



SECUNDAIR ONDERWIJS
ZELZATE • WACHTEBEKE • MARIAKERKE

Sint-Laurens

Geïntegreerde Proef: Home SCADA

--- “Eenvoudig en snel huishoudelijke apparaten aansturen met een gebruiksvriendelijke interface zonder hierbij internet-privacy te verstoren.”

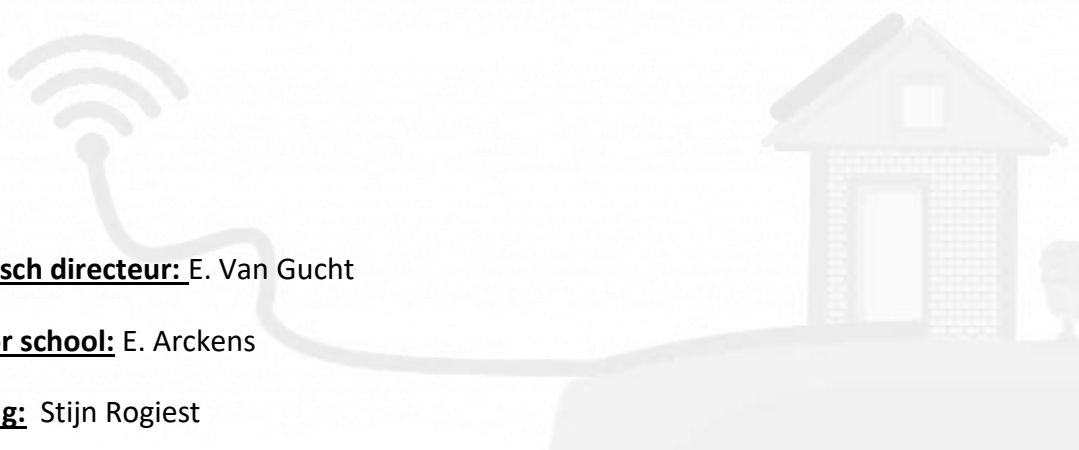
TSO Elektriciteit elektronica

2018 – 2019

Technisch directeur: E. Van Gucht

Mentor school: E. Arckens

Leerling: Stijn Rogiest



Voorwoord

Als eindejaarsleerling in de richting Elektriciteit-Electronica kreeg ik de opdracht om een geïntegreerde proef af te leggen. Dit houdt in dat we een volledig jaar aan een project moeten werken om dan aan het einde van het jaar dit voor te stellen aan een jury.

Na veel brainstormen heb ik ervoor gekozen om een domoticsysteem te maken, dat verschillende elektronische apparaten via een gebruiksvriendelijke interface op een eenvoudige manier aanstuurt.

Deze opdracht heb ik uitgevoerd samen met mijn klasgenoot *Dylan Dedapper*. Om een goede werkomgeving te creëren hebben we gezorgd voor een goede taakverdeling. Dit was zeker niet altijd even makkelijk.

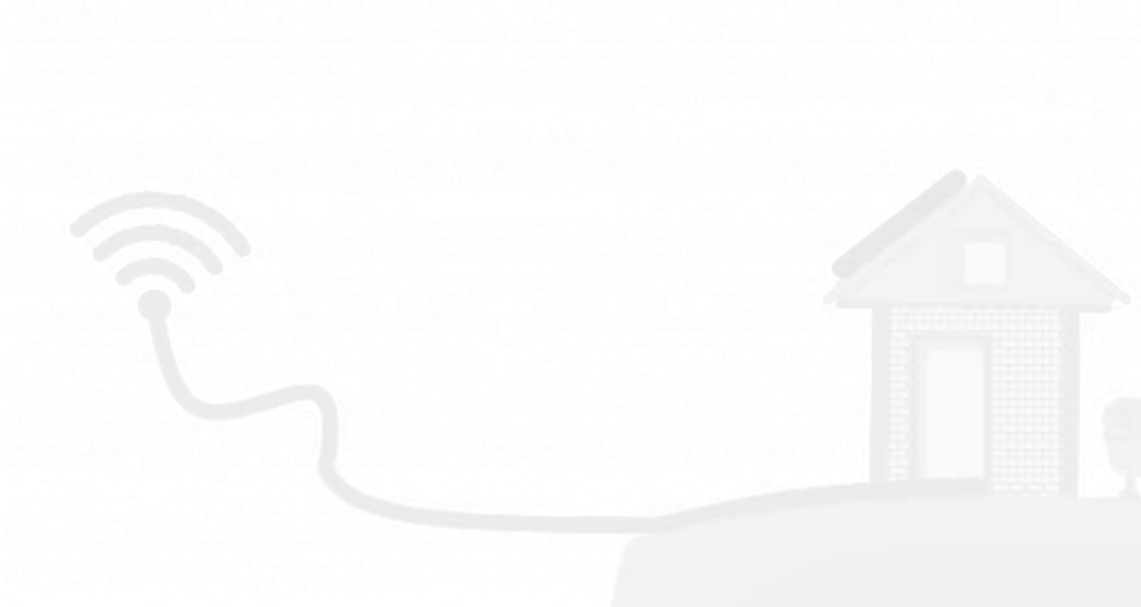
Tijdens het project zijn we verschillende obstakels en uitdagingen tegengekomen. Ik heb tijdens dit project vele dingen bijgeleerd door mijzelf uit te dagen en hiervoor een oplossing te zoeken.

Hierbij wil ik alvast mijn begeleidende leerkrachten meneer Coppejans en meneer Arckens bedanken voor hun hulp en steun gedurende het schooljaar.



1 Inhoud

2	Inleiding	4
3	Omschrijving van de opdracht.....	4
4	Werking van het apparaat.....	5
4.1	Algemene werking.....	5
4.2	Uitgebreide uitleg software.....	5
4.2.1	Bij de 'master'	6
4.2.2	Bij de 'slave'	7
4.2.3	Protocol berekeningen	9
4.3	Uitgebreide uitleg hardware	14
4.3.1	Microcontrollers	14
4.3.2	Communicatie	15
5	Elektrische schema's	16
5.1.1	Prototypes	16
5.1.2	Stroommeting.....	18
5.1.3	Professioneel PCB.....	20
5.1.4	Componentenlijst	24
6	Design en afwerking	26
6.1	Smartphone app	26
6.2	Behuizing	29
7	Bronnenlijst	31
8	Besluit en zelfreflectie	32
9	Logboek	33
10	Bijlagen	44
10.1	GIP-opdracht godsdienst: normen en waarden op het werkterrein.....	44



2 Inleiding

Deze bundel is de schriftelijke weergave van de geïntegreerde proef waaraan ik heel dit schooljaar aan heb gewerkt. Zoals de titel aangeeft, is het hoofddoel van dit eindwerk 'Home SCADA'.

Home is wat het zegt: 'het huis', onze bedoeling is om uw leefomgeving te versterken met elektronische apparaten en hierdoor uw dagelijks leven makkelijker maken.

SCADA is de afkorting voor 'Supervisory Control And Data Acquisition', dit betekent dat er op hetzelfde moment data verwerkt en gemonitord kan worden.

(https://nl.wikipedia.org/wiki/Supervisory_control_and_data_acquisition) Praktisch voorbeeld: een verwarmingselement aansturen maar ook de temperatuur kunnen aflezen.

In dit document zullen verschillende dingen aan bod komen:

- Eerst zal je een duidelijke omschrijving van de opdracht te zien krijgen: *wat is een 'Home Control'-systeem? Wat was belangrijk bij het kiezen van de opdracht?*
- Dan volgt uitgebreide uitleg over de werking van software en hardware: *welke programmatuur was belangrijk? Hoe worden datapakketten verstuurd en opgesteld? Welke elektronische componenten zijn gebruikt en wat doen ze? Ontwerp van de schema's.*
- Vervolgens vind je een logboek, een schriftelijk overzicht over de 'werkuren' binnen deze opdracht.
- Ten slotte een besluit en zelfreflectie: *Wat vind ik van het eindresultaat? Wat heb ik allemaal bijgeleerd? Wat had ik liever beter gedaan?*

Bij het maken van dit eindwerk heb ik enorm veel bronnen van het internet gebruikt, deze zullen zich bevinden waar ik ze raadpleegde. Er is ook een bronnenlijst bijgevoegd aan het einde van dit document.

3 Omschrijving van de opdracht

We kregen de gelegenheid om zelf een GIP-opdracht uit te vinden en voor te stellen aan de leerkrachten. Uiteindelijk kwam dit tot een Dominica-systeem, een systeem dat het leven in huis makkelijker zal maken door gebruik te maken van elektronica. Simpelweg: draadloos lichten bedienen vanaf je smartphone, laptop of tablet.

Privacy was ook enorm belangrijk om even over na te denken, de home-control systemen die zich nu al op de markt bevinden hebben dit internet-privacy probleem. Al deze systemen verbinden met een database om gebruikersactiviteiten te loggen, en dat is 'not done'. Gebruikers van dit soort systemen willen niet dat hun hele leven zich ergens op een computer bevindt zonder dat ze er iets van weten! Daarom wordt er in ons systeem geen verbinding gemaakt met het internet, dus er kan ook geen gelogde data online komen, dit zal al veel mensen geruststellen.

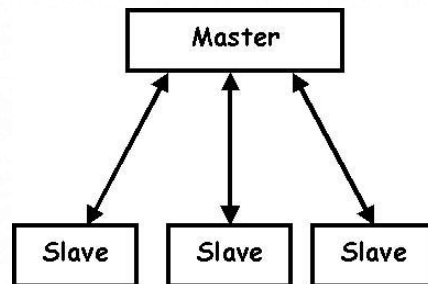
In het kort: een draadloos 'home-control and monitoring' systeem waarbij je eenvoudig en snel huishoudelijke apparaten kan aansturen via een gebruiksvriendelijke smartphone app zonder jezelf vragen te stellen over privacy.

4 Werking van het apparaat

In dit deel leg ik uit hoe de hardware en de software van dit grote project samenvloeien. Het protocol wordt hier zeer uitgebreid uitgelegd, omdat dit ook het hart van het project is, de draadloze communicatie tussen de verschillende modules in het huis.

4.1 Algemene werking

Ons systeem bestaat uit 2 verschillende modules, een slave en een master. Bij een typisch gebruik van dit systeem is er maar één master en zijn er meerdere slaves. De master ontvangt commando's van de smartphone app die hij dan zal doorverwijzen naar zijn slaves. Een simpele schematische voorstelling ziet er zo uit:



Elke slave heeft een uniek adres: hiermee kan elke slave apart bestuurd worden. De master verstuurt een datapakket naar alle slaves met hierin het adres van de ontvanger, als de slave een datapakket ontvangt en het bevat zijn adres, interpreteert de slave de data, anders wordt het pakket genegeerd. Met het ontvangen datapakket kan hiermee een licht (of iets anders) aangestuurd worden. Als het interpreteren succesvol is stuurt de slave een bericht terug naar de master dat alles 'oké' is.

In het volgende onderdeel wordt uitgelegd hoe de communicatie tussen master en slave (de pijlen) op software niveau wordt gerealiseerd. Daarna wordt ook uitgelegd hoe dat op hardware niveau gebeurt.

4.2 Uitgebreide uitleg software

De communicatie tussen de master en de slaves wordt gerealiseerd door een protocol: HCP of 'Home Control Protocol', dit is een enorm onderdeel van dit project. Het hardware aspect van de communicatie wordt later nog uitgebreid uitgelegd.

Voor we tot een stabiel protocol kwamen, hebben we maar liefst 4 versies ontwikkeld. Dat was nodig om ervaring te krijgen met de draadloze communicatie en de programmeertaal C++. Voor de definitieve versie konden we dan een aantal problemen ontwijken. Hierna volgt een lijst van die problemen en hun oplossing in HCP versie 4:

- 2 modules mogen nooit op hetzelfde moment data versturen: dit vergroot de kans van datacorruptie enorm. *Oplossing: een geavanceerd timingssysteem dat ervoor zorgt dat elke slave zijn moment heeft om te sturen, deze momenten zijn niet overlappend.*
- Draadloze communicatie kan zeer snel verstoren, er moet dus een systeem zijn dat ervoor zorgt dat er geen corrupte data worden verwerkt. *Oplossing: een 'Cyclic redundancy check' (of checksum; https://en.wikipedia.org/wiki/Cyclic_redundancy_check) bijvoegen (een code die de inhoud van een datapakket beschrijft in 2 bits, als er 1 bit wordt verstoord in het pakket, verandert de code meteen), hiermee kan er gecheckt worden of data gewijzigd zijn tussen de zender en ontvanger.*
- Adressering en koppelen van slaves: in vorige versies werd aan slaves een adres toegewezen bij het uploaden van het softwareprogramma. Dit beperkt het aantal slaves op planeet aarde omdat elke slave een uniek adres moet hebben + het datapakket mag niet te groot worden

(het adres wordt meegestuurd). *Oplossing: een pair modus ontwikkelen, gebruik maken van een fabriekscode (unieke code voor elke slave op de aarde) en een manier om deze codes op te slaan samen met een adres in de master module. Elke fabriekscode correspondeert met een lokaal adres.*

Een lijst van doelen die ik wil implementeren in HCP versie 4:

- Een overzicht kunnen verkrijgen van de master waarop je kunt zien welke slaves er offline (ontkoppeld) zijn en welke er online zijn.
- Meerdere waardes per pakket versturen. *Praktisch voorbeeld: de 3 rood, groen en blauw waardes die nodig zijn om een ledstrip aan te sturen in 1 pakket versturen i.p.v. aparte pakketten. Dit zal de snelheid enorm verhogen.*
- Een variabele lengte van datapakketten: elk datapakket mag geen onnodige bytes bevatten, dit vertraagt het verzenden en ontvangen van data. Elk datapakket heeft een variabele lengte van 4 bytes tot 20 bytes.

Zowel de master en de slaves maken gebruik van de HCP-library die ik ontwikkeld heb in C++: dit is een verzameling van gedeelde code. Elke aanpassing die ik maak aan de library zorgt voor een aanpassing bij de slaves en bij de master.

De library zorgt voor verschillende functies die zowel de master als de slaves gebruiken: verzenden en ontvangen van datapakketten, inhoud van datapakketten vertalen ...

Hoe dit in zijn werk gaat, leg ik verder uit in het deel 'berekeningen bij het protocol'.

Natuurlijk verschilt het programma van de master van het programma van de slaves. Hieronder leg ik uit hoe het werkt bij elk.

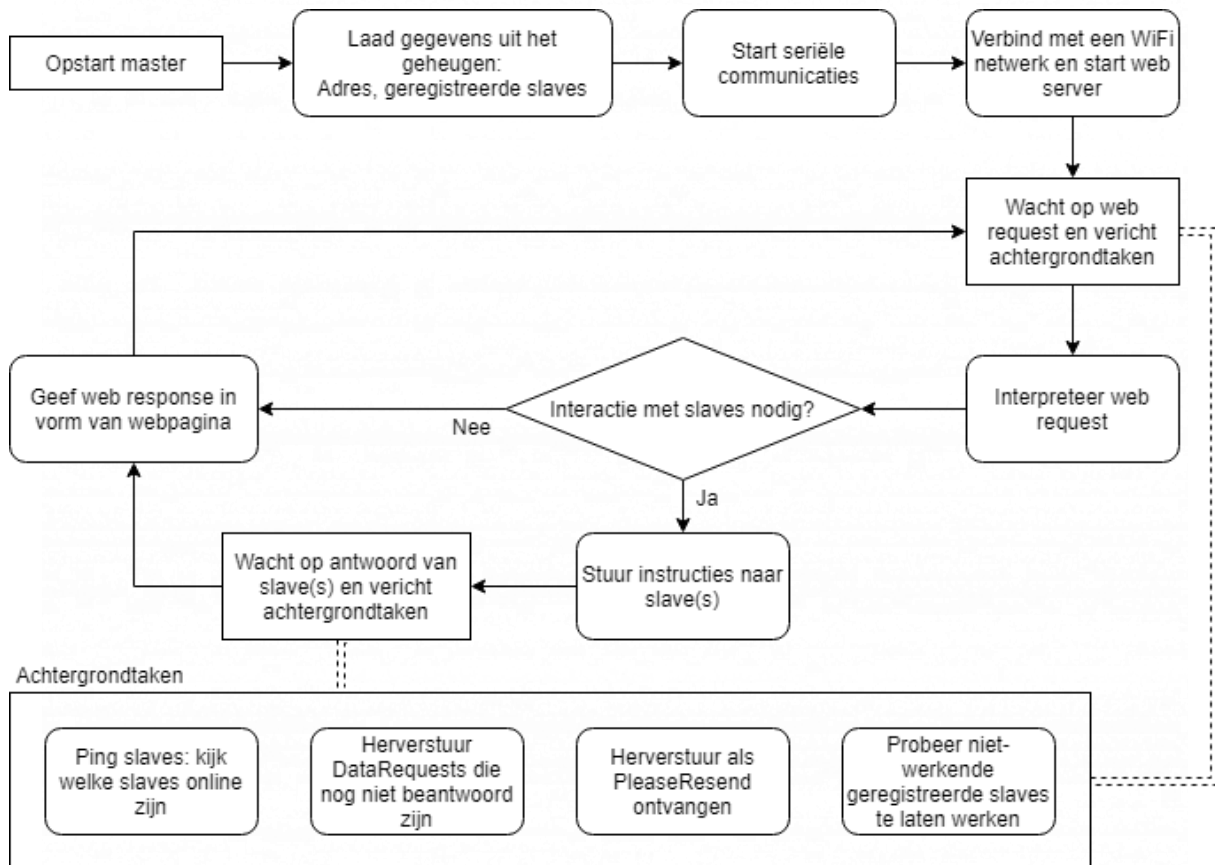
4.2.1 Bij de 'master'

De master heeft het meest complexe programma van allemaal. Alleen de master kan verbinden met het internet en commando's ontvangen van de smartphone app. De master start een seriële connectie met zijn HC12 (module die antenne aanstuurt, zie componentenlijst), die dan berichten kan doorgeven naar andere antennes. De berichten zijn zogenaamde datapakketten die kunnen omgezet worden tot een lijst van bytes (zie nog bij protocolberekeningen). De master gebruikt deze pakketten om commando's te sturen naar zijn geregistreerde slaves.



4.2.1.1 Flowchart

Het werkingsprincipe van de master is hier voorgesteld in deze flowchart:



Ik kan jammer genoeg niet het programma lijn voor lijn uitleggen, dit zou even duren want enkel het master programma bevat al zowat 1000 lijnen code (zonder library). Al mijn code staat op GitHub (zie bronnenlijst).

4.2.2 Bij de 'slave'

Bij de slave wordt er van niet veel meer gebruik gemaakt dan de gedeelde library zelf, omdat deze ook al veel achtergrondberekeningen uitvoert. Elke slave implementeert zijn eigen praktische uitwerking van de commando's die hij ontvangt.

4.2.2.1 Master naar slave commando's

Een tabel met alle commando's die elke slave ondersteund:

Technische naam → hexadecimale afkorting	Hexadecimale afkorting	Uitzending?*	Bedoeling	Verzendformaat* (master)	Antwoordformaat* (slave)
bind	0x10	Ja	De slave bindt zich aan het meegestuurde adres als de meegestuurde fabriekscod matcht met de fabriekscod van de slave. <ul style="list-style-type: none"> - <u>ufid</u>: fabriekscod, altijd 7 bytes lang. - <u>deviceType</u>: informatie over het apparaat: dit zal aangeven waarvoor deze module gebruikt zal worden. 	0x10, ufid[7], address	deviceType[4]

unbind	0x2	Nee	De slave ontbinden van een adres, het adres wordt op 0 ingesteld en verwijderd uit het geheugen van de master.	0x2	0xFF
ping	0x1	Nee	Een commando waar de slave altijd 'oke' op antwoordt, dit is om te checken of een slave online of binnen bereik is.	0x1	0xFF
refresh	0x15	Nee	Als een slave dit commando ontvangt, geeft het een lijst van maximaal 16 waardes terug, dit stelt de live data voor. Zie 'live datasysteem'. - <u>liveData</u> : de opgemeten data van de slave. De master zal dit opslaan in zijn geheugen.	0x15	liveData[16]
prop	0x2	Nee	Dit stelt een 'property' of 'properties' in bij de slave. Zie 'property-principe'. - <u>startPos</u> : de startpositie vanaf waar de slave zijn 'properties' moet instellen. - <u>properties</u> : de waardes die de slave in zijn 'property'-lijst zal moeten instellen vanaf positie startPos.	0x20, startPos, properties[]	0xFF

** Een uitzending is een datapakket met slave adres 0, dit betekent dat elke slave dit pakket zal interpreteren.*

** Het verzendformaat wordt voorgesteld op deze manier: de komma onderscheid elk veld van het datapakket. Het getal tussen de vierkante haakjes geeft de lengte van het veld aan in bytes. Velden zonder haakjes hebben lengte 1. Een veld met 0x?? stelt een constante waarde in hexadecimale vorm.*

** Antwoord '0xFF': dit antwoord geeft aan dat het doorgestuurde commando correct was uitgevoerd.*

Elke constante waarde bij het verzendformaat laat de slave weten welk commando er bedoelt word.

4.2.2.2 Property-principe

Elke slave heeft een lijst met 128 bytes (of 'properties'), de master kan een instructie (prop) sturen om deze bytes te manipuleren. Hiermee kan dan elke individuele slave deze 'properties' gebruiken op een praktische manier. Voorbeeld: byte 1 is de rood-waarde van de ledstrip die een slave aanstuurt, byte 2 de groen-waarde en byte 3 de blauw-waarde, na het instellen van deze bytes, licht de ledstrip op in de doorgestuurde kleur.

Wanneer de master een 'oke'-antwoord ontvangt, onthoud hij het feit dat hij deze heeft ingesteld, zo kunnen we later deze 'property'-waarde opvragen zonder hiervoor een vraag te sturen naar de slave.

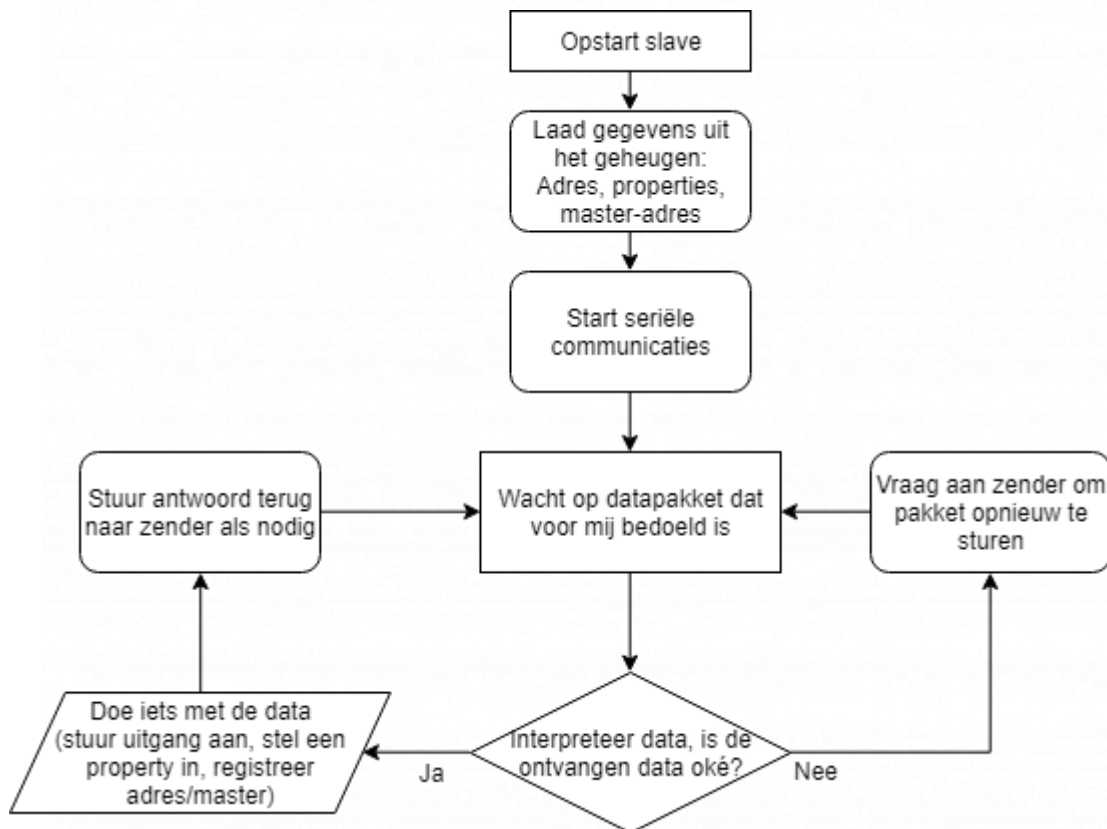
4.2.2.3 Live datasysteem

Live data is een lijst van op het moment gemeten waardes, bevoorbeeld stroommetingen.

Om live data van verschillende slaves op te kunnen slaan is er een systeem ontworpen dat ervoor zorgt dat elke slave om de zoveel tijd zijn live data mag pushen. De slave stuurt een lijst van maximaal 16 bytes (dit is het maximaal aantal data per datapakket). Deze waarden zijn gemeten wanneer de master ernaar vroeg, ze zijn dus live gemeten. Als de master deze waarden ontvangt slaat hij deze informatie op in zijn niet-vluchtige EEPROM-geheugen (zie EEPROM-indeling).

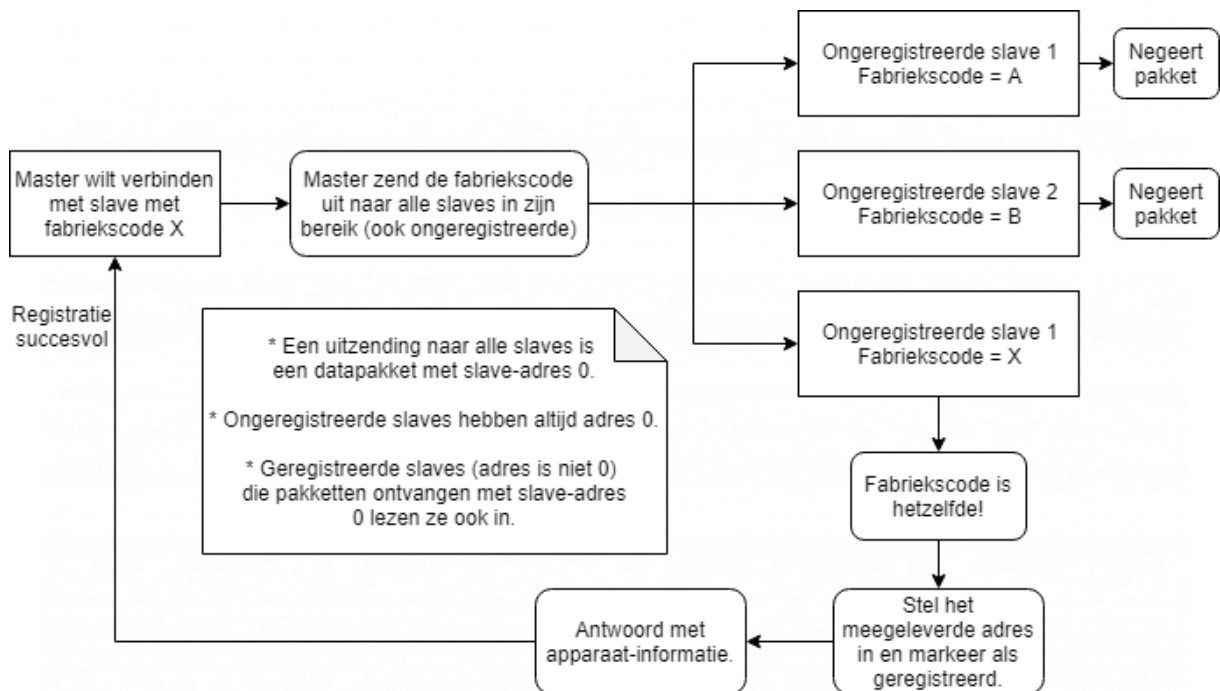
4.2.2.4 Flowchart

Net zoals de master heb ik een vereenvoudigde visuele voorstelling van de werking in een flowchart:



4.2.3 Protocol berekeningen

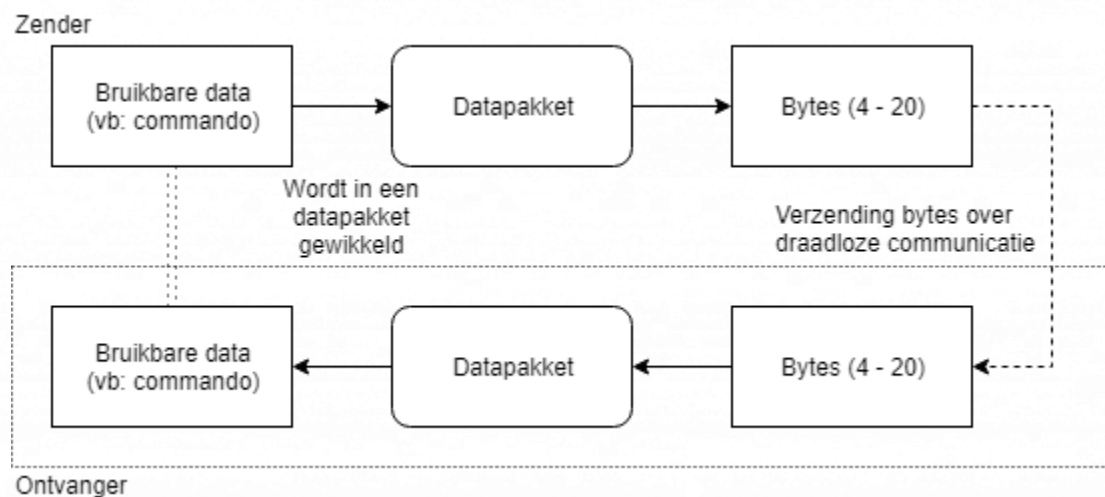
Bij het protocol komen veel wiskundige berekeningen kijken. Om iets te versturen moet er een lijst van bytes worden doorgestuurd naar de HC12-module (zie componentenlijst). Deze module stuurt een antenne aan en verstuurt die bytes naar alle andere HC12 modules en hun antennes. Elke byte die de zender verstuurt, ontvangen alle ontvangers. Er moet dus een systeem ontwikkeld worden om elke slave te onderscheiden. Hiervoor heb ik de 'fabriekscodé' en het 'adres' ingevoerd: de fabriekscodé is een 'global unique id' (https://nl.wikipedia.org/wiki/Globally_unique_identifier), deze code is voor elke slave op de aardbol uniek, deze code is 7 bytes groot en heeft dus ongeveer 255^7 combinaties ($\approx 7 \cdot 10^{16}$ combinaties, de kans op een duplicaat is dus enorm klein). Het 'adres' heeft een waarde van minimum 1 en maximum 63. Elke slave heeft een standaard adres 0, alle data naar het adres 0 worden door elke slave gelezen. Om een slave te registreren, stuurt de master een datapakket met ontvanger 0 (wordt dus door elke slave opgenomen) en met een specifieke fabriekscodé en adres. Enkel de slave die dezelfde fabriekscodé heeft als de ontvangen data, geeft een antwoord terug en stelt zijn adres in naar het ontvangen adres. Hier volgt een visuele voorstelling van dit proces:



Nadat alle slaves geregistreerd zijn kan de master ze allemaal apart bereiken. Elke slave ontvangt alleen een datapakket waarin hun adres zit bijgevoegd, anders wordt het pakket genegeerd.

4.2.3.1 Samenstelling datapakket

Datapakketten zijn een opsomming van bruikbare waardes die kunnen omgezet worden naar bytes en omgekeerd. Deze gebruiken we om dus bruikbare getallen om te zetten naar bytes en te versturen, hier een eenvoudig overzicht:



Dit is het basisprincipe van de datatransmissie, nu nemen we een kijkje hoe het datapakket wordt opgebouwd, hier een visuele voorstelling van alle velden in het pakket:

Identifier (1 byte)	CRC (2 bits)	Slave address (6 bits)	Master address (2 bits)	Packet type (2 bits)	Data length (4 bits)	Multi purpose byte (1 byte)	Data (F bytes)
A	B	C	D	E	F	G	H

- A. Identificer: een byte die het begin van het datapakket aanduidt. Hiermee kan de ontvanger weten vanaf waar hij moet lezen en kan hij dus de ingelezen data begrijpen. Dit deel is 1 byte lang (256 combinaties) en heeft in ons geval altijd de waarde 0x69 (105).
- B. CRC: 'Cyclic redundancy check' (of checksum), dit is een getal dat de inhoud van het datapakket beschrijft in 2 bits (4 combinaties). Als er 1 bit wordt verstoord in het pakket, verandert de code meteen, hiermee kan er gecheckt worden of data zijn gewijzigd tussen de zender en ontvanger. Als dit zo is, lees dan geen data en vraag aan de zender om het pakket nogmaals te versturen (zie 'pakkettypes'). Dit is om te voorkomen dat er corrupte en verkeerde data worden geïnterpreteerd.
(https://en.wikipedia.org/wiki/Cyclic_redundancy_check)
- C. Slave address: Een 6-bit getal (64 combinaties) dat het adres van de slave die aan deze communicatie deelneemt bevat. Er is dus een maximum van 63 slaves per master (slaves met adres 0 tellen niet, deze hebben nog geen adres).
- D. Master address: Een 2-bit getal (4 combinaties) dat het adres van de master die aan de communicatie deelneemt bevat (er is dus een maximum van 4 masters in hetzelfde antennebereik). *Opmerking: een slave kan dus onmogelijk naar een slave sturen en een master onmogelijk naar een master omdat het pakket altijd 1 slave-adres en 1 master-adres bevat.*
- E. Pakket type: Een 2-bit getal (4 combinaties) dat het soort pakket beschrijft, meer hierover in het onderdeel 'pakket-types'.
- F. Data lengte: Een 4-bit getal (16 combinaties) dat de lengte van de komende data aangeeft. Hiermee kan de lengte van elk pakket verschillen: minimum 4 bytes (velden A-G vormen samen 4 bytes) en maximum 20 bytes (4 bytes + F bytes als F = 16).
- G. Multi purpose byte: Deze byte (256 combinaties) is wat het zegt, het heeft meerdere mogelijke gebruikssituaties, je kunt het zien als een extra data-byte.
- H. Data: Hier bevinden zich de bruikbare data, alles wat moet verwerkt worden aan de ontvanger. De lengte van dit veld wordt bepaald door veld F, dit veld kan ook weggelaten worden.

4.2.3.2 Pakket-types

Dit onderdeel beschrijft het werk van veld E in het datapakket. Er zijn 4 pakket types, elk heeft zijn eigen functie:

- DataRequest: Dit geeft aan dat het pakket een 'request' is, het pakket vraagt naar data of naar acties en verwacht altijd een antwoord hierop, om zeker te weten dat er werk verricht was. Elk slave commando wordt op deze manier verstuurd, om zeker te weten of het commando wel degelijk is uitgevoerd. Wanneer een pakket van dit type wordt verstuurd wordt de 'multi-purpose-byte' ingesteld op een code, die later wordt teruggestuurd samen met het antwoord, hiermee weet de master welk verstuurd 'request' bij welk antwoord past.
- Push: Dit geeft aan dat het pakket 'gepushed' wordt, het pakket wordt verstuurd en verwacht geen antwoord. Een ontvanger kan er mee doen wat hij wil.
- Answer: Dit geeft aan dat het pakket een antwoord op een 'request' bevat, dit is dus altijd het gevolg van een 'DataRequest' pakket. Een antwoord wordt als 'oké' beschouwd als de eerste byte van de data gelijk is aan 0xFF (255), elke andere waarde (of geen) markeert de 'request' als mislukt.
- PleaseResend: Dit geeft aan dat de ontvanger van dit pakket zijn vorige pakket opnieuw moet sturen, dit kan gebeuren als de checksum (zie veld B) veranderd is.

4.2.3.3 Praktische uitvoering berekeningen

Deze code neemt verschillende parameters en slaat ze op in een lijst van bytes (data). Op deze manier wordt er een datapakket opgesteld. De onderste functie berekent een checksum.

```
Packet::Packet(unsigned char slaveAddress, unsigned char masterAddress, unsigned char* data,
               unsigned char len, PacketType type, unsigned char multiPurposeByte)
{
    memset(this->data, 0, 20);
    if (len > 0)
        memcpy(&this->data[4], data, len);

    this->data[0] = Packet::identifier;
    this->data[1] = slaveAddress & 0x3F;
    this->data[2] = ((masterAddress & 0x3) << 6) | ((type & 0x3) << 4) | (len & 0xF);
    this->data[3] = multiPurposeByte;

    this->data[1] |= getCurrentCRC() << 6;
}

unsigned char Packet::getCurrentCRC()
{
    unsigned char crc = ~Packet::identifier;

    for (int i = 2; i < 20; i++)
        crc ^= this->data[i];
    crc ^= this->data[1] & 0x3F;

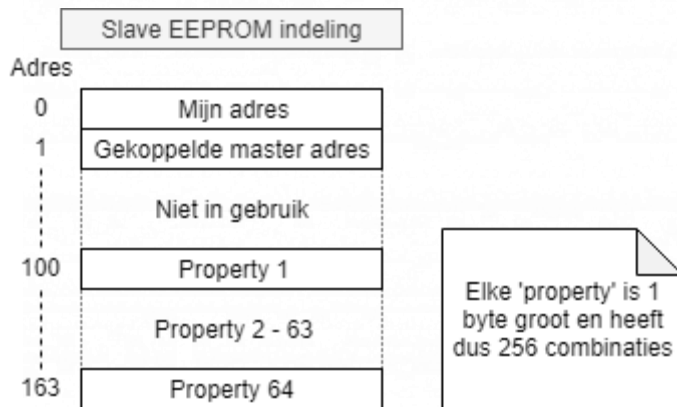
    return (crc ^ (crc >> 2) ^ (crc >> 4) ^ (crc >> 6)) & 0x3;
}
```



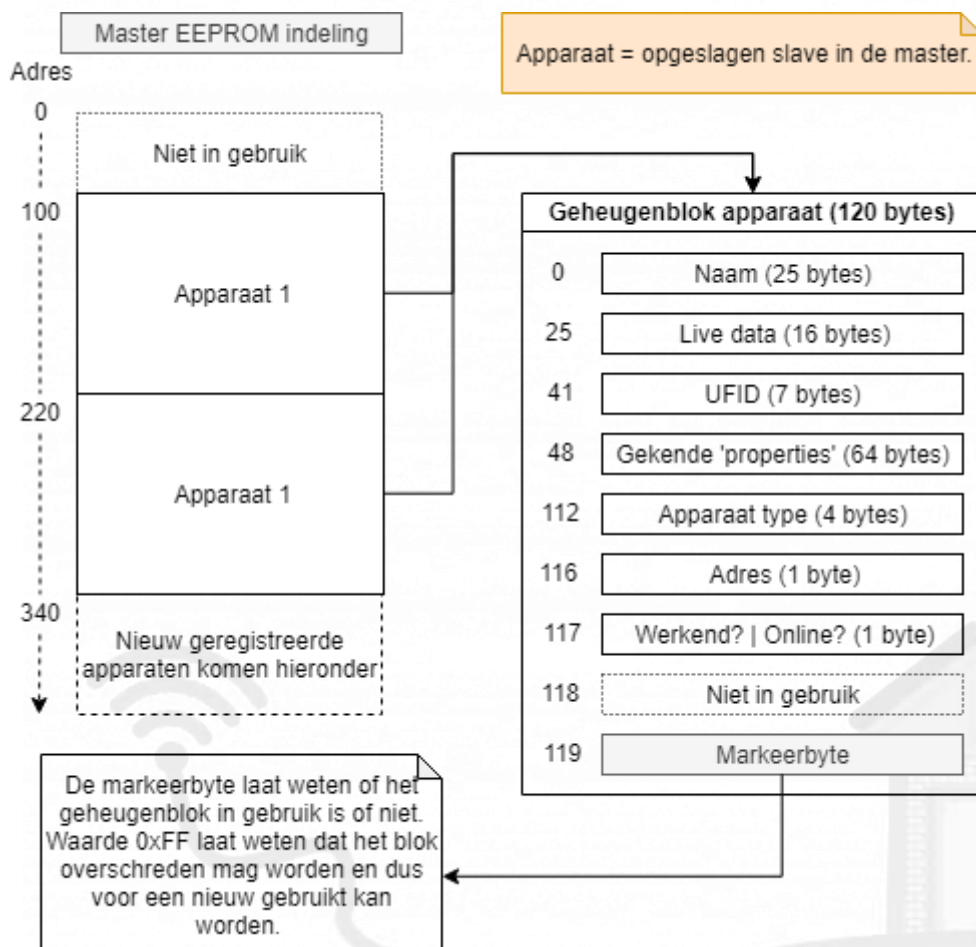
4.2.4 EEPROM-indeling

Elke microcontroller die wij gebruiken heeft een aantal bytes EEPROM-geheugen. Dit is het niet-vluchtige geheugen. Om te voorkomen dat de slave zijn adres niet vergeet bij uitschakelen, gebruiken we dit geheugen. De slave heeft 512 bytes aan EEPROM-geheugen en de master 4096 bytes. De slave gebruikt dit geheugen om zijn adres en het adres van zijn master te onthouden. De master gebruikt dit geheugen om opgeslagen slaves (of apparaten) op te slaan.

Hier een visuele voorstelling van de EEPROM geheugenindeling bij de slave:





Hier een visuele voorstelling van de EEPROM geheugenindeling bij de master:



4.3 Uitgebreide uitleg hardware

4.3.1 Microcontrollers

Bij de ontwikkeling van ons systeem is er van meerdere microcontrollers gebruik gemaakt. Dit is het brein van elke module, je kunt het zien als een kleine computer: het stuurt digitale uitgangen aan of leest ze in, sommige kunnen verbinden met Wi-Fi netwerken of PWM signalen vrijgeven (<https://nl.wikipedia.org/wiki/Pulsbreedtemodulatie>) of seriële communicatie ... Er zijn natuurlijk heel wat soorten. Hier volgt een overzicht van de controllers die wij gebruikt hebben bij zowel de master en de slave:

	Slave	Master
Naam	Arduino Pro Mini (5V type)	ESP8266 module
Foto		
Microprocessor	ATmega168 (@ 16 MHz)	ESP8266 (32-bit low power @ 80MHz)
Werkvoltage	5V	3.3V
Digitale pinnen	16 (waarvan 6 PWM)	9 (waarvan 8 PWM)
Analoge ingangen	8	1
Grootte flash geheugen	16 kilobytes	512 kilobytes
Grootte SRAM	1 kilobytes	80 + 32 kilobyte
Grootte EEPROM geheugen	512 bytes	4 kilobytes
Max uitgang stroom	~40 mA	~12 mA
Wi-Fi verbinding?	Nee	Ja
Nominaal stroomverbruik	~10 mA	~100 mA
Waarom deze module?	De Arduino Pro Mini is klein, verbruikt niet veel stroom en heeft veel uitgangen, dit is ideaal om een slave mee samen te stellen.	De ESP8266 module heeft een krachtige microprocessor die kan verbinden met het internet, dit was noodzakelijk in ons geval. Het heeft ook een enorm flash geheugen en SRAM geheugen (t.o.v. Arduino). Perfect om complexe programma's te laten werken.

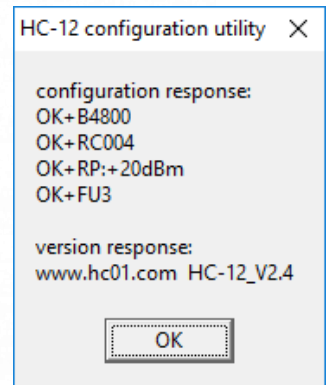
4.3.2 Communicatie

De draadloze communicatie tussen de master en slaves wordt gerealiseerd met een HC12 module (rechts een foto), het is een kleine radiomodule die radiogolven uitstraalt op een frequentieband van 433.4 MHz tot 473.0 MHz. Het genereert de nodige signalen om een antenne aan te sturen, zodat we niet ons eigen veel te ingewikkeld elektronisch circuit moeten ontwikkelen.



Om een stabiele communicatie te voorzien, moeten we eerst wat met de instellingen van de HC12 experimenteren om verschillende eigenschappen in te stellen zoals: kanaal, uitstraalvermogen, seriële datasnelheid, modus ... Wij hebben de volgende eigenschappen voor ALLE HC12 modules gebruikt:

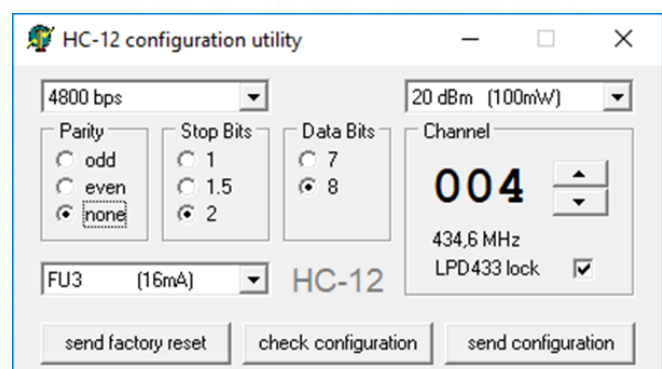
- 4800 baud transmissiesnelheid (relatief traag maar groter bereik)
- Zend- en ontvangkanaal 4
- +20 dBm zendvermogen (hoogste)
- Modus FU3: zorgt voor groot bereik en relatief laag stroomgebruik. Beperkt transmissiesnelheid tot 4800 baud; hoge snelheden zijn voor ons niet noodzakelijk.



Om de eigenschappen te versturen naar de HC12, gebruikten we een computerprogramma genaamd 'HC-12 configuration utility', samen met een UART (https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter) naar usb, zodat we die aan de computer konden aansluiten.

Document over HC12 eigenschappen: <https://www.elecrow.com/download/HC-12.pdf>

De HC12 is compatibel met verschillende antennes, een spiraalantenne (foto rechts) of een andere antenne met coaxiale aansluiting. We kregen van de leerkracht grotere coaxiale antennes (foto links), deze zouden voor een groter omnidirectionaal bereik kunnen zorgen (https://en.wikipedia.org/wiki/Omnidirectional_antenna; een antenne die in alle richtingen golven uitstuurt, de antenne heeft geen zendrichting). Later hebben we vastgesteld dat deze bijna geen verschil geven tegenover de spiraalantennes. Na uren zoekwerk is de oorzaak hiervoor nog altijd een mysterie.



5 Elektrische schema's

Nu dat er over alle software is nagedacht, gaan we over naar het hardware deel.

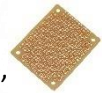
5.1.1 Prototypes

Het moet ergens beginnen, we kregen een paar HC12 modules en antennes en we konden starten met experimenteren. De eerste succesvolle test was al een feit in het eerste uur na het ontvangen van de componenten: een knop die een Arduino Nano aanstuurde om een signaal te versturen naar een andere Arduino Nano die dan een ledje aanstuurde. De problemen kwamen toen een groter bereik werd ingevoerd. Soms moest er 2 keer op de knop gedrukt worden, voordat er een signaal werd opgevangen. Dit was de motivatie om een vraag-antwoord ('requests') systeem in te voeren.

Sommige prototypes zijn gemaakt op een 'breadboard'



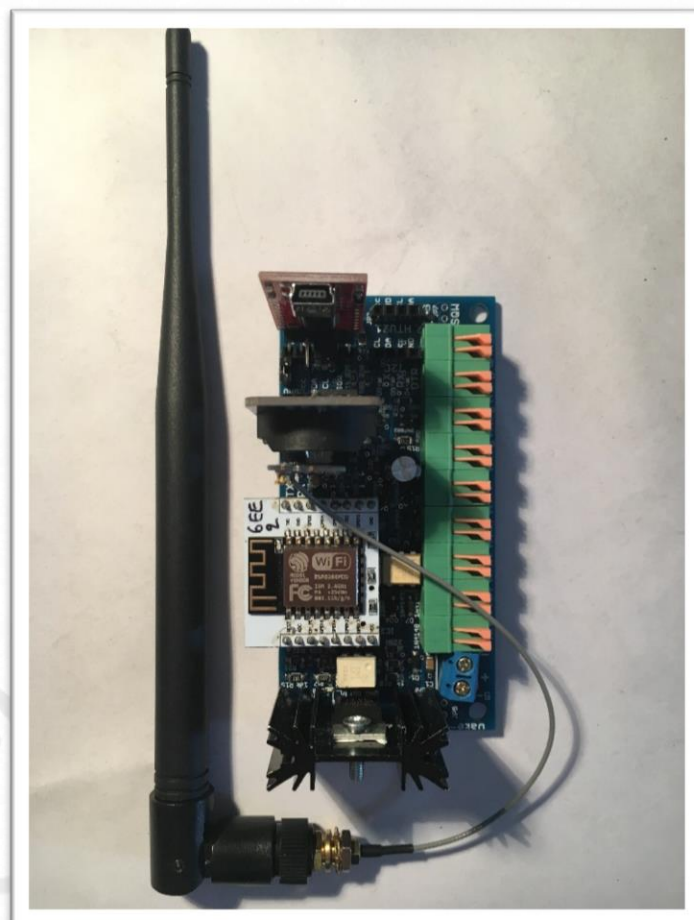
en sommige op 'perfboard'



5.1.1.1 Foto's van de prototypes

We testten de master software eerst via het bordje dat de leerkracht ons gegeven had. Het was makkelijk om naar te uploaden, maar we zijn later nog overgegaan naar andere bordjes omdat de uitgangen niet altijd even ideaal waren voor ons gebruik en omdat dit maar een tijdelijke oplossing was.

Bordje leerkracht:

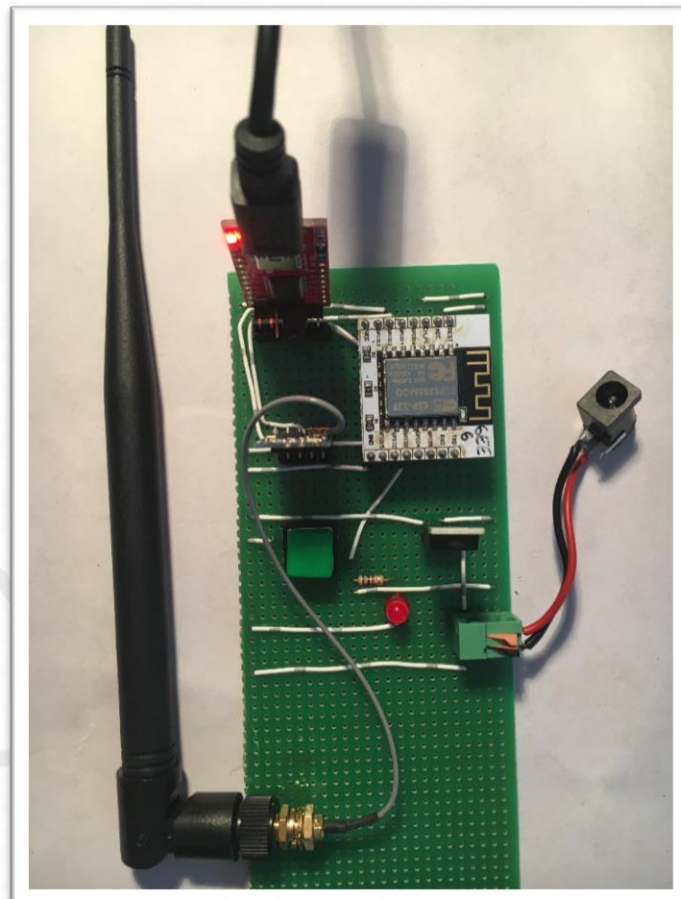


Hier werd er een testbordje gemaakt voor de master module:

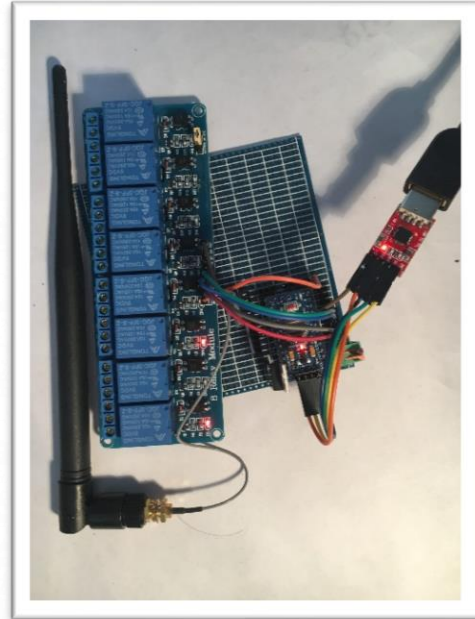


Het is een simpel bordje met een aan-uit-ledje, een programmeerknop (om de controller in programmeermodus op te starten), header voor een HC12 module, een header voor de FTDI en een header voor de 'voltage regulator'. (Zie componentenlijst) We wilden dit maken op een 'perfboard' in plaats van op een 'breadboard' om storing met de antenne en slechte contacten te vermijden.

Hier is het afgewerkte master prototype op 'perfboard':



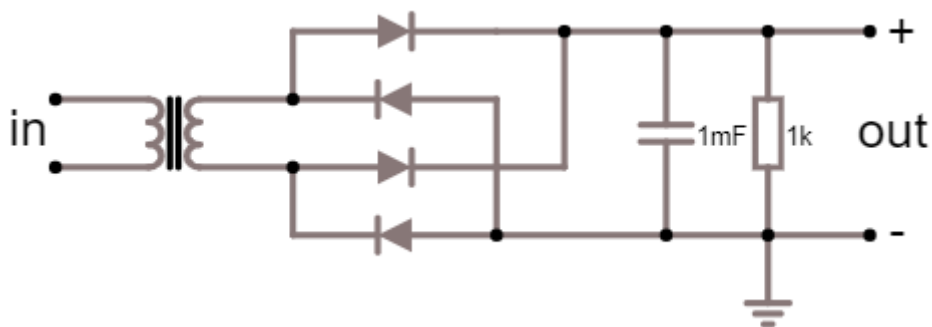
Dit zijn de afgewerkte slave prototypes:



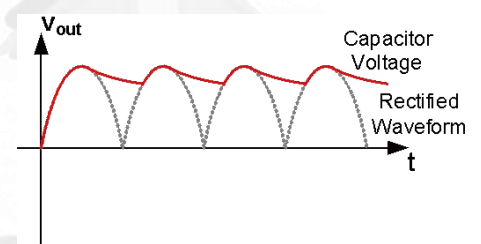
Het eerste slave prototype is gemaakt op een 'breadboard' en stuurt 4 leds aan, het maakt gebruik van een Arduino Nano. Het tweede slave prototype is gemaakt op een 'perfboard' en stuurt maximaal 8 relays (contacten) aan, het maakt gebruik van een Arduino Pro Mini.

5.1.2 Stroommeting

De slaves zijn ook in staat om elektronische signalen in te lezen, dus hier wilden we ook zeker gebruik van maken. We hebben gezorgd voor een systeem dat stroommetingen uitvoert, deze waarden kunnen dan gelezen worden door de master. Om dit te realiseren hebben we het volgende schema gebruikt:



De belasting zetten we in serie met de ingang van dit circuit en we krijgen een opgewekte spanning over de uitgang, die dan ingelezen kan worden door een microcontroller. De transformator heeft een ratio van 1 op 1000, dit betekent dat elke ampère over het net 1 milliampère induceert in ons circuit, deze stroom wordt hierna gelijkgericht van wisselspanning naar gelijkspanning. De condensator van 1 mF zorgt voor een agressieve afvlakking van het signaal (we willen zo min mogelijk rimpel op ons uitgangssignaal, anders wordt inlezen met de microcontroller een ramp). De spanning over de weerstand wordt gemeten door de microcontroller via een analoge



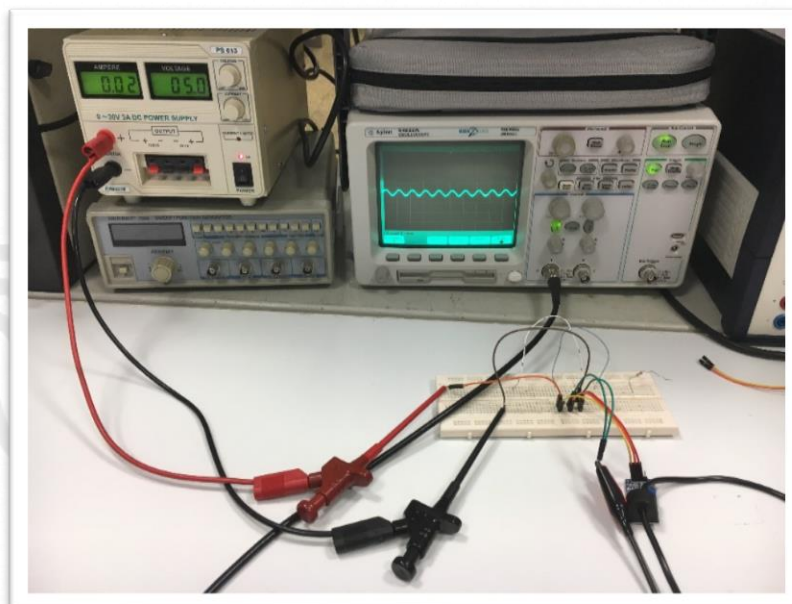
pin. (Ideale situatie: 1 mA zorgt voor 1V over de weerstand (belasting) als de weerstand 1 kilo ohm bedraagt, zonder rekening te houden met spanningsvallen over de diodes)



Hier werd de stroommeting getest met een wisselspanningsbron en een gloeilamp, hiermee konden we onze microcontroller kalibreren door ook te meten met de nauwkeurige stroommeter. Onze waarnemingen met verschillende belastingen in het stroommetingscircuit:

Belasting	Spanningsval bij 1A op net	Spanningsval bij 2A op net
500 Ω	1.44 V	2.82 V
1000 Ω	2.75 V	5.38 V
2000 Ω	5.47 V	10.13V

Verdere berekeningen om de stroommetingen te kalibreren worden uitgevoerd door de microprocessoren van de slaves.

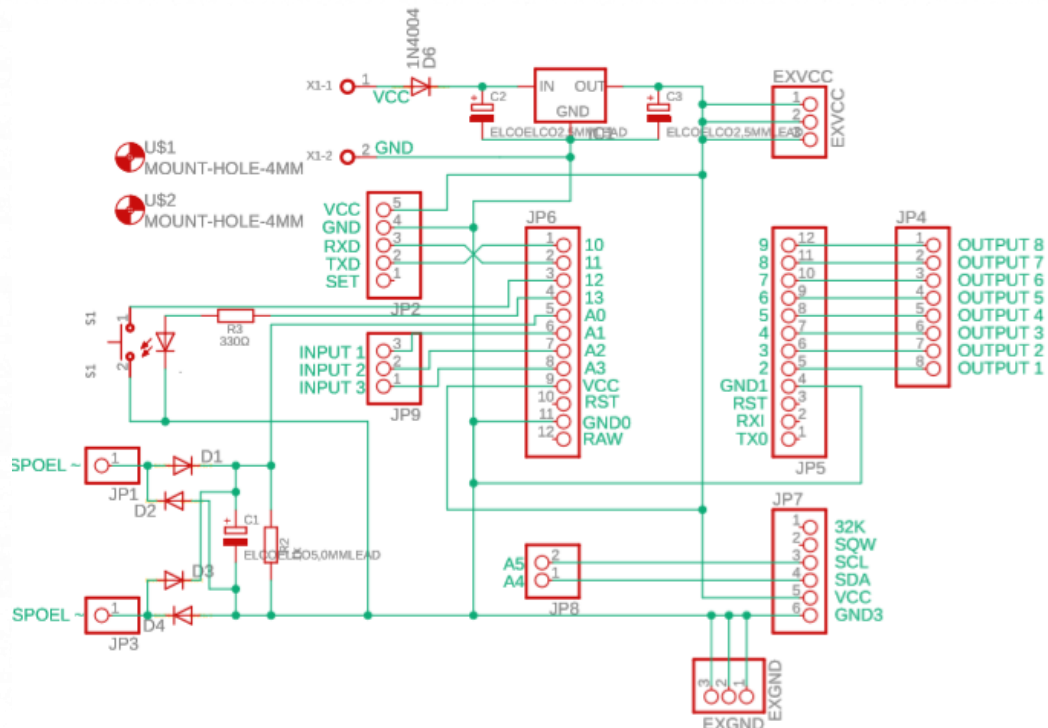


5.1.3 Professioneel PCB

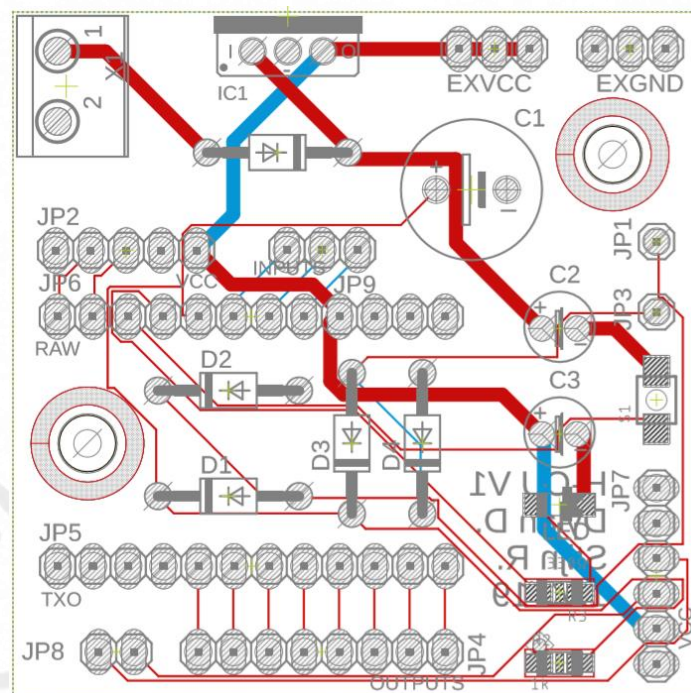
Toen we er zeker van waren hoe we wilden dat ons eindresultaat eruit ging zien en welke doelen het moest volbrengen, moesten er schema's en PCB's (printed circuit board) gemaakt worden. Dit heeft Dylan Dedapper uiteindelijk gedaan met Autodesk Eagle.



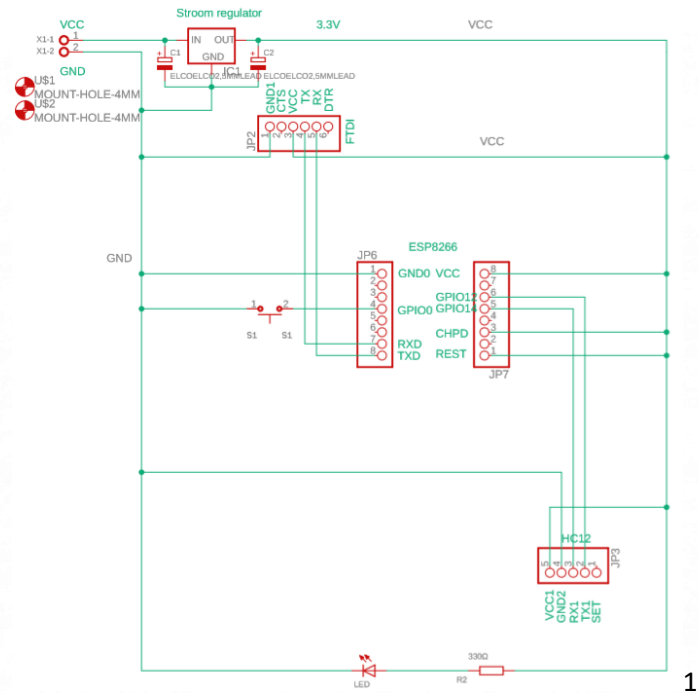
Schema slave versie 1



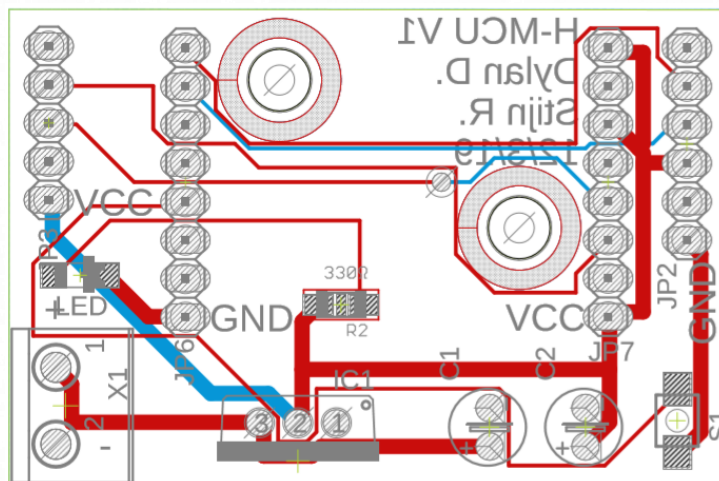
Bord slave versie 1



Schema master versie 1

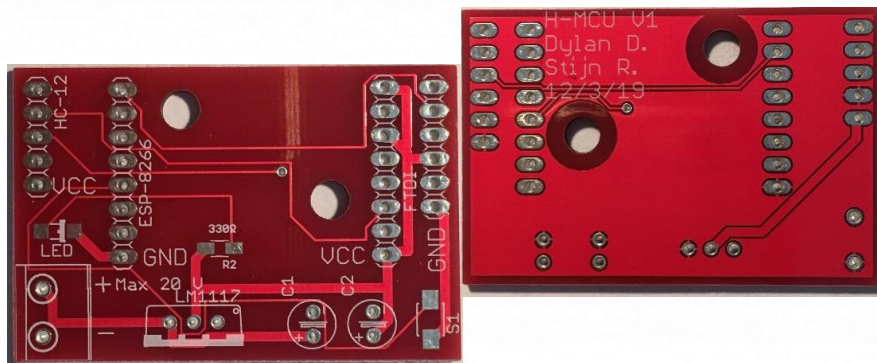
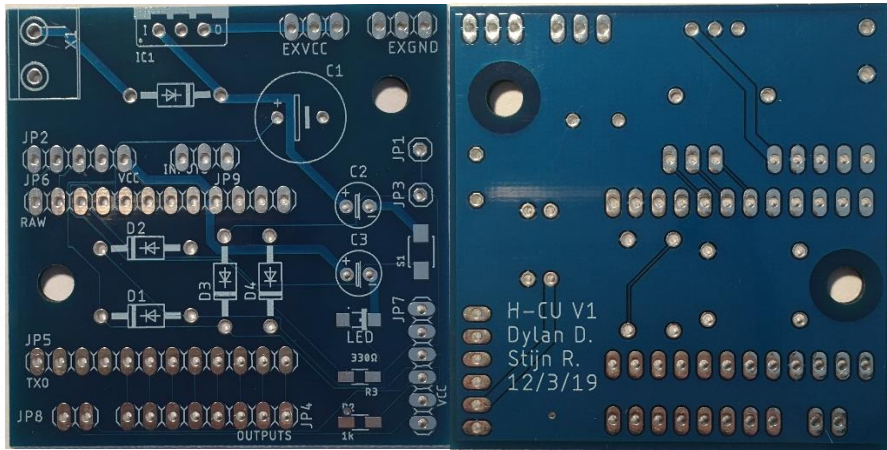


Bord slave versie 1



Toen onze PCB's en schema's op papier stonden, konden we ze laten maken. Door deze bestanden te exporteren en door te sturen naar een PCB fabrikant in Hong Kong, kwamen de bordjes 2 weken later toe in ons klaslokaal. Op het eerste zicht zag het er prachtig uit (blauw = slave, rood = master):

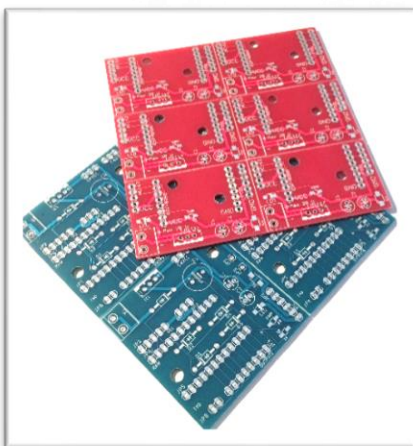
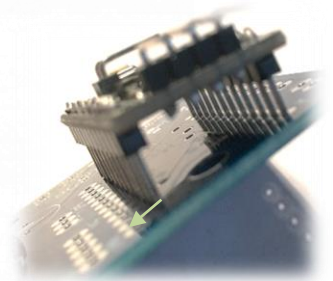




Er was al snel iets mis met het slave bordje, de microcontroller paste niet in zijn plaats (te zien hier rechts op de foto), de afstand was incorrect.

De master had ook een probleem: de socket voor de microcontroller was gespiegeld en kon dus niet correct aangesloten worden aan zijn benodigde componenten.

Beide bordjes waren dus onbruikbaar.

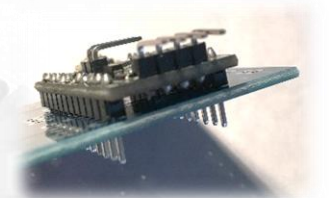


Master en slave versie 2

Na de problemen die we hadden met de eerste bordjes, moest er een vervolg komen dat wel klopte, daarom waren de bordjes al snel aangepast en terug naar Hong Kong gestuurd.

Na 3 weken kwam versie 2 van de slave en de master binnen, en het was tijd om te solderen.

De master werkte na vele testen succesvol en dit kon ons prototype bordje vervangen!

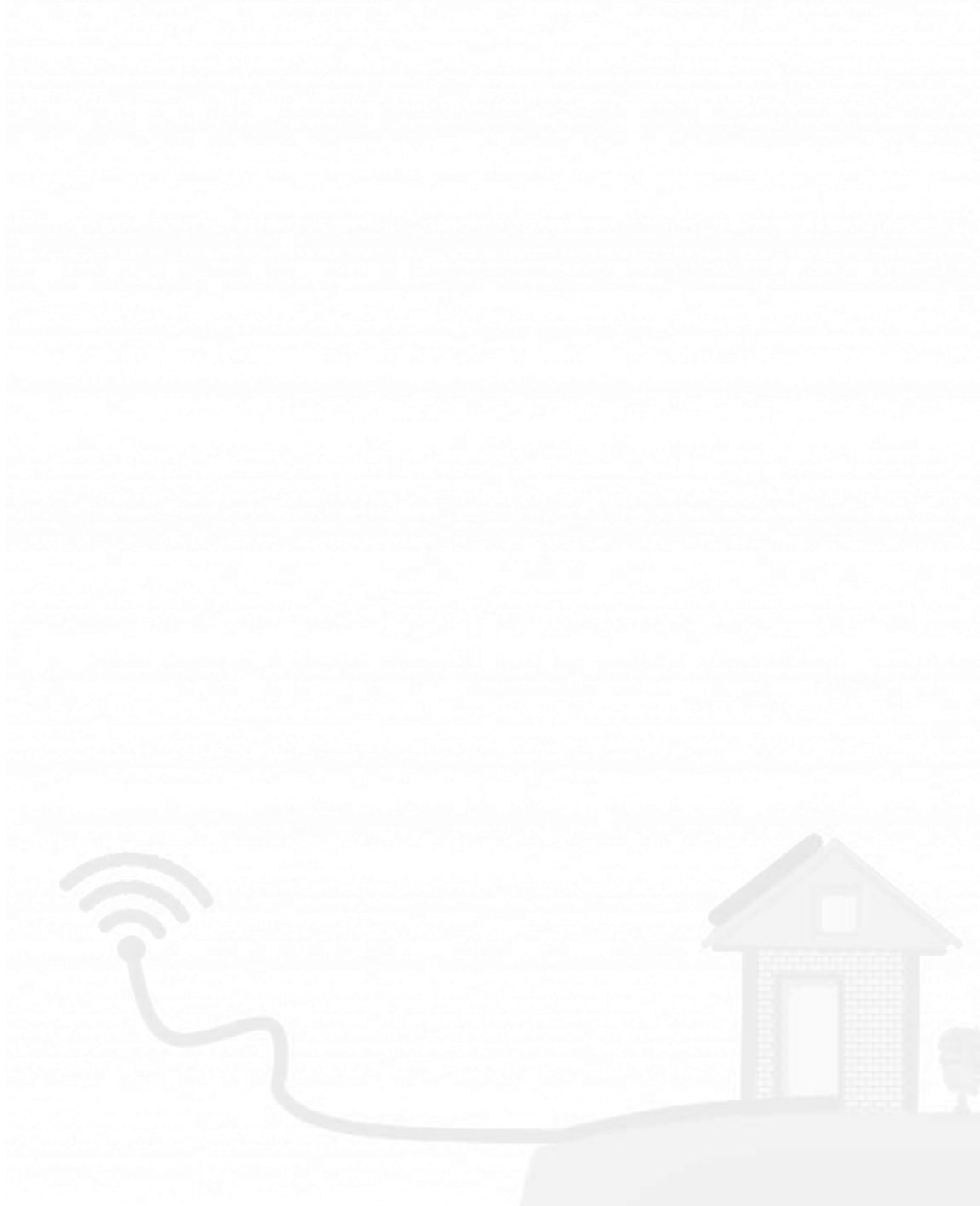
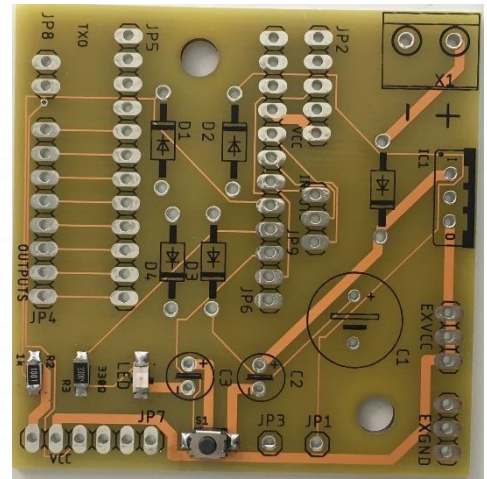


Maar de slave had nogmaals een probleem: na het solderen zag ik dat de socket voor onze microcontroller gespiegeld was. Dit maakte onze slavemodules onbruikbaar en Dylan moest opnieuw een correct bordje ontwikkelen.

Slave versie 3

Na al deze problemen kwam versie 3 binnen. Deze zou alle problemen moeten oplossen die we met vorige bordjes hadden: gespiegelde en verplaatste sockets.

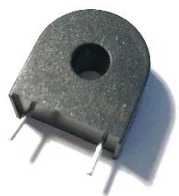
Dit bordje was geslaagd voor alle testen die het doorging: de stroommeting, spanningsregelaar, status-led, socket, HC12 connectie, DS3132 connectie ... Alles werkte correct.



5.1.4 Componentenlijst

Als we een kijkje nemen naar alle componenten die we hebben gebruikt in ons eindresultaat, krijgen we deze opsomming:

Foto	Naam	Gebruik
	Voltage regulator LM1117 (3.3V) LM1117 (5V)	Deze IC gaat ervoor zorgen dat elke microcontroller een juiste en stabiele ingangsspanning gevoed krijgt.
	Female headers	Hierin kunnen we onze microcontrollers prikken en kunnen we kabels doorverbinden. Dit wordt ook gebruikt als socket voor onze microcontroller.
	Aansluitklemmen	Deze klemmen gaan er voor zorgen dat we externe aansluitingen kunnen garanderen. (vb: om een voedingsspanning te voorzien)
	Elektrolytische condensatoren: 1 mF 47 μ F	De 1 millifarad condensator wordt gebruikt als signaal-afvlakker bij de stroommeting en de andere is om storingen weg te filteren.
	SMD Leds	Kleine leds die zowel bij de master als bij de slave zullen aangeven wanneer deze actief zijn.
	SMD Weerstanden: 1 k Ω 330 Ω	Deze mini weerstanden worden gebruikt als belasting bij de stroommeting en worden gebruikt als stroomlimiet om leds aan te sturen.
	1N1184 schottky diode	Deze diodes zullen gebruikt worden om de gelijkrichter samen te stellen voor de stroommeting. Ook gebruikt om inverse aansluiting tegen te gaan.
	SMD drukknop	Een kleine drukknop die zich zal bevinden op zowel master als slave. Om master in programmeermodus te brengen en bij slave om te koppelen.
	IRLB8721	Deze N-channel MOSFET wordt gebruikt om een belasting aan te sturen via een digitale of analoge pin bij de slave. (vb: een ledstrip)



Transformator
HWCT (5A/5mA)

De transformator die wordt gebruikt om de stroom op het net te meten. De netkabel hoort door de opening te gaan. De transformator neemt de magnetische velden van de kabel op en zet het om in een spanning.



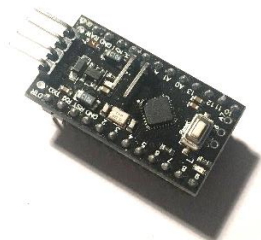
HC12

De radiomodule die de antennes aanstuurt en de draadloze communicatie tot stand brengt. Gebruikt bij alle slave en master modules.
(Zie communicatie)



ESP8266

De microcontroller die wordt gebruikt bij de mastermodule.
(Zie microcontrollers)



Arduino Pro Mini

De microcontroller die wordt gebruikt bij de slavemodules.
(Zie microcontrollers)



DS3231

De module die gebruikt wordt bij de slaves om tijd en temperaturen op te meten.



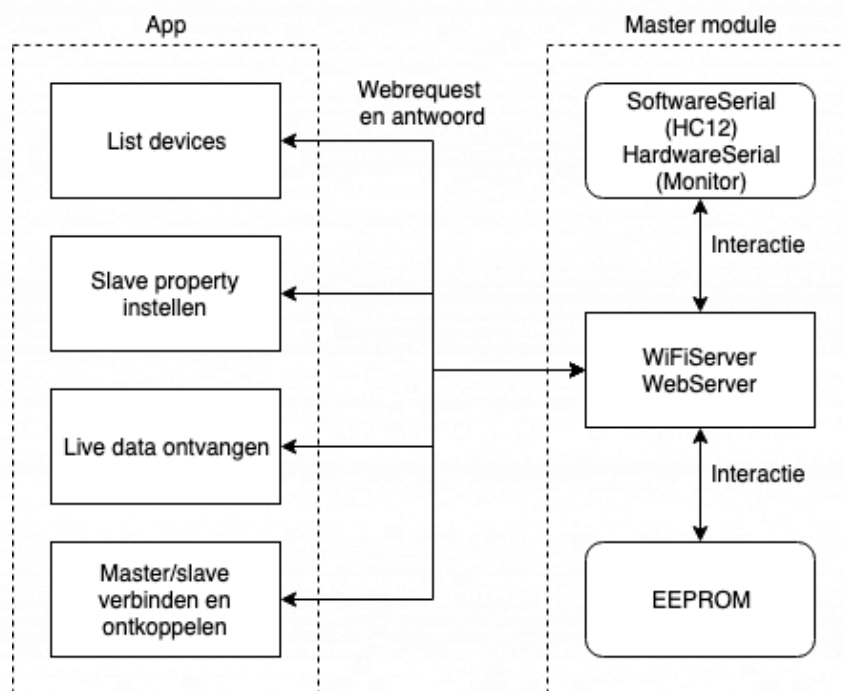
6 Design en afwerking

6.1 Smartphone app

De smartphone app is ook een groot deel van het systeem, hij moet eenvoudig en krachtig zijn. Een opsomming van functies die onze app zeker moet bevatten:

- Een overzichtelijke lijst van alle geregistreerde slaves.
- De status van elke slave kunnen bekijken: offline of online?
- Stroommetingen kunnen zien per slave.
- Slaves registreren en registratie van slaves ongedaan maken.
- Snel en eenvoudig slave commando's sturen: licht aan, licht uit.

Een technisch schema van deze doelen wordt op deze manier opgesteld:



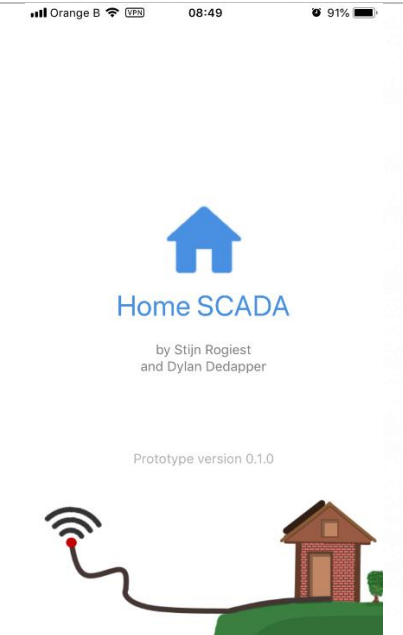

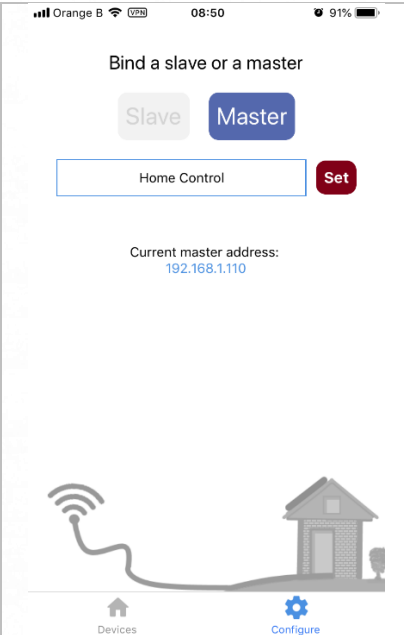
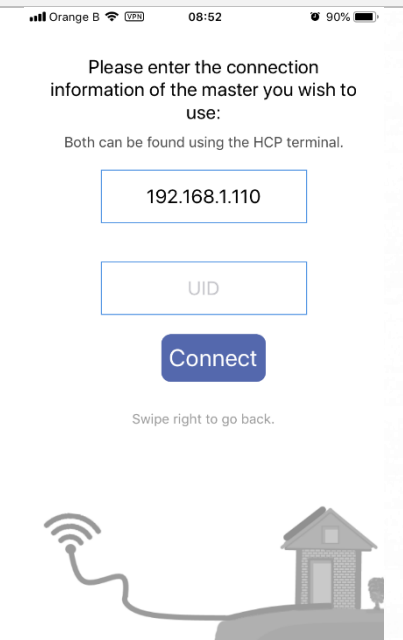
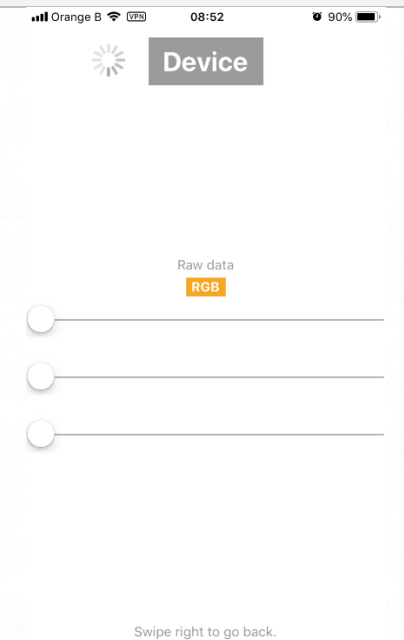
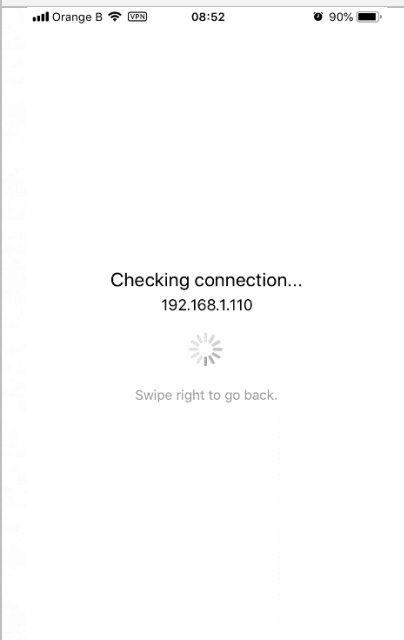
Eerst werd de smartphone app ontwikkeld met Android Studio en Java, maar dit nam meer tijd in dat verwacht, dus ben ik overgeschakeld naar <https://thunkable.com/>. Via dit platform kan je snel en eenvoudig apps bouwen voor zowel Android en iOS.



Zie volgende pagina voor resultaten.

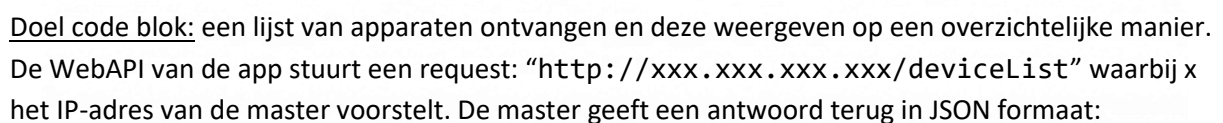


Na vele weken werk en wat problemen met het platform, zijn we tot de volgende resultaten gekomen:

		
<p>Start scherm, hier is auteurs- en versie informatie.</p>	<p>Scherf dat een live lijst ophaalt met alle apparaten. Klik op een apparaat om informatie op te halen.</p>	<p>Configuratiescherf, op dit scherm kan er van alles ingesteld worden.</p>
		
<p>Een scherm dat ervoor zorgt dat een master gekoppeld kan worden met de smartphone app.</p>	<p>Scherf dat wordt weergegeven wanneer een apparaat wordt aangeklikt. Hier kunnen alle 'properties' van het apparaat worden ingesteld. (In dit geval sturen de RGB sliders een ledstrip aan)</p>	<p>Dit scherm laat weten wat er gebeurt tijdens het koppelen van een master of een slave.</p>

(Niet alle toegankelijke schermen zijn weergegeven)

De onderstaande code request een lijst van apparaten van de gekoppelde master. De master module stuurt een geformatteerd JSON bestand terug met vele informatie over alle slaves.



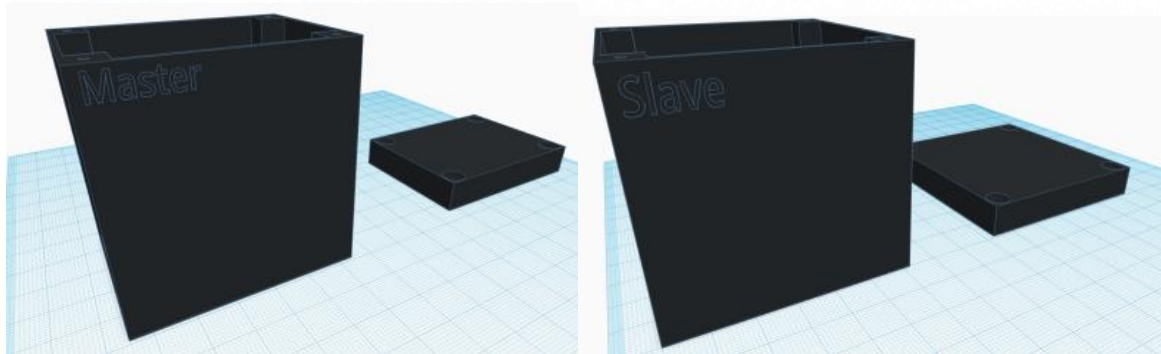
Deze JSON indeling wordt hierna omgezet naar een object die de app op vele manieren zal kunnen gebruiken.

28

6.2 Behuizing

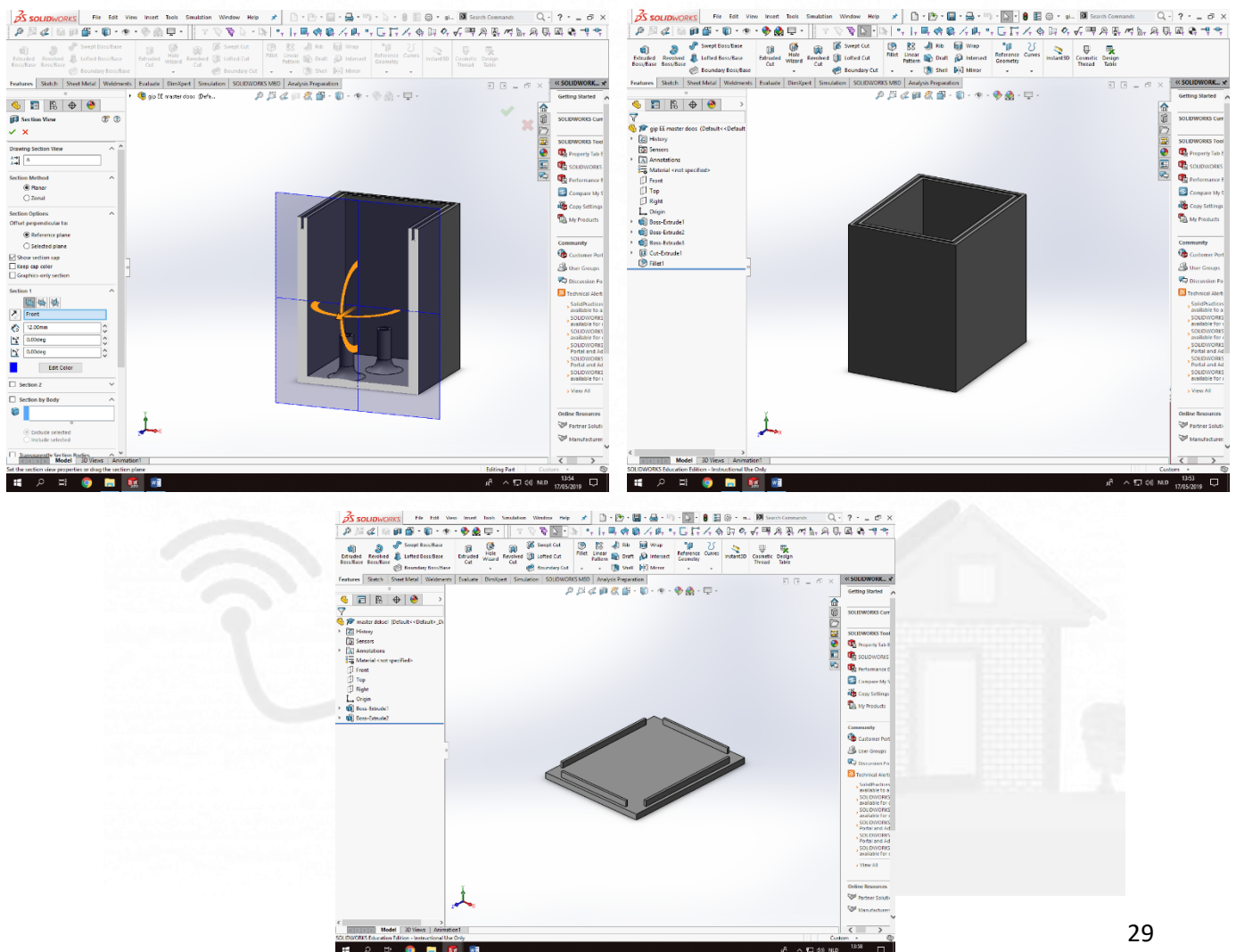
Toen onze bordjes klaar waren, hadden we een plaats nodig om ze netjes in op te bergen. Daarom hebben we gezorgd voor een 3D-geprinte behuizing om zowel onze slave als onze master in op te bergen. De 3D prototypemodellen zijn gemaakt met <https://www.tinkercad.com/> door Dylan Dedapper.

Prototype behuizing master & slave

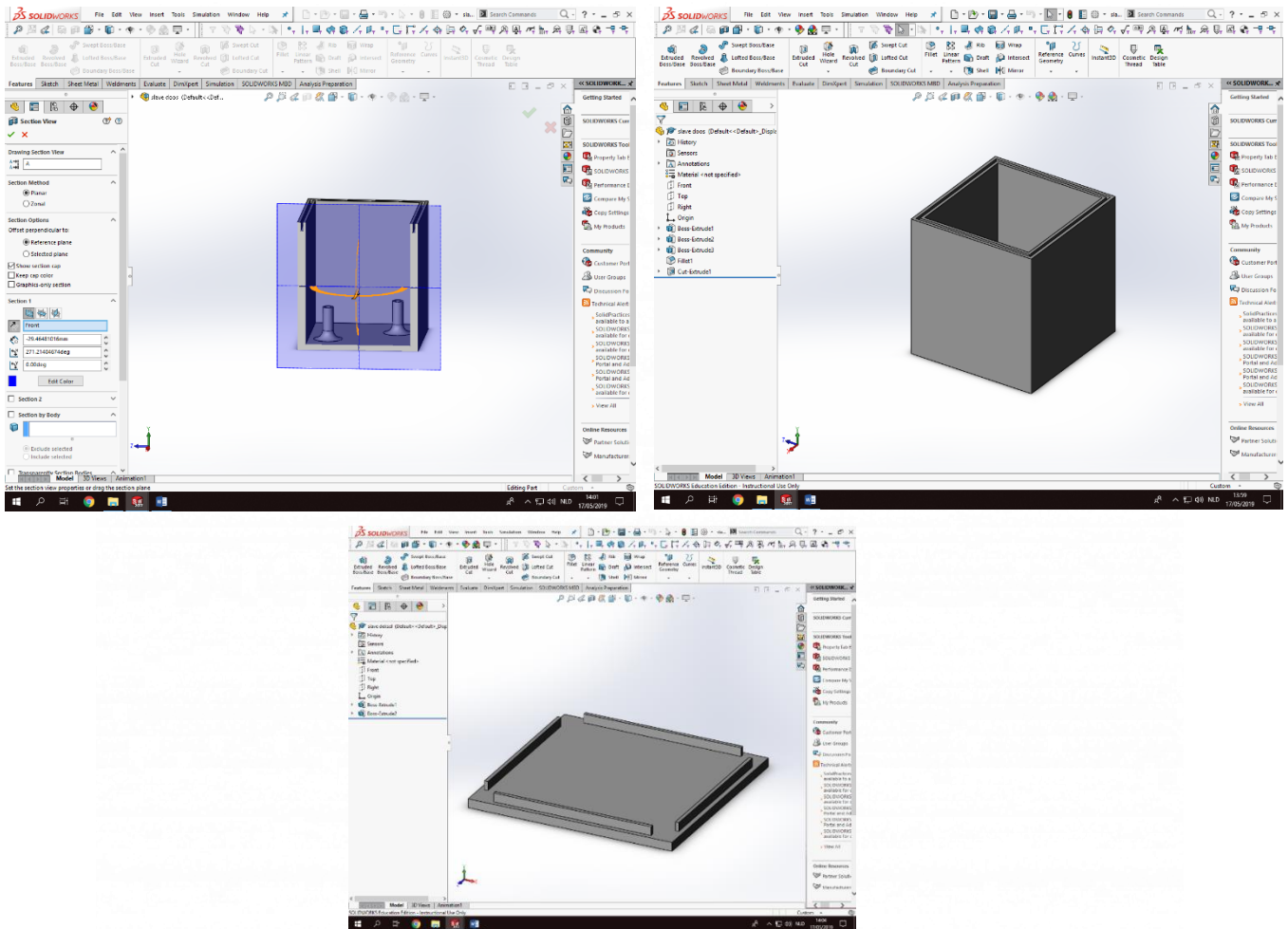


Nadat deze prototype modellen klaar waren, gingen we in gesprek met de mechaniekklas en hun leerkracht om te vragen om ons model af te printen, hier bleek uit dat het model nog veel fouten bevatte (zwevende balken, gaten, te dunne wanden). Dus gingen Sander Dornon en Tony De Bonte aan de slag om een correct 3D model te ontwikkelen met <https://www.solidworks.com/>.

Afgewerkte behuizing master

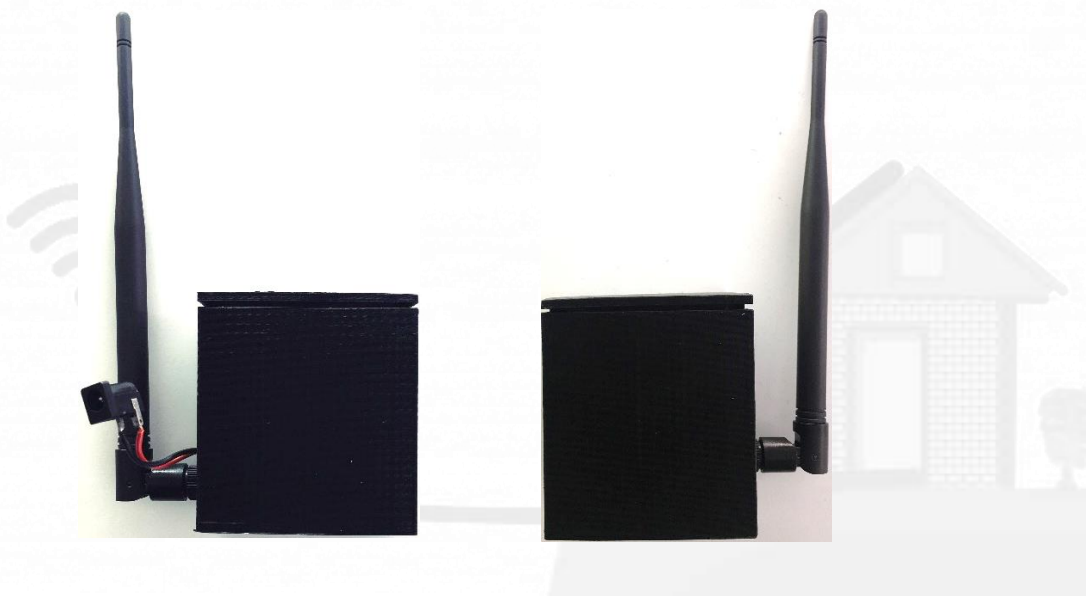


Afgewerkte behuizing slave



De afgewerkte doosjes hadden montagegaten voor onze bordjes en een aanklikbaar deksel. Geen schroeven waren dus nodig om het deksel op het doosje te monteren.

Resultaten doosjes master & slave



7 Bronnenlijst

Hieronder een concrete lijst van internetbronnen die ik heb gebruikt bij het werken aan dit project.

- <https://tttapa.github.io/ESP8266/Chap10%20-%20Simple%20Web%20Server.html>
- <https://www.arduino.cc/en/Reference/EEPROM>
- <http://www.cplusplus.com/doc/tutorial/pointers/>
- <https://www.arduino.cc/en/Reference/softwareSerial>
- <https://stackoverflow.com/questions/3698043/static-variables-in-c>
- <https://randomnerdtutorials.com/esp8266-web-server/>
- http://arduino.esp8266.com/stable/package_esp8266com_index.json
- https://en.wikipedia.org/wiki/Multicast_DNS
- https://en.wikipedia.org/wiki/Cyclic_redundancy_check#CRC-32_algorithm
- <https://www.elecrow.com/download/HC-12.pdf>
- <https://en.wikipedia.org/wiki/Baud>
- https://nl.wikipedia.org/wiki/Globally_unique_identifier
- https://en.wikipedia.org/wiki/Omnidirectional_antenna
- https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter
- <https://nl.wikipedia.org/wiki/Pulsbreedtemodulatie>
- https://nl.wikipedia.org/wiki/Supervisory_control_and_data_acquisition
- <https://pdf1.alldatasheet.com/datasheet-pdf/view/1009957/ISC/IRLB8721.html>
- <https://datasheets.maximintegrated.com/en/ds/DS3231.pdf>

Ik schrijf natuurlijk niet alle links op die ik bezocht heb, dat zou een enorm tijdverlies zijn en het zou niemand interesseren, dit waren enkele belangrijke links die ik had genoteerd.

Alle flowcharts zijn door mij gemaakt met behulp van <https://www.draw.io/>.

De mobiele app voor zowel Android en iOS is gemaakt met <https://thunkable.com/>. Alle achtergronden zijn ook door mij gemaakt.

De 3D-modellen zijn gemaakt door Dylan Dedapper met <https://www.tinkercad.com/>. Hierna aangepast door Sander Dornon en Tony De Bonte met <https://www.solidworks.com/>.

Al mijn bestanden (code, documentaties, presentaties, flowcharts, schema's ...) die ik heb gebruikt doorheen het jaar en zowel in deze bundel, staan op **GitHub** (auteursrechten zijn actief):

<https://github.com/CodeStix/Home-Control-GIP>



8 Besluit en zelfreflectie

In dit schooljaar is er enorm veel gewerkt aan de GIP, op school maar ook zeker thuis. Na deze ervaring is er zeker nog iets dat ik kwijt wil.

Het was slopender dan verwacht: in een team de taakverdeling en deadlines navolgen was de grootste uitdaging die er was voor mij. Ik raad het hierbij ook niet aan om dit soort opdracht in team uit te voeren zonder hiermee goed om te kunnen gaan. Het was moeilijk, zeker omdat dit mijn eerste serieuze groepsproject was. Het eens zijn over keuzes, communicatie en bereikbaarheid, eerlijke taakverdeling, taken pushen zonder teamconflicten te krijgen, het is zeker niet evident. De volgende keer zal ik beter weten wat er mij te wachten staat.

Maar dat betekent niet dat het niet waard was! Zeker niet! Ik heb enorm veel plezier gehad doorheen het jaar. Ik kan over mijzelf zeggen dat ik erg veel heb bijgeleerd, zowel op het vlak van hardware als software, en dat vind ik belangrijk. Zo had ik in het begin van het schooljaar nauwelijks ervaring met object georiënteerde C++ en pointers, ik kan u vertellen dat ik dit zeker onder de knie heb! Apps ontwikkelen met Java en Kotlin in Android Studio, ik had hier nog nooit mee gewerkt, zo zie je toch dat we zijn gekomen tot prachtige resultaten! Zo zie je nu eenmaal dat het niet uitmaakt als je iets niet kan, doorzetten is hierbij de sleutel tot succes. Laat jezelf niet te snel in de steek en als je het nu eenmaal haalt, heb je eigenlijk stiekem van het leerproces genoten.



9 Logboek

Schoolweek : 41

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
14/10/2018	EE	Nagedacht over mogelijke projecten, drank dispenser?

Schoolweek : 42

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
19/10/2018	EE	Besproken met klasgenoot, beslist project: Smart Home.
20/10/2018	EE	Brainstormen van hardware modules en protocol(HCP) tussen modules + schema uitgewerkt.
21/10/2018	EE	Mail opgemaakt en doorgestuurd naar leerkrachten. Zie bijlage.

Schoolweek : 43

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
24/10/2018	EE	PROTOTYPE 1: communicatie tussen modules: led aansturen via button.
25/10/2018	EE	INFOAVOND GIP: presentatie GIP in de klas samen met ouders, klasgenoten en met de leerkracht, demo (jammer genoeg kortsluiting gemaakt).
26/10/2018	EE	Troubleshooting modules want kortgesloten.

Schoolweek : 44

Datum	Vak	Omschrijving van de studie, taak of voorbereiding.
30/10/2018	EE	HCP verder uitgewerkt.
1/11/2018	EE	HCP verder uitgewerkt, SLE? Veiligheid bij het versturen.

Schoolweek : 45

Datum	Vak	Omschrijving van de studie, taak of voorbereiding.
5/11/2018	EE	Troubleshooting modules.
6/11/2018	EE	Troubleshooting modules.
9/11/2018	EE	Direct aansluiten van een esp8266 module.
11/11/2018	EE	Direct aansluiten van een esp8266 module, meer research + FTDI.

Schoolweek : 46

Datum	Vak	Omschrijving van de studie, taak of voorbereiding.
12/11/2018	EE	Modules zijn gerepareerd, we kunnen verder.
13/11/2018	EE	Maken van verschillende prototypes om te testen + test programma uploaden.
14/11/2018	NED	Zakelijke mail naar bedrijf voor informatie te verkrijgen over verschillende elementen die we zullen gebruiken bij onze GIP.
15/11/2018	EE	Verschillende opties in de HC12 uittesten met AT-commando's, vaststelling klein bereik?
16/11/2018	EE	Succesvolle connectie tussen modules met adressering.
17/11/2018	EE	Research i.v.m. HC12 bereik.

Schoolweek : 47

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
19/11/2018	EE	HC12's ingesteld op een baud van 4800 en op mode FU3.
20/11/2018	EE	Hardware van verschillende modules aangepast en op dezelfde pinnen ingesteld.
22/11/2018	EE	Aanpassing aan het protocol 2 bytes -> 4 bytes (startbyte, van, naar, data).
23/11/2018	EE	Aanmaken van gemeenschappelijke library voor master en slave + direct aansturen van esp8266.

Schoolweek : 48

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
26/11/2018	EE	Verder werken aan gemeenschappelijke library met C++, header file + source file. Pro mini kan niet werken met HC12 module?
27/11/2018	EE	Library klaar voor gebruik; invoegen in programma van de master en van de slave. Eerste test mislukt (compiler fouten).
29/11/2018	EE	Fouten in library gezocht en gevonden, de 'constructor' van 'SoftareSerial' moest 'geinheriteerd' worden met de 'constructor' van onze library. De fouten zijn opgelost.
30/11/2018	EE	Uitbereiding programma H-MCU: 'console' systeem via de seriële monitor, hiermee kunnen we makkelijk fouten zoeken, vinden en hierna oplossen. Probleem Pro Mini ontdekt, pin verkeerd aangesloten, hierbij opgelost en getest met ons nieuw 'console' systeem.
2/12/2018	EE	Integer to bytes; bytes to integer, script geschreven voor onze goede vriend Jorik.

Schoolweek : 49

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
3/12/2018	EE	Nagedacht over wat er in de kerstvakantie zou moeten gebeuren; Dylan maakt een schema voor de master + voor de slave.
4/12/2018	EE	Stroomdetectie door een kabel, het gebruikt van een gelijkrichter en een grote condensator om deze informatie in te lezen op een analoge pin op de Arduino. (zie bijlage 3)

Schoolweek : 1

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
7/01/2019	EE	1 uur : Verder testen van de stroom detectie en programma kalibreren. + stroomversterker gekregen + testen aan de scope (zie bijlage 2)
11/01/2019	EE	2 uur : Nogmaals testen van de connectie tussen modules en start van stroomdetectie te integreren in ons programma. 1 uur : Afwerken van de Powerpoint voor GIP presentatie voor de jury.

Schoolweek : 2

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
14/01/2019	EE	Dylan start met solderen van een prototype van de master-module op een voorgekerfde pcb. Stijn werkt verder aan de programmatie van het protocol.
15/01/2019	EE	Solderen van master-module, nadenken over lay-out van het pcb-tje. Stijn verder gewerkt aan protocol: library andere naam gegeven + prototype van 'resending' data.
17/01/2019	EE	Coppejans geholpen met corrupte bestanden. Dylan ziek.
18/01/2019	EE	Coppejans geholpen met corrupte bestanden, gefixt.

Schoolweek : 3

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
21/01/2019	EE	Dylan; solderen van een slave module, Stijn; nagedacht over het opnieuw versturen van gegevens, na 1 seconde opnieuw sturen wanneer geen respons van slave.
22/01/2019	EE	Dylan; slave boardje afgewerkt + test. Stijn; testen van het opnieuw versturen: succesvol. Testslave maken met 4 aangestuurde relais. Relais sturen verschillende lampen aan.
24/01/2019	EE	Stijn; HTTP server geïntegreerd in de master module + test van een GET-request, het maken van een simpele website met een simpele interface.
25/01/2019	EE	Dylan ziek. Stijn; zorgen voor een input form (HTTP server) voor het aansturen van alle modules i.p.v. 2 simpele knoppen voor het aansturen van maar 1 module. + aansturen van RGB led-strip via de seriële monitor.

Schoolweek : 4

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
28/01/2019	EE	Ledstrip aansturen via MOSFETs aan de slave, externe kleurselectie via de master realiseren. Aanpassing protocol: het gebruiken van een databuffer van 256 bytes.

29/01/2019	EE	Aanpassing aan het protocol voor het aanpassen van de databuffer aan de slave-kant, verschillende functies toevoegt, SET-commando, FETCH-commando, om data te verkrijgen en data in te stellen.
31/01/2019	EE	Aansturen van een adresseerbare RGB led strip via een Arduino met een specifieke library namelijk <i>FastLed</i> .

Schoolweek : 5

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
5/02/2019	EE	Stroommeting realiseren zonder versterkerbordje, rauw inlezen op de Arduino.
7/02/2019	EE	Protocol aanpassen voor SET-commando, slave module zou mogelijk kunnen crashen bij slecht bereik en een SET-commando.
8/02/2019	EE	Nogmaals protocol aanpassen.

Schoolweek : 6

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
4/02/2019	EE	SET-commando gerealiseerd, deze stelt een waarde in een lijst van 256 variabelen die gebruikt kunnen worden voor verschillende dingen bij de slave.
5/02/2019	EE	Uploaden van de nieuwe slave-programma's naar alle slaves met unieke adressering. + testen en debuggen van SET-commando, realisatie vastlopen bij gefaalde dataoverdracht.
7/02/2019	EE	Debuggen van SET-commando. + eerste succesvolle test van RGB-ledstrip aansturen met SET-commando i.p.v. gewone dataoverdracht zonder SET.
8/02/2019	EE	Uitstap technologiebeurs Kortrijk-Xpo

Schoolweek : 7

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
-------	-----	---

11/02/2019	EE	Library van protocol uitbreiden, zowel hardware-serial en software-serial is geïntegreerd en is selecteerbaar. HTTP-server pagina uitgebreider gemaakt, nagedacht over een layout voor het SET-commando (slider?, RGB-sliders?).
12/02/2019	EE	Aanpassing gemaakt aan protocol: het resenden werkt op een instelbaar interval (standaard 625ms), het is ook uitschakelbaar. Integratie in de Aduino library.
14/02/2019	EE	Stroommeting overleg met Dylan, hij gaat 2 aparte schemas maken. Een slave met stroommeting en een slave zonder stroommeting. Ik werk de testsoftware voor stroommeting af en ik doe nog wat kalibraties m.b.v. multimeter.

Schoolweek : 8

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
18/02/2019	EE	Bespreken met Dylan over PCB-tjes, welke lay-out, hoe voeden, grootte ...
19/02/2019	EE	Verschillende slave-modules klaargemaakt met een verse upload en setup voor de presentatie van donderdag.
21/02/2019	EE	Vorbereiding presentatie + probleem met uploadsnelheid opgelost: uploaden met 5V op 16MHz
22/02/2019	EE	Protocol: SET-commando veilig gemaakt: niet vastlopen bij gefaalde overdracht van data. + nagedacht over SET-RANGE-commando om meerdere SET-commando's in 1 keer uit te voeren.

Schoolweek : 9

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
25/02/2019	EE	SET-RANGE en SET-commando verder uitgevoerd: het wachten op een OK feedback voordat er data mag worden doorgestuurd.
26/02/2019	EE	SET-RANGE en SET-commando verder uitgevoerd: ontvangen van het aantal data dat nodig is voor het SET-RANGE commando + het verzekeren van ontvangst.
28/02/2019	EE	Overleg met Dylan voor het finaliseren van de master en slave PCBs. Dylan laten werken aan de computer wegens gebrek aan computer, laptop word voor de volgende keer geregeld. Ondertussen andere medeleerlingen geholpen.

Schoolweek : 11

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
12/03/2019	EE	Het herschrijven van het home control protocol: redenen: betere methoden, meerdere requests in 1 keer toestaan, gebruik maken van klassen, herkennen van slave modules en naamgeving, was niet mogelijk in de vorige versie en implementeren was moeilijk ...
13/03/2019	EE(CS)	Verder herschrijven van het protocol.

Schoolweek : 12

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
19/03/2019		Bekijken GIP-dossier. Extra informatie vragen aan leerkracht: wat wordt er verwacht tegen wanneer?
20/03/2019	EE(CP)	Verder werken protocol: testen met een c++ programma, Packet klas, Request klas, compileren met G++ compiler.

Schoolweek : 13

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
25/03/2019	EE	Verder uitwerken van protocol in C++. Programma installeren voor ontwikkeling app.
26/03/2019	EE	Verder uitwerken van protocol in C++, de eerste succesvolle testen.
28/03/2019	EE	Verder uitwerken van protocol in C++. Jorik helpen met yagi-antenne, afstandstest.
31/03/2019	EE	Een test app was gemaakt. Met wat ervaring opnieuw begonnen met een nieuwe app, een frisse start met nieuwe ideeën.

Schoolweek : 14

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
1/04/2019	EE	Verder werken aan protocol. Standaard C-code compatibel maken met de Arduino C++ variant.

2/04/2019	EE	Duidelijke bespreking met Dylan over wat er moet gebeuren in de paasvakantie. Dylan zal enkel zorgen voor een 3D model voor de behuizing van onze slave-modules. (en misschien ook master) Ik zorg voor de benodigde afmetingen.
5/04/2019	EE	Samenrapen van elektronica die ik mee zal meenemen naar huis om verder te kunnen werken. Dit bevat alle werkende slave- en master modules.

Schoolweek : 15

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
13/04/2019	EE	Verder werken aan protocol: porten van C-code naar de Arduino C++ variant. Probleem met String-formatering: alternatief zoeken voor <i>sprintf()</i> die compatibel is met Arduino. Eigen formateer methode uitgevonden.
14/04/2019	EE	Afwerken van de port van C naar de Arduino C++ variant. De eerste C-code kon gecompileerd worden zonder Arduino, dit maakte ontwikkeling van het protocol veel efficiënter want er was geen upload tijd verlies.

Schoolweek : 16

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
17/04/2019	EE	Het protocol uitbreiden, praktische situaties invoeren, klein starten: een slave-module die een led aanstuurt. Alleen maar succesvolle resultaten wegens de nauwkeurige ontwikkeling van het protocol, hier is veel beter over nagedacht.
18/04/2019	EE	Het ontwikkelen van een tijdschema systeem dat datapakketten opnieuw verstuurd wanneer een slave-module een datapakket ontvangt met de verkeerde checksum. Elk datapakket dat is verstuurd door het nieuwe frisse protocol bevat een 2-bits checksum.
19/04/2019	EE	Het invoeren van een 'property'-systeem, de master kan verschillende variabelen van slaves manipuleren in 1 keer. Praktisch voorbeeld: de kleuren van een ledstrip aansturen, i.p.v. rood, groen en blauw waardes apart te versturen, verstuurd het nieuwe protocol dit allemaal in 1 datapakket. Elk datapakket kan een verschillende lengte hebben. De maximale grootte van 1 datapakket is 20 bytes, minimum 4 bytes.

20/04/2019	EE	<p>Een systeem ontwikkelen dat kan uitvinden of slaves online zijn of niet. De master stuurt om de 5 seconden een 'ping' naar elke slave, elke slave dat antwoord is online, de anderen zijn offline.</p> <p>Elke slave zijn 'properties' worden vanaf nu opgeslagen in EEPROM. Net zoals hun adres en unieke fabriekscodes. De master gebruikt het EEPROM om de slaves te onthouden, zodat deze niet elke keer vergeten worden wanneer de spanningsbron wordt ontkoppeld.</p> <p>Starten met implementeren van de webserver, hiermee kan de smartphone app (of andere applicatie) interactie hebben met het protocol.</p>
20/04/2019	GOD	GIP-opdracht godsdienst: normen op de werkvloer.
21/04/2019	EE	<p>Implementeren van webserver voltooien.</p> <p>Verder gewerkt aan de smartphone app: een knop voor apparaat te registreren en een lijst van apparaten. Deze werden opgeslagen in de master module en opgehaald via de webserver.</p>

Schoolweek : 19

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
6/05/2019	EE	Herinstellen van alle HC12 modules: 4800 baud met de juiste stroombesparingsmodus.
7/05/2019	EE	Verder werken aan de app in Android Studio. Upload van webserver test programma naar de master.
8/05/2019	EE	Experimenteren met de webserver van de ESP8266, verbinden met meerdere wifi netwerken: MutliWifi library.
9/05/2019	EE	Verder werken aan de app in Android Studio. Realisatie tijdsnood bij app maken? Zoeken naar een ander platform?

Schoolweek : 20

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
-------	-----	---

13/05/2019	EE	Nieuw platform voor appontwikkeling: Thunkable, snelle voortuitgang. Dit platform kan exporteren naar zowel Android als iOS.
14/05/2019	EE	Naar mechanica voor het 3D model van de behuizing voor zowel de master als voor de slave: er zijn nog veel fouten in het 3D model: zwevende balken, gaten, dunne wanden ... Sander zal dit oplossen samen met meneer De Bonte.
16/05/2019	EE	Ontwikkeling app voortzetten: scherm voor het koppelen van master en pictogrammen zijn geïntegreerd. Nieuwe webrequests invoeren in de webserver van de master module: properties instellen en ping.
17/05/2019	EE	Ontwikkeling in app voortzetten: scherm voor het testen van de verbinding met de master en een opstartscherm die alle nodige opgeslagen variabelen laadt zoals IP-adres van de master.

Schoolweek : 21

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
20/05/2019	EE	Uploaden van alle nodige programmatuur naar slave modules, dit bevat shared en aangepaste native code.
21/05/2019	EE	Bordjes versie 3 binnengekomen met voor gesoldeerde smd componenten: Dylan gaat aan de slag met solderen, ik zorg ervoor dat ze getest worden. Er werd ook een bordje gemaakt om een ledstrip aan te sturen.
23/05/2019	EE	Naar het techniek festival: onze GIP voorstellen aan bedrijven en leerlingen.
24/05/2019	EE	Ontwikkeling app: grafische onderdelen van start scherm en lijst van apparaten. Scherm voor aansturen van apparaat maken. Succesvolle test met webserver van de master module.

Schoolweek : 22

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
27/05/2019	EE	Bespreken met Dylan over het demo-huis: in dit huis zullen onze modules worden voorgesteld aan het publiek.

28/05/2019	EE	Kritisch Thunkable probleem wegens hernamen van variabel, zo snel mogelijk een oplossing proberen vinden...
30/05/2019	Thuis	Hermaken van de app. onoplosbaar probleem door het Thunkable platform.
31/05/2019	Thuis	Hermaken van de app...

Schoolweek : 23

Datum	Vak	Omschrijving van de studie, taak of voorbereiding
3/06/2019	EE	App hermaken -> problemen zijn opgelost, werkende aansturing van een RGB ledstrip.
4/06/2019	EE	Demo huis voorbereiden voor gebruik: stabiele ondergrond en verwijdering dak zorgt voor voldoende plaats. App afwerken + RGB ledstrip meerdere testen.
6/06/2019	EE/Wisk	Afwerken van verschillende slave modules en het demo huis. Dit is de laatste schooldag dat we zullen werken aan onze GIP.
3/06/2019	EE	App hermaken -> problemen zijn opgelost, werkende aansturing van een RGB ledstrip.



10 Bijlagen

10.1 GIP-opdracht godsdienst: normen en waarden op het werkterrein

Beroepswereld:

Ik zie mijn toekomst als pure cybertechnieken, namelijk programmeur, dit betekent dat ik erg veel met computers bezig zal zijn dus de soft- en hardware die hierbij komt kijken. Het nadenken over de dingen die mogelijk zijn met computers, het uitwerken, en erg belangrijk: constant resultaten zien, hierbij blijf je gemotiveerd om verder te werken. De werkomgeving zal eigenlijk niet veel voorstellen: hoogstwaarschijnlijk een bureau en een bureaustoel en een raam met wat zonlicht. Maar dit houdt me zeker niet tegen om deze job toch te beoefenen.

Samenwerken is heel erg belangrijk als je toekomst er uit ziet zoals die van mij, dit betekent taakverdeling en elkaar constant evalueren, je moet elkaar zo snel mogelijk kunnen begrijpen, je collega's moeten dus zeker weten waar ze mee bezig zijn. 1 miscommunicatie kan meteen fataal zijn. De lat moet voor elk even hoog liggen.



Beroepsethiek:

--- *"Het nadenken over de dingen die mogelijk zijn met computers, het uitwerken, en erg belangrijk: constant resultaten zien, hierbij blijf je gemotiveerd om verder te werken."*

Dit is het proces van programmeren, het schrijven van woorden die de gemiddelde mens niet begrijpt, maar jij en de computer wel, dat is toch prachtig? De manier van denken die hierbij nodig is krijg je niet in 1 2 3 onder de knie. Het is de kunst om eenvoudig en neutraal te denken, alles in zo'n elektronisch systeem werkt stap voor stap, en op deze manier moeten je hersenen ook denken. Een computer kan niet 'denken', het is opgebouwd uit elektronische signalen die of hoog of laag staan (aan of uit). Zet een paar van deze signalen naast elkaar en je krijgt verschillende combinaties van aan- en uitwaardes. Op deze manier werkt elke computer in ons dagelijks leven, combinaties van 1(aan) en 0(uit), van usb stick tot gsm. Het getal 141 kan een computer bevooroordeeld herkennen als 10001101. Hierbij worden er 8 signalen gebruikt om het getal 141 uit te drukken.

Tekst kan natuurlijk ook vertaald worden, bv: het woord 'godsdienst' word vertaald tot 01100111011011110110010001110011011001000110100101100101011011100111001101110100.

Nog iets typisch over dit beroep: elk probleem dat je oplost worden er 2. Als je programmeertalen (je kan het vergelijken met misvormd Engels, hier rechts een voorbeeld dat ik in mijn GIP gebruik om een licht te laten knipperen, erg eenvoudig) gebruikt en scripts schrijft, ga je duizenden fouten maken, niemand is perfect en zeker niet op dit vlak. De kunst van het programmeren en omgaan met computers is probleemoplossend denken, de kunst van problemen oplossen, je gaat meer tijd steken in problemen oplossen dan vooruitgang boeken, dat is nu eenmaal zo. Dit kan soms echter zeer frustrerend zijn, maar je mag niet zomaar opgeven, dit is doorzetten! Daarom is het dus zeer

```
void loop()
{
  digitalWrite(LED_PIN, true);
  delay(500);
  digitalWrite(LED_PIN, false);
  delay(500);
}
```

belangrijk om constant resultaten te zien, zodat je gemotiveerd bent om verder te werken.

Arbeidsattitudes

Als programmeur moet je van staat zijn om zelfstandig te werken, niet afhankelijk willen zijn van iedereen, dit is tijdverspilling omdat jammer genoeg niet iedereen perfect hetzelfde kan denken. Je moet zelf de oplossing vinden voor het probleem, zelf conclusies trekken. Dit betekent niet dat je niet moet luisteren naar anderen! Zeker niet! Dat is juist erg belangrijk, ideeën uitwisselen met anderen en zelf uitwerken.

Samenwerking is belangrijk, maar taakverdeling is veel belangrijker. Hoe beter de taakverdeling en de organisatie van een project hoe beter. Dit betekent orde en naleving van de deadlines en regels. Het is ook belangrijk dat je in staat bent om mensen te helpen, want als 1 teamspeler vast zit, komt de rest ook vast te zitten, nogmaals: taakverdeling is 'key'.

Jezelf overtuigen om pauzes te nemen is ook belangrijk, urenlang staren naar lichtgevende tekst op een beeldscherm is niet super prettig.

Arbeidcompetenties

Dit is nogal moeilijk te omschrijven want er zijn honderden programmeertalen die dagelijks worden gebruikt en je kan je niet in al deze talen verdiepen. Maar natuurlijk zijn er wel een paar waar je zeker iets van af moet weten! Dit bevat talen zoals: HTML (hier is elke website op planeet aarde mee geschreven), C (of een variant hiervan, dit is de taal waarmee de computerwereld is ontstaan, Bill Gates heeft hier veel aan bijgedragen) en een object-georiënteerde programmeertaal (bv: Java, simpelweg: als een computer denken en objecten uit de echte wereld beschrijven). Deze 3 talen (of varianten hiervan, nogmaals: er zijn erg veel talen) zie ik als noodzakelijk in het dagelijkse leven van een programmeur omdat je deze zeker dagelijks zal tegenkomen!

Maar je moet niet alleen talen kunnen! Natuurlijk niet! De manier van uitvoeren is ook erg belangrijk, dit houdt in: probleemoplossend denken, doorzettingsvermogen, logisch en eenvoudig denken, goed kunnen plannen, orde ...

