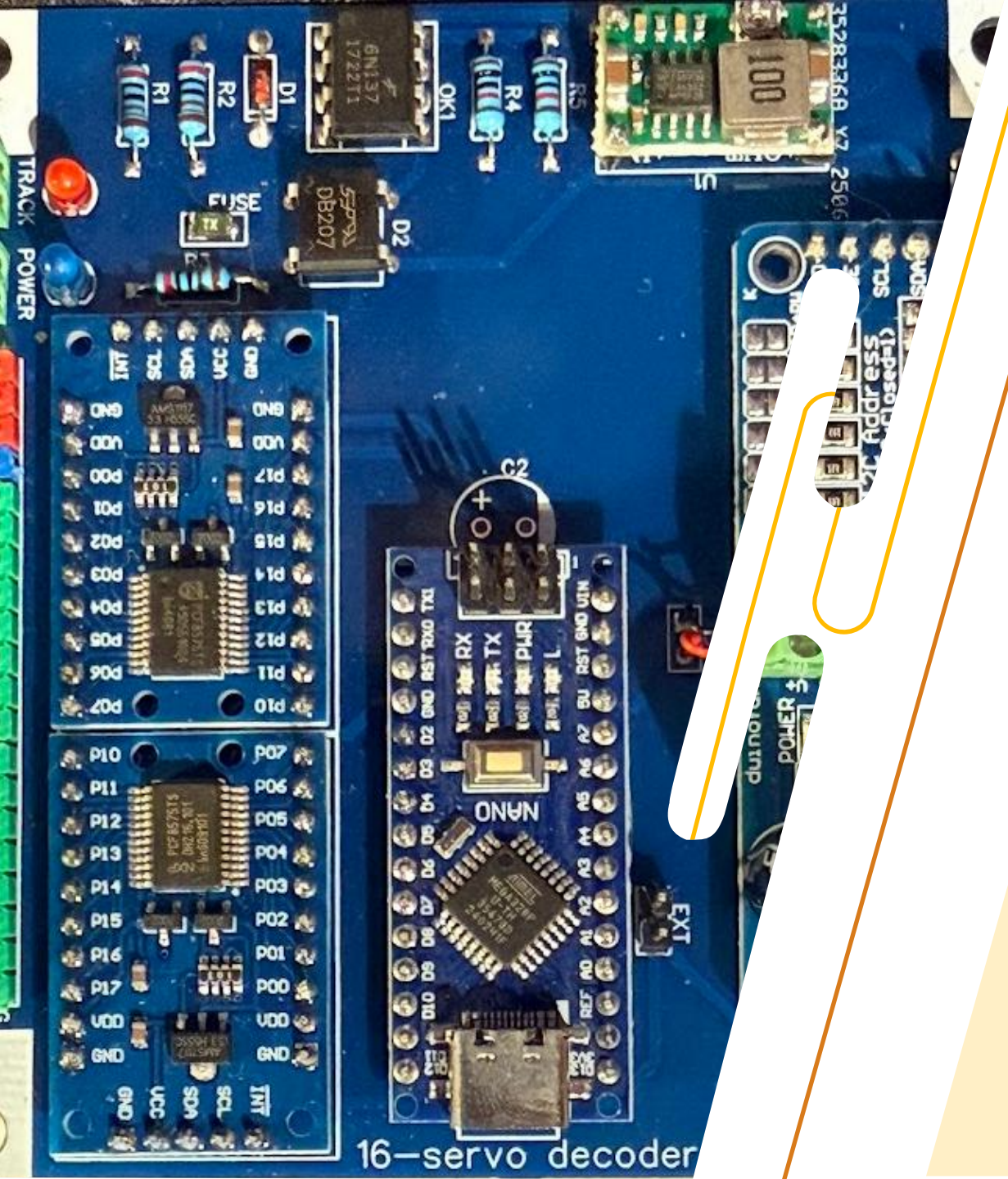


Microcontrollers op de treinbaan

PRESENTATIE DOOR:

JACK VAN AS
EN
ERIK WAJER

Onderwerpen

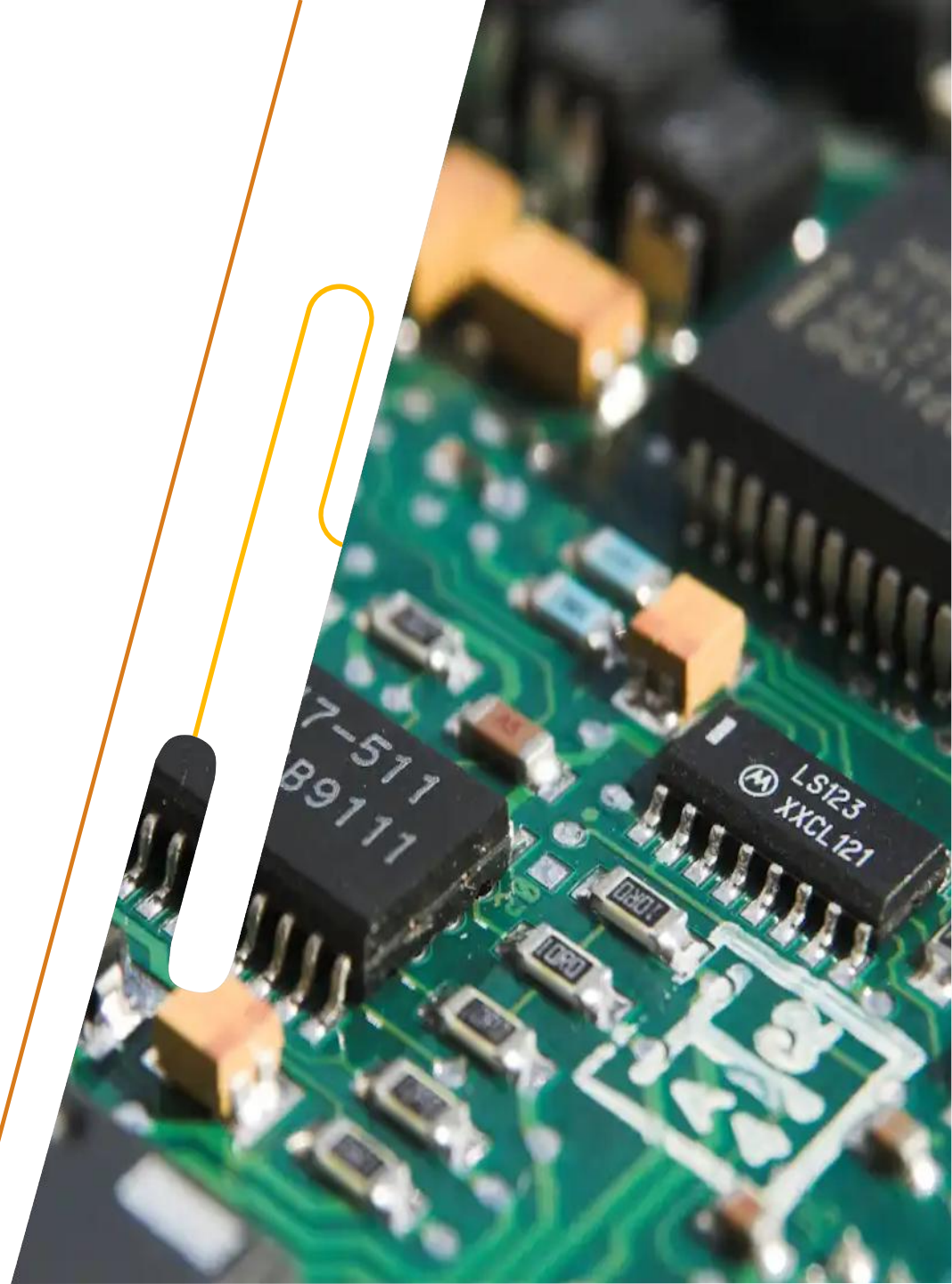
01. Introductie

02. Hardware ontwerp

03. Software ontwikkeling

04. Prototyping, productie en testen

05. Doorontwikkeling



Introductie

Wie zijn wij?

- Erik Wajer : Hardware ontwerp en Software ontwikkeling
- Jack van As : Test, Productie, Sales en Support
- Samen : Hobby voor Hobby

Introductie:
Onze doelstellingen

Wat willen we niet :

- Nabouwen wat anderen ook al gemaakt hebben
- Complexe oplossingen

Wat willen we dan wel :

- Vernieuwende techniek
- Eenvoudig gebruik
- Goedkoop zelf te produceren

Hardware ontwerp:
Wisseldecoder

De 8-wisseldecoder was het eerste project wat we besloten professioneel aan te pakken. Het begon allemaal met het idee dat een **Arduino** 16 bruikbare **uitgangen** heeft. Uitgangspunt waren eerdere experimenten met het gebruik van **libraries** voor het verwerken van DCC en Motorola (MM) opdrachten vanuit diverse centrales.

Om het signaal vanuit de centrale veilig te kunnen verwerken, heb ik gekozen voor een **optocoupler** (6n137). Dit zorgt ervoor dat de electronica galvanisch gescheiden is van het ingangssignaal.

Om voldoende vermogen voor het aansturen van ook de zwaarste kruiswissels te kunnen leveren, koos ik voor de **uln2803**. Door uitgangen te bundelen kan deze 1 Ampère per wisselspoel leveren, beveiligd met een **autofuse**.

Verder zorgen twee **schakelaars** ervoor dat het adres ingesteld kan worden en de uitgangen getest kunnen worden.

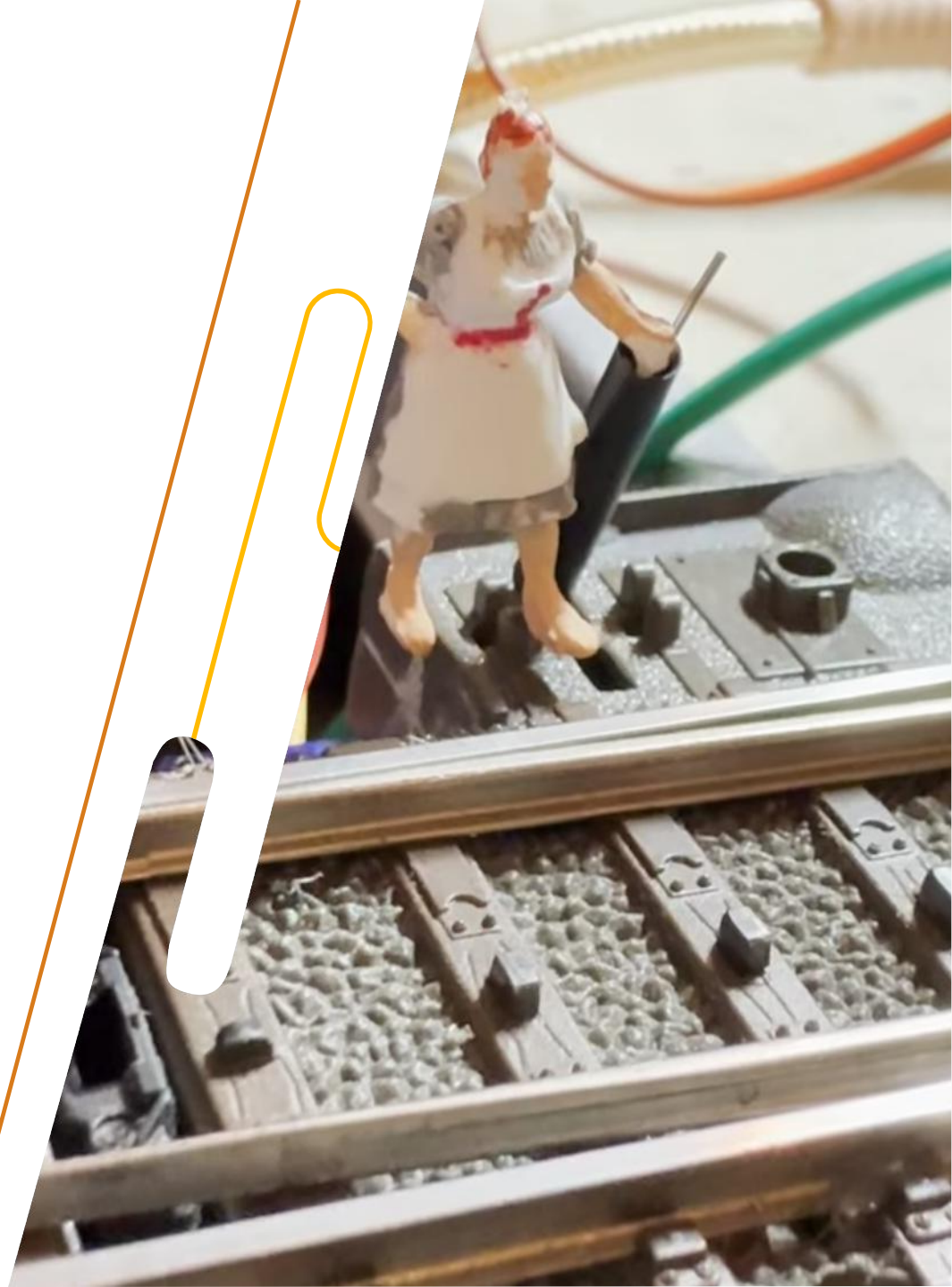
Hardware ontwerp:
16-servo decoder

Op basis van de ervaringen met de wisseldecoder leek het ons leuk ook een **servo-decoder** te gaan maken. De basis is een **pca9685** board, waarmee 16 servo's aangestuurd kunnen worden. Deze heb ik gecombineerd met twee **pcf8575** boards, zodat ik 32 i/o pennen beschikbaar krijg. Deze boards worden allemaal via een **i2c** bus aangestuurd, ieder op een eigen adres.

De 32 **i/o** pennen van de pcf8575 boards en de i2c bus heb ik samen met wat voeding-pennen uitgevoerd op een 40-pen connector.

Om **terugmelding** te geven van de stand van de servo's en de werking van de decoder, heb ik extra pins aangebracht op de print voor het aansluiten van een **ws2812b** ledstrip.

Om ook weer eenvoudig het **basisadres** van de decoder in te stellen, het ik ook hier een programmeer-knopje aangebracht, en net zoals op de 8-wisseldecoder zijn er **leds** bij de ingangen die aangeven of er spanning aanwezig is.

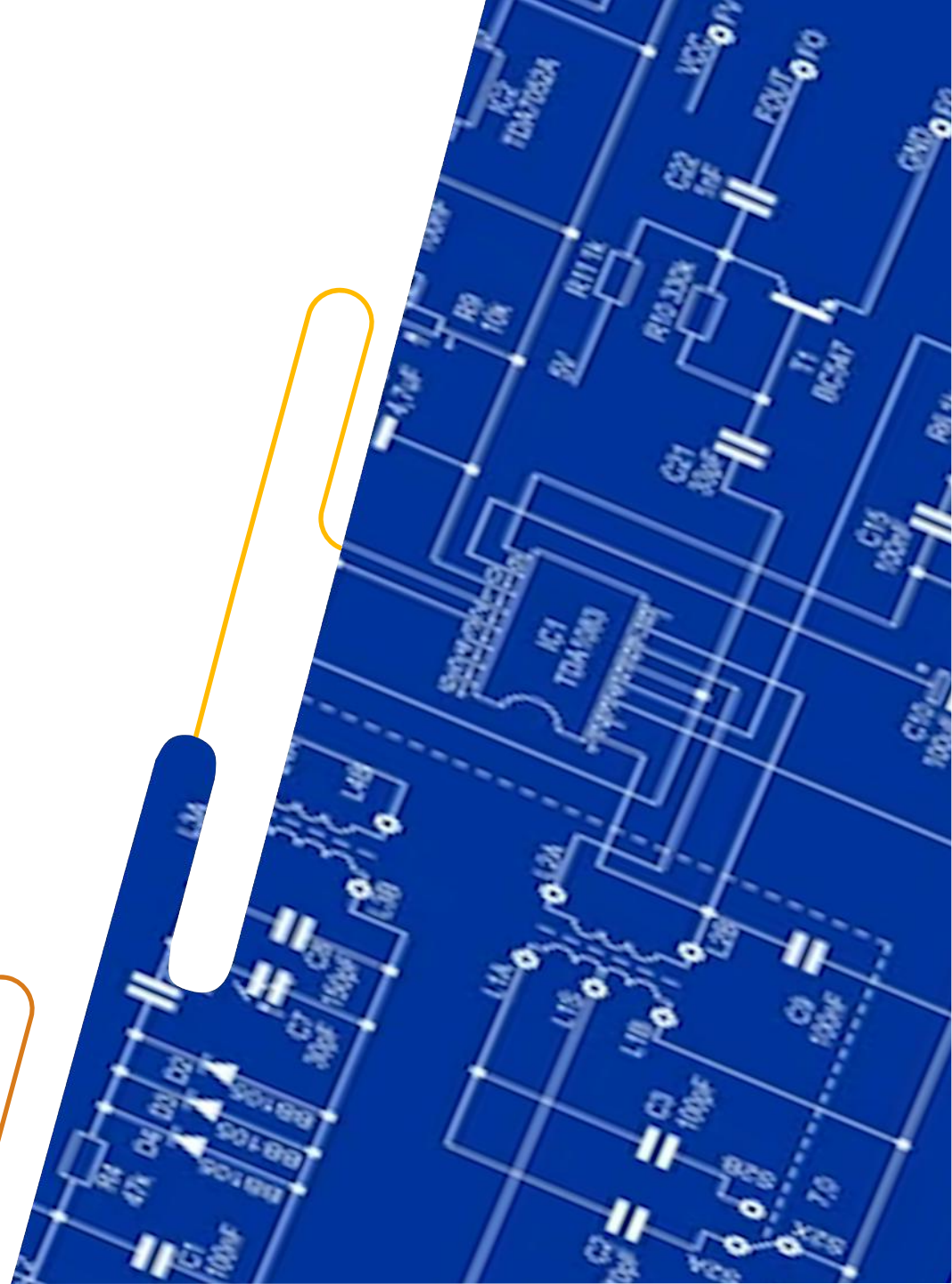


Hardware ontwerp:
Schema ontwikkeling

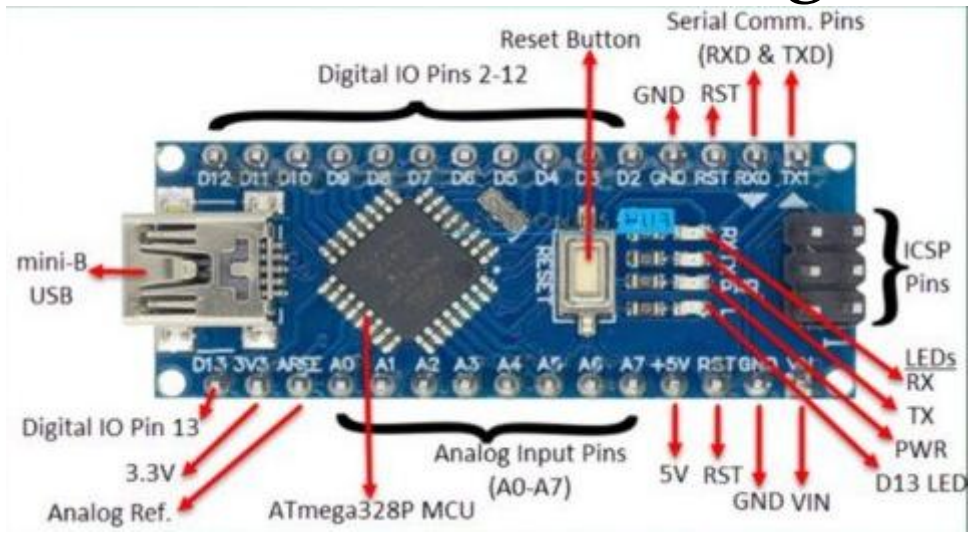
Voor het ontwerpen van het **schema** en de uiteindelijke printen gebruik ik **EasyEDA**. Dit is een gratis en relatief eenvoudig pakket voor het ontwerpen van schakelingen. Het pakket werkt volledig online en kan standaard Gerber-bestanden genereren voor het laten maken van de printplaten.

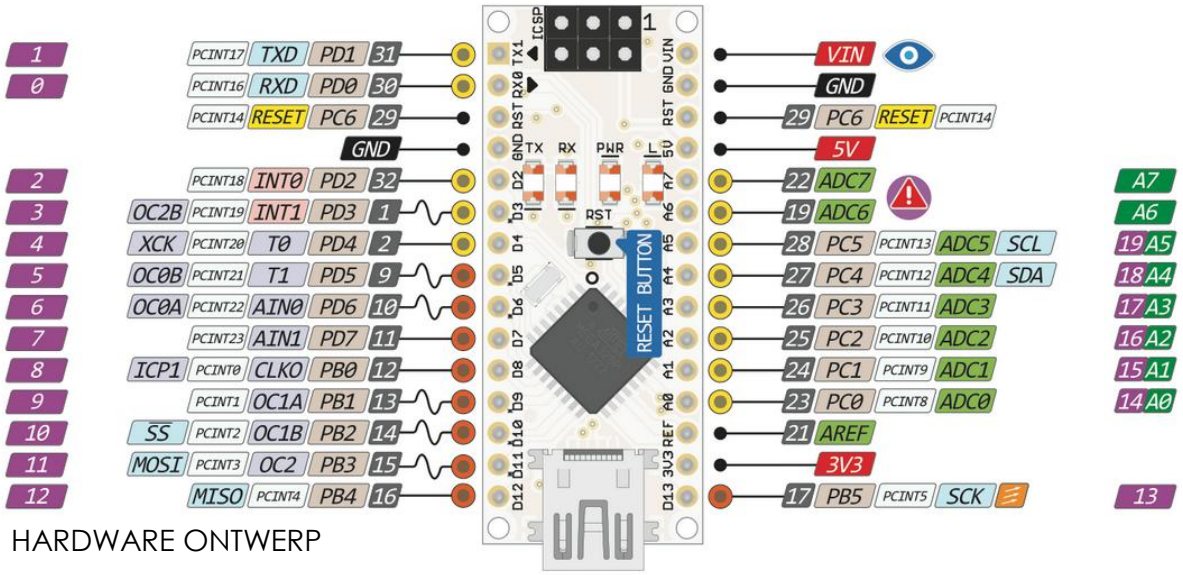
EasyEDA bevat diverse **libraries** met daarin de meeste standaardcomponenten. Het is belangrijk de afmetingen te dubbelchecken, er zijn vaak meerdere uitvoeringen van componenten, waarbij de afmetingen soms net niet overeenkomen. Als componenten niet in de juiste afmetingen beschikbaar zijn, kan altijd teruggevallen worden op simpele header-connectoren.

Om de printen ook zelf te kunnen maken, heb ik gekozen voor de **through-hole** techniek, dus geen SMD componenten. Om de prijs laag te houden, koos ik er daarnaast voor zo veel mogelijk standaard **modules** te gebruiken : Arduino, pca9685, pcf8575, allemaal prima verkrijgbaar voor een lage prijs.

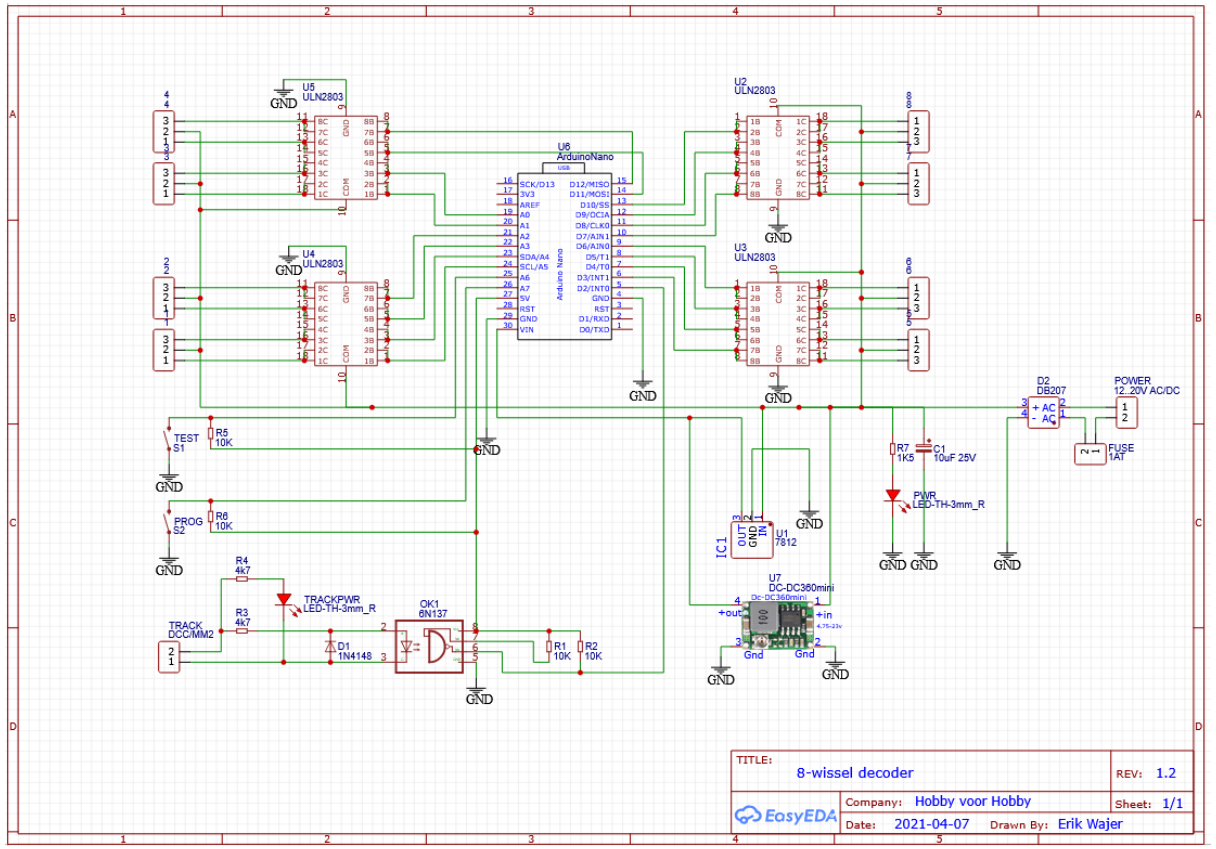


Hardware ontwerp:
Schema ontwikkeling



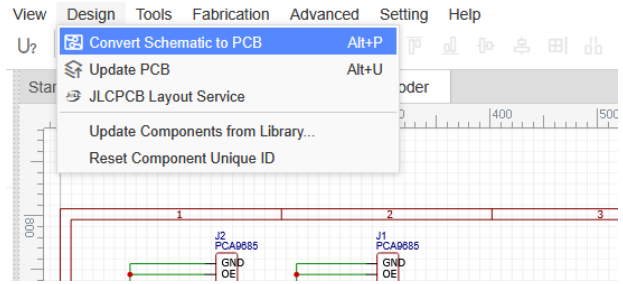


Hardware ontwerp:
Schema ontwikkeling



Hardware ontwerp:
Printontwerp

Na het tekenen van het **schema** kan deze omgezet worden naar een **printontwerp**. Ook dit kan met EasyEDA gedaan worden.



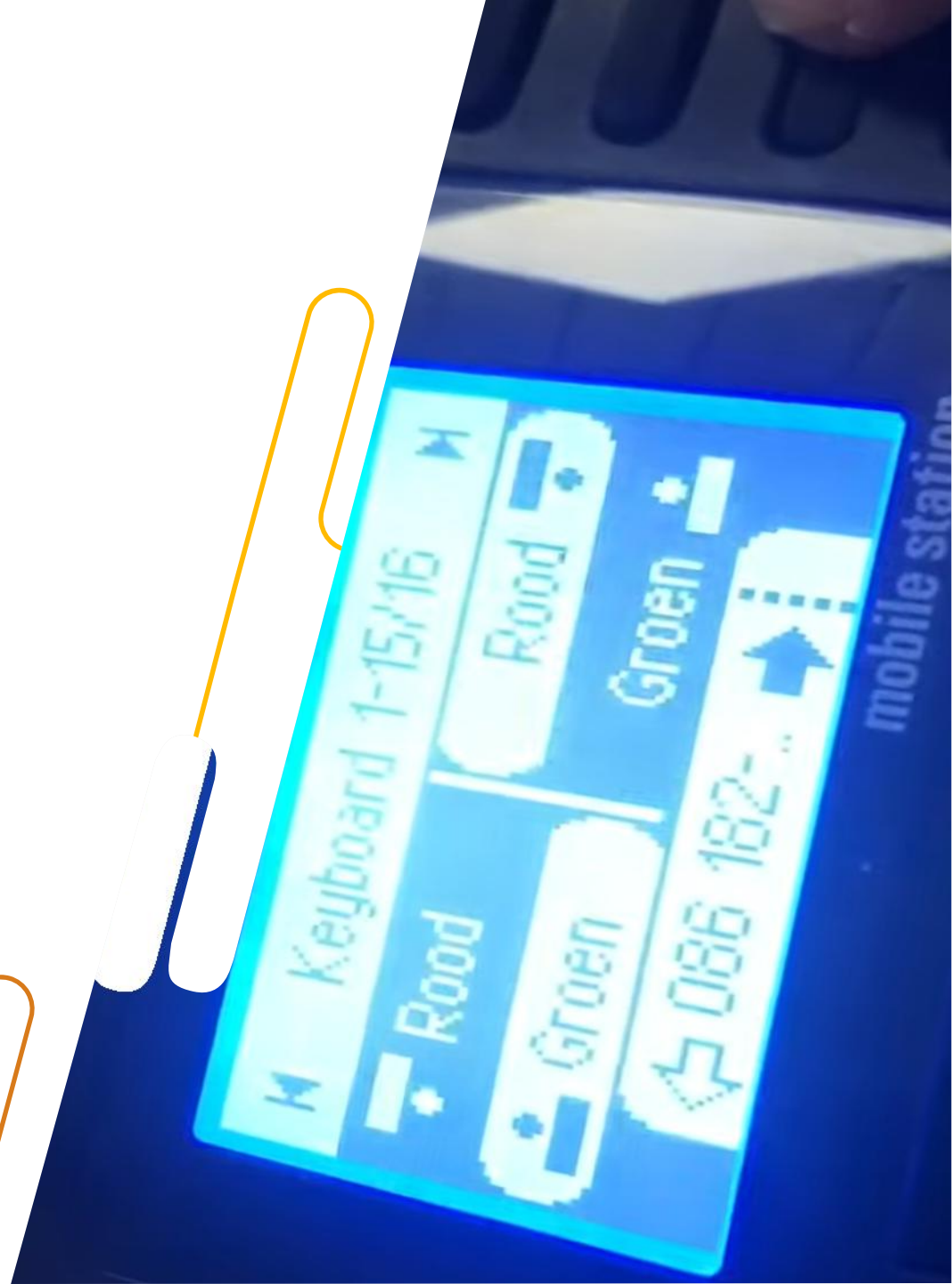
Het mooie van EasyEDA is dat eventuele **wijzigingen** in het schema ook direct verwerkt kunnen worden in het ontwerp, en dat continu **gecontroleerd** wordt of alle verbindingen nog kloppen.

Om de kosten laag te houden, heb ik gekozen voor **twee-laags** printplaten.

Hardware ontwerp:
Printontwerp

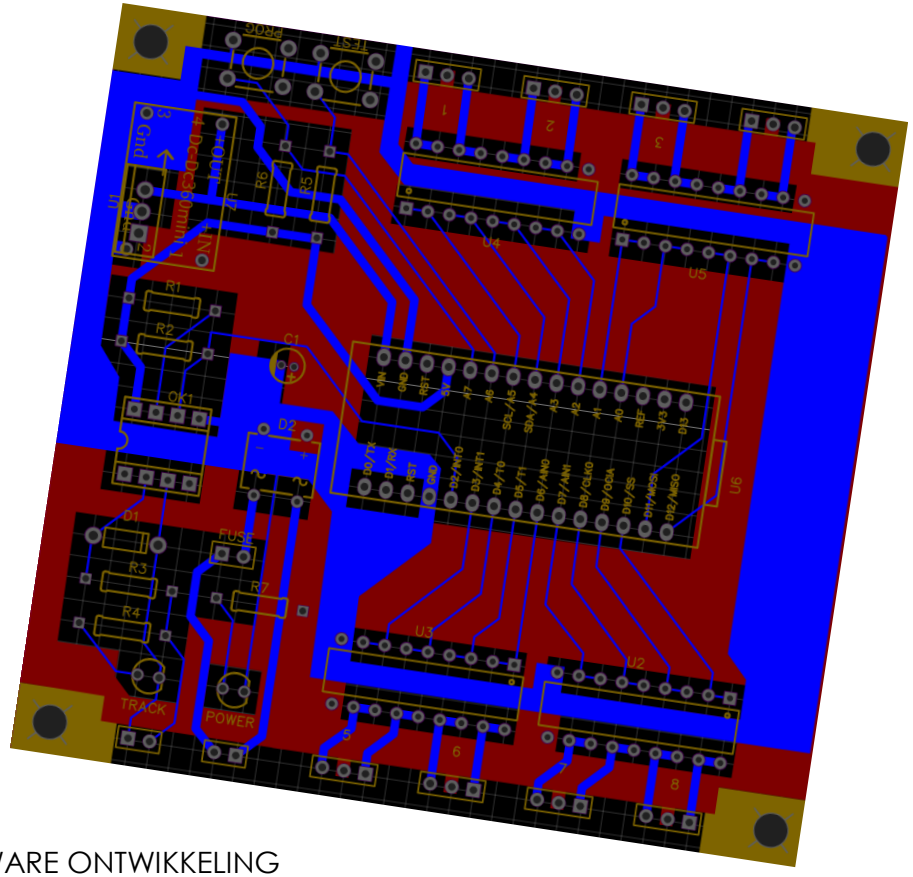
Bij het **ontwerpen** van de printplaat moet met een aantal zaken rekening gehouden worden:

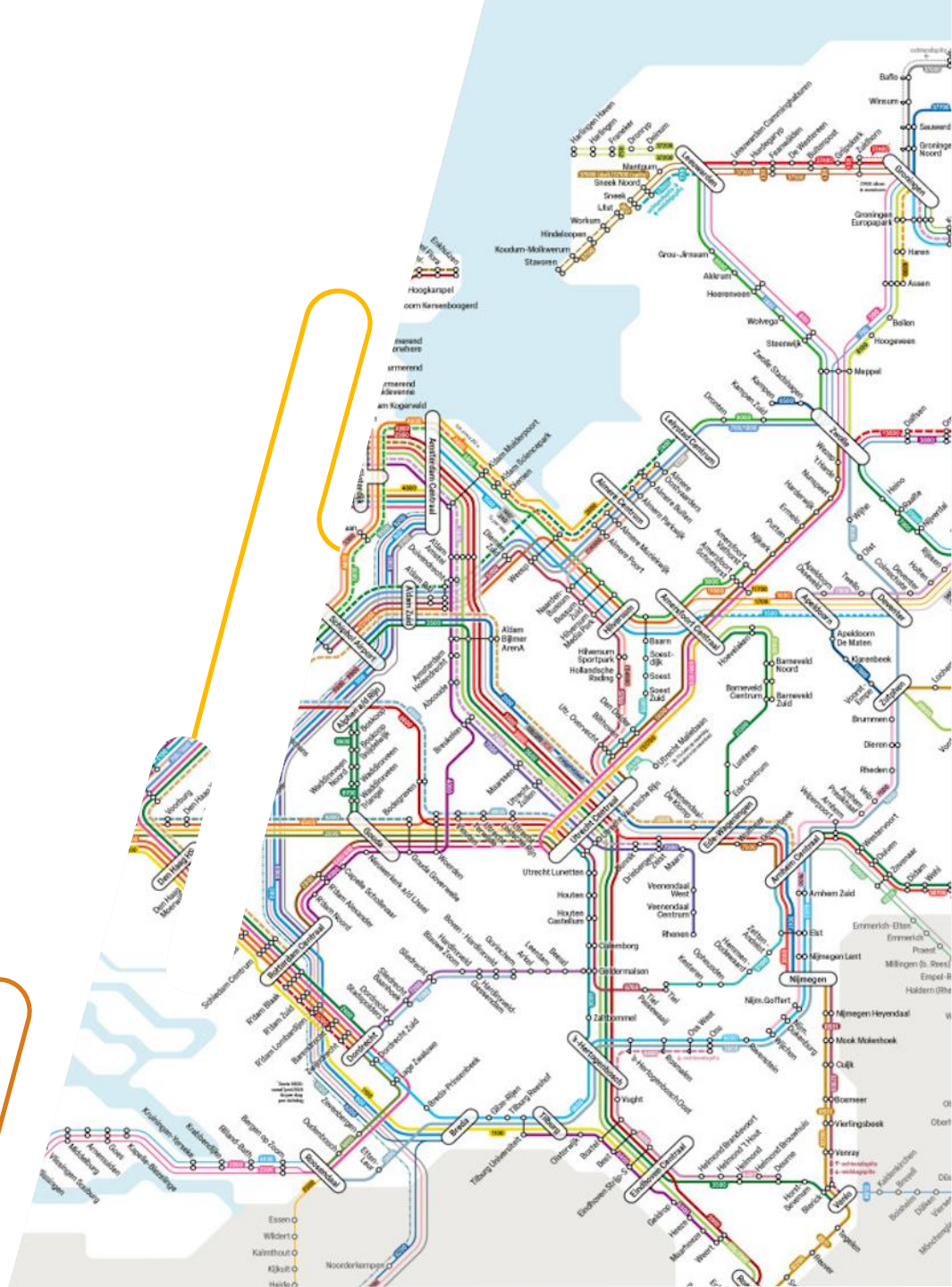
- Alle **aansluitingen** aan de buitenzijde;
- **Statusleds** naast de ingangen;
- **Schakelaars** goed bereikbaar;
- Logische **verdeling** van de componenten;
- Zo weinig mogelijk **kruisende** printbanen;
- USB-poort van de Arduino niet **blokkeren**;
- Duidelijke **markering** van de aansluitingen;
- Rekening houden met de afmetingen van de **behuizing**;

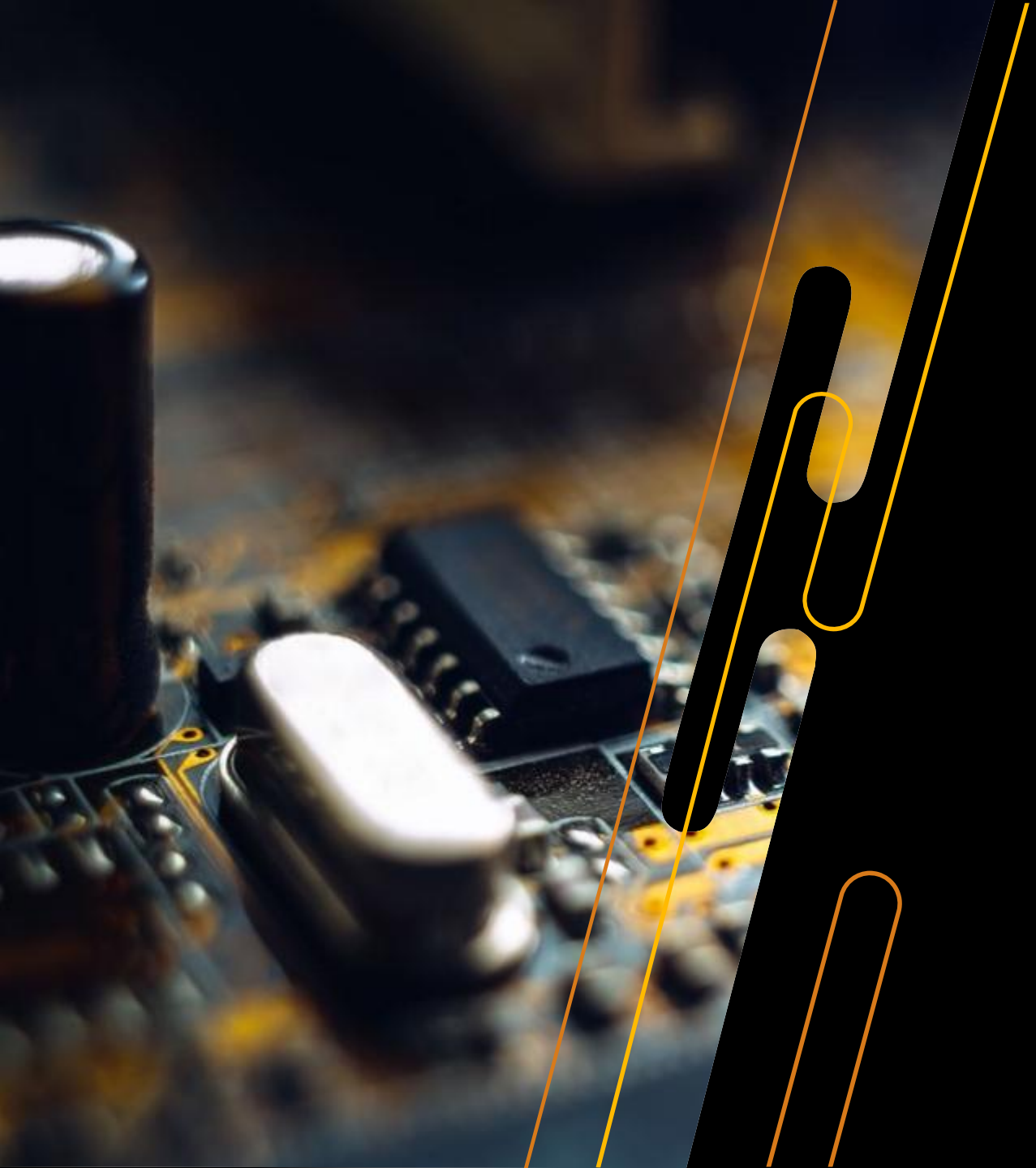


Hardware ontwerp:
Printontwerp

Bij het ontwerpen van de wisseldecoder zijn zo breed mogelijke sporen en vlakken gebruikt voor een goede geleiding van de **stroom**.







Conclusie

Hardware ontwerp is nog niet zo eenvoudig!

Prototyping

Het kiezen van de juiste onderdelen is belangrijk. Hier is veel prototyping voor nodig, bijvoorbeeld met breadboards.

Arduino

De ene Arduino is de andere niet. Van de Nano hebben we bijvoorbeeld zo'n 8 verschillende uitvoeringen gezien!

Kwaliteit

Om een betrouwbare print te maken, moet gelet worden op een goed doordachte opzet van de print.

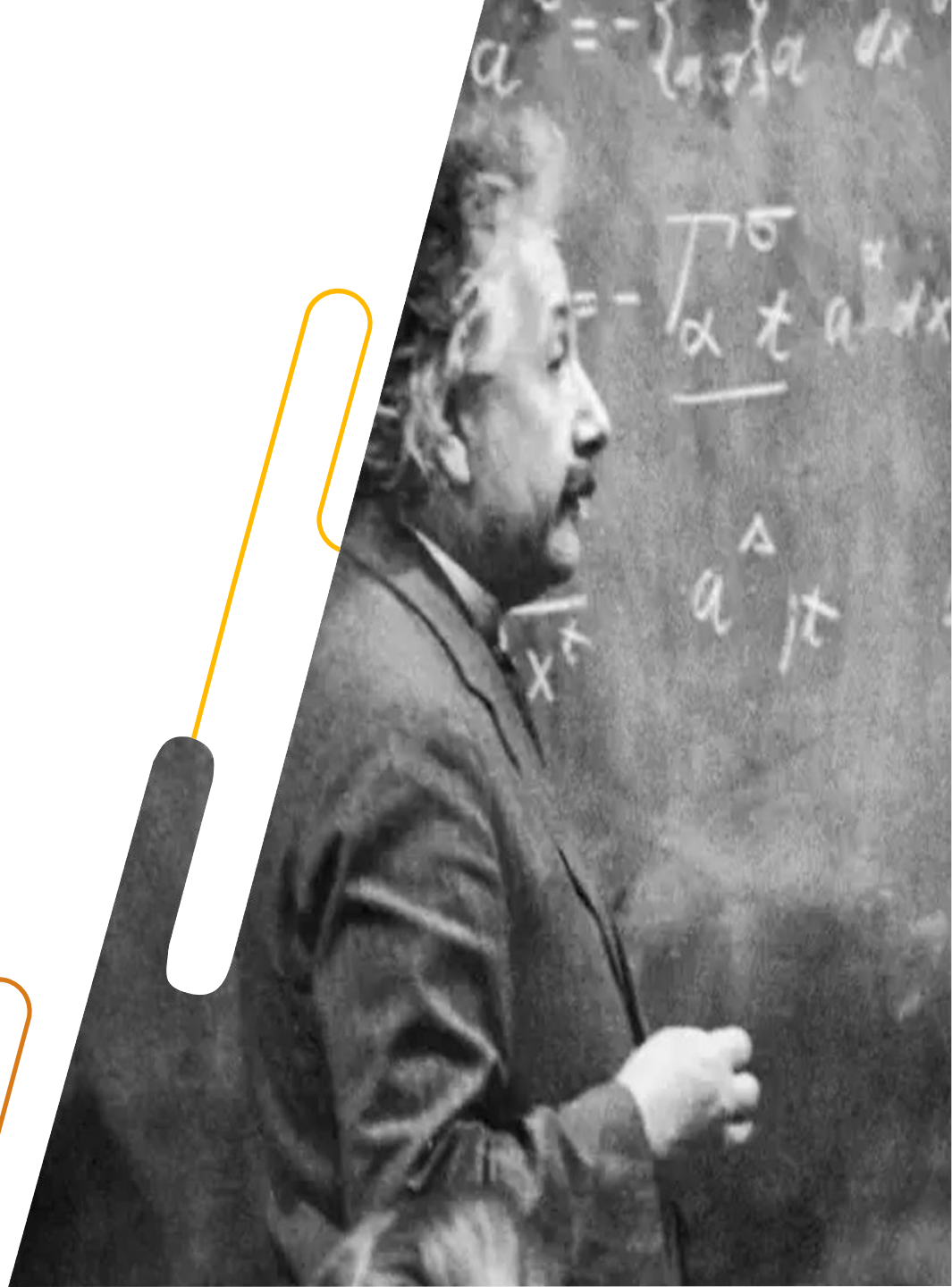
Sotware ontwikkeling:
Algemeen

Voor het ontwikkelen van de software voor beide decoders gebuik ik de **Arduino IDE**. Dit is een gratis ontwikkelomgeving voor microcontrollers. Het programmeren van de Arduino doe ik in **C++**.

Voor de meeste hardware zijn wel **libraries** beschikbaar. Bij deze libraries zitten ook vaak **examples**, die goed gebruikt kunnen worden om de hardware te testen en die als voorbeeld kunnen dienen voor het gebruik.

De Arduino Nano kan op verschillende chips gebaseerd zijn. We hebben gemerkt dat Nanos met de **Atmega328p** en USB-C aansluiting het beste werken. Andere chips zijn lastiger te programmeren, verwerken interrupts niet, of hebben minder geheugen.

Door de galvanische scheiding van het railsignaal kunnen de decoders geprogrammeerd worden terwijl ze aangesloten zijn. Dit versnelt het **ontwikkelproces** aanmerkelijk.



Sotware ontwikkeling:
Ontwerp

Het globale ontwerp van de decoders is relatief simpel :

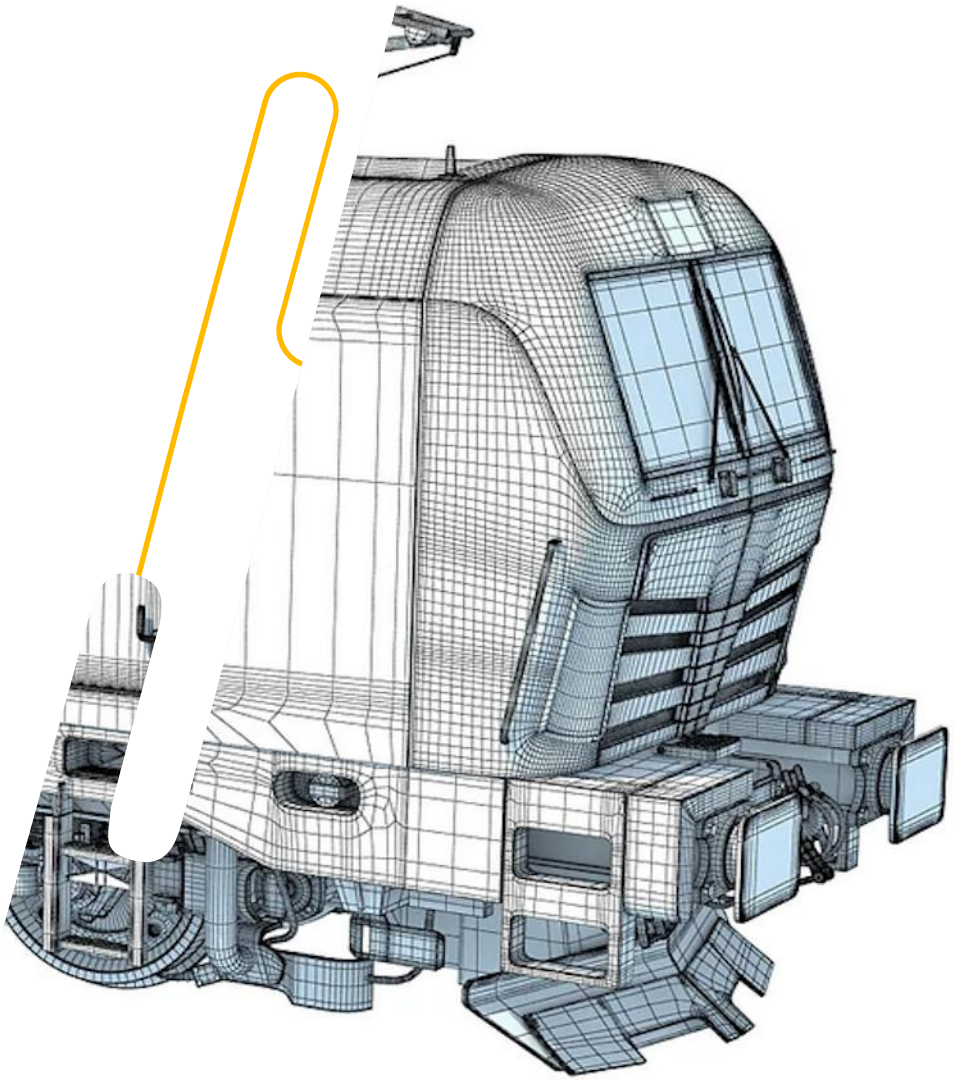
- Bij het binnenkomen van een wissel-commando deze status opslaan in EEPROM;

- Als de status van een wissel niet overeenkomt met de status in EEPROM, hiernaar handelen;

- Bij het opstarten de status uit EEPROM direct verwerken;

- Voor de servo's : substatussen voor snelheid en bounce;

- Voor de wisseldecoders : watchdog voor bescherming van wisselspoelen;



Sotware ontwikkeling:
Library selectie

Er zijn diverse **libraries** beschikbaar voor het verwerken van het railsignaal met een Arduino. Voor DCC heb ik uiteindelijk gekozen voor de **NmraDcc** library. Voor MM koos ik voor de **MaerklinMotorola** library, gemaakt door Laserlight. Beide zijn in de ArduinoIDE te downloaden.

De **wisseldecoder** is hierdoor zowel geschikt voor het DCC- als voor het Märklin/Motorola (MM) protocol. Dit zijn wel twee verschillende versies van de software. De **servodecoder** is alleen geschikt voor het DCC-protocol. Door een tekort aan geheugen is het (nog) niet mogelijk alle funtionaliteit te gebruiken in combinatie met de Motorola library.

Voor het aansturen van de servo's gebruik ik de **Adafruit_PWMServoDriver** library.

Voor de terugmeldleds op de servodecoder gebruik ik de **FastLED** library. Dit is helaas wel een geheugenvreter, maar ik heb nog geen beter alternatief gevonden.

Voor de PCF8575 gebruik ik de **PCF8575** en **wire** libraries.

Sotware ontwikkeling:
Wissel-commando’s

Beide decoders verwerken **wissel-commando’s**. De meeste centrales versturen deze commando's meerdere keren achter elkaar om te voorkomen dat een commando gemist wordt. De commando’s bestaan uit een wisseladres (Address), een richting (Direction) en aan/uit (Power).

Het **Address** is een waarde tussen 1 en 320 of hoger, afhankelijk van de centrale. Opvallend is dat commando’s vanuit een Roco Multimaus een offset van -4 hebben.

De **Direction** is een 0 (afbuigend) of een 1 (rechtdoor).

De **Power** is een 0 (uit) of 1 (aan). Er is geen garantie dat er een 0 commando binnenkomt (!), en de tijdsduur tussen een 1 en een 0 verschilt per centrale (!).

```
void notifyDccAccTurnoutOutput(uint16_t Addr, uint8_t Direction, uint8_t OutputPower)
{
```

Prototyping, productie en testen :
Eerste proefjes

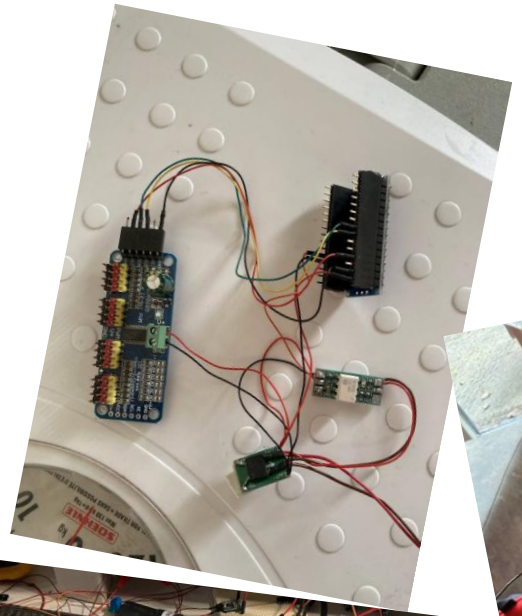
Op diverse fora vond ik voorbeelden van zelfbouwdecoders. Ik besloot wat onderdelen te bestellen. Voor de **microcontroller** heb ik gekeken naar de Raspberry pi, de ESP32 en diverse Aduino boards zoals de Uno, Mega, Mini en de Nano. Uiteindelijk is mijn keuze op deze laatste gevallen vanwege het formaat en de beschikbaarheid van stabiele libraries.

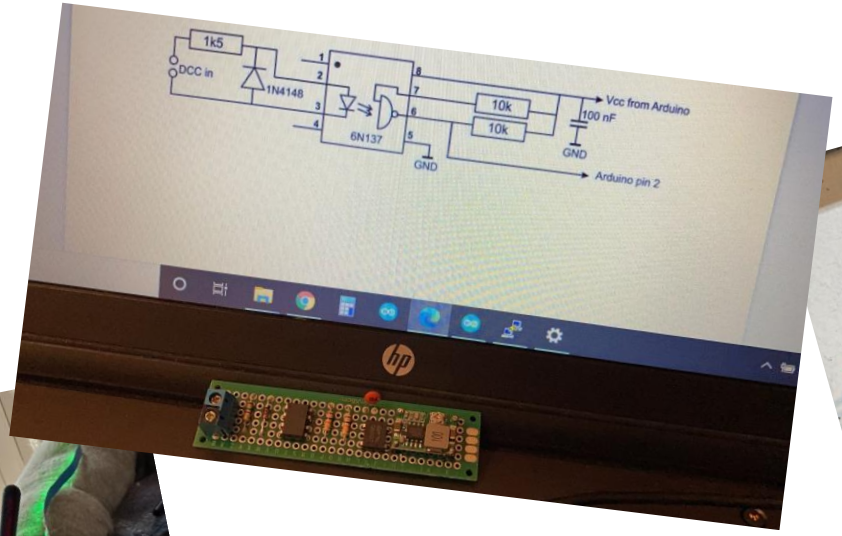
Voor het koppelen van het **railsignaal** bestaan twee manieren : via een opto-coupler of rechtstreeks met een weerstand. Na een aantal opgeblazen nano's is de keuze definitief op opto-couplers gevallen.

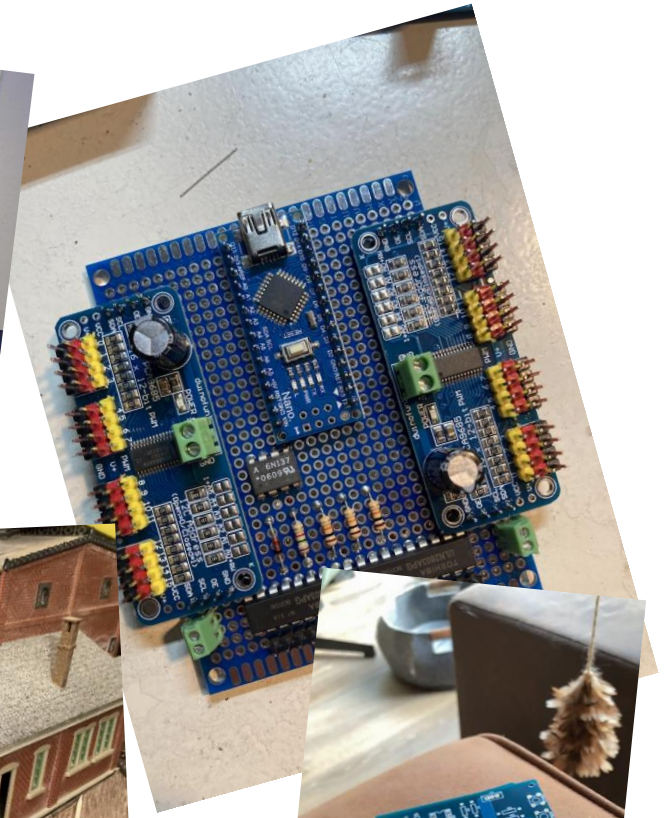
Nu dit geregeld was, moest besloten worden wat ik wilde gaan **ontwikkelen** op basis van de mogelijkheden die de soft- en hardware mij boden. Ik heb wat geëxperimenteerd met stappenmotoren, servo's, leds etc. Omdat ik wisseldecoders nodig had, besloot ik daarmee te beginnen.

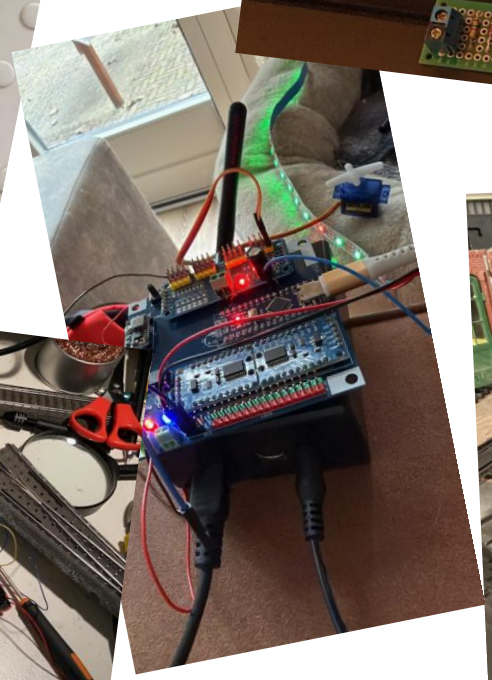
Een eerste **prototype** werd op gaatjesprint gemaakt, daarna volgde een printplaat die met een freesplotter geproduceerd werd.

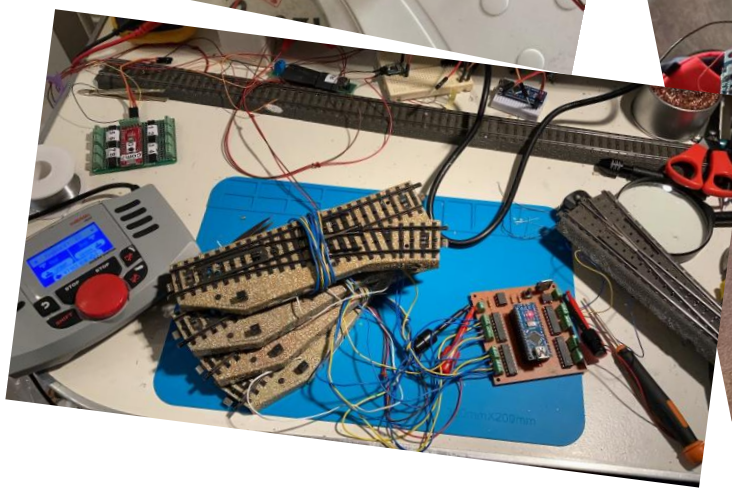
Prototyping, productie en testen :
Eerste proefjes











Prototyping, productie en testen :
Productie en testen

De wisseldecoder werd eerst op mijn eigen treinbaan getest, en daarna ook op die van vrienden. Voor de servo-decoder hebben we een bèta-tester kunnen vinden die waardevolle feedback gaf.

De uiteindelijk ontworpen printplaten laten we in China maken. De onderdelen en behuizingen bestellen we over het algemeen ook daar. We solderen de printplaten zelf, en testen deze daarna zorgvuldig.

De decoders zijn prima geschikt voor mensen die hun treinbaan willen gaan digitaliseren of als vervanging van bestaande decoders. Lage kosten en eenvoudig gebruik maken het een aantrekkelijk alternatief.

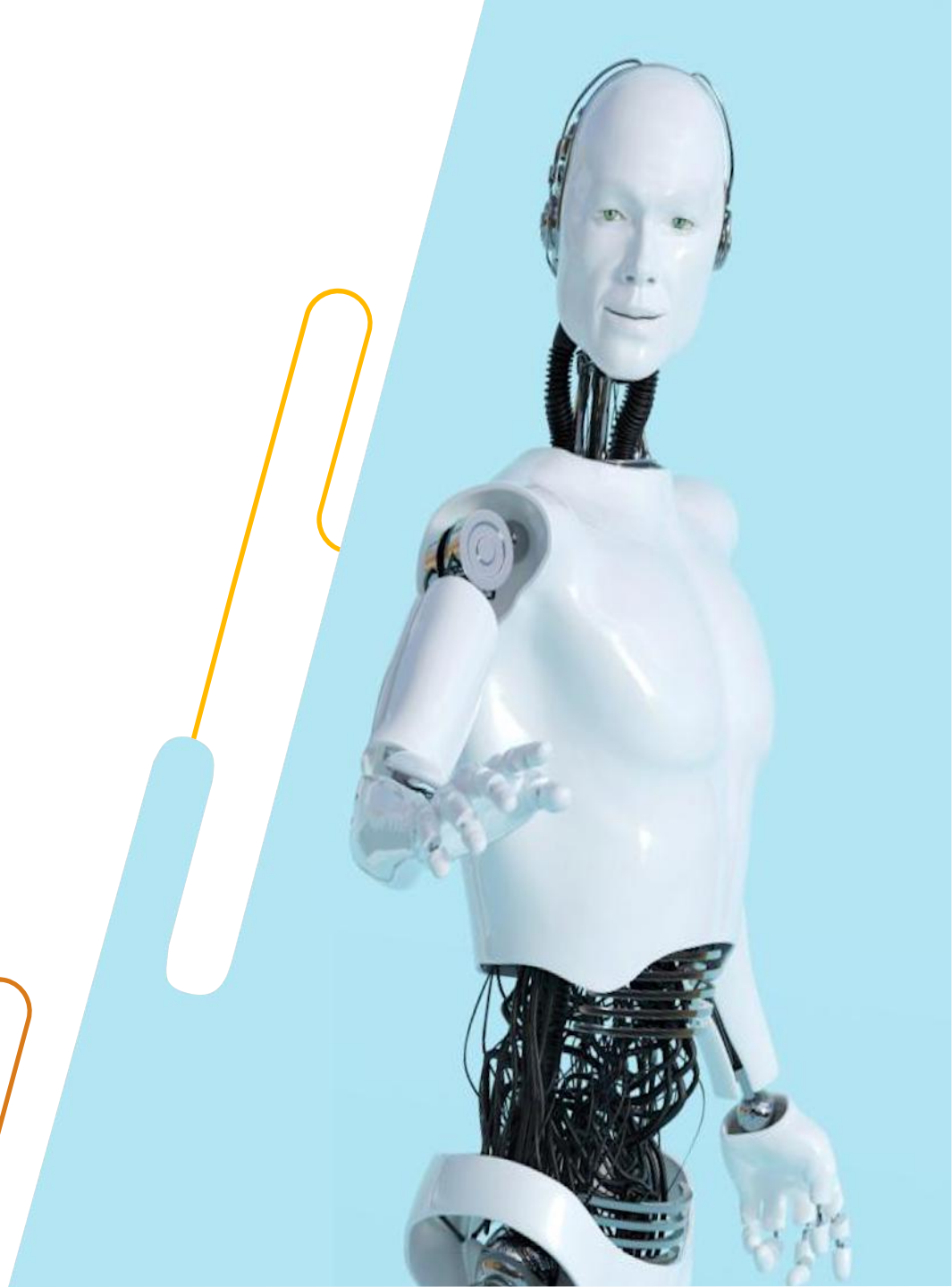
Omdat er veel vraag naar de decoders bleek te zijn, hebben we besloten een website te maken. Inmiddels hebben we ook al decoders verzonden naar Noorwegen en zelfs naar Canada!

Doorontwikkelen :
Plannen voor de toekomst

Voor de toekomst zijn er nog voldoende ideeën:

- Doorontwikkelen wisseldecoder : terugmeldleds, cv programmering etc;
- Doorontwikkelen servo-decoder : ESP32 met configuratie via wifi, 32 uitgangen naast de 32 ingangen;
- Functiedecoder op basis van de ESP32 met configuratie via wifi;
- Locdecoder voor stappenmotoren?
- MFX slave-decoder;
- S88 hub met led-terugmelding;

Voor een aantal van deze ideeën zijn al wat opzetjes gemaakt.



Bedankt!

Erik en Jack

Voor meer informatie:

www.hobbyvoorhobby.nl