10 V	2.25-5.5 V	1.62-3.6 V
Max stroom	20-31mA (Normal speed)	38.9 mA (TX, VDD=3.3 V)	36 mA (acquisition)	500 mA (max output stroom)	6.5 mA (conversion in progress)	730uA (temp meting)
Andere componenten	pushbuttons	TPS22860 (aan/uit zetten van het module)	TPS22860 (aan/uit zetten van het module)		magneet	

Keuze van de componenten en technologieën

1. Microcontroller: ESP32

Oorspronkelijk hadden we een idee om verschillende bordjes met temperatuursensoren te hebben die met elkaar via Bluetooth communiceren binnen de container. Dan hebben we de keuze gemaakt voor ESP32 (WiFi en Bluetooth/BLE beschikbaar). Uiteindelijk bleek het niet nodig om verschillende temperatuursensoren te hebben, en werd er geen gebruik gemaakt van Bluetooth. Toch hebben we de ESP32 behouden. We zouden alhoewel een andere MCU kunnen gebruiken die de job ook zou doen (bvb met een ARM processor).

Op de ESP32 zijn er UART en I2C bussen beschikbaar, en genoeg GPIOs om het volledige systeem te besturen. Twee pushbuttons werden op de PCB geplaatst om de ESP32 te kunnen programmeren.

Op de PCB zijn er ook twee headers geplaatst: één voor de programmer, en de andere header zorgt voor testpoints.

2. LoRaWAN: RN2483

Voor de draadloze connectiviteit was LoRa voor ons een duidelijke keuze: het is low-power, long-range en werkt ongeveer over heel Europa (volgens LoRa-Alliance zijn alleen Denemarken, Montenegro en Kosovo niet bediend in het Europees continent).

LoRa wordt alleen gebruikt als de temperatuur hoger wordt dan de drempelwaarde, en moet dus alleen gevoed worden als data gestuurd moet worden. Dit wordt geregeld met de TPS22860, die wordt aangestuurd via de microcontroller. Communicatie tussen het LoRa module en de ESP32 gebeurt via **UART.**

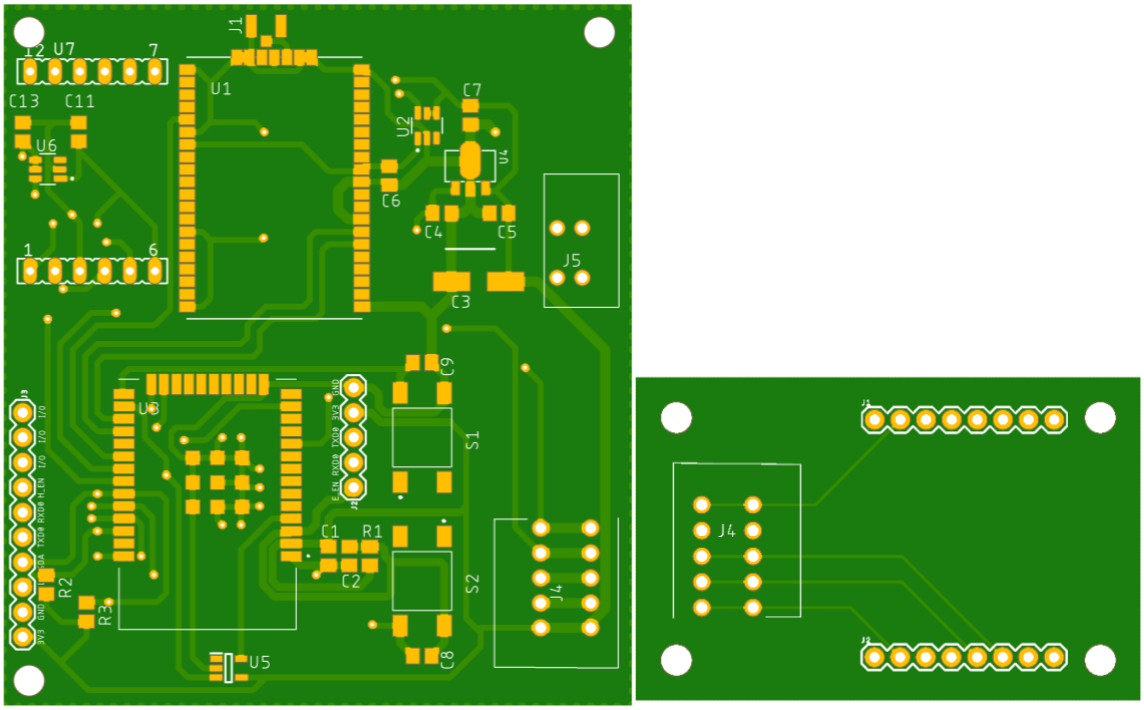
De LoRa module stuurt ook een Interrupt pin aan van de ESP32 (GPIO16).

3. Power: XC6231A332PR-G

Aangezien het niet zo handig zou zijn om ons IoT node te heropladen, hebben we ervoor gekozen om AA batterijen te gebruiken (2 in serie en 2 in parallel, wat ons een spanning van 3 V geeft, en een capaciteit van 6000 mAh). Hierna gebruiken we een LDO om een spanning van 3,3 V te hebben voor het volledige circuit, aangezien al onze componenten met deze spanning kunnen gevoed worden. Deze kan een VIN hebben tussen 2 en 10 V.

PCBs

- We hebben twee PCBs moeten maken voor ons circuit. De hoofd PCB wordt buiten de container geplaatst, in een behuizing, en aan de deur van de container vastgemaakt. Hierop zijn de meeste componenten geplaatst. De tweede PCB werd gemaakt voor de temperatuursensor, die dus een voeding moet krijgen alsook de I2C bus.

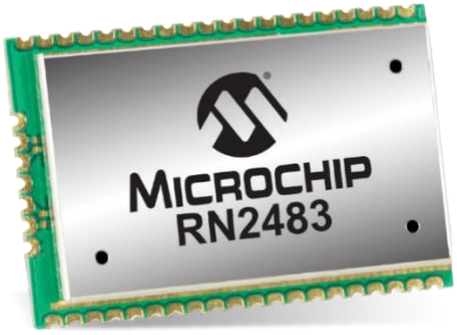


LoRa Data Transmit

RN2483

De RN2483 low-power, Long Range LoRaTechnology Transceiver-module van Microchip biedt een gebruiksvriendelijke, energiezuinige oplossing voor draadloze datatransmissie over lange afstand. Met een werkspanning van 2.1V-3.6V kon deze in het project gebracht worden zonder nood te hebben aan nieuwe spanningsniveaus. We hebben gekozen om gebruik te maken van de sub GHz (868MHz) frequentie vanwege bron [1]. De module heeft 15 km dekking in voorsteden en tot 5 km dekking in stedelijk gebied. De ingangsimpedantie van de RFH pin is 50Ohm die we dan met een U.FL connector verbinden. Om connectiviteit te verbeteren wordt de antenne buiten de behuizing geplaatst zodat indien nodig zelf een metalen behuizing gebruikt kan worden. Met een U.FL naar SMA kabel wordt de

antenne aansluiting op de behuizing bevestigd. De antenne is dan ook een 50 Ohm ¼ golflengte dipoolantenne met een frequentie range van 862MHz – 876MHz en SMA connector als aansluiting.



Figuur 2: RN2483 module

De communicatie met de module gebeurt via UART op de 2de seriële poort van de ESP32.

Specification	Description
Baud Rate	57600 bps
Packet Length	8 bit
Parity Bit	No
Stop Bits	1 bit
Hardware Flow Control	No

Figuur 3: instellingen UART

De instellingen van de module gebeuren over UART aan de hand van de datasheet RN2483 *LoRaTM TECHNOLOGYMODULE COMMAND REFERENCEUSER'S GUIDE.* Volgende instellingen worden over de UART gedaan om de module te kunnen laten communiceren.

- **mac reset {868}** : Stelt de standaardwaarden in en selecteert de 868 standaardband.
- **mac get deveui {deveui}**: Kijken of de module opgestart is en de communicatie werkt
- **mac set appeui {appeui}**: Deze opdracht stelt de applicatie-ID voor de module in.
- **mac set appkey {appkey}**: Dit commando stelt de applicatiesleutel voor de module in.
- **mac set pwridx {5}**: Dit commando stelt het uitgangsvermogen in dat bij de volgende transmissies moet worden gebruikt.
- **mac set ar {off}**: Dit commando stelt de status van het automatische antwoord in. Door het automatische antwoord in te schakelen, zal de module een pakket zonder payload verzenden onmiddellijk nadat een bevestigde downlink is ontvangen. Indien ingesteld op UIT, wordt er geen automatisch antwoord verzonden.
- **mac set adr {off}**: Deze opdracht stelt in of de adaptieve gegevenssnelheid (ADR) moet worden ingeschakeld of uitgeschakeld. De server wordt geïnformeerd over de status van de ADR van de module in elk uplink-frame dat het ontvangt van het ADR-veld in het uplink-datapakket. Als ADR is ingeschakeld, optimaliseert de server de gegevenssnelheid en het zendvermogen van de module op basis van de informatie die via het netwerk wordt verzameld.
- **mac set dr {1}**: Deze opdracht stelt de gegevenssnelheid in die voor de volgende verzending moet worden gebruikt.
- **mac set adr {on}**

- **mac join {otaa}**: Dit commando informeert de RN2483-module dat het moet proberen om verbinding te maken met het geconfigureerde netwerk.
- **mac tx {uncnf } {1} {data}**: Deze opdracht maakt het mogelijk om data te versturen naar het netwerk.

Dit is de volgorde waarin de commando's naar de RN2483 om data door te sturen naar het netwerk. Omdat niet gespecifieerd werd in hoeverre de containers verplaatst zal worden hebben we geopteerd om OTAA te gebruiken over ABP. Hierdoor is het mogelijk om verschillende netwerken zonder problemen te betreden waardoor we grotere afstanden kunnen overbruggen.

Cayenne LPP

De Cayenne Low Power Payload (LPP) biedt een handige en gemakkelijke manier om gegevens over LPWAN-netwerken zoals LoRaWAN te verzenden. De Cayenne LPP zorgt voor een beperking van de grootte van de payload, die kan worden verlaagd tot 11 bytes, waardoor het apparaat meerdere sensorgegevens tegelijk kan verzenden. Bovendien stelt de Cayenne LPP het apparaat in staat om verschillende sensorgegevens in verschillende frames te verzenden. Om dat te doen, moeten elke sensorgegevens worden voorafgegaan door twee bytes [2]:

- **Datakanaal**: identificeert op unieke wijze elke sensor in het apparaat over frames heen, bijv. "Binnensensor"
- **Gegevenstype**: identificeert het gegevenstype in het frame, bijv. "temperatuur".

Type	IPSO	LPP	Hex	Data Size	Data Resolution per bit
Digital Input	3200	0	0	1	1
Digital Output	3201	1	1	1	1
Analog Input	3202	2	2	2	0.01 Signed
Analog Output	3203	3	3	2	0.01 Signed
Illuminance Sensor	3301	101	65	2	1 Lux Unsigned MSB
Presence Sensor	3302	102	66	1	1
Temperature Sensor	3303	103	67	2	0.1 °C Signed MSB
Humidity Sensor	3304	104	68	1	0.5 % Unsigned
Accelerometer	3313	113	71	6	0.001 G Signed MSB per axis
Barometer	3315	115	73	2	0.1 hPa Unsigned MSB
Gyrometer	3334	134	86	6	0.01 °/s Signed MSB per axis
GPS Location	3336	136	88	9	Latitude : 0.0001 ° Signed MSB
					Longitude : 0.0001 ° Signed MSB
					Altitude : 0.01 meter Signed MSB

Figuur 4: verschillende gegevenstypes

In onze toepassing maakten we gebruik van de GPS locatie en temperatuur. Dit werd in de code geïmplementeerd in de methoden *void transmit_location(double latitude, double longitude, double altitude)* en *void transmit_temp(float temp)*. De payload ziet er dan uit als op figuur 4. Het datakanaal maakt het mogelijk om meerdere temperatuur sensoren te voorzien om mogelijkse uitbreidingen te doen.

1 Byte	1 Byte	N Bytes	1 Byte	1 Byte	M Bytes	...
Data1 Ch.	Data1 Type	Data1	Data2 Ch.	Data2 Type	Data2	...

Figuur 5: payload Cayenne LPP

The Things Network V3

De data die de RN2483 verstuurt, wordt dan ontvangen via TTNV3. De implementatie van V3 is nog bezig maar na verloop van tijd zal V2 wegvallen. Daarom hebben we ervoor gekozen om meteen voor V3 te gaan. Door een account te maken op The Things Network kan je ook een AppEUI, DevEUI en een AppKey genereren wat nodig is om toegang te krijgen tot het netwerk. De mogelijke integraties in TTNV3 zijn:

- MQTT
- Webhooks
- Storage Integration
- AWS IoT
- LoRa Cloud

MQTT

De gebruikte integratie in ons project is MQTT. Op een raspberry pi, computer of server kan de software draaien die nodig is om te subscriben op de MQTT uitzendingen van TTN. De code is geschreven in python. Om te kunnen subscriben heb je een API key nodig die te genereren is onder het vakje API keys bij TTN en de applicatie ID die je hebt gekozen. Je hebt deze nodig voor authenticatie via MQTT:

- Username =”{applicationID}@ttn”
- Password = “{API Key}

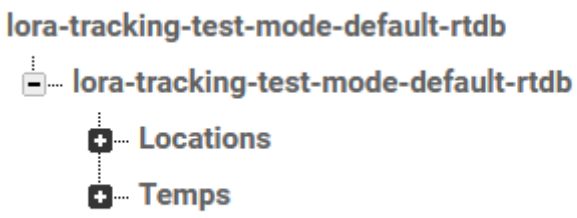
Omdat de V3 nog in implementatie zit kan je momenteel nog niet subscriben op een specifiek topic maar wel op de volledige data. Het topic ziet er als volgt uit:

“V3/{applicationID}@ttn/device/{deviceID}/up”

De data die je dan ontvangt is in json formaat. Na wat omvormingen te doen is het eenvoudig om de temperatuur en GPS waardes er uit te halen.

Firebase

Firebase is een platform ontwikkeld door Google voor het maken van mobiele en webapplicaties. Het was oorspronkelijk een onafhankelijk bedrijf opgericht in 2011. In 2014 verwierf Google het platform en het is nu hun platform voor app-ontwikkeling. Je kan je aanmelden bij Firebase met je gmail account. Na het aanmaken van een account kan je projecten toevoegen in de console. Wij maken gebruik van een realtime database. Dit is gratis zolang je onder de 10 000 queries per maand blijft.

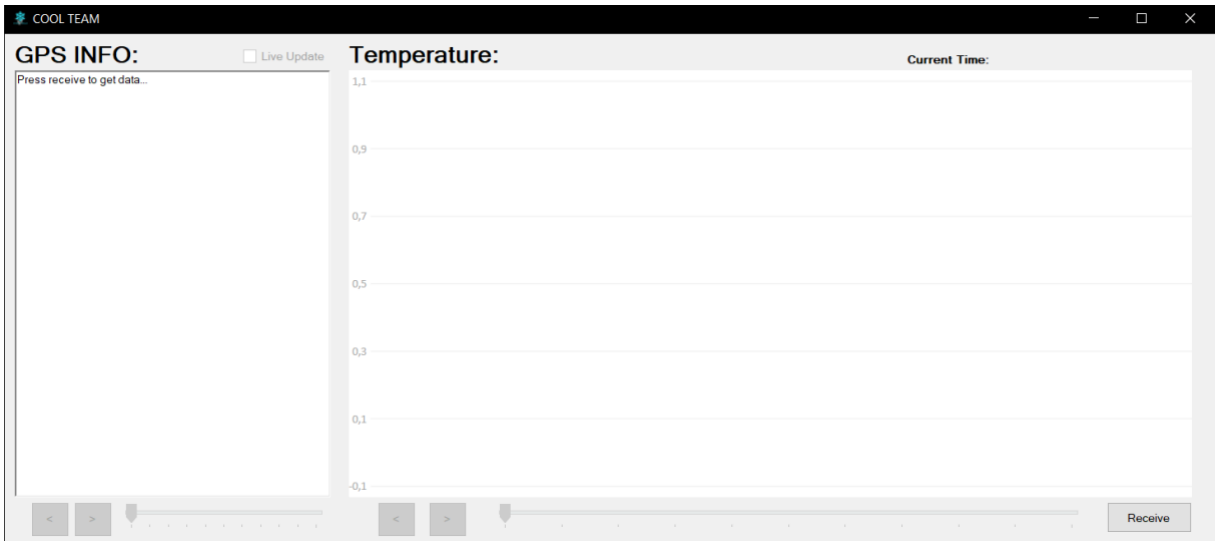


Figuur 6: database in de console

Het schrijven naar de database gebeurt in dezelfde python file die MQTT implementeert. Daarom moet deze file dan ook het best draaien op een server zodat er altijd gemonitord wordt of er data is om naar de database te sturen. De Firebase SDK ondersteunt programmeren in C ++, Java, JavaScript, JavaScript / Node.js, Objective-C, Swift en Python. Angular, Backbone, Ember en React worden ondersteund via koppelingen met de database. Google heeft een aantal helperbibliotheken toegevoegd: FirebaseUI, Geofire, Firebase Queue, FirebaseJobDispatcher. Hun naam geeft aan wat hun doel is. Firebase ondersteunt ook het importeren van grote JSON-gegevenssets en integratie met ElasticSearch.

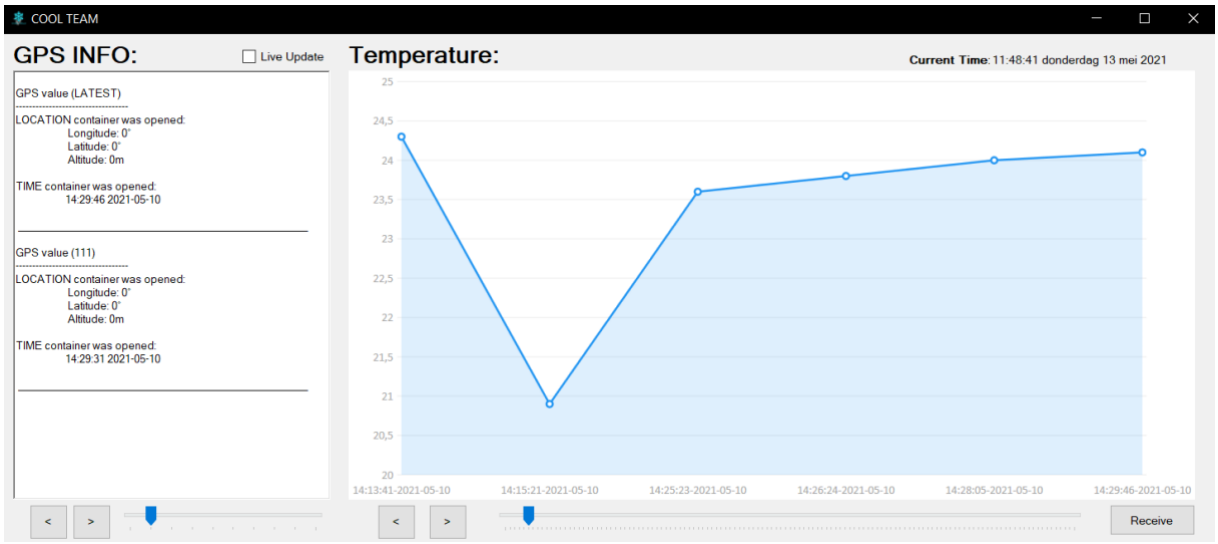
C#

Als laatste is er een GUI voorzien geschreven in C# om de data te kunnen visualiseren. De code vraagt dus de gegevens op die opgeslagen zijn in de realtime database van Firebase. Bij opstart ziet de GUI er als volgt uit:



Figuur 7: opstart

Na het klikken op *Receive* wordt de GUI geactiveerd.

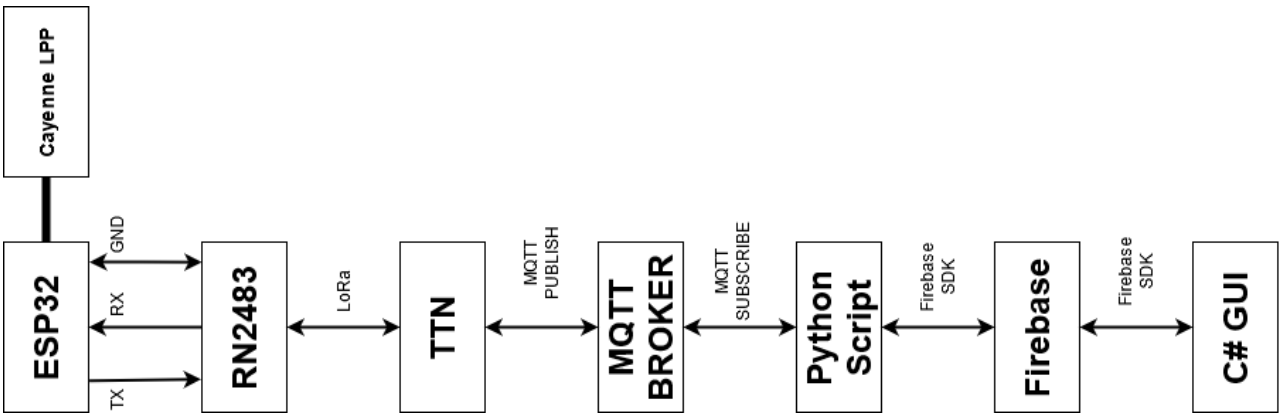


Figuur 8: GUI actief

Met de sliders onderaan kan men instellen hoeveel gegevens er tegelijk getoond moeten worden. Met het pijltje ernaast kan men dan door de data gaan om bepaalde momenten op te zoeken. Als de *Live Update* checkbox aangevinkt staat dan worden de andere knoppen gedisabled. Nu wordt iedere 5 seconden gekeken of er updates zijn van waardes om zo live data te kunnen volgen. Bij de live update mode is de visualisatie van de temperatuur beperkt tot 5 waardes en wordt enkel de laatste GPS waarde weergegeven.

Samenvatting data transmit deel

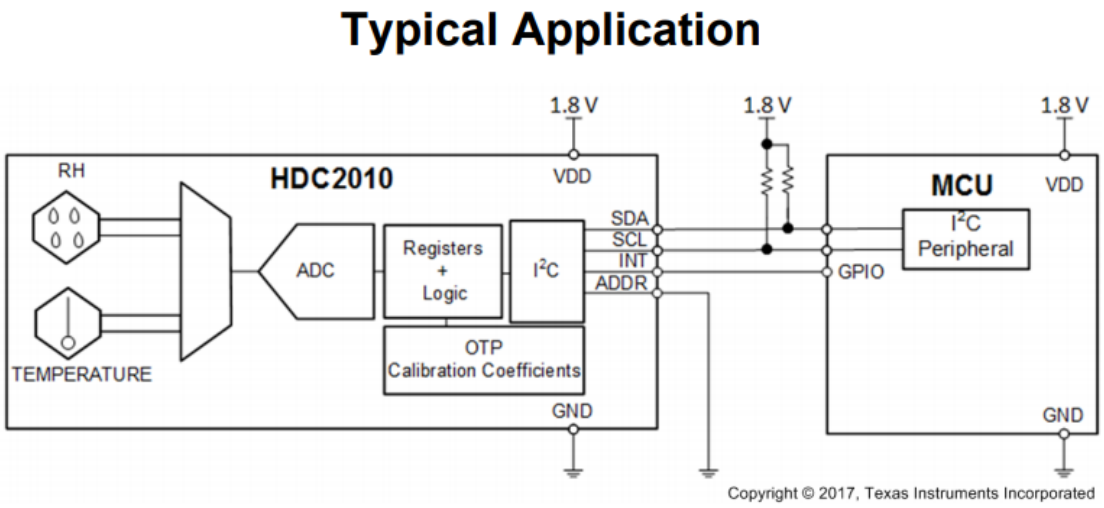
De temperatuur wordt verstuurd iedere 10 minuten en de GPS locatie bij het opengaan van de deur. Er is een IC voorzien die de spanning van de LoRa module (RN2483) volledig kan afleggen. Bij opstart moet de module opnieuw worden ingesteld via de commando's over UART. De data kan daarna verzonden worden door deze volgens de Cayenne LPP structuur te verpakken. Als The Things Network V3 data ontvangt van de node dan zet hij deze op zijn MQTT broker onder een bepaald topic. Een python script is voorzien die naar dit topic kijkt om data binnen te halen. De ontvangen data wordt opgeslagen in een database van Firebase. Met de windows applicatie geschreven in C# is het mogelijk om de data weer te geven.



Sensoren

Temperatuur en vochtigheid - HDC2010

De HDC2010 is een temperatuur- en vochtsensor met een slaap verbruik van 50 nA. Indien er een meting wordt uitgevoerd waarbij de relatieve vochtigheid en de temperatuur waardes worden gemeten met een 11 bit nauwkeurigheid dan verbruikt de sensor 550 nA. De sensor zelf maakt gebruik van capacitieve eigenschappen om de parameters op te meten. De sensor kan blijven functioneren tussen een temperatuur van -40°C en 85°C. Op Figuur 9 is een blokschema te zien hoe de sensor typisch wordt geïntegreerd.

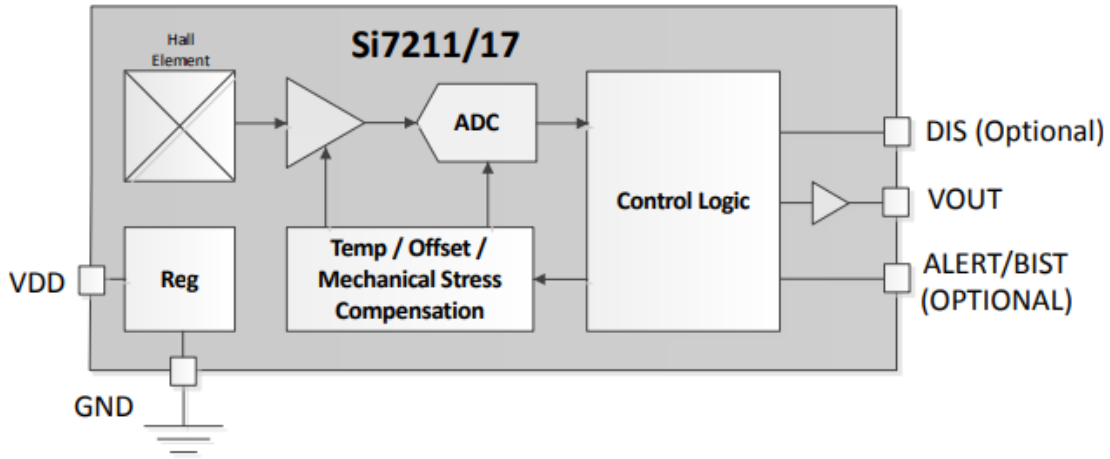


Figuur 9: typische hdc2010 toepassing [3]

Om de sensor in te stellen wordt de I2C bus gebruikt. In ons project hebben we de sensor ingesteld dat deze elke minuut een interruptie trigger geeft naar de ESP32 waarmee hij laat weten dat er een meting klaar staat om uitgelezen te worden. Verder kunnen ook interrupts gegenereerd worden als een drempel waarden overschreden word.

Si7217 Hall-effect Sensor

De Si7217 is een analoge sensor die een magnetisch veld kan detecteren. Tijdens het ontwikkelen van het project werd ingezien dat er beter geopteerd werd voor een digitale versie omdat deze dan ook als wake-up source kan gebruikt worden voor de ESP32. Het slaap-verbruik van de sensor is 50 nA en tijdens het meten kan dit oplopen tot 5.5 mA. De sensor heeft ook een DSP blok om temperatuur drift te voorkomen. Op Figuur 10 is het blokschema volledig te zien. De sensor kan blijven functioneren tussen een temperatuur van -40°C en +125°C.



Figuur 10: Blok schema Si7217 [4]

Adafruit Mini GPS PA1010D GPS module

De Adafruit Mini GPS is een zeer compacte GPS module gebaseerd op de PA1010D chip, die in een package van 2,5 op 2,5 cm een ingebouwde antenne heeft. Interfacen werkt met UART of I2C – zelf hebben we voor I2C gekozen, om één I2C bus te hebben voor de GPS en het temperatuursensor.

Ook al verbruikt de GPS module veel energie bij ontvangen van data (36 mA bij 3,3 V VDD), toch is het niet echt een probleem in onze applicatie. Het wordt alleen gebruikt als de deur van de container open gaat, wat normaal gezien heel uitzonderlijk gebeurt (alleen bij vertrek en toekomst). De Time-to-First-Fix (TTFF) is ook vrij laag (typisch 35 seconden, maximaal 60 seconden), wat genoeg tijd moed zijn voor dat de container verplaatst wordt (veel sneller dan het op- of ontladen van een container). Als de GPS niet wordt gebruikt (en als de locatie bepaald en gestuurd werd), wordt de voeding van de module volledig uitgeschakeld – aan de hand van de TPS22860, die via de ESP32 wordt bestuurd – om het verbruik van het circuit zo laag mogelijk te houden.

De GPS communicatie werkt met de NMEA 0183 [5] standaard (gedefinieerd door de National Marine Electronics Association), die de specificaties bepaalt voor GPS ontvangers en dergelijke. De communicatie gebeurt onder de vorm van zinnen die meestal met een dollarteken ($) en een header beginnen. De header bepaalt van waar de informaties ontvangen worden, en het type van bericht dat wordt ontvangen. In dit geval kan er bijvoorbeeld gekozen worden tussen verschillende Global Navigation Satellite Systems (GNSS): GPS (USA), GNSS(Rusland) en Galileo (EU) [6], maar bij default is het in GPS+GLONASS mode (welke we behouden hebben). Elk systeem bestaat uit een aantal satellieten, en heeft dus een eigen nauwkeurigheid. De verschillende “output sentences” die ontvangen kunnen worden met de GPS module houden bepaald informatie in (zie Tabel 2).

Tabel 2: Each of the NMEA output sentences

Option	Description
GGA	Time, position and fix type data.
GSA	GNSS receiver operating mode, active satellites used in the position solution and DOP values.
GSV	The number of GNSS satellites in view satellite ID numbers, elevation, azimuth, and SNR values.
RMC	Time, date, position, course and speed data. Recommended Minimum Navigation Information.
VTG	Course and speed information relative to the ground.

Voor ons prototype wordt er gebruik gemaakt van RMC en GGA data (om boven de longitude en de latitude ook de altitude te krijgen – andere parameters die we ontvangen houden in datum, tijd en snelheid). Uiteindelijk zullen we hiervan allen maar de latitude, longitude en altitude gebruiken. Meer informatie hierover kan gevonden worden in de datasheet van het GPS module “CD_PA1010D_Datasheet_v.03.pdf” in de Git (folder Datasheets).

Voor de nodige parameters die we nodig hebben is de gekregen data is onder de vorm degree decimal minutes (DDM), die omgevormd kunnen worden naar degree minute second (DMS: DD° MM’ SS.S”) of decimal degrees (DD: DD.DDDDDD, DDD.DDDDDD) formaat aan de hand van converters (vb: https://www.pgc.umn.edu/apps/convert/) of manuele berekening:

- Latitude: ddmm.mmmm => omgevormd naar dd mm.mmmm (kan zo in Maps ingediend worden)
- N/S indicator: N

- Longitude: dddmm.mmmm => omgevormd naar ddd mm.mmmm
- E/W indicator: E
- MSL Altitude: x.x m (altitude boven/onder gemiddelde zeeniveau (Mean-Sea-Level)

Ook al werkte de code bij onafhankelijk testen van de GPS module, toch zijn we er niet geraakt om die in ons uiteindelijke prototype te doen werken. Het blijkt dat er byte per byte werd gelezen, in plaats van per “NMEA sentence”, waardoor de code nooit een “fix” kon vinden, en dus de locatie kon uitlezen en doorsturen, hoewel het ledje van de GPS blinkte en het module wel degelijk een fix had gevonden. Dat hebben we dus niet kunnen regelen tijdens het laatste labo. Het feit dat het ook noodzakelijk was om naar buiten te gaan om deze code te testen, maakte het ook niet gemakkelijk om het uit te proberen (zeker omdat het weer niet speciaal mee was).

Een ander aspect dat problemen zou kunnen geven (als de code gefixt werd), zou het feit dat het module vertikaal wordt geplaatst, waardoor de antenne horizontaal ligt, hoewel het vertikaal zou moeten staan om een goede GPS signaal te kunnen ontvangen.

Software

De software werd stapsgewijs opgebouwd. Eerst werden de functionaliteit van de sensoren voorzien samen met de slaap strategie. Er wordt gebruik gemaakt van 2 slaap modes die hieronder in meer detail worden toegelicht. Als het systeem voor de eerste keer wordt opgestart zal een éénmalige setup uitgevoerd worden. Een variabele die gealloceerd staat in het RTC geheugen wordt gebruikt om bij te houden als het systeem uit een power-down komt of uit een deep sleep cycle. Als het systeem wakker wordt van deep sleep cycle zal een wake-up routine uitgevoerd worden. Meestal zal hierbij enkel gekeken worden als de deur open staat door de hall-sensor uit te lezen en zal de temperatuur opgemeten worden. Na 10x zal de temperatuur ook doorgestuurd worden via LoRa. De code kon nog verbeterd worden door pas de temperatuur te verzenden over LoRa indien er een bepaalde afwijking is tussen de huidige meting en de vorige meting.

Deep Sleep

In deep sleep worden de CPU’s, grootste deel van het RAM en alle digitale randapparatuur uitgeschakeld. De enige delen die actief blijven zijn de ULP coprocessor en de RTC systemen. Dit betekend dus dat na het wakker worden uit deep sleep de CPU niet meer weet welke code voor het laatste uitgevoerd wordt. Data die niet verloren mag gaan na deze slaap modus wordt opgeslagen in het RTC geheugen. Op de GPIO pin waar de interrupt pin van de HDC2010 is aangesloten word ingesteld als wake-up source.

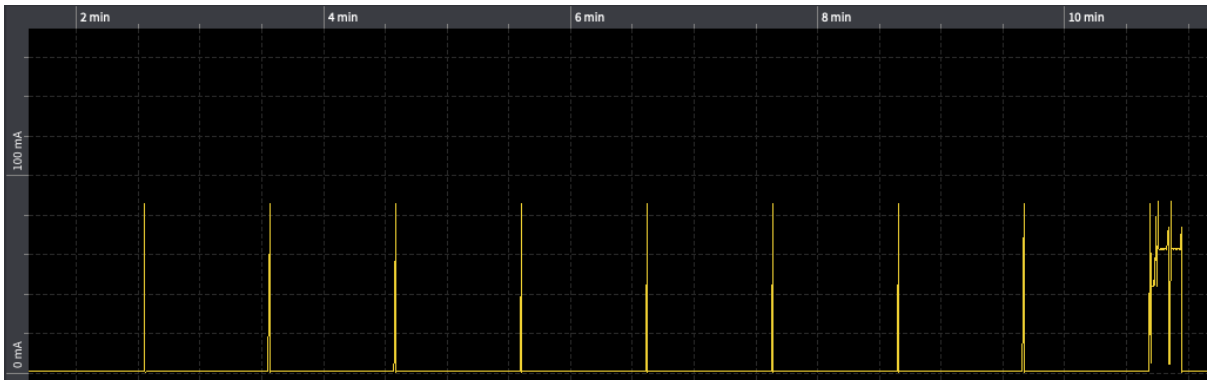
Light Sleep

In het programma werd oorspronkelijk veel gebruik gemaakt van delay functies als er gewacht moest worden op een ander deel hardware. Dit is geoptimaliseerd door gebruik te maken van light sleep. Tijdens deze mode worden de CPUs, het meeste RAM en digitale randapparatuur ge clock-gated. Ook wordt de voedingsspanning verlaagd. Na het wakker worden uit deze mode gaat het programma gewoon door. Hiermee is een delay functie gemaakt met als parameter het aantal milliseconden de ESP32 moet slapen. Ook werd geëxperimenteerd om in light sleep te gaan als er gewacht wordt op een antwoord van de LoRa module. Hierbij werd de UART dus als wake-up source ingesteld. Maar dit werkte niet omdat het eerste karakter op de bus niet geregistreerd werd.

Power

De PCB werd voorzien van een batterij-houder waar 4 AA batterijen in kunnen geplaatst worden. Elke cel heeft een spanning van 1,5 V en een capaciteit van 3000 mAh. Indien er 2 cellen in serie en 2 cellen parallel geschakeld worden, bekomen we een totale capaciteit van 6000 mAh en 3 V. Met deze gegevens hebben we een LDO gekozen die de batterijspanning zal omvormen naar een stabiele 3,3 V. De uiteindelijk gekozen LDO is uit de XC6231 series van torex [7]. De regulator heeft een slaap verbruik van 35 µA en heeft een werktemperatuur van -40°C tot 85°C.

Door gebruik te maken van de OTII stroom meter kon een inschatting gemaakt worden hoelang de uitgewerkte node kan meegaan tot de batterijen leeg zijn. Op Figuur 11 is te zien dat een volledige cycle ongeveer 10 minuten duurt. Tijdens de korte pieken wordt de esp32 even wakker en gaat hij na indien hij iets moet versturen of een GPS locatie moet doorsturen. De GPS module was tijdens de laatste labo zitting nog niet operationeel dus deze is niet opgenomen in deze meting. De laatste brede is het verbruik indien de LoRa module actief is. Via een ingebouwde tool van het OTII softwarepakket kon een schatting gemaakt worden hoelang het toestel zou meegaan en dit resulteerde in 3,4 maand.



Figuur 11: Stroomverbruik

Design behuizing

De behuizing is ontworpen met behulp van Fusion 360. Als eerste is het Eagle board gexporteerd naar Fusion zodat hier rond een behuizing kan worden getekend. Op Figuur 12 is het design te zien. Echter na het printen werd duidelijk dat er enkele foutjes in het ontwerp zaten. Ten eerste was het antenne gat op een verkeerde hoogte voorzien. En er was ook een fout gemaakt in het ontwerp van het deksel waardoor de 2 delen niet kunnen vast klikken in elkaar.

Figuur 12: 3D design

Testen

De werking van alle componenten met eigen software werd eerst afzonderlijk getest met de ESP32.

Hierna werd de synthese code getest. Debugging stuk na stuk gedaan. Buiten testen (voor de gps) is niet echt handig want het weer is niet altijd mee.

Wat kan beter

Een GPS module zonder breakout board zou minder componenten hebben en geen ledjes, het verbruik zou dus verminderd worden. En GPS met externe antenne zou ook precieser zijn.

Een openbare lid aan de onderkant van de behuizing toevoegen, om de batterijen te kunnen vervangen zonder alles te openen.

Besluit

De gerealiseerde node bevat grotendeel alle gevraagde specificaties. Het enige systeem dat niet 100% functioneren is, is het GPS deel. De gekozen module was niet robuust genoeg en kon bij veel omstandigheden geen connectie maken met de satellieten. Verder is er ook plaats voor optimalisatie wanneer de node de temperatuur over het LoRa netwerk stuurt. Voorlopig gebeurt dit periodiek waardoor er veel energie verloren gaat. Een beter oplossing, eigenlijk zoals opgegeven in de specs, zou de module enkel moeten zenden indien een grenswaarde werd overschreden. Dit is echter via software makkelijk aan te passen maar door de korte project-looptijd was dit over het hoofd gezien. Tenslotte was het een robuuster idee om de ESP32 in deep-sleep te zetten waarbij een interne klok als wake-up source gebruikt wordt. Nu is de super low power temperatuursensor de wake-up source maar dit bleek uiteindelijk een minder veelzijdige oplossing te zijn. Zo zal bijvoorbeeld de node volledig falen indien de sensor offline is door defect. Hierdoor kan er dus geen alarm signaal doorgestuurd worden.

Bijlagen:

GitHub: https://github.com/manoelledeviron/CoolTeam

BOM: GitHub/CoolTeam/Documenten/BOM PCB.xlsx

Datasheets: GitHub/CoolTeam/Datasheets

Software: GitHub/CoolTeam/Software/Basile/CoolTeam indu prototype

Hardware design: GitHub/CoolTeam/Schema/SchemaPCB1.pdf,
GitHub/CoolTeam/Schema/Sensorbordje/pcb2.pdf

Verwijzingen

[1] I. Bobkov, A. Rolich, M. Denisova en L. Voskov, „Study of LoRa Performance at 433 MHz and 868 MHz Bands Inside a Multistory Building," April 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9067427/authors#authors.

[2] Cayenne Docs , „Cayenne Low Power Payload," [Online]. Available: https://developers.mydevices.com/cayenne/docs/lora/#lora-cayenne-low-power-payload. [Geopend april 2021].

[3] T. Instruments, „HDC2010 Low-Power Humidity and Temperature Digital Sensors," GitHub/CoolTeam/Datasheets/hdc2010.pdf, 2017.

[4] S. Labs, „Si721x Field Output Hall Effect Magnetic," GitHub/CoolTeam/Datasheets/si721x-data-sheet-hall.pdf, 2018.

[5] E. S. Raymond, „NMEA revealed," 04 2021. [Online]. Available: https://gpsd.gitlab.io/gpsd/NMEA.html. [Geopend 18 05 2021].

[6] R. Timbrook, „Expert World Travel," 12 05 2021. [Online]. Available: expertworldtravel.com/gps-vs-glonass-vs-galileo. [Geopend 18 05 2021].

[7] Torex, „XC6231Series 10V Input, 500mA, High Speed LDO Regulators," GitHub/CoolTeam/Datasheets/XC6231-846901.pdf.