Software Architecture Documentation -

4-on-1-row robot

Version 3
Date: 13-4-2022

ALTEN
Pascal Faatz

Versie history

Versie	Thatum	StatUS	Writer	Remark
1	31-3-2022	Plan	Pascal Faatz	First draft up to Chapter 8
2	13-4-2022	Plan	Pascal Faatz	Process feedback Jeedella, Aniel and Berend
3	3-5-2022	Completion	Pascal Faatz	Chapter 8 onwards

Acroniems and abbreviations

Term	Explanation
BSP	Board Support Package

Referenced documents

Id	Reference	Titel	Thatum	Writer
D1	SRD_Pascal_Faatz_V3.docx	SRD	16-3-2022	Pascal Faatz

Table of contents

List of figures

List of tables

1 Introduction and goals

ALTEN has developed a robot in-house that can play 4-in-1 row against a human opponent using an algorithm. Using industrial components, the 4-on-1 row robot is built to demonstrate the knowledge of different systems. The robot will be used as a demonstration unit at trade fairs and open days, where it is available for passers-by to play a round. At the front, the player can then place his/her move on the board, after which the 4-on-1-row robot will devise and execute a move. For this, the system keeps track of which moves have been played by his opponent. When the move is determined by the robot, the combination of the X-Z platform with rotating vacuum gripper will pick up a stone and enter it at the chosen spot in the game. Once the game is finished or reset, the 4-on-1-row robot will collect all the chips and sort them by color to be ready for the next round.

Currently, the 4-on-1 row robot runs on a Raspberry Pi + STM32 microcontroller (Figure 1 1). On the Raspberry Pi runs the algorithm to determine the next move of the system and on the STM32 controller runs the operating system. Within ALTEN, this project was initially used to give consultants who are not on a project in between periods a challenge. As a result, several consultants worked on it over a longer period of time. This has resulted in a very unclear and unclear architecture of the software and hardware. This also applies to the control system of the 4-on-1-row robot.

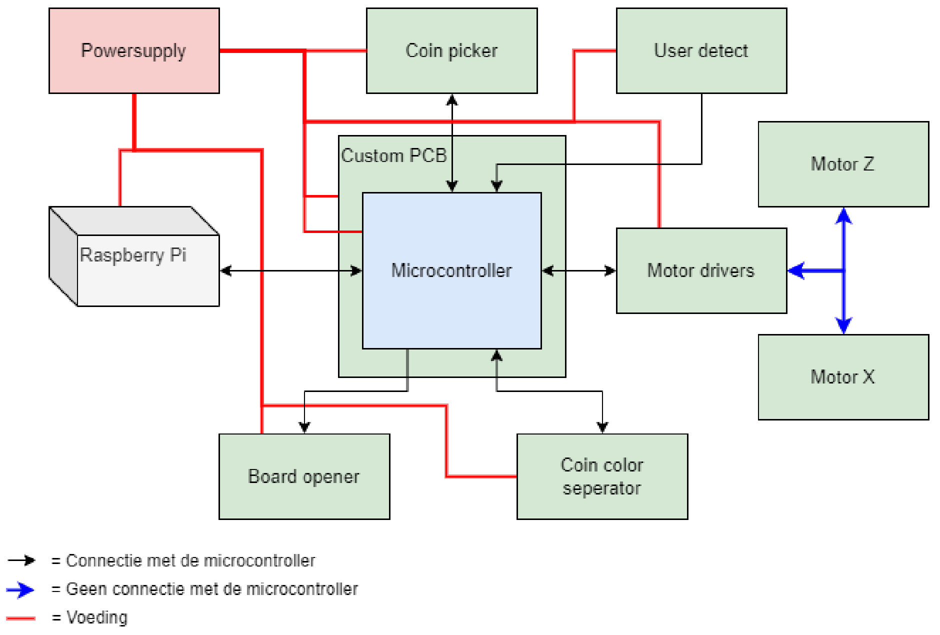

Figure 1 1 4-on-1 row

1.1 Requirements oversight

Due to the unclear architecture, it is very difficult to expand or upgrade the operating system. Because the 4-on-1-row robot is also at trade fairs and shows what ALTEN has to offer, it is necessary that the robot is easy to maintain and can easily undergo an upgrade. This requires a complete re-design of the operating system to solve the unclear and unclear architecture and enable upgrades.

One of the main requirements is therefore to implement a structured architecture and modular implementation. Furthermore, it is important to draw up state and flow diagrams to visually show the operation of the system. A BSP (Board Support Package) must also be created.

For the full list of requirements refer to the SRD (D1).

1.2 Quality goals

The quality goals are essential to indicate which quality requirements the system must meet. These goals will also be tested at a later stage and checked whether the system complies with this.

Table 1 Quality goals

Priority	Quality goal	Concrete scenario
1	Easy to understand (Structured architecture)	People who are going to work on the 4-on-1-row robot must understand at once how the hardware and software are related by reading the architecture.
2	Maintenance and upgrades (Modular construction of software blocks)	People who are going to upgrade or modify the 4-on-1 row should be able to modify or replace software blocks without affecting other parts of the system.
3	Robust	The 4-on-1 row must be reliable and operate in all conditions when the system is running.

1.3 Stakeholders

The stakeholders are an important part of the project. They determine the requirements and set requirements for the end product. Figure 2 the distribution of stakeholders for this project.

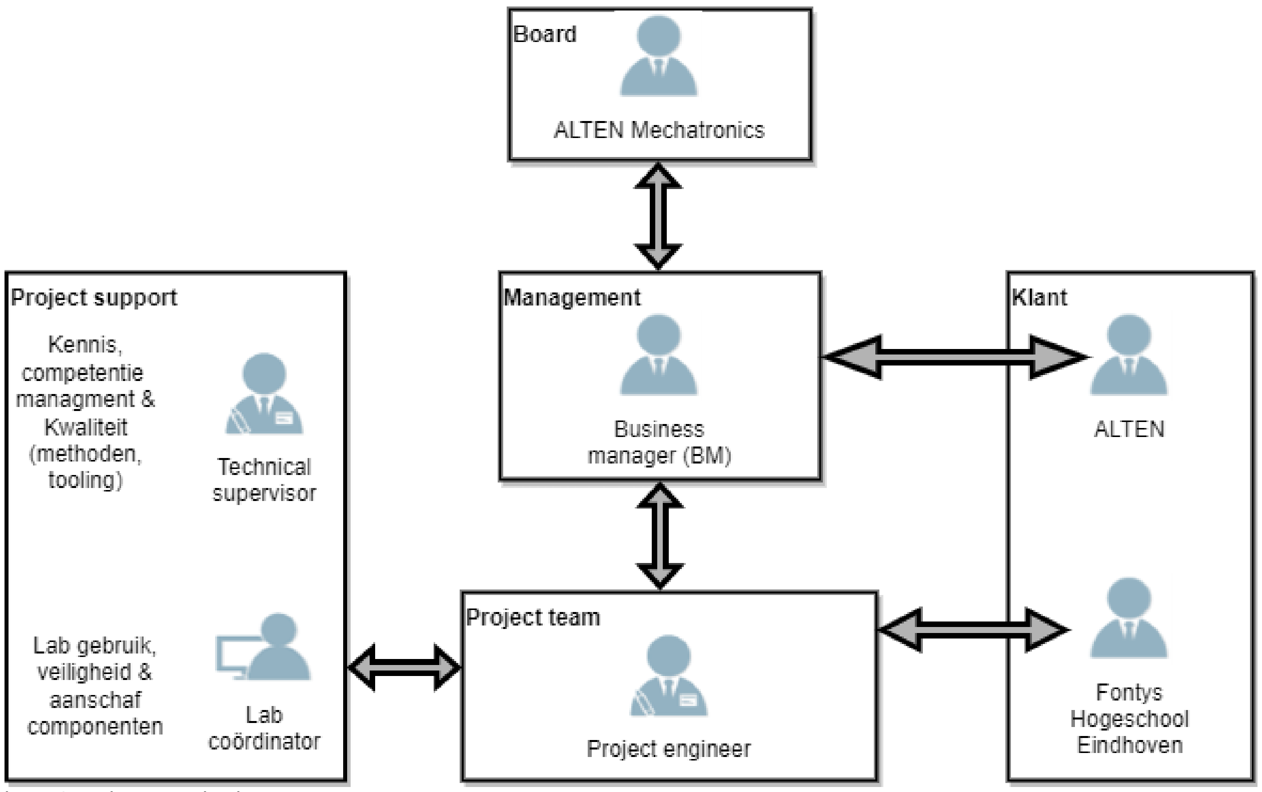

Figure 2 Project organization

Table 2 Stakeholders

Name	Role	Responsibility
Pascal Faatz	Project engineer	Carry out the project for the relevant customers and board/management.
Aniel Shri	Technical supervisor	Within the project give technical support to the project team. This involves challenging the team to come to a good solution and elaboration.
Gijs Haans	Business manager	Within the project personal development support the project team. Furthermore, the business manager keeps.
Jeedella Jeedella	Fontys University of Applied Sciences Eindhoven	Support the project from school on a technical and personal level. At the end, give an assessment about the project.
Jeroen Wilbers	Lab coordinator	The safe use of the lab and the point of contact for orders that need to be made.
Aniel Shri and Gijs Haans (as customer)	ALTEN	As an end customer, assess the requirements and use the end product.
Chris Kalis	ALTEN Mechatronics unit	The management of all project parties within ALTEN. Furthermore, the department is ultimately responsible for the project.

2 Architectural limitations

Table 3 Architecture limitations

Architecture limitation	Definition
An STM32H755 dual-core nucleo-144 board **is used**.	The choice of microcontroller is fixed. This microcontroller must therefore be used.
The hardware of the 4-on-1 row is fixed.	The hardware that is there now is used to prescribe the software.
The game course of the 4-in-1 row is fixed.	Because the 4-on-1 row has already been working, the global works are known in advance.
The project must be completed within 100 working days.	The project takes place within a school semester and must therefore be carried out within this time.
The Raspberry Pi passes the next move.	The Raspberrry Pi is the processor that passes the next move to the microcontroller.

3 Project scope and context

Figure 3 gives a global view of the people who work with the system and have what they say as input on the system or what they get back from the system. Furthermore, it shows the scope of the project. The scope of the project is the STM32H7 dual-core and its architecture.

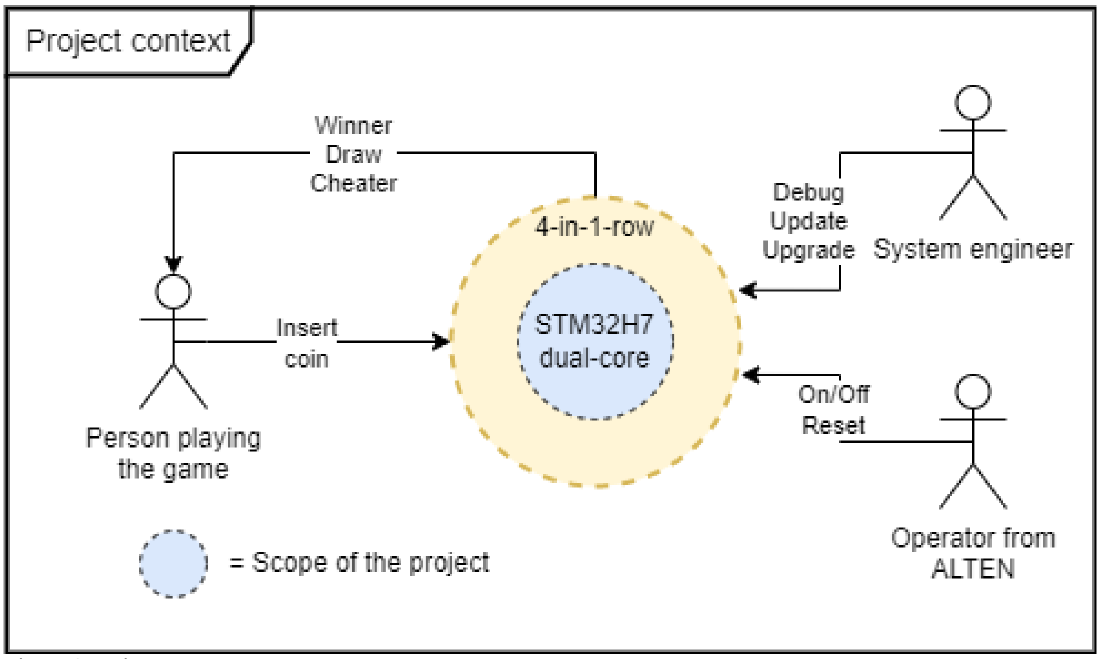

Figure 3 Project context

Table 4 Description Project context

Users	Description
Person playing 4-in-1 row	The person playing 4-in-1 row should think carefully when it's his turn column he wants to do his chip to beat the robot. If a move is known, the chip is put in the 4-on-1-row board. Furthermore, the person can see what the result is at the end of the game.
System engineer	The system engineer can debug the 4-on-1 row if an error occurs. Furthermore, the engineer can implement an update/upgrade.
Operator from ALTEN operating the 4-on-1 row	This person serves the 4-on-1 row at trade shows or other occasions. This person can turn the 4-on-1 row on/off or reset it.

3.1 System context

Figure 3 geeft a global overview of the persons involved in the system. Figure 4gives a visual representation of the subparts of the 4-on-1 row. It also shows the relationship with the STM32H7 dual-core through the inputs and outputs. This is important so that all stakeholders get an idea of what is happening broadly within the system.

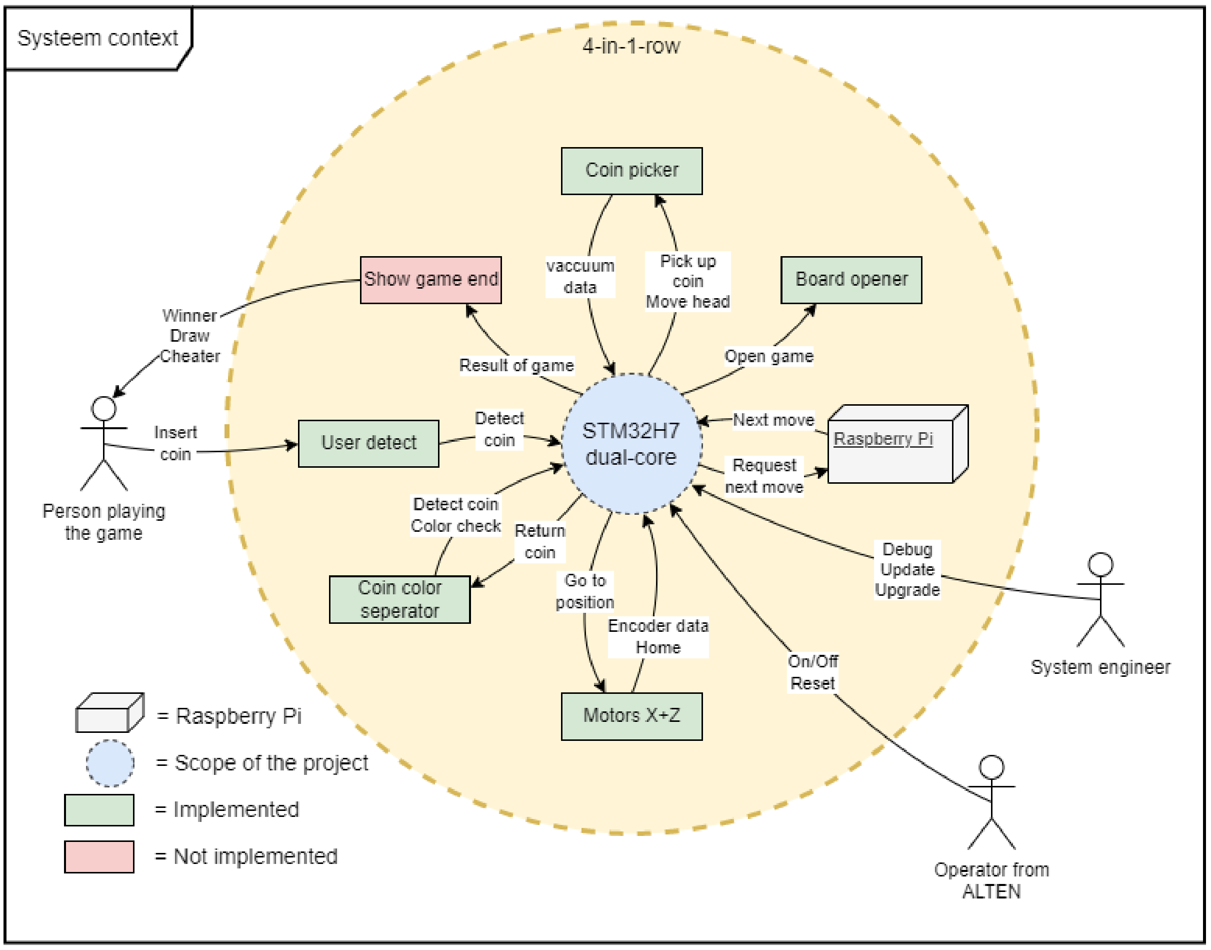

Figure 4 System context

Table 5 Description System context

Hardware block	Function
Raspberry Pi	The Raspberry Pi runs an algorithm that determines the robot's next steps.
Motors X+Z	The motors allow the robot to move its vaccuum gripper over an X and Z axis of the 4-on-1 row board.
Coin color seporator	When the game is over, the coin color seporator ensures that each chip is recognized by color. If the chip has the color of the player, the chip is shod to the player's box. If the chip has the color of the robot, the chip is placed in its own pile.
User detect	If the player puts a chip in the game, this is detected. Each column of the 4-on-1 row board contains a detection point.
Coin picker	The coin picker is the robot's arm. This is moved by the engines but can also move to put a chip in the 4-on-1-row board. Picking up or releasing a chip is done by means of a vaccuum gripper.
Board opener	The board opener ensures that all chips fall out of the 4-on-1-row board when the game is over.

3.2 Technical context

Figure 5 converts Figure 4 into a description into a technical representation of the relationship between the components and the scope. The connections are not described globally but show by what kind of protocol or input/output signal they are connected to. This is important for the stakeholders who get to work with the hardware or architecture.

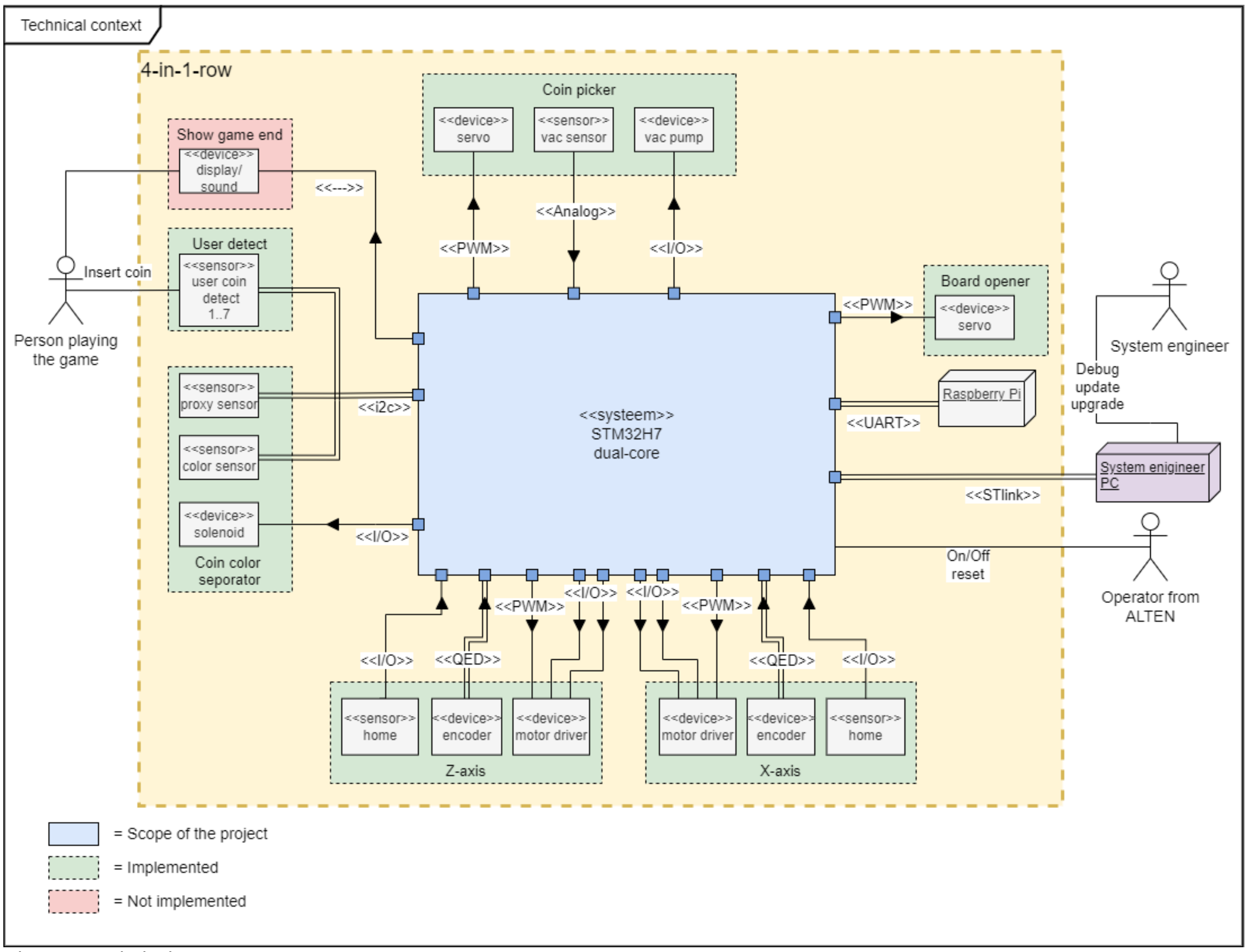

Figure 5 Technical context

Table 6 Description Technical context

Hardware block	Sub-blocks	Definition	Input to STM32H7	Number of lines	Output of STM32H7	Number of lines
Raspberry Pi	-	The Raspberry Pi communicates with the STM32H7 by means of a UART connection.	UART Rx	1	UART Tx	1
Motors/ driver x	Motor with encoder	The motor is controlled by a PWM signal, a line for the direction and a line for the ready signal. An encoder was used to determine the position of the engine. The encoder communicates with an A and B line.	QED (A line and B line)	2	1 PWM, 2 I/O	3
	Home/ End stop	The home/end stop indicates when the engine has reached the home point.	I/O	1	-	-

Engine/ driver z	Motor with encoder	See Engine X.	QED (A line and B line)	2	1 PWM, 2 I/O	3
	Home/ End stop	See Engine X.	I/O	1	-	-
Coin color seporator	Proxy sensor	The proxy sensor checks whether a sheet is present in the distribution system.	i2c (SDA, SCL)	2	-	-
	Color sensor	The color sensor checks which color the sheet is that is in the distribution system.	i2c (SDA, SCL)	2	-	-
	Solenoid	If a chip is the color of the player, the solenoid shoots the chip to the player's box by means of a pinball.	-	-	I/O	1
User detect	-	The user detect detects when a chip is thrown into the system by the player.	i2c (SDA, SCL)	2	-	-
Coin picker	Servo	The servo ensures that the vaccuum gripper can be moved.	-	-	PWM	1
	Vaccuum pump	The vaccuum gripper can suck and release a chip.	-	-	I/O	1
	Vac sensor		Analogous	1	-	-
Board opener	-	The game opener uses a servo to ensure that all chips fall out of the 4-on-1-row board when the game is over.	-	-	PWM	1

4 Solution strategy

Table 7 Solution strategy

Goal/requirement	Approach	Link to chapter
Structured architecture	To set up a structured architecture for the software of the 4-on-1 row, this document is used. The chapters and diagrams in this document show how the 4-in-1 row behaves, the software blocks are related and the software communicates with the hardware. Furthermore, a top-down approach is used by starting with the requirements. From the requirements onwards, the system is increasingly dissected.	
Modular construction of software blocks	The software is modular to ensure that an upgrade or adjustment is easy to facilitate. First, it is worked out which software blocks there are. Then how these blocks communicate with each other.	(Chapter 5,6,8)
Robust	The 4-on-1 row must operate under normal conditions as determined in advance . This can be guaranteed by testing all software blocks individually and extensively.	

The V-model is used throughout the project. The V-model is a project method that brings structure to the course of the project. For each specification or design phase on the left, there is a corresponding integration phase on the right. Each phase on the right side of the project can be verifiedand validated by the phase on the left (Figure 6).

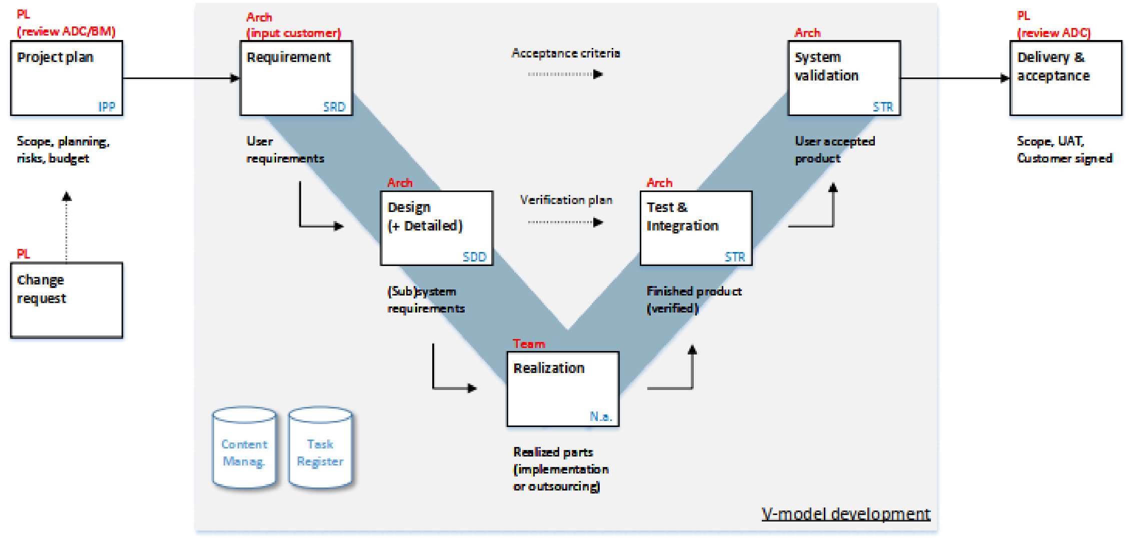

Figure 6 V model

5 Building block view

This chapter takes a closer look at the diagrams presented in Chapter 3. The idea of a building block view is to zoom in on the system step by step. This is done through levels. Level 0 is for this project is Figure 5 where the STM32H7 dual-core is a black box. In level 1, this STM32H7 dual-core becomes a white box that again consists of multiple black boxes. In level 2, the black boxes from level 1 become white boxes with black boxes there. During each step, the black boxes describe the responsibility and tasks. This is a good way to dissect a system in a structured way. It is ideal to consult with the stakeholders on an abstract level without further details on how the implementation will take place. After this, modular software blocks can be created.

5.1 White box STM32H7 dual-core

Figure 7 shows level 1 and zooms in on the STM32H7 dual-core. It also shows which core the inputs/outputs of STM32H7 dual-core go to.

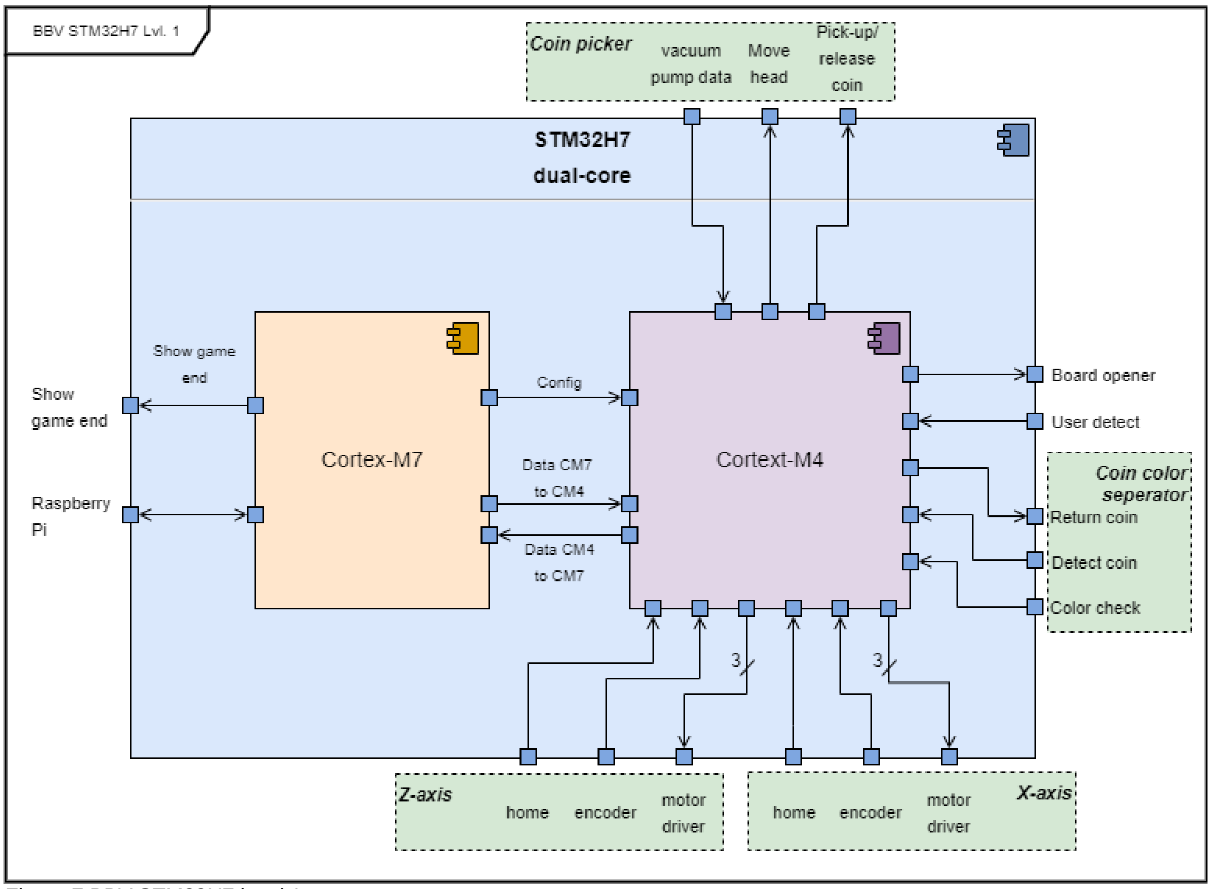

Figure 7 BBV STM32H7 level 1

Table 8 Description BBV STM32H7 level 1

Black box	Description
Cortex-M7	The Cortex-M7 core of the STM32H7 is used for machine handling. This core is responsible for the gameplay, communication with the Raspberry Pi, initializing the entire system and generating an output about the result of the game.
Cortex-M4	The Cortex-M4 core of the STM32H7 is used for real-time motion control. This core is responsible for handling all hardware components such as the Coin picker, Motors, Coin seporator, Board opener and User detect.

5.2 Level 2

This chapter turns the relevant level 1 blackboxes into white boxes and describes how the internal blocks are connected.

5.2.1 White box Cortex-M7

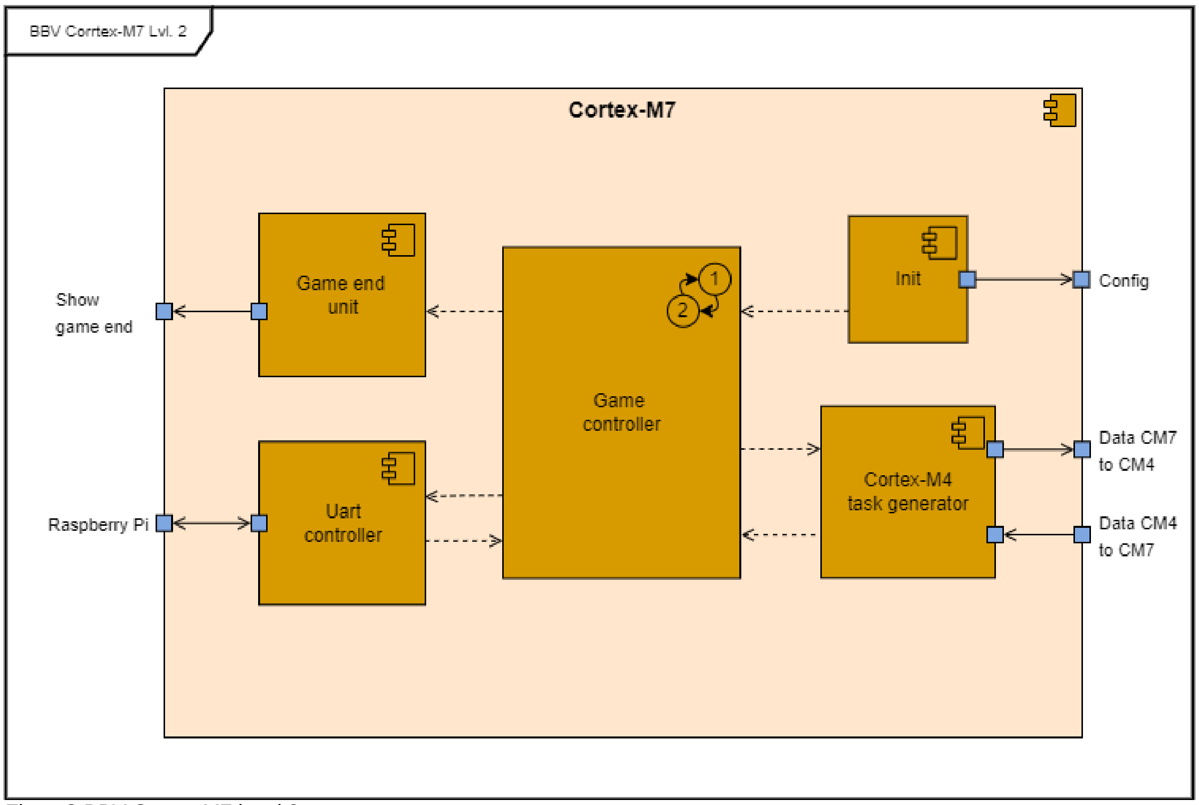

Figure8 BBV Cortex-M7 level 2

Table 9 Description BBV Cortex-M7 level 2

Black box	Description
Game controller	The Game controller is responsible for the gameplay by means of a state machine.
Init	Rotates once to initialize and boot the Cortex-M7.
UART controller	The UART controller implements all communication via UART.
Cortex-M4 task generator	The Cortex-M4 task generator implements communication with the Cortex-M4.
Game end unit	The Game end unit implements all communication to an output for the player.

5.2.2 White box Cortex-M4

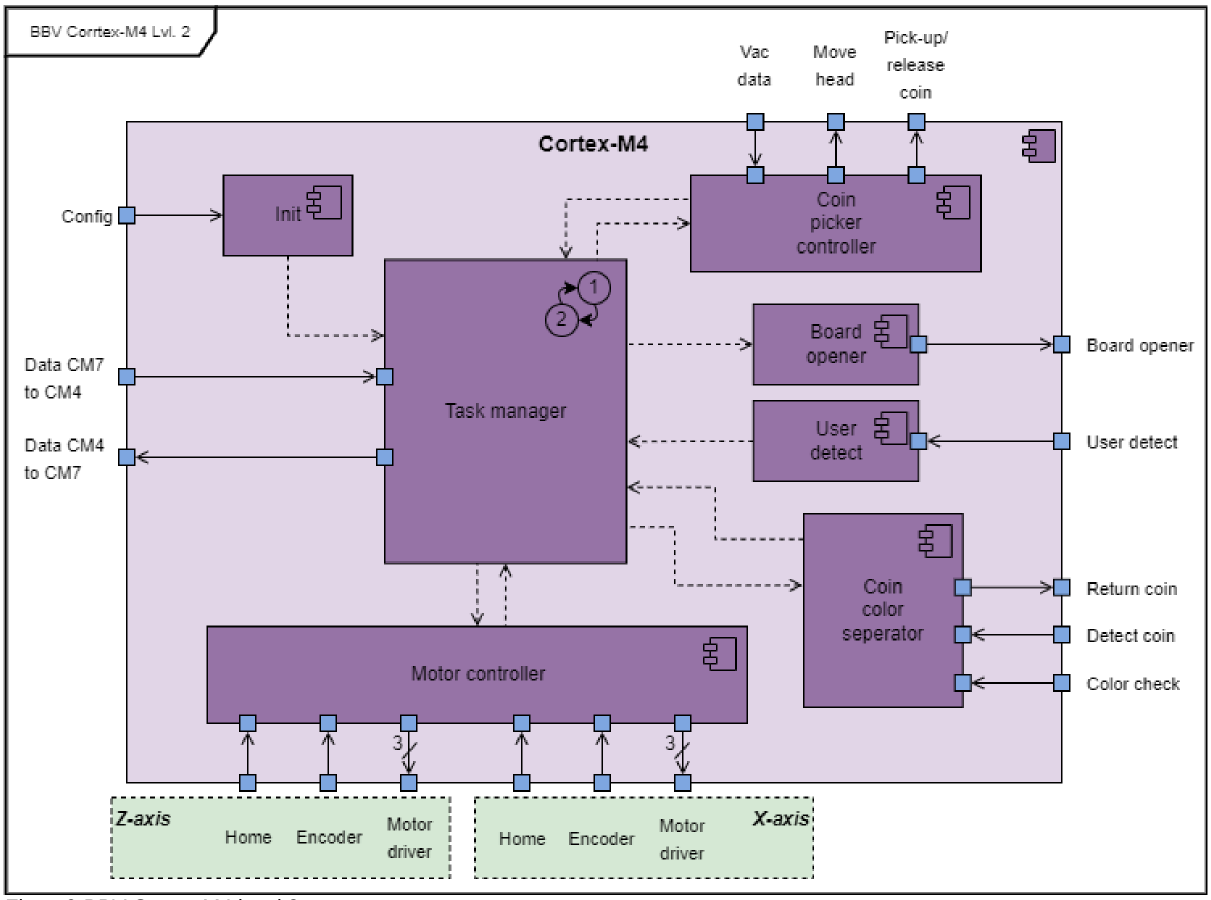

Figure9 BBV Cortex-M4 level 2

Table 10 Description BBV Cortex-M4 level 2

Black box	Description
Task manager	The Task manager is responsible for the progress of the tasks to be executed of the Cortex-M7 by means of a state machine. Byusing a "Task manager", the functionality of the other parts can be built modularly. After all, they do not have to know about each other's existence/status. This information is kept by the Task Manager.
Init	Rotated once to initialize and boot the Cortex-M4. In this phase, the sensors are tested for presence and the motors perform a "homing" protocol.
Motor controller	The Motor controller implements all communication with the motor drivers and the PID controller.
Coin color Separator	The Coin color separator implements all functions for handling the separation of the chips.
User detect	The User detect implements all functions for handling the sensorand for the input of the player.
Board opener	The Board opener implements all the functions for handling the open of the 4-on-1 row board when the game is finished.
Coin picker controller	The Coin picker controller implements all functions for handling the picking up and release of a chip.

5.3 Level 3

This chapter turns the relevant level 2 blackboxes into white boxes and describes how the internal blocks are connected.

5.3.1 White box Motor controller

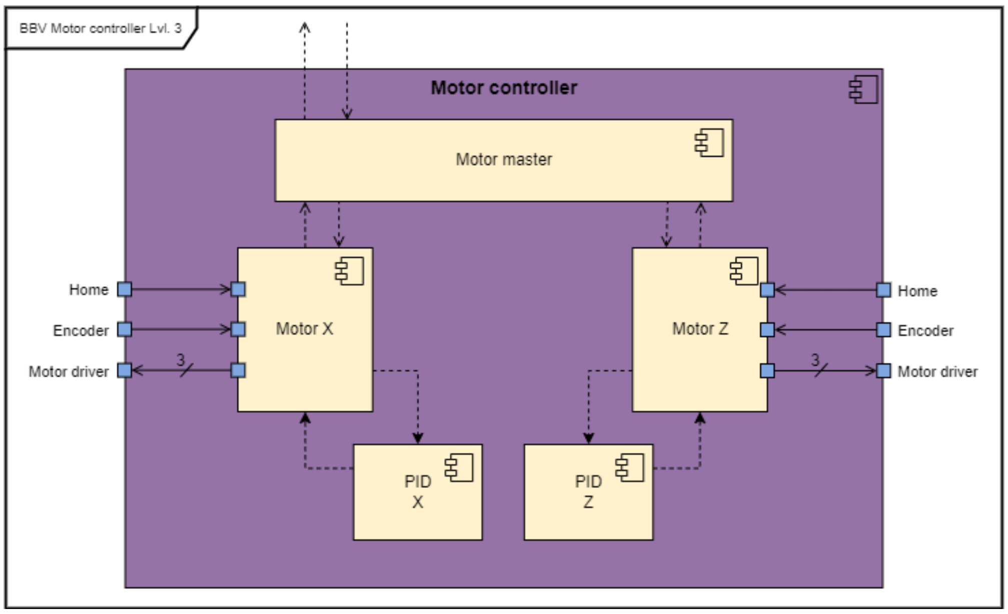

Figure10 BBV Motor controller level 3

Table 11 Description BBV Motor controller level 3

Black box	Description
Engine master	The engine master is responsible for controlling both engines.
Engine X	Engine X controls communication with the X-axis motor.
PID X	PID X creates the control loop with feedback for engine X.
Motor Z	Motor Z controls communication with the Z-axis motor.
PID Z	PID Z creates the control loop with feedback for engine Z.

5.3.2 White box Coin color separator

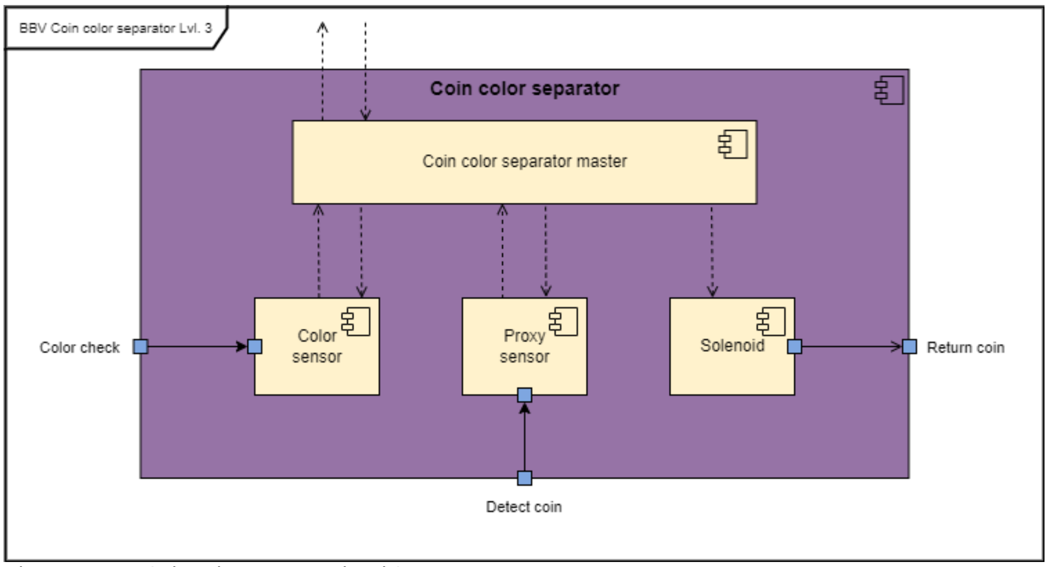

Figure11 BBV Coin color seperator level 3

Table 12 Description BBV Coin color seperator level 3

Black box	Description
Coin color separator master	The Coin color separator master is responsiblefor receiving the sensor data and controlling the solenoid.
Color sensor	Requests the data on the color of a sheet.
Proxy sensor	Receives data whether a sheet is present.
Solenoid	Is responsible for activating the flipper.

5.3.3 White box Coin picker

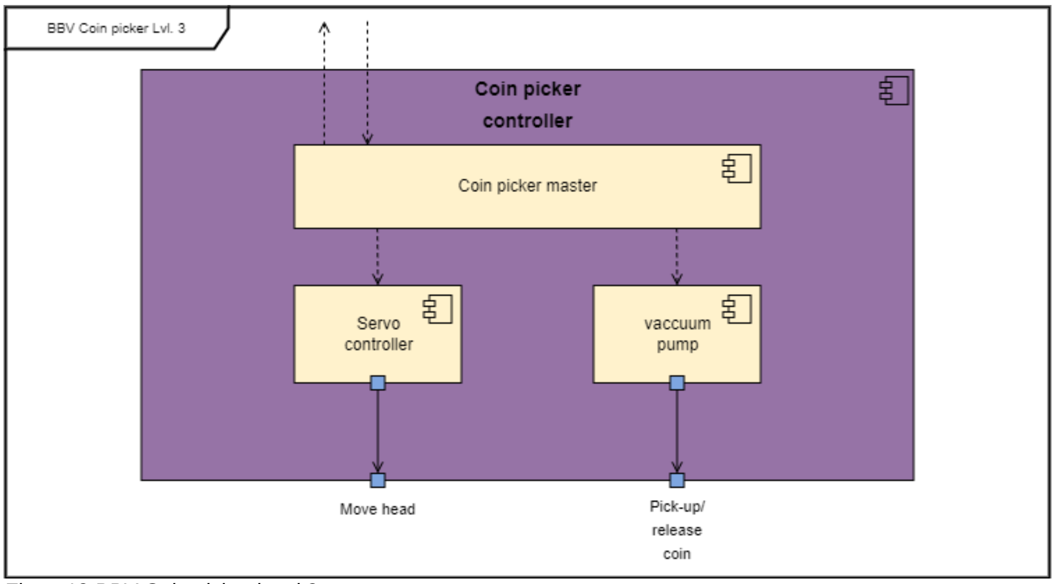

Figure12 BBV Coin picker level 3

Table 13 Description BBV Coin picker level 3

Black box	Description
Coin picker master	The Coin picker master is responsible for controlling the servo and the vaccuum pump of the vaccuum gripper.
Servo controller	Controls the control of the servo motor.
Vaccuum pump	Controls the control of the vaccuum gripper to pick up the chips.

6 Runtime view

The Runtime view describes the concrete behavior and interactions of the building blocks of the system. This is done through a high-level state machine (Figure...) and scenarios that are shown in the subchapters.

6.1 High level overview

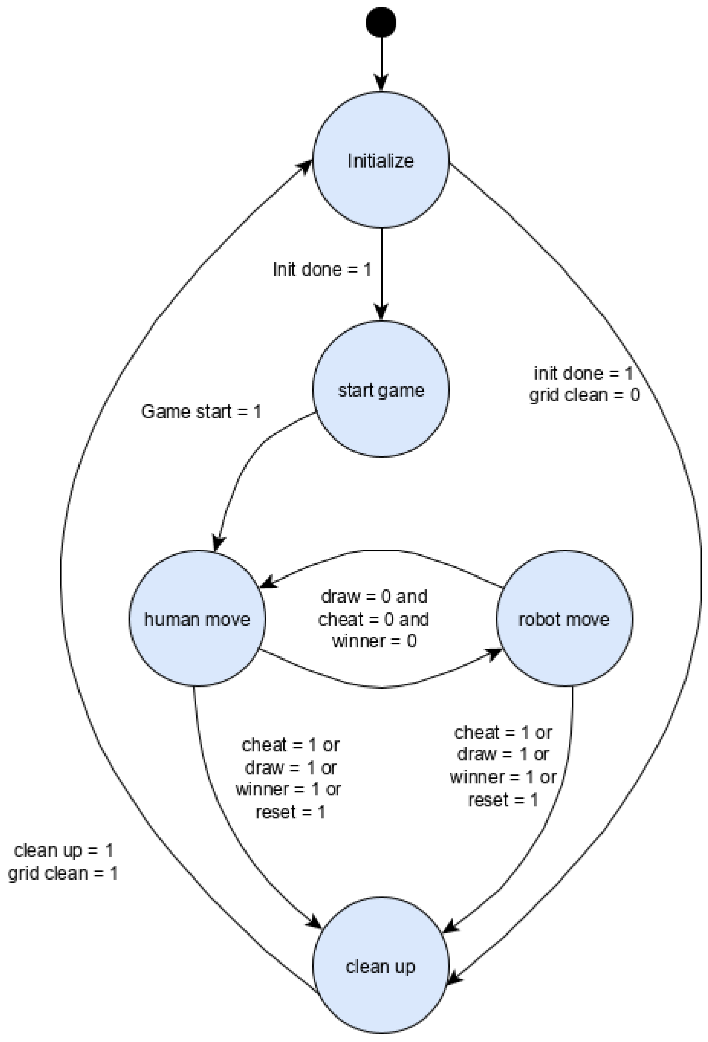

Figure 13 State machine Cortex-M7

Figure 13 shows how the 4-on-1 row acts at the highest level. This Statemachine runs on the Cortext-M7 within the game controller and is therefore the main flow of the 4-on-1 row. The first state is the "initialize" state. In this state, the entire system is booted and initialized. Once this state is ready, the system enters the "start game" state. In this state, the system waits for the game to begin. When the game is started, it is the player's turn to be the first to do his/her chip in the 4-on-1 row game and the system is in the "human move" state. Each turn, the robot checks whether there is a winner, cheated or draw. If this is not the case, the turn goes to the robot and the system goes to the state "robot move". In this state, the robot performs the calculated move. As soon as the game is finished, the "clean up" state starts and all chips are cleared. The red chips go to the robot and the yellow ones to the player's box.

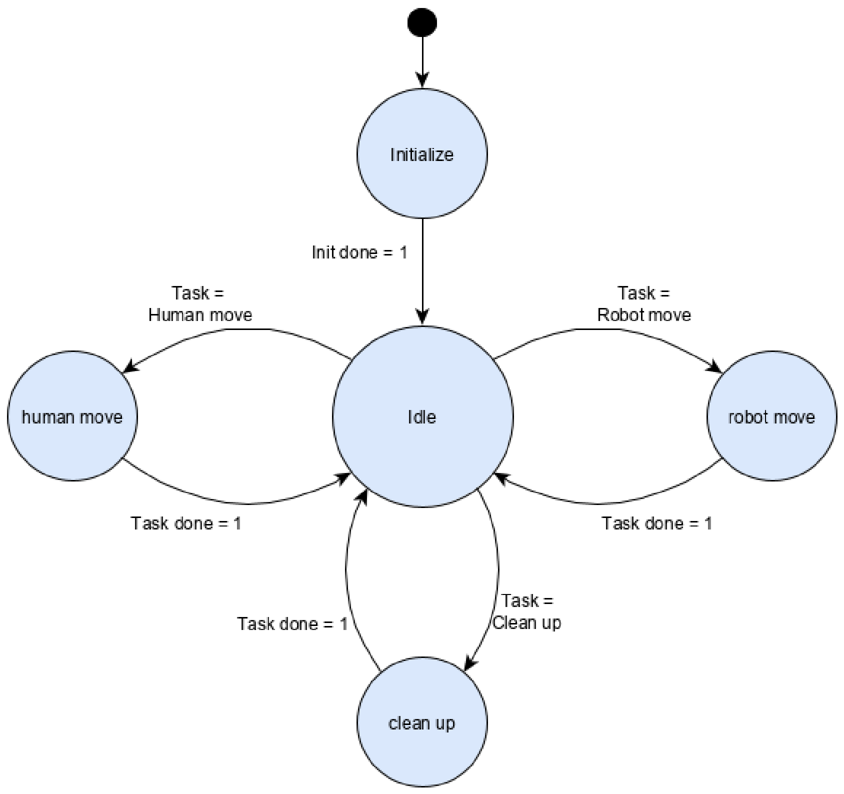

Figure 14 State machine Cortex-M4

As shown in Figure9 there is also a state machine on the Cortex-M4. This state machine is shown in Figure 14 and runs on the Task manager module. The first state is the "Initialize" state. In this state, the hardware components are configured and configured. After that, the Task manager has an "Idle" state. In this state, they wait until the Task manager gets a task from the Cortex-M7. As soon as a task comes in, the next state is determined on the basis of the task. This can be the "human move", "robot move" or "clean-up" state. Each of these states drives the hardware to complete the task. When the task is complete, the Task manager returns to the Idle state.

In the following subchapters, the states "robotmove", "human move" and "clean-up" are shown in a sequence diagram. A sequence diagram shows how the different building blocks relate to each other by time and what kind of signal they send.

6.2 Robot move

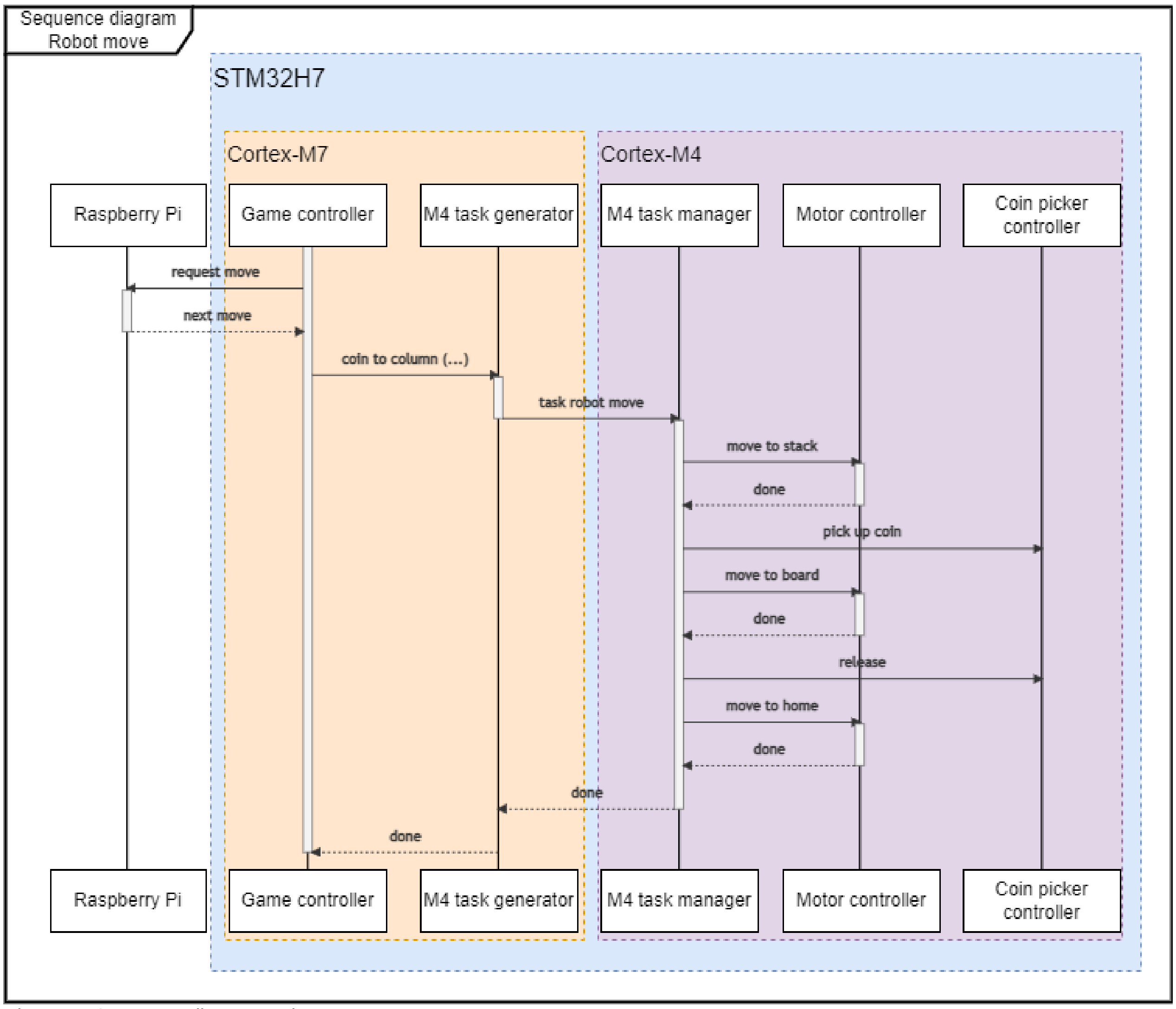

Figure 15 Sequence diagram Robot move

The "Robot move" in Figure 15 requesting the next move from the Raspberry Pi. The Raspberry Pi returns the next move, after which the Game controller tells the M4 task generator what kind of assignment needs to be made. Once the command is created, the M4 task generator sends the command to the M4 task manager on the Cortex-M4. To put a chip in the board, the robot must first go to the place where the chips are stored. As soon as the robot has arrived at the storage, a sheet must be picked up. Now the robot can move to the board and then release the chip. Finally, the robot has to go back to its "home" position. When the move is made, the M4 task manager passes on to the M4 task generator which in turn informs the Game controller that the turn is complete. The Game controller goes to the next state.

6.3 Human move

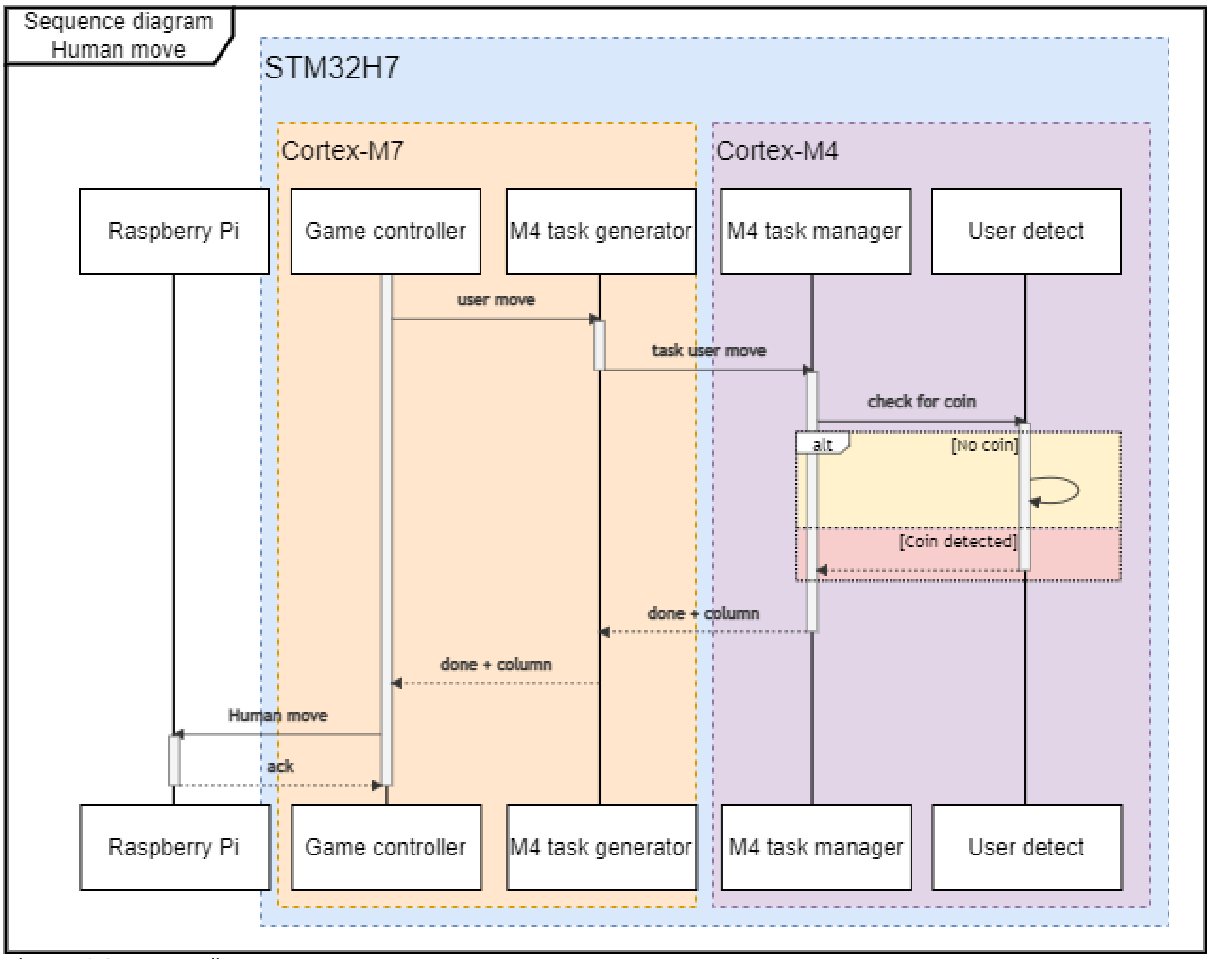

Figure 16 Sequence diagram Human move

The "Human move" in Figure 16 begins with the notification that the player is on the move to the M4 task generator. The M4 task generator generates the command and sends it to the M4 task manager on the Cortex-M4. The M4 task manager in turn instructs the User detect that a sheet is expected. If no chip is played by the player, the system will continue to wait for a move. As soon as a chip is played by the player, it is linked back to the M4 task manager. This informs the M4 task generator that a move has been made and in which column the sheet has been put. The M4 task generator passes this on to the Game controller after which it passes the move to the Raspberry Pi. The Raspberry Pi confirms getting the move and sends back whether the game is finished or can continue.

6.4 Clean up

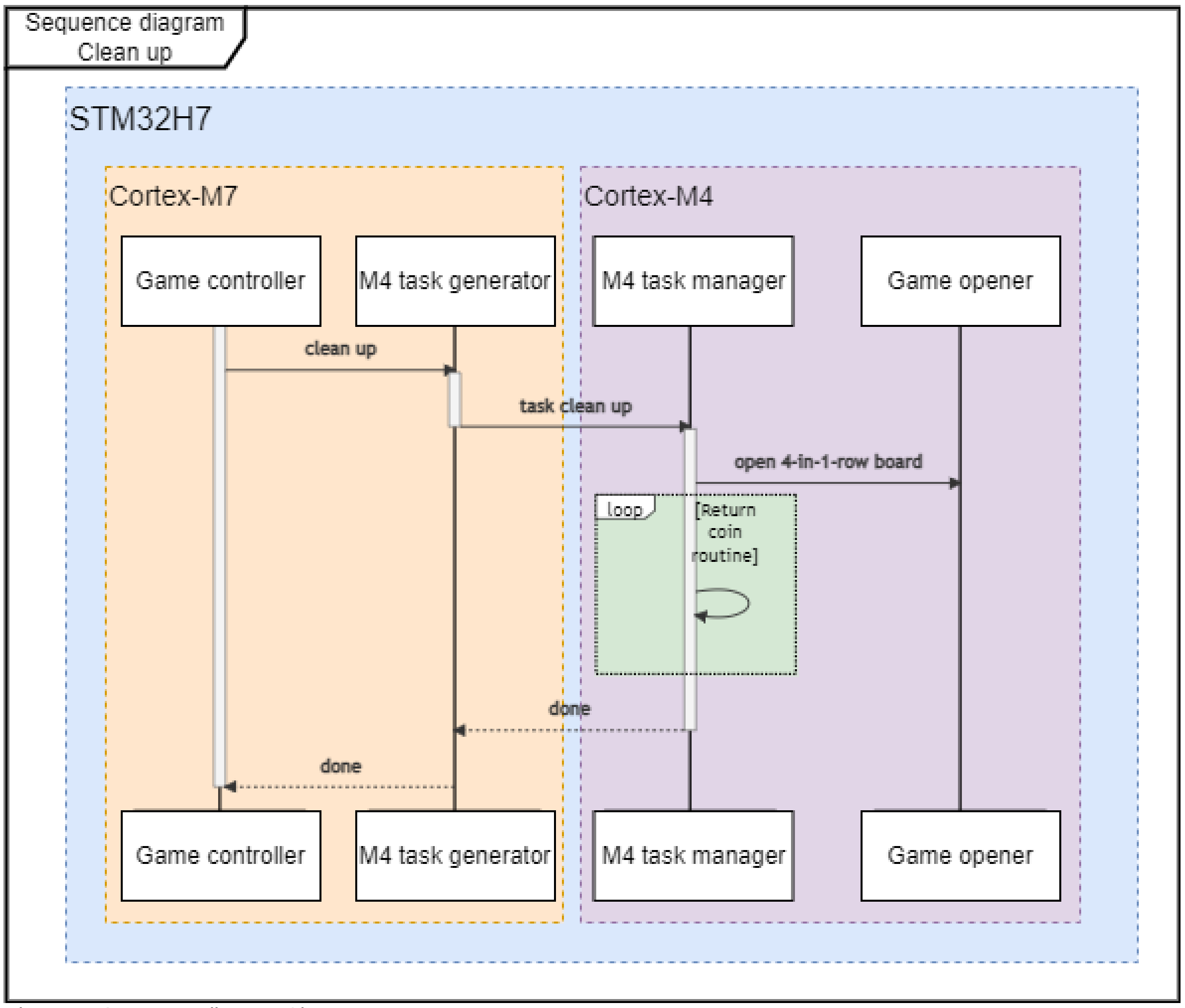

Figure 17 Sequence diagram Clean-up

The "Clean up" inFigure 17 begins by notifying that the game is over and the cleanup routine can begin on the M4 task generator. The M4 task generator generates the command and sends it to the M4 task manager on the Cortex-M4. The M4 task manager opens the 4-on-1 row board and starts the "Return coin routine" this loop is performed until all chips are cleared. Once that is done, the M4 task manager informs the M4 task generator that the game has been cleaned up. The M4 task generator passes this on to the Game controller after which the game can start again.

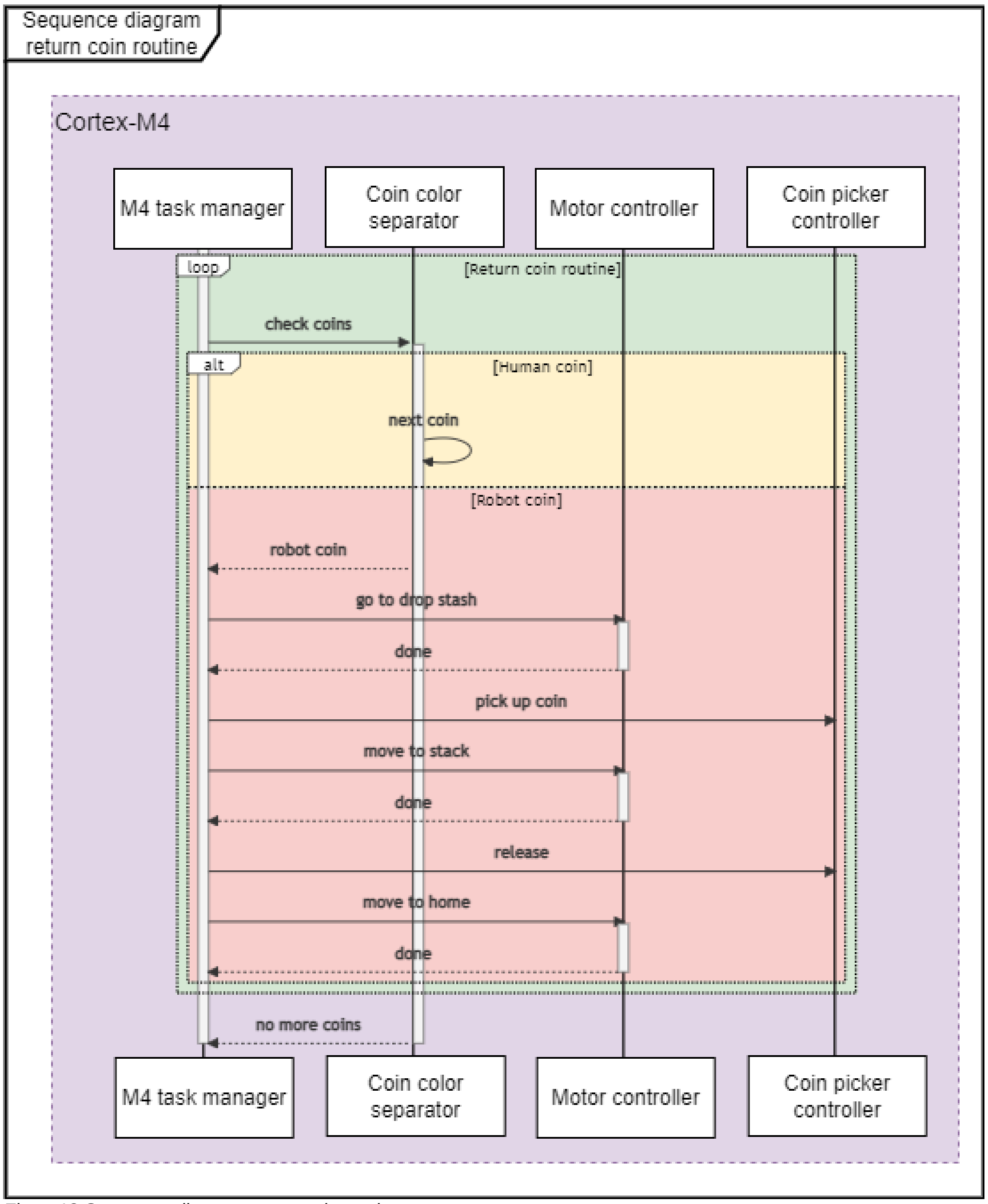

Figure18 Sequence diagram return coin routine

The "Return coin routine"Figure18Clean up" from Figure 17 and describes how the M4 task manager ensures that all chips are cleaned up. First, the M4 task manager sends to the Coin color separator that chips are expected. As soon as a sheet is detected, the Coin color separator will check what color this sheet has. If the chip belongs to the player, it is shot by a flipper to the player's box. If the sheet is of the robot, the M4 task manager is informed that it is a sheet of the robot. The M4 task manager directs the Motor controller to go to the sheet. The chip is then picked up and the robot moves to its own chips storage. Here the chip is released again and the robot moves to its "home" position. This loop is carried out until all chips have been cleared.

7 Deployment View

7.1 NUCLEO-H755ZI-Q

To implement the software architecture, a Nucleo-H755ZI-Q is present. This is a development board with the STM32H7 chip taps. Furthermore, dit board has all the necessary pin-outs and hardware to facilitate the implementation of the software blocks. The plan is to eventually switch to an ALTEN-designed PCB for the STM32H7 chip with all the necessary connections. However, this is outside the scope of this project and therefore the Nucleo-H755ZI-Q is being used (Figure 19).

Figure 4. Nucleo-144 board top layout

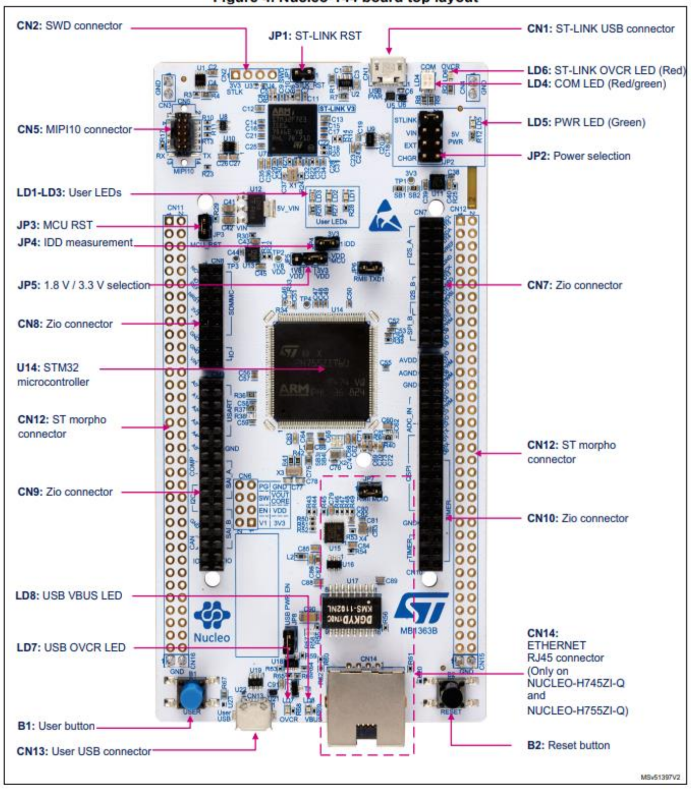

Figure 19 NUCLEO-H755ZI-Q

7.2 Pin-out NUCLEO-H755ZI-Q

To realize all the functions of the 4-on-1 row, the pin-out in Figure20 composed. This pin-out takes into account the function of each pin and what the pin should facilitate. Table Table14 performed by a pin, what kind of signal is on the pin and to which pin the function is connected. As can be seen, pins have also been set for ethernet. Ethernet is a feature that may be used in the future but falls outside the scope of this project. But it is very important to assign the function to the pins to prevent those pins from being used for other purposes.

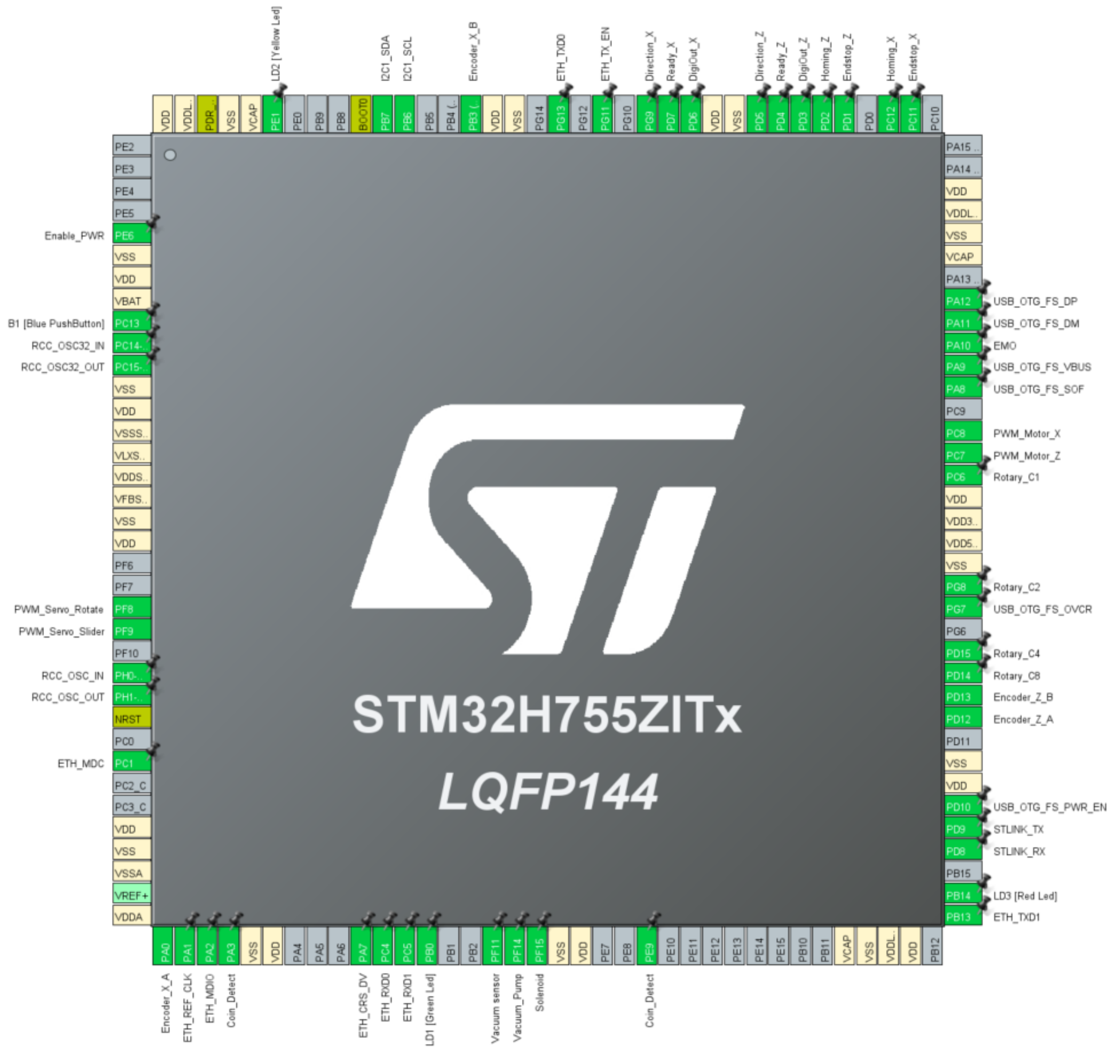

Figure20 pin-out STM32H755ZITx

Table14 pin-out table

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 layout

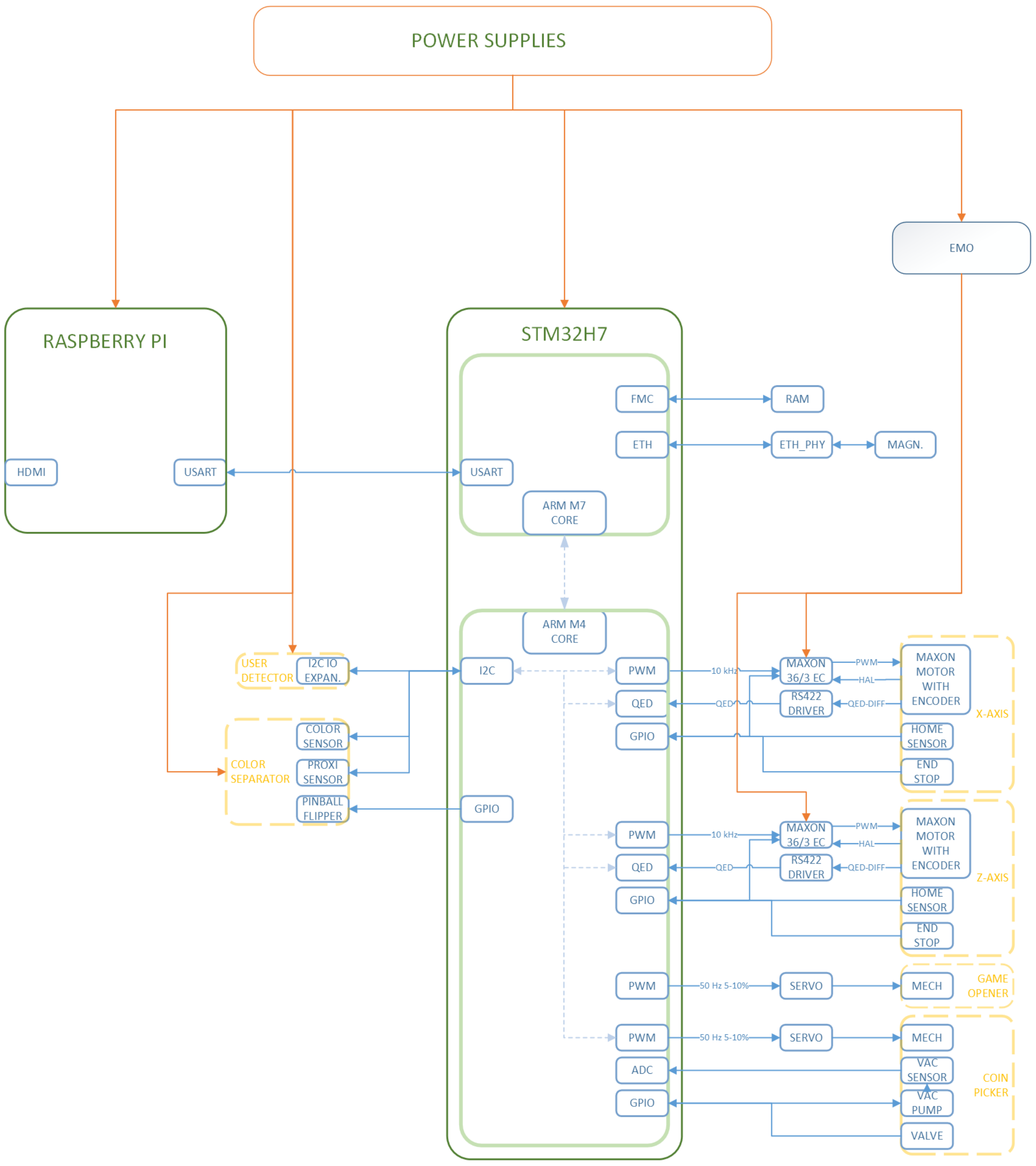

Figure 21 Hardware layout 4-on-1 row with STM32H7

7.4 Master-Minion ratio

To indicate how the processors compare in a Master-Minion ratio, Figure 22 been drawn up. This shows that the Cortex-M7 is the top-master of the system. On the Cortex-M7 the game flows and that determines the entire game. The Raspberry Pi and Cortex-M4 are a minion of the Cortex-M7 and perform tasks when the Cortex-M7 demands it. The Cortex-M4 is also a master for the hardware. All hardware systems of the 4-on-1 row are a minion of the Cortext-M4 and perform tasks as the Cortex-M4 demands.

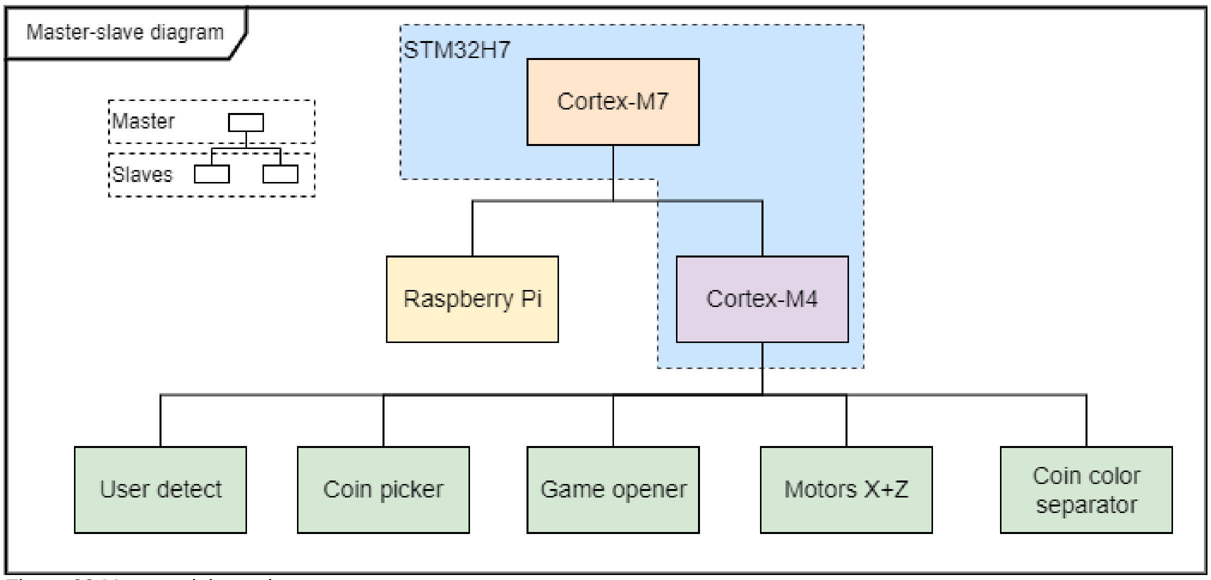

Figure 22 Master-minion ratio

8 Modular software implementation

(Chapter 5) shows which software blocks must be present to implement a fully working 4-on-1 row robot. It doesn't show how these are related from the top-level to the hardware. This is further explained in this chapter using diagrams.

8.1 Cortex-M7 core

Figure8 all software blocks of the Cortex-M7 core. To show how the blocks relate to the hardware, the diagram in Figure 23 drawn up.

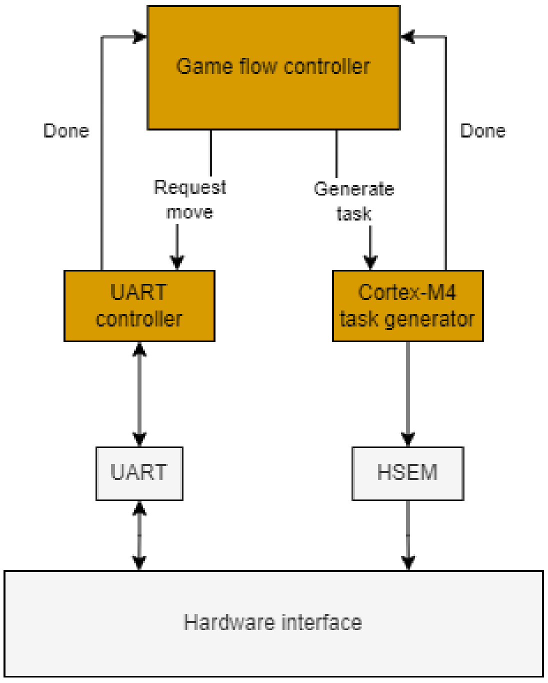

Figure 23 IBD game controller

8.2 Cortex-M4 core

Figure9 all software blocks of the Cortex-M4 core. To show how the blocks relate to the hardware, all diagrams in the subchapters have been drawn up.

8.2.1 Motor controller

Figure 24 diagram of the engines. The blocks for Motor X are the same as for Motor Z. These blocks are shown in green.

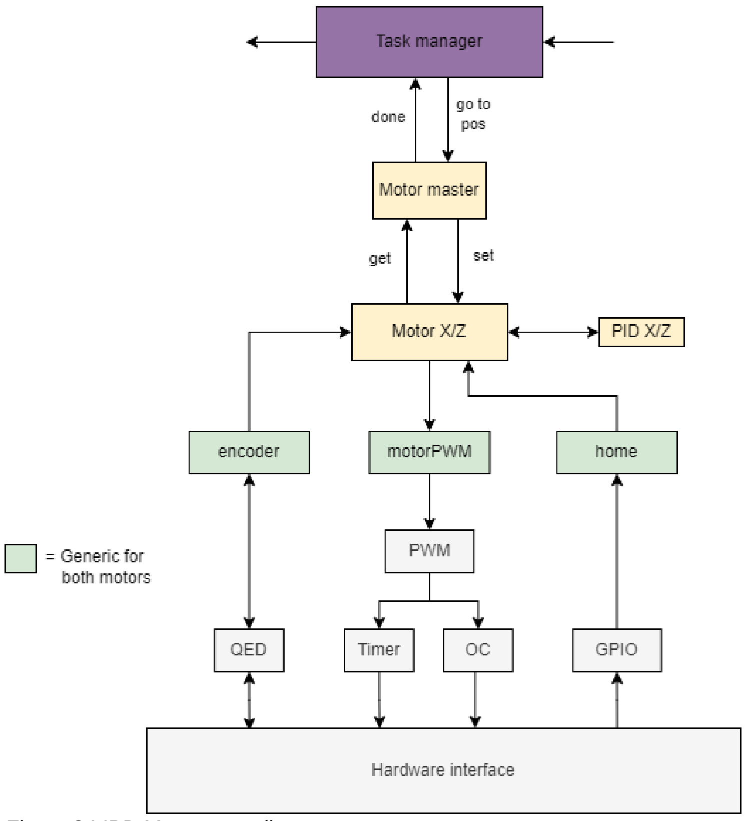

Figure 24 IBD Motor controller

8.2.2 Coin color separator

Figure 25 shows the diagram for the Coin color separator. The i2c device block is ageneric block that handles all information for an i2c sensor.

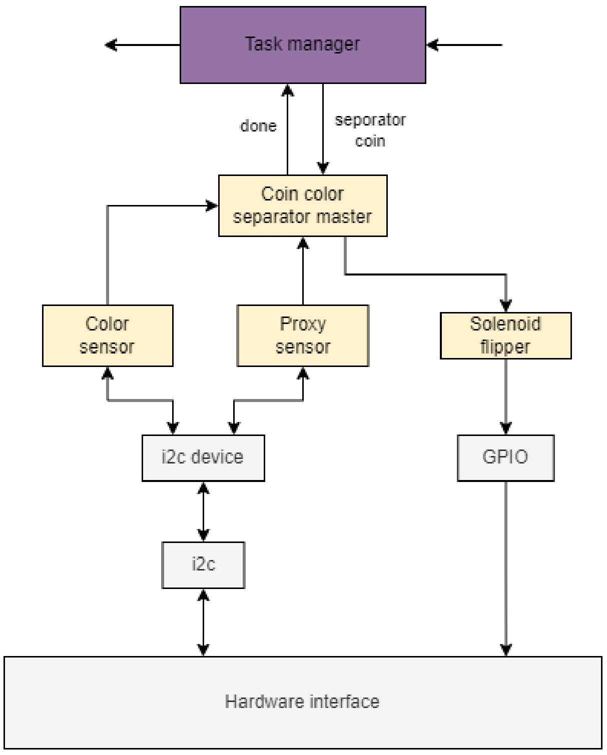

Figure 25 IBD Coin color separator

8.2.3 Coin picker

Figure 26 shows the diagram for the Coin picker.

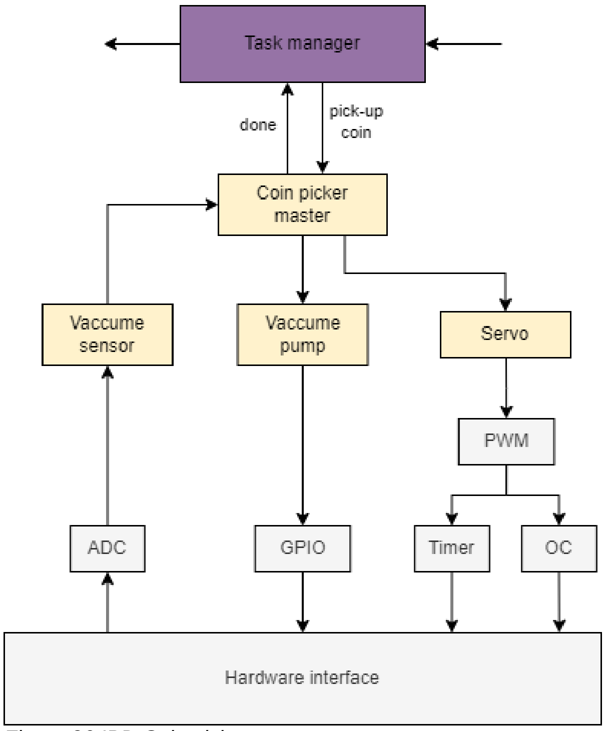

Figure 26 IBD Coin picker

8.2.4 Board opener

Figure 27 shows the diagram for the Board opener.

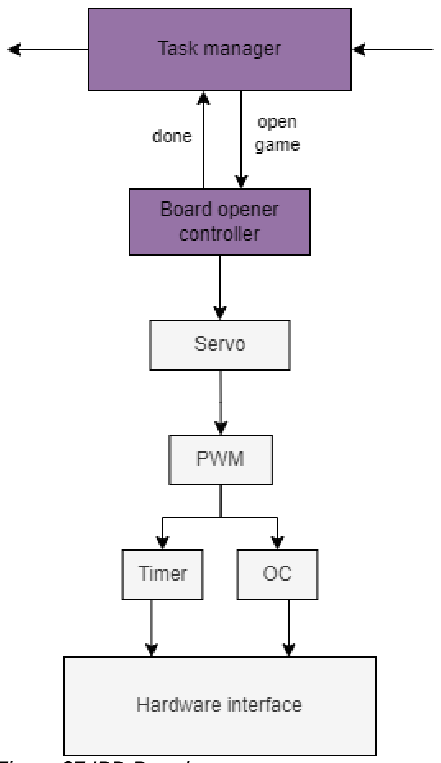

Figure 27 IBD Board opener

8.2.5 User detect

Figure 28 shows the diagram for the User detect.

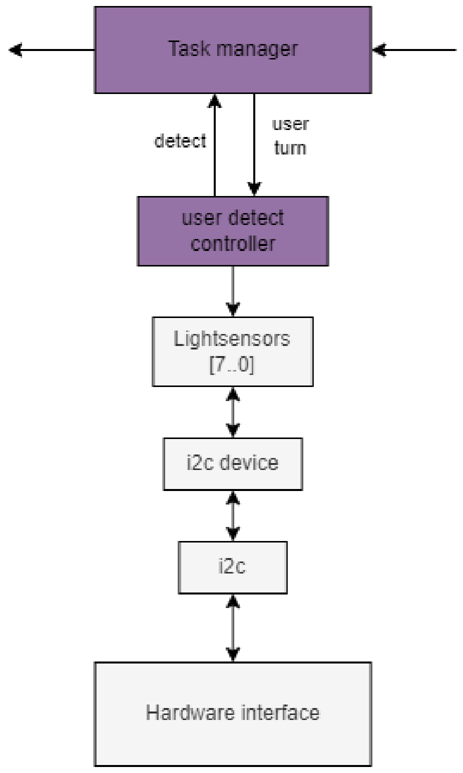

Figure 28 IBD User detect

9 Error handling

According to the architecture described in previous chapters, the system can run smoothly. Nevertheless, there is a chance that an error will occur. If the motors do not properly perform the "homing" phase, one of the sensors stops responding or the motors fail when a chip is brought to the board. As soon as the system gives an error message, it must be handled. This means turning off hardware for security and giving an indication to the Operator that the system has detected an error. Figure 29 shows how an error is handled within the main states of the Game flow controller.

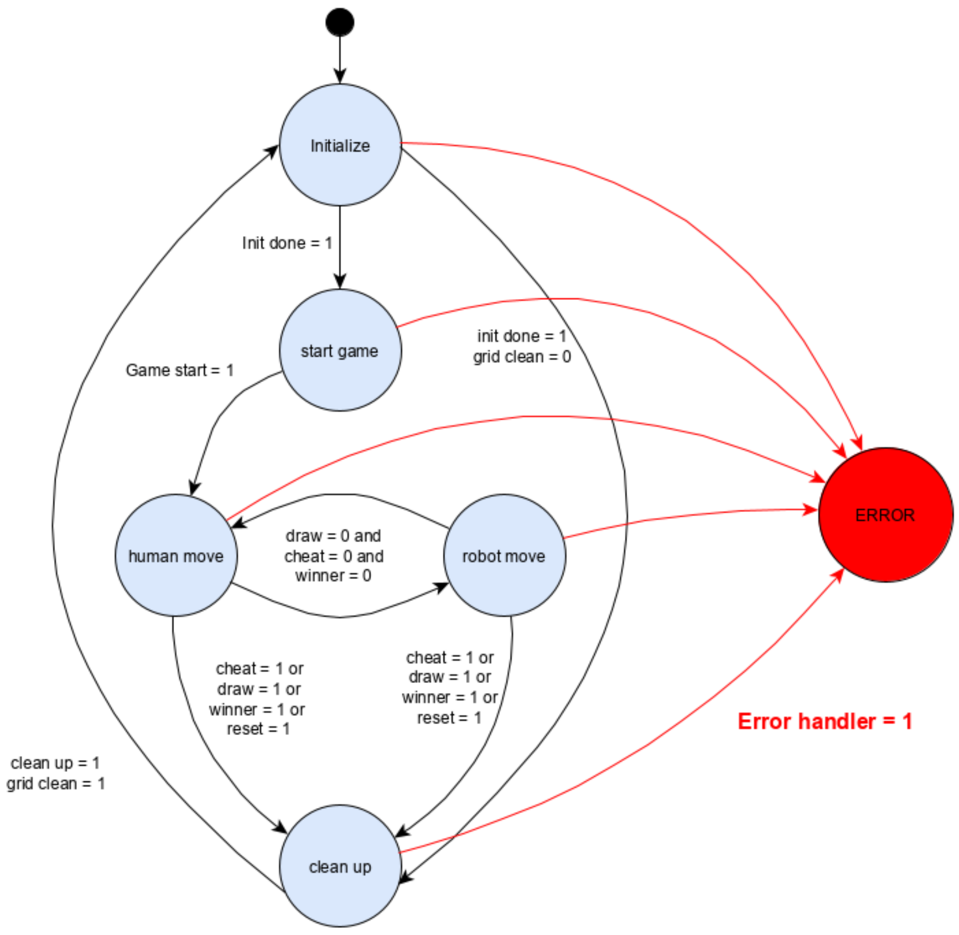

Figure 29 State machine with error handler

In the "ERROR" state, all engines and other hardware components are turned off within the two cores of the STM32H7 and the system remains in this state until it is reset.