Rescue Device

Verslag

Embedded Devices

Ben Janssen – 2IOT
Lian Aarts – 2IOT
Niels Cuyvers – 2IOT
Pieter Janssen – 2IOT

Academiejaar 2020-2021

Campus Geel, Kleinhoefstraat 4, BE-2440 Geel

LID VAN ASSOCIATIE KU LEUVEN

THOMAS MORE

INHOUD

Voorwoord

Voor het behalen van een credit voor het vak embedded devices in het tweede jaar IOT, is het maken van een project een vereiste. Deze bundel bevat de schriftelijke documentatie.

De voorbije weken paste we alle theoretische kennis toe in de praktijk, door middel van deze opdracht. Het was een zeer uitdagende, maar leerrijke ervaring. Bij het uitwerken kregen we hulp van onze docent. Hierbij danken wij Quinten Desmyter voor de interessante lessen die we nodig hadden om ons project te kunnen realiseren.

Inleiding

Tijdens de eerste zes weken van embedded devices leerde we veel bij over microcontrollers en een nieuwe programmeertaal C++. Alle leerstof die we hebben behandeld tijdens de lessen moeten we samenbrengen in een groepsopdracht. We hebben voor project drie gekozen. Dit project bevat zowel het ontwerpen van een embedded systeem, maar ook het verwezenlijken van een fysiek product dat van een hoogte kan vallen. Dit brengt veel problemen met zich mee, hier willen we graag een oplossing voor zoeken.

Hieronder een beschrijving van het project:

Het gebruik van drones in zoek- en reddingsoperaties verminderd de zoektijd enorm. De tijd nodig die nodig is om iemand te bereiken is echter niet verbeterd. Om dit probleem op te lossen maken we een toestel dat we van een drone laten vallen. Dit toestel bevat alle benodigdheden die nodig zijn zoals: eerste hulp, communicatie en lokalisatie. Dit kan de patiënt gebruiken in afwachting van de zoekploeg.

We hebben eerst een toepassingsgebied bepaald. De keuze ging uit naar het gebruik op zee of in water. Hierdoor is het meegeven van EHBO-materiaal overbodig, omdat dit te veel krachten kan kosten aan de patiënt. Hierna hebben we bepaald welke sensoren en actuatoren noodzakelijk zijn. Ten derde, rest ons het probleem dat we op een locatie zitten zonder ontvangst. Wat is hier de oplossing voor? Verder moet het product veilig aan de grond geraken, na het loskoppelen van de drone. Als laatste is het belangrijk dat de patiënt het systeem makkelijk kan gebruiken.

Het doel is om een zo compact mogelijk systeem te maken dat toch heel veel hulp kan bieden. Ook moet het betrouwbaar en gebruiksvriendelijk zijn, zodat iedereen ermee kan werken.

MoSCoW

Must have (this)	• GPS-lokalisatie • Licht stroboscope (redder kan dit aan/uit zetten) • Draadloze verbinding • Waterdicht • Batterij • Dropmechanisme (magnetisch?) • IR-strobe • 3D geprinte container (high visibility) PLA/PetG • Printplaat • Foam (floatation) • Fluitje
Should have (this if at all possible)	• Visualisatie naar locatie (magnetometer) voor zoekers • Wireless charging • Chocolade reep
Could have (this if it does not affect anything else)	• Hartslag sensor (ECG?) • Beeldscherm (OLED) • Drijfmechanisme doos (opblaasbaar) • Reddingsvest • Road flare
Won't have (Would like to have but won't have this time around)	• Eten • Drinken • Wereldwijde communicatie

Tabel 1: MoSCoW

Materiaal

1.1 Hartslag sensor – MAX30100

Er zijn veel technieken die men kan gebruiken voor het meten van een hartslag. Elke techniek heeft zijn voor- en nadelen. Er zijn twee verschillende technieken:

Meting via elektrodes:

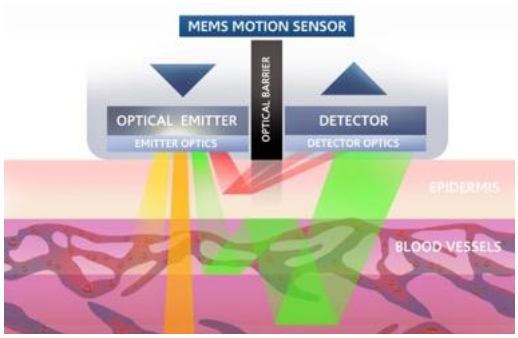
Elektrocardiografie is een manier van meten die men in de gezondheidszorg gebruikt. Men bevestigt meerdere elektrodes op de huid. Deze elektrodes zullen de kleine elektrische signalen van het hart waarnemen. Meerdere meetpunten zijn nodig voor een juiste meting. De locatie van deze elektrodes is ook van groot belang. Grote spiergroepen zoals het hart zullen een variërend elektrisch signaal creëren en dus de meting beïnvloeden. Het plaatsen van deze elektrodes kan dus enkel gebeuren door iemand met de juiste kennis. Het plaatsen van deze elektrodes in een vochtige omgeving kan ook voor problemen zorgen.

Optische meting:

Omdat optische metingen vaak aan de pols gebeurt zijn deze metingen vaak onbetrouwbaar. Maar het toenemend gebruik van deze sensoren in smartwatches en fitness trackers heeft de betrouwbaarheid doen stijgen. Het gebruik van een optische sensor is ook beter in situaties van stress en trauma. Het plaatsen van deze sensor kan zonder enige medische kennis.

Figuur 1: Hartslagsensor

De optische sensor schijnt light op de huid en meet de hoeveelheid van het weerkaatste light. Dit principe steunt op het gedrag van licht dat verandert afhankelijk van het vloeien van bloed.



Figuur 2: Werking hartslagsensor

1.2 Batterij – LiPo 104050

De Lithium batterij is een batterij die veel voorkomt in het dagelijkse leven in allerlei elektrische toepassingen zoals: auto's, gsm's, fototoestellen, ….

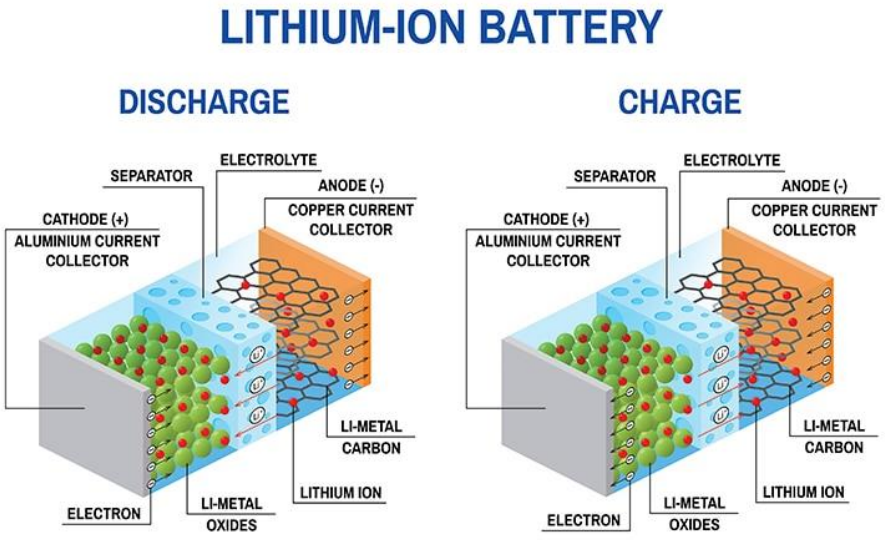
Figuur 3: Lithium-ion batterij

Een LiPo-accu mag men nooit te ver ontladen. Dit kan lijden tot ontsteking of explosie van de batterij. Daarom is een Battery Monitoring System (BMS) essentieel voor het veilig gebruik van deze batterij. In moderne LiPo batterijen is een BMS vaak ingebouwd. Het opladen moet ook altijd op de juiste manier gebeuren

De vermogensdichtheid van een LiPo batterij is aanzienlijk groter (100-265Wh/kg) dan die van een Nikkel metaal hydyde (NiMH) of een Nikkel Cadnium (NiCd) batterij. De ontlaad snelheid van een LiPo batterij is ook hoger, dit wil zeggen dat de batterij veel meer energie kan leveren in een korte tijd.

De bronspanning van 3,7V is ook bijna drie keer zo hoog als die van NiMH- en NiCd-accu's.

Grafiet gevuld met lithium ionen vormt de anode. Een lithium metaaloxide vormt de kathode. Een vloeibare elektrolyt laat de beweging van ionen toe. Wanneer de batterij ontlaadt, vloeien de ionen van anode naar de kathode.



Figuur 4: Uitleg werking Lithium-ion batterij

1.3 Draadloos opladen

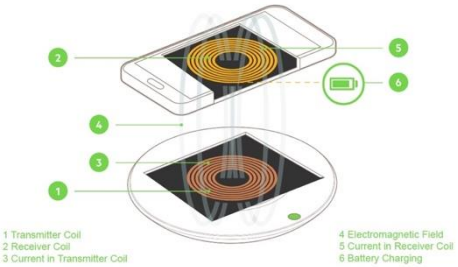
Draadloos opladen werkt d.m.v. een magnetisch veld, dit is anders dan het traditionele opladen met behulp van een kabel.

Figuur 5: Draadloos opladen

De meest voorkomende draadloze oplaadmethode is inductief laden. Dit werkt door middel van twee spoelen een elektrische stroom kunnen verplaatsen met behulp van elektromagnetische inductie. Inductief laden kunnen we vergelijken met de werking van een transformator.

Om elektriciteit draadloos te verplaatsen zijn er dus twee spoelen nodig, een spoel die elektriciteit verzendt en een spoel die ontvangt. De spoel die verzendt, zoekt contact met de ontvangstspoel. Wanneer de spoelen elkaar vinden ontstaat er een magnetisch veld.

Wanneer er een wisselende elektrische stroom aanwezig is in een geleider, zal er een veranderend magnetisch veld ontstaan rond deze geleider. Wanneer er vervolgens een geleider aanwezig is in dit een veranderend magnetisch veld zal er in die geleider een stroom ontstaan. Zo zal een verplaatsing van elektrische stroom gebeuren



Figuur 6: Onderdelen draadloos opladen

<u>Qi standaard</u>

Sinds 2009 is er een bepaalde standaard ontwikkeld voor het draadloos opladen. Dit is de Qi (spreekt men uit als: chi) standaard. Dit houdt in dat er een energieoverdracht van met laagvermogen plaats vindt, tot maximum 5 watt met een maximale afstand van vier cm.

De nieuwe manier van draadloos opladen is ontwikkeld door het Wireless Power Consortium. WPC is een groep van meer dan 200 grote fabrikanten.

<u>Voor- en nadelen</u>

- Er is geen draad meer nodig is. Dit zorgt ervoor dat we geen poort moeten voorzien voor het opladen van de module. Hierdoor kunnen we alles waterdicht maken.
- Het opladen is minder efficiënt is, er is een groot energieverlies tijdens het opladen.

1.4 LoRa – TTGO LoRa32 OLED v2.1.6

Figuur 7: TTGO LoRa32 v2.1.6

De LoRa32 is een microcontroller gemaakt door LilyGO. Het bevat een esp32 bord, een SSD1306 OLED scherm en een LoRa chip. De minicontroller is het communicatiemiddel alsook de besturing van alle hardware bij de patiënt. Zo kan de module van de patiënt zijn gps-locatie versturen naar de module van de zoekers. Het is ook mogelijk om een hartslag te meten en dit door te sturen of te tonen op het scherm dat aanwezig is op de TTGO LoRa32.

1.4.1 Inleiding

LoRa staat voor ”long range”. LoRa is een communicatieprotocol dat gebruik maakt van de vrije sub-gigahertz frequenties. In Europa is de frequentie van het signaal 868 MHz. Via deze weg is het mogelijk om dat aan 0.3 tot 5.5 kbit/s te versturen.

LoRa is in 2015 gelanceerd door Cycleo die later overgenomen zijn door Semtech. LoRa Alliance is de non-profit organisatie die de open standaard van LoRa verder ontwikkeld en promoot. Proximus en The Things Network maken in België gebruik van LoRa.

1.4.2 Waarom LoRa

Technologie	Draadloze Communicatie	Afstand	Tx Verbruik
Bluetooth	Korte afstand	10 m	2.5 mW
Wifi	Korte afstand	50 m	80 mW
3G/4G	Cellulair	5 km	5000 mW
LoRa	LPWAN	• 2-5 km (stedelijk) • 5-15 km (landelijk) • > 15 km (zichtlijn)	20 mW

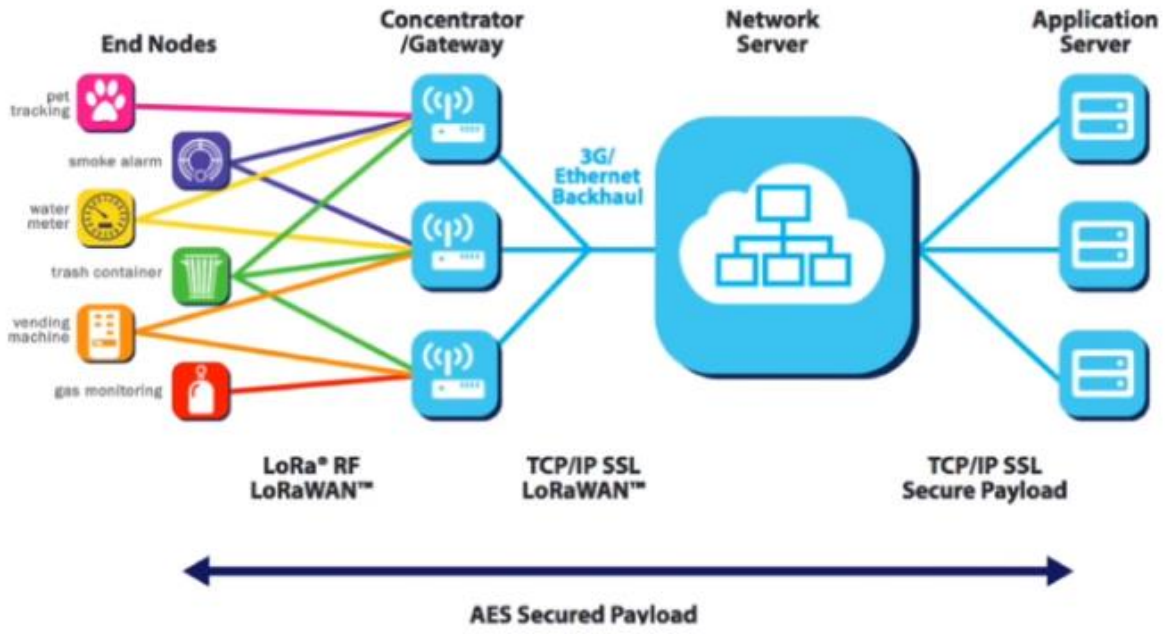
Tabel 2: Vergelijking afstand tegenover verbruik

LoRa heeft de beste karakteristieke in verband met afstand en verbruik, dit gaat ten koste van transfer speed. Dit is de reden dat de communicatietechnologie LPWAN uitgerust is om op een groot gebied kleine datapakketten te versturen. Vandaar de naam Low Power Wide Area Network.

1.4.3 Het LoRa Netwerk (TTN)

Een LoRa end-node maakt het mogelijk om gelezen data van sensoren op te sturen naar het LoRa netwerk met behulp van Gateways.

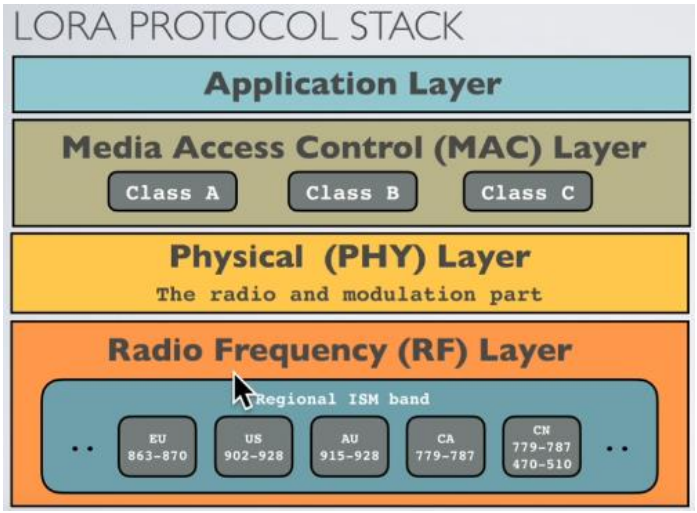
Een Gateway is een key component in het LoRa netwerk omdat het de communicatie dat hij binnenkrijgt van de end-nodes op het internet kan plaatsen om zo verder te kunnen gebruiken in applicaties of databanken. Ook kan de gateway gebruikt worden om communicatie tussen end-nodes te verrijken. In deze opdracht wordt **geen** gebruik gemaakt van het LoRa netwerk enkel point to point communicatie.



Figuur 8: Voorbeeld van een LoRa netwerk

1.4.4 LoRa TCP/IP model

LoRa heeft zijn eigen TCP/IP model om protocollen op te volgen tussen apparaten. Zo hebt je de Applicatie laag, de Media Access Control laag, de Fysieke laag en de Radio Frequentie laag.



Figuur 9: LoRa protocol stack

1.4.5 Regels

Binnen LoRa zijn er regels die behouden moeten worden. LoRa werkt buiten de ISM (Industrial, Scientific and Medical) radio banden die wereldwijd verkrijgbaar zijn. Deze banden worden getoond in volgende tabel.

Werelddeel	Frequentiegebied (MHz)
Azië	433
Australië	915 – 928
Europa, Rusland, India, Afrika	863 – 870
Canada	779 – 787
China	779 – 787, 470 – 510
Verenigde Staten van Amerika	902 – 928

Tabel 3: Frequentie gebieden

In België worden volgende frequenties 863 – 870 en 433 gebruikt.

Iedereen kan gebruik maken van de banden die buiten het ISM vallen zonder kosten. Dit betekent echter dat er meer storing aanwezig is op deze banden. Dit kan een negatieve effecten hebben op de datasnelheid en betrouwbaarheid van het signaal.

In Europa zijn er nog extra regels waar men zich moet aan houden.

- Voor uplink is de maximum transmissie kracht gelimiteerd tot 25mW (14dBm).
- Voor downlink is de maximum transmissie kracht gelimiteerd tot 0.5W (27dBM).
- Er is een 0.1% en 1.0% duty cycle per dag afhankelijk van het kanaal.
- De maximum toegelaten antenne behalen +2.15dBi.

<u>Duty cycle per dag:</u>

Wanneer een signaal is verzonden van de zender heeft het een tijd nodig voordat de ontvanger het signaal ontvangt. Dit noemt men Time on Air (ToA)

Voor 1.0% duty cycle betekent het dat bijvoorbeeld een ToA van 670ms,

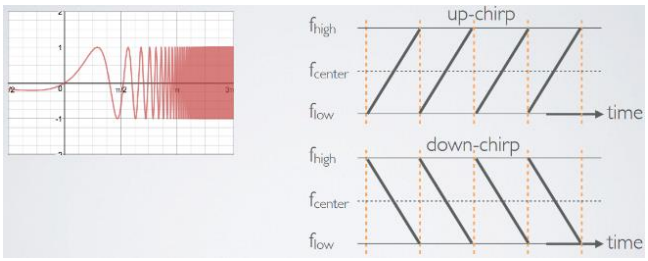
$$99 * 670 = 66330 = \frac{66330}{1000} = 66{,}33\ seconden$$

Er moet 66,33 seconden worden gewacht vooraleer het volgende bericht verzonden mag worden.

1.4.6 Modulatie typen en Chirp Spread Spectrum

LoRa is een gepatenteerde verspreidingsspectrum modulatie techniek dat is gebaseerd op Chirp Spread Spectrum modulatie (CSS)

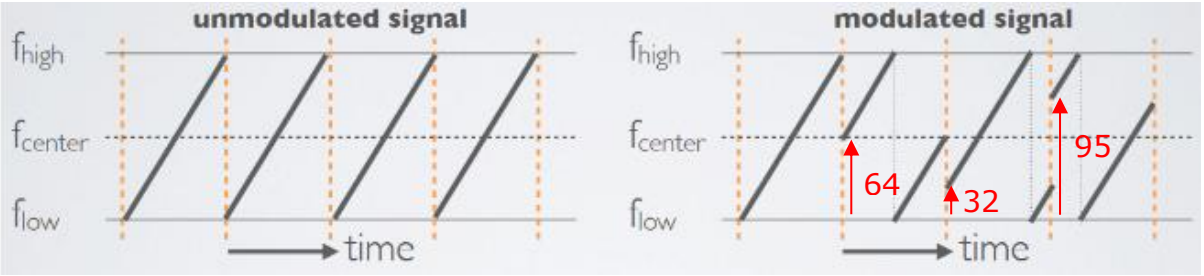
CCS is een techniek die men gebruikt om informatie te encoderen door gebruik te maken van chirp pulsen.



Figuur 10: Chirp pulsen

1.4.7 Symbool, Spreiding factor en Chip

Basis chirps zijn een stijgende lijn van de laagste frequentie tot de hoogste frequentie, genaamd up-chirp. Deze kan ook inverse zijn en van de hoogste frequentie tot de laagste frequentie gaan, genaamd down-chirp. Het aantal keer per seconden dat de modem de fases veranderd, wordt de chrip rate genoemd.



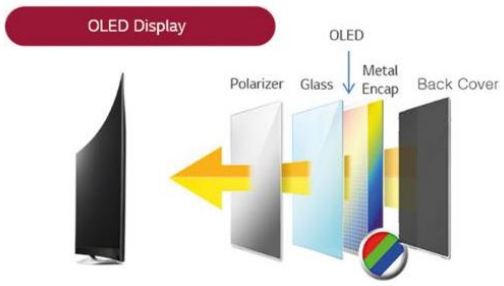
Figuur 11: Gemoduleerde en niet-gemoduleerde signalen

1.5 Display - OLED

De reddingswerkers over een module met een kleine display. Hierop kunnen ze de hartslag, de afstand en een pijl in de richting naar de patiënt zien.

De LoRa32 module is uitgerust met een OLED-scherm. Dit is een zeer efficiënt en kwalitatief scherm. De levensduur hiervan ligt hoger dan vele andere schermen ongeveer 80.000u wat bijna dubbel zo veel is als dat van een normaal lcd-scherm. Het heeft een afmeting van 0,96 inch (128 x 64 pixels) en maakt gebruik van het I²C protocol. Het display werkt op een spanning van 3 – 5,2 V en heeft een stroomverbruik van 15 mA.

Een OLED-paneel is opgebouwd uit pixels, elke pixel is een led die we individueel kunnen aangestuurd. Daardoor kunnen we een deel van het paneel uitschakelen wanneer het niet in gebruikt is. Een OLED-paneel is dus vele male efficiënter dan een lcd-display.



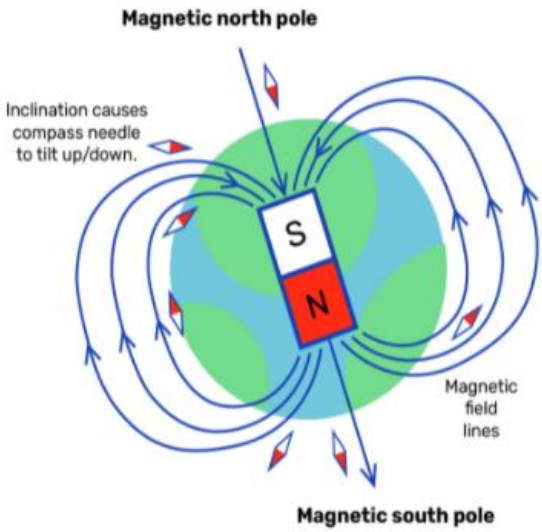
Figuur 12: Lagen OLED display

OLED kan worden vergeleken met een standaard led diode. Het P- en N materiaal zijn nu opgebouwd uit een organisch materiaal in plaats van halfgeleidermateriaal.

1.6 Magnetometer - GY-271

Op de module van de reddingswerkers komt een display met een pijl in de richting naar de patiënt. Door middel van de magnetometer weet de module van de reddingswerkers waar deze zich bevinden en waar het noorden is. De gps-locatie van de patiënt wordt doorgegeven aan deze module samen met de gps-locatie van de redders.

De magnetometer weet dan naar welke richting de reddingswerkers hun module houden. De magnetometer gaat het pijltje naar de richting van de patiënt laten wijzen.



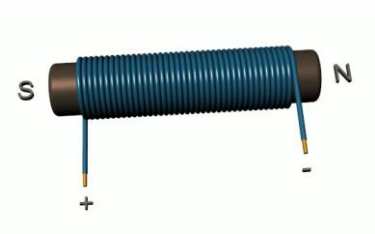
Figuur 13: Magnetisch veld aarde

De GY-271 magnetometer gebruikt magneto-resestive technologie. De weerstand van deze sensor zal veranderen afhankelijk van het magnetisch veld. Deze sensor is zeer precies een bestand tegen schokken en dergelijks.

1.7 Elektromagneet – CL-P20/15

Om het pakket aan de drone vast te maken en te vallen zijn er verschillende mogelijkheden. De mogelijkheden zijn: een servo, een touwtje dat wordt doorgebrand of nog iets anders zoals een elektromagneet. Deze magneet werkt wanneer er spanning op wordt gezet.

Omdat het toepassingsgebied water is, moet de elektromagneet waterresistent zijn. Hij werkt optimaal op 6 V. Dit voltage kan bereikt worden met behulp van een spanningsregelaar. Wanneer de magneet 6 V krijgt, kan het een gewicht van 2,5 kilo dragen. Dit is voldoende omdat het pakket maar een beperkt gewicht heeft.



Figuur 14: Elektromagneet

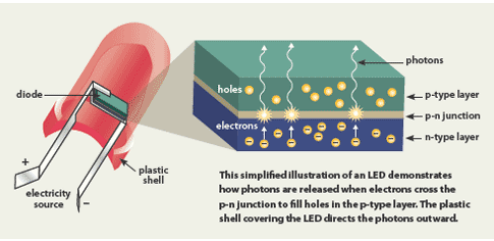
Een elektromagneet bestaat uit een kern met daarrond een spoel gewikkeld. Wanneer er een stroom door de spoel vloeit, zal de metalen kern de karakters krijgen van een magneet. Zo krijgt het een noord- en zuidpool en kan het andere metalen tot zich aantrekken. Op deze polen zal de magnetische werking ook het sterkst zijn. Wanneer er geen stroom door de spoel vloeit, zal de kern zicht gedragen als een stukje ijzer. Het zal dus geen andere voorwerpen tot zich kunnen aantrekken. De esp32 de naar de elektromagneet. Wanneer het pakket boven het droppunt komt, wordt deze logischerwijs uitgezet

1.8 Strobe licht, IR en zichtbaar

In de nacht kan het lastig zijn om de patiënt te vinden. De oplossing is om het pakket zo zichtbaar mogelijk te maken. Hiervoor is een LED licht zeer handig. In dit project wordt gebruik gemaakt van een 3 watt led die tot 700 mA stroom kan verbruiken.

Hoe werkt een LED?

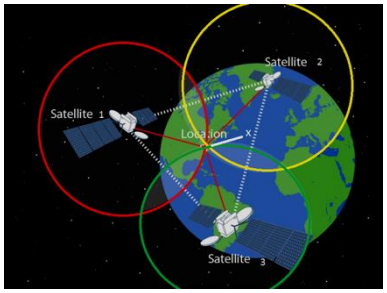
Een led of light emitting diode heeft de werking van een diode. Een led heeft een positieve kant (anode) en negatieve kant (kathode). Wanneer er stroom door de led vloeit, zal de led licht uitstralen. Elektronen vullen gaten in de halfgeleider waardoor er energie vrij komt. Deze energie vormt een foton. De kleur van de LED is afhankelijk van het materiaal dat men gebruikt in de led. Wit licht is altijd een samenvoeging van blauw en geel licht.



Figuur 15: Werking LED

1.9 GPS module - NEO-6M-0-001

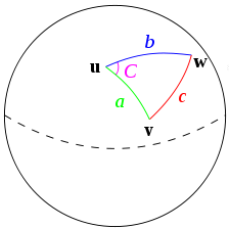
31 satellieten vormen het Global Positioning System (GPS). Elke satelliet gebruikt een atomische klok om de exacte tijd te bepalen. Deze tijd versturen de satellieten naar de ontvanger. De ontvanger zal de verstreken tijd gebruiken om de afstand naar de satelliet te bepalen. Er zijn drie satellieten nodig om een exacte locatie te kunnen bepalen.



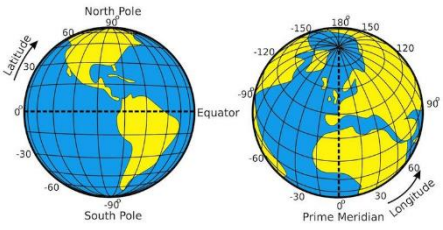
Figuur 16: gps-lokalisatie

De GPS-module die voor het project gebruiken is de NEO-6M-0-001, deze module is makkelijk aan te sturen. Via de LoRa32 is het mogelijk om de breedtegraden, lengtegraden en aantal verbonden satellieten te krijgen. Wanneer de breedte- en lengtegraad uitgelezen worden, kan een coördinaat bepalen.

Ook op de LoRa32 module van de zoekers zal een gps zitten. Doordat de zoekers en de patiënt beide een coördinaat hebben, kan de afstand bepalen tussen de modules. Hiervoor wordt de Haversine methode gebruikt.



Figuur 17: Haversine tekening



Figuur 18: Wereldbollen

Een betrouwbaar signaal krijgen is af en toe een uitdaging. Obstakels hebben veel invloed op de signaalsterkte. De module wordt gebruikt op open zee is dit echter geen probleem.

De module gebruikt een seriële verbinding.

1.10 3D print

De 3D print is geprint uit PolyLactic Acid (PLA). Dit is een zeer sterk bio plastic dat ideale eigenschappen heeft voor het 3D-printen. PLA heeft betere hygroscopische eigenschappen vergeleken Polyethyleentereftalaat Glycol (PETG). Dit betekent dat PLA minder water zal absorberen in vochtige omstandigheden.

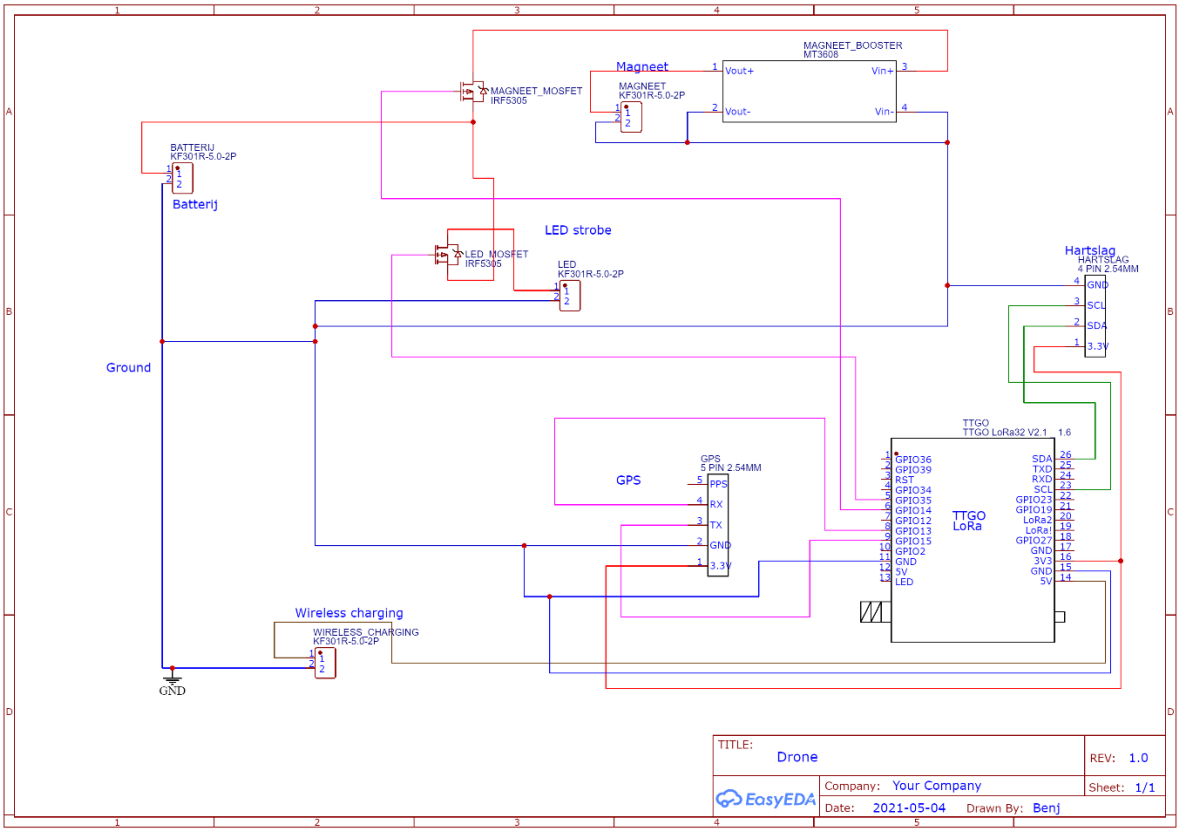
Omdat het toepassingsgebied de zee is, moet de case waterdicht zijn zodat de elektronica niet beschadigd kan worden. Verder zal het pakketje van de reddingswerkers ook een case krijgen, zodat ook zij hun module makkelijk kunnen meenemen.

De keuze ging naar Prusament PLA Prusa Orange. De orange kleur maakt de module goed zichtbaar. De kleine toleranties verbeteren de kans op een waterdichte behuizing. De externe parameters zijn hebben de grootste invloed op de waterdichtheid van de print. Deze hebben ook de meeste invloed op het gewicht van de behuizing. De juiste verhouding tussen waterdichtheid en gewicht zal het succes van de behuizing bepalen.

1.11 PCB

Een doelstelling van dit project is het maken van een eigen Printed Circuit Board (PCB) te maken. Een PCB zorgt ervoor dat al de componenten op de juiste manier verbonden zijn zonder gebruik te maken van losse draden. Dit maakt het eindproduct mooier en makkelijker in omgang. Doordat er geen draden kunnen loskomen zal de betrouwbaarheid ook beter zijn.

Voor het maken van de PCB werd EasyEDA gebruikt. Dit is een website die het makkelijk maakt om PCB te ontwerpen en later te bestellen. Op onderstaande foto vindt u een werkend schema waar een TTGO LoRa32 verbonden is met externe componenten via 2-polige klemmen. Deze componenten zitten rechtstreeks op het bord of worden later verbonden met male-to-female headers.

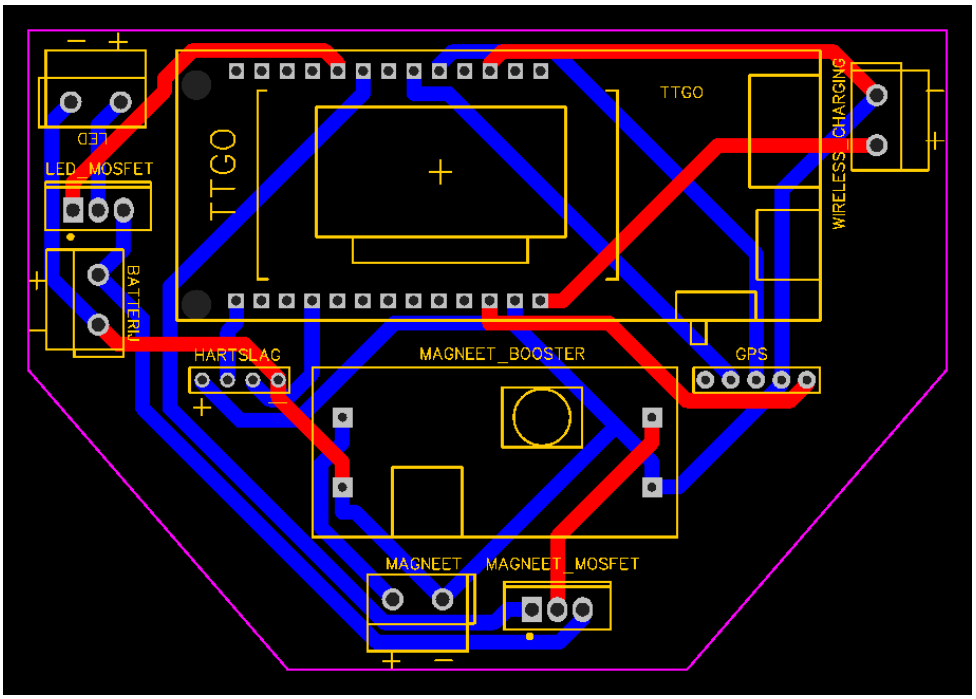


Figuur 19: Schema tekening

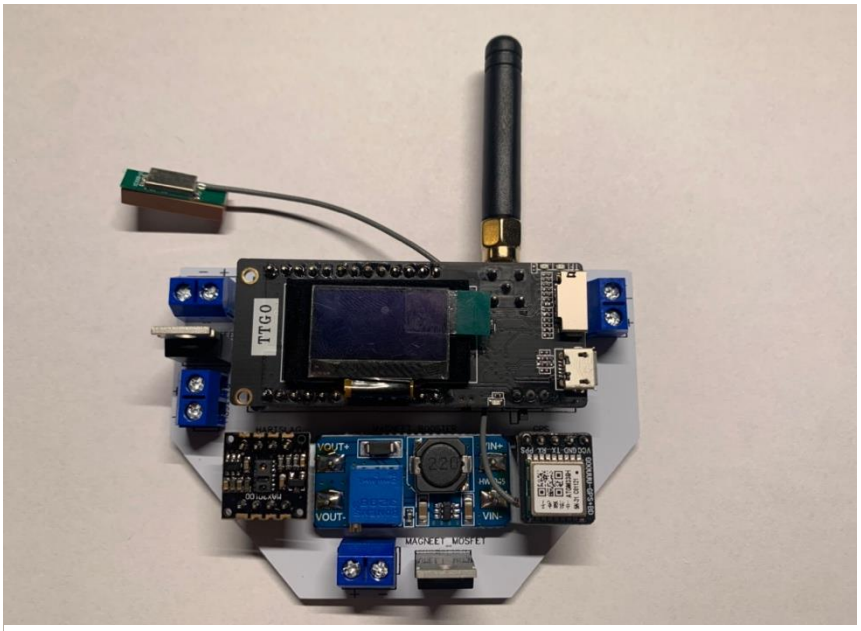
Na het maken van het schema moeten alle componenten op de PCB geplaatst worden. De positionering van elk component is belangrijk, er moet rekening worden gehouden met de fysieke eigenschappen van elk component. Bv. de micro-USB-aansluiting van de TTGO LoRa32.

Voor het berekenen van de baan dikte is gebruik gemaakt van de website Advanced Circuits. Deze hielp met het berekenen van de kabeldikte door middel van meegegeven parameters. De site gaf op dat de kabel 2mm dik moet zijn op de PCB om en maximum van 3A doorheen het circuit te krijgen.

Uiteindelijk is dit het resultaat.



Figuur 20: Pcb-tekening



Figuur 21: PCB met componenten

1.12 Chocoladereep

Voeding is een essentieel element van het pakket. De docent gaf ons het advies om voor een chocoladereep te gaan. Dit is voeding die niet snel vervalt, het bevat veel energie en weinig mensen zijn hier allergisch aan.

Een gemiddelde reep Melkchocolade bevat:

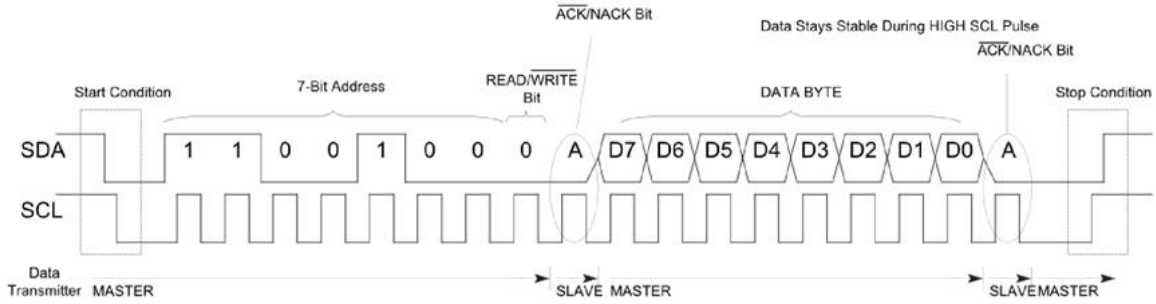
- 540 kcal of 2259 kJ
- 31,00g vet
- 19,00g verzadigd vet
- 54,50g koolhydraten
- 53,50g suiker
- 3,30g vezels
- 8,30g eiwitten
- 0,35g zout

Figuur 22: Chocoladereep

1.13 I²C

De I²C bus wordt gebruikt om data uit te wisselen. Het werd eerst ontwikkeld in 1979 door Philips de elektronicagigant.

De bus maakt gebruik van 2 buslijnen. Een seriële clock (SCL) en een seriële data (SDA). De communicatie vindt plaats tussen master en slave. Deze master zal het kloksignaal, start-bit en stop-bit genereren. Slaves communiceren wanneer de master hiervoor een verzoek verstuurt.



Figuur 23: Werking I²C

Dataoverdracht:

1. Al de data die die we verzenden, krijgen een startbit. Deze startbit zal een signaal geven naar de seriële data. De seriële datalijn zal laag gaan wanneer de seriële clock hoog is.
2. De master zal de juiste slave aanspreken door het 7 bit adres te versturen. De read/write bit bepaalt of de slave moet verzenden of ontvangen.
3. De slave met het adres zal reageren met een bevestiging.

Code

1.14 Zender

Functies:

- Gps-locatie bepalen
- Hartslag meten
- Magneet aansturen
- Strobe licht aansturen
- LoRa data verzenden

1.14.1 Libraries

De libraries worden toegevoegd om bepaalde sensoren en actoren in het project te laten werken. Er wordt FreeRTOS gebruikt om het project in te delen in verschillende processen over verschillende cores van de ESP32. Ook zal een GPS de exacte locatie te krijgen van de module. LoRa wordt gebruikt om communicatie tussen patiënt en zoekers mogelijk te maken. Een OLED-scherm dat zich bevindt op de ESP32 wordt gebruik om instructies te geven aan de patiënt en de hartslag wordt hiermee ook getoond.

```
#pragma region Includes
// FreeRTOS
#include <Arduino.h>

// GPS
#include <SoftwareSerial.h>
#include <TinyGPS++.h>

// LoRa
#include <SPI.h>
#include <LoRa.h>

// OLED scherm
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// Hartslagmeter
#include <MAX30100_PulseOximeter.h>

#pragma endregion Includes
```

1.14.2 Defines

De defines worden tijdens het compilen van de code vervangen met de meegegeven waarde van de define. Het gebruik van defines maakt de code overzichtelijker. Bv.: de pinnen voor het aansturen van het scherm.

```
#pragma region Defines
// FreeRTOS

// GPS

// LoRa
#define SCK 5
#define MISO 19
#define MOSI 27
#define SS 18
#define RST 23
#define DIO0 26
#define BAND 866E6 // voor Europa
//#define BAND 433E6 // voor Azië
//#define BAND 915E6 // voor Noord-Amerika

// OLED scherm
#define OLED_SDA 21
#define OLED_SCL 22
#define OLED_RST 16
#define SCREEN_WIDTH 128 // De breedte van het scherm.
#define SCREEN_HEIGHT 64 // De hoogte van het scherm.

#define PINLED 35
#define PINMAGNEET 14
#define PINHARTSLAG 19

// Hartslagmeter
#define REPORTING_PERIOD_MS 2000

#pragma endregion Defines
```

1.14.3 Globale variabelen

De globale variabele zijn nodig om variabele doorheen functies van het programma te gebruiken. De variabele ‘display’ wordt opgeslagen. Deze wordt later gebruikt tijdens de setup van bepaalde sensoren om gemakkelijk te laten zien wat hun status is, en om de coördinaten van de modulen te krijgen.

```
#pragma region GlobaleVariabele
// FreeRTOS

// GPS
String cor{""}; // Deze variabele is nodig om de lengtegraad en breedtegraad i
n een string op te slagen zodat deze kan verzonden worden via LoRa.
TinyGPSPlus gps; // Een variabele die nodig is om de locatie van deze module t
e weten.
```

```
SoftwareSerial ss(15, 13); // De Rx en Tx pinnen waarde de GPS module aanhangt
.

// LoRa


// OLED scherm
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RST); // De
declaratie van de display zodat deze variabele later kan gebruikt worden om te
kst op het scherm te tonen.

// Hartslag meter
PulseOximeter pox; // Deze variabele is nodig voor de lezing van de sensor.
float BPM{}, SpO2{}; // Deze variabele zijn er om de harslag lezingen makkelij
ker op te slagen en op te roepen.
uint32_t tsLastReport{}; // Hier wordt de laatste opgenomen tijd opgeslaagt.
String hart{""}; // Deze variabele is nodig om de BPM en Sp02 in een string op
 te slagen zodat deze kan verzonden worden via LoRa.

// LED
bool toggleLed{true}; // Hier wordt bijgehouden wat de stand van de LED is.
int secondenTussenLedInterval{4}; // De tijd tussen dat de LED aan en uit staa
t

#pragma endregion GlobaleVariabele
```

1.14.4 Functies

Hier worden de standaard functies van Arduino gedefinieerd en hulperfuncties om het programma duidelijker te maken en zich te laten herhalen wanneer nodig.

```
#pragma region FunctiesDefiniatie
```

Standaard Arduino functies.

```
void setup();
void loop();
```

Helper functies.

```
void loopLedEnMagneet(void *parameter);
void loopHartSlagSensorGpsEnLora(void *parameter);
#pragma endregion FunctiesDefiniatie

void setup()
{
```

Initializering serial monitor.

```
Serial.begin(9600);
ss.begin(9600);
```

Reset OLED scherm via software.

```
pinMode(OLED_RST, OUTPUT);
digitalWrite(OLED_RST, LOW);
delay(20);
digitalWrite(OLED_RST, HIGH);
```

Initializering OLED scherm.

```
Wire.begin(OLED_SDA, OLED_SCL);
if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3c, false, false))
{
  Serial.println(F("SSD1306 scherm niet gevonden!"));
  for (;;)
    ;
}
```

Informatie over het programma wordt getoond op het scherm.

```
display.clearDisplay();
display.setTextColor(WHITE);
display.setTextSize(1);
display.setCursor(0, 0);
display.print("LORA ZENDER ");
display.display();

Serial.println("LoRa zender test");
```

SPI LoRa pinnen.

```
SPI.begin(SCK, MISO, MOSI, SS);
```

Setup LoRa transceiver module.

```
LoRa.setPins(SS, RST, DIO0);

if (!LoRa.begin(BAND))
{
  Serial.println("LoRa starten mislukt!");
  while (1)
    ;
}
```

```
  Serial.println("LoRa initializatie OK!");
  display.setCursor(0, 10);
  display.print("LoRa initializatie OK!");
  display.display();
```

Hartslag setup.

```
  pinMode(19, OUTPUT);

  Serial.print("Initializing Pulse Oximeter..");
  display.setCursor(0, 20);
  display.print("Initializing Pulse Oximeter..");
  display.display();

  if (!pox.begin())
  {
    Serial.println("Mislukt!");
    display.setCursor(0, 30);
    display.print("Mislukt!");
    display.display();
    for (;;)
      ;
  }
  else
  {
    Serial.println("Gelukt!");
    display.setCursor(0, 30);
    display.print("Gelukt!");
    display.display();
  }
```

LED en magneet setup

```
//LED en Magneet
  pinMode(PINLED, OUTPUT);
  pinMode(PINMAGNEET, OUTPUT);
```

De standaard stroom van de IR LED is 50mA en het can be changed by uncommenting the following line.

```
  pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);
```

Er wordt een proces aangemaakt voor de hartslagsensor, GPS en LoRa communicatie.

```
  xTaskCreatePinnedToCore(
      loopHartSlagSensorGpsEnLora,     // De Functie die wordt opgeroepen.
      "loopHartSlagSensorGpsEnLora", // De naam dat je de task wilt geven.
```

```
      1024, // De grote dat de task krijgt als opslag.
      NULL, // De parameter nodig voor de task.
      1, // De prioriteit van de task. Hoe groter het nummer hoe hoger de prio
riteit.
      NULL, // De task handle om de gemaakte task bij te houden.
      1); // De CPU core waar dat je de task wilt op runnen.
}

void loop()
{
  xTaskCreatePinnedToCore(
      loopLedEnMagneet,
      "loopLedEnMagneet",
      1024,
      NULL,
      2,
      NULL,
      0);
```

Activeert de idle state voor x aantal second en gaat dan door. Tijdens deze idle state kunnen andere proces worden uitgevoerd.

```
  vTaskDelay((secondenTussenLedInterval * 1000) / portTICK_PERIOD_MS);
}
```

Een functie om de LED en de magneet te besturen.

```
void loopLedEnMagneet(void *parameter)
{

  if (release == true)
  {
    Serial.println("Magneet en LED loop");
    counterLed++;

    if (toggleLed)
    {
      digitalWrite(PINLED, LOW);
      digitalWrite(PINMAGNEET, LOW);
    }
    else
    {
      digitalWrite(PINLED, HIGH);
      digitalWrite(PINMAGNEET, HIGH);
    }
    toggleLed = !toggleLed;
  }
```

Deze task wordt maar 1 keer gerund en dan verwijdert.

```
  vTaskDelete(NULL);
}
```

Een functie om de hartslagsensor en GPS te lezen in dit door te sturen via LoRa communicatie.
Ook wordt hier de status van de LED en de magneet bepaald.

```
void loopHartSlagSensorGpsEnLora(void *parameter)
{
  for (;;)
  {
```

De module wacht totdat het een bericht krijgt om de magneet los te laten. Als er een pakket binnenkomt via LoRa wordt de variabele packetSize 1. Als het bericht "release" wordt de magneet op afgezet en begint de LED te flikkeren.

```
while (release == false)
    {
      display.clearDisplay();
      display.setCursor(0, 0);
      display.print("waiting for data");
      display.display();

      int packetSize = LoRa.parsePacket();
      if (packetSize)
      {
        while (LoRa.available())
        {
          LoRaData = LoRa.readString();
          Serial.println(LoRaData);

          if (LoRaData == "release")
          {
            digitalWrite(PINMAGNEET, HIGH);
            digitalWrite(PINLED, HIGH);
            delay(200);
            digitalWrite(PINMAGNEET, LOW);
            digitalWrite(PINLED, LOW);
            delay(200);
            digitalWrite(PINMAGNEET, HIGH);
            digitalWrite(PINLED, HIGH);
            delay(2000);

            release = true;
          }
          else
          {
            digitalWrite(PINMAGNEET, LOW);
```

```
            digitalWrite(PINLED, HIGH);
            release = false;
          }
        }
      }
    }
```

```
    pox.update();
    BPM = pox.getHeartRate();
    SpO2 = pox.getSpO2();
```

De hartslag wordt gemeten en elke 'REPORTING_PERIOD_MS' miliseconden wordt deze getoond in de seriële monitor.

```
    if (millis() - tsLastReport > REPORTING_PERIOD_MS)
    {
      Serial.print("Heart rate:");
      Serial.print(BPM);
      Serial.print(" bpm / SpO2:");
      Serial.print(SpO2);
      Serial.println(" %");
      tsLastReport = millis();
    }
```

De gelezen waarde van de hartslagsensor wordt opgeslagen in een string met twee cijfers achter de komma.

```
    hart = String(BPM, 2) + " BPM" + "   " + String(SpO2, 2) + "% 02";
```

Als de GPS actief is en de locatie is anders dan de voorgaande gelezen locatie dan wordt dit in een String object genaamd 'cor' gestoken, en wordt dit samen met de hartslagsensor lezingen opgestuurd via LoRa communicatie.

```
    while (ss.available() > 0)
    {
      gps.encode(ss.read());
      if (gps.location.isUpdated())
      {
        cor = String(gps.location.lat(), 6) + ";" + String(gps.location.lng(),
6);
        Serial.println(cor);

        LoRa.beginPacket();
        LoRa.print(cor);
        LoRa.print(hart);
        LoRa.endPacket();
        Serial.println("data send");
```

Hier heeft men de kans om de module te herstarten doormiddel van een LoRa pakket met data "reset". Deze while loop zal 5 seconden bestaan totdat het verdergaat met de code en in de volgende cyclus zich hier weer bevindt. Als LoRa een pakket binnen krijgt met data "release" dan wordt de release variabele terug op false gezet en is de magneet terug actief en kan de module weer gedropt worden.

```
      while (currMil - prevMil < wait && release)
        {

          currMil = millis();
          // Serial.println(currMil-prevMil);
          int packetSize = LoRa.parsePacket();
          if (packetSize)
          {
            while (LoRa.available())
            {
              LoRaData = LoRa.readString();
              Serial.println(LoRaData);
              Serial.println("waiting for reset");

              if (LoRaData == "reset")
              {
                release = false;
                Serial.println("RESET");
                digitalWrite(PINMAGNEET, LOW);
                digitalWrite(PINLED, HIGH);
              }
            }
          }
        }
      }
    }
```

Het scherm toont de opgestuurde waarde.

```
    display.clearDisplay();
    display.setCursor(0, 0);
    display.print("send:");
    display.setCursor(0, 20);
    display.print(cor);
    display.setCursor(0, 30);
    display.print(hart);
    display.display();
  }
```

Dit wordt nooit uitgevoerd omdat de lus hierboven oneindig is, maar je weet maar nooit.

```
  vTaskDelete(NULL);
}
```

1.14.5 Gehele code

```
#pragma region Includes
// FreeRTOS
#include <Arduino.h>

// GPS
#include <SoftwareSerial.h>
#include <TinyGPS++.h>

// LoRa
#include <SPI.h>
#include <LoRa.h>

// OLED scherm
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// Hartslagmeter
#include <MAX30100_PulseOximeter.h>

#pragma endregion Includes

#pragma region Defines
// FreeRTOS

// GPS

// LoRa
#define SCK 5
#define MISO 19
#define MOSI 27
#define SS 18
#define RST 23
#define DIO0 26
#define BAND 866E6 // voor Europa
//#define BAND 433E6 // voor Azië
//#define BAND 915E6 // voor Noord-Amerika

// OLED scherm
#define OLED_SDA 21
#define OLED_SCL 22
```

```
#define OLED_RST 16
#define SCREEN_WIDTH 128 // De breedte van het scherm.
#define SCREEN_HEIGHT 64 // De hoogte van het scherm.

#define PINLED 35
#define PINMAGNEET 14
#define PINHARTSLAG 19

// Hartslagmeter
#define REPORTING_PERIOD_MS 2000

#pragma endregion Defines

#pragma region GlobaleVariabele
// FreeRTOS

// GPS
String cor{""};            // Deze variabele is nodig om de lengtegraad en bre
edtegraad in een string op te slagen zodat deze kan verzonden worden via LoRa.
TinyGPSPlus gps;           // Een variabele die nodig is om de locatie van dez
e module te weten.
SoftwareSerial ss(15, 13); // De Rx en Tx pinnen waarde de GPS module aanhangt
.

// LoRa

int wait = 5000;

unsigned long prevMil;
unsigned long currMil;

// OLED scherm
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RST); // De
declaratie van de display zodat deze variabele later kan gebruikt worden om te
kst op het scherm te tonen.
int counterHartslag{};
int counterLed{};

// Hartslag meter
PulseOximeter pox;       // Deze variabele is nodig voor de lezing van de sens
or.
float BPM{}, SpO2{};     // Deze variabele zijn er om de harslag lezingen makk
elijker op te slagen en op te roepen.
uint32_t tsLastReport{}; // Hier wordt de laatste opgenomen tijd opgeslaagt.
String hart{""};         // Deze variabele is nodig om de BPM en Sp02 in een s
tring op te slagen zodat deze kan verzonden worden via LoRa.

// LED
```

```
bool toggleLed{true};             // Hier wordt bijgehouden wat de stand van d
e LED is.
int secondenTussenLedInterval{1}; // De tijd tussen dat de LED aan en uit staa
t

bool release{false};
String LoRaData;

#pragma endregion GlobaleVariabele

#pragma region FunctiesDefiniatie
// Standaard Arduino functies
void setup();
void loop();

// Helper functies
void loopLedEnMagneet(void *parameter);
void loopHartSlagSensorGpsEnLora(void *parameter);
#pragma endregion FunctiesDefiniatie

void setup()
{
  // Initializering serial monitor
  Serial.begin(9600);
  ss.begin(9600);

  // Reset OLED scherm via software
  pinMode(OLED_RST, OUTPUT);
  digitalWrite(OLED_RST, LOW);
  delay(20);
  digitalWrite(OLED_RST, HIGH);

  // Initializering OLED scherm
  Wire.begin(OLED_SDA, OLED_SCL);
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3c, false, false))
  {
    Serial.println(F("SSD1306 scherm niet gevonden!"));
    for (;;)
      ;
  }

  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setTextSize(1);
  display.setCursor(0, 0);
  display.print("LORA ZENDER ");
  display.display();
```

```
Serial.println("LoRa zender test");

// SPI LoRa pinnen
SPI.begin(SCK, MISO, MOSI, SS);
// Setup LoRa transceiver module
LoRa.setPins(SS, RST, DIO0);

if (!LoRa.begin(BAND))
{
  Serial.println("LoRa starten mislukt!");
  while (1)
    ;
}
Serial.println("LoRa initializatie OK!");
display.setCursor(0, 10);
display.print("LoRa initializatie OK!");
display.display();

// Hartslag setup
pinMode(PINHARTSLAG, OUTPUT);

//LED en Magneet
pinMode(PINLED, OUTPUT);
pinMode(PINMAGNEET, OUTPUT);

Serial.print("Initializing Pulse Oximeter..");
display.setCursor(0, 20);
display.print("Initializing Pulse Oximeter..");
display.display();

if (!pox.begin())
{
  Serial.println("Mislukt!");
  display.setCursor(0, 40);
  display.print("Mislukt!");
  display.display();
}
else
{
  Serial.println("Gelukt!");
  display.setCursor(0, 40);
  display.print("Gelukt!");
  display.display();
}

delay(5000);
display.clearDisplay();
display.display();
```

```
  // De standaard stroom van de IR LED is 50mA en het can be changed by uncomm
enting the following line.
  pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);

  xTaskCreatePinnedToCore(
      loopHartSlagSensorGpsEnLora,   // De Functie die wordt opgeroepen.
      "loopHartSlagSensorGpsEnLora", // De naam dat je de task wilt geven.
      10240,                         // De grote dat de task krijgt als opslag
.
      NULL,                          // De parameter nodig voor de task.
      1,                             // De prioriteit van de task. Hoe groter
het nummer hoe hoger de prioriteit.
      NULL,                          // De task handle om de gemaakte task bij
 te houden.
      1);                            // De CPU core waar dat je de task wilt o
p runnen.
}

void loop()
{
  xTaskCreatePinnedToCore(
      loopLedEnMagneet,
      "loopLedEnMagneet",
      10240,
      NULL,
      2,
      NULL,
      0);
  vTaskDelay((secondenTussenLedInterval * 1000) / portTICK_PERIOD_MS); // Acti
veerd de idle state voor x aantal second en gaat dan door. tijdens deze idle s
tate kunnen andere process worden uitgevoerd
}

// Een functie om de LED en de magneet te besturen
void loopLedEnMagneet(void *parameter)
{

  if (release == true)
  {
    Serial.println("Magneet en LED loop");
    counterLed++;

    if (toggleLed)
    {
      digitalWrite(PINLED, LOW);
      digitalWrite(PINMAGNEET, LOW);
    }
```

```
    else
    {
      digitalWrite(PINLED, HIGH);
      digitalWrite(PINMAGNEET, HIGH);
    }
    toggleLed = !toggleLed;
  }

  vTaskDelete(NULL); // Deze task wordt maar 1 keer gerund en dan verwijdert.
}

// Een functie om de hartslagsensor en GPS te lezen in dit door te sturen via
LoRa communicatie
// Ook wordt hier de aanspraak van de LED en de magneet
void loopHartSlagSensorGpsEnLora(void *parameter)
{
  for (;;)
  {
    while (release == false) // De module wacht totdat het een bericht krijgt
om de magneet los te laten.
    {
      display.clearDisplay();
      display.setCursor(0, 0);
      display.print("waiting for data");
      display.display();

      int packetSize = LoRa.parsePacket(); // Als er een pakket binnenkomt via
 LoRa wordt deze variabele 1 krijgen.
      if (packetSize)
      {
        while (LoRa.available())
        {
          LoRaData = LoRa.readString(); // De string van het LoRa pakket wordt
 gelezen en in een variabele geplaatst
          Serial.println(LoRaData);

          if (LoRaData == "release") // Als het bericht "release" is worden vo
lgende lijnen code uitgevoert.
          {
            digitalWrite(PINMAGNEET, HIGH);
            digitalWrite(PINLED, HIGH);
            delay(200);
            digitalWrite(PINMAGNEET, LOW);
            digitalWrite(PINLED, LOW);
            delay(200);
            digitalWrite(PINMAGNEET, HIGH);
            digitalWrite(PINLED, HIGH);
            delay(2000);
```

```
            release = true;
          }
          else // Zolang het bericht niet "release" is worden volgende lijnen
code uitgevoert.
          {
            digitalWrite(PINMAGNEET, LOW);
            digitalWrite(PINLED, HIGH);
            release = false;
          }
        }
      }
    }
    // Als de module losgelaten is gaat men verdere in de code.

    pox.update(); // Hier wordt de Hartslag modulde geupdate
    BPM = pox.getHeartRate(); // Hier wordt de hartslag gemeten
    SpO2 = pox.getSpO2(); // Hier wordt het bloedzuurstof gemeten.
    // De hartslag wordt gemeten en elke 'REPORTING_PERIOD_MS' miliseconden wo
rdt deze getoont in de seriele monitor.
    if (millis() - tsLastReport > REPORTING_PERIOD_MS)
    {
      counterHartslag++;

      // Serial.print("Heart rate:");
      // Serial.print(BPM);
      // Serial.print(" bpm / SpO2:");
      // Serial.print(SpO2);
      // Serial.println(" %");
      tsLastReport = millis();
    }
    // De geleze waarde van de hartslagsensor wordt opgeslagen in een string m
et 2 cijfers achter de komma
    hart = String(BPM, 2) + " BPM" + "   " + String(SpO2, 2) + "% 02";

    // Als de GPS actief is en de locatie is anders dan de voorgaande gelezen
locatie dan wordt dit in een String object genaamd 'cor' gestoken,
    // en wordt dit samen met de hartslagsensor lezingen opgestuurd via LoRa c
ommunicatie.
    while (ss.available() > 0)
    {
      gps.encode(ss.read());
      if (gps.location.isUpdated())
      {
        cor = String(gps.location.lat(), 6) + ";" + String(gps.location.lng(),
 6);
        Serial.println(cor);
```

```
        LoRa.beginPacket();
        LoRa.print(cor);
        LoRa.print(hart);
        LoRa.endPacket();

        prevMil = currMil;

        currMil = millis();

        while (currMil - prevMil < wait && release)
        {
          // Hier heeft men de kans om de module te herstarten doormiddel van
een LoRa pakket met data "reset".
          // Deze while loop zal 5 seconden bestaan totdat het verdergaat met
de code en in de volgende cyclus zich hier weer bevindt.
          // Als LoRa een pakket binnen krijgt met data "release" dan wordt de
 release variabele terug op false gezet en is de
          // magneet terug actief en kan de module weer gedropt worden.

          currMil = millis();
          // Serial.println(currMil-prevMil);
          int packetSize = LoRa.parsePacket();
          if (packetSize)
          {
            while (LoRa.available())
            {
              LoRaData = LoRa.readString();
              Serial.println(LoRaData);
              Serial.println("waiting for reset");

              if (LoRaData == "reset")
              {
                release = false;
                Serial.println("RESET");
                digitalWrite(PINMAGNEET, LOW);
                digitalWrite(PINLED, HIGH);
              }
            }
          }
        }
        // Serial.println("data send");
      }
    }
    // Het scherm toont de opgestuurde waarde
    display.clearDisplay();
    display.setCursor(0, 0);
    display.print("send:");
    display.setCursor(0, 20);
```

```
    display.print(cor);
    display.setCursor(0, 30);
    display.print(hart);
    display.setCursor(0, 50);
    display.print(counterHartslag);
    display.setCursor(30, 50);
    display.print(counterLed);
    display.display();
  }
  vTaskDelete(NULL); // Dit wordt nooit uitgevoerd omdat de lus hierboven onei
ndig is, maar je weet maar nooit.
}
```

1.15 Ontvanger

Functies:

- Gps-locatie zoekers en vermist persoon vergelijken
- Afstand tot vermist persoon weergeven
- Zoekrichting aangeven
- Hartslag vermist persoon tonen
- LoRa data ontvangen

1.15.1 Libraries

Bij de zender werden er libraries toegevoegd om sensoren en actoren te laten werken. Bij de ontvanger is dit ook het geval. Een GPS wordt gebruikt om de exacte locatie te krijgen van de module. LoRa wordt gebruikt om communicatie tussen zoekers en patiënt mogelijk te maken. Een OLED-scherm dat zich bevindt op de ESP32 wordt gebruik om instructies te geven aan de patiënt en de hartslag wordt hiermee ook getoond.

```
#include <SoftwareSerial.h>
#include <TinyGPS++.h>


//Libraries for LoRa
#include <SPI.h>
#include <LoRa.h>

//Libraries for OLED Display
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

1.15.2 Defines

Op de module van de zoekers zal er ook gebruik gemaakt worden van de pinnen. Deze moet eerst gedefinieerd worden.

```
//define the pins used by the LoRa transceiver module
#define SCK 5
#define MISO 19
#define MOSI 27
#define SS 18
#define RST 23
#define DIO0 26

//433E6 for Asia
//866E6 for Europe
//915E6 for North America
#define BAND 866E6

//OLED pins
#define OLED_SDA 21
#define OLED_SCL 22
#define OLED_RST 16
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RST);
```

1.15.3 Globale variabelen

De globale variabele zijn nodig om variabele doorheen functies van het programma te gebruiken. De gegevens van de GPS zullen in een variabelen gestoken worden.

```
String cor = "";
float glat = 0;
float glng = 0;

float lat1 = 0;
float lng1 = 0;
float lat2 = 0;
float lng2 = 0;

TinyGPSPlus gps;
SoftwareSerial ss(15, 13);

String LoRaData;
```

1.15.4 Setup

Opzetten van de display.

```
  //reset OLED display via software
  pinMode(OLED_RST, OUTPUT);
  digitalWrite(OLED_RST, LOW);
  delay(20);
  digitalWrite(OLED_RST, HIGH);

  //initialize OLED
  Wire.begin(OLED_SDA, OLED_SCL);
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3c, false, false)) { // Address 0x
3C for 128x32
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }
  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setTextSize(1);
  display.setCursor(0,0);
  display.print("LORA RECEIVER ");
  display.display();

  Serial.println("LoRa Receiver Test");
```

Opzetten LoRa communicatie.

```
  //SPI LoRa pins
  SPI.begin(SCK, MISO, MOSI, SS);
  //setup LoRa transceiver module
  LoRa.setPins(SS, RST, DIO0);

  if (!LoRa.begin(BAND)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }
  Serial.println("LoRa Initializing OK!");
  display.setCursor(0,10);
  display.println("LoRa Initializing OK!");
  display.display();
```

Functie om de afstand tussen de twee modules te berekenen. De Haversine formule wordt vaak gebruikt omdat het ook nauwkeurig is over korte afstanden. Het maakt gebruik van cirkels om een afstand te krijgen.

```
float HaverSine(float lat1, float lon1, float lat2, float lon2)
{
  float ToRad = PI / 180.0;
```

```
  float R = 6371;   // radius earth in Km

  float dLat = (lat2-lat1) * ToRad;
  float dLon = (lon2-lon1) * ToRad;

  float a = sin(dLat/2) * sin(dLat/2) +
        cos(lat1 * ToRad) * cos(lat2 * ToRad) *
        sin(dLon/2) * sin(dLon/2);

  float c = 2 * atan2(sqrt(a), sqrt(1-a));

  float d = R * c;
  return d;
}
```

1.15.5 Loop

In de loop worden de berekeningen gemaakt tussen de 2 GPS coördinaten (patiënt en zoekers). Zo wordt de afstand berekent, de richting in graden en de windrichting waar de patiënt zich bevindt vanuit de zoekers hun positie.

Nadien wordt er gekeken of er LoRa communicatie binnenkomt en de display wordt gerefresht met de nieuwe informatie.

```
  distance = HaverSine(lat1, lng1, lat2, lng2) * 1000; // De afstand tussen de
 patiënt en de zoekers wordt berekent.

  graden = gps.courseTo(lat1, lng1, lat2, lng2); // De richting van de patiënt
 wordt bepaald vanop de zoekers hun standpunt.

  compass.read(); // Het kompas wordt ingelezen.

  x = compass.getAzimuth(); // Doormiddel van de getAzimuth functie van de QMC
5883LCompass library kunnen we de richting van het kompas krijgen in graden.
  x =  -
x - 97 + graden; // Hier worden de graden omgedraait zodat het kompas tegenwij
zers zin draait met een kleine offset
  // die nodig was om het kompas naar het noorden te draaien en de graden waar
 de patiënt in het water ligt.

  // Hier wordt nagekeken in welke richting de patiënt ligt. Dit wordt dan get
oont met de juiste windrichting op het scherm.
  if (graden < 15 || graden > 345)
  {
    kompas = "N";
  }
  else if (graden > 15 && graden < 30)
  {
    kompas = "NNE";
```

```
}
else if (graden > 30 && graden < 60)
{
  kompas = "NE";
}
else if (graden > 60 && graden < 75)
{
  kompas = "ENE";
}
else if (graden > 75 && graden < 105)
{
  kompas = "E";
}
else if (graden > 105 && graden < 120)
{
  kompas = "ESE";
}
else if (graden > 120 && graden < 150)
{
  kompas = "SE";
}
else if (graden > 150 && graden < 165)
{
  kompas = "SSE";
}
else if (graden > 165 && graden < 195)
{
  kompas = "S";
}
else if (graden > 195 && graden < 210)
{
  kompas = "SSW";
}
else if (graden > 210 && graden < 240)
{
  kompas = "SW";
}
else if (graden > 240 && graden < 255)
{
  kompas = "WSW";
}
else if (graden > 255 && graden < 285)
{
  kompas = "W";
}
else if (graden > 285 && graden < 300)
{
  kompas = "WNW";
```

```
  }
  else if (graden > 300 && graden < 330)
  {
    kompas = "NW";
  }
  else if (graden > 330 && graden < 345)
  {
    kompas = "NNW";
  }

  LoRaLezen(NULL);
  UpdateDisplay(NULL);
```

1.15.6 Volledige code

```
#include <SoftwareSerial.h>
#include <TinyGPS++.h>
//Libraries for LoRa
#include <SPI.h>
#include <LoRa.h>
//Libraries for OLED Display
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <QMC5883LCompass.h>
//define the pins used by the LoRa transceiver module
#define SCK 5
#define MISO 19
#define MOSI 27
#define SS 18
#define RST 23
#define DIO0 26
//433E6 for Asia

//866E6 for Europe
//915E6 for North America
#define BAND 866E6
//OLED pins
#define OLED_SDA 21
#define OLED_SCL 22
#define OLED_RST 16
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RST);
String cor = "";
float glat = 0;
```

```
float glng = 0;
float lat1 = 51.242844;
float lng1 = 5.139560;
float lat2 = 51.238881;
float lng2 = 5.140225;
int hr{};

String kompas;
double graden{0};
float distance{};
float angleTest{};
int x, y, z;

TinyGPSPlus gps;
SoftwareSerial ss(15, 13);
String LoRaData;
String berichtMagneet;
bool isReleased{false};

QMC5883LCompass compass;

const int PushButton{12};

void LoRaLezen(void *parameter);
void UpdateDisplay(void *parameter);

void setup()
{
  //initialize Serial Monitor
  Serial.begin(9600);
  ss.begin(9600);

  //initialize OLED
  Wire.begin(OLED_SDA, OLED_SCL);
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3c, false, false))
  { // Address 0x 3C for 128x32
    Serial.println(F("SSD1306 allocation failed"));
    for (;;)
      ; // Don't proceed, loop forever
  }

  pinMode(PushButton, INPUT);

  //reset OLED display via software
  pinMode(OLED_RST, OUTPUT);
  digitalWrite(OLED_RST, LOW);
  delay(20);
  digitalWrite(OLED_RST, HIGH);
```

```
  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setTextSize(1);
  display.setCursor(0, 0);
  display.print("LORA RECEIVER ");
  display.display();
  Serial.println("LoRa Receiver Test");

  //SPI LoRa pins
  SPI.begin(SCK, MISO, MOSI, SS);
  //setup LoRa transceiver module
  LoRa.setPins(SS, RST, DIO0);
  if (!LoRa.begin(BAND))
  {
    Serial.println("Starting LoRa failed!");
    while (1)
      ;
  }
  Serial.println("LoRa Initializing OK!");
  display.setCursor(0, 10);
  display.println("LoRa Initializing OK!");
  display.display();

  compass.init();
  compass.setCalibration(0, 3053, -3594, 0, -3198, 0);

  delay(2000);
}

float HaverSine(float lat1, float lon1, float lat2, float lon2)
{
  float ToRad = PI / 180.0;
  float R = 6371; // radius earth in Km

  float dLat = (lat2 - lat1) * ToRad;
  float dLon = (lon2 - lon1) * ToRad;

  float a = sin(dLat / 2) * sin(dLat / 2) +
            cos(lat1 * ToRad) * cos(lat2 * ToRad) *
                sin(dLon / 2) * sin(dLon / 2);

  float c = 2 * atan2(sqrt(a), sqrt(1 - a));

  float d = R * c;
  return d;
}
```

```
void loop()
{
  distance = HaverSine(lat1, lng1, lat2, lng2) * 1000; // De afstand tussen de
 patiënt en de zoekers wordt berekent.

  graden = gps.courseTo(lat1, lng1, lat2, lng2); // De richting van de patiënt
 wordt bepaald vanop de zoekers hun standpunt.

  compass.read(); // Het kompas wordt ingelezen.

  x = compass.getAzimuth(); // Doormiddel van de getAzimuth functie van de QMC
5883LCompass library kunnen we de richting van het kompas krijgen in graden.
  x =  -
x - 97 + graden; // Hier worden de graden omgedraait zodat het kompas tegenwij
zers zin draait met een kleine offset
  // die nodig was om het kompas naar het noorden te draaien en de graden waar
 de patiënt in het water ligt.

  // Hier wordt nagekeken in welke richting de patiënt ligt. Dit wordt dan get
oont met de juiste windrichting op het scherm.
  if (graden < 15 || graden > 345)
  {
    kompas = "N";
  }
  else if (graden > 15 && graden < 30)
  {
    kompas = "NNE";
  }
  else if (graden > 30 && graden < 60)
  {
    kompas = "NE";
  }
  else if (graden > 60 && graden < 75)
  {
    kompas = "ENE";
  }
  else if (graden > 75 && graden < 105)
  {
    kompas = "E";
  }
  else if (graden > 105 && graden < 120)
  {
    kompas = "ESE";
  }
  else if (graden > 120 && graden < 150)
  {
    kompas = "SE";
  }
```

```
  else if (graden > 150 && graden < 165)
  {
    kompas = "SSE";
  }
  else if (graden > 165 && graden < 195)
  {
    kompas = "S";
  }
  else if (graden > 195 && graden < 210)
  {
    kompas = "SSW";
  }
  else if (graden > 210 && graden < 240)
  {
    kompas = "SW";
  }
  else if (graden > 240 && graden < 255)
  {
    kompas = "WSW";
  }
  else if (graden > 255 && graden < 285)
  {
    kompas = "W";
  }
  else if (graden > 285 && graden < 300)
  {
    kompas = "WNW";
  }
  else if (graden > 300 && graden < 330)
  {
    kompas = "NW";
  }
  else if (graden > 330 && graden < 345)
  {
    kompas = "NNW";
  }

  LoRaLezen(NULL);
  UpdateDisplay(NULL);
}

void LoRaLezen(void *parameter)
{
  int Push_button_state = digitalRead(PushButton); // De digitale pin van de k
nop wordt afgelezen.
  if (!isReleased)
  {
```

```
    if (Push_button_state == HIGH) // Als deze digitale pin 1 is dan wordt het
 pakketje gedropt.
    {
      LoRa.beginPacket(); // Er wordt een pakket aangemaakt om een LoRa berich
t te sturen.
      LoRa.print("release"); // De inhoud van het pakket wordt gevult.
      LoRa.endPacket(); // Het pakket wordt afgesloten en opgestuurd.
      berichtMagneet = "Released!";
      isReleased = true;
      delay(200);
    }
    else// Als deze digitale pin 0 is dan blijft het pakketje hangen.
    {
      berichtMagneet = "Still hanging.";
    }
  }
  else
  {
    if (Push_button_state == HIGH) // Als deze digitale pin 1 is wordt het sys
teem gereset.
    {
      LoRa.beginPacket();
      LoRa.print("reset");
      LoRa.endPacket();
      berichtMagneet = "Reset";
      isReleased = false;
      delay(200);
    }
    while (ss.available() > 0) // Als de GPS aanwezig is wordt de lokatie gele
zen en wordt deze informatie opgeslagen in variabelen.
    {
      gps.encode(ss.read());
      if (gps.location.isUpdated())
      {
        lat1 = (gps.location.lat());
        lng1 = (gps.location.lng());
        cor = String(lat1, 6) + ";" + String(lng1, 6);
      }
    }
    //try to parse packet
    int packetSize = LoRa.parsePacket(); // Als er een pakket binnenkomt via L
oRa wordt deze variabele 1 krijgen.
    if (packetSize)
    {
      while (LoRa.available())
      {
        LoRaData = LoRa.readString();
        //Serial.print(LoRaData);
```

```
      }

      int index = LoRaData.indexOf(";");
      // Serial.println(index);
      lat2 = LoRaData.substring(0, index).toFloat();
      // Serial.println(lat2);
      lng2 = LoRaData.substring(index + 1).toFloat();
      // Serial.println(lng2);
      hr = LoRaData.substring(index + 2).toInt();
    }
  }
}

void UpdateDisplay(void *parameter)
{
    // In deze functie wordt alle benodigde informatie op het scherm getoont.
    // Zo weten de zoekers de coördinaten van de patiënt en kan met ook te wet
en
    // komen in welke richting de patiënt zich van hun bevind.
    // De berekende afstand tussen de zoekers en de patiënt wordt getoont in m
eter.
    // Er is ook een wijzer aanwezig die naar de richting van de patiënt wijst
.

  float circle{(2 * PI) / 360.f}, radius{15.f}, positionX{(display.width() / 4
.f) * 3.f}, positionY{display.height() / 2.f + 5.f};
  float topArrowPoint{-90.f}, leftArrowPoint{-250.f}, rightArrowPoint{-290.f};

  display.clearDisplay();

  display.setCursor(0, 0);
  display.println("Patient:");
  display.print("lat:");
  display.println(lat2, 6);
  display.print("lng:");
  display.println(lng2, 6);

  display.setCursor(0, display.height() - 18);
  display.print(graden, 0);
  display.print(" graden");

  display.setCursor(cos(circle * (topArrowPoint + x)) * (radius + 10) + positi
onX - 7.f, sin(circle * (topArrowPoint + x)) * (radius + 10) + positionY - 3.f
);
  display.print(kompas);

  display.setCursor(0, display.height()- 10);
  display.print("afstand:");
```

```
  display.print(distance, 0);
  display.print("m");

  display.setCursor(0, 40);
  display.print(berichtMagneet);

  display.fillTriangle(cos(circle * (topArrowPoint + x)) * radius + positionX,
 sin(circle * (topArrowPoint + x)) * radius + positionY,
                      cos(circle * (leftArrowPoint + x)) * radius + positionX
, sin(circle * (leftArrowPoint + x)) * radius + positionY,
                      cos(circle * (rightArrowPoint + x)) * radius + position
X, sin(circle * (rightArrowPoint + x)) * radius + positionY, SSD1306_INVERSE);
  display.drawCircle(positionX, positionY, radius, SSD1306_INVERSE);

  display.display();
}
```

Bronnen

A. (n.d.-a). *adafruit/Adafruit-GFX-Library*. GitHub. Retrieved May 30, 2021, from https://github.com/adafruit/Adafruit-GFX-Library

A. (n.d.-b). *adafruit/Adafruit_SSD1306*. GitHub. Retrieved May 30, 2021, from https://github.com/adafruit/Adafruit_SSD1306

A. (2019, October 10). *How Does GPS Tracker Work in a Car? Are They Good or Bad?* Cyblance. https://www.cyblance.com/car-dealership/gps-tracker-work-car-good-bad/

Advantages of lithium-ion batteries. (n.d.). Lektsii.Org. Retrieved May 30, 2021, from https://lektsii.org/14-3388.html

B. (2014, August 8). *What is LED*. News about Energy Storage, Batteries, Climate Change and the Environment. https://www.upsbatterycenter.com/blog/led/

FatSecret. (n.d.). *Calorieën in Côte d'Or Melk en Voedingswaarde Informatie*. Retrieved May 30, 2021, from https://www.fatsecret.nl/calorie%C3%ABn-voedingswaarde/c%C3%B4te-dor/melk/100g

Frequency Plans by Country. (n.d.). The Things Network. Retrieved May 30, 2021, from https://www.thethingsnetwork.org/docs/lorawan/frequencies-by-country/

Harris, T. C. P. W. F. (2021, February 12). *How Light Emitting Diodes (LEDs) Work*. HowStuffWorks. https://electronics.howstuffworks.com/led.htm

Kraudel, R. (2021, April 5). *Optical Heart Rate Monitoring: What You Need to Know*. Valencell. https://valencell.com/blog/optical-heart-rate-monitoring-what-you-need-to-know/

Lemmens, H. (2019, April 12). *UC San Diego Works to Build Batteries of the Future*. Accelerating Microscopy. https://www.thermofisher.com/blog/microscopy/uc-san-diego-works-to-build-batteries-of-the-future/

LoRa — LoRa documentation. (n.d.). Lora. Retrieved May 30, 2021, from https://lora.readthedocs.io/en/latest/

M. (n.d.-c). *mikalhart/TinyGPSPlus*. GitHub. Retrieved May 30, 2021, from https://github.com/mikalhart/TinyGPSPlus

M. (n.d.-d). *mprograms/QMC5883LCompass*. GitHub. Retrieved May 30, 2021, from https://github.com/mprograms/QMC5883LCompass

Newton, A. (2021, March 14). *Monitor SpO2/BPM with ESP32 & MAX30100 Pulse Oximeter on Blynk*. How To Electronics. https://how2electronics.com/esp32-max30100-pulse-oximeter-blynk/

OLEDs (Organic LEDs) and LEPs (light-emitting polymers). (2021, May 17). Explain That Stuff. https://www.explainthatstuff.com/how-oleds-and-leps-work.html

Oving, A. (2021, May 10). *Lcd-tv en oled*. consumentenbond. https://www.consumentenbond.nl/tv/lcd-tv-en-oled

Redactie. (n.d.). *Draadloos opladen | Wikitronics*. wikitronics. Retrieved May 30, 2021, from https://wikitronics.nl/smartphones/draadloos-opladen/#:%7E:text=Om%20draadloos%20elektriciteit%20over%20te,door%20een%20signaal%20te%20versturen

Santos, S. (2019a, October 19). *TTGO LoRa32 SX1276 OLED with Arduino IDE*. Random Nerd Tutorials. https://randomnerdtutorials.com/ttgo-lora32-sx1276-arduino-ide/

Santos, S. (2019b, October 19). *TTGO LoRa32 SX1276 OLED with Arduino IDE*. Random Nerd Tutorials. https://randomnerdtutorials.com/ttgo-lora32-sx1276-arduino-ide/

Satellite Navigation - GPS - How It Works. (2020, July 22). Faa.Gov. https://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gnss/gps/howitworks/

Sciencespace - De werking van de transformator. (n.d.). Sciencespace. Retrieved May 30, 2021, from https://www.sciencespace.nl/technologie/artikelen/4280/de-werking-van-de-transformator

Shenoy, A. (2020, May 7). *Wireless Charging Explained: How it Works and What You Need to Know*. NDTV Gadgets 360. https://gadgets.ndtv.com/mobiles/features/wireless-charging-explained-android-iphone-samsung-oneplus-qi-2224420

Wikipedia-bijdragers. (2020, October 21). *Oled*. Wikipedia. https://nl.wikipedia.org/wiki/Oled

Wikipedia-bijdragers. (2021a, May 6). *Led*. Wikipedia. https://nl.wikipedia.org/wiki/Led

Wikipedia-bijdragers. (2021b, May 19). *Lithium-ion-polymeer-accu*. Wikipedia. https://nl.wikipedia.org/wiki/Lithium-ion-polymeer-accu

Figuurlijst

Tabellen