

**Geïntegreerde Proef:**

**Home Control**

--- “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 Elekriciteit-Electronica kreeg ik de opdracht om een geïntrigeerde 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 gekozen om een domotica systeem te maken, dit houdt in: het eenvoudig aansturen van verschillende elektronische apparaten via een gebruiksvriendelijke interface.

Deze opdracht heb ik uitgevoerd met een klasgenoot genaamd *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.

Inhoud

[2 Inleiding 4](#_Toc6826759)

[3 Omschrijving van de opdracht 4](#_Toc6826760)

[4 Werking van het apparaat 5](#_Toc6826761)

[4.1 Algemene werking 5](#_Toc6826762)

[4.2 Uitgebreide uitleg software 5](#_Toc6826763)

[4.2.1 Bij de ‘master’ 5](#_Toc6826764)

[4.2.2 Bij de ‘slave’ 5](#_Toc6826765)

[4.3 Uitgebreide uitleg hardware 5](#_Toc6826766)

[4.3.1 Microcontrollers 5](#_Toc6826767)

[4.3.2 Communicatie 5](#_Toc6826768)

[4.3.3 Componentenlijst 5](#_Toc6826769)

[5 Design 5](#_Toc6826770)

[5.1 Elektrische schema’s 5](#_Toc6826771)

[5.1.1 Prototypes 5](#_Toc6826772)

[5.1.2 Ontwerp 5](#_Toc6826773)

[5.1.3 Eindwerk 5](#_Toc6826774)

[5.2 Behuizing 5](#_Toc6826775)

[6 Bronnenlijst 5](#_Toc6826776)

[7 Logboek 6](#_Toc6826777)

[8 Besluit en zelfreflectie 6](#_Toc6826778)

[9 Bijlagen 6](#_Toc6826779)

# Inleiding

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

In dit document zullen verschillende dingen aan bod komen:

* Eerst zul je een duidelijke omschrijving van de opdracht te horen krijgen: *wat is een ‘Home Control’-systeem? Wat was belangrijk bij het kiezen van de opdracht?*
* Dan een uitgebreide uitleg over de werking van software en hardware: *welke programmatuur was belangrijk? Hoe worden datapakketten verstuurd? Welke elektronische componenten zijn gebruikt + wat doen ze? Ontwerp van de schema’s*.
* Vervolgens 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 raad pleegde. Er is ook een bronnenlijst bijgevoegd aan het einde van dit document.

# 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 gerust stellen.

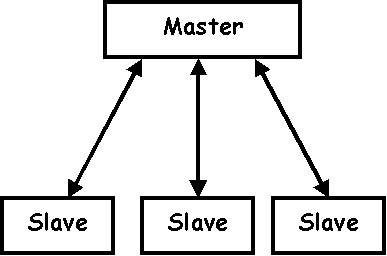
In het kort: een draadloos home-control systeem waarbij je eenvoudig en snel huishoudelijke apparaten kan aansturen via een gebruiksvriendelijke smartphone app zonder jezelf vragen te stellen over privacy.

# Werking van het apparaat

In dit deel leg ik uit hoe de hardware en de software van dit grote project samenvloeit. 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.

## Algemene werking

Ons systeem bestaat uit 2 verschillende modules, een slave en een master. Bij een typisch gebruik van dit systeem is een maar 1 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 verstuurd 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. Wanneer het interpreteren succesvol was 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 word gerealiseerd. Hierna ook op hardware niveau.

## Uitgebreide uitleg software

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

Om uiteindelijk een stabiel protocol ontwikkeld te hebben, heeft het protocol maar liefst 4 versies opgevolgd. Na de eerste 3 versies, was ik klaar om de 4de versie zo foutloos mogelijk te ontwikkelen, ik had al heel wat ervaring opgelopen met de draadloze communicatie en de programmeertaal C++, ik wist welke problemen er ontweken moesten worden. Ik ga het in dit deel dus alleen hebben over hoe de 4de versie van het protocol werkt. (hoewel ze allemaal volgens hetzelfde principe werken)

Een lijst van problemen die ontweken moesten worden in HCP versie 4 + de oplossing hiervoor:

* Er mag op geen enkel moment 2 modules op het zelfde moment data versturen: dit vergroot de kans van datacorruptie enorm. *Oplossing: een geavanceerd timings-systeem 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 zorgt dat er geen corrupte data wordt verwerkt. *Oplossing: een ‘Cyclic redundancy check’ (of checksum; https://en.wikipedia.org/wiki/Cyclic\_redundancy\_check) bijvoegen (een code die een datapakket zijn inhoud beschrijft in 2 bits, als er 1 bit wordt verstoord in het pakket, veranderd de code meteen), hiermee kan er gecheckt worden of data is gewijzigd tussen de zender en ontvanger.*
* Adressering en koppelen van slaves: in vorige versies werden slaves een adres toegewezen bij het uploaden van het softwareprogramma. Dit beperkt het aantal slaves op planeet aarde omdat elk een uniek adres moet hebben + het datapakket mag niet te groot worden (het adres wordt meegestuurd). *Oplossing: een pair modus uitvogelen, 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 kan zien welke slaves er offline (ontkoppeld) zijn en welke er online zijn.
* Meerdere commando’s 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 vertraagd het verzenden en ontvangen van data. Elk datapakket heeft een variabele lengte van 4 bytes tot 20 bytes.

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

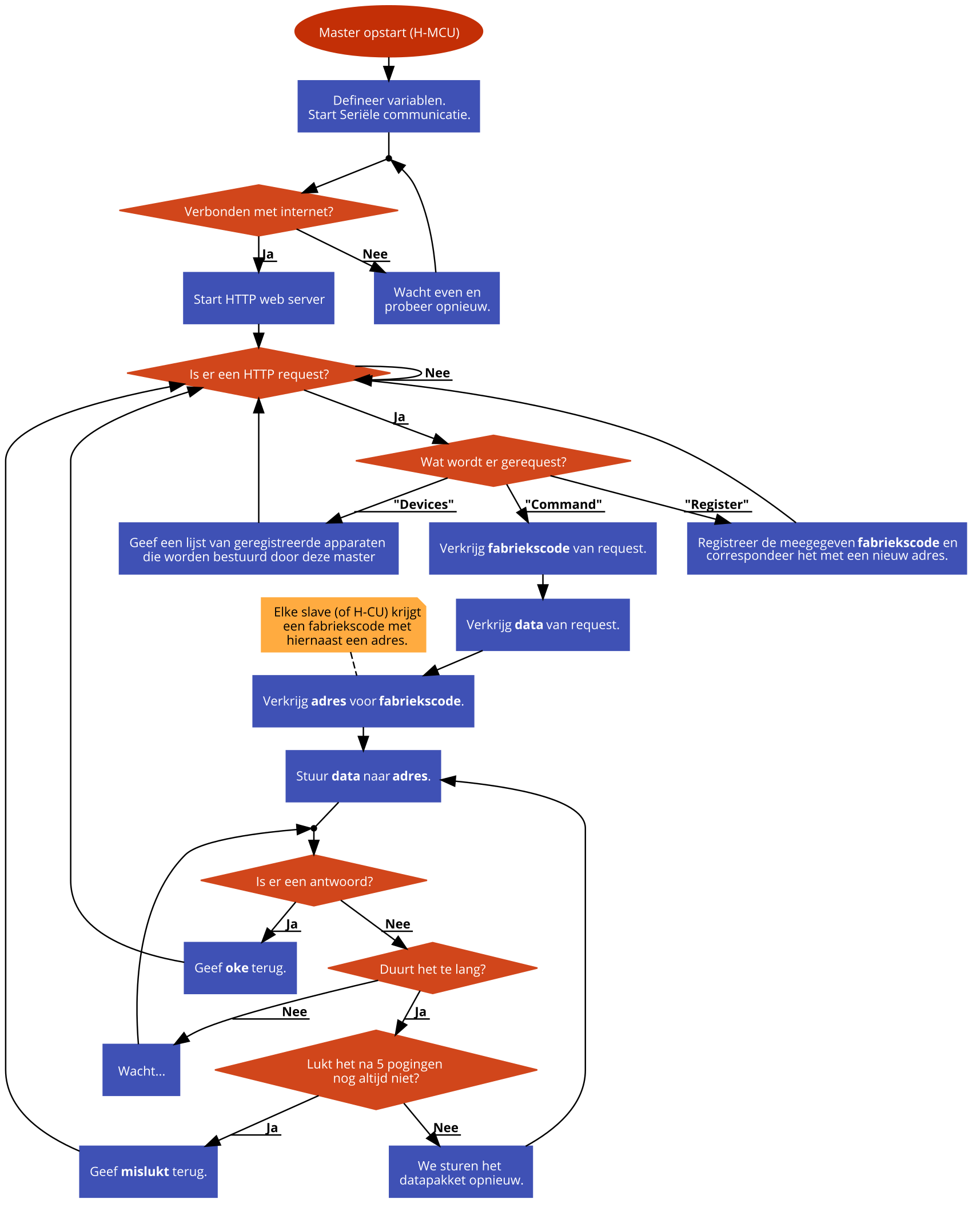
De library zorgt voor verschillende functies die zowel de master als de slave gebruiken: verzenden en ontvangen van datapakketten, inhoud van datapakketten vertalen ...

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

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

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



Ik kan jammer genoeg niet het programma lijn voor lijn uitleggen, dit zal even duren want het master programma bevat zowat 700 lijnen code (zonder library). Daarom heb ik gezorgd voor een extreem versimpeld programmatje met bijgevoegde commentaren. (Voor het volledige programma, zie bronnenlijst -> GitHub)

1. // Dit voert 1 keer uit bij opstart.
2. **void** setup()
3. {
4. // Start de communicatie met de aangesloten computer zodat we kunnen
5. // zien wat de master aan het doen is.
6. startSerieleCommunicatieMetComputer(19200);
8. // Laad de vorig opgeslagen apparaten uit het niet-vluchig EEPROM geheugen.
9. // (https://www.arduino.cc/en/Reference/EEPROM)
10. laadApparatenUitROM();
12. // Wacht tot we verbonden zijn met het lokale internet.
13. **while** (wifi.verbind() != VERBONDEN)
14. {
15. // Als we niet verbonden zijn, wacht 250 milliseconden
16. // en probeer dan opnieuw.
17. delay(250);
18. }
20. // Start de multicast DNS service.
21. // (https://en.wikipedia.org/wiki/Multicast\_DNS)
22. startMulticastDNS();
24. // Start de communicatie met de aangesloten antenne module zodat we
25. // data kunnen versturen naar andere antennes.
26. startSerieleAntenne(2400);
27. }
29. // Dit wordt oneindig keren herhaald.
30. **void** loop()
31. {
32. // Laat een led knipperen wanneer nodig als visueel
33. // weergave over wat er aan de hand is.
34. laatDebugLedKnipperenAlsNodig();
36. // Is er een commando ontvangen?
37. **if** (commandoOntvangen())
38. {
39. // Voer het ontvangen commando uit.
40. voerCommandoUit();
41. }
43. // Is er een datapakket ontvangen?
44. **if** (dataPakketOntvangen())
45. {
46. // Is de zender van het pakket gekoppend aan deze master?
47. **if** (gekoppeldAanMaster(dataVanSlave))
48. {
49. // Interpreteer de ontvangen data.
50. interpreteerData(dataVanSlave);
51. }
52. **else**
53. {
54. // Koppel de slave die de ontvangen data had verstuurd.
55. koppelNieuweSlave(dataVanSlave);
56. }
57. }
59. // Herverstuur alle pakketten die al zijn verstuurd
60. // naar slaves maar waarvan geen 'oke' bericht was teruggegeven.
61. herverstuurNietOnvangenPakketten();
63. // Om de 30 seconden:
64. **if** (millis() % 30000 == 0)
65. {
66. // Probeer alle niet werkende slaves aan de praat te krijgen.
67. // (niet werkend: geregistreerde slaves die niet antwoorden)
68. koppelNietWerkendeSlaves();
69. }
71. // Om de 5 seconden:
72. **if** (millis() % 5000 == 0)
73. {
74. // Noteer welke slaves online zijn en welke offline zijn.
75. checkWelkeSlaveOnlineZijn();
76. }
78. // Laat de onderliggende webserver werken zodat deze module commando's
79. // kan ontvangen van (als voorbeeld) de smartphone app.
80. laatWebserverWerken();
81. }

### Bij de ‘slave’

Todo

### Protocol berekeningen

todo

## Uitgebreide uitleg hardware

### Microcontrollers

todo

### Communicatie

todo

### Componentenlijst

todo

# Design

## Elektrische schema’s

Todo inleiding

### Prototypes

Todo: breadbords, purfboard

### Ontwerp

todo: gemaakt met eagle

### Eindwerk

Todo: hongkong

## Behuizing

Todo: 3 model, afmetingen

# 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://en.wikipedia.org/wiki/Baud>

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

Alle mijn bestanden (code, documentaties, presentaties, flowcharts) die ik heb gebruikt doorheen het jaar en zowel in deze bundel, staan op **GitHub** onder de MIT Licentie: <https://github.com/CodeStix/Home-Control-GIP>

# Logboek

Hier logboek invoegen

# Besluit en zelfreflectie

todo

# Bijlagen