

TEAM TURNTABLE

Labo Embedded System Design II

Daan Delabie
Pieter Vanherck
Tuur Vandecauter
Jona Cappelle
Niels Bauwens

07/05/2020

Inhoud

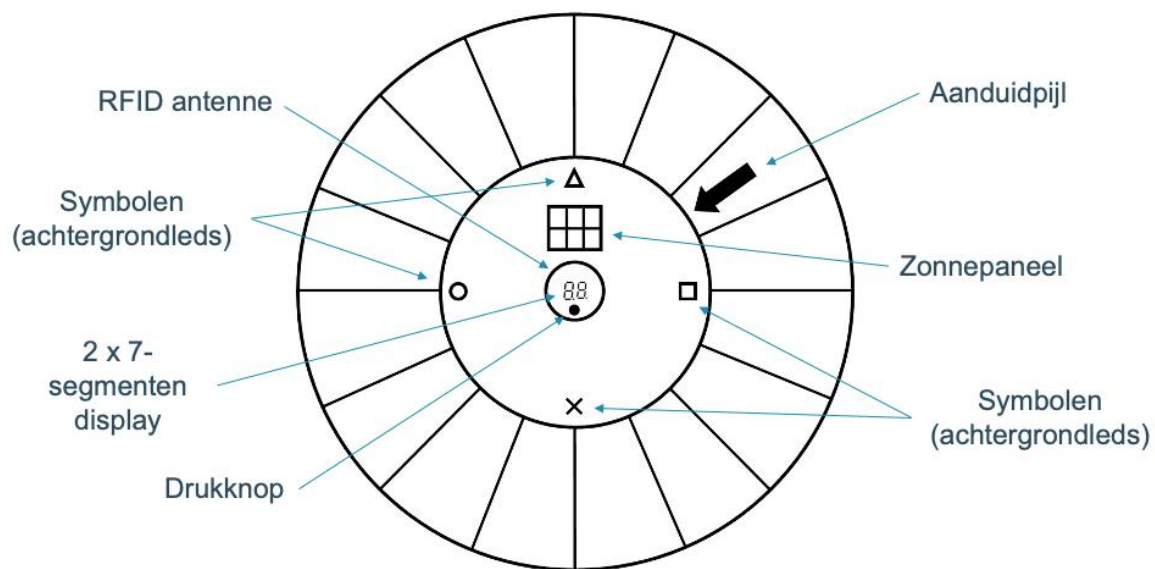
1	Conceptbeschrijving	4
2	Specificaties	5
2.1	Vereisten	5
2.2	Systeemarchitectuur	6
3	Codering van het spel	7
4	Behuizing	9
5	Speler interactie componenten	12
6	Sensoren	14
6.1	Software	14
6.2	Kompas	15
6.3	Kalibratie	16
6.4	Accelerometer	17
6.5	Energiemetingen	17
6.6	Design PCB	17
7	Energy Harvesting	19
7.1	Assumpties voor de berekeningen	19
7.2	Berekeningen	19
7.3	Zonnepaneel	21
7.4	Supercap	21
7.5	Hardware	22
7.5.1	Opladen van de supercap	22
7.5.2	Opladen van de batterij	22
7.5.3	PCB ontwerp	23
7.6	Evaluatie	23
8	Connectiviteit en Visualisatie	24
8.1	Lokaal	24
8.1.1	Payload Encoder	24
8.1.2	LoRa	25
8.2	Cloud	26
8.2.1	Payload Decoder & Data Storage	27
8.2.2	Mysql Database	28
8.2.3	Web Visualisatie	29
9	Besluit	32
9.1	Verantwoordelijkheden	33
10	Referenties	34

Figurenlijst

Figuur 1: Concept ontwerp draaitafel	4
Figuur 2: Systeemarchitectuur	6
Figuur 3: Flowchart code spel	7
Figuur 4: Technische tekening behuizing (schijf)	9
Figuur 5: Doorsnede schijf.....	9
Figuur 6: Kettingwartel (links), anti-diefstal vijs (rechts)	10
Figuur 7: Onderzijde behuizing zonder afdichtplaat	11
Figuur 8: Bovenzijde behuizing.....	11
Figuur 9: Onderzijde behuizing met afdichtplaat.....	11
Figuur 10: Gegraveerde symbolen	12
Figuur 11: Opbouw achtergrond LEDs	12
Figuur 12: Druknop.....	12
Figuur 13: Dual alfanumeriek display.....	13
Figuur 14: RFID module.....	13
Figuur 15: Meting kompas 360° rotatie	15
Figuur 16: Assen kompas voor kalibratie (links), 3D plot voor kalibratie (rechts)	16
Figuur 17: Assen kompas na kalibratie (links), 3D plot na kalibratie (rechts)	16
Figuur 18: Main PCB voorkant (links), achterkant (rechts)	18
Figuur 19: Schema voor het opladen van de supercap	22
Figuur 20: Schema voor het opladen van de batterij.....	23
Figuur 21: Voorkant (links) en achterkant (rechts) van de Energy Harvesting PCB	23
Figuur 22: Connectiviteit (lokaal)	24
Figuur 23: RN2483 LoRa Modem.....	25
Figuur 24: Energiemeting RN2483	26
Figuur 25: Connectiviteit en visualisatie (cloud)	26
Figuur 26: TTN Decoder output.....	27
Figuur 27: Swagger API a.d.h.v. cURL.....	28
Figuur 28: Python output	29
Figuur 29: Toegevoegd record in MYSQL database	29
Figuur 30: Website "Today's Best"	30
Figuur 31: Website "Hall of Fame" (links) en "Latest plays" (rechts)	30
Figuur 32: Website volledig.....	31

1 Conceptbeschrijving

Op het terras bij het studentenrestaurant van Technologicampus Gent zijn enkele tafels voorzien om buiten te zitten, eten, werken, Studenten blijven echter niet lang zitten. Met het project 'TurnTable' wordt ervoor gezorgd dat mensen langer blijven zitten aan de voorziene tafels. Hierdoor wordt een leuke campussfeer gecreëerd. Om dit te doen wordt van de tafel een interactief spel gemaakt dat gebaseerd is op het spel 'Simon'. Dit is een geheugenspel waarbij een bepaalde sequentie van vier verschillende symbolen wordt weergegeven. De speler moet de combinatie van die symbolen herhalen. Indien de speler hierin slaagt wordt een extra symbool toegevoegd aan de sequentie. Dit gaat verder tot de speler een fout maakt. Zijn score is gelijk aan het maximum aantal symbolen die de speler onthouden heeft.



Figuur 1: Concept ontwerp draaitafel

De tafels zijn voorzien van een draaiend platform in het midden. Hiervan wordt gebruik gemaakt om het spel een nieuwe dimensie te geven. Het volledig systeem wordt ingewerkt in de schijf. Indien de schijf beweegt wordt met behulp van een accelerometer de aanwezigheid van een speler vastgelegd. Zo kan het systeem uit *sleepmode* gehaald worden. De positie waarin de schijf gedraaid staat bepaalt het symbool. Om de sequentie van symbolen te vormen in het spel moet aan de schijf gedraaid worden tot deze op een bepaalde positie staat, dit voor ieder symbool. Er wordt een drukknop voorzien waarbij de speler de symbool positie van de schijf kan bevestigen. De positie van de schijf wordt bepaald aan de hand van een kompas sensor.

De speler krijgt na het spel zijn score te zien op twee 7-segment displays. Daarnaast wordt de score via LoRa doorgestuurd naar de *cloud* zodat spelers online een *highscore* tabel kunnen raadplegen. Om een identiteit te geven aan de spelers wordt vooraleer het spel te starten de studentenkaart van de speler gescand via RFID. Op die manier kunnen ook enkel studenten afkomstig van de campus dit spel spelen. Het volledige *embedded system* werkt autonoom met behulp van een batterij en een zonnepaneel. Bij dit ontwerp wordt rekening gehouden met het retrofitten van de tafel, die bovendien onderhevig is aan de Belgische weersomstandigheden, waarbij gestreefd wordt naar een esthetisch mooi design.

2 Specificaties

Na het uitwerken van het concept idee zijn we aan de slag gegaan met het opstellen van de specificaties van het volledige systeem. Hierdoor hebben we bij het verder uitwerken van het project steeds een houvast, en kan er bij het maken van keuzes teruggekoppeld worden of het voldoet aan de vooropgestelde specificaties.

2.1 Vereisten

Functionele vereisten

- Autonome werking voor minimum 5 jaar
- Data moet na elk spel verzonden worden naar de cloud
- Data moet altijd online beschikbaar zijn met een performantie van 90%
- Communicatie link moet een performantie hebben van 90%
- Het spel moet 97% van de tijd goed werken
- Het spel moet gedurende een uur kunnen gespeeld worden zonder stoppen
- De positie van de schijf moet geregistreerd worden met een afwijking tussen -35° tot 35° gemeten tussen de symbool positie en de pijl.
- De speler moet een score kunnen zien
- De speler moet een voorbeeld sequentie kunnen zien van het "Simon" spel
- De tafel moet eender waar buiten op de technologiecampus gent functioneren
- Spel moet op duidelijke manier gespeeld kunnen worden zonder veel extra uitleg

Andere technische vereisten

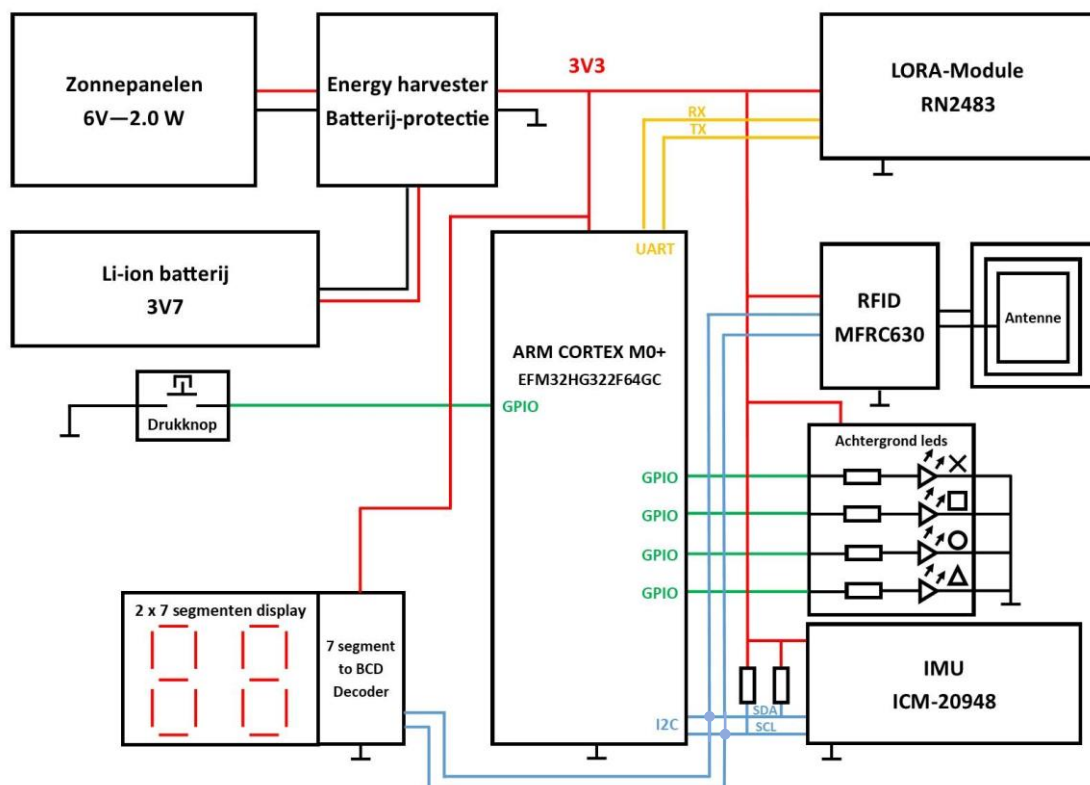
- Retrofitten van een bestaande tafel die buiten staat, met in het midden een draaiende schijf
- Waterdicht IP46 (bestendig tegen stortbuien)
- Bestand tegen rukwinden van 100 km/u
- Bestand tegen buitentemperaturen in alle seizoenen (-15 tot 45°C)
- Gewicht van maximum 5 kg
- Moet werken in omstandigheden waarbij de zonnestraling volgende waarden heeft in België: Gemiddeld 530 Wh/m^2 in december en 5070 Wh/m^2 in juni, gemeten op een horizontaal oppervlak

Niet-technische vereisten

- Esthetisch mooi verwerkt in de voorziene tafel
- Anti diefstal
- Wegneembaar/afneembaar
- Kostprijs van maximum € 100
- Deadline werkend prototype: 7 mei 2020
- Er mag geen schade aangebracht worden aan de tafel (niet boren, zagen, schijven, frezen, ...)
- De speler moet het spel kunnen spelen met een goede ergonomische houding, al zittend op de bank bij de tafel

2.2 Systeemarchitectuur

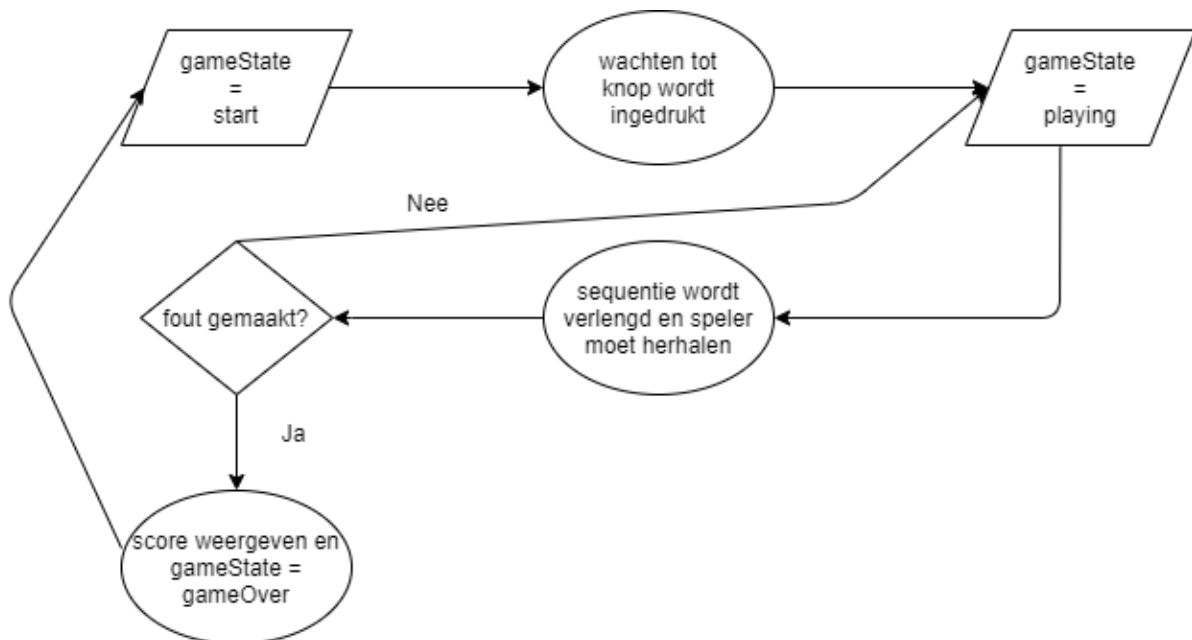
Onderstaande afbeelding (Figuur 2) geeft de elektronische architectuur van ons systeem weer. Links bovenaan ziet u de drie componenten van de energievoorziening. Aan de hand van het zonnepaneel wordt een Li-ion batterij opgeladen. Rechts bovenaan ziet u de LoRa-module die aangestuurd wordt via UART. Daaronder ziet u de RFID module en de bijhorende antenne om de studentenkaart in te lezen. Deze module wordt aangestuurd via I²C, net zoals de 7-segment displays en de Inertial Measurement Unit (kompas + accelerometer). In het midden ziet u de speler interactie componenten, namelijk de drukknop en de verschillende achtergrond LEDs van de symbolen. In ons systeem primeert energiezuinigheid boven rekenkracht en (overbodige) functionaliteiten. Daarom hebben we voor de aansturing van het geheel gekozen voor een ARM Cortex M0+ microcontroller.



Figuur 2: Systeemarchitectuur

3 Codering van het spel

Hierna volgt wat uitleg over de code die geschreven werd om het geheugenspelletje te implementeren. U kan de volledige source-code, voorzien van wat commentaar, terugvinden op GitHub (zie referenties). Gezien de omstandigheden waren er wat beperkingen waardoor ik een versie heb gemaakt die werkt met 4 LEDs en 4 knoppen.



Figuur 3: Flowchart code spel

De flowchart geeft de algemene gedachtegang van het programma weer. Ook hier is er weer gebruik gemaakt van enkele zelf geschreven libraries die toelaten enkele functies makkelijk opnieuw te gebruiken. Als u de code er even bij haalt, kan u zien dat we dus vertrekken met 2 arrays: een array voor de knoppen en een array voor de LEDs. De belangrijkste functies in volgorde waarin ze gebruikt worden in de code zijn:

- `waitToStart();`

We controleren via de for-loop de toestand van elke knop (ingedrukt of niet). Hierbij wordt elke LED even aan gezet ter controle van hun werking. De speler heeft een bepaalde tijd om een knop in te drukken en het spel te starten wat ons naar de volgende functie brengt. Indien de voorwaarde voor de tweede if-statement vervuld is, wordt er overgegaan naar playing als gameState.

- `waitForButton();`

Er wordt gebruik gemaakt van 2 verschillende while-loops. Bij de eerste wordt gecontroleerd of er een knop is ingedrukt. Natuurlijk moeten we er op letten dat de knop daarna wel degelijk is los gelaten door de speler vooraleer verder te gaan. Dit wordt gerealiseerd door de tweede loop. Deze functie returnt een integer die aangeeft of er gedrukt is of niet.

- `flashLeds();`

Telkens we deze functie doorlopen, wordt onze sequence array aangevuld met 1 getal. De getallen in deze array worden gebruikt als index in de arrays voor de knoppen en LEDs. Vandaar dat er telkens een getal van 0-3 wordt gegenereerd. Hierna moet de speler de sequentie dan herhalen.

- `inputSequence();`

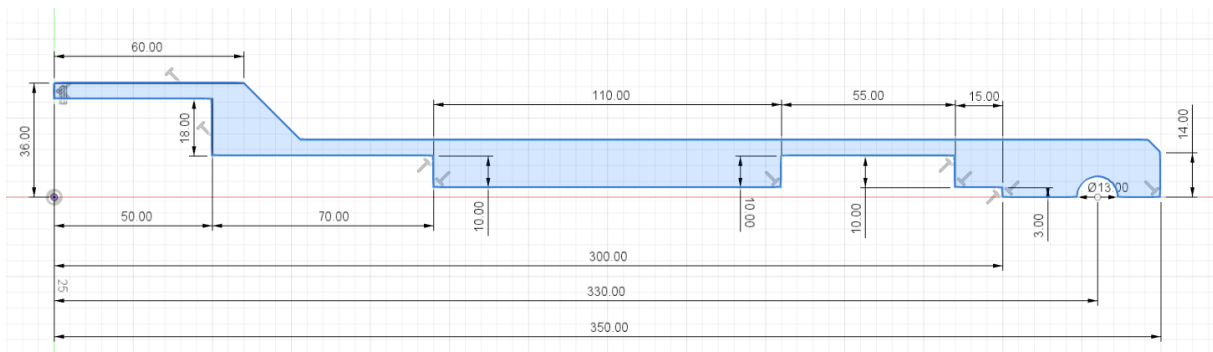
De speler krijgt een zekere tijd om de sequentie succesvol te herhalen. Indien hij slaagt, wordt er terug gekeerd naar `flashLeds()`. Indien er een fout wordt gemaakt gaan we terug naar start als `gameState` en de score zou weergegeven moeten worden op het display. Alle LEDs knipperen eens om aan te geven dat het spel is afgelopen.

De codering voor de displays heb ik helaas niet kunnen afwerken gezien de omstandigheden. Ik was eerst begonnen aan een versie voor 7-segment displays die dus iets meer straight-forward zijn om te gebruiken. Er zijn voor de door ons gebruikte displays nochtans volledige libraries voorhanden met functies die makkelijk te gebruiken zijn om dan verschillende dingen weer te geven. Helaas zijn deze bedoeld voor Arduino. Ik ben er dus niet in geslaagd deze succesvol te vertalen naar een door ons bruikbare library. Dit stuk code stond normaal nog bij `giveScore()`.

4 Behuizing

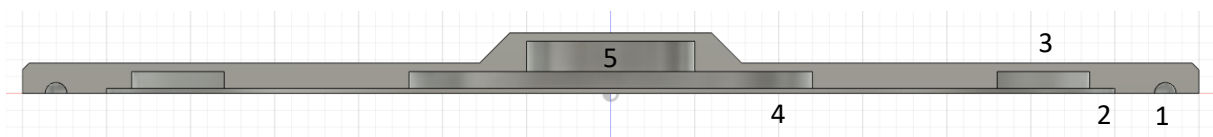
De grootste mechanische uitdaging van het project was het ontwerpen en maken van de behuizing, met andere woorden een nieuwe schijf waarin het spel geïntegreerd is. De keuze om alles te integreren in de roterend schijf bood vele voordelen aan zoals dat alle elektronica op één plaats gemonteerd zit, er geen overgang moet zijn tussen roterende delen en vaste delen, er geen aanpassingen moeten gemaakt worden aan de tafel zelf en dat dit zorgt voor een weersbestendig geheel. Echter kwam dit ook met het nadelen van te weinig inbouwruimte waardoor de vorm en afmetingen licht moesten aangepast worden.

De originele ronde schijf heeft een diameter van 700 mm en dikte van 14 mm. Om elektronica in de schijf te kunnen monteren moeten er secties uitgefreesd worden. Om een voldoende grote inbouwdiepte te verkrijgen werd de dikte verhoogd naar 18 mm met in het midden nog een extra verdikking naar 36 mm, later hier meer over. Op Figuur 4 is de technische tekening te zien en op Figuur 5 de doorsnede van de schijf die we vanaf nu de behuizing noemen.



Figuur 4: Technische tekening behuizing (schijf)

Op beide tekeningen is te zien dat aan de onderkant verschillende secties uitgefreesd zijn. De eerste uitsparing (nr. 1) dient om de wieltjes, aanwezig op de tafel, te laten in geleiden. Dit is hoe de originele schijf kan roteren dus werd dit overgenomen. Vervolgens komt een verdieping van 3 mm (nr. 2) over de volledige onderkant van de behuizing. Deze dient om later een plaat op te monteren zodat alle elektronica spatwaterdicht kan afgeschermd worden. Hierna volgt een verdieping van 10 mm (nr.3). Deze dient om de achtergrondverlichting van de symbolen in te monteren maar later hier meer over. Als laatste zijn er uitsparingen nr. 4 en nr. 5. Deze dienen om alle overige hardware in te monteren. Nr. 4 zal de mogelijkheid bieden voor printplaten, randcomponenten en draden in kwijt te kunnen. Uitsparing nr. 5 zal dan dienen om de drukknop, display en batterij in te kunnen monteren. Deze hebben een zekere inbouwdiepte nodig dus werd in de midden een verdikking geplaatst van 36 mm. Deze verdikking zorgt ook voor dat de drukknop en display geaccentueerd worden.



Figuur 5: Doorsnede schijf

Ook zijn er nog extra uitsparingen die sectie nummer 3 en 4/5 met elkaar verbinden. Dit zijn kanalen waar draden kunnen lopen naar de verschillende achtergrond LEDs voor de symbolen. Als laatste is er ook nog een stuk aan de bovenkant uitgefreesd waar het zonnepaneel zich bevindt zodat deze ook verwerkt zit in de behuizing. (zie foto's 4, 5 en 6 van het resultaat)

Nu de specifieke vorm bepaald is kan bekeken worden welke materialen in aanmerking komen om de behuizing te maken. Hiervoor werden offertes en raad gevraagd bij de groothandel. Er kwamen drie mogelijke materialen in aanmerking:

- Trespa 20 mm € 139,55 / m²
- Tricoya 18 mm € 62,41 / m²
- MDF (groen) 18 mm € 13,41 / m²

Het eerste materiaal, Trespa, is het materiaal dat gebruikt wordt bij de originele draaischijf. Deze platen zijn gemaakt uit houtvezels en hars waardoor deze uitermate geschikt zijn voor buitentoepassingen. Een tweede materiaal, Tricoya, is een variant van MDF dat gemaakt wordt om te gebruiken in zeer vochtige omgevingen en dus ook geschikt is voor buitenomgeving. Doordat beide materialen gemaakt zijn voor buitentoepassingen komt dit ook met een prijskaartje. In een finaal product zou voor een duurzaam materiaal moeten gekozen worden zodat de behuizing bestand is tegen het buitenweer. Echter hier, met het oog op een prototype, werd gekozen voor MDF vanwege de prijs en de gemakkelijke bewerkbaarheid.

Een laatste gegeven waar rekening moest met gehouden worden is anti-diefstalaspect. Doordat de schijf/behuizing los ligt op de wieltjes van de tafel kan deze er zomaar uitgehaald worden. Om dit probleem op te lossen zijn twee maatregelen getroffen. Ten eerste kan aan de plaat die de behuizing aan de onderkant afdicht een kettingwartel (zie Figuur 6 links) voorzien worden. Deze wartel zorgt ervoor dat de behuizing kan vrij blijven draaien langs de ene kant en langs de andere kant met ketting kan verbonden worden het frame van de tafel. Ten tweede kunnen anti-diefstal vijzen, te zien in Figuur 6 (rechts), gebruikt worden om de plaat te bevestigen aan de behuizing. Dit zal voorkomen dat met een eenvoudige schroevendraaier het geheel kan gedemonteerd worden en alsnog kan meegenomen worden.

Een laatste gegeven waar rekening moest met gehouden worden is anti-diefstalaspect. Doordat de schijf/behuizing los ligt op de wieltjes van de tafel kan deze er zomaar uitgehaald worden. Om dit probleem op te lossen kan aan de plaat die de behuizing aan de onderkant afdicht een kettingwartel (zie Figuur 6) voorzien worden. Deze wartel zorgt ervoor dat de behuizing kan vrij blijven draaien langs de ene kant en langs de andere kant met ketting kan verbonden worden het frame van de tafel.

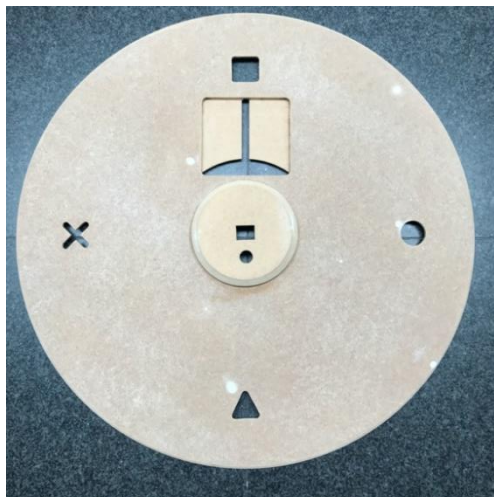


Figuur 6: Kettingwartel (links), anti-diefstal vijz (rechts)

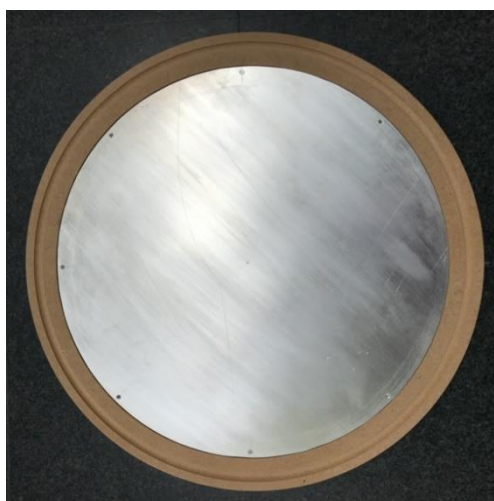
Op onderstaande foto's is het eindresultaat te zien van de behuizing.



Figuur 7: Onderzijde behuizing zonder afdichtplaat



Figuur 8: Bovenzijde behuizing



Figuur 9: Onderzijde behuizing met afdichtplaat

* Opmerking: De platen zijn gemaakt uit overschotten MDF die beschikbaar waren tijdens de corona periode, de witte cirkels zijn schroefgaten die opgevuld werden met filler.

5 Speler interactie componenten

Een eerste belangrijk gegeven van het spel is een manier om de sequentie van symbolen duidelijk te maken. Hier ging de voorkeur naar een LED met diffuser die het volledige symbool zou laten oplichten. Op de markt was een witte backlight module terug te vinden met afmetingen 45 mm x 86 mm gemaakt door Adafruit die zeer geschikt leek. Hierbij was het echter belangrijk om te testen of deze fel genoeg zijn om zichtbaar te zijn in het zonlicht. Dit is niet getest kunnen worden doordat de LEDs niet geleverd werden door coronatoestand.

In onderstaande Figuur 10 zijn de vier figuren terug te vinden die zouden moeten opgelicht worden. Deze hebben afmetingen van 40 mm op 40 mm met afgeschuinde kanten waarvan de radius 5 mm bedraagt. Het was voorzien om de figuren ook uit te lazieren in plexiglas zodat de behuizing waterdicht kon gemaakt worden en men toch de LEDs zou zien wanneer deze oplichten. Opnieuw door de coronamaatregeling is dit niet gelukt om dit te maken.



Figuur 10: Gegraveerde symbolen

Geïnstalleerd zou dit de volgende opbouw geven te zien op Figuur 11. Hier is de doorsnede getekend sectie nr. 3 met alle onderdelen.



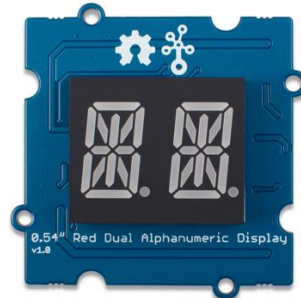
Figuur 11: Opbouw achtergrond LEDs

Een volgend belangrijk gegeven is hoe een symbool kan bevestigd worden voor een reeks. Hier is geopteerd om dit te doen via drukknop (zie figuur 12) omdat dit vrij moeilijk leek om dit ook te doen via de IMU. Dit zou bijvoorbeeld problemen geven wanneer tweemaal hetzelfde symbool zou bevestigd moeten worden. Bovendien kan deze knop naast de IMU ook gebruikt worden om de microcontroller uit slaap te halen en om het spel effectief te starten. Hiervoor werd een eenvoudig en goedkoop model gezocht die waterdicht is en een lage inbouwdiepte heeft.



Figuur 12: Drukknop

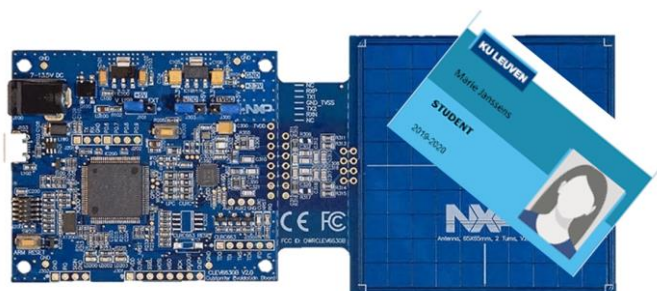
Een voorlaatste interactie met de speler is om de score aan te geven. Er werd geopteerd om dit te doen met een 2x7-segmeten display. Hier moest ook aandacht gegeven worden aan de lichtsterkte van de LEDs. Deze moeten ook zichtbaar zijn in buiten in de zon. Hiervoor zijn we te rade gegaan bij een elektronicawinkel die ons in contact heeft gesteld met mensen van Velleman. Deze konden ons vertellen dat er moest gekozen worden voor displays met een lichtintensiteit van minstens 15000 μcd . Met deze informatie zijn we uitgekomen op onderstaande module, te zien op Figuur 13.



Figuur 13: Dual alfanumeriek display

Deze module heeft twee alfanumerieke displays met een voldoende grote lichtintensiteit. Ook is er een decoder voorzien die werkt doormiddel van I2C op 3V3 werkspanning. Er kon niet gekozen worden voor een BCD-to-7segment decoder, die origineel in het ontwerp gepland stond, omdat deze niet beschikbaar is in een 3V3 voedingsspanning. Het rechtstreeks aansturen van displays was ook geen optie omdat er te weinig GPIO-pinnen ter beschikking zijn. De felle LEDs zelf worden gevoed met een hogere spanning dus is er ook een boost-converter voorzien op de module.

Ten slotte hebben we onderstaande RFID module (Figuur 14) toegevoegd aan de tafel. Wanneer een speler een spel start legt hij zijn studenten/personeelskaart op de RFID module en wordt zijn ID uitgelezen. Wanneer de speler klaar is met zijn spel zal de score aan deze ID gekoppeld worden en wordt deze combinatie doorgestuurd naar de *cloud*. De studentenkaarten die gebruikt worden op de KULeuven zijn volgens de ISO/IEC 14443A standaard en werken op 13,56 MHz. Onderstaande module bevat een CLRC663 chip die compatibel is met deze standaard. De keuze voor deze CLRC663 plus module is gemaakt omdat ze beter geschikt is voor werking op een batterij. Zo ligt de werkspanning tussen 2,5V en 5V, wat lager is dan zijn voorganger, en valt onze systeemspanning van 3,3V hier perfect in. Verder gebeurt de communicatie tussen de EFM32 en deze module via de I²C standaard.



Figuur 14: RFID module

6 Sensoren

Voor de sensoren werden een accelerometer en kompas geselecteerd als meest geschikt voor deze toepassing. Er bestaan IMU's (Inertial Measurement Units) met als doel bewegingen te meten. Voor de kleine package en de bijna identieke prijs van het kopen van een accelerometer en kompas apart, werd hier voor de IMU gekozen. Het voordeel van de IMU is dat deze ook een gyroscoop aan boord heeft (waar in de toekomst eventueel ook nog gebruik van zou kunnen gemaakt worden). De IMU die geselecteerd werd was de ICM20948 vanwege zijn zeer laag vermogen verbruik, talrijke instelmogelijkheden en goede nauwkeurigheid.

Deze chip werkt op 1.8 V en verbruikt (onder andere daardoor) slechts 2.5 mW. Hierdoor zal er in het hardware design ook level shifting en een spanningsregelaar voorzien worden. De chip beschikt over een accelerometer met programmeerbare FSR van $\pm 2g$, $\pm 4g$, $\pm 8g$ en $\pm 16g$ en een kompas met een groot bereik van $\pm 4900 \mu T$. Deze waarden worden omgezet door on-chip 16-bit ADCs. De chip beschikt ook over een digitale temperatuursensor om temperatuurcorrecties door te voeren. Deze kan ook uitgelezen worden en mogelijks gebruikt worden om bijvoorbeeld de buitentemperatuur te monitoren en mee door te sturen.

6.1 Software

Om de samenvoeging van de verschillende onderdelen in dit project vlot te laten verlopen werden een aantal functies geschreven om sensorwaarden uit te lezen. Deze werden ondergebracht in een library. Door deze library te includen, de juiste pinnen aan te passen en de voorgestelde functies te gebruiken kan op een minimum van tijd geïnterfaced worden met de sensor.

Functies die geschreven werden zijn:

- `ICM_20948_Init();`

Een functie voor het initialiseren van de ICM20948 sensor, dit bevat zowel de gyroscoop, accelerometer als het kompas. Hier wordt ook instellingen van de I2C interface correct gezet en de communicatie wordt gestart. Er wordt nagegaan of de sensor wel degelijk verbonden is door het "Who am I" register uit te lezen. Ook sample rates worden hier ingesteld. Het kompas (dat een aparte chip is in de ICM20948) wordt door middel van een I2C bypass multiplexer rechtstreeks verbonden met de I2C bus.

- `ICM_20948_accelDataRead(data.ICM_20948_accel);`

Een functie om de accelerometer waarden uit te lezen in drie dimensies (x, y en z). Voor het uitlezen van alle waarden wordt er gebruik gemaakt van een "struct" (hier genaamd: data), een soort structure die alle uitgelezen waarden bijhoudt.

- `ICM_20948_magDataRead(data.ICM_20948_magn);`

Een functie die de kompas waarden uitleest en omzet naar μT , ook in drie dimensies. Dit gebeurt aan 100 Hz.

- `ICM_20948_magn_to_angle(data.ICM_20948_magn, data.angle);`

Een functie die de "raw" kompas waarden omzet naar een hoek in graden. Deze hoek kan variëren tussen -180° en $+180^\circ$ en kan gebruikt worden om de positie van de schijf te bepalen.

- `ICM_20948_calibrate_mag(data.offset, data.scale);`

Een functie die het kompas kalibreert. Deze waarden worden opgeslagen en worden na kalibratie ook mee in rekening gebracht in de functie: “ICM_20948_magDataRead” en “ICM_20948_magn_to_angle”. Er is dus een volledige integratie in de geschreven library en er zijn achteraf geen correcties van offsets meer nodig.

- `ICM_20948_wakeOnMotionITEnable(true, 50, 50);`

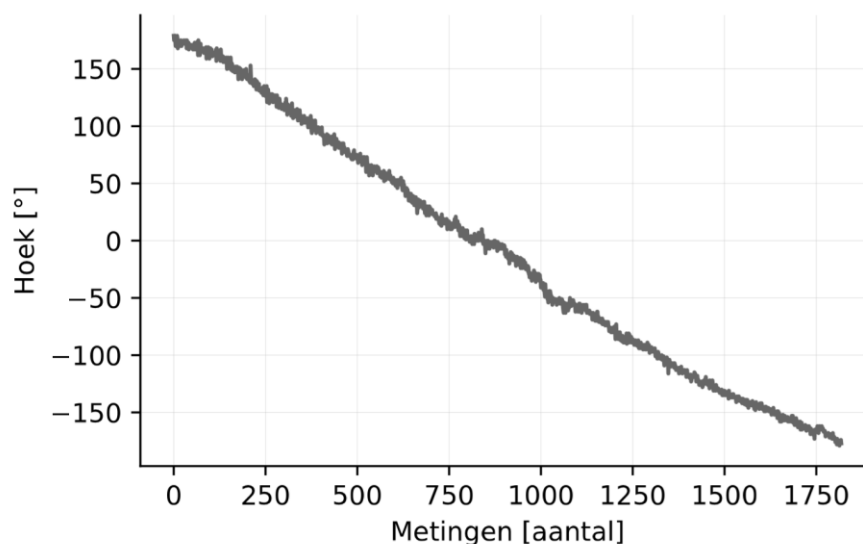
Een functie die alle sensoren uit zet en enkel de accelerometer aanzet in low power mode. Het tweede argument bepaald de sample rate en het derde argument bepaald de threshold in mG's. De sample rate wordt automatisch omgezet naar een waarde die in het register van de sample rate divider past. (bijvoorbeeld 50 Hz wordt in de praktijk 48.9 Hz) Wanneer de accelerometer een verandering meet groter dan deze threshold zal deze een interrupt genereren en het systeem wakker maken.

De volledige source-code is terug te vinden op GitHub. (Zie referenties)

6.2 Kompas

Er werd gekozen voor een kompas om de oriëntatie van de draaiende schijf te bepalen. Aangezien het aards magnetisch veld zo goed als constant is, kunnen we dit gebruiken als ‘ideale’ referentie voor de positionering. Het kompas dat hiervoor gekozen werd is de ak09916 (geïntegreerd in de ICM20948). Dit kompas kan het aards magnetisch veld meten aan 100 Hz, wat zeker snel genoeg is om het spel te kunnen spelen.

In Figuur 15 is een meting te vinden die de correcte werking van het kompas illustreert. In deze meting werd het kompas 360 graden rond zijn as gedraaid. Dit werd gedaan door deze handmatig vast te nemen, door de onregelmatigheden in de draaibeweging is de grafiek niet perfect. Wat we hiermee willen aantonen is dat de nauwkeurigheid zeker voldoende is om vier kwadranten te onderscheiden van elkaar.



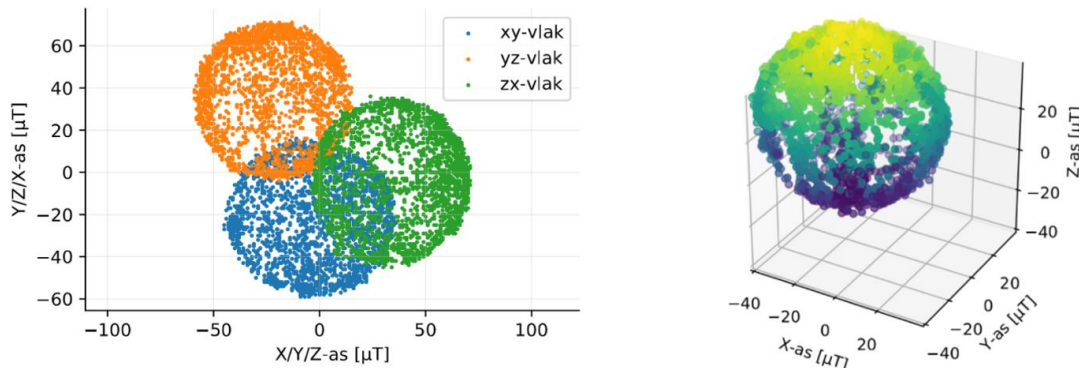
Figuur 15: Meting kompas 360° rotatie

6.3 Kalibratie

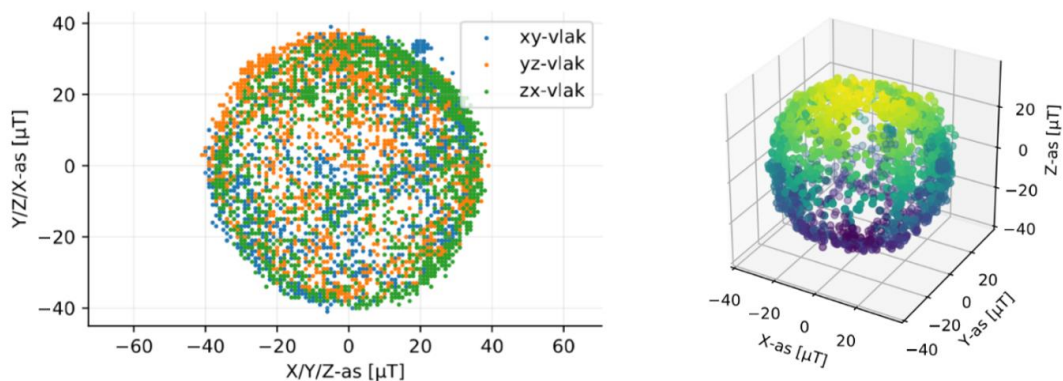
Om nauwkeurige metingen te bekomen is kalibratie van sensoren zeker een meerwaarde. Vooral bij het kompas is dit zeker aan te raden daar deze sensor veel last heeft van fouten. Zo bestaan er twee types vervormingen van de data: harde distorties en zachte distorties.

Harde distorties worden veroorzaakt door objecten die zelf een magnetisch veld creëren, bijvoorbeeld een magneet in een luidspreker. Dit type distorties blijft steeds op eenzelfde plaats ten opzichte van het referentiesysteem van het kompas. Hierdoor krijgt men dus een constante offset. Corrigeren voor harde distorties gebeurt door de offset in x, y en z richting te bepalen en deze in rekening te brengen bij de metingen. De offset van de verschillende assen is goed te zien op Figuur 16 (links) **Fout! Verwijzingsbron niet gevonden..** Figuur 17 (links) geeft de sfeer weer na kalibratie. De offsets zijn ook in 3D weergegeven in Figuur 16 (rechts) voor kalibratie en Figuur 17 (rechts) na kalibratie.

Zachte distorties zullen het magnetisch veld op een bepaalde manier vervormen. Bijvoorbeeld zal de x-component meer verzwakt worden dan de y- en de z-component. Dit gebeurt typisch door een materiaal dat een magnetisch veld vervormd maar er zelf geen maakt. Een voorbeeld hiervan is ijzer. Hiervoor corrigeren is moeilijker, is meer rekenintensief en vereist een 3x3 transformatie matrix. In software werd dit geïmplementeerd door gedurende enkele seconden (vb. 15 seconden) metingen te doen terwijl men de sensor in alle mogelijke richtingen draait. Hierna wordt voor elke as (x, y en z) de maximale en minimale waarde bepaald. Deze worden genormaliseerd om overeen te komen met een cirkel. De som en het verschil van deze waarden wordt berekend. De offset wordt bepaald aan de hand van de helft van het verschil tussen minimalen maximale waarde. Voor de schaal correctie wordt eerst het gemiddelde van de verschillen van de drie assen berekend en deze gedeeld door het verschil van elke as.



Figuur 16: Assen kompas voor kalibratie (links), 3D plot voor kalibratie (rechts)



Figuur 17: Assen kompas na kalibratie (links), 3D plot na kalibratie (rechts)

6.4 Accelerometer

Om detectie van aanwezigheden te doen wordt er gebruik gemaakt van een accelerometer. Dit type sensor meet versnellingen. Voor de activiteit detectie zou ook voor een gyroscoop kunnen gekozen worden. Dit werd echter niet gedaan omdat een gyroscoop typisch veel meer (enkele mA) verbruikt dan een accelerometer (enkele tientallen μA in deze sensor). De accelerometer zal een interrupt pin hoog zetten wanneer beweging gedetecteerd wordt. Hierdoor zal de microcontroller wakker gemaakt worden, zullen de LED's gaan branden en kan het spel beginnen.

6.5 Energiemetingen

Door het niet ontvangen te hebben van de ontworpen PCB is het niet mogelijk geweest energiemetingen van deze sensoren te verzamelen. De ICM20948 werd uitgetest en geprogrammeerd aan de hand van een breakout bordje van SparkFun. Hierop werd een LDO geïntegreerd met een zeer grote I_q (lekstroom). Hierdoor zouden geen representatieve metingen verkregen worden aangezien we op de PCB zouden gebruik maken van een LDO met een I_q van $1\ \mu\text{A}$.

Enkele berekeningen werden wel gedaan, waarvan een voorbeeld:

In de datasheet is een verbruik van $36.1\ \mu\text{A}$ terug te vinden bij een sample frequentie van $48.9\ \text{Hz}$ bij de accelerometer.

LDO (TPS7A05): $I_q = 1\ \mu\text{A}$ (typ.), $3\ \mu\text{A}$ (max.)

$$P = (3\ \mu\text{A} + 36.1\ \mu\text{A}) * 3.3\ \text{V} = 129.03\ \mu\text{W}$$

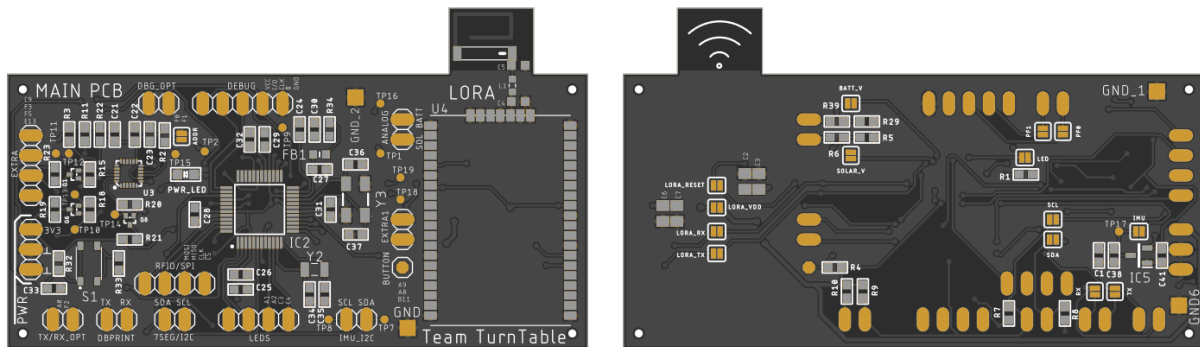
6.6 Design PCB

In Figuur 18 is de ontworpen PCB te zien. Hier werden enkele ontwerpkeuzes gemaakt. Een eerste keuze is het goed scheiden van de draadloze communicatie en het microcontroller en sensor-gedeelte. De antenne werd ook naar buiten gebracht om zo een beter bereik te kunnen garanderen. Er werd gekozen voor een antenne op de PCB en er was geen nood aan een antenne die signalen verder kon versturen aangezien we op de campus een LoRa gateway hebben en hier zeer dicht bij zitten.

De microcontroller wordt gevoed op $3.3\ \text{V}$. De $3.3\ \text{V}$ komt van het energy harvesting bordje. De sensoren werden ook geïntegreerd op deze PCB. De IMU die gebruikt wordt, werkt op $1.8\ \text{V}$ en wordt voorzien door een LDO. Ook de communicatie met de IMU werkt op deze lagere spanning, hiervoor werd ook de gepaste level shifting voorzien. De LoRa communicatie werkt op $3.3\ \text{V}$ en aangezien deze veel stroom kan trekken (typisch $\pm 40\text{mA}$) kon dit niet via een pin uit de microcontroller gevoed worden. Deze module werd rechtstreeks aangesloten op de voedingsspanning. Ook werden de traces hiervoor iets breder genomen.

Op deze PCB werd ervoor gekozen om alle pinnen naar buiten te brengen. Zo kon het gebruikt worden als een soort development bordje voor het prototype. Ook werden tal van testpunten en jumpers voorzien om het debuggen te vergemakkelijken.

Het resultaat is een PCB met afmetingen: 73.9 mm x 43.4 mm, wat zeker klein genoeg is om in de roterende schijf te passen. Alle schema's en design files van deze PCB zijn terug te vinden op GitHub (zie referenties).



Figuur 18: Main PCB voorkant (links), achterkant (rechts)

7 Energy Harvesting

Energie voorziening is een onmisbaar deel bij een embedded system. Aangezien ons embedded system zich buiten bevindt kan een zonnepaneel gebruikt worden als energiebron. Het is belangrijk op voorhand een schatting te maken om de energy harvesting componenten te kunnen dimensioneren. Bovendien kunnen alle onderdelen gevoed worden met 3,3 V wat het ontwerp eenvoudiger maakt.

7.1 Assumpties voor de berekeningen

Dit embedded system heeft nogal een onvoorspelbaar karakter op vlak van energie noden. De nodige energie hangt af van enkele factoren die kunnen veranderen, zoals de speelduur, het aantal spelletjes per dag, de hoeveelheid zon per dag, het aantal highscores per dag,

Om hier toch een schatting te kunnen maken worden enkele assumpties opgesteld.

- Het spel moet 3 uur (na elkaar) speelbaar zijn per dag
- Er kunnen 30 spelletjes in 3 uur gespeeld worden (6 min per spel inclusief aanmelding)
- Per drie uur kan de score in de top drie 2x verbroken zijn
- Tijd nodig per dag om de highscore te zenden via LoRa: 1 s
(gemeten zendtijd 1 pakket : 63,6 ms)
- De 7 segment displays flikkeren gedurende ieder spelletje ongeveer 20 seconden
- Slechts één backlight kan tegelijk oplichten
- De RFID aanmelding kan 2 minuten duren (zo wordt tijd gegeven om de studentenkaart uit te halen)
- De accelerometer dient als interrupt om het systeem uit sleepmode te halen

7.2 Berekeningen

Met behulp van de datasheet gegevens van de gekende componenten en bovenstaande assumpties wordt een berekening gemaakt waarbij het energieverbruik per dag bepaald wordt die nodig is om te opereren in actieve mode. Aangezien dit een schatting is worden nog extra verliezen bijgerekend die mogelijks afkomstig kunnen zijn van andere componenten die zich in het schema bevinden. Zo worden warmteverliezen, verliesstromen, sleepmodes van RFID en LoRa module, ... ook in rekening gebracht. Deze berekening is terug te vinden in **Fout! Verwijzingsbron niet gevonden.. In Fout! Verwijzingsbron niet gevonden.** kunnen de berekeningen voor de sleepmode voor die ene dag (21 uur) terug gevonden worden.

In actieve mode is het energieverbruik per dag gelijk aan 1839,9 J. Het maximaal te leveren vermogen (indien alles aan staat) bedraagt 880 mW. De maximaal te leveren stroom indien alles aan staat bedraagt 380,5 mA.

In sleep mode is het energieverbruik per dag gelijk aan 674 J. Het totale energieverbruik per dag bedraagt dus 2512,9 J.

	IMU	LDO (IMU)	RFID (zenden)	LoRa (zenden)	EFM32	2x 7-segment displays	4x Backlight LEDs	Verliezen bij andere componenten
Stroomverbruik (mA)	3,11	0,001	20	38,9	3,5	2 x 7 x 20 mA = 280 mA	4 x 20 mA	15
Werkspanning (V)	1,8	1,5	3,3	3,3	3,3	2	3	3,3
Vermogen verbruik (mW)	5,598	0,0015	66	128,37	11,55	560	4 x 60 mW	50
Tijd actief per dag (uur)	3	3	30x0,033	0,000277	3	0,17	2,5	3
Tijd actief per dag (seconden)	10800	10800	3600	1	10800	600	9000	10800
Energie verbruik (J)	60,458	0,016	237,600	0,128	124,740	336,000	540,000	540,000

Tabel 1: Berekeningen actieve mode (3 uur per dag)

	IMU	LDO (IMU)	RFID (zenden)	LoRa (zenden)	EFM32	2x 7-segment displays	4x Backlight LEDs	Verliezen bij andere componenten
Stroomverbruik (µA)	3100	1	0,04	1,6	2	0	0	10
Werkspanning (V)	1,8	1,5	3,3	3,3	3,3	3,3	3,3	3300
Vermogen verbruik (µW)	5598	1,5	0,132	5,28	6,6	0	0	33
Tijd actief per dag (uur)	21	21	21	21	21	21	21	21
Tijd actief per dag (seconden)	75600	75600	75600	75600	75600	75600	75600	75600
Energie verbruik (J)	423,209	0,113	0,010	0,399	0,499	0,000	0,000	249,480

Tabel 2: Berekeningen sleep mode (21 uur per dag)

Indien een autonome werking (zonder opladen via het zonnepaneel) van 10 dagen gewenst is kan volgende berekening uitgevoerd worden:

Nodige energie voor 10 dagen = nodige energie/ dag · 10 dagen

$$2512,9 \frac{\text{J}}{\text{dag}} \cdot 10 \text{ dagen} = 25129 \text{ J}$$

De nodige capaciteit van de Lithium-ion batterij met een nominale spanning van 3,7 V is dan:

$$\text{capaciteit batterij (mAh)} = \frac{E}{3600 \cdot U} \cdot 1000$$

$$\frac{25129}{3600 \cdot 3,7} \cdot 1000 = 1886 \text{ mAh}$$

Er wordt gekozen voor een batterij (Panasonic NCR18650PF) met een capaciteit van 2900 mAh, enerzijds omdat deze in voorraad is bij één van de groepsleden, anderzijds omdat zo voldoende reserve genomen wordt indien de batterij in verloop van tijd afzwakt of wanneer deze gedurende lange tijd niet wordt opgeladen omdat het zonnepaneel bedekt is of dergelijke. Bovendien is de energie berekening een schatting en worden enkele onvoorspelbare assumpties vastgelegd die zeer sterk kunnen wijzigen in werkelijkheid.

De batterij kan gebruikt worden bij buitentemperaturen en wordt als veilig beschreven aangezien deze ook gebruikt kan worden voor medische toestellen. Ook de zelfontlading is voldoende klein.

7.3 Zonnepaneel

Bij het maken van een keuze van het zonnepaneel zijn vooral factoren zoals kras- en waterbestendigheid belangrijke factoren aangezien dit paneel op een buiten tafel ligt waar studenten eten. Het is belangrijk dat een plateau waar eten op staat geen schade toebrengt aan het zonnepaneel wanneer een student deze verschuift over het zonnepaneel.

Het gekozen zonnepaneel bezit een IP67 waarde waardoor deze ideaal is voor deze toepassing. De piekspanning bedraagt 6,5 V, de open klem spanning bedraagt 7,7 V en het rendement is gelijk aan 19 % waardoor met behulp van een oppervlak van 0,0143 m² een maximaal vermogen van 2 W geleverd kan worden.

Bij bewolkt weer in de winter kan de ingestraalde zonne-energie 0,5 kWh/m² bedragen (Zonne-energie gids, n.d.). Voor deze situatie wordt bekeken of dit paneel voldoet aan de gewenste voorwaarden.

$$E = \frac{\text{instraling}}{\text{m}^2} \cdot \text{oppervlakte} \cdot \text{rendement}$$

$$E/\text{dag} = 0,5 \frac{\text{kWh}}{\text{m}^2} \cdot 0,0143 \text{ m}^2 \cdot 0,19 \cdot 5 \frac{\text{zonne uren}}{\text{dag}} = 6,75 \text{ Wh} = 24300 \text{ J}$$

Deze bepaalde energiewaarde is ruim voldoende aangezien 25129J nodig is om het gehele systeem gedurende 10 dagen autonoom te laten werken. Er kan dus snel genoeg opgeladen worden. Bovendien ligt het zonnepaneel plat en kan het bedekt zijn met bladen, sneeuw, ... waardoor deze overschot gewenst is aangezien de opbrengst een stuk lager zal liggen.

7.4 Supercap

Om zoveel mogelijk energie afkomstig van het zonlicht op te vangen wordt tussen de batterij en het zonnepaneel een supercap voorzien met een capaciteit van 3,3F en een maximale spanning van 2,7V.

Een supercap kan namelijk veel sneller opladen dan een batterij. Deze energie wordt dan overgebracht naar de batterij. De mogelijke hoeveelheid opgeslagen energie in deze supercap is gelijk aan:

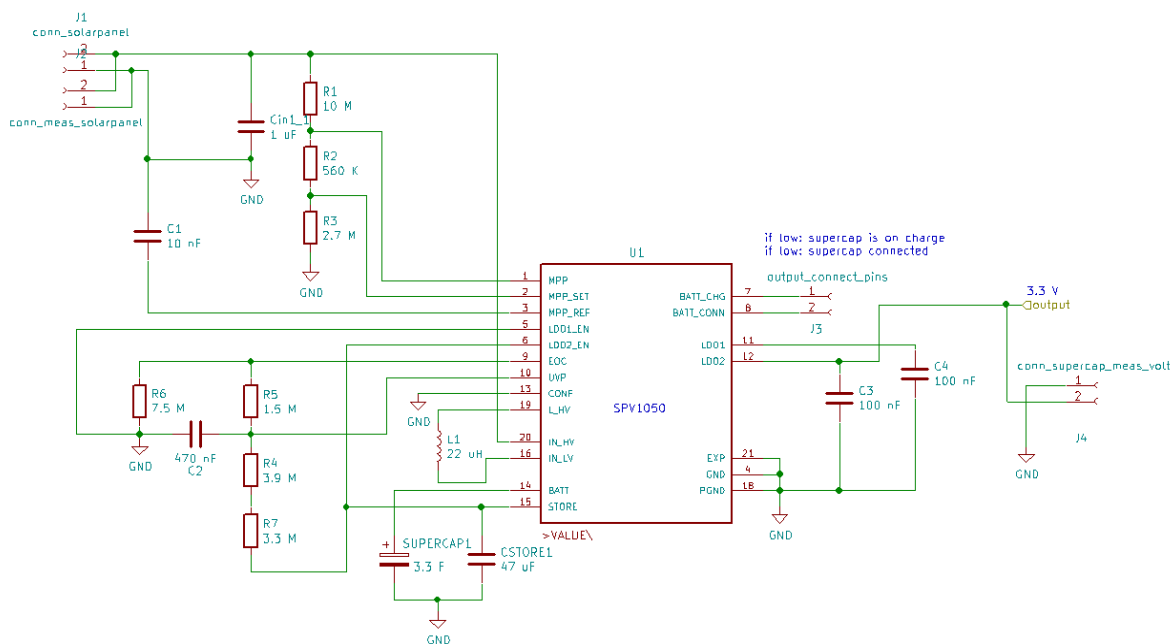
$$E = \frac{C \cdot U^2}{2} = \frac{3,3 \cdot 2,7^2}{2} = 12 \text{ J}$$

7.5 Hardware

Het energy harvesting schema bestaat uit twee delen. Enerzijds is er het deel die ervoor zorgt dat de supercap opgeladen wordt en deze beveiligd tegen een eventuele te grote spanning. Anderzijds volgt hierna een tweede circuit die de batterij oplaadt.

7.5.1 Opladen van de supercap

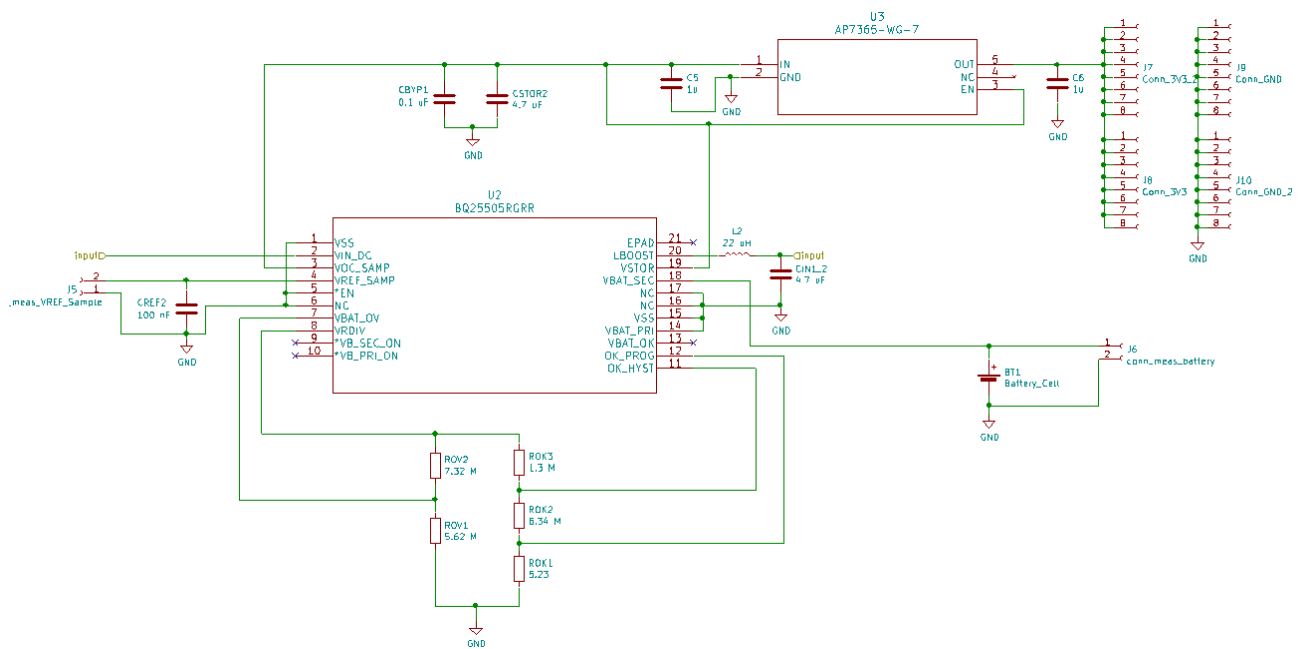
In Figuur 19 wordt het schema weergegeven die gebruikt wordt om de supercap op te laden. Met behulp van de SPV1050 wordt de energie afkomstig van het zonnepaneel opgeslagen in de supercap. Er was weinig keuze op vlak van toepasbare IC's aangezien het zonnepaneel een 'grote' spanning van 6,5 V kan genereren. Na de berekening met behulp van formules uit de datasheet van de SPV1050 worden gepaste component waarden gekozen. Zo wordt overvoltage en undervoltage protection voorzien en wordt de spanning afkomstig van het zonnepaneel zodanig afgesteld zodat de gewenste input waarden verkregen worden aan de ingang van de SPV1050. De uitgang bedraagt een 3,3 V spanning die als ingang van het volgende circuit gebruikt wordt. Deze IC wordt gevoed met de energie afkomstig van het zonnepaneel.



Figuur 19: Schema voor het opladen van de supercap

7.5.2 Opladen van de batterij

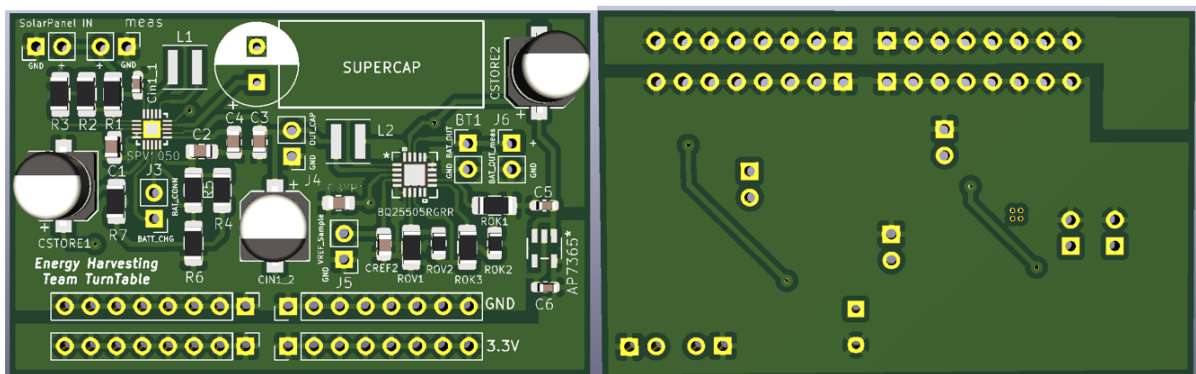
In Figuur 20 wordt het schema weergegeven die instaat voor het opladen van de batterij. De kern van dit circuit bestaat uit de BQ25505RGR. Opnieuw worden parameters ingesteld zoals de over- en undervoltage protection zodat de batterij niet stuk gaat. De AP7365-WG-7 spanningsregelaar kan voldoende stroom leveren en zorgt voor een constante 3,3 V spanning.



Figuur 20: Schema voor het opladen van de batterij

7.5.3 PCB ontwerp

Zoals weergegeven in Figuur 21 worden heel wat uitgangen voorzien zodat alle sensoren en actuatoren gevoed kunnen worden. Er wordt bewust gekozen om geen pinheaders te voorzien zodat de verbindingen gesoldeerd kunnen worden. Hierdoor is de hardware robuuster, wat gewenst is in een bewegende schijf. Verder worden meetpinnen (om te het deze hardware te testen) en verbindingspinnen voorzien om het zonnepaneel en da batterij te verbinden. De supercap wordt plat op de PCB geplaatst zodat deze minder ruimte in beslag neemt waardoor deze PCB in de schijf verwerkt kan worden. De PCB heeft een lengte van 5,7 cm op 3,5 cm.



Figuur 21: Voorkant (links) en achterkant (rechts) van de Energy Harvesting PCB

7.6 Evaluatie

Wegens de corona maatregelen is het niet mogelijk deze PCB te testen aangezien deze nog niet geleverd is. Bovendien bezit ik geen oscilloscoop ter controle. Ook bij het ontwerpen waren heel wat uitdagingen aangezien de footprints van de aanwezige componenten op school niet altijd gekend zijn vanop afstand. Gelukkig had ik foto's van de aanwezige componenten (die toen nog aanwezig waren). Ook de energie berekening kan niet gecontroleerd worden met de werkelijke meetwaarden van het volledige systeem wegens de corona maatregelen. Bovendien was dit een moeilijke opgave aangezien er zeer weinig informatie hieromtrent terug te vinden is op het internet.

8 Connectiviteit en Visualisatie

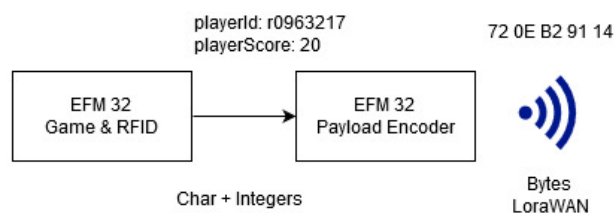
Wanneer een speler zijn spel afgerond heeft krijgt hij meteen zijn behaalde score te zien op de 7-segment displays. Deze score is slechts een korte tijd zichtbaar tussen twee spellen in, maar de tijd is ook beperkt omdat de displays grote energieverbruikers zijn. Om het systeem energiezuiniger te maken en om de competitiviteit tussen spelers te verhogen hebben we er voor gekozen de scores te visualiseren op een website. In dit hoofdstuk worden de verschillende stappen beschreven die nodig zijn voor het doorsturen (Lokaal) en visualiseren (Cloud) van de scores. De stappen worden beschreven a.d.h.v. een fictief voorbeeld waarbij speler 'r0963217' een score van 20 punten behaald heeft. In Figuur 22 (voor het lokaal gedeelte) en Figuur 25 (voor het cloud gedeelte) zijn de stappen visueel weergegeven. Boven elke stap staat de inhoud van het datapakket. Onderaan staat aangegeven in welke vorm het datapakket doorgegeven wordt aan de volgende stap.

Bestanden die in dit hoofdstuk beschreven worden:

- Lokaal:
 - o Payload encoder & LoRa zender (Simplicity Studio export):
EFM32-RN2483-LoRa-Turntable.sls
- Cloud:
 - o The Things Network payload decoder:
CloudTheThingsNetwork - Payload Decoder.js
 - o Data overdragen van TTN Data Storage naar MYSQL database:
swaggerAPI_to_mysqlDB.py
 - o Webpagina met highscores:
index.php

8.1 Lokaal

Het encoderen en verzenden van de relevante gegevens gebeurt uiteraard lokaal op de EFM32 microcontroller aanwezig in de tafel. Bij de start van het spel meldt een speler zich aan door zijn studenten/personeelskaart op de RFID lezer te leggen. Als de speler zijn spel afgerond heeft wordt de behaalde score aan zijn id gekoppeld en doorgestuurd.



Figuur 22: Connectiviteit (lokaal)

8.1.1 Payload Encoder

In Figuur 22 is te zien dat de score en id tijdelijk opgeslagen worden op de EFM32. Dit gebeurt in volgende standaardvorm:

1. Id prefix (char): r, s of u
2. Id nummer (uint_32): nummer van 7 cijfers
3. Score (uint_8): 0 – 99

De benodigde tijd voor het verzenden van de data moet zo kort mogelijk gehouden worden. Hierdoor zal de LoRa modem (RN2483) minder lang actief zijn, en wordt het energieverbruik laag gehouden. Dit bekomen we door de data (char, int) te encoderen naar bytes en in een standaard vorm te zetten. Hiervoor zijn 3 functies geschreven:

- `bool AddPlayerIdPrefix(Encode_Buffer_t *b, char data){}`

Vormt het karakter idPrefix om naar zijn ASCII waarde en voegt 1 byte toe aan de buffer.

- `bool AddPlayerId(Encode_Buffer_t *b, uint32_t data){}`

Voegt de student/personeelsnummer (playerId) toe aan de buffer. Het nummer bestaat steeds uit 7 cijfers, al dan niet vooraan aangevuld met nullen. Deze nullen worden niet doorgestuurd, maar in de decoder aangevuld indien nodig. Er worden 3 bytes toegevoegd.

- `bool AddGameScore(Encode_Buffer_t *b, uint8_t data){}`

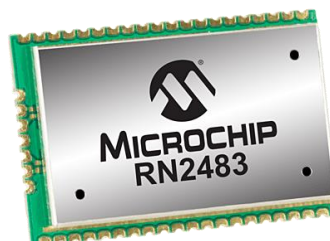
Voegt de score, 1 byte, van de speler toe aan de buffer. De score is een getal van 0 tot en met 99.

Er worden steeds 5 bytes toegevoegd aan de buffer, waarna deze doorgestuurd kan worden. Zoals in Figuur 22 aangegeven wordt zal de data voor de fictieve speler 'r0963217' met een score van 20 punten omgezet worden in:

	idPrefix	playerId	Score
Data	r	(0)963217	20
Bytes	72	0E B2 91	14

8.1.2 LoRa

Zowel het studenten/personeelsnummer als de score zijn nu omgezet naar een datapakket van 5 bytes en klaar om te verzenden. Voor het verzenden maken we gebruik van The Things Network (TTN). Dit is een open Long Range Wide Area network (LoRaWAN) dat steunt op zijn gebruikers die zelf een gateway plaatsen om apparaten met het netwerk te verbinden. LoRa is een Low Power Wide Area (LPWA) netwerk protocol en is ontworpen voor het opzetten van draadloze verbindingen met apparaten die op een batterij werken ("About LoRaWAN® | LoRa Alliance®", 2020). Het gebruik van de TTN service is gratis, en bevat enkele handige tools voor een vlotte opzet van de draadloze verbinding. Gezien de aanwezigheid van een TTN gateway op de campus, en de nood aan een low-power draadloze verbinding, hebben we gekozen om hiervan gebruik te maken.



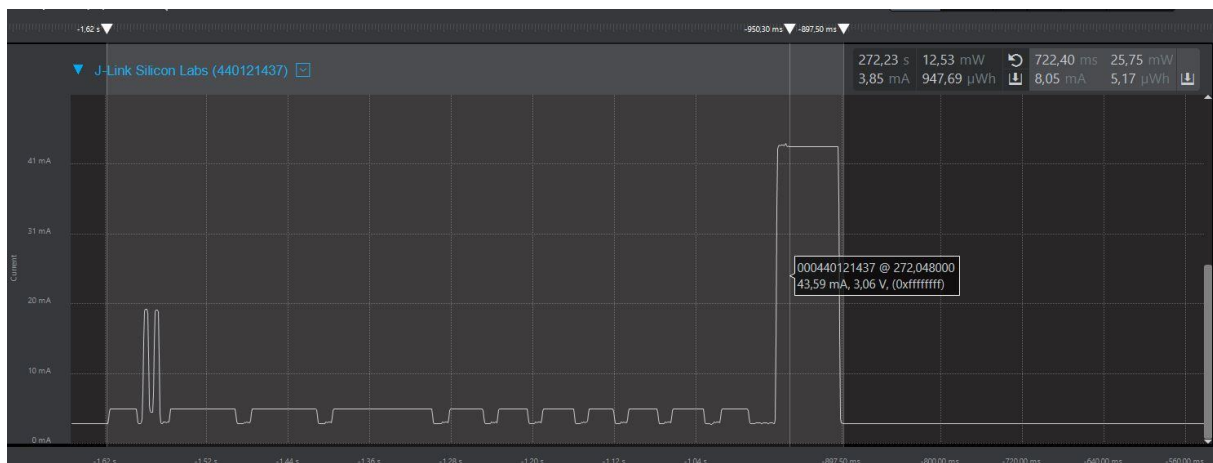
Figuur 23: RN2483 LoRa Modem

Verbinden met de TTN gateway op de campus doen we met de RN2483 (Figuur 23) LoRa Modem. Deze module zendt op de 868 MHz frequentie en wordt aangestuurd met ASCII commando's over UART.

De RN2483 werkt op een spanning van 2,1V – 3,6V en verbruikt $\pm 40\text{mA}$ tijdens het zenden. In sleep mode is het verbruik slechts $1,6\mu\text{A}$. Gezien de volledige tafel gevoed wordt met een batterij maakt dit de RN2483 ideaal in dit project. Ook is de werkingstemperatuur van belang. De tafel staat het hele jaar buiten en moet bestand zijn tegen vrieskou en direct zonlicht. Deze LoRa modem is geschikt voor gebruik bij een temperatuur van -40°C tot $+85^{\circ}\text{C}$.

Voor het verzenden van de data met de RN2483 hebben we ons gebaseerd op de LoRa Tutorial code van Dramco (<https://github.com/DRAMCO/EFM32-RN2483-LoRa-Node>).

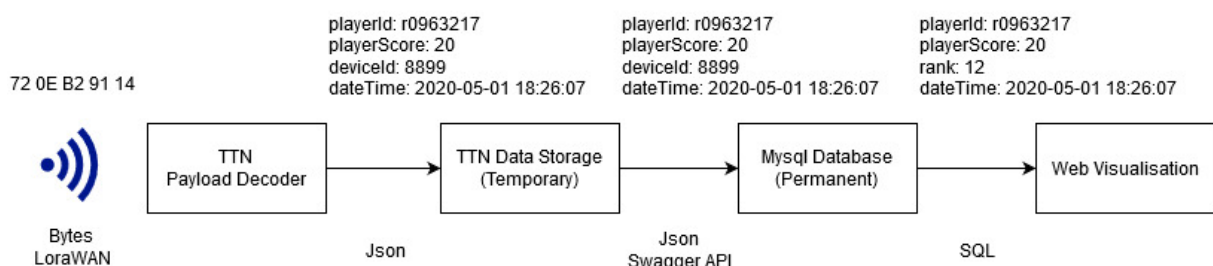
Op Figuur 24 ziet u het resultaat van de stroommeting in Simplicity Studio. Het geselecteerde gebied bevat het stroomverbruik van één zend cyclus. We onderscheiden twee gebieden, namelijk het activeren en communiceren met de RN2483, en op het einde het zenden. Gezien er een LoRa gateway aanwezig is op de campus en de afstand voor het zenden dus vrij beperkt is, zenden we met de kortst mogelijke spreading factor: SF7. Hierdoor wordt de benodigde zendtijd (en dus ook het stroomverbruik) beduidend lager. Het verzenden van het datapakket van 5 bytes neemt 69ms in beslag waarbij een stroom van $43,59\text{mA}$ gemeten werd. De totale cyclus duurt 722,4ms met een gemiddeld stroomverbruik van $8,05\text{mA}$.



Figuur 24: Energiemeting RN2483

8.2 Cloud

Het datapakket met het nummer van de speler en zijn behaalde score is verzonden over het LoRaWAN en komt via de TTN gateway, aanwezig op de campus, binnen in de online TTN console. Hier wordt de data gedecodeerd en tijdelijk opgeslagen in de “Data Storage” service. TTN biedt een API waarmee de data uit de tijdelijke opslag gehaald wordt en permanent bijgehouden wordt in een MYSQL database om de visualisatie te vergemakkelijken. Op onderstaande afbeelding (Figuur 25) ziet u het volledige proces van LoRaWAN gateway tot de web visualisatie.



Figuur 25: Connectiviteit en visualisatie (cloud)

8.2.1 Payload Decoder & Data Storage

Zoals eerder vermeld wordt het gecodeerde datapakket ontvangen door de TTN gateway waarna het gedecodeerd wordt. Het decoderen houdt in dat de bytes weer omgezet worden in leesbare data in Json formaat. De decoder wordt geprogrammeerd in JAVA.

De ontvangen *payload* bestaat uit 3 onderdelen, en worden omgezet in twee Json variabelen:

- `playerIdPrefix = String.fromCharCode(bytes[0]);`

Zet de eerste byte om in de student/personeelsnummer prefix. (vb. HEX 72 = 'r')

- `playerId = String((bytes[1] << 16) + (bytes[2] << 8) + bytes[3]);`

Zet de middelste 3 bytes om in de student/personeelsnummer (vb. HEX 0E B2 91 = '963217'). Gezien de studenten/personeelsnummer een vaste lengte heeft van 7 cijfers worden indien nodig voor de nummer nullen toegevoegd. In het voorbeeld wordt er één nul toegevoegd en wordt dit: '0963217'.

- `decoded.playerScore = bytes[4];`

Zet de laatste byte om in de behaalde score (vb. HEX 14 = '20')

`playerIdPrefix` en `playerId` worden hierna samengevoegd tot één string die de volledige studenten/personeelsnummer bevat. In het voorbeeld is dit dus 'r0963217'. Figuur 26 is een voorstelling van de ontvangen (payload) en gedecodeerde (`playerId`, `playerScore`) data.



Figuur 26: TTN Decoder output

Naast het nummer (`playerId`) en de score (`playerScore`) van de speler die we zelf versturen en decoderen voegt TTN automatisch metadata toe aan dit pakket. In Figuur 25 worden de belangrijkste twee weergegeven:

1. `deviceId`: Dit is het id van het apparaat dat de data uitzendt. Deze id is van belang als het project ooit uitgebreid wordt met meerdere tafels. Zo kunnen scores ook per tafel bijgehouden worden.
2. `dateTime`: Dit is de datum en het tijdstip waarop de data ontvangen werd. We gebruiken deze om weer te geven op de website zodat gebruikers kunnen zien wanneer een bepaalde score gespeeld werd.

Het bovenstaande datapakket bevat nu `playerId`, `playerScore`, `deviceId` en `dateTime` en wordt tijdelijk, voor maximaal 7 dagen, opgeslagen in de database van TTN. Omdat we de gespeelde spellen en hun scores voor meer dan 7 dagen willen bijhouden, om zo een relevant scorebord te kunnen opstellen, moeten we deze data overzetten van TTN's Data Storage naar een andere (permanente) database.

8.2.2 Mysql Database

Gezien we het score bord willen weergeven op een website is de eenvoudigste manier de data in een MYSQL database bij te houden. De data moet dus overgedragen worden van de tijdelijke TTN database naar onze permanente MYSQL database. Hiervoor heeft TTN een API voorzien genaamd Swagger. Deze API maakt het mogelijk de data in Json formaat uit te lezen via cURL (zie Figuur 25). Swagger voorziet op zijn website een testpagina waar een *request URL* getest kan worden. Op Figuur 27 is de output van deze URL te zien. In deze URL kan er meegegeven worden van welke periode we data willen ophalen uit de tijdelijke database. De periode kan ingesteld worden om data op te halen van de laatste 1 minuut tot data van de laatste 7 dagen. In dit voorbeeld halen we de data op van de laatste 5 minuten: "last=5m" en zien we dat er data van één spel wordt weergegeven.

Request URL: <https://lps-tut-turntable.data.thethingsnetwork.org/api/v2/query?last=5m>



Figuur 27: Swagger API a.d.h.v. cURL

Bovenstaande *request URL* gebruiken we in een Python script dat met een interval van één minuut herhaaldelijk uitgevoerd wordt. Via de URL in onderstaand stuk Python script wordt steeds de data van de laatste 5 minuten opgehaald. Zo zijn we zeker dat alle data in de MYSQL database opgenomen wordt. Uiteraard wordt er nagekeken of de informatie van een bepaald spel reeds in de database zit of niet, om geen dubbele data bij te houden.

```
#Get cURL data, authenticate with TTN Access Key
headers = {
    'Accept': 'application/json',
    'Authorization': 'key ttn-account-v2.8_*****yMRvdeOelp*****NXqfUY42t_*****',
}

#Get data from past 5 minutes (xm), hour(xh), day (xd)
params = (
    ('last', '5m'),
)

response = requests.get('https://lps-tut-turntable.data.thethingsnetwork.org/api/v2/query',
headers=headers, params=params)
```

Nu we de data opgehaald hebben en nagekeken dat deze nog niet aanwezig is in onze database, vormen we de Json structuur om en slaan we de data op in lokale variabelen. In onderstaand stuk Python code wordt een sql-query uitgevoerd die de lokale variabelen INSERT in de MYSQL database.

```
#Add values to database
inscursor = mydb.cursor()
sql = "INSERT INTO loradata (device, playerId, playerScore, dateTime) VALUES (%s, %s, %s, %s)"
val = (localdeviceid, localplayerId, localplayerScore, localdateTime)
inscursor.execute(sql, val)
```

```
#Commit data to database
mydb.commit()
```

In onderstaande afbeelding (Figuur 28) is de uitvoer van de Python code te zien. Elke minuut wordt de ontvangen data van de laatste 5 minuten opgehaald. De eerste keer zien we dat alle 3 opgehaalde records reeds in de database aanwezig zijn, er wordt dus geen data meer toegevoegd. Eén minuut later wordt er opnieuw data opgehaald (4 records). We zien dat er ondertussen een spel gespeeld is, en dat deze nieuwe data toegevoegd wordt aan de database. De andere 3 records zijn ook opgehaald, maar zijn zoals eerder gezegd reeds aanwezig in de database. De derde keer dat het Python script uitgevoerd wordt, weer één minuut later, is er geen spel meer gespeeld en dus ook geen nieuwe data doorgestuurd. De laatste 4 records zijn reeds aanwezig in de database en worden niet opnieuw toegevoegd.

```
>>>
= RESTART: C:\Users\Pieter\Desktop\Embedded system design 2 - Labo\Lora\Get data
  from Swagger\getDataToDatabase.py
Convert data to json finished
Fill mysql database finished
-> 0 records inserted
-> 3 records already exist

>>>
= RESTART: C:\Users\Pieter\Desktop\Embedded system design 2 - Labo\Lora\Get data
  from Swagger\getDataToDatabase.py
Convert data to json finished
Fill mysql database finished
-> 1 records inserted
-> 3 records already exist

>>>
= RESTART: C:\Users\Pieter\Desktop\Embedded system design 2 - Labo\Lora\Get data
  from Swagger\getDataToDatabase.py
Convert data to json finished
Fill mysql database finished
-> 0 records inserted
-> 4 records already exist
```

Figuur 28: Python output

Onderstaande afbeelding (Figuur 29) toont het voorbeeld record dat aan de database toegevoegd is.

← T →		id	device	playerId	playerScore	dateTime
		1				
<input type="checkbox"/>	Wijzigen	62	8899	r0963217	20	2020-05-01 18:26:07

Figuur 29: Toegevoegd record in MYSQL database

8.2.3 Web Visualisatie

De laatste stap in de visualisatie van de highscores is het weergeven op een webpagina. Dit maakt het mogelijk voor spelers om hun eigen score te zien, en deze te vergelijken met die van anderen. We maken hiervoor gebruik van sql-queries in PHP om de MYSQL database uit te lezen. De responsieve website is opgebouwd met Bootstrap 4 en is te bekijken op computers, tablets en smartphones.




Op de webpagina zijn drie tabellen te zien. De eerste tabel (Figuur 30) "Today's best" geeft de beste drie scores van de dag weer. Deze tabel wordt opgesteld met onderstaande sql-query. Ze haalt uit de databaseloradata de playerId, playerScore en dateTime op. Daarnaast wordt voor elke score ook de rank berekend. De rank geeft weer op welke plaats (all time) de speler staat met zijn score.

```
$sql = "SELECT subquery.playerId, subquery.playerScore, subquery.dateTime,
subquery.rank FROM (SELECT playerId, playerScore, dateTime, @curRank
:=@curRank + 1 AS rank FROM loradata, (SELECT @curRank := 0) q ORDER BY
```

```
playerScore DESC, dateTime ASC) AS subquery WHERE CAST(dateTime AS DATE) =
CURDATE() ORDER BY playerScore DESC, dateTime ASC LIMIT 3";
$result = mysqli_query($conn, $sql);
```

Today's Best

Best 3 scores of the day.

	Rank	Player	Score	Date & Time
	9	r0158963	26	2020-05-01 09:18:22
	12	r0237784	19	2020-05-01 10:22:51
	24	r0158963	7	2020-05-01 09:12:43




Figuur 30: Website "Today's Best"

De tweede tabel geeft de "Hall of Fame" weer (Figuur 31 links). Dit zijn de 10 hoogste scores die ooit behaald zijn. De tabel wordt opgesteld aan de hand van volgende sql-query:

```
$sql = "SELECT subquery.playerId, subquery.playerScore, subquery.dateTime,
subquery.rank FROM (SELECT playerId, playerScore, dateTime, @curRank
:=@curRank + 1 AS rank FROM loradata, (SELECT @curRank := 0) q ORDER BY
playerScore DESC, dateTime ASC) AS subquery ORDER BY playerScore DESC,
dateTime ASC LIMIT 10";
$result = mysqli_query($conn, $sql);
```

Hall of Fame

Best 10 scores of all time.

	Rank	Player	Score	Date & Time
	1	r0298876	33	2020-04-30 15:47:43
	2	r0124475	32	2020-04-30 15:59:36
	3	r0963211	31	2020-04-15 23:46:31
	4	r0584637	31	2020-04-30 15:13:10
	5	r6397415	28	2020-04-28 12:20:20
	6	r1620008	28	2020-04-30 15:26:02
	7	r0459963	28	2020-04-30 15:50:56
	8	r6497216	27	2020-04-30 15:14:12
	9	r0158963	26	2020-05-01 09:18:22
	10	r0036927	24	2020-04-30 15:38:07

Latest plays

Latest 5 plays.


Rank	Player	Score	Date & Time
12	r0237784	19	2020-05-01 10:22:51
9	r0158963	26	2020-05-01 09:18:22
24	r0158963	7	2020-05-01 09:12:43
11	r0114772	22	2020-04-30 17:28:55
15	r0559666	14	2020-04-30 17:19:30

Figuur 31: Website "Hall of Fame" (links) en "Latest plays" (rechts)

De derde en laatste tabel (Figuur 31 rechts) geeft de score van de laatste vijf spellen weer. Hierdoor kunnen spelers steeds hun eigen score bekijken na het spelen, ook al is deze niet goed genoeg om terecht te komen in "Today's Best" of de "Hall of Fame". Ook hier wordt de rank weergegeven, waardoor spelers hun score kunnen vergelijken met anderen en deze eventueel kunnen proberen te verbeteren. Het opstellen van deze tabel gebeurt met volgende sql-query:

```
$sql = "SELECT subquery.playerId, subquery.playerScore, subquery.dateTime,
subquery.rank FROM (SELECT playerId, playerScore, dateTime, @curRank
:=@curRank + 1 AS rank FROM loradata, (SELECT @curRank := 0) q ORDER BY
playerScore DESC, dateTime ASC) AS subquery ORDER BY dateTime DESC LIMIT
5";
$result = mysqli_query($conn, $sql);
```

Tot slot een voorstelling van de volledige “Turntable Highscores” webpagina (Figuur 32):


Turntable Highscores
Home
Game Rules

Today's Best

Best 3 scores of the day.

Rank	Player	Score	Date & Time	
1	9	r0158963	26	2020-05-01 09:18:22
2	12	r0237784	19	2020-05-01 10:22:51
3	24	r0158963	7	2020-05-01 09:12:43

Latest plays

Latest 5 plays.

Rank	Player	Score	Date & Time
26	r0697361	5	2020-05-01 10:38:46
12	r0237784	19	2020-05-01 10:22:51
9	r0158963	26	2020-05-01 09:18:22
24	r0158963	7	2020-05-01 09:12:43
11	r0114772	22	2020-04-30 17:28:55

Hall of Fame

Best 10 scores of all time.

Rank	Player	Score	Date & Time	
1	1	r0298876	33	2020-04-30 15:47:43
2	2	r0124475	32	2020-04-30 15:59:36
3	3	r0963211	31	2020-04-15 23:46:31
4	4	r0584637	31	2020-04-30 15:13:10
5	5	r6397415	28	2020-04-28 12:20:20
6	6	r1620008	28	2020-04-30 15:26:02
7	7	r0459963	28	2020-04-30 15:50:56
8	8	r6497216	27	2020-04-30 15:14:12
9	9	r0158963	26	2020-05-01 09:18:22
10	10	r0036927	24	2020-04-30 15:38:07

Project Embedded System Design 2 Labo - Team Turntable © 2020

Icons made by Freepik

Figuur 32: Website volledig

9 Besluit

Voor dit project bedachten we een leuke manier om de anders vrij saaie tafels die zich buiten bevinden op de Technologicampus Gent leuker te maken. Persoonlijk vinden wij dit concept en idee geslaagd om studenten en personeelsleden naar buiten te lokken om zo eventueel ervoor te zorgen dat nieuwe sociale contacten gelegd worden. Met behulp van een overzichtelijke website wordt het competitie aspect toegevoegd waardoor dit spel aantrekkelijk blijft.

Bij het ontwerp werd steeds nagedacht over de praktische mogelijke situaties, waarbij het belangrijk is rekening te houden dat dit embedded system zich in een buiten omgeving bevindt. Het moet bestand zijn tegen verschillende weersomstandigheden en tegen de agressieve student die niet tegen zijn verlies kan. Het is ook belangrijk om een “keep it simple” mindset te hanteren zodat dit gebruiksvriendelijk is voor iedereen die dit spel wil spelen. Ook een esthetisch mooi design is van groot belang.

Naast deze belangrijke factoren trachtten we ook om dit embedded system zo low power mogelijk te maken. Dit was niet altijd eenvoudig aangezien de bespeelbaarheid en robuustheid hierdoor niet in het gedrang mag komen. Zo moeten de 7-segment displays en de LEDs voldoende licht geven om dit spel te spelen indien de zon volop schijnt. Ter compensatie werd een energy harvesting systeem ontwikkeld die gebruik maakt van zonlicht om aan de energie noden te voorzien. Volgens berekening zijn we erin geslaagd om een autonome werking te voorzien in alle mogelijke omstandigheden.

Gezien de actuele omstandigheden i.v.m. het coronavirus zijn niet alle componenten tijdig bij de juiste persoon geraakt, waardoor het uitwerken van o.a. de RFID module en 7-segment displays niet afgewerkt is. We hebben echter wel zoveel mogelijk moeite gedaan om de verschillende delen apart werkende te krijgen. Bovendien werd ermee rekening gehouden dat de overlappende integratie delen goed afgesproken zijn waardoor volledige integratie eenvoudig zou moeten mogelijk zijn. Zo werden zeer handige en mooi uitgewerkte functies voorzien om de sensoren aan te spreken vanuit de main file om de software integratie te voorzien. Wegens de sociale beperkingen is het onmogelijk om dit volledige systeem als een volledig eindproduct af te leveren, wat wij persoonlijk zelf ook spijtig vinden. We hadden dit graag volledig werkende gezien in real life.

Ook de verantwoordelijkheden werden iets anders verdeeld aangezien bijvoorbeeld het communicatieonderdeel in de eerste weken niet uitgewerkt kon worden wegens de afwezigheid van een LoRa communicatie mogelijkheid. Hierdoor werd het RFID gedeelte volledig verschoven naar Pieter. Uiteindelijk is Pieter erin geslaagd om toch een LoRa gateway te voorzien waardoor hij (mits een grote inzet) ook zijn opdracht kon vervullen. De RFID module kon niet in software uitgewerkt worden aangezien deze niet tijdig geleverd werd.

Iedereen in de groep heeft zijn verantwoordelijkheid voor zijn eigen taak nageleefd. Hierbij voorziet iedereen zijn eigen gewenste kwaliteit voor het uiteindelijke resultaat. Verder werd veel overlegd binnen dit project om de groepssfeer er toch in te houden en om de gemeenschappelijke aspecten te respecteren.

9.1 Verantwoordelijkheden

Technisch:

Sensors (HW en SW)	IMU	Jona
	RFID	Pieter
Draadloze communicatie (LoRa) & Visualisatie (website)		Pieter
Power/Energie voorzieningen		Daan
Zonnepaneel		Tuur
Spel zelf in software met drukknoppen en LED 's		Niels
Hardware integratie met sensoren en μ controller		Jona
Mechanisch design		Tuur
Aansturing 7-segment displays		Niels

Niet-Technisch:

Integratie manager	Tuur
Time manager	Daan
Verslaggever	Pieter
Presentatie manager	Jona

10 Referenties

De C-code voorzien van gedetailleerde Doxygen documentatie samen met de design files en hardware schema's is terug te vinden op: <https://github.com/ionacappelle/Embedded-Systems-II-Lab>

Zonne-energie gids. (n.d.). Aantal zonuren in Vlaanderen. Opgehaald van www.zonne-energiegids.be: <https://www.zonne-energiegids.be/aantal-zonuren-in-vlaanderen/>

About LoRaWAN® | LoRa Alliance®. (2020). Geraadpleegd op 3 mei 2020, van <https://loralliance.org/about-lorawan>