

Prime Waferstepper

Eind verslag

OGO 1.3: Groep 7

door:

Etienne van Delden	0618959
Gijs Direks	0611093
Sanne Ernst	0588898
Bas Goorden	0598669
Stef Sijben	0607426
Coen van der Wel	0608467

Versie 1.0

Inhoudsopgave

1 Inleiding	4
2 Systeemontwerp	1
2.1 Inleiding van het systeemontwerp	1
2.2 Hardware	1
2.2.1 Opzet van het model	1
2.2.2 Foto's van het model	4
2.2.3 Ontwerpbeslissingen	16
2.3 UPPAAL model	17
2.3.1 Templates	17
2.3.2 Ontwerp Beslissingen	21
3 Specificatie elektrische interface	23
3.1 Priklijst	24
3.2 Bedradingsschema	26
3.3 Beluchtingsschema	27
4 Programmaontwerp	28
4.1 Inleiding van het programmaontwerp	28
4.2 Het UPPAAL ontwerp voor de subroutines	28
4.2.1 Bediening lopende banden	28
4.2.2 Bediening deuren	31
4.2.3 Het belichten van een wafer: Burn	32
4.2.4 Verloren wafers afhandelen: HandleFalloff	33
4.2.5 Hulpautomaat om crashes te detecteren: ManageCrash	34
4.3 Het hoofdprogramma	34
4.3.1 Het ontwerp	34
4.4 Implementatie	37

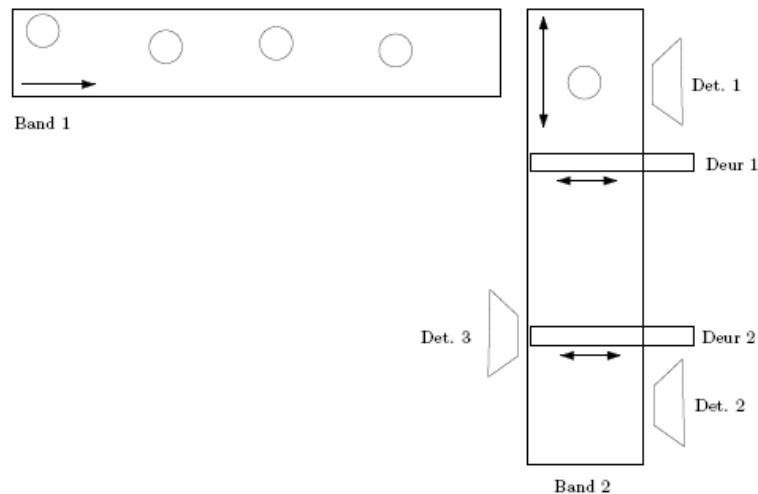
5 Systeemanalyse	38
5.1 Inleiding van de systeemanalyse	38
5.2 Belangrijke eigenschappen functionaliteit	38
5.3 Methodiek simuleren en verifiëren eigenschappen	39
5.4 Simulatie- en verificatieresultaten	39
5.5 Conclusie van de systeemanalyse	40
6 Testresultaten	41
6.1 Testresultaten en problemen	41
6.1.1 Problemen en mislukte tests	41
6.1.2 Geslaagde onderdelen	42
7 Procesdocument	43
7.1 Tijdverdeling	43
7.1.1 Vergadering	43
7.2 Taakverdeling	43
7.2.1 Voorzitter, Notulist	43
7.2.2 Werkplan	43
7.3 Evaluatie	45
7.3.1 Persoonlijke evaluatie van Etienne van Delden	45
7.3.2 Persoonlijke evaluatie van Gijs Direks	46
7.3.3 Persoonlijke evaluatie van Sanne Ernst	46
7.3.4 Persoonlijke evaluatie van Bas Goorden	47
7.3.5 Persoonlijke evaluatie van Stef Sijben	48
7.3.6 Persoonlijke evaluatie van Coen van der Wel	48
8 Conclusie	50
8.1 Conclusie	50
A Assembler code	51
A.1 main	51
B Logboeken	54

Hoofdstuk 1

Inleiding

Het doel van dit OGO project is het maken van een UV waferstepper. Een waferstepper is een machine die silicium wafers belicht om er IC's van te maken. Omdat de details op de wafers zo klein zijn dat normaal licht een te grote golflengte heeft, wordt UV licht gebruikt. Dit UV licht heeft echter als nadeel dat ze door de atmosfeer geabsorbeerd wordt. Daarom gebeurt het belichten van de wafers in een vacuüm. Bovendien gaan de lenzen die gebruikt worden bij het belichten stuk wanneer ze aan de buitenlucht worden blootgesteld.

De waferstepper die wij gaan maken, moet er als volgt uit komen te zien.



Figuur 1.1: Schema van de waferstepper

De belichtingsunit van de waferstepper is bereikbaar via een sluis met twee deuren.

Deze deuren mogen nooit tegelijkertijd open staan. Er zijn twee lopende banden. Eén waar te verwerken wafers klaar liggen, deze kan maar één richting op, en één waarop steeds een wafer door de deuren naar de belichtingsunit gaat en weer terug. Wanneer een wafer bij de belichtingsunit komt, wordt de band stilgezet, zodat de wafer belicht kan worden. Dit belichten duurt precies twee seconden. Daarna gaat de wafer via de sluisdeuren naar de verzamelbak.

Er bevinden zich twee licht detectoren in het systeem om de wafers te lokaliseren. Deze detectoren bevinden zich aan de uiteinden van de tweede band. Als er een wafer van de tweede band afvalt of af wordt gehaald, dan wordt dit aangegeven door een extra led op het processorbord te laten branden. Als er zo vijf wafers verloren gaan, stopt het systeem en kan het alleen gerestart worden door een reset.

De wafers worden handmatig op de eerste band gelegd. Er is een drukknop om aan te geven dat er wafers klaarliggen. Ook is er een noodknop om het systeem te stoppen, en daarna weer te hervatten.

De waferstepper moet ten alle tijde aan de volgende voorwaarden voldoen:

- Wanneer de noodknop wordt ingedrukt stopt het systeem resoluut. Als de noodknop nu opnieuw wordt ingedrukt, hervat het systeem zijn oude taken, behalve als op zo'n moment een wafer belicht werd. Deze wafer is dan namelijk mislukt en zal niet opnieuw belicht worden.
- Beide sluisdeuren mogen nooit tegelijkertijd open staan, want dit zou de lens onherstelbaar beschadigen. Als een deur weigert te sluiten en dit gedetecteerd wordt door de sensor bij de deur, mag de andere deur dus niet open gaan.
- Wanneer er op de eerste band wafers klaar liggen en de ready button is ingedrukt, worden alle wafers zo snel mogelijk verwerkt en in de bak afgeleverd. Tenzij een wafer van de band valt. Het aantal van de band gevallen wafers komt overeen met het aantal brandende ledjes op het processorbord.

Verder geldt als algemeen design principe dat motoren, luchtschakelaars en verlichting niet onnodig aangezet worden.

We maken eerst een systeemontwerp en bouwen de waferstepper. Hierna maken we een programmaontwerp in UPPAAL. Met de Systeemanalyse controleren we of het programmaontwerp goed werkt. Ten slotte implementeren we het programmaontwerp in assembly, zodat we een werkende waferstepper krijgen.

Samenvatting

Tijdens dit OGO-project hebben we een waferstepper nagemaakt. Een waferstepper is een machine die silicium wafers belicht met UV-licht om er IC's van te maken. Dit belichten gebeurt in een vacuüm, omdat UV-licht door de atmosfeer wordt geabsorbeerd en omdat de lens kapot gaat wanneer deze aan de buitenlucht wordt blootgesteld.

De waferstepper zelf hebben we gebouwd met behulp van Fischertechnik. Hiermee zijn eenvoudig de echte loopbanden, sluisdeuren, lamp en sensoren na te bootsen. We sturen de machine aan met een processorbord met daarop een Siemens SAB-C504 processor. Deze hebben we geprogrammeerd met assembleertaal. Voordat we de processor geprogrammeerd hebben, hebben we een programmaontwerp gemaakt en getest in UPPAAL.

Hoofdstuk 2

Systeemontwerp

2.1 Inleiding van het systeemontwerp

Het systeemontwerp beschrijft op een (betrekkelijk) abstract niveau de opzet van ons ontwerp voor de Wafer stepper. Hierbij hebben wij de aanwijzingen gevolgd uit het document “Projectwijzer OGO1.3”. Wij bespreken zowel de implementatie in het hardware model, als het ontwerp in UPPAAL. Hierbij wordt nog niet uitgebreid ingegaan op de software.

2.2 Hardware

2.2.1 Opzet van het model

We hebben ons systeem ingedeeld in de volgende onderdelen:

- Lamp (UV en/of LED)
- Sensor
- Lopende Band (vooruit)
- Lopende Band (voor- en achteruit)
- Startknop
- Noodknop
- Deur
- FallOff**
- Luchtklep*

- Compressor*
- Pneumatische Cilinder*
- Bak (gelukt en afval)

De onderdelen gevuld door ** kunnen niet in het hardware model worden opgenomen, dit zijn geen fysieke componenten.

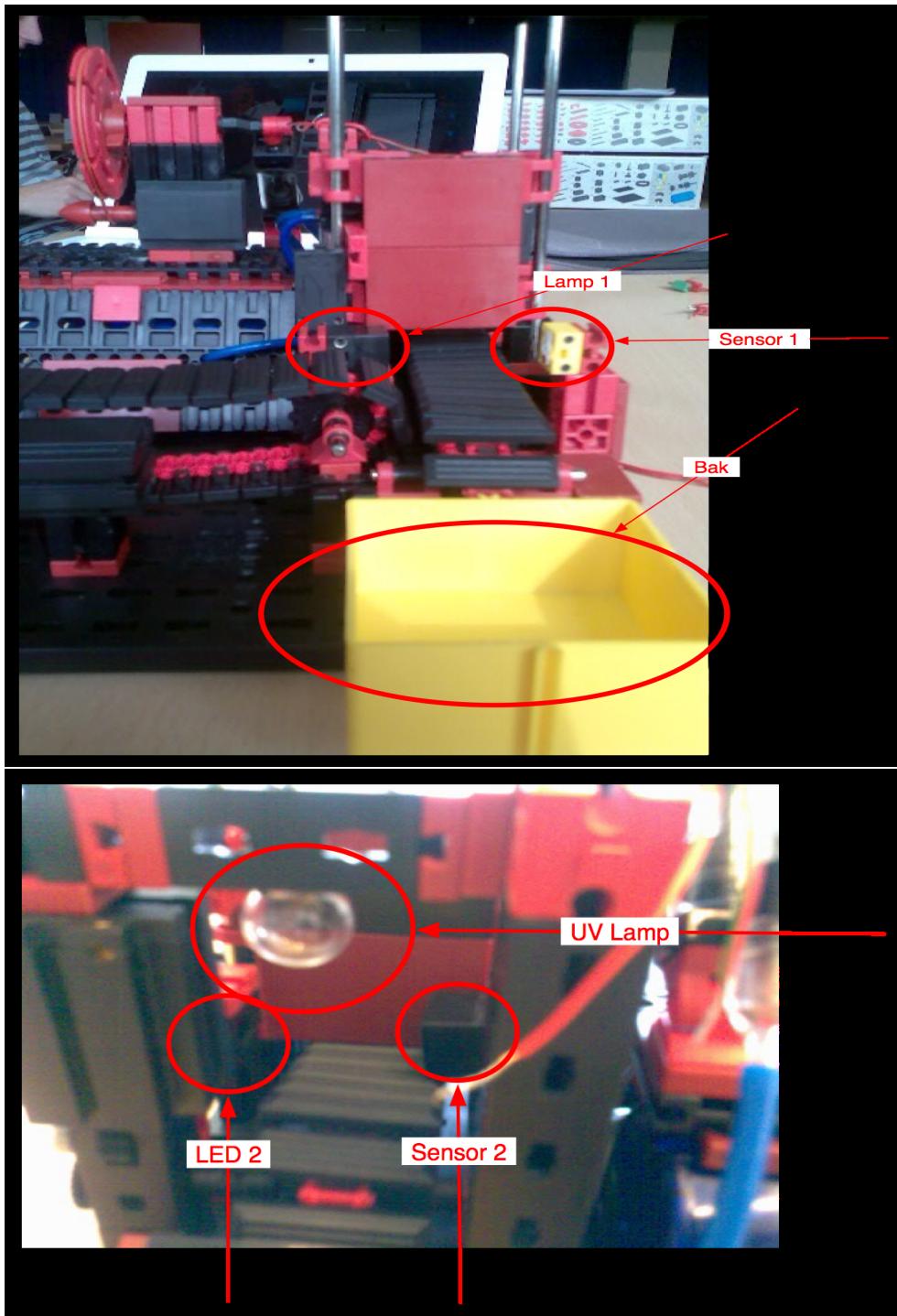
De onderdelen gevuld door * zijn niet opgenomen in het UPPAAL model. Onder andere omdat ze altijd aan moeten zijn, of door een ander onderdeel, dat wél gemodelleerd is, bestuurd worden.

Hieronder bespreken wij alle onderdelen aan de hand van foto's van ons model. Hierbij hebben wij gebruik gemaakt van foto's waarbij, na omcirkeling, een onderdeel nog herkenbaar is.

Wegens de conversie door L^AT_EX bij het invoeren van plaatjes met transparency, zijn er zwarte vlakken om onze foto's ontstaan. Onze excuses voor eventuele ongemak.

2.2.2 Foto's van het model

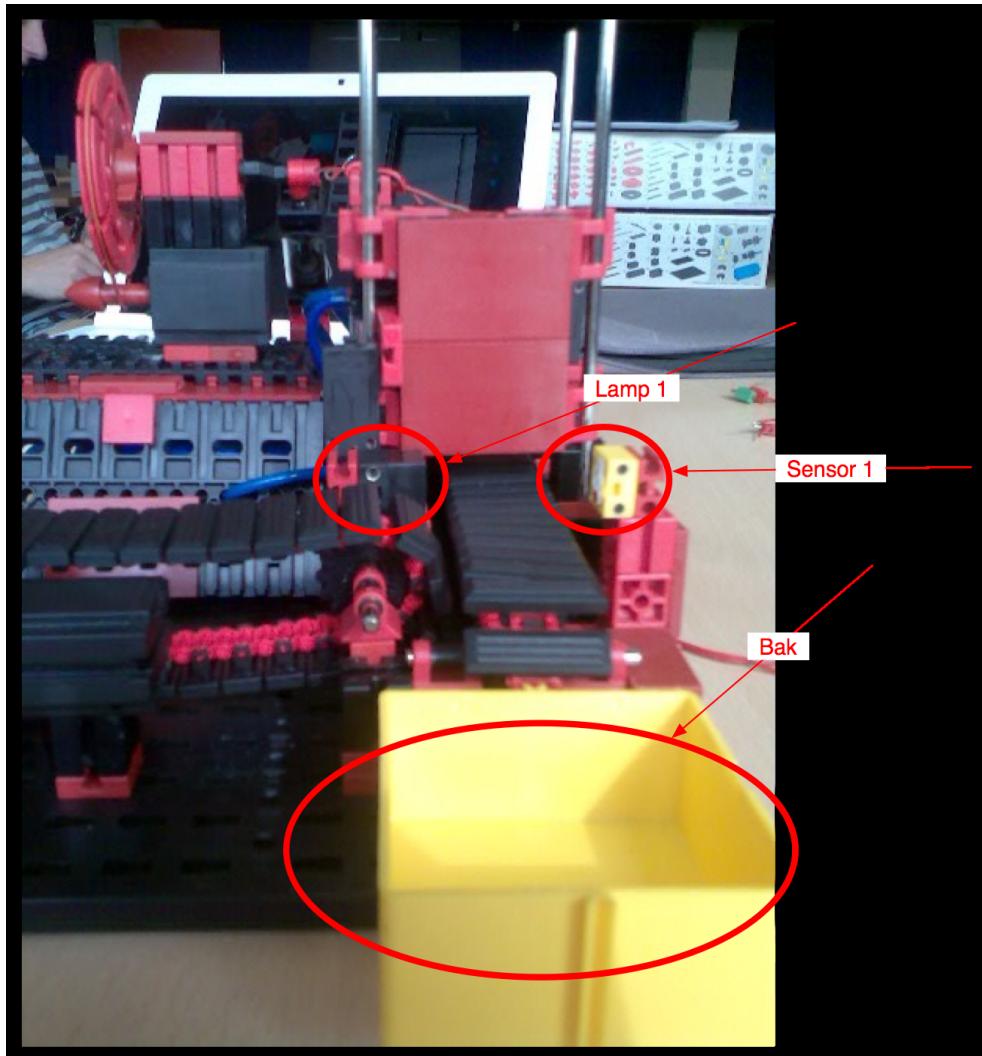
Lamp

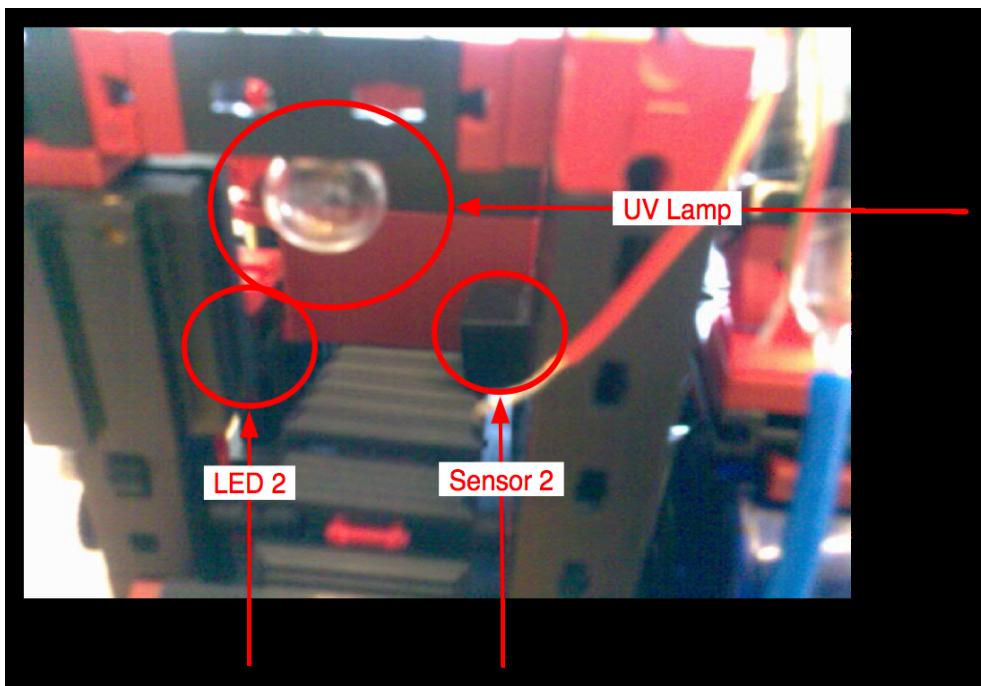


Op deze foto's zijn de eerste LED respectievelijk de UV-lamp te zien. De tweede LED kan niet worden gefotografeerd wegens obstructies.

Tegenover de twee LED's staat bij ieder één sensor, deze worden hierna behandeld.

Sensor

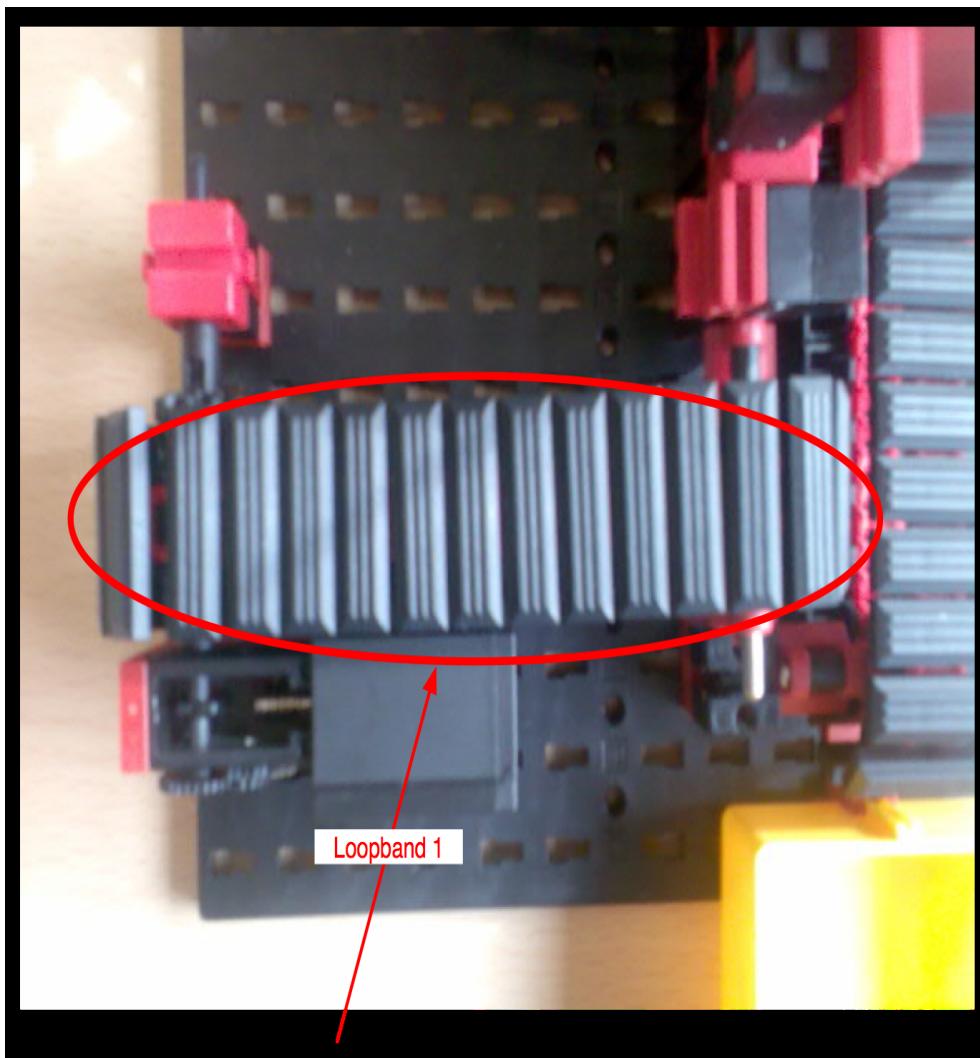




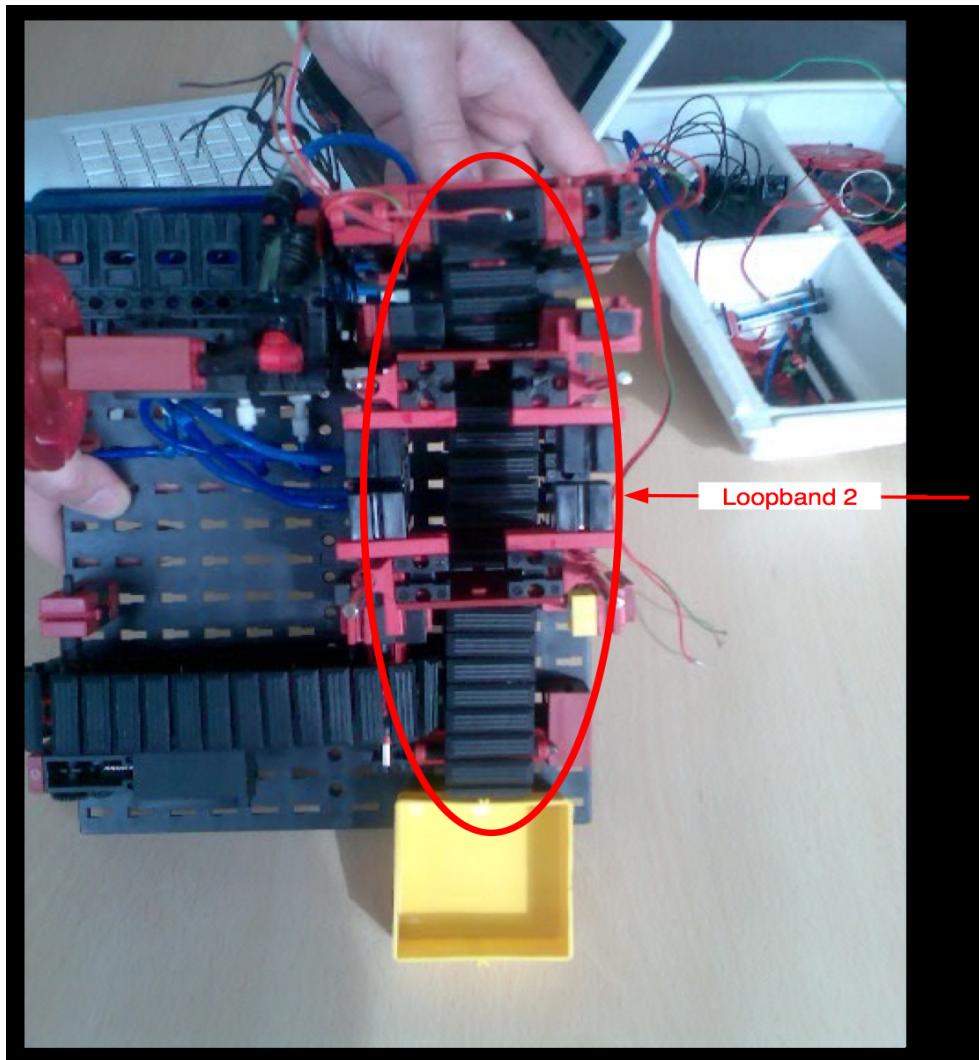
Op de foto's zijn sensor één respectievelijk sensor twee weergegeven, om te illustreren hoe deze zijn geïmplementeerd in het hardware model.

In totaal maken wij gebruik van vier sensoren, namelijk:

- R1, deze sensor “ziet” de wafer voor de eerste deur
- R2, deze sensor “ziet” de wafer achter de tweede deur
- D1, deze sensor bevestigt het sluiten van deur één
- D2, deze sensor bevestigt het sluiten van deur één

Lopende band (vooruit)

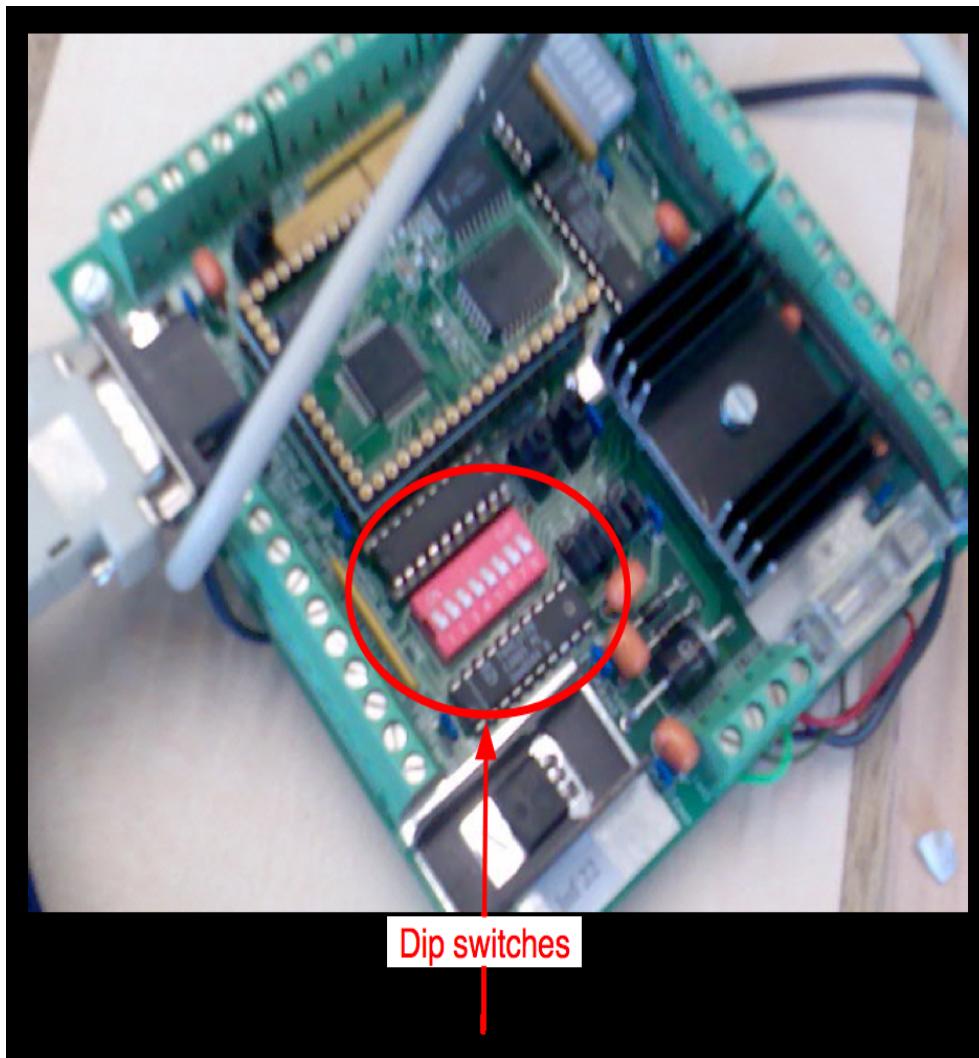
Dit is de lopende band die de wafers aanlevert. Deze kan alleen vooruit lopen.
Het grote blok ten hoogte van het midden van de band is de motor die deze loopband aandrijft.

Lopende band (vooruit)

Dit is de lopende band die de wafers door de twee poorten brengt. Deze band kan voor-en achteruit.

De motor die deze loopband aandrijft is meegenomen in de structuur van het model. Deze maakt deel uit van de pilaar waar de UV-lamp op hangt. Hierdoor hebben wij geen duidelijke foto kunnen maken van de aandrijfmotor.

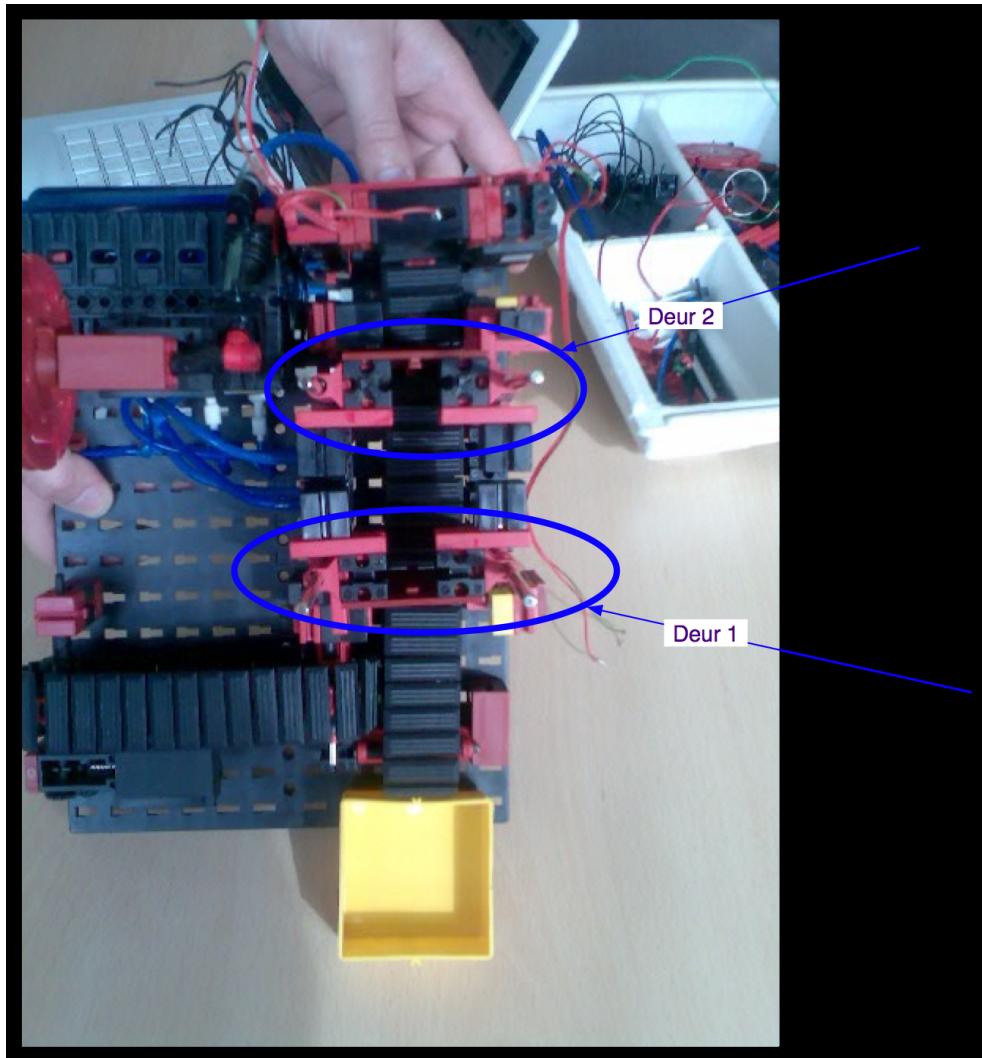
Knopen



Wij maken gebruik van een DIP-switch, op het geleverde processor bord, als startknop.
Als noodknop hebben we een aparte knop maar hier is geen foto van.

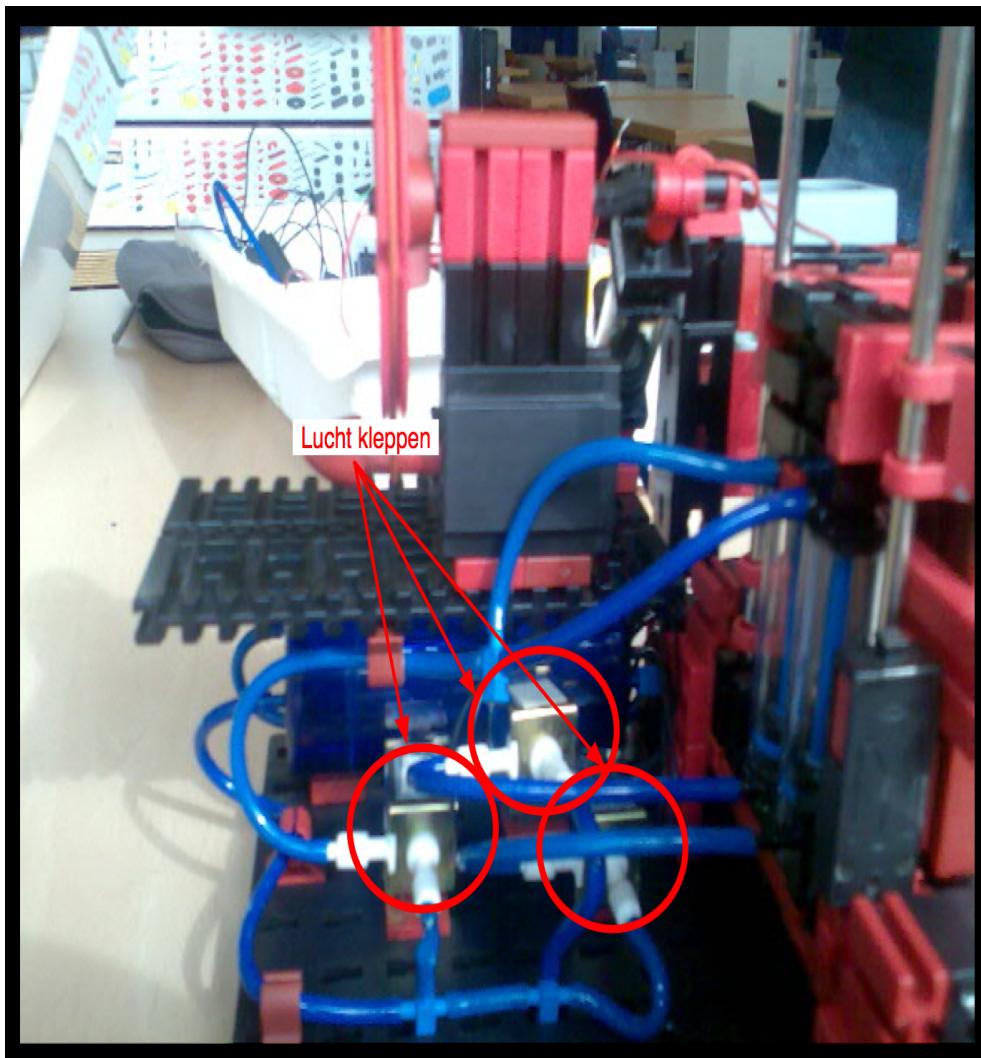
Deur



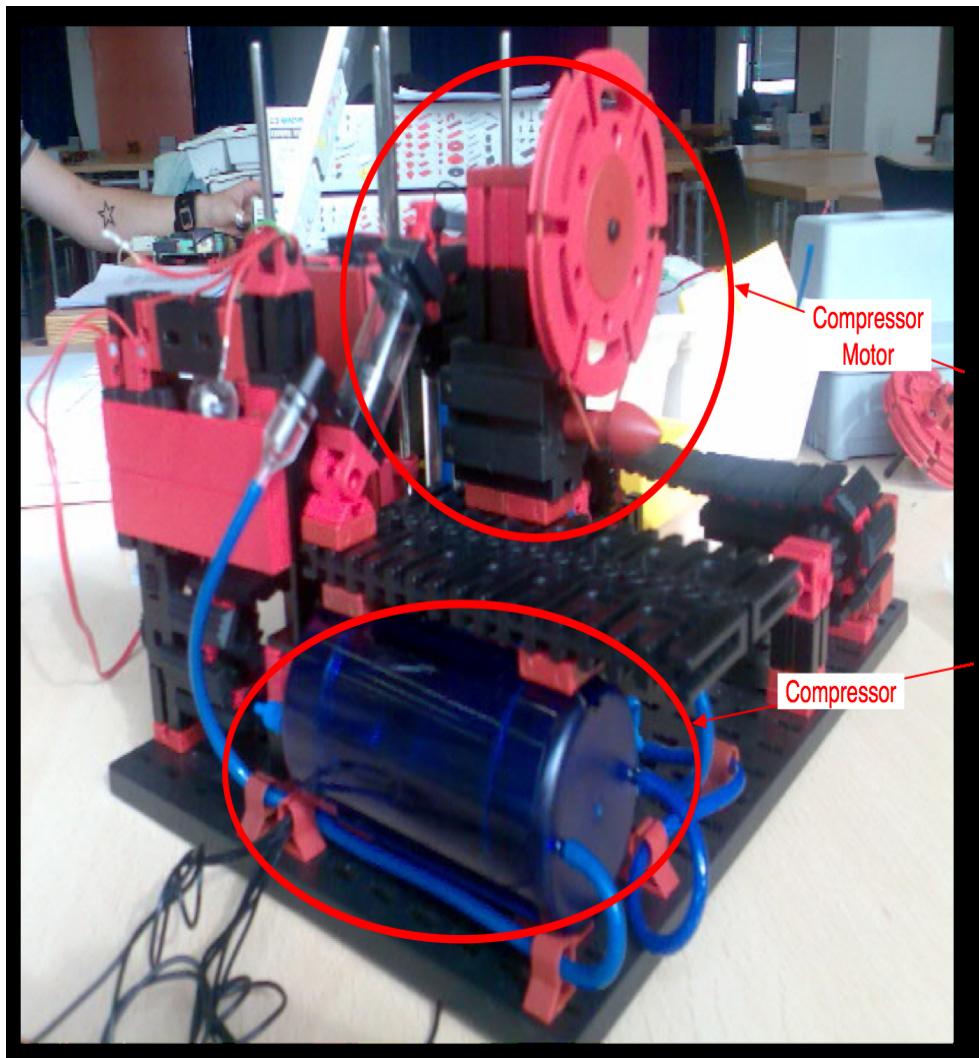


Een voor en boven aanzicht van de twee deuren die de UV-lamp van de buitenwereld scheiden, zodat deze niet beschadigd kan worden. Wegen obstructie is er geen vooraanzicht foto van de tweede deur.

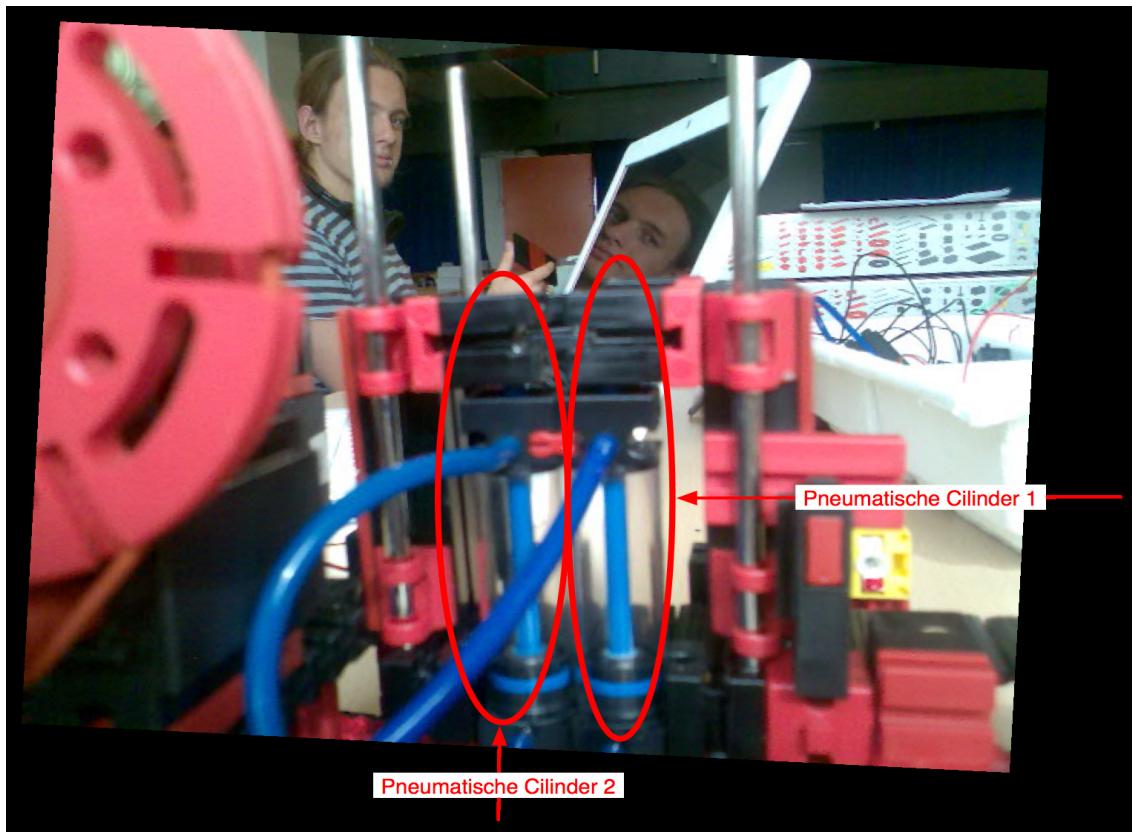
Luchtkleppen



Drie van de vier luchtkleppen die gebruikt worden om lucht wel of niet door te laten. De vierde luchtklep is na het maken van deze foto geplaatst, bij de andere drie. Er is hier geen afbeelding van.

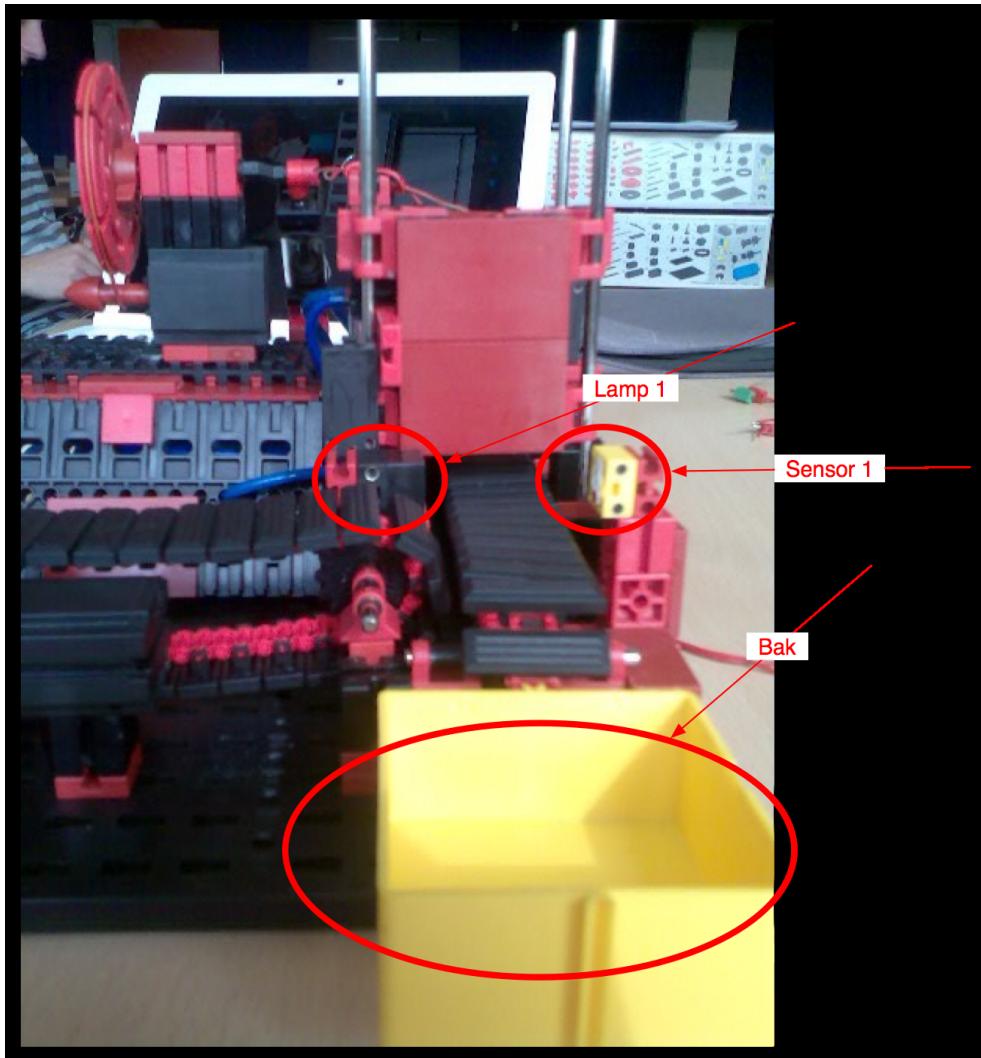
Compressor

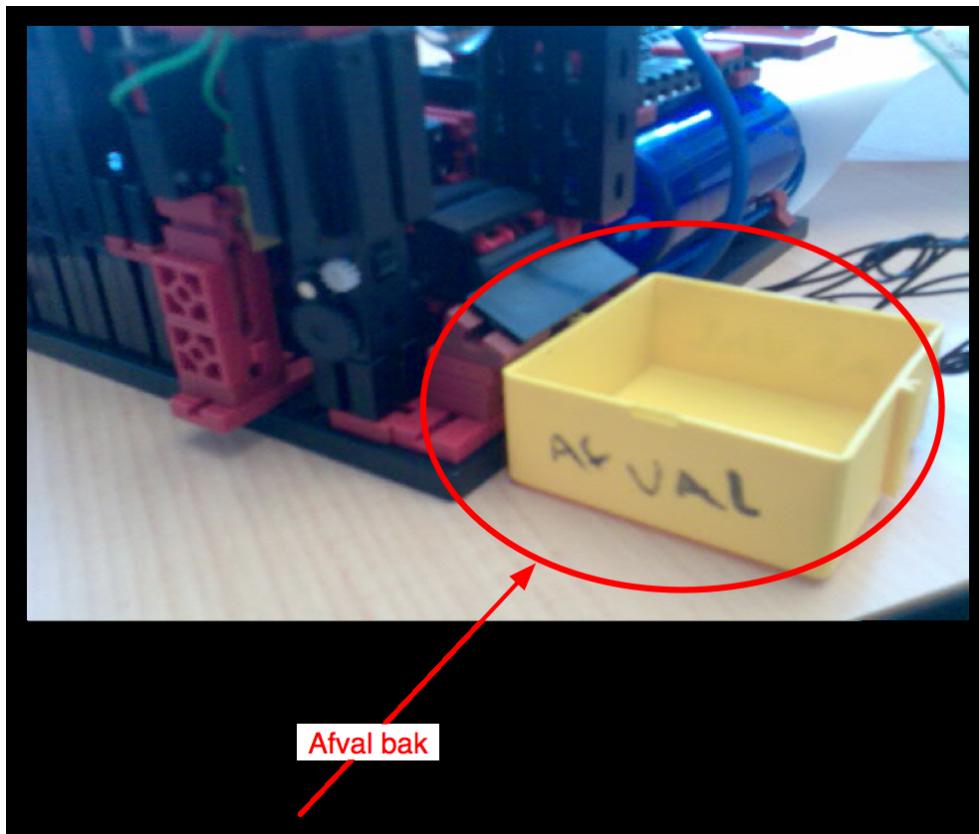
De compressor en de motor die de compressor vult. Deze begint met draaien zodra het model ingeplugged wordt op net-stroom. De compressor en diens motor draaien geheel onafhankelijk van alle andere onderdelen en kunnen niet worden beïnvloed door de processor.

Pnematische Cilinder

De pneumatische cilinder duwt de deur omhoog en trekt hem ook weer omlaag. Wij maken gebruik van twee cilinders, ieder voor één deur. De cilinders worden door de luchtkleppen aangestuurd en zijn daarom niet in het UPPAAL model opgenomen.

Bak





De twee bakken voor de gelukte wafers en de afval bak. De afvalbak staat het dichtst bij de UV-lamp. De bak voor de gelukte wafers staat ver weg van de UV-lamp, voorbij de twee deuren en ter hoogte van waar de volgende wafer wordt aangeleverd.

2.2.3 Ontwerpbeslissingen

Wij maken gebruik van twee fysieke deuren, die helemaal dicht zijn en voorkomen dat er een wafer voorbij onze deur komt. Onze deur wordt omhoog geduwd door één cilinder per deur. Dit leek ons het meest logische, een deur die van boven af wordt bestuurd houdt in dat je de luchtkleppen zou moeten inverteren (dicht is dan de valve aan zetten). Een tweede cilinder om de deur omhoog te duwen werkt niet, onze compressor kan daar niet genoeg druk voor opbouwen. Daarom maken wij gebruik van stalen staafjes waar de deuren makkelijk over glijden.

Wij hebben besloten om een wafer die mislukt is (omdat er tijdens het bestralen op de noodknop is gedrukt) in een andere bak te deponeren, de afval bak. Hoewel het makkelijker

is om de wafer te behandelen als een normale wafer, is het logischer om deze wafers apart te houden, ze zijn immers onbruikbaar.

Wij hebben ervoor gekozen om beiden deuren apart aan te sturen. Het is namelijk mogelijk om één luchtklep aan te sluiten aan beide deuren, zodat deze beiden tegelijkertijd sluiten. Omwille van mogelijk onderhoud dat gepleegd zou kunnen worden bij een echte wafer stepper, hebben wij ervoor gekozen omdat beiden deur onafhankelijk van elkaar aan te sturen.

Wij maken gebruik van twee druksensoren om te kijken of deur dicht is. Hoewel aangekondigd was dat er niet genoeg druksensoren waren om beiden deuren te controleren, hadden wij in onze dozen twee druksensoren. De zijn beiden gekoppeld aan de onderkant van de deur. Beiden deuren hebben een uitsteeksel om de druksensor te activeren indien de deur dicht is.

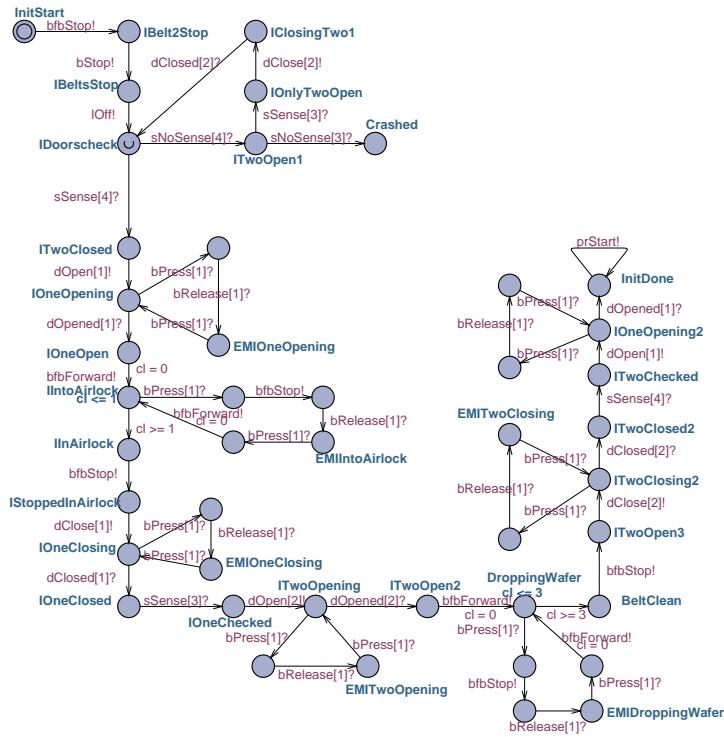
2.3 UPPAAL model

2.3.1 Templates

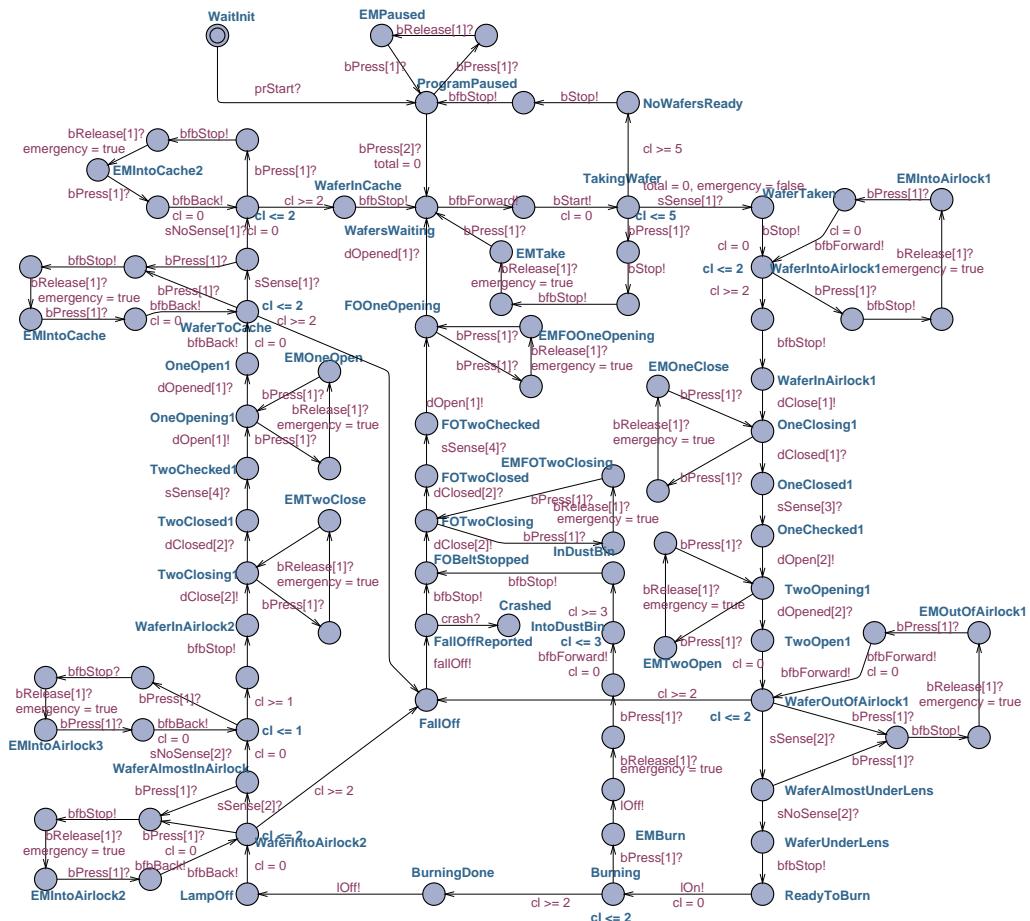
Dit zijn al onze UPPAAL componenten. *Opmerking:* deze versies van ProgramInit en ProgramRun zijn verouderd. Hierdoor kloppen een paar dingen niet meer. Zo waren bPress[1] en bRelease[1] eerst kanalen die aangaven dat de noodknop ingedrukt respectievelijk losgelaten werd, maar zijn er nu een nieuwe noodknop-template en daardoor (handigere) emStart en emStop kanalen. Zie voor een goede versie van het hoofdprogramma het programmaontwerp.

- “ProgramInit” wordt aan het begin van het programma uitgevoerd. Dit zorgt ervoor dat alle actoren in een wenselijke toestand zijn. Als ProgramInit klaar is, wordt ProgramRun aangeroepen. *Opmerking:* we mogen er van uit gaan dat alles in het begin in de wenselijke toestand is, dit hebben we in het programmaontwerp dan ook gedaan, waardoor daar geen ProgramInit of equivalent daarvan te vinden is.

Program Init

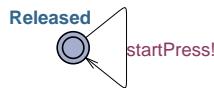


Program Run



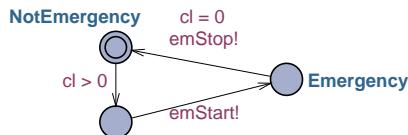
- “StartButton” staat voor de startknop. Deze kan alleen ingedrukt worden, er wordt dan een signaal verstuurd waardoor het systeem start.

StartButton



- “EMButton” staat voor de noodknop. Hier zijn drie toestanden mogelijk. Eén toestand stelt een emergency voor, één toestand wordt bereikt als het systeem uit een emergency komt en de laatste toestand zorgt ervoor dat het systeem niet gelijk weer in emergency kan gaan als een emergency afgelopen is.

EMButton



- “Lamp” staat voor een LED of de UV-lamp. Deze kan aan of uit staan.

Lamp



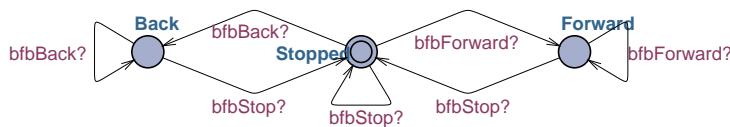
- “Conveyor” is de lopende band die maar 1 kant op kan lopen. Deze kan dus ‘aan’ (running) of ‘uit’ (stopped) zijn.

Conveyor



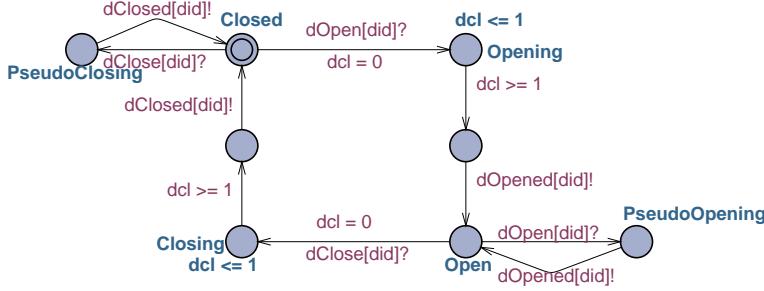
- “ConveyorBF” kan heen en terug lopen. Er zijn dus toestanden voor back, forward en stopped nodig.

ConveyorBF



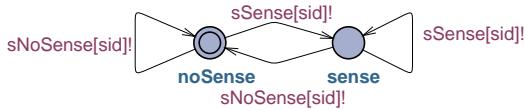
- “Door” is het model van een deur. Een deur kan open of dicht zijn. Als een geopende deur een open-signaal krijgt (of een gesloten deur een sluit-signaal), dan wordt dit verwerkt door meteen aan te geven dat de deur klaar is met openen (of sluiten).

Door



- “Sensor” is een licht- of drucksensor. Een sensor kan Sense of NoSense zijn - hij neemt iets waar of hij neemt niets waar.

Sensor



- “FallOff” - dit is een speciaal geval. Het is niet echt een hardware-component (ondanks dat er wel LEDs aan en uit gaan ten gevolge van de toestand van dit onderdeel). FallOff houdt bij hoeveel wafers van de band gevallen zijn. Zodra er vijf wafers verloren zijn gegaan ‘crasht’ ons systeem (dit is een noodtoestand). We houden dit op deze manier bij omdat de hoeveelheid verloren wafers van groot belang is voor de werking van het systeem.

falloff



Veel van de componenten hebben een onbekende beginstand, een gegeven dat gemodelleerd is door vanuit Init een stap te maken naar een willekeurige toestand.

Daarnaast wordt van veel componenten verwacht dat ze een signaal kunnen versturen, (“deze deur staat open!”) - dit wordt gedaan door in elke ‘stabiele’ toestand (de deur staat open) een loopje naar dezelfde toestand te maken. Deze loop stuurt een signaal over het gewenste kanaal.

2.3.2 Ontwerp Beslissingen

We hebben expliciet gekozen om geen wafer-element toe te voegen. Dit zou namelijk betekenen dat de wafers zelf te besturen zijn, iets dat niet zo is. Er is helemaal niets bekend

over de wafer, alleen over het feit dat sensoren iets registreren of niet. Uit deze informatie kan dan weer worden afgeleid waar een wafer is.

“FallOff” is expliciet gedeclareerd omdat er bepaalde lampjes zijn die overeenkomen met een aantal wafers die weg zijn. Elke toestand van “FallOff” bestuurd nu impliciet die lampjes. Daarnaast is het zo dat zodra de vijfde wafer van de band af valt er een noodstop gemaakt moet worden.

Hoofdstuk 3

Specificatie elektrische interface

3.1 Priklijst

Verklaring van afkortingen:

Mx	Motor X.
Mx+ / Mx-	Motor X, + pool, -pool
Lox	Lopende Band X
Vx	Valve X (de luchtkleppen)
Lx	Lamp X
Rx	Receiver X – de fotocellen
Dx	Druksensor X
Kx	Knop X

Electrische aansluitingen:

Er is 1 (één) ground, waarop elke groundpool aangesloten wordt! Deze ground wordt op de scartkabel aangesloten op pin 20. De +5 V wordt aangesloten op pin 19.

Apparaat	Pluspool	Minpool	Scartpin
M1	PWR0	Ground	1/20
M2**	PWR1	PWR2	2/3
V1	PWR3	Ground	4/20
V2	PWR4	Ground	5/20
V3	PWR5	Ground	6/20
V4	PWR6	Ground	7/20
D1	Ground	P1.4	20/8
D2	Ground	P1.6	20/9
L1	+5V (Voeding)	Ground	-/-
L2	+5V (Voeding)	Ground	-/-
L3 (UV)	PWR7	Ground	10/20
R1*	Ground	P3.4	20/11
R2*	Ground	P3.5	20/12
K1 (Start/Ready)	IN0	IN0	-/- (DIP)
K2 (Noodstop)	P3.2	Ground	13/20

* Alle fotocellen worden dmv een comparator op de processor aangesloten, omdat de fotocellen een variabele weerstand hebben die zich verhoudt tot de hoeveelheid licht die op de cel valt. Deze comparator is tussen de processor en de eerste scartplug aangesloten.

Aanvulling:

Na aanpassing van de solderingen, wordt pin 19 op de scartplug niet meer gebruikt, de +5V aansluitingen die we nog gebruiken zitten aan een aparte voeding.

Pneumatische apparaten:

Apparaat	In	Uit
Compressor	-	T-Splitter
V1	Compressor	Deur 1 open
V2	Compressor	Deur 1 dicht
V3	Compressor	Deur 2 open
V4	Compressor	Deur 2 dicht

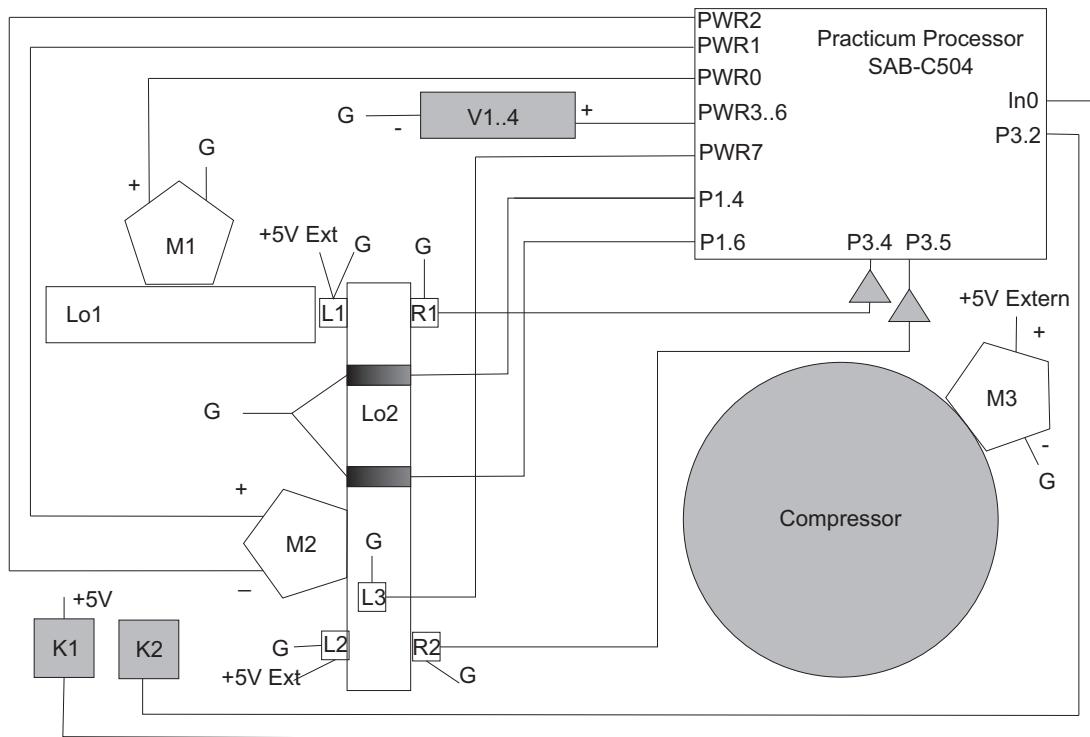
Extra benodigdheden (naast de fischerdoos dus)

Naast wat kabeltjes en een soldeerbout:

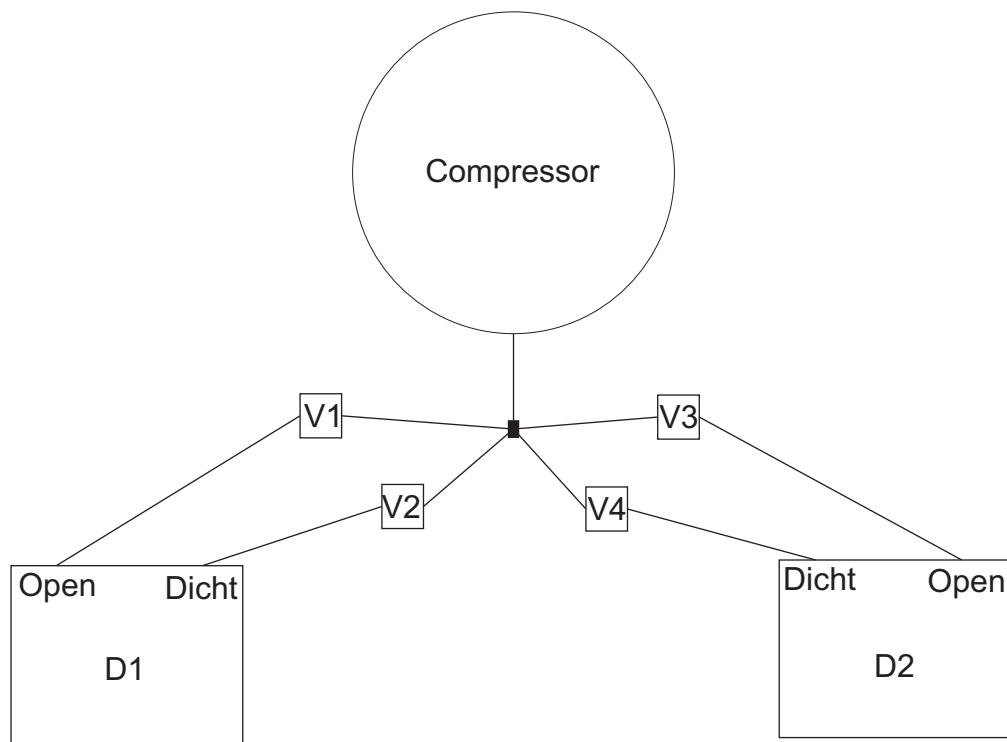
3 comparators. Deze zijn toegelaten door dhr Moussavi.

1 Drukknop voor de noodstop

3.2 Bedradingsschema



3.3 Beluchtingsschema



We gebruiken één compressor waarop vier luchtkleppen zijn aangesloten. Eén luchtklep dient om deur 1 te openen, één om deur één te sluiten, er is er één om deur twee te openen, de laatste sluit deur twee. We hebben overwogen om één luchtklep te gebruiken om beide deuren te sluiten, omdat toch nooit beide deuren gelijk open mogen zijn. We hadden toen echter ons beluchtingsschema al ingeleverd en hebben het dus maar bij het originele ontwerp gelaten.

Hoofdstuk 4

Programmaontwerp

4.1 Inleiding van het programmaontwerp

Het programmaontwerp is een netwerk van UPPAAL-automaten die beschrijven hoe het gedrag van het programma dat de waferstepper aanstuurt hoort te zijn. In dit document zullen we beschrijven hoe we dit netwerk van automaten gemodelleerd hebben en waarom we dat op deze manier gedaan hebben.

We gaan er in dit verslag vanuit dat alle aangesloten apparaten een nette begintoestand hebben, dat wil zeggen: de deuren zijn gesloten, lopende banden staan stil, de lamp voor het belichten is uit en sensoren rapporteren niets.

4.2 Het UPPAAL ontwerp voor de subroutines

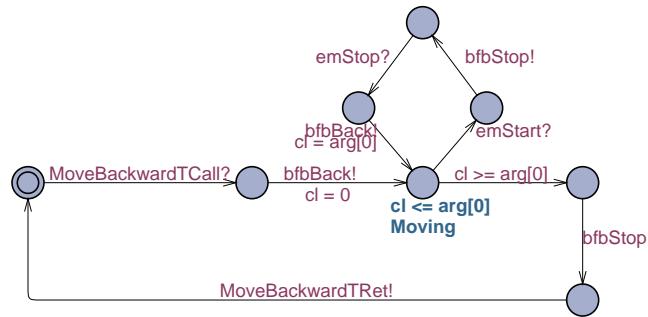
Het UPPAAL ontwerp is verdeeld in verschillende templates die allemaal verantwoordelijk zijn voor een bepaalde actie en daarnaast een template die alle anderen aanstuurt. Al deze templates zullen in deze sectie beschreven worden, behalve die van het hoofdprogramma, die in de volgende sectie aan bod komt.

De templates in dit hoofdstuk maken gebruik van de templates uit hoofdstuk 2, die de daadwerkelijke hardware modelleren.

4.2.1 Bediening lopende banden

Er zijn in ons programmaontwerp een zevental automaten die op een bepaalde manier een lopende band aansturen, die weer verder verdeeld kunnen worden in drie groepen:

- Gedurende een bepaalde tijd loopband twee voor- of achteruit bewegen
- Loopband twee voor- of achteruit bewegen totdat een sensor iets ziet of totdat er een time-out optreedt



Figuur 4.1: UPPAAL-automaat voor MoveBackwardT. MoveForwardT is vrijwel identiek.

- Een wafer van de eerste loopband proberen te pakken en op loopband twee te leggen.

Een bepaalde tijd bewegen: MoveBackwardT en MoveForwardT

Figuur 4.1 bevat een afbeelding van de automaat die MoveBackwardT implementeert. De automaat van MoveForwardT is vrijwel identiek, het enige verschil is dat er bfbForward! bij de pijlen naar de Moving toestand staat in plaats van bfbBack!.

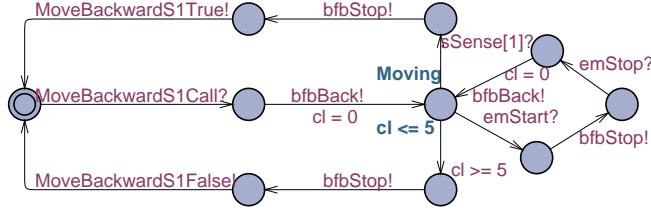
Deze routine geeft de loopband opdracht om te beginnen met draaien, komt dan in de Moving toestand en blijft daar totdat de band gedurende de gevraagde tijd gedraaid heeft. Vervolgens laat het de band weer stoppen en geeft een MoveBackwardTRet signaal om het hoofdprogramma zijn uitvoering weer te laten hervatten. De gevraagde tijd is de waarde die arg[0] heeft, een variabele die de eerste parameter bij het aanroepen van een routine voorstelt.

Een uitzondering hierop is als de noodknop wordt ingedrukt terwijl deze routine in de Moving toestand is. Op dat moment wordt de band ook stopgezet totdat er weer op de noodknop wordt gedrukt. Daarna zal de band opnieuw de volledige gevraagde tijd draaien. We zijn ons ervan bewust dat dit in feite niet de bedoeling is, maar we konden in UPPAAL geen makkelijke manier vinden om dit correct te modelleren, in het Assembly-programma wordt dit wel correct gplementeerd.

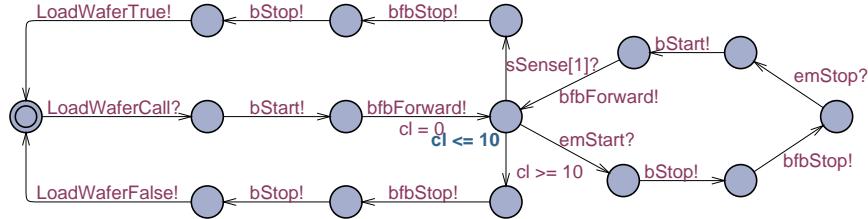
Bewegen totdat een sensor iets waarneemt: MoveForwardS1, MoveForwardS2, MoveBackwardS1, MoveBackwardS2

Figuur 4.2 is de automaat van MoveBackwardS1, de andere automaten lijken hier sterk op, behalve de richting waarin ze de band laten draaien en welke sensor ze controleren.

Deze subroutine geeft de tweede lopende band opdracht in een bepaalde richting te gaan bewegen en komt dan in de Moving toestand. Vervolgens zijn er twee mogelijke uitkomsten:



Figuur 4.2: UPPAAL-automaat voor MoveBackwardS1. MoveBackwardS2, MoveForwardS1 en MoveForwardS2 zijn vrijwel identiek.



Figuur 4.3: UPPAAL-automaat voor LoadWafer.

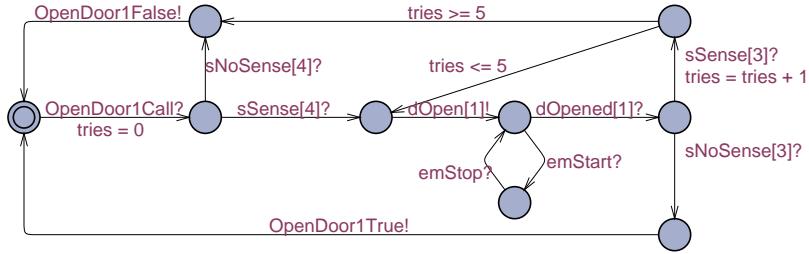
- Als de lichtsensor iets waarneemt, wordt de band gestopt en een MoveBackwardS1True signaal aan het hoofdprogramma gegeven, dat aangeeft dat er inderdaad iets voor de sensor ligt.
- Als de lichtsensor na een bepaalde tijd nog niets gezien heeft, wordt de band gestopt en een MoveBackwardS1False signaal gegeven aan het hoofdprogramma om aan te geven dat er een time-out opgetreden is zonder dat de lichtsensor iets gedetecteerd heeft.

Een laatste mogelijkheid is nog dat er op de noodknop wordt gedrukt terwijl het programma in de Moving toestand is. In dit geval zal de band stoppen en als er weer op de noodknop wordt gedrukt begint de band opnieuw te lopen met de time-out weer teruggezet naar zijn begintoestand. In het assemblyprogramma zal deze time-out gewoon zijn waarde behouden.

Een wafer van de eerste lopende band pakken: LoadWafer

Deze subroutine start beide lopende banden en komt dan in de Moving toestand. er zijn weer twee mogelijkheden om de moving toestand te verlaten:

- Als lichtsensor één iets waarneemt, ligt er blijkbaar een wafer klaar op band twee.



Figuur 4.4: UPPAAL-automaat voor OpenDoor1. OpenDoor2 is vrijwel identiek.

Dan worden beide banden gestopt en krijgt het hoofdprogramma een LoadWaferTrue signaal om aan te geven dat er succesvol een wafer geladen is.

- Als er tien seconden voorbij zijn en lichtsensor één heeft nog steeds niets waarnomen, lagen er blijkbaar geen wafers klaar op de eerste loopband. In dat geval worden beide loopbanden stilgezet en krijgt het hoofdprogramma een LoadWaferFalse signaal, om aan te geven dat er niet succesvol een wafer van de eerste band is gehaald.

Als er op de noodknop wordt gedrukt terwijl het programma in de Moving toestand is worden beide loopbanden gestopt. Als de noodknop dan weer wordt ingedrukt beginnen beide loopbanden weer en wordt begint de time-out weer in zijn begintoestand.

Het is mogelijk dat er al een tweede wafer op de tweede lopende ban ligt voordat de lichtsensor de eerste detecteert. Volgens de opdrachtbeschrijving zouden de wafers echter met voldoende tussenruimte op de eerste band liggen op het moment dat er op de startknop wordt gedrukt en in dat geval levert dit geen problemen op.

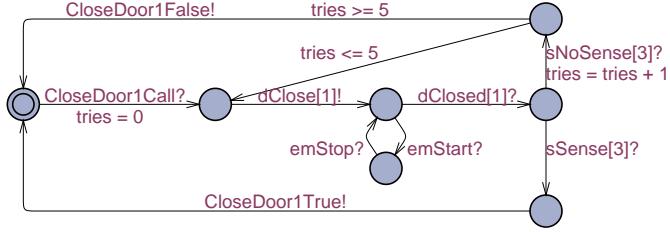
4.2.2 Bediening deuren

Voor de bediening van de deuren zijn er vier subroutines, die verdeeld kunnen worden in twee groepen:

- Deuren openen
- Deuren sluiten

Deuren openen: OpenDoor1, OpenDoor2

In figuur 4.4 is de automaat van OpenDoor1 afgebeeld, die van OpenDoor2 is hetzelfde, behalve dat die met andere sensoren en met de andere deur communiceert.



Figuur 4.5: UPPAAL-automaat voor CloseDoor1. CloseDoor2 is vrijwel identiek.

Als deze routine wordt aangeroepen wordt eerst d.m.v. de druksensor bij deur twee gekeken of deze deur wel echt dicht is.

Als de deur open blijkt te zijn krijgt het hoofdprogramma een OpenDoor1False signaal, als de deur wel dicht is geeft deze routine opdracht om deur één te openen. Daarna controleert de subroutine of de deur ook echt opengegaan is d.m.v. de druksensor bij deur één.

Als de deur inderdaad open is krijgt het hoofdprogramma een OpenDoor1True signaal om aan te geven dat de deur succesvol opengemaakt is.

Als de deur niet succesvol opengegaan is zal dit opnieuw geprobeerd worden, als de deur na vijf pogingen nog steeds niet open is krijgt het hoofdprogramma een OpenDoor1False signaal om aan te geven dat het openen van de deur mislukt is.

Deuren sluiten: CloseDoor1, CloseDoor2

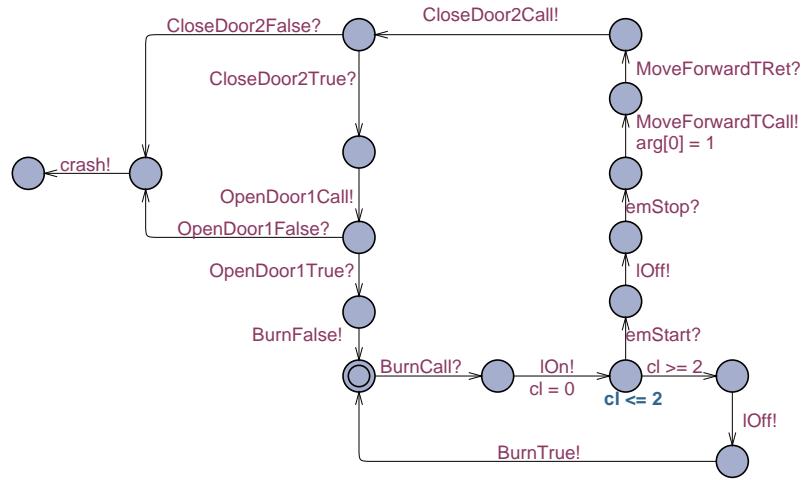
Figuur 4.5 is de automaat van CloseDoor1, die van CloseDoor2 is identiek, behalve dat die met de andere deur en een andere sensor communiceert.

In feite is deze subroutine ook identiek aan de OpenDoor-subroutines, behalve natuurlijk dat de deur hier gesloten wordt in plaats van geopend. Een tweede verschil is dat hier niet eerst wordt gecontroleerd in welke toestand de andere deur is, omdat het sluiten van een deur altijd is toegestaan.

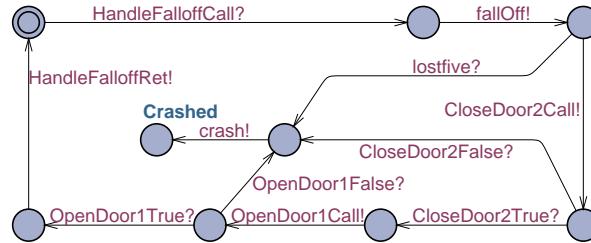
4.2.3 Het belichten van een wafer: Burn

De subroutine hierboven handelt het belichten van een wafer af. In normale omstandigheden doorloopt deze subroutine alleen de onderste lus, die simpelweg de lamp inschakelt, twee seconden wacht, de lamp weer uitschakelt en dan een BurnTrue signaal geeft aan het hoofdprogramma om aan te geven dat het belichten goed gegaan is.

Als echter de noodknop wordt ingedrukt terwijl de lamp aan is gaat de subroutine naar de bovenste lus, die in eerste instantie alleen maar de lamp uitschakelt. Als de noodknop weer wordt ingedrukt ten teken dat het systeem weer verder mag gaan zal de wafer naar de



Figuur 4.6: UPPAAL-automaat voor Burn.



Figuur 4.7: UPPAAL-automaat voor HandleFalloff.

afvalbak vervoerd worden en vervolgens worden de deuren weer in de juiste stand gezet voor het verwerken van een volgende wafer, dus deur twee dicht en deur één open. Vervolgens krijgt het hoofdprogramma een signaal BurnFalse om aan te geven dat het belichten mislukt is en dat begonnen kan worden met de verwerking van een eventuele volgende wafer. Merk op dat bij de afhandeling van deze noodstop gebruik wordt gemaakt van een aantal eerder in dit document behandelde subroutines.

Als n van deze deuren niet goed open of dicht gaat komt de subroutine in de twee toestanden uiterst links en zal het programma crashen.

4.2.4 Verloren wafers afhandelen: HandleFalloff

Als deze subroutine wordt aangeroepen wordt eerst een fallOff signaal gegeven aan de teller die het aantal verloren wafers bijhoudt. Als dan blijkt dat dit de vijfde wafer was



Figuur 4.8: UPPAAL-automaat voor ManageCrash

die verloren ging gaat het programma naar de toestand in het midden en crasht het.

Als er nog geen vijf wafers kwijtgeraakt zijn, zal deze subroutine zorgen dat de waferstepper klaar is om de volgende wafer af te handelen, dat wil zeggen: deur twee wordt gesloten en deur één geopend. Als één van deze deuren niet goed open of dicht gaat, gaat de subroutine naar de toestanden in het midden en vervolgens crasht het programma.

N.B.: het lijkt alsof op de noodstop drukken in deze subroutine geen effect heeft, dit is misleidend omdat deze subroutine bijna helemaal bestaat uit andere subroutines waarin de noodknop wél wordt afgevangen. Het lostFive! signaal wordt verzonden vanuit de fallOff template in sectie 2.3.1.

4.2.5 Hulpautomaat om crashes te detecteren: ManageCrash

Zolang het programma draait zit deze automaat in de “running” state. Zodra het programma ergens om een bepaalde reden crasht, dit kan zijn als een deur weigert te openen/sluiten of als er vijf wafers kwijt zijn, wordt deze automaat in de “crashed” state gezet. Het nut van deze automaat is dat we gemakkelijk kunnen testen of het programma gecrasht is of niet, we hoeven nu namelijk niet alle afzonderlijke crash toestanden in de verschillende andere automaten te testen.

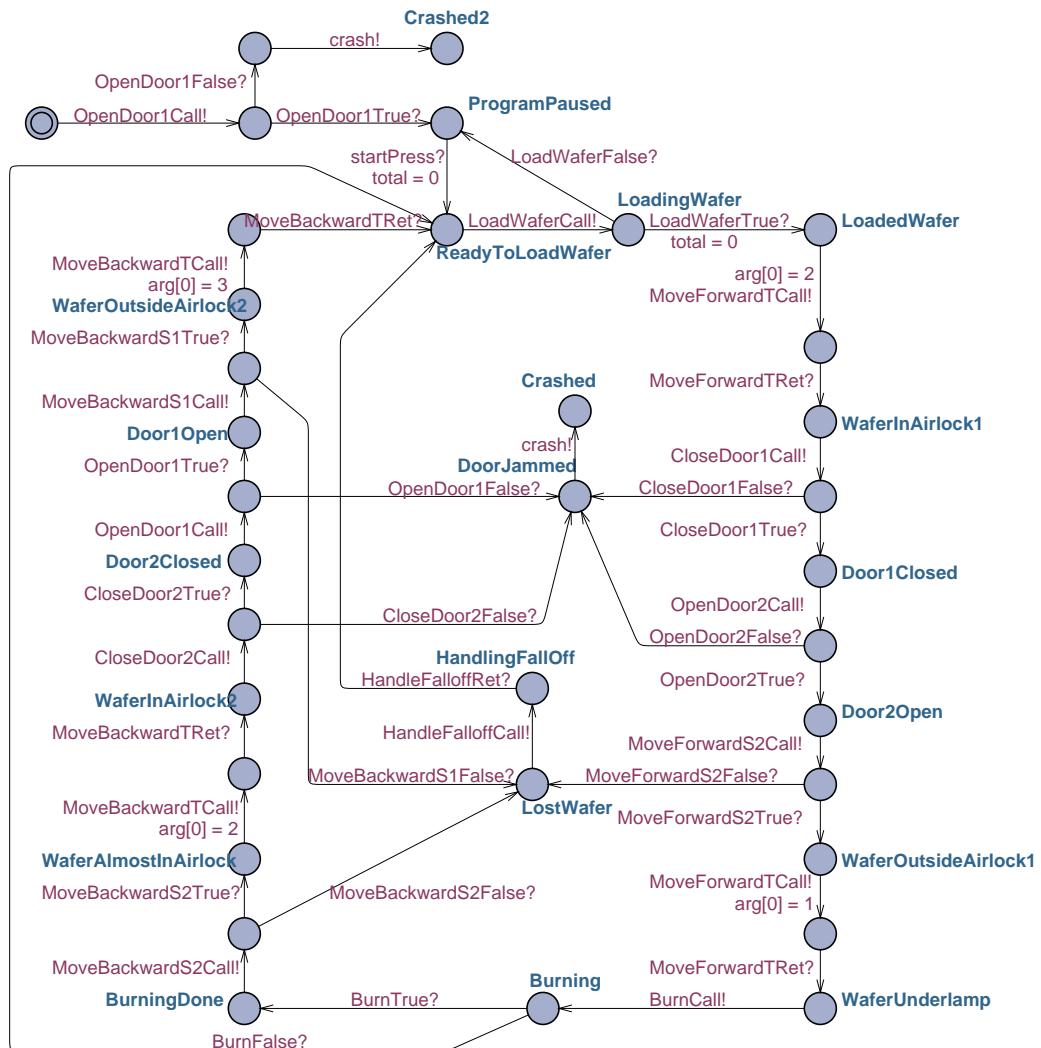
4.3 Het hoofdprogramma

Het hoofdprogramma is gemodelleerd als een UPPAAL-template die de templates uit het vorige hoofdstuk aanstuurt op een zodanige manier dat het programma uiteindelijk doet wat het zou moeten doen: Het belichten van wafers en ze vervolgens in de daarvoor bestemde bak deponeren.

4.3.1 Het ontwerp

Figuur 4.9 bevat de UPPAAL template voor het hoofdprogramma.

Het eerste dat het programma doet is deur één openen, omdat in de begintoestand beide deuren gesloten zijn. Als dit mislukt crasht het programma onmiddellijk, anders gaat het naar de ProgramPaused toestand.



Figuur 4.9: UPPAAL-automaat van het hoofdprogramma.

In deze toestand wacht het programma totdat er op de startknop wordt gedrukt die aangeeft dat er wafers klaarliggen op de eerste band. Als deze knop wordt ingedrukt wordt LoadWafer aangeroepen. Dan wacht het programma totdat het een signaal terugkrijgt van deze subroutine.

Als dit signaal LoadWaferFalse is, is er geen wafer van de eerste band gehaald en gaat het programma terug naar ProgramPaused, omdat er blijkbaar toch geen wafers klaarlagen.

Als het programma een LoadWaferTrue signaal krijgt, ligt er een wafer klaar vlak voor de eerste deur van de luchtsluis en zal het programma deze wafer door de luchtsluis transporteren en onder de lamp leggen met behulp van de subroutines voor het bewegen van de lopende band en het openen en sluiten van deuren. Als dit alles goed gaat komt het programma in de WaferUnderLamp toestand.

In dit traject kunnen echter enkele dingen misgaan. Er kan namelijk een deur weigeren open of dicht te gaan. In dat geval gaat het programma naar de DoorJammed toestand en vervolgens zal het crashen en in de Crashed toestand terechtkomen.

Iets anders dat fout kan gaan is dat de wafer de tweede lichtsensor niet bereikt op het moment dat dat wel zou moeten. In dat geval wordt geconcludeerd dat de wafer verloren is gegaan en gaat het programma naar de LostWafer toestand. Daarna zal HandleFalloff aangeroepen worden. Als deze routine returnt is het systeem klaar om de volgende wafer te verwerken en gaat het programma naar ReadyToLoadWafer.

Als alles echter goed gaat, ligt de wafer op dit moment onder de lamp en wordt er een aanroep gedaan naar Burn. Dit zorgt er in principe voor dat de wafer belicht wordt en dat het hoofdprogramma een BurnTrue signaal krijgt.

Als er echter tijdens het belichten een noodstop plaatsvindt krijgt het hoofdprogramma een BurnFalse signaal. In dit geval is het systeem klaar voor het verwerken van de volgende wafer en gaat het programma dus terug naar ReadyToLoadWafer.

Als alles goed gaat tijdens het belichten komt het programma in de toestand Burning-Done. Vervolgens wordt de wafer door de luchtsluis naar de opvangbak getransporteerd. Dit proces en de zaken die hierbij mis kunnen gaan zijn vergelijkbaar met het naar binnen brengen van de wafer en zal hier ook niet verder toegelicht worden.

Als de wafer in de bak ligt komt het systeem in toestand ReadyToLoadWafer en zal proberen de volgende wafer van de eerste band te halen. Als dit lukt begint het hele verhaal weer opnieuw, anders zijn blijkbaar alle wafers verwerkt en dan gaat het programma naar de ProgramPaused toestand, waar het wacht tot er op de startknop wordt gedrukt om aan te geven dat er weer wafers klaarliggen.

4.4 Implementatie

Dit programma hebben wij geïmplementeerd in assembly. Zie hiervoor bijlage A.

Hoofdstuk 5

Systeemanalyse

5.1 Inleiding van de systeemanalyse

Bij de systeem analyse controleren we of ons UPPAAL-model correct is. Om dit te controleren moeten we eerst definiëren wat we verstaan onder „correct”. Wij zullen dit definiëren met behulp van eigenschappen waaraan ons systeem dient te voldoen; wanneer dit het geval is, achten wij het systeem correct. Deze eigenschappen zullen we toelichten, vertalen naar UPPAAL-queries, en testen volgens het verificatiesysteem van UPPAAL zelf. De resultaten zullen we ook evalueren in dit document.

5.2 Belangrijke eigenschappen functionaliteit

Hieronder is beschreven aan welke eigenschappen ons systeem moet voldoen. Het vierde item kan echter niet gegarandeerd worden bij indrukken van de noodknop; met deze uitzondering zal bij de simulatie en verificatie rekening worden gehouden.

1. Er zijn nooit twee deuren tegelijk open.
2. De lamp is alleen aan bij belichten van een wafer.
3. Vlak voordat een wafer aan „zijn cyclus” begint is deur één open en deur twee dicht.
4. Voor alle paden geldt dat een wafer binnen twintig seconden in een bak ligt of verloren is gegaan, mits er niet op de noodknop wordt gedrukt.
5. Het systeem crasht alleen wanneer het moet crashen.

5.3 Methodiek simuleren en verifiëren eigenschappen

Voor het simuleren van het systeem gebruiken wij de Simulator welke in UPPAAL beschikbaar is. Deze laten we random acties simuleren op de hoogste snelheid voor een duur van maximaal vijf minuten. Dit proces herhalen we enkele malen om er zeker van te zijn dat ons systeem inderdaad geen deadlocks bevat.

Bij het verifiëren maken wij gebruik van z.g.n. queries, welke uitdrukken wat precies wij willen verifiëren. De nummering is gelijk aan die van de vorige lijst:

1. $A[]$ (door1.Closed or door1.PseudoClosing or door2.Closed or door2.PseudoClosing)
Er moet altijd gelden dat er minstens één deur gesloten is. De toestandsnaam PseudoClosing zou kunnen impliceren dat de deur aan het sluiten is; dit is echter een tussentoestand in UPPAAL en in feite is de deur dan al gesloten.
2. $A[]$ (lamp.On) imply (main.Burning)
Als de UV-lamp aanstaat, moet gelden dat er een wafer wordt gebrand.
3. $A[]$ main.ReadyToLoadWafer imply (door1.Open and door2.Closed)
Voordat er een wafer wordt geladen, staat deur één open en is deur twee gesloten. “main” is de naam van het hoofdprogramma.
4. $A[]$ main.ReadyToLoadWafer imply (main.total ≤ 15)
Voordat er een nieuwe wafer wordt geladen en dus de vorige reeds klaar is, moet gelden dat het aantal verstrekken seconden niet hoger is dan 15 sinds laden van die wafer.
5. $A[]$ managecrash.Crashed or not deadlock
Het systeem is ofwel in een crash-toestand, of draait zoals normaal.

5.4 Simulatie- en verificatieresultaten

Volgens de eerste beschreven methode hebben wij het systeem gesimuleerd. Formeel zijn de resultaten hieronder beschreven. Informeel komt het erop neer dat al onze simulaties goed zijn verlopen en we dus ons systeem als correct mogen beschouwen. Helaas komt het regelmatig voor (dankzij UPPAAL’s testsysteem) dat de deuren vast- lopen en komt daardoor in een crash-toestand. Dit is echter de bedoeling wanneer een deur vastloopt, dus is een resultaat van deze aard een gewenst resultaat.

Duur	Opmerkingen	Resultaat
9s	deur weigert -> crash	succesvol
2s	deur weigert -> crash	succesvol
1s	deur weigert -> crash	succesvol
5s	deur weigert -> crash	succesvol
4s	deur weigert -> crash	succesvol

Volgens de tweede beschreven methode hebben wij het systeem geverifieerd. Bij al onze opgegeven queries gaf het systeem het resultaat „Property is satisfied”, wat dus inhoudt dat in alle gevallen het systeem voldoet aan de opgegeven eigenschappen.

5.5 Conclusie van de systeemanalyse

We kunnen dus concluderen dat ons systeem voldoet aan de gestelde eisen en eigenschappen.

Hoofdstuk 6

Testresultaten

6.1 Testresultaten en problemen

Wij hebben ons systeem natuurlijk ook getest. Hier volgen enkele tests die wij uitgevoerd hebben, samen met de resultaten hiervan. Daarnaast hebben we hier ook een paar problemen aangestipt die we onderweg tegenkwamen.

6.1.1 Problemen en mislukte tests

We beginnen met de problemen die we zijn tegengekomen en hoe we deze overwonnen hebben.

- Lampjes: Hier hebben we toch wel wat problemen mee gehad, vooral omdat de lampjes eigenlijk op 12 volt werken. De enige aansluitingen die we hadden op de processor waren +5V, +16V en ground. 16 volt bleek te veel, de lampjes werden erg warm, wat de behuizing niet ten goede kwam. Een combinatie van 5 en 16 volt (resulterend in 11 volt) leek een goede keus, maar de processor vertoonde op slag kuren. Na wat meetwerk kwamen we erachter dat de 5 volt van de processor op deze manier omhooggetrokken werd tot 7 volt - dit kon nooit goed zijn. Hierna hebben we er voor gekozen om de lampjes toch maar op 5 volt aan te sluiten.
- Sensoren (1): De sensoren hebben ons vrij veel problemen opgeleverd. Het eerste probleem was dat de sensoren aangesloten waren op een input-poort en de +5 volt. Aangezien de processor pull-up weerstanden heeft, kon het verschil tussen wel of geen actie niet waargenomen worden. Dit was gelukkig vrij makkelijk te verhelpen door de sensoren aan te sluiten op de ground in plaats van de +5 volt.
- Sensoren (2): Omdat de lampjes niet op volle kracht werkten, konden we het verschil in lichtsterkte niet direct meten. In het begin hadden we het idee om een comparator tussen de sensoren en de processor te zetten, zo hoeven we niet in software

de sensoren te ijken. We hadden in het begin de lampjes op 16 volt aangesloten, waarbij de comparator niet meer nodig was. Uiteindelijk hadden we de lampjes op 5 volt aangesloten en zijn weh dus toch maar comparators gaan halen, samen met een potmeter om de spanning in te stellen.

- **Sensoren (3):** Zelfs met comparator waren de sensorwaarden niet uitleesbaar, deze keer omdat de spanning over de sensor gelijk bleef, alleen de weerstand veranderde. We hadden dus nog twee weerstandjes nodig die de verhouding van de spanning zou veranderen. Na deze weerstandjes toegevoegd te hebben konden we eindelijk de sensoren uitlezen.
- **Noodknop:** De subroutine die de noodknop in de gaten hield gaf problemen: zodra dit bestand opgenomen werd in het hoofdprogramma en er op de noodknop gedrukt werd, reageerde TScope niet meer. Na de code goed nagekeken te hebben bleek dat de afhandeling van de interrupt niet correct was.
- **Compressormotor:** De motor van onze compressor is aangesloten op 16 volt. Dit zorgt natuurlijk voor de nodige luchtdruk, maar het heeft ook een nadeel: de motor en de aandrijfassen trillen aanzienlijk. Hierdoor bleven de wafers niet goed liggen tijdens het branden en verdwenen zij weer in de luchtsluis. Dit probleem was na enig nadenken verholpen door een verstevigende balk tegen het motorblok aan te brengen. De trilligen zwakten af en er kon normaal gewerkt worden.

6.1.2 Geslaagde onderdelen

Hier volgen de onderdelen die functioneerden zoals we gehoopt hadden, zonder al te veel aanpassingen.

- **Motoren:** De motoren van de lopende banden waren vrij makkelijk in te stellen. We hadden verwacht dat we Pulse Width Management moesten gebruiken om de juiste snelheid van de banden te bepalen en na een korte test op volle kracht bleek dit ook waar te zijn. Na een kleine verandering in de code liepen de banden perfect.
- **Deuren:** Naast een paar luchtslangen die zo nu en dan losschoten deden de deuren het goed. De kleppen reageerden netjes op het programma en de drucksensoren gaven (na de aanpassingen zoals hierboven beschreven) de goede waardes terug.
- **Verloren Wafers:** Ook deze tests verliepen goed. Zodra een wafer als verloren is gemarkeerd springt een extra LED op het processorbordje aan. Als de vijfde wafer verloren is gegaan crasht het programma.
- **Daarnaast waren grote onderdelen van de programmatekst vrijwel meteen goed geschreven.** Het UPPAAL-ontwerp heeft hier natuurlijk ook ontzettend aan bijgedragen, maar toch liep alles veel soepeler dan we verwacht hadden.

Hoofdstuk 7

Procesdocument

7.1 Tijdverdeling

Iedereen heeft een logboek bijgehouden op een online systeem tue.coenvdwel.com/lbs. Deze logboeken zijn in de bijlage te vinden.

7.1.1 Vergadering

De vergadering werd elke donderdagmiddag gehouden en duurde ongeveer een half uur.

7.2 Taakverdeling

7.2.1 Voorzitter, Notulist

Wie er voorzitter en notulist waren tijdens de vergadering veranderde per week. Iedereen is minstens één keer voorzitter en één keer notulist geweest.

id-nummer	naam	voorzitter	notulist
0618959	Etienne van Delden	week 13	week 14 en 21
0611093	Gijs Direks	week 14 en 21	week 15 en 20
0588898	Sanne Ernst	week 15 en 22	week 16 en 23
0598669	Bas Goorden	week 16 en 23	week 17 en 22
0607426	Stef Sijben	week 17	week 18
0608467	Coen van der Wel	week 18 en 20	week 14

7.2.2 Werkplan

Mechanisch

- Ontwerp (Gijs)

- Schets
- Specificatie technische details
- Bouwen ontwerp (Sanne)

Software

- Schrijven basis (Stef)
 - I/O subroutines
 - Interrupt handlers
- Aansturen mechanisch ontwerp (Coen)
 - Index-based

Misc.

- Documentatie (hele groep)
- Demonstratie met uitleg (Etienne)
- Handleiding (Bas)
- Verslag (Bas)

Deadlines

Onderdeel	Verantwoordelijke	Start datum	Interne deadline	Externe deadline
Ontwerp en operationele specificatie UPPAAL	Gijs/Etienne Gijs	26-03 26-03	18-04 16-04*	23-04 23-04
Tekeningen (foto's)	Etienne	26-03	18-04	23-04
Specificatie elektrische interface Bedradingsschema	Gijs/Sanne Gijs	26-03 26-03	23-04* 16-04*	23-04 23-04
Model (hardware)	Sanne	26-03	23-04	
Programma ontwerp (UPPAAL)	Stef	16-04	23-04*	01-05
Systeem analyse (UPPAAL)	Coen	23-04	27-04	
Implementatie Assembler code	Stef/Coen Stef	23-04 23-04	21-05* 21-05*	29-05 29-05
Aansturen mechanisch ontwerp	Coen	23-04	21-05*	29-05
Verslag en documentatie	Bas	26-03	04-05*	11-06
Demonstratie met uitleg	Etienne	30-05	n.v.t.	14-06

voor 12.00 uur inleveren

7.3 Evaluatie

7.3.1 Persoonlijke evaluatie van Etienne van Delden

Dit was voor mij de derde ogo, met als grootste verschil dat we twee dubbele P-ers en twee minoren in het groepje hadden. Dit maal kwamen onze roosters dus niet overeen. Hierdoor moesten we af en toe door werken met minder mensen als normaal. Ook verschildde de ervaringen met (OGO) groepsprojecten, waardoor we in het begin een moeilijke start hadden. Sommige dingen die voor Coen en mij logisch waren, zoals verslagen op studyweb zetten, waren voor de anderen nog niet logisch. Gelukkig was alles goed gekomen, toen we duidelijker afspraken maakten, alles goed gingen doorspreken en ikzelf aangesteld werd als quality-manager.

Ik heb mij vooral bezig gehouden met het hardware-model. Ik heb meegeholpen aan de bouw ervan en ik heb de scart-interface grotendeels aangesloten. Het aansluiten van de scart-interface om het processorbord en het model makkelijk aan te sluiten was mijn idee. Hoewel het zeker voordelen heeft, hebben we hierdoor wel veel tijd verloren. Het aansluiten van alle 24 kabels duurde langer als verwacht é er waren een aantal fouten gemaakt, Gelukkig is alles nog gelukt met de scart aansluiting.

Ook was ik verantwoordelijk voor de presentatie. Hoewel ik op moment van schrijven nog niets kan zeggen over de presentatie op de laatste ogo dag, samen met de demonstratie, had ik de eindpresentatie wel al gebruikt bij de presentatie training. Coen van der Wel en ikzelf hebben daar de eindpresentatie gegeven, waarbij die als erg goed werd beoordeeld.‘

Hoewel ik vrij weinig code heb geschreven, heb ik wel veel geleerd over de problemen die onstaan bij het maken van een embedded systeem. Voor het eerst hebben we eerst een model in UPPAAL gemaakt, van zowel het hardware als het software model, en pas daarna gewerkt aan de software zelf. We hebben daarbij ondervonden dat dit enorm helpt, omdat je dan een richtlijn hebt voor hoe je code moet worden.

Qua verantwoordelijkheid wou ik me meer op de achtergrond houden, uit vorige ogo's heb ik gemerkt dat ik soms me teveel met anderen bemoei. Dit was toch niet helemaal gelukt. Door de moeilijke start werd ik uiteindelijk aangesteld als quality-manager, waarbij ik de verantwoordelijkheid draag voor de ingeleverde documenten. Mijn taak resulteerde uiteindelijk tot externe agenda voor de anderen. Ik hoefde zelf niet opdrachten te geven, maar mij werd gevraagd wat er moest gebeuren. Dit maakte het voor mij prettiger, omdat ik niet opdrachten hoef uit te geven , maar ook voor de anderen die een vraagbaak hadden en iemand die wist wat er allemaal moest gebeuren.

7.3.2 Persoonlijke evaluatie van Gijs Direks

Dit was mijn eerste OGO, mijns inziens ook een zeer geslaagde. Na een paar opstartproblemen, vooral te wijten aan het feit dat maar twee personen al eerder OGO gehad hebben, liep alles lekker door. We hebben met zes man hard en goed gewerkt aan dit project.

Ik heb ook veel geleerd van deze OGO. Naast het omgaan met andere mensen was er natuurlijk ook het technische aspect. Fischertechnik is natuurlijk geen ontzettend hoogstaand materiaal en brengt menige beperking met zich mee, maar dit levert ook uitdagingen die overwonnen moeten worden. Enkele van deze problemen vergden 'nieuwe' oplossingen, die een andere kijk met zich meebrachten.

Ook de practicumprocessor heeft enkele kopzorgen gebaard: bij het ontwerp van de elektrische interface had ik er bijvoorbeeld geen rekening mee gehouden dat er pull-up weerstanden aan de poorten zitten - dit had als effect dat ik de noodknop en de sensoren op een ingangspoort en +5V aangesloten had. De processor las dan ook geen bruikbare waarden uit. Na een opmerking van Stef en wat solderen was dit toch weer snel verholpen.

Al met al vond ik dit een zeer geslaagde OGO, het was een interessant probleem en ik heb met leuke mensen samengewerkt.

7.3.3 Persoonlijke evaluatie van Sanne Ernst

Bij wiskunde heb ik modelleervakken gevolgd, daarentegen was OGO voor mij nieuw. In plaats van werken in groepjes van twee moesten we nu met zes man werken. Dit maakt het werken in een groep een stuk gezelliger en je bent ook niet gebonden aan één partner. Dit was voor mij een nieuwe en ook prettige ervaring.

Wat voor mij totaal nieuw was, waren de wekelijkse vergaderingen. Hierbij heb ik het belang van vergaderen geleerd, namelijk het periodieke spreken van de opdrachtgever. Echter vond ik de vergaderingen te formeel en over het algemeen te lang duren.

Voor OGO 1.3 had ik nog niet zoveel ervaring met het ontwerpen en maken van programmatuur. Ik had wel al ervaring met het programmeren van java-applicaties bij modelleren. Nu ben ik voor het eerst begonnen met het maken van een model alvorens de applicatie zelf te maken. Dit werkte zeer prettig, omdat ik nu een richtlijn had voor het schrijven van code. Ook heb ik meer ervaring opgedaan met het schrijven van assembler code.

Verder heb ik meegeholpen met het bouwen van het hardware-model. Dit vond ik heel erg leuk om te doen, omdat dit een heel gepuzzel was. De vorm van de stukjes maakt

het soms lastig om te bepalen wat het doel ervan is. Het werken met Fischertechnik deed mij denken aan het spelen met Lego. Wat ik onder andere ontworpen heb zijn de lopende banden.

Hoewel we een moeilijke start hadden, werd het uiteindelijk een heel geslaagde OGO.

7.3.4 Persoonlijke evaluatie van Bas Goorden

Dit was ook voor mij de eerste OGO. De major die ik volg is Technische Wiskunde en dit was één van de vakken die ik in mijn minor gekozen heb.

Bij wiskunde hebben we intussen toch al een behoorlijk aantal Modelleren-projecten gedaan. Hierdoor was ik bang dat ik aan dit vak, op het gebied van samenwerken, weinig zou hebben. Het feit dat de groepen bij dit vak uit zes in plaats van uit twee personen bestaan had echter zijn gevolgen. Bij Modelleren weet je namelijk altijd precies wat er gebeurt en als je zelf niks doet gebeurt er ook niks. Het was daarom zaak om in de gaten te houden waar de rest van de groep mee bezig was en om daar vragen over te stellen als zij iets gedaan hadden wat ik nodig had. Daarnaast ben je, als je deel uitmaakt van een groep van twee personen, zelf betrokken bij en verantwoordelijk voor alle belangrijke beslissingen. Bij OGO was dit niet zo, onze groep bestond namelijk voornamelijk uit mensen die geen enkel probleem hebben met het nemen van belangrijke beslissingen. Hier heb ik mij aan aangepast, door mezelf niet bij alle belangrijke beslissingen te betrekken. Dit viel mij minder zwaar dan ik verwacht had. Ik heb bij dit OGO-project dus eens een andere manier van samenwerken meegebracht en dat heeft zeker een bijdrage geleverd aan mijn vaardigheden op dat gebied.

Wat natuurlijk ook belangrijk is, is dat ik veel geleerd heb wat de informatica zelf betreft. Ik heb mij eigenlijk een beetje met alle belangrijke onderdelen van dit project beziggehouden. Eerst met het bouwen van het hardware-model, vervolgens met het maken van een programmaontwerp in UPPAAL, daarna met het schrijven van een aantal subroutines en tot slot met het verslag. Vooral van het maken van het programmaontwerp en van het schrijven van code heb ik veel geleerd. Met de presentatie heb ik mij helemaal niet beziggehouden, maar bij Modelleren heb ik al een behoorlijk aantal presentaties gehouden, dus ik denk niet dat dat erg is.

Al met al heb ik er dus veel van geleerd. Het is niet meer dan de kers op de taart, maar toch het vermelden waard, dat ik het onderwerp van het project interessant vond en dat de samenwerking binnen de groep goed was. Al met al was dit dus een geslaagde OGO.

7.3.5 Persoonlijke evaluatie van Stef Sijben

Deze OGO begon enigszins moeizaam, waarschijnlijk mede doordat in onze groep slechts twee personen waren die al eerder een OGO hadden gedaan. Na een aantal weken ging het echter beter en vanaf dat moment zijn er weinig grote problemen meer geweest. De samenwerking met de rest van de groep is in het algemeen goed verlopen.

Het gebruik van Fischertechniek bracht de nodige beperkingen met zich mee, maar het was een leuke uitdaging om hier omheen te werken, mede omdat we graag wilden proberen de hele waferstepper op één bordje te bouwen.

Daarnaast hebben we geleerd dat het bij een eventueel volgend project handig is om de elektrische aansluitingen wat beter te organiseren. Nu waren er namelijk nogal wat draden fout aan elkaar gesoldeerd, waardoor het model niet goed werkte.

Ondanks dat ik al de nodige ervaring had met embedded systemen, heb ik op dit gebied toch ook nog de nodige dingen geleerd omdat deze processor op sommige gebieden nogal verschillend is van de processoren waar ik tot nu toe mee gewerkt heb.

Omdat ik intensief gewerkt heb aan het ontwerp van de UPPAAL-modellen heb ik ook op het gebied van specificeren veel nieuwe dingen geleerd, ik was hiermee namelijk nog niet eerder in aanraking gekomen.

Met de presentatie heb ik me niet beziggehouden, maar ik heb wel al een presentatie gedaan voor modelleren, dat ik in blok D gevuld heb in het kader van de dubbele proefdeuse die ik doe. Ik denk dan ook dat dat niet echt een probleem zal zijn.

Wel heb ik het nodige geleerd van verslaglegging, ik heb namelijk het verslag van het programmaontwerp grotendeels geschreven. De verslaglegging daarvan is toch wel anders dan andere verslagen die ik tot nu toe geschreven heb.

Deze OGO is in mijn ogen dus goed verlopen en ik heb er veel van geleerd.//

7.3.6 Persoonlijke evaluatie van Coen van der Wel

Persoonlijk vond ik dit een geslaagde OGO. Vanwege een kleine miscommunicatie in het begin was er even een probleem met het samenstellen van de groepen, maar uiteindelijk werd de groep samengesteld uit 3 van mijn vrienden en 2 minor-studenten, waarmee ik inmiddels ook goed overweg kan. Ik denk dat dat zeker heeft bijgedragen aan het resultaat, want een goede relatie met collega's is ook echt belangrijk.

Verder heb ik veel geleerd over pneumatiek en elektrische interfaces, ik heb zeker mijn deel

bijgedragen aan het uitwerken van deze twee dingen. Ook heb ik mijn LEGO™ ervaring uit kunnen breiden door het bouwen van het model met Fischertechnik™. De beperktheden van deze techniek leidden tot een meer gecompliceerd model, maar die uitdaging is juist leuk. Het programmeren van het model en het kennismaken met een embedded systeem is uiteraard ook een belangrijk aspect geweest en heb ik met veel plezier gedaan.

De presentatie heeft Etienne grotendeels verzorgd, maar hierbij heb ik toch zeker geen onbelangrijke rol gespeeld. De presentatietraining was voor mij niet echt nuttig, ik heb er namelijk niet erg veel van geleerd. Het rapporteren en verslagleggen van (tussen)resultaten was zoals bij de vorige OGO's een beetje een blok aan het been, maar gaat naar mijn mening wel steeds beter. Ik ontken dan ook niet dat deze verslagen nuttig zijn, maar het spelen met het model is natuurlijk veel leuker ;-)

Qua verantwoordelijkheden heb ik mezelf een beetje teruggetrokken. Bij vorige OGO's is mijn ervaring dat ik me met alles bemoei en dat uiteindelijk alles op mij neerkomt; bij deze OGO heb ik me meer afzijdig gehouden, met de keerzijde dat ik me niet erg nuttig heb gevoeld. Er moet vast een tussenweg zijn... dat als verbeterpunt voor mezelf voor een volgende OGO.

Maar al met al vind ik het een geslaagde OGO!

Hoofdstuk 8

Conclusie

8.1 Conclusie

Na drie maanden hard werken en een moeilijke start, is onze Prime Waferstepper af. We hebben allemaal hard samengewerkt om een goed werkend en compact model van een waferstepper te maken dat, naar onze mening, voldoet aan de gegeven eisen.

Het gebruik van de SVN server heeft onze samenwerking goed bevorderd, iedereen had de nieuwste versie van een verslag of code. Niet iedereen had hier ervaring mee, maar al heel snel werd dit systeem goed gebruikt en geprezen.

Om de software op de processor te zetten moesten wij gebruik maken van tScope. Dit programma is naar onze mening zeer onstabiel, op het moment dat het crashed moet je je computer opnieuw opstarten. Ook het feit dat het crasht na interrupt handling, dat bij de noodknop essentieel is, heeft voor veel frustratie gezorgd.

Ondanks alle tegenslagen, hebben we allemaal veel geleerd over het maken van een embedded system. Ook hebben wij kennisgemaakt met UPPAAL en hebben wij allen het nut van het maken en controleren van een model voorgaande aan het programmeren zelf ondervonden. Dit brengt structuur in de manier van werken.

Bijlage A

Assembler code

A.1 main

TITLE 'Main Program' SBttl 'OG01.3 group 7, 2006-2007' ; ; This is the main module of the waferstepper controller. It implements the following subroutines: ; Crash: A 'subroutine' that will loop to itself forever. ; Labels used are (mostly) equivalent to those in the UPPAAL model. Bootup:

```
ORG      00h
AJMP    main           ;Skip reserved memory

; skip first bytes + includes
INCL 'int_vect.asm'

ORG 0080h
INCL 'mem_subs.asm'
INCL 'wt_subs.asm'
INCL 'iol_subs.asm'
INCL 'ioh_subs.asm'
INCL 'emergenc.asm'
INCL 'init.asm'

main:  ;Start of program
MOV     SP, #8
ACALL   init           ;Initialise all components to their desired states.
```

ProgramPaused:

```
MOV      DPTR, #OFF00h          ;The adress of the dip-switches, of which switch 0
MOVX    A, @DPTR
JNB     Acc.0, ProgramPaused

;Ready button has beed enabled.

ReadyToLoadWafer:
ACALL   LoadWafer           ;Loads a wafer. C = 1: succes, C = 0: Failure
JNC     ProgramPaused

;We want to move the wafer forward for 1 second
MOV      A, #20d
;TODO: check the timing of this!
ACALL   MoveForwardT

ACALL   CloseDoor1           ;Check whether the door actually closed
JNC     DoorJammed

ACALL   OpenDoor2            ;Check whether the door actually opened
JNC     DoorJammed

ACALL   MoveForwardS2         ;Moves the wafer to the sensor
JNC     WaferLost            ;Check whether wafer has arrived

MOV      A, #10d
;TODO: check the timing of this!
ACALL   MoveForwardT         ;move wafer under UV lamp
ACALL   Burn                  ;burn the wafer
JNC     ReadyToLoadWafer

ACALL   MoveBackwardS2        ;move wafer back to sensor
JNC     WaferLost

MOV      A, #20d
;TODO: check the timing of this!
ACALL   MoveBackwardT

ACALL   CloseDoor2            ;Closes door 2
JNC     DoorJammed           ;Check whether door has closed

ACALL   OpenDoor1             ;Opens door 1
```

```
JNC     DoorJammed           ;Check whether door has opened

ACALL   MoveBackwardS1        ;Move wafer to sensor 1
JNC     WaferLost            ;Check whether wafer has arrived

MOV     A, #60d
;TODO: check the timing of this!
ACALL   MoveBackwardT        ;Move burnt wafer off the belt

AJMP   ReadyToLoadWafer

;This state is reached when a door is stuck. It moves to Crash.
DoorJammed:
AJMP   CrashMain

CrashMain:
AJMP   CrashMain             ;Program Crash, restart required.

WaferLost:
ACALL   LostWafer            ;This wafer is lost, prepare for handling the next
AJMP   ReadyToLoadWafer      ;And then start handling the next one if possible

END Bootup
```

Bijlage B

Logboeken

Op de volgende pagina's staan de logboeken van alle groepsleden (in alfabetische volgorde).

Logboek

Gebruiker: Bas Goorden

Van	Tot	Duur	Omschrijving
Maart 2007			
Week 13			
27-03-2007 13:30	27-03-2007 16:15	2 uur en 45 minuten	Vergadering + onderdelen tellen
29-03-2007 13:30	29-03-2007 17:15	3 uur en 45 minuten	Vergadering + werkplan maken + misc.
			<i>Sub-totaal: 6 uur en 30 minuten</i>
April 2007			
Week 14			
03-04-2007 13:45	03-04-2007 17:45	4 uur	bouwen
05-04-2007 13:30	05-04-2007 17:15	3 uur en 45 minuten	college UPPAAL + bouwen
Week 15			
10-04-2007 14:45	10-04-2007 17:30	2 uur en 45 minuten	bouwen
12-04-2007 13:30	12-04-2007 17:15	3 uur en 45 minuten	vergadering + werkplan aanpassen + bouwen
Week 16			
17-04-2007 15:45	17-04-2007 17:15	1 uur en 30 minuten	bouwen
19-04-2007 13:30	19-04-2007 17:15	3 uur en 45 minuten	vergadering + UPPAAL
Week 17			
24-04-2007 13:30	24-04-2007 17:15	3 uur en 45 minuten	Systeemontwerp
26-04-2007 13:30	26-04-2007 17:15	3 uur en 45 minuten	vergadering + systeemontwerp + IO subroutines
			<i>Sub-totaal: 27 uur</i>
Mei 2007			
Week 18			
01-05-2007 13:45	01-05-2007 17:15	3 uur en 30 minuten	hogere laag van IO subroutines geschreven
03-05-2007 13:30	03-05-2007 17:15	3 uur en 45 minuten	vergadering + aanpassen systeemontwerp
Week 20			
15-05-2007 13:30	15-05-2007 17:15	3 uur en 45 minuten	vergadering + programmaontwerp
Week 21			
22-05-2007 13:30	22-05-2007 16:30	3 uur	systeemanalyse, assembly
24-05-2007 15:15	24-05-2007 17:15	2 uur	vergadering + assembly
Week 22			
29-05-2007 13:30	29-05-2007 17:15	3 uur en 45 minuten	testen subroutines + verslag
31-05-2007 15:25	31-05-2007 17:15	1 uur en 50 minuten	vergadering + verslag
			<i>Sub-totaal: 21 uur en 35 minuten</i>
Juni 2007			
Week 23			
04-06-2007 08:45	04-06-2007 11:15	2 uur en 30 minuten	verslag
05-06-2007 13:30	05-06-2007 17:15	3 uur en 45 minuten	verslag
07-06-2007 15:30	07-06-2007 17:15	1 uur en 45 minuten	vergadering + verslag
Week 24			
12-06-2007 14:30	12-06-2007 17:15	2 uur en 45 minuten	verslag
			<i>sub-totaal: 10 uur en 45 minuten.</i>

Totaal aantal uren: 65 uur en 50 minuten. [< terug](#)