Software Architectuur Documentatie -

4-op-1-rij robot

Version 3
Date: 13-4-2022

ALTEN
Pascal Faatz

Vertrouwelijk
Opzet

Versie geschiedenis

Versie	Datum	Status	Schrijver	Opmerking
1	31-3-2022	Opzet	Pascal Faatz	Eerste opzet tot hoofdstuk 8
2	13-4-2022	Opzet	Pascal Faatz	Verwerken feedback Jeedella, Aniel en Berend
3	3-5-2022	Afronding	Pascal Faatz	Hoofdstuk 8 en verder

Acroniemen en afkortingen

Term	Uitleg
BSP	Board Support Package

Gerefereerde documenten

Id	Referentie	Titel	Datum	Schrijver
D1	SRD_Pascal_Faatz_V3.docx	SRD	16-3-2022	Pascal Faatz

Inhoudsopgave

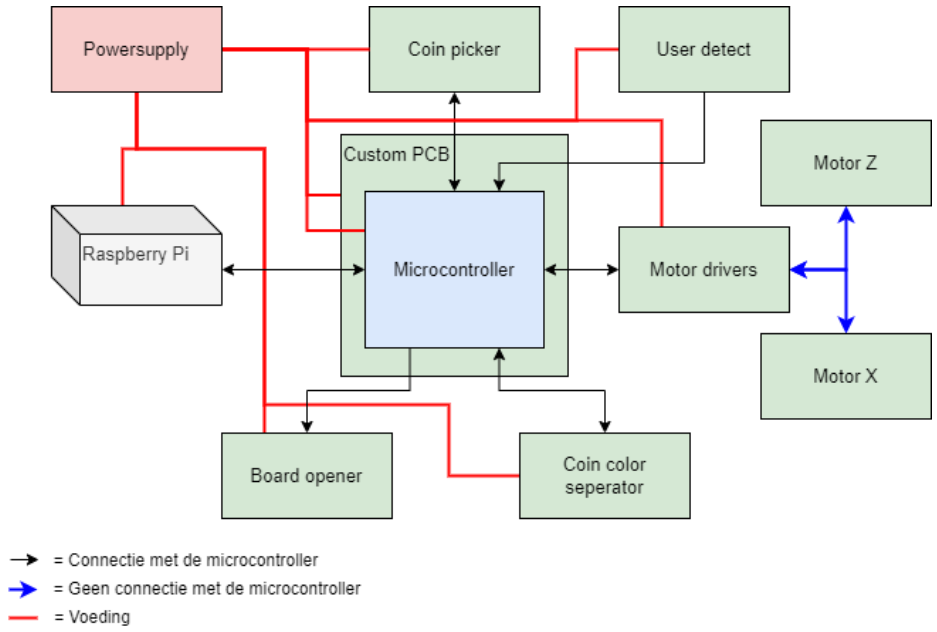
Lijst met figuren

Lijst met tabellen

1 Introductie en goals

ALTEN heeft in eigen beheer een robot ontwikkeld die middels een algoritme 4-op-1-rij kan spelen tegen een menselijke tegenstander. Met behulp van industriële componenten wordt de 4-op-1-rij robot gebouwd om de kennis van verschillende systemen te demonstreren. De robot zal onder andere op beurzen en open dagen gebruikt worden als demonstratie unit, waarbij deze beschikbaar is voor voorbijgangers om een ronde te spelen. Aan de voorzijde kan de speler dan zijn/haar zet plaatsen op het bord, waarna de 4-op-1-rij robot een zet zal bedenken en uitvoeren. Hiervoor houdt het systeem bij welke zetten door zijn tegenstander gespeeld zijn. Wanneer de zet door de robot bepaald is zal de combinatie van het X-Z platform met roterende vacuümgripper een steen op pakken en op de gekozen plek in het spel invoeren. Zodra het spel afgelopen of gereset is, zal de 4-op-1-rij robot alle fiches verzamelen en op kleur sorteren om zo klaar te zijn voor de volgende ronde.

Momenteel draait de 4-op-1-rij robot op een Raspberry Pi + STM32 microcontroller (Figuur 1). Op de Raspberry Pi draait het algoritme om de volgende zet van het systeem te bepalen en op de STM32 controller draait het besturingssysteem. Dit project is binnen ALTEN in eerste instantie gebruikt om consultants die op een tussen periode niet op een project zitten toch een uitdaging te geven. Hierdoor is er, over een langere periode, door meerdere consultants aan gewerkt. Dit heeft geresulteerd in een zeer onoverzichtelijke en onduidelijke architectuur van de software en hardware. Dit geld dus ook voor het besturingssysteem van de 4-op-1-rij robot.



Figuur 1 Blok diagram 4-op-1-rij

1.1 Requirements overzicht

Door de onduidelijke architectuur is het zeer lastig om het besturingssysteem uit te breiden of een upgrade toe te passen. Omdat de 4-op-1-rij robot ook op beurzen staat en aantoont wat ALTEN in huis heeft is het van noodzaak dat de robot goed te onderhouden is en makkelijk een upgrade kan ondergaan. Hiervoor is een complete re-design van het besturingssysteem nodig om de onoverzichtelijke en onduidelijke architectuur op te lossen en upgrades mogelijk te maken.

Een van de hoofd requirements is dan ook om een gestructureerde architectuur en modulaire implementatie in te voeren. Verder is het van belang state- en flowdiagrammen op te stellen om de werking van het systeem visueel weer te geven. Ook moet er een BSP (Board Support Package) gemaakt worden.

Voor de volledige lijst van requirements refereer aan het SRD (D1).

1.2 Quality goals

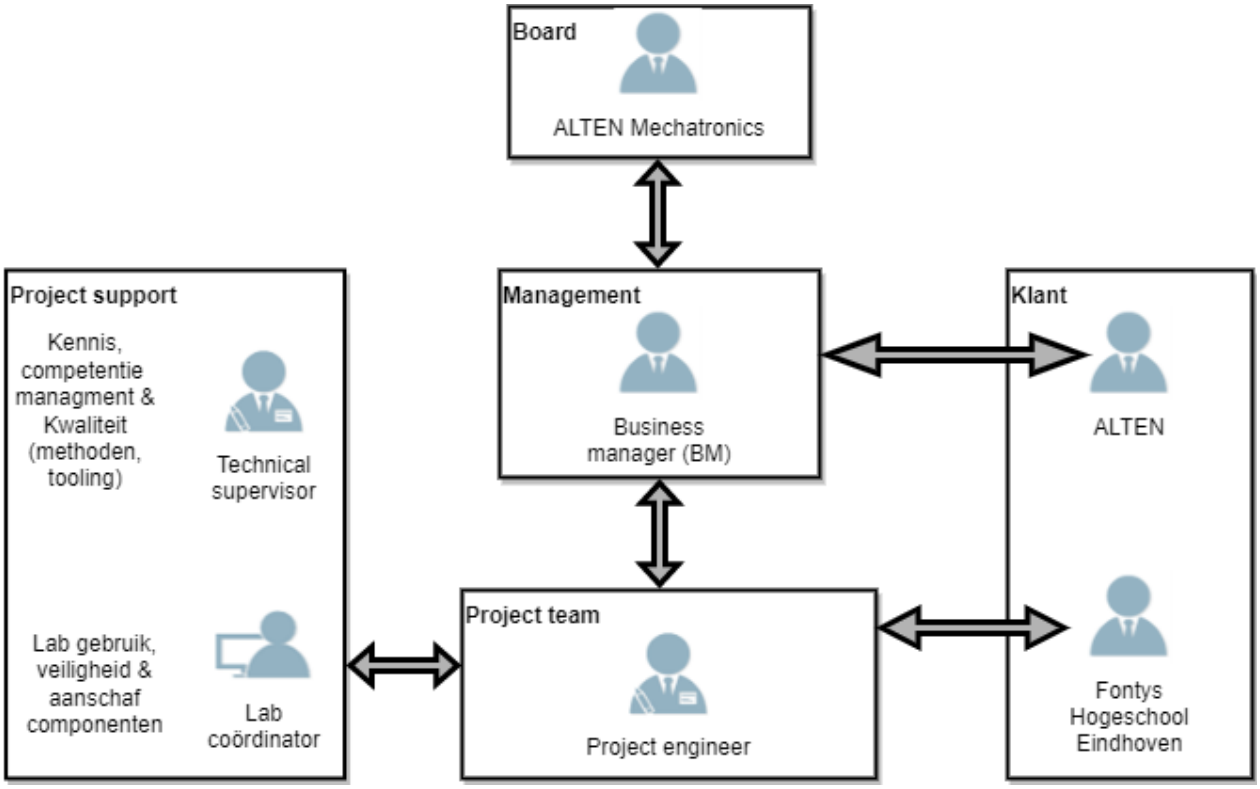
De quality goals zijn essentieel om aan te geven aan welke kwaliteitseisen het systeem moet voldoen. Deze goals worden in een later stadium ook getest en gecheckt of het systeem hier aan voldoet.

Tabel 1 Quality goals

Prioriteit	Quality goal	Concreet scenario
1	Makkelijk te begrijpen (Gestructureerde architectuur)	Mensen die aan de 4-op-1-rij robot gaan werken moet met het lezen van de architectuur in een keer snappen hoe de hardware en software samen hangt.
2	Onderhoud en upgrades (Modulaire opbouwen van software blokken)	Mensen die een upgrade of aanpassing gaan doen aan de 4-op1-rij moeten software blokken kunnen aanpassen of vervangen zonder dat dit andere delen van het systeem beïnvloed.
3	Robuust	De 4-op-1-rij moet betrouwbaar zijn en werken onder alle omstandigheden wanneer het systeem draait.

1.3 Stakeholders

De stakeholders zijn een belangrijk onderdeel van het project. Zij bepalen de requirements en stellen eisen aan het eind product. In Figuur 2 is de verdeling van stakeholders voor dit project weergegeven.



Figuur 2 Project organisatie

Tabel 2 Stakeholders

Naam	Rol	Verantwoordelijkheid
Pascal Faatz	Project engineer	Het project uitvoeren voor de betreffende klanten en board/management.
Aniel Shri	Technical supervisor	Binnen het project technische steun geven aan het project team. Dit houdt het uitdagen van het team om tot een goede oplossing en uitwerking te komen.
Gijs Haans	Business manager	Binnen het project persoonlijke ontwikkeling ondersteunen van het project team. Verder houdt de business manager.
Jeedella Jeedella	Fontys Hogeschool Eindhoven	Het project vanuit school ondersteunen op technisch en persoonlijk vlak. Aan het eind een beoordeling geven over het project.
Jeroen Wilbers	Lab coördinator	Het veilig gebruik maken van het lab en het aanspreekpunt voor bestelling die gedaan moeten worden.
Aniel Shri en Gijs Haans (als klant)	ALTEN	Als eindklant de requirements beoordelen en het eindproduct gebruiken.
Chris Kalis	ALTEN Mechatronics unit	De leiding over alle project partijen binnen ALTEN. Verder is de afdeling eindverantwoordelijke voor het project.

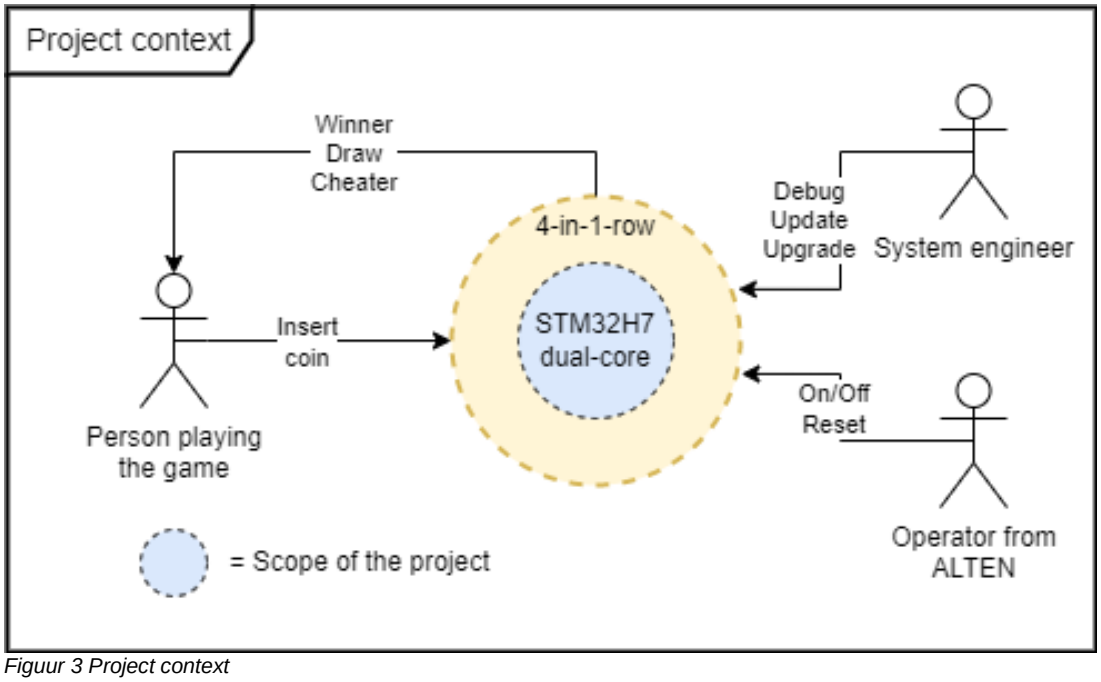
2 Architectuur beperkingen

Tabel 3 Architectuur beperkingen

Architectuur beperking	Omschrijving
Er wordt gebruik gemaakt van een STM32H755 dual-core nucleo-144 board.	De keuze van de microcontroller staat vast. Met deze microcontroller moet dus gewerkt worden.
De hardware van de 4-op-1-rij staat vast.	De hardware die er nu is wordt gebruikt om de software voor te schrijven.
Het spel verloop van de 4-op-1-rij staat vast.	Omdat de 4-op-1-rij al werkend is geweest is de globale werken van te voren bekend.
Het project moet binnen 100 werkdagen worden uitgevoerd.	Het project vind plaats binnen een school semester en moet dus binnen deze tijd worden uitgevoerd.
De Raspberry Pi geeft de volgende zet door.	De Raspberrry Pi is de processor die de volgende zet door geeft aan de microcontroller.

3 Project scope en context

Figuur 3 geeft een globale weergave van de personen die met het systeem werken en wat zei als input hebben op het systeem of wat ze van het systeem terug krijgen. Verder toont het de scope van het project. De scope van het project is de STM32H7 dual-core en de architectuur daarvan.



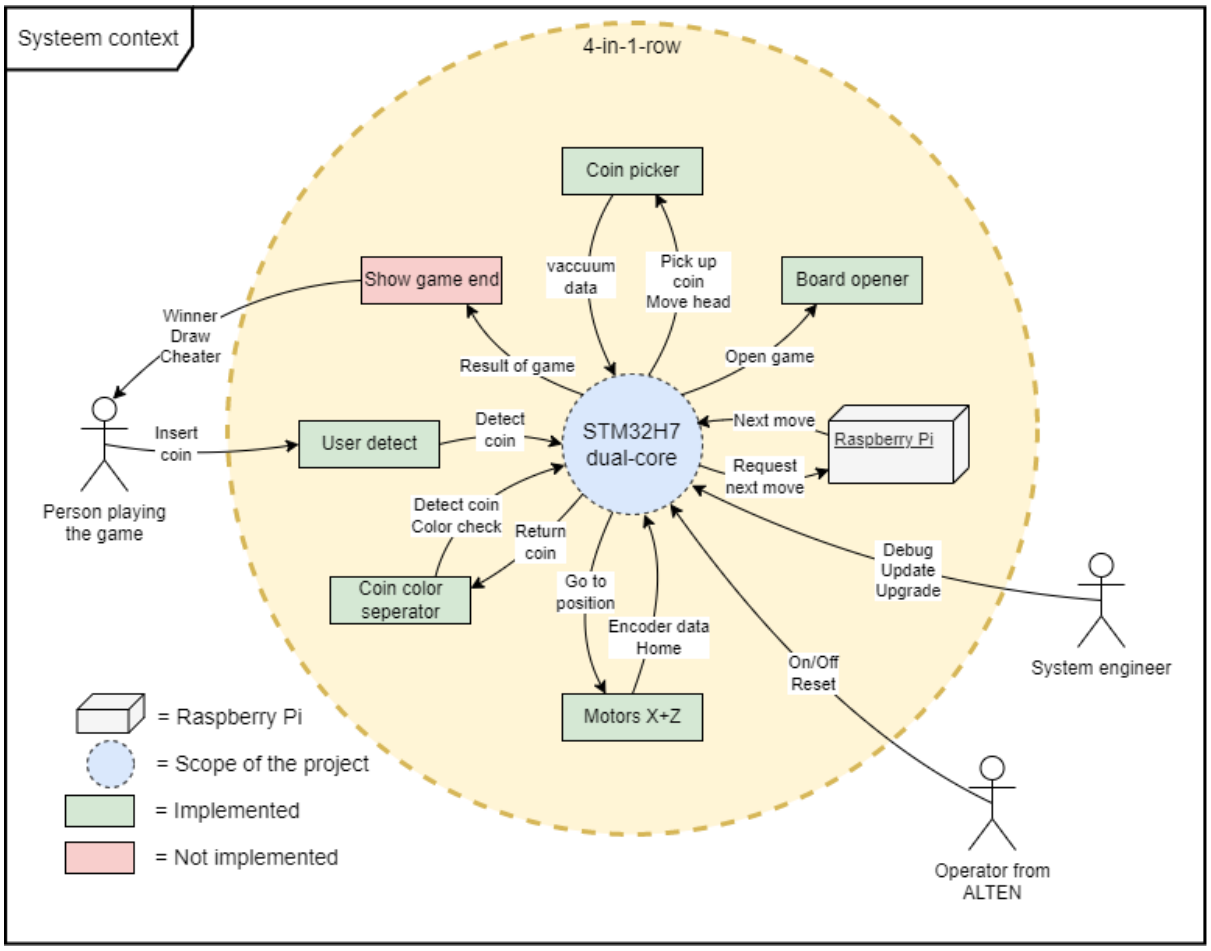
Figuur 3 Project context

Tabel 4 Beschrijving Project context

Gebruikers	Beschrijving
Persoon die 4-op-1-rij speelt	De persoon die 4-op-1-rij speelt moet als het zijn beurt is goed nadenken kolom hij zijn fiche wil doen om de robot te verslaan. Als een zet bekent is wordt het fiche in het 4-op-1-rij bord gestopt. Verder kan de persoon aan het einde van het spel zien wat het resultaat is.
Systeem engineer	De systeem engineer kan de 4-op-1-rij debuggen als er een fout optreedt. Verder kan de engineer een update/upgrade implementeren.
Operator vanuit ALTEN die de 4-op-1-rij bedient	Deze persoon bediend de 4-op-1-rij op beurzen of anderen gelegenheden. Deze persoon kan de 4-op-1-rij aan-/uitzetten of hem resetten.

3.1 Systeem context

Figuur 3 geeft een globaal overzicht van de betrokken personen bij het systeem. Figuur 4 zoomt hier verder op in en geeft een visuele representatie van de subonderdelen van de 4-op-1-rij. Ook laat het de relatie zien met de STM32H7 dual-core door middel van de inputs en outputs. Dit is van belang zodat alle stakeholders een idee krijgen wat er in grote lijnen binnen het systeem gebeurt.



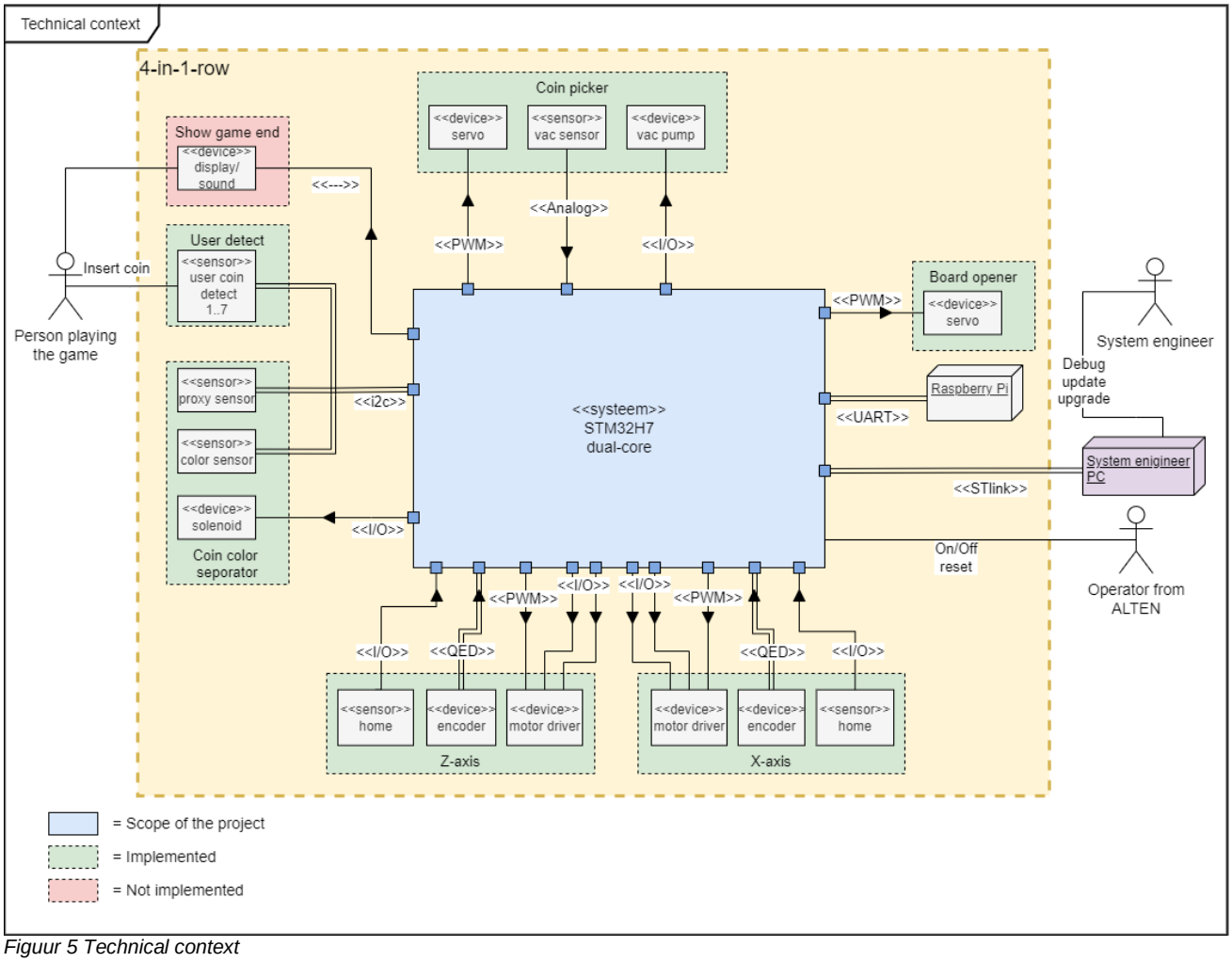
Figuur 4 Systeem context

Tabel 5 Beschrijving Systeem context

Hardware blok	Functie
Raspberry Pi	Op de Raspberry Pi draait een algoritme dat de volgende stappen van de robot bepaalt.
Motors X+Z	De motoren zorgen ervoor dat de robot zijn vaccuum gripper over een X- en Z-as van het 4-op-1-rij bord kan bewegen.
Coin color seporator	Als het spel is afgelopen zorgt de coin color seporator ervoor dat elk fiche op kleur herkent wordt. Heeft het fiche de kleur van de speler dan wordt het fiche naar de bak van de speler geschoten. Heeft het fiche de kleur van de robot dan wordt het fiche in de eigen stapel gedaan.
User detect	Als de speler een fiche in het spel doet wordt dit gedetecteerd. Elke kolom van het 4-op-1-rij bord bevat een detectie punt.
Coin picker	De coin picker is de arm van de robot. Deze wordt verplaats door de motoren maar kan zelf ook bewegen om een fiche in het 4-op-1-rij bord te doen. Een fiche oppakken of los laten gebeurt doormiddel van een vaccuum gripper.
Board opener	De board opener zorgt ervoor dat alle fiches uit het 4-op-1-rij bord vallen als het spel is afgelopen.

3.2 Technische context

Figuur 5 zet Figuur 4 om in een beschrijving om in een technische representatie van de verhouding tussen de componenten en de scope. De verbindingen wordt niet globaal beschreven maar weergegeven door welke soort protocol of in-/output signaal ze verbonden zijn. Dit is van belangen voor de stakeholders die aan de slag gaan met de hardware of architectuur.



Figuur 5 Technical context

Tabel 6 Beschrijving Technical context

Hardware blok	Sub-blokken	Omschrijving	Input naar STM32H7	Aantal lijnen	Output van STM32H7	Aantal lijnen
Raspberry Pi	-	De Raspberry Pi communiceert met de STM32H7 doormiddel van een UART verbinding.	UART Rx	1	UART Tx	1
Motors/ driver X	Motor met encoder	De motor wordt aangestuurd door een PWM signaal, een lijn voor de richting en een lijn voor het ready signaal. Om de positie van de motor te bepalen is een encoder gebruikt. De encoder communiceert met een A en B lijn.	QED (A lijn en B lijn)	2	1 PWM, 2 I/O	3

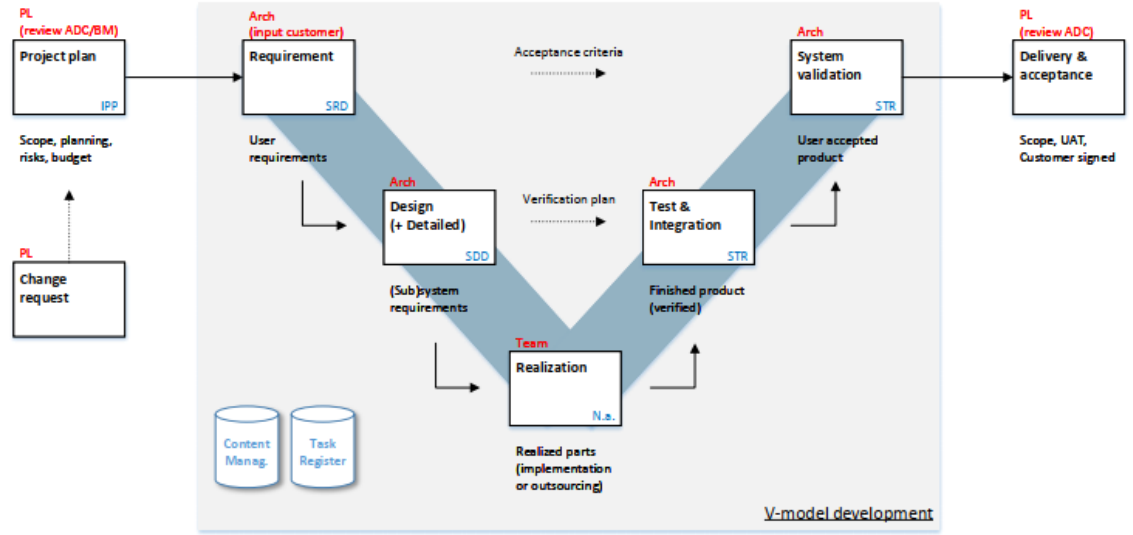
	Home/ End stop	De home/end stop geeft aan als de motor op het home punt gekomen is.	I/O	1	-	-
Motor/ driver Z	Motor met encoder	Zie Motor X.	QED (A lijn en B lijn)	2	1 PWM, 2 I/O	3
	Home/ End stop	Zie Motor X.	I/O	1	-	-
Coin color seporator	Proxy sensor	De proxy sensor checkt of er een fiche aanwezig is in het verdeel systeem.	i2c (SDA, SCL)	2	-	-
	Color sensor	De color sensor checkt welke kleur het fiche heeft die in het verdeel systeem ligt.	i2c (SDA, SCL)	2	-	-
	Solenoid	Als een fiche de kleur van de speler heeft schiet de solenoid d.m.v. een flipper het fiche naar de bak van de speler.	-	-	I/O	1
User detect	-	De user detect detecteert als er door de speler een fiche in het systeem geworpen wordt.	i2c (SDA, SCL)	2	-	-
Coin picker	Servo	De servo zorgt ervoor dat de vaccuum gripper bewogen kan worden.	-	-	PWM	1
	Vaccuum pump	De vaccuum gripper kan een fiche vast zuigen en loslaten.	-	-	I/O	1
	Vac sensor		Analoog	1	-	-
Board opener	-	De game opener zorgt doormiddel van een servo dat alle fiches uit het 4-op-1-rij bord vallen als het spel is afgelopen.	-	-	PWM	1

4 Oplossingsstrategie

Tabel 7 Oplossingstrategie

Goal/requirement	Aanpak	Link naar hoofdstuk
Gestructureerde architectuur	Om een gestructureerde architectuur voor de software van de 4-op-1-rij op te zetten wordt gebruik gemaakt van dit document. Aan de hand van de hoofdstukken en diagrammen in dit document wordt duidelijk hoe de 4-op-1-rij zich gedraagt, de software blokken samenhangen en de software met de hardware communiceert. Verder wordt er een top-down approach gebruikt door te beginnen bij de requirements. Vanaf de requirements wordt het systeem steeds verder ontleed.	
Modulaire opbouwen van software blokken	De software wordt modulaire opgebouwd om er voor te zorgen dat een upgrade of aanpassing makkelijk te faciliteren is. Eerst wordt uitgewerkt welke software blokken er zijn. Daarna hoe deze blokken met elkaar communiceren.	(Hoofdstuk 5,6,8)
Robuust	De 4-op-1-rij moet onder normale omstandigheden werken zoals van te voren vast gesteld. Dit kan worden gegarandeerd door alle software blokken individueel en samen uitgebreid te testen.	

Binnen het hele project wordt gebruik gemaakt van het V-model. Het V-model is een project methode die structuur aanbrengt in de doorloop van het project. Voor elke specificatie- of ontwerpfase aan de linkerzijde, is een corresponderende integratiefase aan de rechterzijde. Elke fase aan de rechterzijde van het project kan geverifieerd en gevalideerd worden door de fase aan de linkerzijde (Figuur 6).



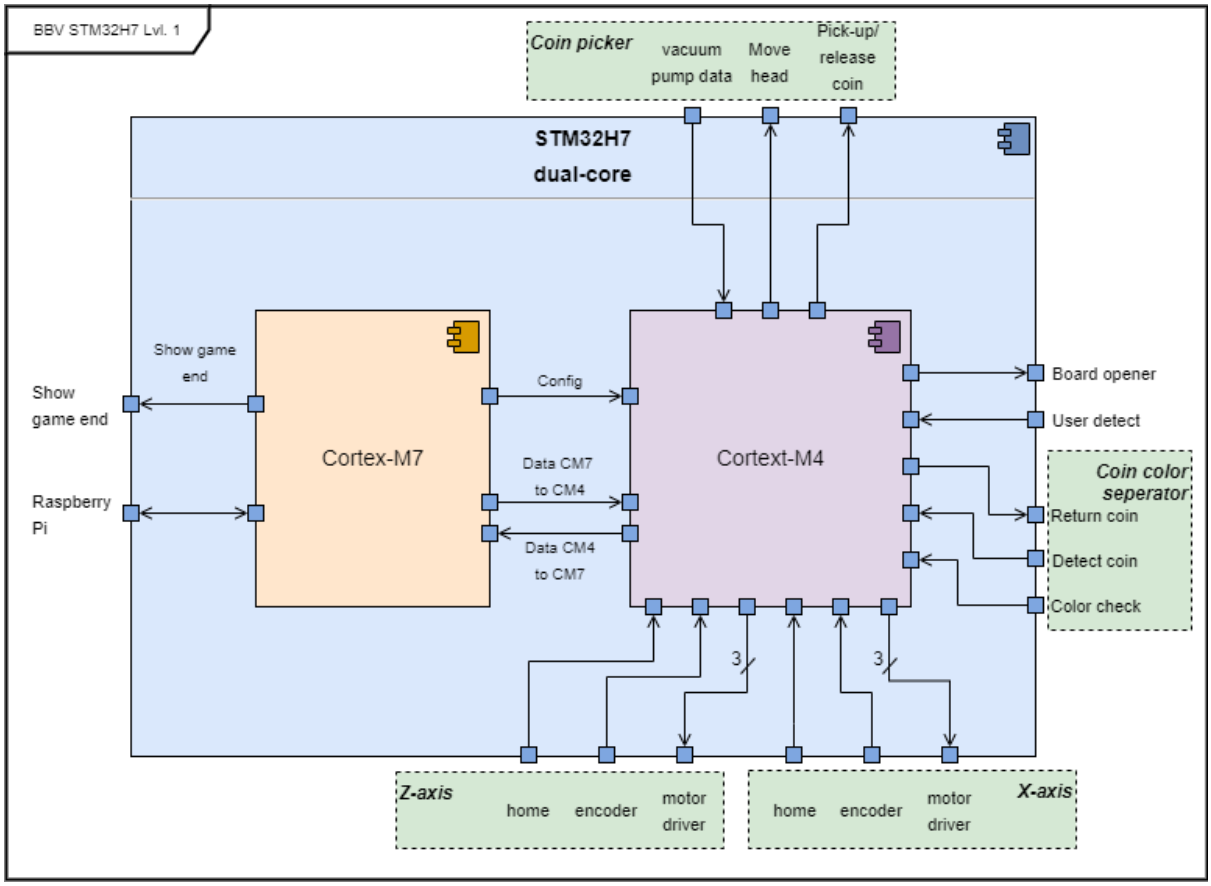
Figuur 6 V-model

5 Building block view

Dit hoofdstuk gaat dieper in op de in hoofdstuk 3 gepresenteerde diagrammen. Het idee van een building block view is om op het systeem stap voor stap verder in te zoomen. Dit wordt gedaan door middel van levels. Level 0 is voor dit project is Figuur 5 waarbij de STM32H7 dual-core een black box is. In level 1 wordt deze STM32H7 dual-core een white box die weer bestaat uit meerdere black boxes. In level 2 worden de black boxes uit level 1 white boxes met daarin black boxes. Tijdens elke stap worden de black boxes beschreven wat de verantwoordelijkheid en taken zijn. Dit is een goede manier om een systeem gestructureerd te ontleden. Het is ideaal om in overleg te gaan met de stakeholders op een abstract niveau zonder verdere details op hoe de implementatie plaats gaat vinden. Hierna kunnen er modulaire software blokken gecreëerd worden.

5.1 White box STM32H7 dual-core

Figuur 7 geeft level 1 weer en zoomt in op de STM32H7 dual-core. Ook toont het naar welke core de in-/outputs van STM32H7 dual-core gaan.



Figuur 7 BBV STM32H7 level 1

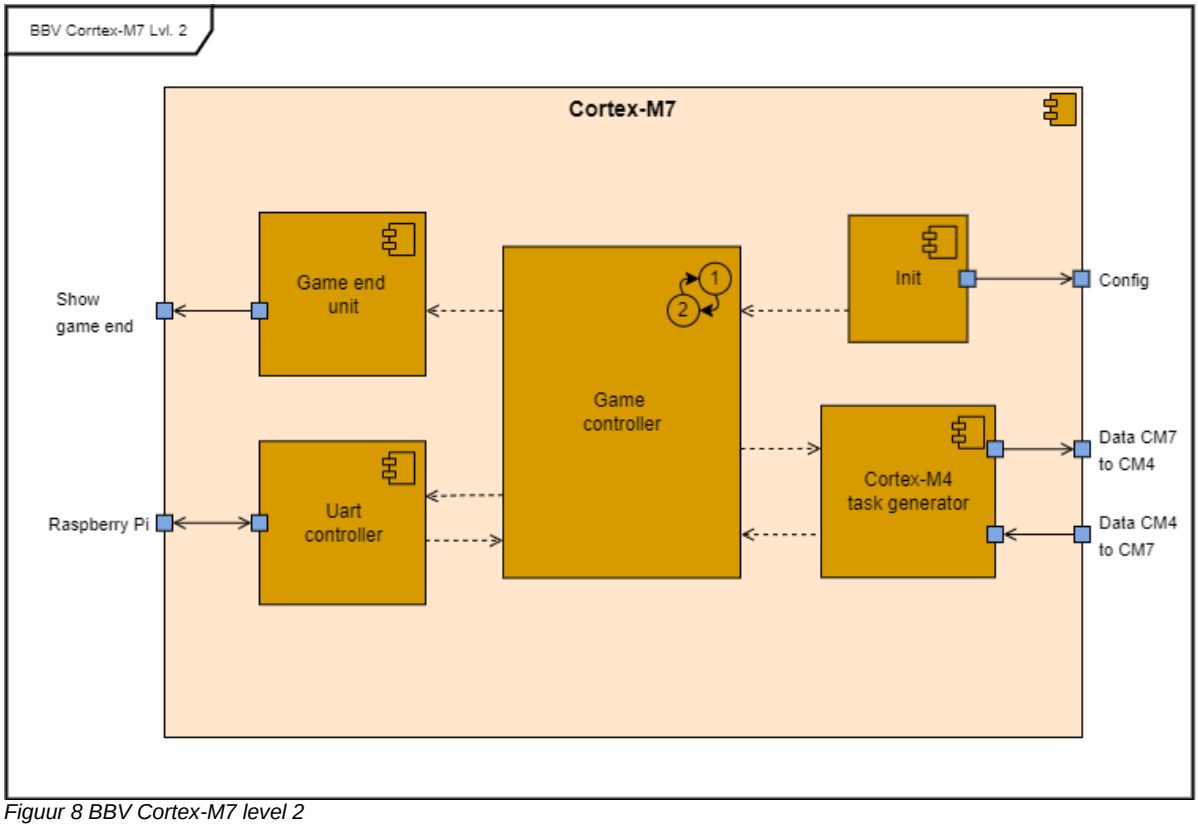
Tabel 8 Beschrijving BBV STM32H7 level 1

Black box	Beschrijving
Cortex-M7	De Cortex-M7 core van de STM32H7 wordt gebruikt voor de machine handling. Deze core is verantwoordelijk voor het spelverloop, de communicatie met de Raspberry Pi, Het initialiseren van het hele systeem en een output genereren over het resultaat van het spel.
Cortex-M4	De Cortex-M4 core van de STM32H7 wordt gebruikt voor de real-time motion control. Deze core is verantwoordelijk voor het afhandelen van alle hardware componenten zoals de Coin picker, Motoren, Coin seporator, Board opener en User detect.

5.2 Level 2

Dit hoofdstuk maakt van de relevante blackboxes uit level 1 white boxes en beschrijft hoe de interne blokken verbonden zijn.

5.2.1 White box Cortex-M7

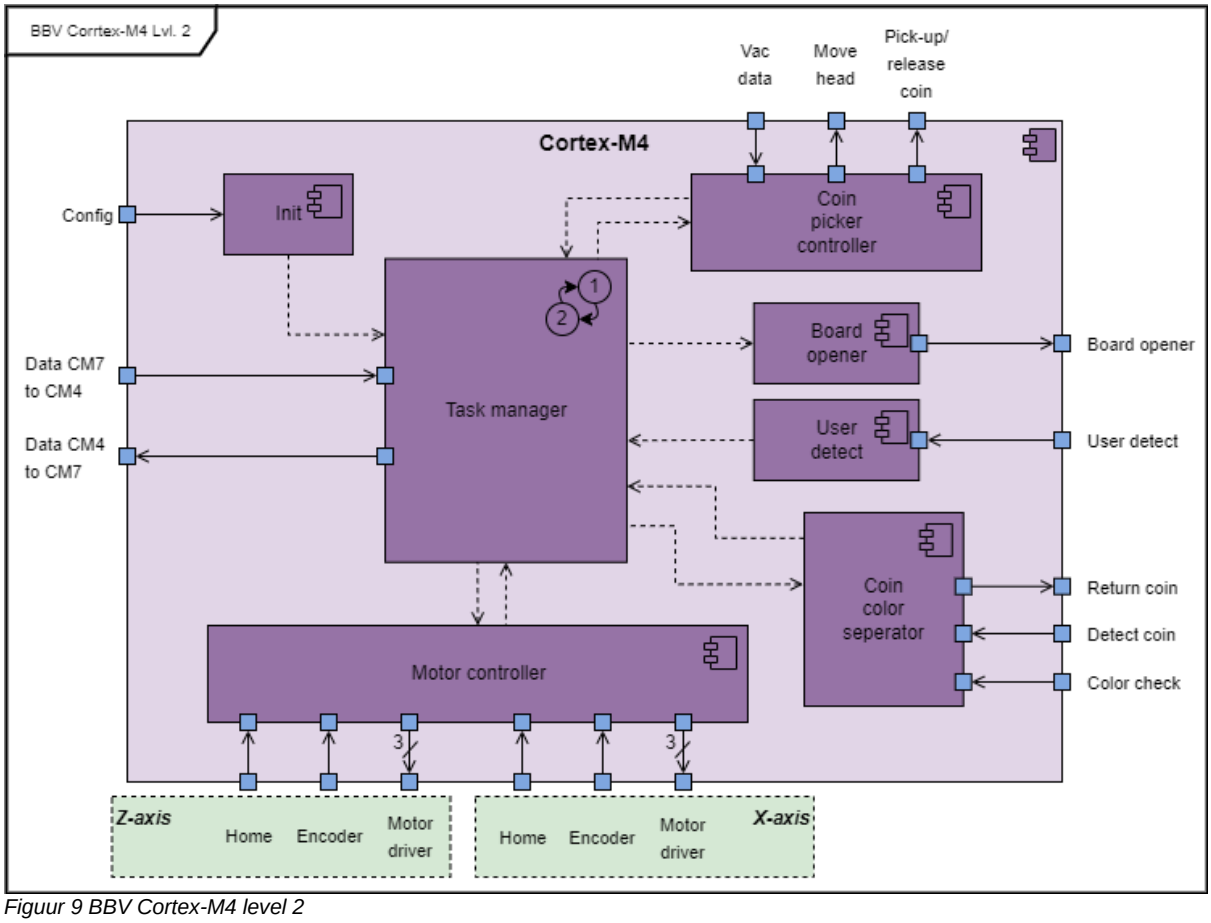


Figuur 8 BBV Cortex-M7 level 2

Tabel 9 Beschrijving BBV Cortex-M7 level 2

Black box	Beschrijving
Game controller	De Game controller is verantwoordelijk voor het spelverloop doormiddel van een state machine.
Init	Wordt een keer gedraaid om de Cortex-M7 te initialiseren en op te starten.
UART controller	De UART controller implementeert alle communicatie via UART.
Cortex-M4 task generator	De Cortex-M4 task generator implementeert de communicatie met de Cortex-M4.
Game end unit	De Game end unit implementeert alle communicatie naar een output voor de speler.

5.2.2 White box Cortex-M4



Figuur 9 BBV Cortex-M4 level 2

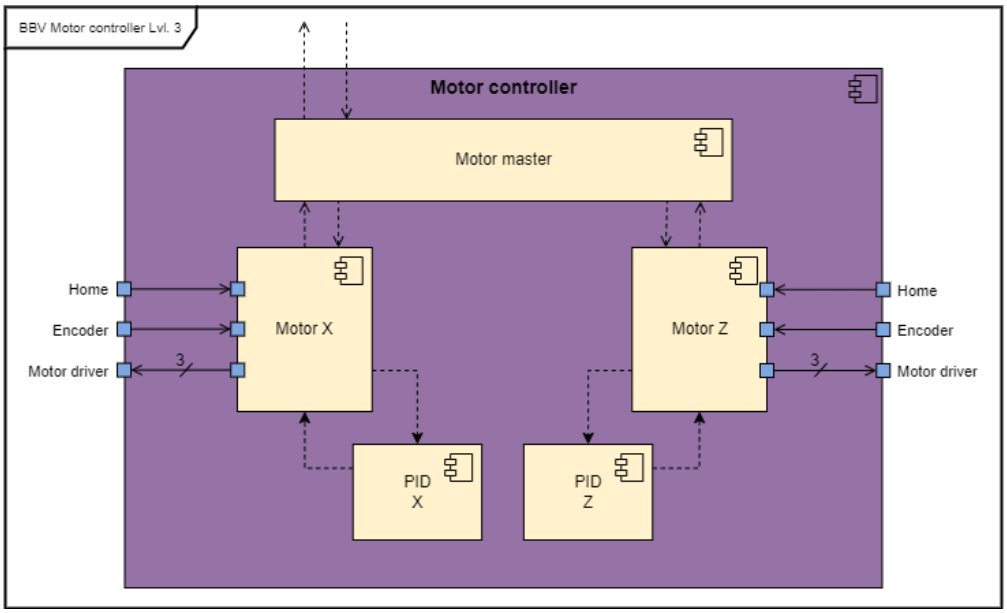
Tabel 10 Beschrijving BBV Cortex-M4 level 2

Black box	Beschrijving
Task manager	De Task manager is verantwoordelijk voor het verloop van de uit te voeren opdrachten van de Cortex-M7 doormiddel van een state machine. Door gebruik te maken van een “Task manager” kan de functionaliteit van de andere onderdelen modulair gebouwd worden. Die hoeven immers niet van elkaars bestaan/status af te weten. Die informatie wordt door de Task Manager bijgehouden.
Init	Wordt een keer gedraaid om de Cortex-M4 te initialiseren en op te starten. In deze fase worden de sensoren getest op aanwezigheid en de motoren voeren een “homing” protocol uit.
Motor controller	De Motor controller implementeert alle communicatie met de motor drivers en de PID regelaar.
Coin color separator	De Coin color separator implementeert alle functies voor het afhandelen van het scheiden van de fiches.
User detect	De User detect implementeert alle functies voor het afhandelen van de sensoren voor de input van de speler.
Board opener	De Board opener implementeert alle functies voor het afhandelen van het open van het 4-op-1-rij bord als het spel klaar is.
Coin picker controller	De Coin picker controller implementeert alle functies voor het afhandelen van het oppakken en loslaten van een fiche.

5.3 Level 3

Dit hoofdstuk maakt van de relevante blackboxes uit level 2 white boxes en beschrijft hoe de interne blokken verbonden zijn.

5.3.1 White box Motor controller

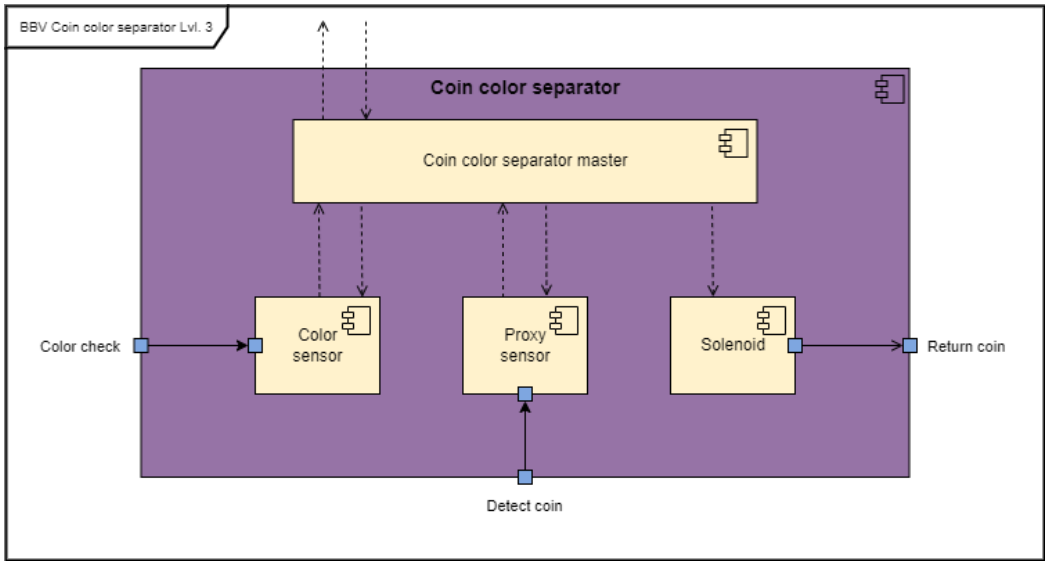


Figuur 10 BBV Motor controller level 3

Tabel 11 Beschrijving BBV Motor controller level 3

Black box	Beschrijving
Motor master	De motor master is verantwoordelijk voor het aansturen van beide motoren.
Motor X	Motor X regelt de communicatie met de motor voor de X-as.
PID X	PID X creëert de control loop met feedback voor motor X.
Motor Z	Motor Z regelt de communicatie met de motor voor de Z-as.
PID Z	PID Z creëert de control loop met feedback voor motor Z.

5.3.2 White box Coin color separator

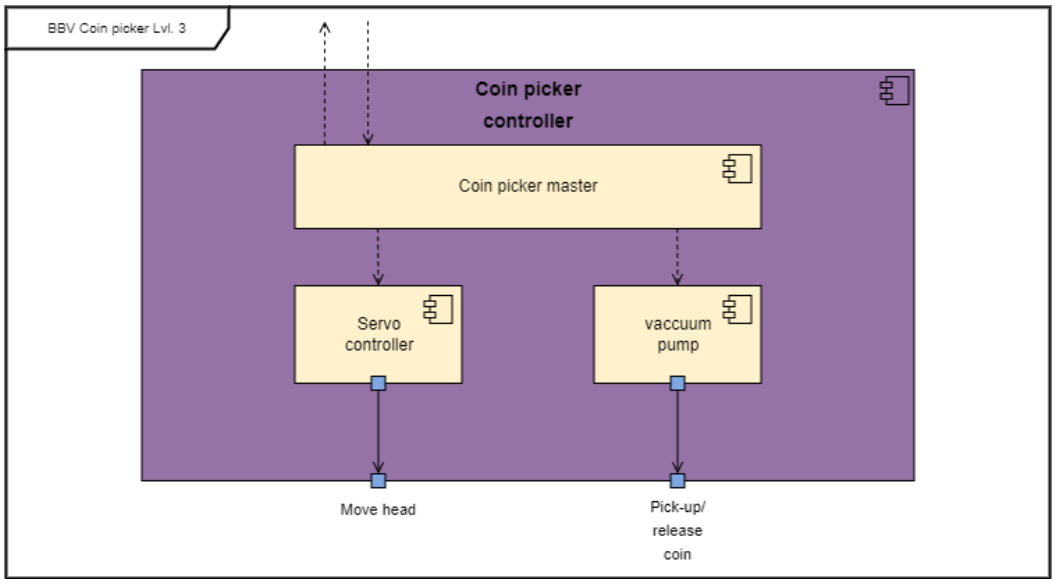


Figuur 11 BBV Coin color seperator level 3

Tabel 12 Beschrijving BBV Coin color seperator level 3

Black box	Beschrijving
Coin color separator master	De Coin color separator master is verantwoordelijk voor het ontvangen van de sensor data en het aansturen van de solenoid.
Color sensor	Vraagt de data op over de kleur van een fiche.
Proxy sensor	Krijgt data of er een fiche aanwezig is.
Solenoid	Is verantwoordelijk voor het activeren van de flipper.

5.3.3 White box Coin picker



Figuur 12 BBV Coin picker level 3

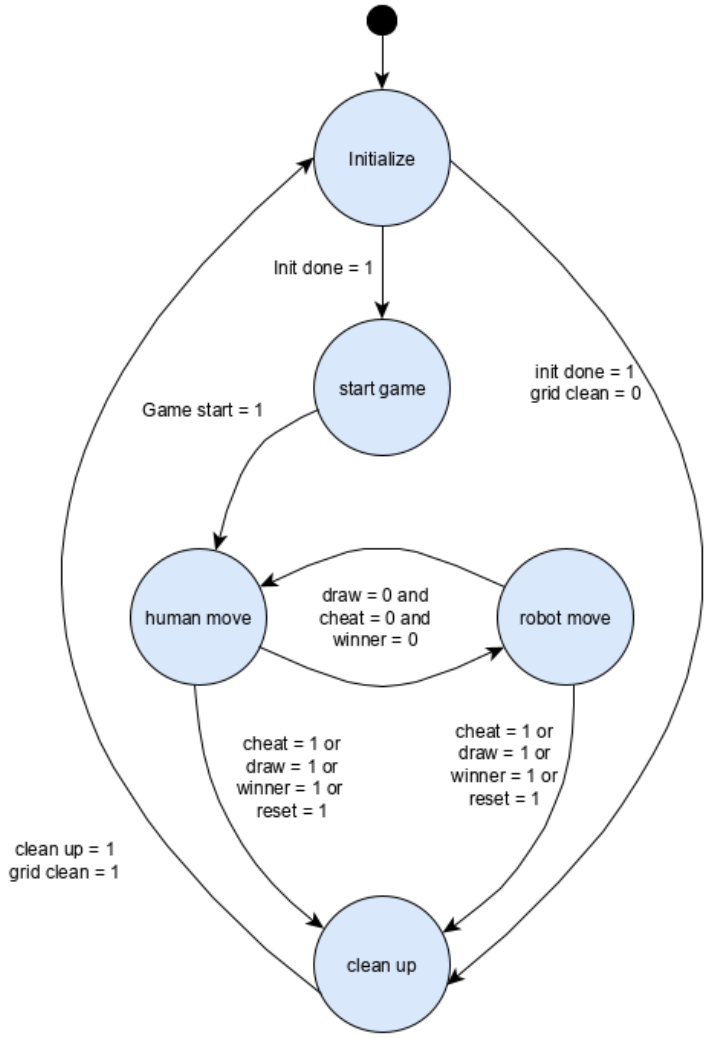
Tabel 13 Beschrijving BBV Coin picker level 3

Black box	Beschrijving
Coin picker master	De Coin picker master is verantwoordelijk voor het aansturen van de servo en de vaccuum pomp van de vaccuumgripper.
Servo controller	Regelt de aansturing van de servo motor.
Vaccuum pump	Regelt de aansturing van de vaccuum gripper om de fiches op te pakken.

6 Runtime view

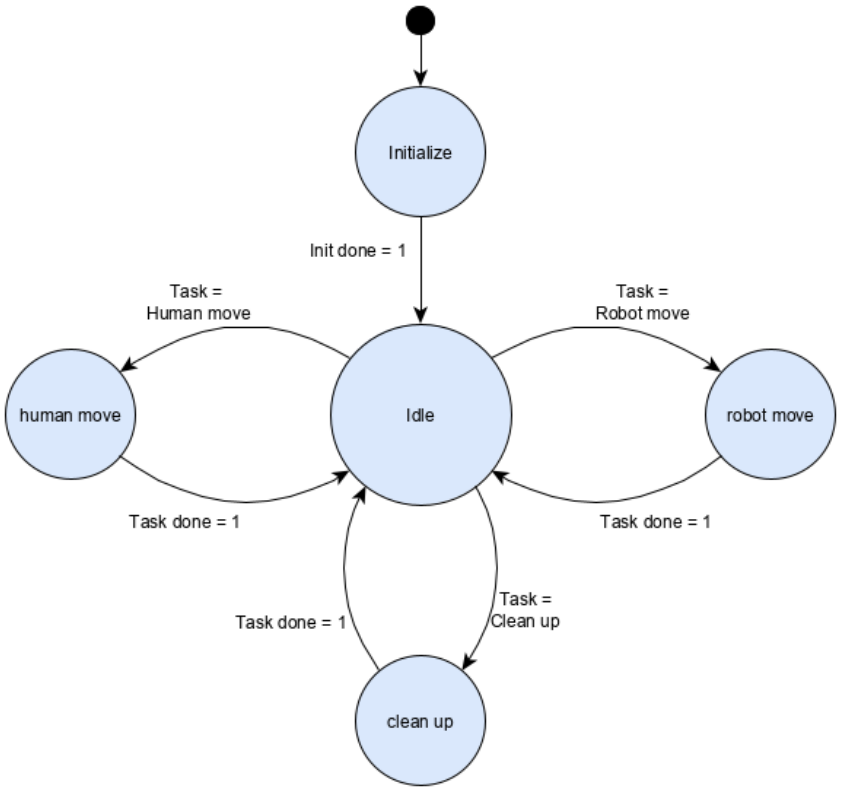
De Runtime view beschrijft het concrete gedrag en de interacties van de building blocks van het systeem. Dit wordt gedaan door middel van een highlevel statemachine (Figuur …) en scenario's die worden weer gegeven in de subhoofdstukken.

6.1 High level overzicht



Figuur 13 State machine Cortex-M7

De Statemachine in Figuur 13 geeft aan hoe de 4-op-1-rij op het hoogste niveau handelt. Deze Statemachine draait op de Cortext-M7 binnen de game controller en is daarmee de main flow van de 4-op-1-rij. De eerste state is de "initialize" state. In deze state wordt het hele systeem opgestart en geïnitialiseerd. Zodra deze state klaar is gaat het systeem in de "start game" state. In deze state wacht het systeem tot het spel begint. Als het spel gestart is, is het de beurt aan de speler om als eerste zijn/haar fiche in het 4-op-1-rij spel te doen en bevind het systeem zich in de "human move" state. Elke beurt controleert de robot of er een winnaar, valsgespeeld of gelijk spel is. Als dit niet het geval is gaat de beurt naar de robot en gaat het systeem naar de state "robot move". In deze state voert de robot de berekende zet uit. Zodra het spel is afgelopen gaat de "clean up" state van start en worden alle fiches opgeruimd. De rode fiches gaan naar de robot en de gele naar de bak van de speler.

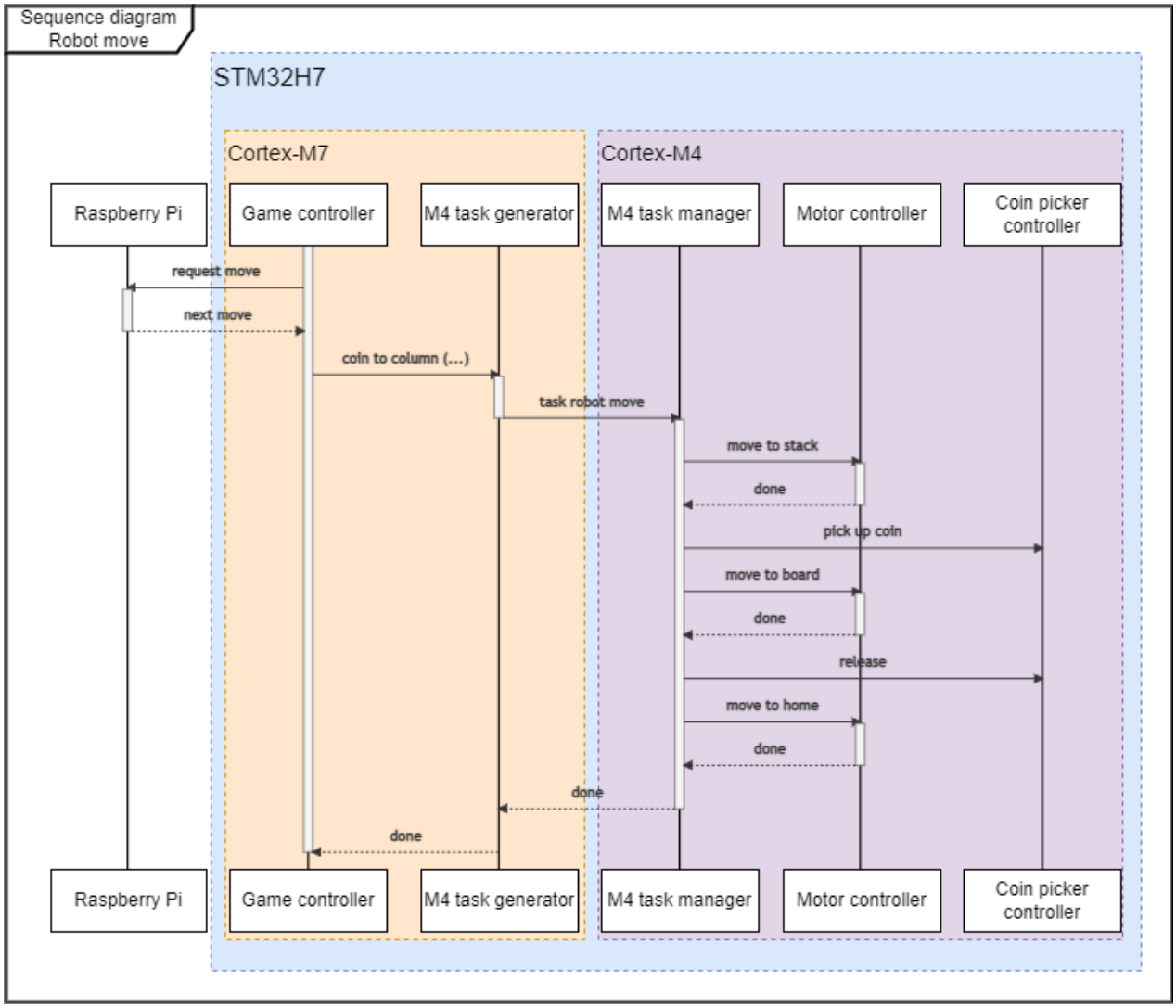


Figuur 14 State machine Cortex-M4

Zoals te zien in Figuur 9 bevindt zich op de Cortex-M4 ook een statemachine. Deze state machine is weergegeven in Figuur 14 en draait op de module Task manager. De eerste state is de “Initialize” state. In deze state worden de hardware componenten geconfigureerd en ingesteld. Daarna heeft de Task manager een “Idle” state. In deze state wordt gewacht tot de Task manager een task krijgt van de Cortex-M7. Zodra een task binnenkomt wordt aan de hand van de task de volgende state bepaald. Dit kan de “human move”, “robot move” of “clean-up” state zijn. Elk van deze states stuurt de hardware aan om de task te voltooien. Als de task voltooid is gaat de Task manager terug naar de Idle state.

In de volgende subhoofdstukken worden de states “robotmove”, “human move” en “clean-up” weergegeven in een sequence diagram. Een sequence diagram geeft aan hoe de verschillende building blocks zich tot elkaar verhouden door de tijd en wat voor signaal ze sturen.

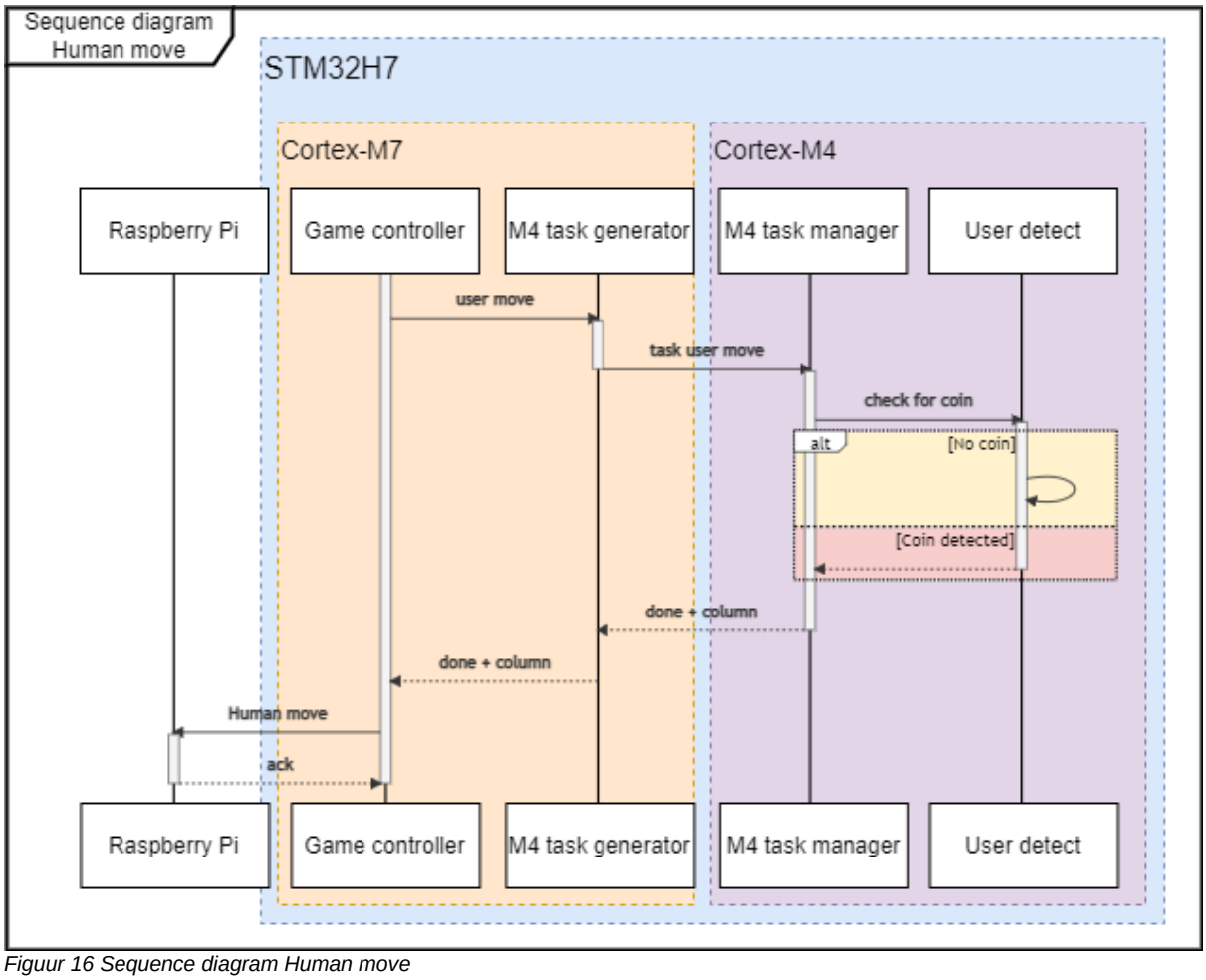
6.2 Robot move



Figuur 15 Sequence diagram Robot move

De “Robot move” in Figuur 15 begint met het opvragen van de volgende zet aan de Raspberry Pi door de Game controller. De Raspberry Pi geeft de volgende zet terug waarna de Game controller aan de M4 task generator doorgeeft wat voor opdracht er gemaakt moet worden. Zodra de opdracht gemaakt is stuurt de M4 task generator de opdracht naar de M4 task manager op de Cortex-M4. Om een fiche in het bord te doen moet de robot eerst naar de plek waar de fiches zijn opgeslagen. Zodra de robot bij de opslag is aangekomen moet een fiche worden opgepakt. Nu kan de robot naar het bord bewegen om vervolgens het fiche los te laten. Tot slot moet de robot weer naar zijn “home” stand. Als de zet is uitgevoerd geeft de M4 task manager door aan de M4 task generator die op zijn beurt weer doorgeeft aan de Game controller dat de beurt voltooid is. De Game controller gaat zo naar de volgende state.

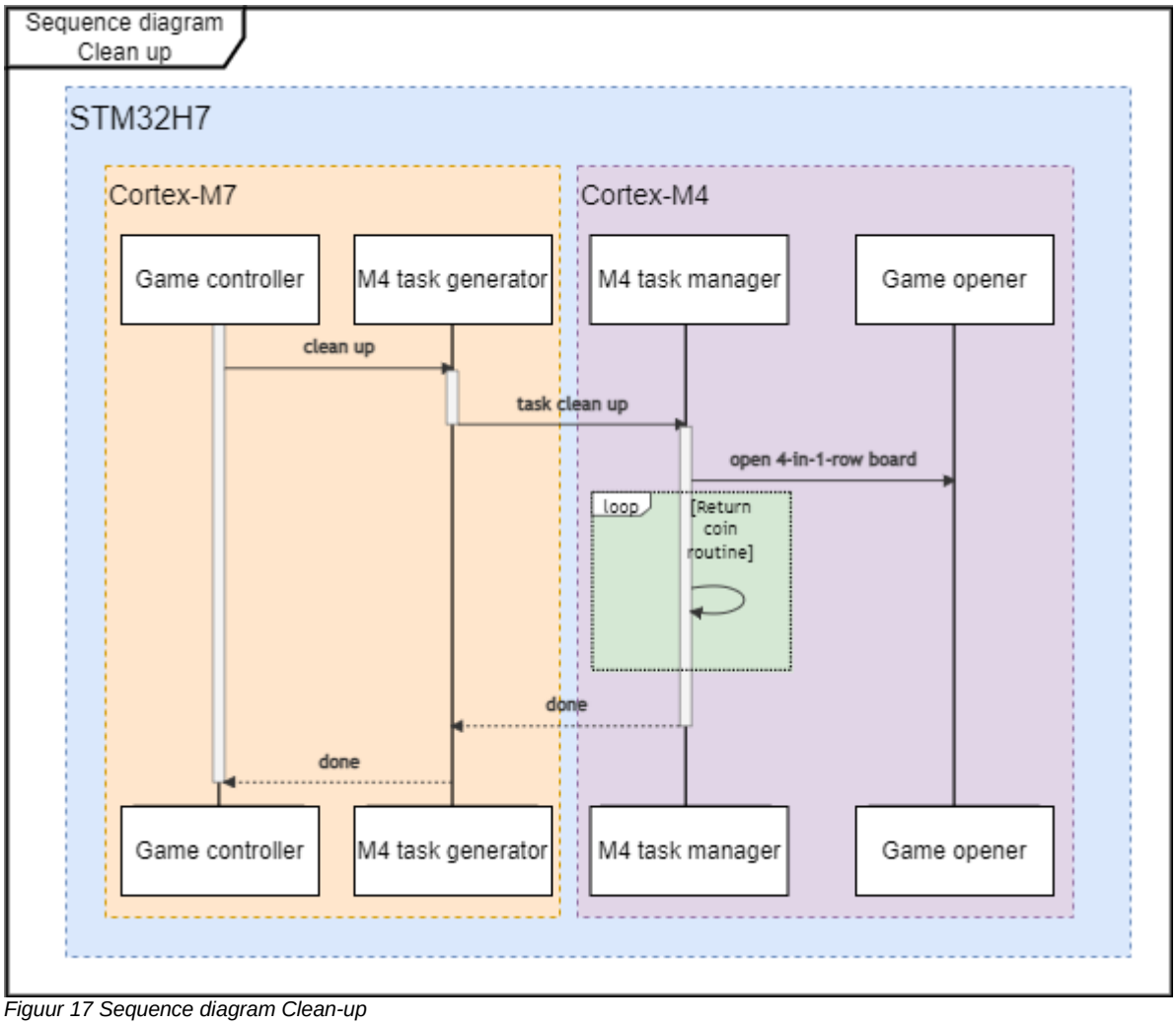
6.3 Human move



Figuur 16 Sequence diagram Human move

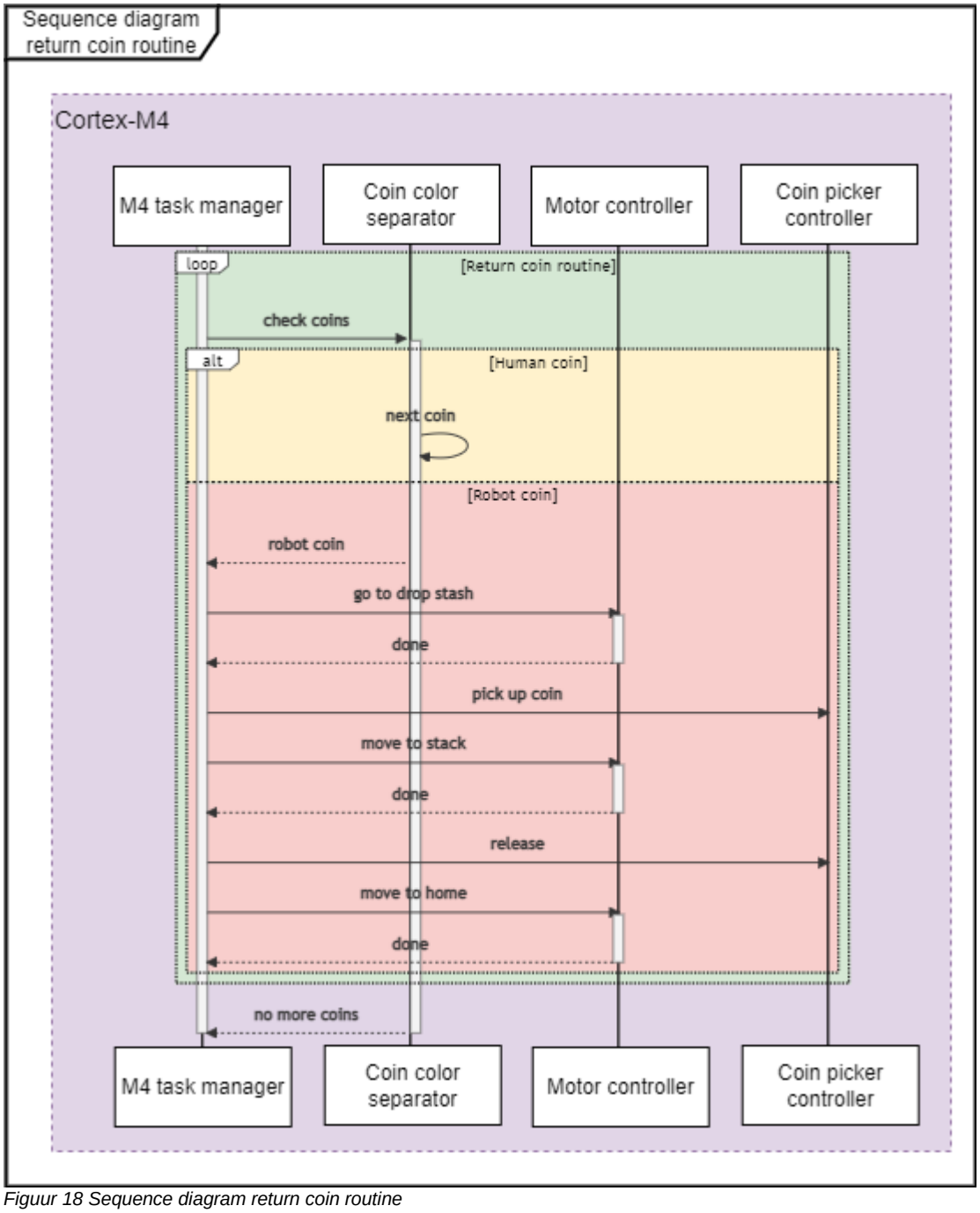
De “Human move” in Figuur 16 begint met het doorgeven dat de speler aan zet is aan de M4 task generator. De M4 task generator genereerd de opdracht en stuurt deze naar de M4 task manager op de Cortex-M4. De M4 task manager op zijn beurt geeft de instructie aan de User detect dat er een fiche verwacht wordt. Als er geen fiche wordt gespeelt door de speler blijft het systeem wachten op een zet. Zodra er een fiche gespeeld wordt door de speler wordt dit terug gekoppeld aan de M4 task manager. Deze geeft door aan de M4 task generator dat er een zet gedaan is en in welke kolom het fiche is gestopt. De M4 task generator geeft dit door aan de Game controller waarna deze de zet doorgeeft aan de Raspberry Pi. De Raspberry Pi bevestigt het krijgen van de zet en stuurt terug of het spel klaar is of verder kan.

6.4 Clean up



Figuur 17 Sequence diagram Clean-up

De “Clean up” in Figuur 17 begint met het doorgeven dat het spel afgelopen is en de opruim routine kan beginnen aan de M4 task generator. De M4 task generator genereerd de opdracht en stuurt deze naar de M4 task manager op de Cortex-M4. De M4 task manager opent het 4-op-1-rij bord en start de “Return coin routine” deze loop wordt uitgevoerd tot dat alle fiches opgeruimd zijn. Zodra dat gebeurt is geeft de M4 task manager door aan de M4 task generator dat het spel opgeruimd is. De M4 task generator geeft dit door aan de Game controller waarna het spel opnieuw kan beginnen.



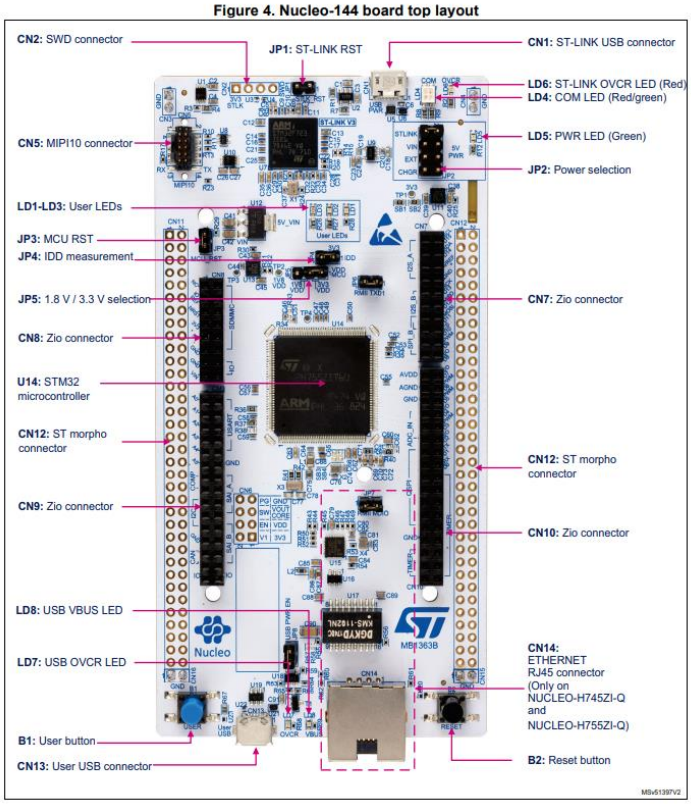
Figuur 18 Sequence diagram return coin routine

De “Return coin routine” in Figuur 18 is onderdeel van de “Clean up” uit Figuur 17 en beschrijft hoe de M4 task manager er voor zorgt dat alle fiches worden opgeruimd. Eerst stuurt de M4 task manager naar de Coin color separator dat er fiches verwacht worden. Zodra er een fiche gedetecteerd wordt gaat de Coin color separator kijken welke kleur dit fiche heeft. Als het fiche van de speler is wordt deze door een flipper naar de bak van de speler geschoten. Als het fiche van de robot is wordt de M4 task manager geinformeerd dat het gaat om een fiche van de robot. De M4 task manager stuurt de Motor controller aan om naar het fiche toe te gaan. Vervolgens wordt het fiche op gepakt en beweegt de robot naar zijn eigen fiches opslag. Hier wordt het fiche weer losgelaten en beweegt de robot naar zijn “home” stand. Deze loop wordt uitgevoerd tot alle fiches zijn opgeruimd.

7 Deployment View

7.1 NUCLEO-H755ZI-Q

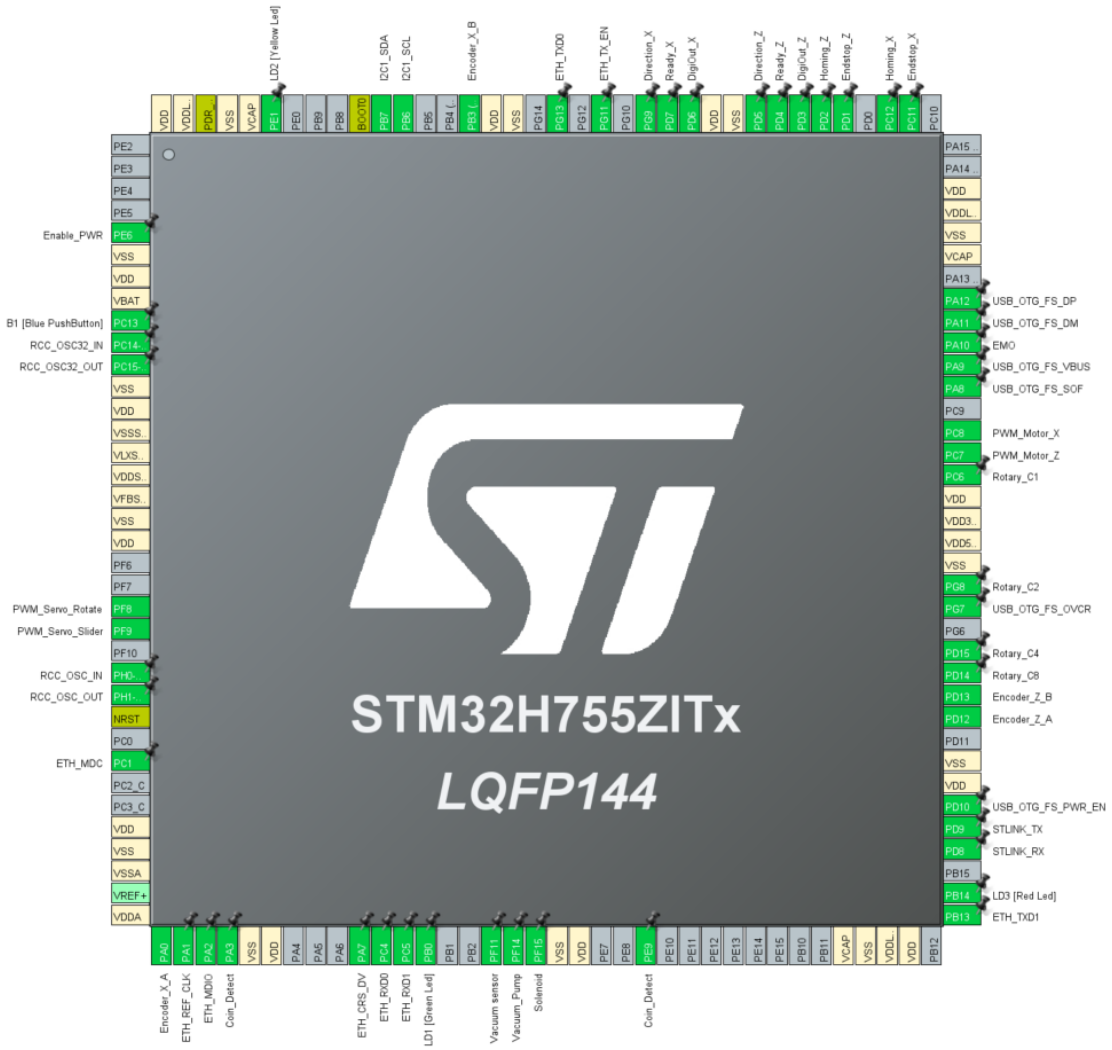
Om de software architectuur te implementeren is een Nucleo-H755ZI-Q aanwezig. Dit is een development board met de STM32H7 chip aanboort. Verder heeft dit boord alle benodigde pin-outs en hardware om het implementeren van de software blokken te faciliteren. Het plan is om op ten duur over te gaan naar een door ALTEN ontworpen PCB voor de STM32H7 chip met alle benodigde connecties. Dit valt echter buiten de scope van dit project en daarom wordt er gewerkt met de Nucleo-H755ZI-Q (Figuur 19).



Figuur 19 NUCLEO-H755ZI-Q

7.2 Pin-out NUCLEO-H755ZI-Q

Om alle functies van de 4-op-1-rij te realiseren is de pin-out in Figuur 20 samengesteld. In deze pin-out is rekening gehouden met de functie van elke pin en wat de pin moet faciliteren. In Tabel 14 staat de functie die uitgevoerd wordt door een pin, wat voor signaal er op de pin staat en aan welke pin de functie verbonden is. Zoals te zien zijn er ook pinnen vastgesteld voor ethernet. Ethernet is een functie die in de toekomst misschien gebruikt gaat worden maar buiten de scope van dit project valt. Maar het is wel van groot belang om de functie wel toe te wijzen aan de pinnen om te voor komen dat die pinnen gebruikt worden voor andere doeleinden.



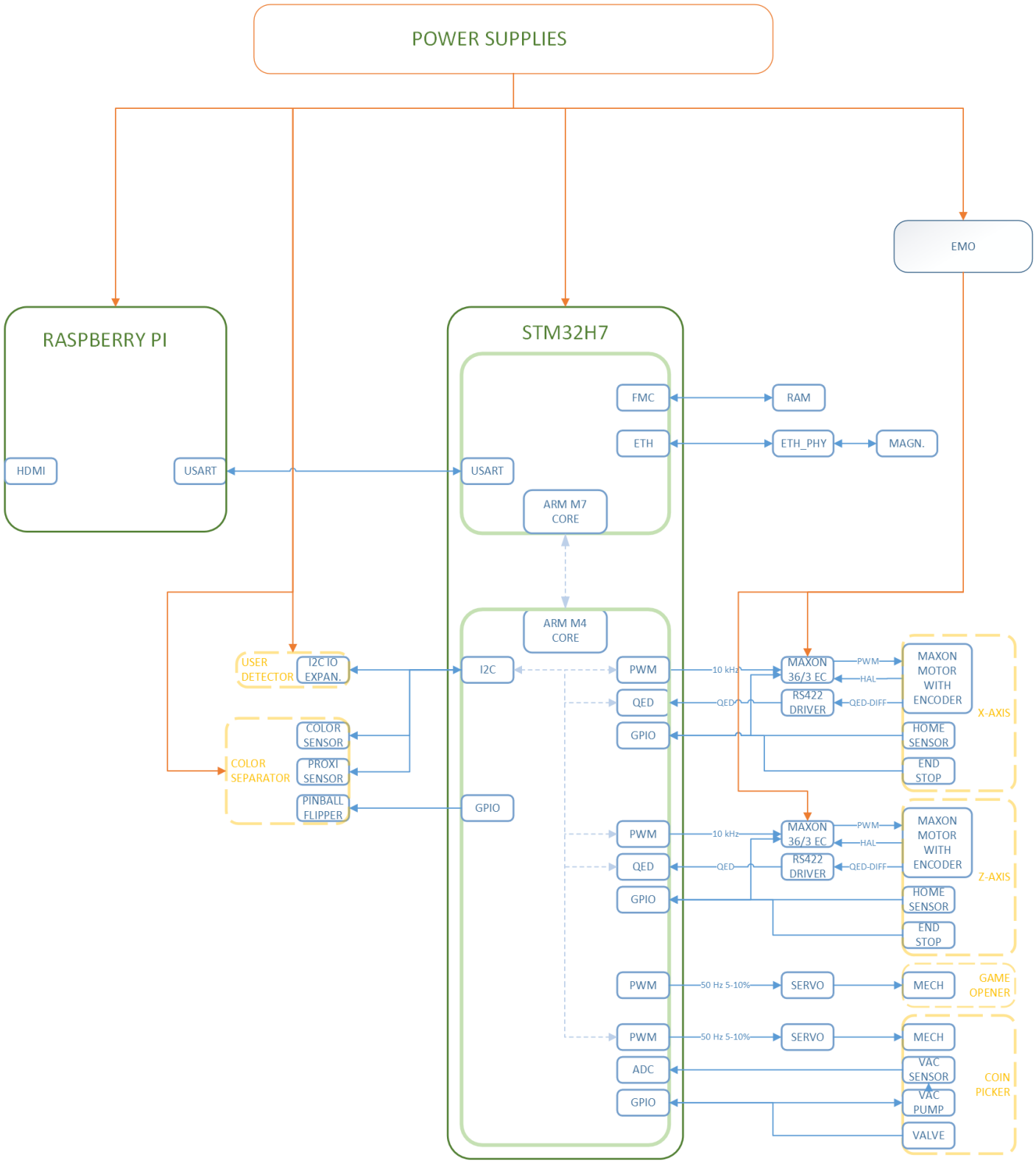
Figuur 20 pin-out STM32H755ZITx

Tabel 14 pin-out tabel

Function	STM Function	STM Pin	Connector
LD1 (Green)	GPIO	PB0	
LD2 (Yellow)	GPIO	PE1	
LD3 (Red)	GPIO	PB14	
Push button	GPIO	PC13	
Encoder X	A	PA0	
	B	PB3	
Encoder Z	A	PD12	
	B	PD13	
PWM X	PWM Timer	PC8	
PWM Z	PWM Timer	PC7	

Direction X	GPIO	PG9	
Direction Z	GPIO	PD5	
Ready X	GPIO	PD7	
Ready Z	GPIO	PD4	
DigOut X	GPIO	PD6	
DigOut Z	GPIO	PD3	
PWM Servo Slider	PWM Timer	PF9	
PWM Servo Rotate	PWM Timer	PF8	
Solenoid	GPIO	PF15	
Vacuum pump	GPIO	PF14	
Vacuum sensor	ADC	PF11	
Read EMO	GPIO	PA10	
Enable PWR	GPIO	PE6	
ProxInt 1	GPIO interrupt	PE9	
I2C	I2C-SLC	PB6	
	I2C-SDA	PB7	
Coin detect Interrupt	GPIO interrupt	PA3	
Rotary switch	C1 - GPIO	PC6	
	C2 – GPIO	PG8	
	C4 – GPIO	PD15	
	C8 - GPIO	PD14	
End stops	X - GPIO	PC11	
	Z - GPIO	PD1	
Homing	X - GPIO	PC12	
	Z - GPIO	PD2	
RPI UART	Tx	PD9	
	Rx	PD8	
Ethernet	ETH_REF_CLK	PA1	
	ETH_MDIO	PA2	
	ETH_CRS_DV	PA7	
	ETH_TXD1	PB13	
	ETH_MDC	PC1	
	ETH_RXD0	PC4	
	ETH_RXD1	PC5	
	ETH_TX_EN	PG11	
	ETH_TXD0	PG13	

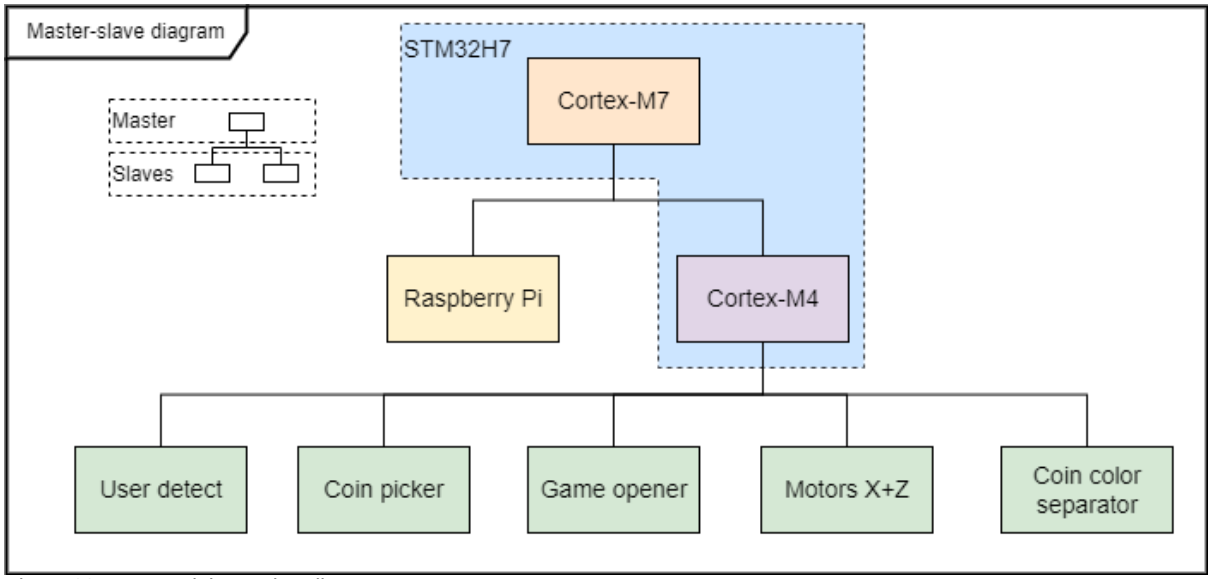
7.3 Hardware lay-out



Figuur 21 Hardware lay-out 4-op-1-rij met STM32H7

7.4 Master-Minion verhouding

Om aan te geven hoe de processoren zich verhouden in een Master-Minion verhouding is Figuur 22 opgesteld. Hierin is te zien dat de Cortex-M7 de top-master van het systeem is. Op de Cortex-M7 draait de game flow en die bepaald dus het volledige spel verloop. De Raspberry Pi en Cortex-M4 zijn een minion van de Cortex-M7 en voeren taken uit als de Cortex-M7 dat vraagt. De Cortex-M4 is ook een master voor de hardware. Alle hardware systemen van de 4-op-1-rij zijn een minion van de Cortext-M4 en voeren taken uit als de Cortex-M4 dat vraagt.



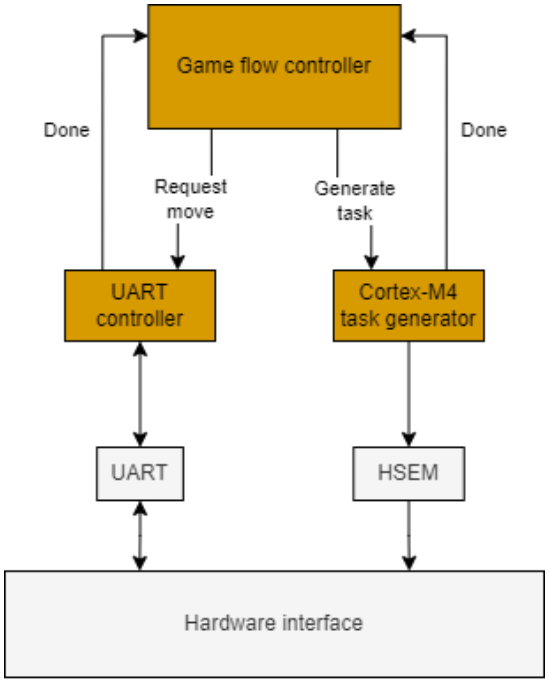
Figuur 22 Master-minion verhouding

8 Modulaire software implementatie

(Hoofdstuk 5) geeft weer welke software blokken aanwezig moeten zijn om een volledig werkende 4-op-1-rij robot te implementeren. Hier is niet weergegeven hoe deze van het top-level tot de hardware samen hangen. Dat wordt in dit hoofdstuk verder uitgelegd aan de hand van diagrammen.

8.1 Cortex-M7 core

In Figuur 8 worden alle software blokken van de Cortex-M7 core weergegeven. Om weer te geven hoe de blokken zich verhouden tot de hardware is het diagram in Figuur 23 opgesteld.



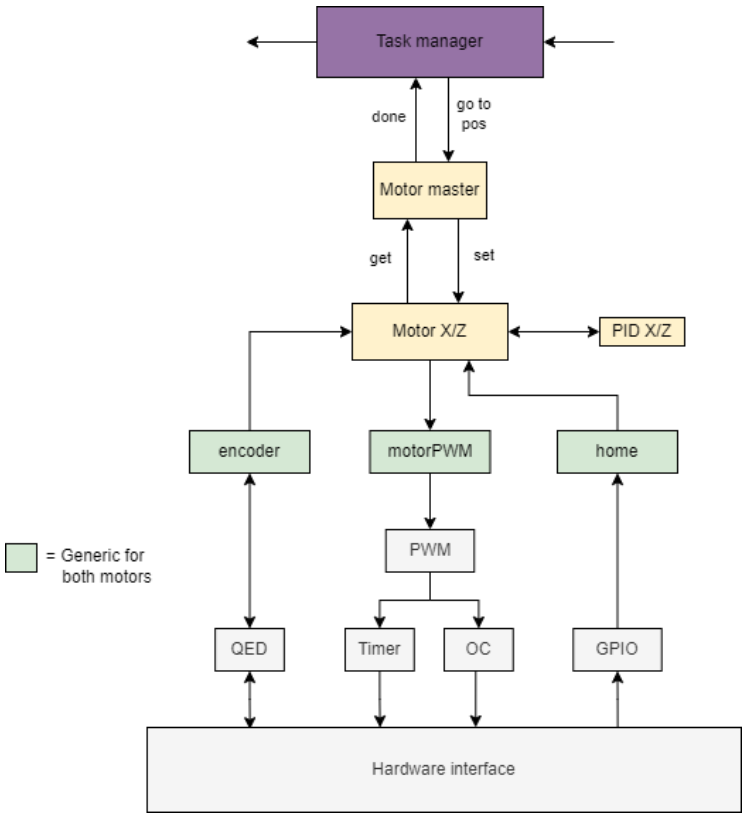
Figuur 23 IBD game controller

8.2 Cortex-M4 core

In Figuur 9 worden alle software blokken van de Cortex-M4 core weergegeven. Om weer te geven hoe de blokken zich verhouden tot de hardware zijn alle diagrammen in de subhoofdstukken opgesteld.

8.2.1 Motor controller

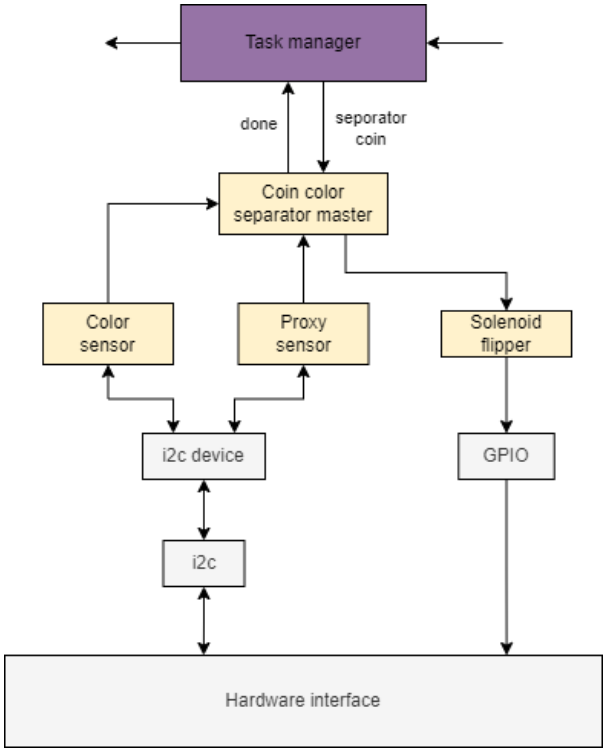
In Figuur 24 geeft het diagram van de motoren weer. De blokken voor Motor X zijn hetzelfde als voor Motor Z. Deze blokken zijn in weergegeven in het groen.



Figuur 24 IBD Motor controller

8.2.2 Coin color separator

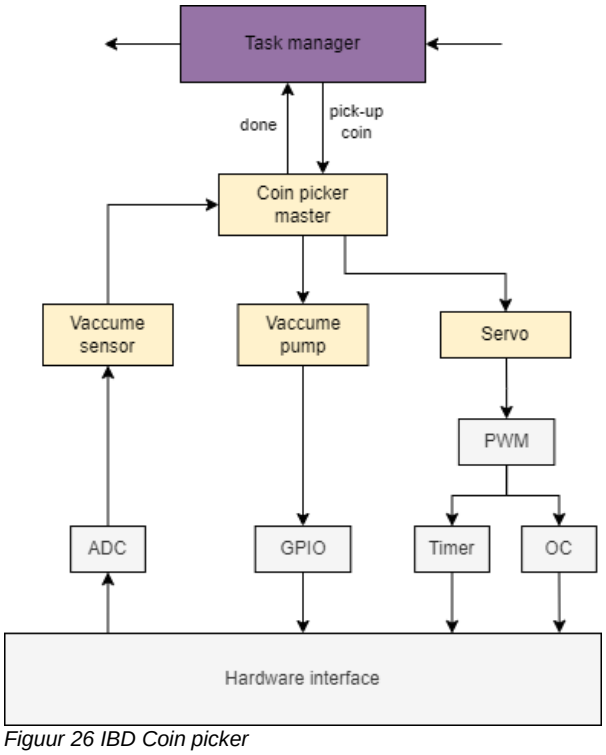
Figuur 25 geeft het diagram weer voor de Coin color separator. Het i2c device blok is een generiek blok dat alle informatie voor een i2c sensor afhandelt.



Figuur 25 IBD Coin color seperator

8.2.3 Coin picker

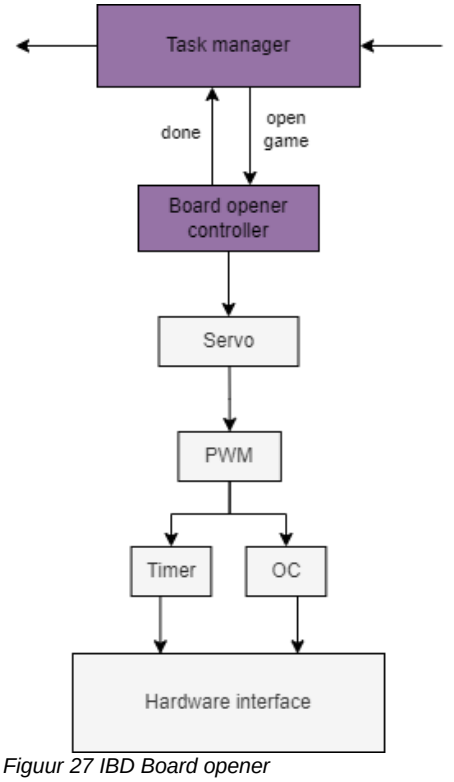
Figuur 26 geeft het diagram weer voor de Coin picker.



Figuur 26 IBD Coin picker

8.2.4 Board opener

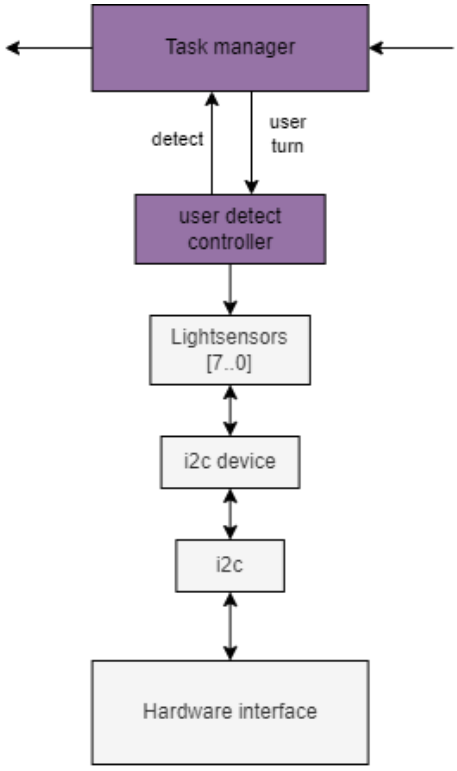
Figuur 27 geeft het diagram weer voor de Board opener.



Figuur 27 IBD Board opener

8.2.5 User detect

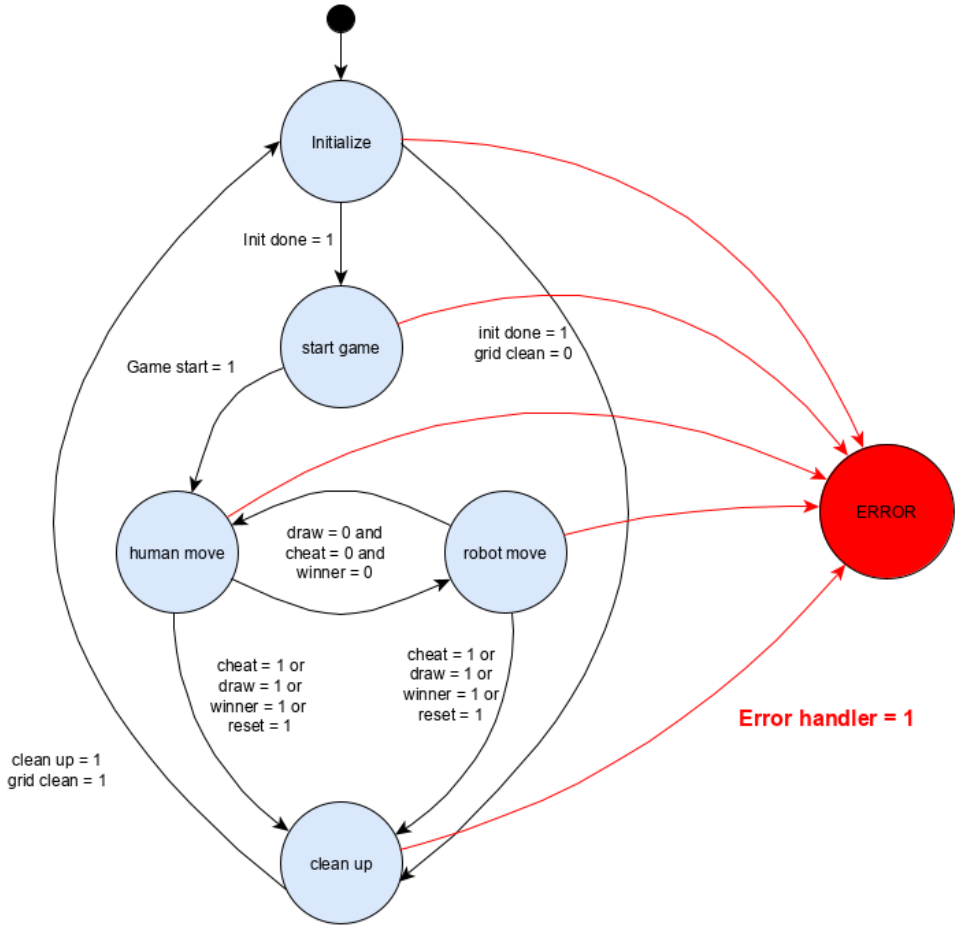
Figuur 28 geeft het diagram weer voor de User detect.



Figuur 28 IBD User detect

9 Foutafhandeling

Volgens de architectuur die in voor gaande hoofdstukken is beschreven kan het systeem vloeiend draaien. Toch bestaat er een kans dat er een fout optreedt. Als de motoren niet goed de “homing” fase uitvoeren, een van de sensoren reageert niet meer of de motoren falen als een fiche naar het bord wordt gebracht. Zodra het systeem een error melding geeft moet deze afgehandeld worden. Dit betekent voor de veiligheid hardware uitzetten en een indicatie geven aan de Operator dat het systeem een fout heeft gedetecteerd. Figuur 29 laat zien hoe een error binnen de hoofd states van de Game flow controller wordt afgehandeld.



Figuur 29 State machine met error handler

In de “ERROR” state worden alle motoren en andere hardware componenten uitgezet binnen de twee cores van de STM32H7 en blijft het systeem in deze state tot deze gereset wordt.