Software Architecture Documentation -

4 in 1 row robot

Version 3
Date: 4/13/2022

ALTEN
Pascal Fatz

Confidential
Design

Version history

Version	Date	Status	Writer	Remark
1	3/31/2022	Design	Pascal Fatz	Initial draft to chapter 8
2	4/13/2022	Design	Pascal Fatz	Process feedback Jeedella, Aniel and Berend
3	5/3/2022	Completion	Pascal Fatz	Chapter 8 and beyond

Acronyms and abbreviations

Term	Explanation
BSP	Board Support Package

Referenced Documents

ID	Reference	Title	Date	Writer
D1	SRD_Pascal_Faatz_V3.docx	SRD	3/16/2022	Pascal Fatz

Table of contents

List of figures

List of tables

1 Introduction and goals

ALTEN has developed a robot in-house that can play 4-on-1-row against a human opponent using an algorithm. Using industrial components, the 4-in-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 will be 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-in-1-row robot will devise and execute a move. For this purpose, the system keeps track of which moves have been played by its opponent. When the move has been determined by the robot, the combination of the XZ platform with rotating vacuum gripper will pick up a stone and insert it into the game at the chosen spot. Once the game is over or reset, the Match 4 robot will collect all the chips and sort them by color to be ready for the next round.

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

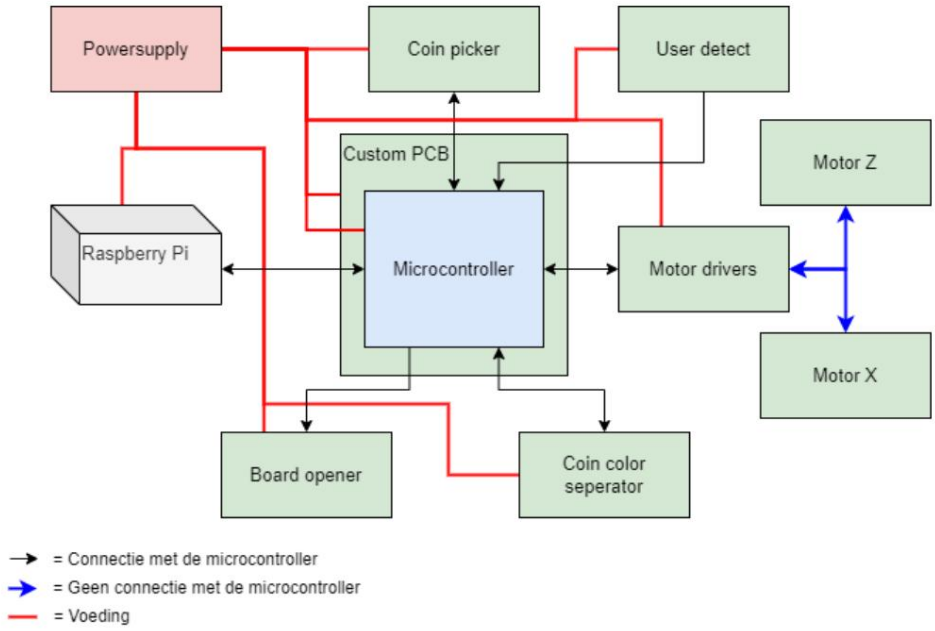

Figure 1 Block diagram 4-in-1 row

1.1 Requirements overview

Due to the unclear architecture, it is very difficult to expand or upgrade the operating system. Because the 4-in-1-row robot is also exhibited at trade fairs and demonstrates what ALTEN has to offer, it is essential that the robot is easy to maintain and can easily be upgraded. This requires a complete redesign of the operating system to solve the cluttered and unclear architecture and to enable upgrades.

One of the main requirements is therefore to introduce a structured architecture and modular implementation. It is also important to draw up state and flow diagrams to visually represent the operation of the system. A BSP (Board Support Package) must also be made.

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 are also tested at a later stage and checked whether the system meets these requirements.

Table 1 Quality goals

Priority 1	Quality goal	Concrete scenario
	Easy to understand (Structured architecture)	People who are going to work on the 4-in-1-row robot must immediately understand how the hardware and software are connected by reading the architecture.
2	Maintenance and upgrades (Modular construction of software blocks)	People who are going to make an upgrade or modification to the 4-to-1 row must be able to modify or replace software blocks without affecting other parts of the system.
3	Robust	The 4-in-1 row must be reliable and work under 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 shows the distribution of stakeholders for this project.

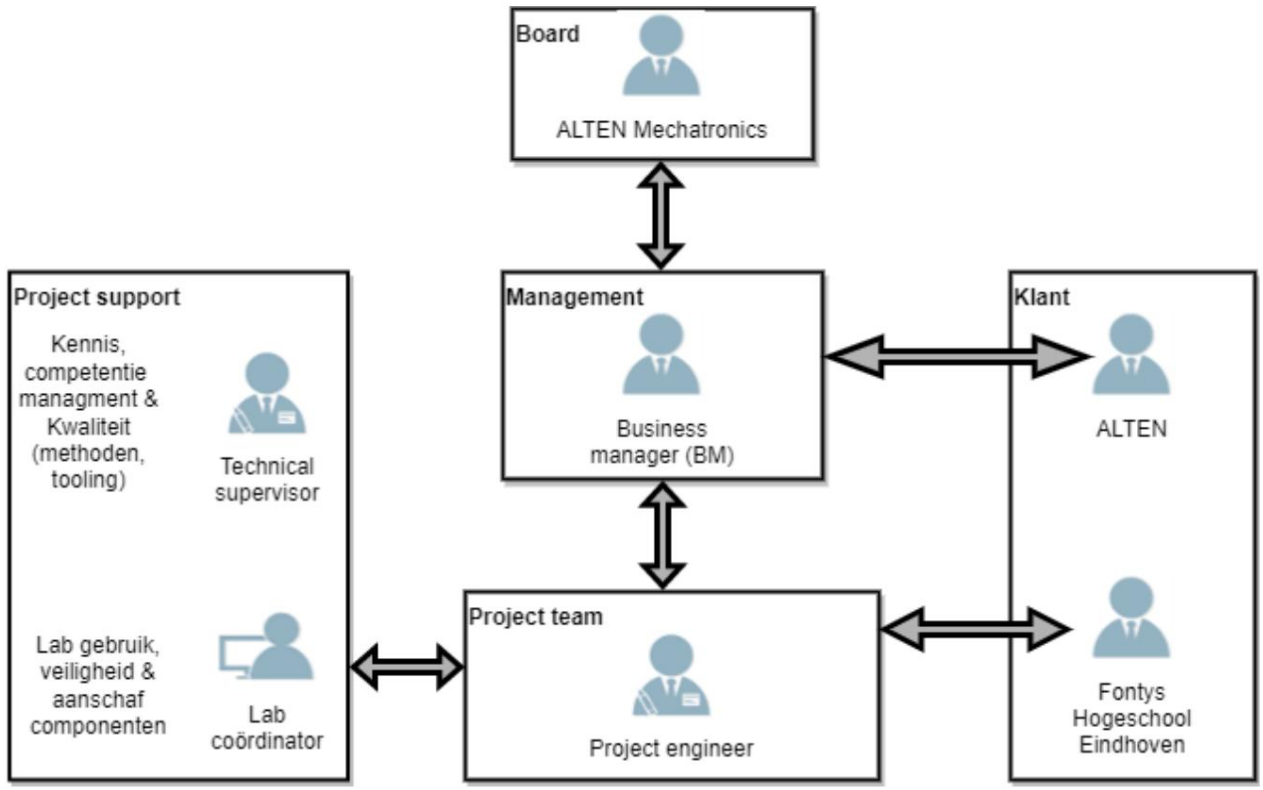

Figure 2 Project organization

Table 2 Stakeholders

Name	Role	Responsibility
Pascal Fatz	Project engineer	Carrying out the project for the relevant customers and board/management.
Aniel Shri	Technical supervisor	Provide technical support to the project team within the project. This involves challenging the team to come up with a good solution and implementation.
Gijs Haan	Business manager	Supporting the project team within the project personal development. Furthermore, the business manager.
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 of the project.
Jeroen Wilbers	lab coordinator	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	Assess the requirements as an end customer 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 Architecture constraints

Table 3 Architecture

constraints **Architecture constraint**	**Description** A microcontroller
is used The choice of the microcontroller is fixed. With the	STM32H755 dual-core nucleo-144 board. microcontroller must therefore be worked.
The hardware of the 4-in-1 row is fixed. The hardware that is now available is used to write the	software.
The game progression of the 4-on-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 representation of the people who work with the system and what they have 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.

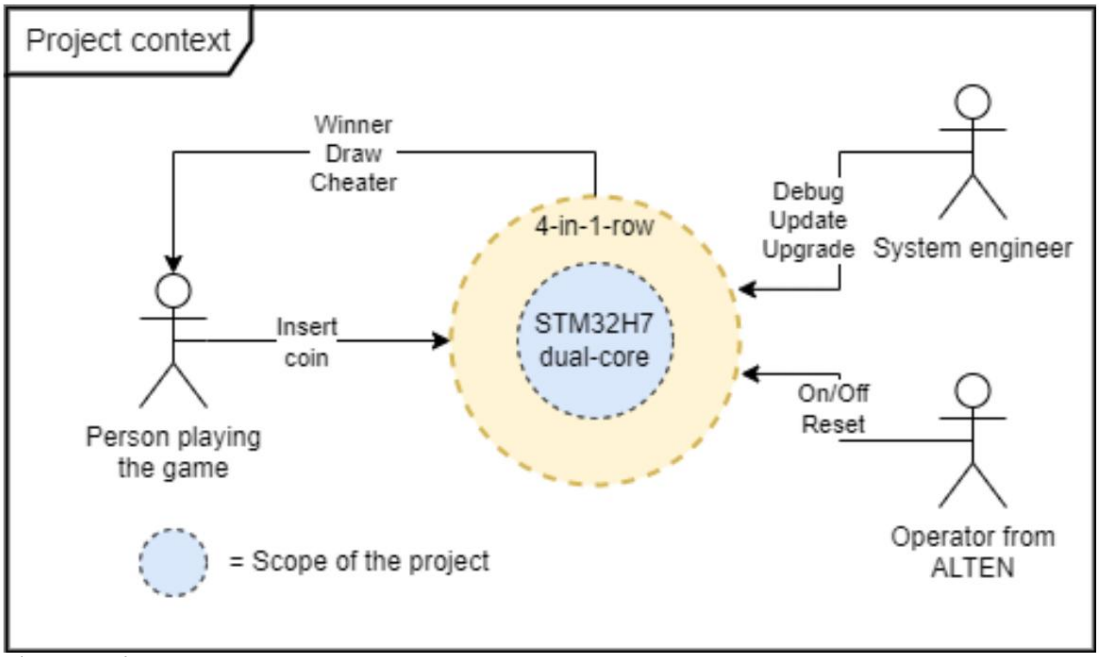

Figure 3 Project context

Table 4 Description Project context

Users	Description
Person playing 4-in-1	The person who plays 4-in-1 has to think carefully when it is his turn to put his chip to defeat the robot. When a move is known, the chip is placed in the 4-in-1 board. Furthermore, at the end of the game, the person can see what the result is.
System engineer	The system engineer can debug the 4-in-1 array if an error occurs. Furthermore, the engineer can implement an update/upgrade.
Operator from ALTEN operating the 4-in-1 row	This person operates the 4-on-1 row at fairs or other occasions. This person can turn the 4-in-1 row on/off or reset it.

3.1 System context

Figure 3 gives a global overview of the people involved in the system. Figure 4 zooms in further on this and provides a visual representation of the subparts of the 4-to-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 in the system in general terms.

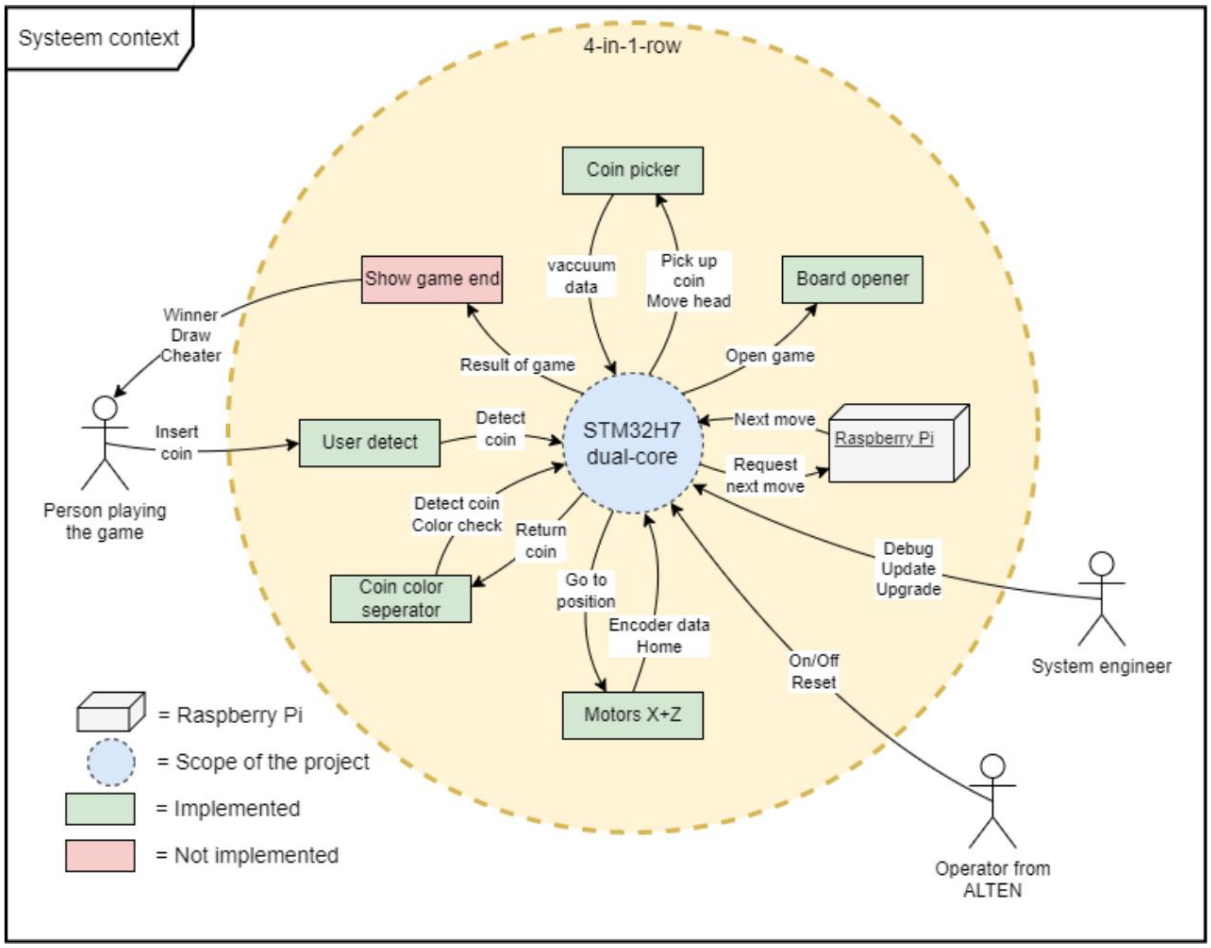

Figure 4 System context

Table 5 Description System context

Hardware block	Function
RaspberryPi	The Raspberry Pi runs an algorithm that determines the robot's next steps.
Engines X+Z	The motors allow the robot to move its vacuum gripper over an X and Z axis of the 4-in-1 row board.
Coin color separator	When the game is over, the coin color separator ensures that each chip is recognized by color. If the chip has the player's color, the chip is shot to the player's bin. If the chip has the color of the robot, the chip is placed in its own stack.
User detect	When the player puts a chip in the game, this is detected. Each column of the 4-in-1 board contains a detection point.
coin picker	The coin picker is the arm of the robot. This is moved by the engines, but can also move itself to put a chip in the 4-in-1-row board. Picking up or releasing a chip is done by means of a vacuum gripper.
Board opener	The board opener causes all chips to fall out of the 4-in-1 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 are shown by what kind of protocol or input/output signal they are connected. This is of interest to the stakeholders who get to work with the hardware or architecture.

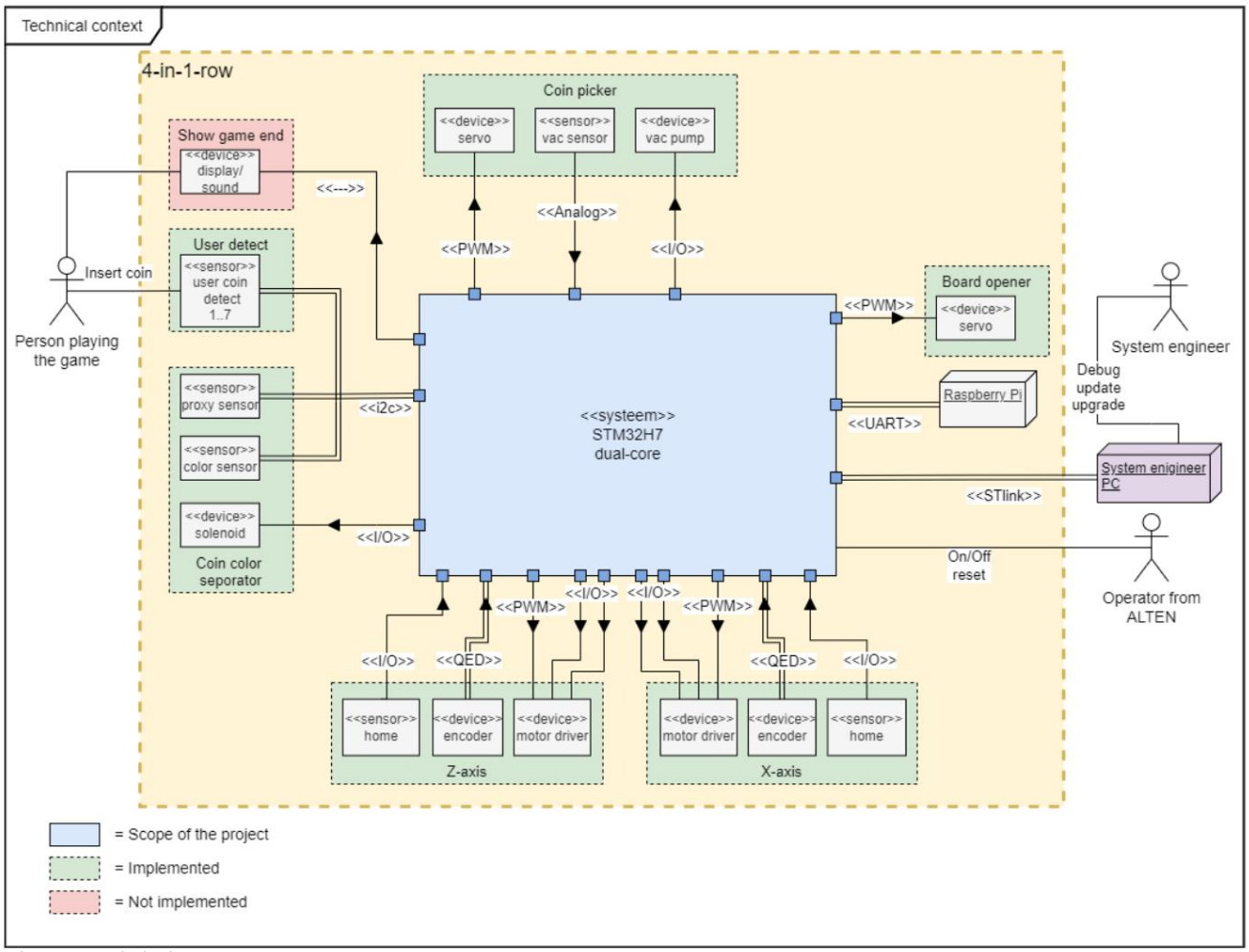

Figure 5 Technical context

Table 6 Description Technical context

Hardware block	Sub blocks	Description	Input to STM32H7	Number of lines	output from STM32H7	Number of lines
raspberry Pi	-	The Raspberry Pi communicates with the STM32H7 through 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 is used to determine the position of the motor. 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 motor has reached the home point.	IO	1	-	-
Motor/ 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.	IO	1	-	-
Coin color separator	proxy sensor	The proxy sensor checks whether a chip is present in the distribution system.	i2c (SDA, SCL)	2	-	-
	Colour sensor	The color sensor checks which color the chip in the distribution system has.	i2c (SDA, SCL)	2	-	-
	Solenoid	When a chip has the player's color, the solenoid shoots the chip to the player's bin by means of a flipper.	-	-	IO	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 vacuum gripper can be moved.	-	-	PWM	1
	Vacuum pump	The vacuum gripper can suck and release a chip.	-	-	IO	1
	Vac sensor		analog	1	-	-
Board opener	-	The game opener uses a servo to ensure that all chips fall out of the 4-in-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	This document is used to set up a structured architecture for the software of the 4-in-1 row. The chapters and diagrams in this document make it clear how the 4-to-1 row behaves, the software blocks are connected and the software communicates with the hardware. Furthermore, a top-down approach is used by starting with the requirements. From the requirements, the system is increasingly dissected.	
Modular construction of software blocks	The software is built up modularly 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-in-1 row must operate under normal conditions as predetermined. This can be guaranteed by testing all software blocks individually and extensively together.	

The V-model is used throughout the project. The V-model is a project method that structures the progress of the project. For each specification or design phase on the left, there is a corresponding integration phase on the right. Each stage on the right side of the project can be verified and validated by the stage on the left (Figure 6).

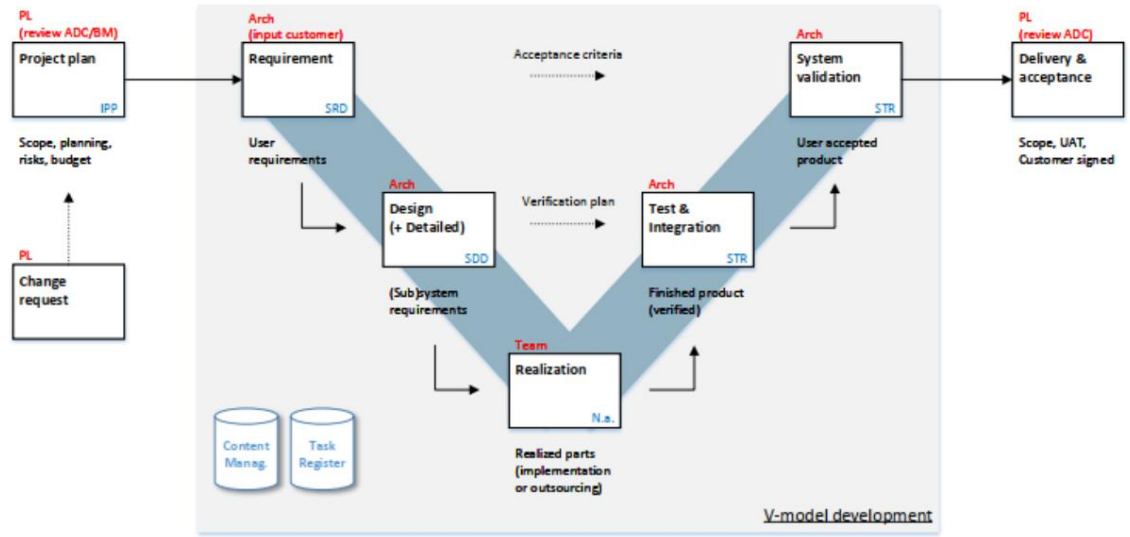

Figure 6 V model

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 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 consists of several black boxes. In level 2, the black boxes from level 1 become white boxes containing black boxes. During each step, the black boxes describe what the responsibility and tasks are. This is a good way to analyze 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.

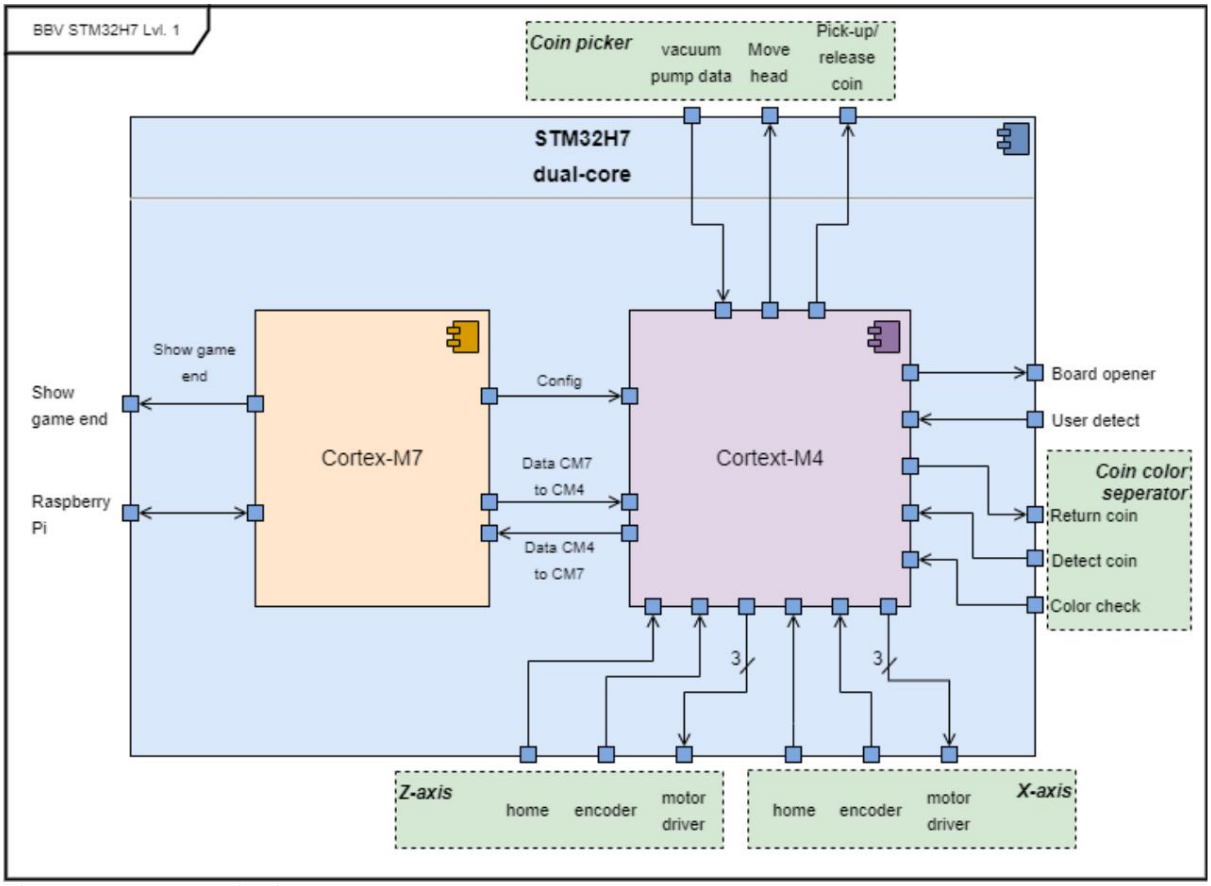

Figure 7 BBV STM32H7 level 1

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 game flow, communicating with the Raspberry Pi, initializing the whole 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, Engines, Coin separator, Board opener and User detect.

5.2Level 2

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

5.2.1 White box Cortex-M7

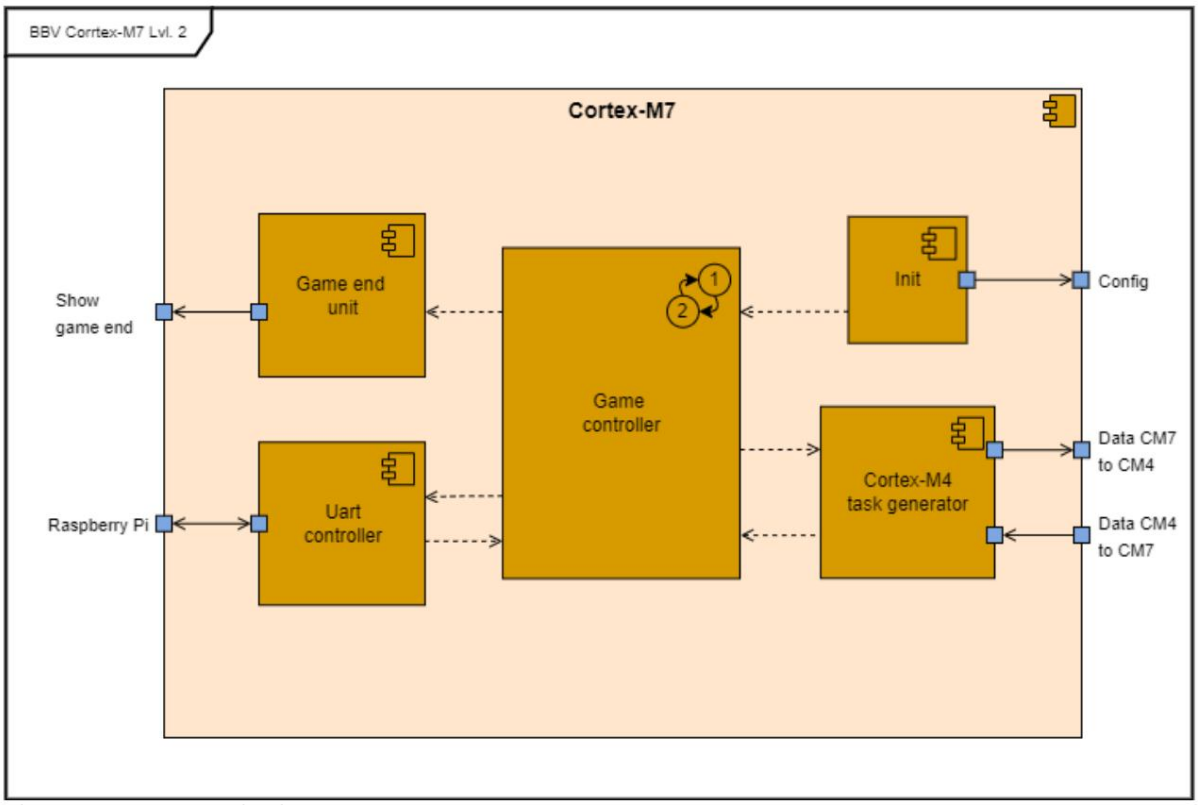

Figure 8 BBV Cortex-M7 level 2

Table 9 Description BBV Cortex-M7 level 2

Black box	Description
Game controller	The Game controller is responsible for the game flow through a state machine.
Init	Runs once to initialize and power up 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 communications to an output for the player.

5.2.2 White box Cortex-M4

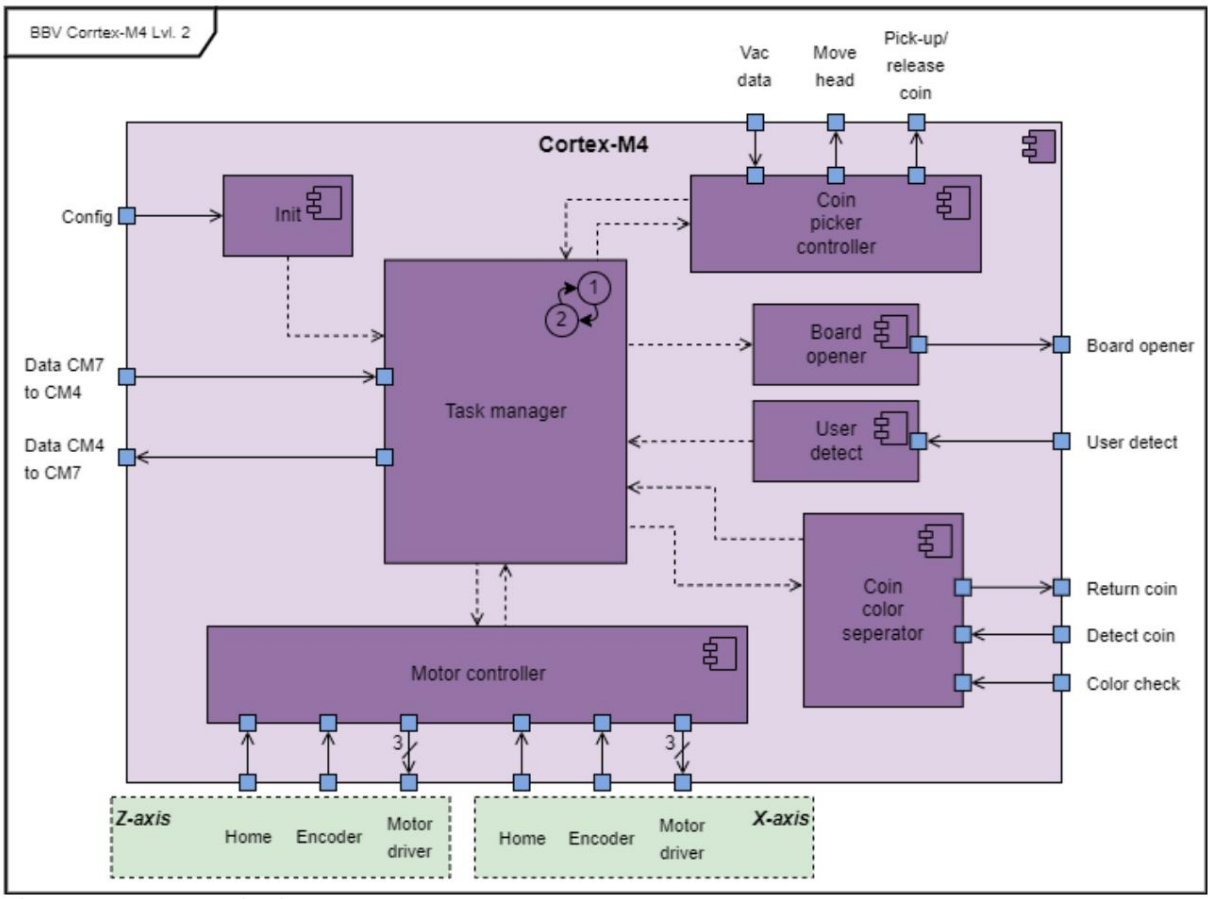

Figure 9 BBV Cortex-M4 level 2

Table 10 Description BBV Cortex-M4 level 2

Black box Description	The Task manager
Task manager	is responsible for the execution of the commands to be executed by the Cortex-M7 by means of a state machine. By using a “Task manager” the functionality of the other components can be built modularly. After all, they don't need to know about each other's existence/ status. This information is maintained by the Task Manager.
Init	Runs once to initialize and power up the Cortex-M4. In this phase, the sensors are tested for presence and the motors perform a “homing” protocol.
engine controller	The Motor controller implements all communication with the motor drivers and the PID controller.
Coin color separator	The Coin color separator implements all the functions for handling the chip separation.
User detect	The User detect implements all functions for handling the sensors for the player's input.
Board opener	The Board opener implements all functions for handling the opening of the 4-in-1 board when the game is over.
Coin picker controller	The Coin picker controller implements all the functions for handling the pick up and release of a chip.

5.3 Level 3 This

chapter turns the relevant black boxes out of level 2 white boxes and describes how the internal blocks are connected.

5.3.1 White box Motor controller

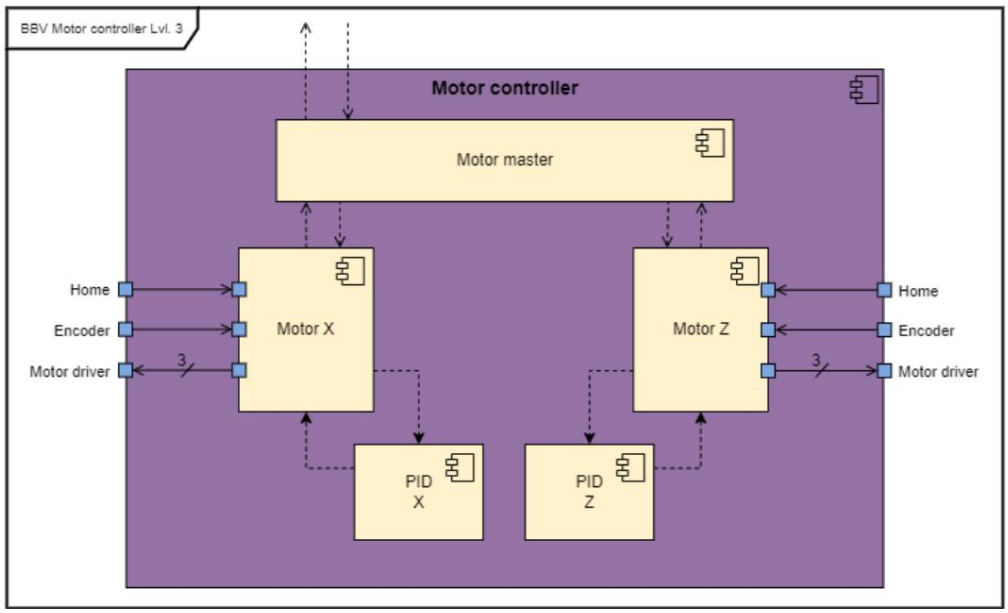

Figure 10 BBV Motor controller level 3

Table 11 Description BBV Motor controller

blackbox	*level 3*
Engine master	**Description** The motor master is responsible for controlling both motors.
Engine X	Motor X controls communication with the motor for the X axis.
PID X	PID X creates the control loop with feedback for motor X.
Engine Z	Motor Z controls communication with the motor for the Z axis.
PID Z	PID Z creates the control loop with feedback for motor Z.

5.3.2 White box Coin color separator

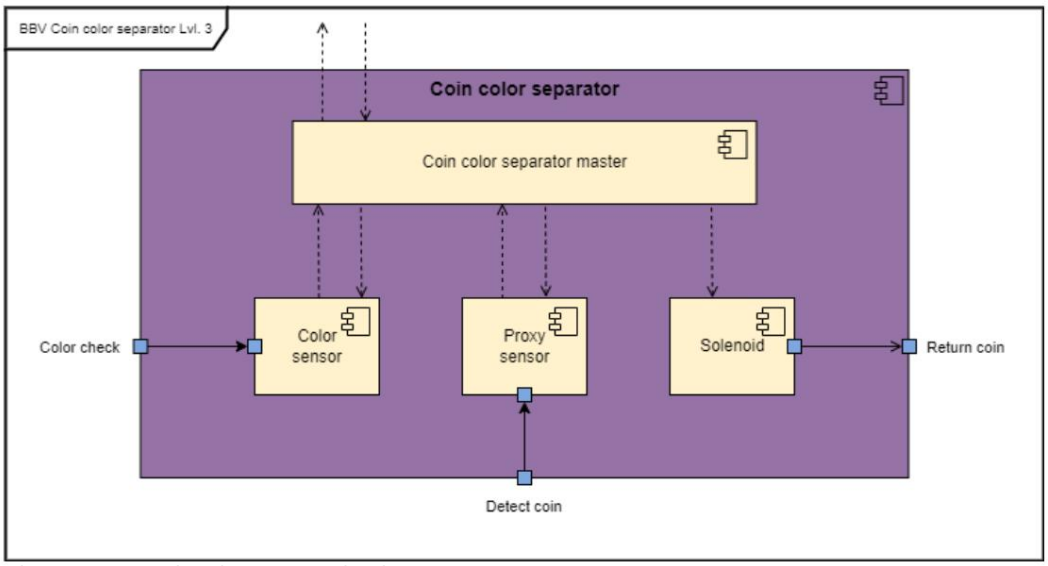

Figure 11 BBV Coin color separator level 3

Table 12 Description BBV Coin color separator level 3

Black box Description The Coin color separator	
Coin color separator master	master is responsible for receiving the sensor data and controlling the solenoid.
color sensor	Requests the data about the color of a chip.
proxy sensor	Gets data if a chip is present.
solenoid	Is responsible for activating the flipper.

5.3.3 White box coin picker

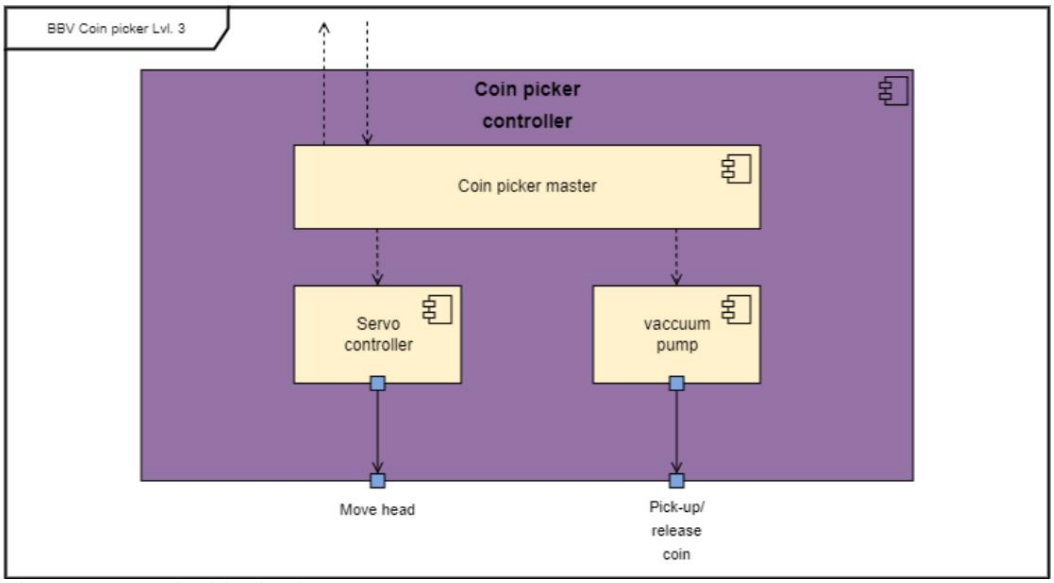

Figure 12 BBV Coin picker level 3

Table 13 Description BBV Coin picker level 3

Black box Description Coin picker The	
Coin picker master is responsible for controlling the **master** servo and the vacuum pump of the vacuum 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 Run-time 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 presented in the subchapters.

6.1 High level overview

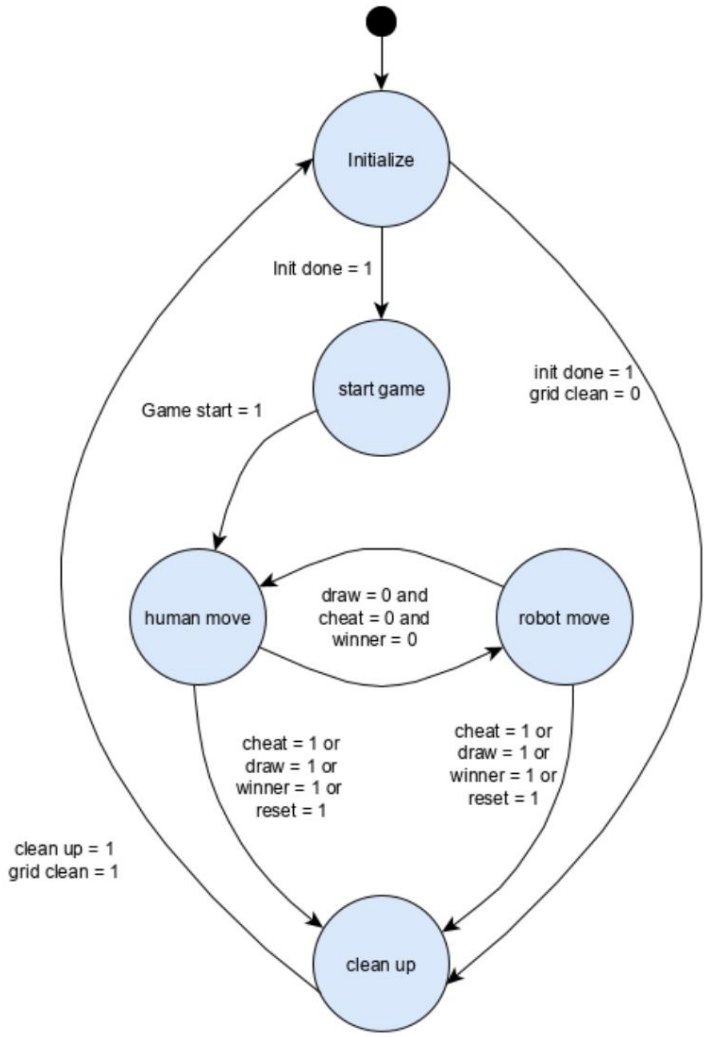

Figure 13 State machine Cortex-M7

The Statemachine in Figure 13 shows how the 4-to-1 rank 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-in-1 row. The first state is the “initialize” state. In this state, the entire system is booted and initialized. Once this state is finished the system goes into the “start game” state. In this state the system waits for the game to start. When the game has started, it is the player's turn to be the first to put his/her chip in the 4-in-1-row game and the system is in the “human move” state. Each turn, the robot checks whether there is a winner, cheat or tie. If this is not the case, the turn goes to the robot and the system goes to the “robot move” state. In this state, the robot performs the calculated move. Once the game is over, the “clean up” state starts and all chips are cleared. The red chips go to the robot and the yellow chips to the player's bin.

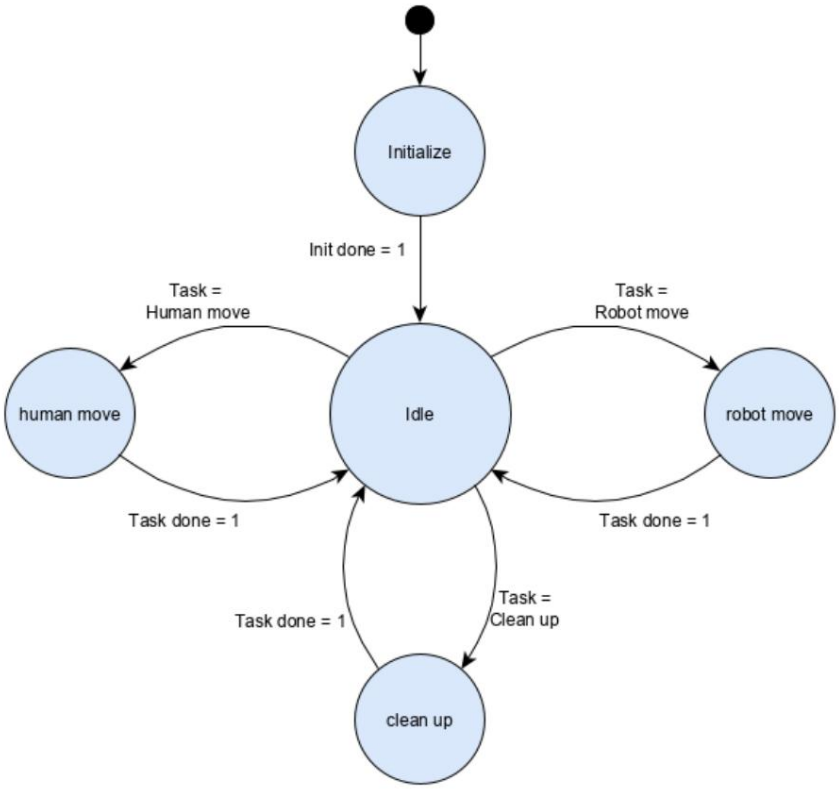

Figure 14 State machine Cortex-M4

As shown in Figure 9, the Cortex-M4 also houses a state machine. 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 set up. After that, the Task manager has an “Idle” state. In this state, the Task Manager waits for a task from the Cortex-M7. As soon as a task arrives, the next state is determined based on the task.
This can be the “human move”, “robot move” or “clean-up” state. Each of these states directs the hardware to complete the task. When the task is completed, the Task manager returns to the Idle state.

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

6.2 Robot moves

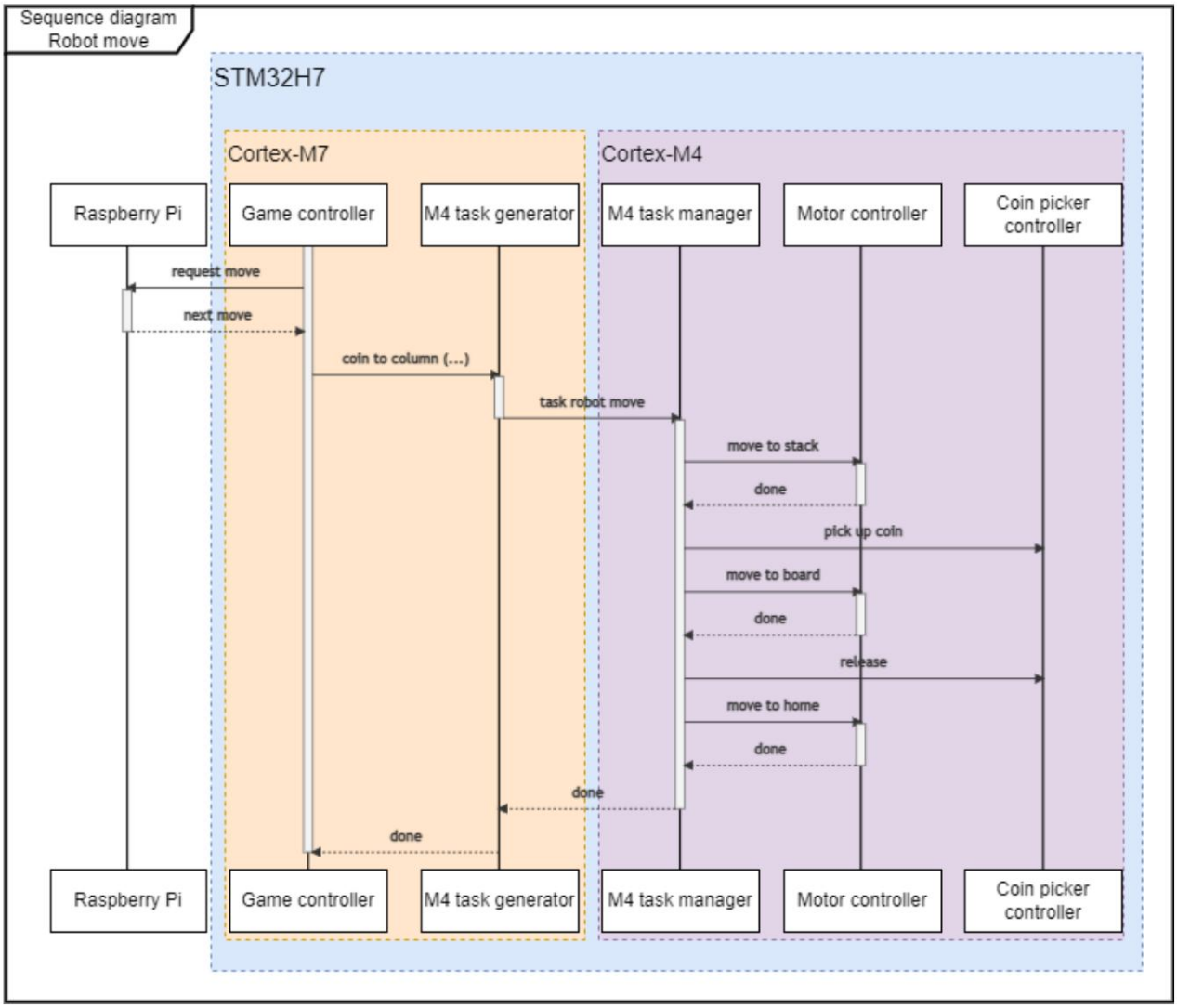

Figure 15 Sequence diagram Robot move

The “Robot move” in Figure 15 starts with the Game controller 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 job is created, the M4 task generator sends the job 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. Once the robot has arrived at the storage, a token must be picked up. Now the robot can move to the board and then release the token. Finally, the robot must return to its “home” position. When the move has been made, the M4 task manager informs 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

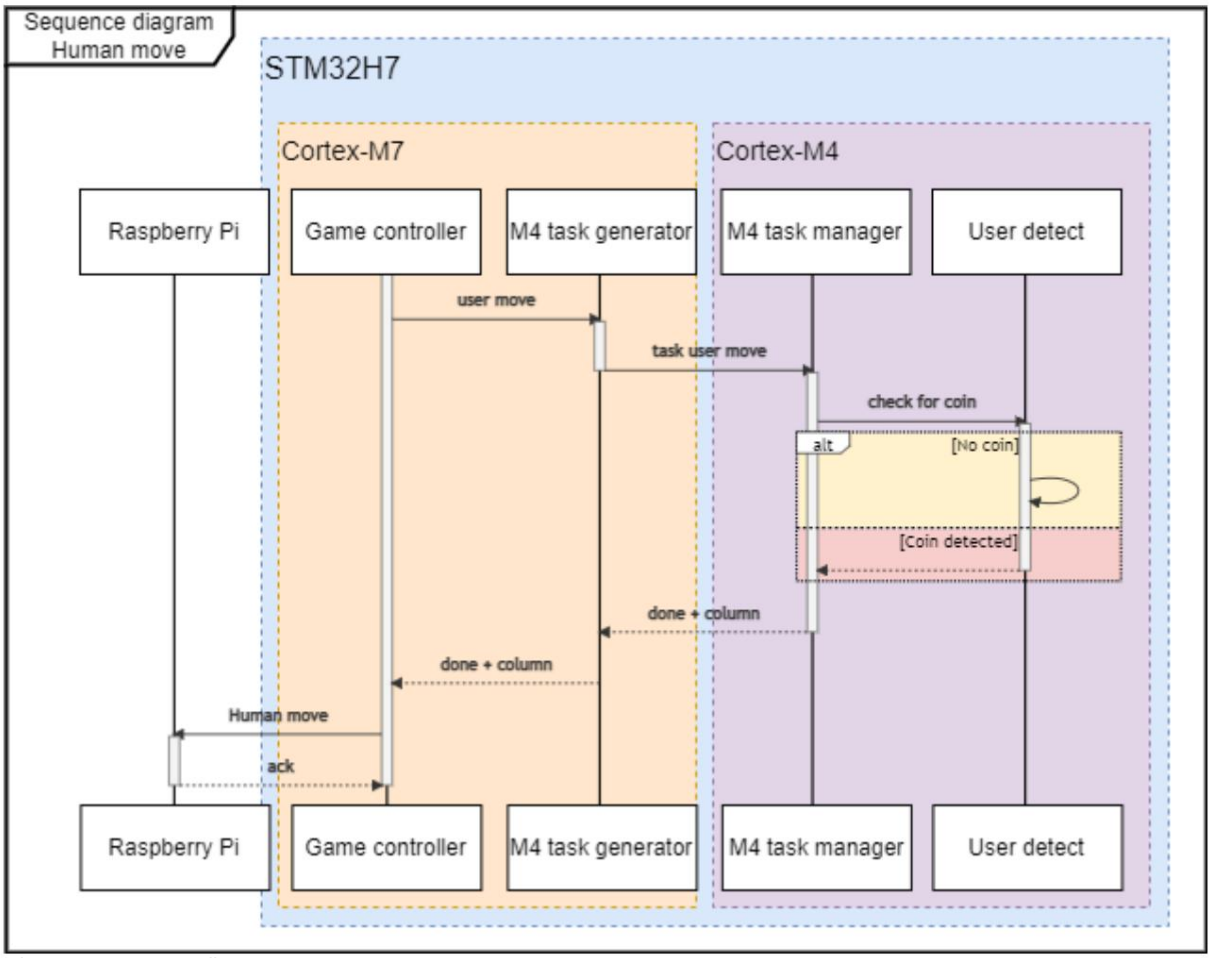

Figure 16 Sequence diagram Human move

The “Human move” in Figure 16 begins by signaling the player's 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 chip is expected. If no chip is played by the player, the system continues 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 chip is placed. The M4 task generator passes this on to the Game controller, after which it passes on the move to the Raspberry Pi. The Raspberry Pi acknowledges getting the move and sends back whether the game is done or can continue.

6.4 Cleanup

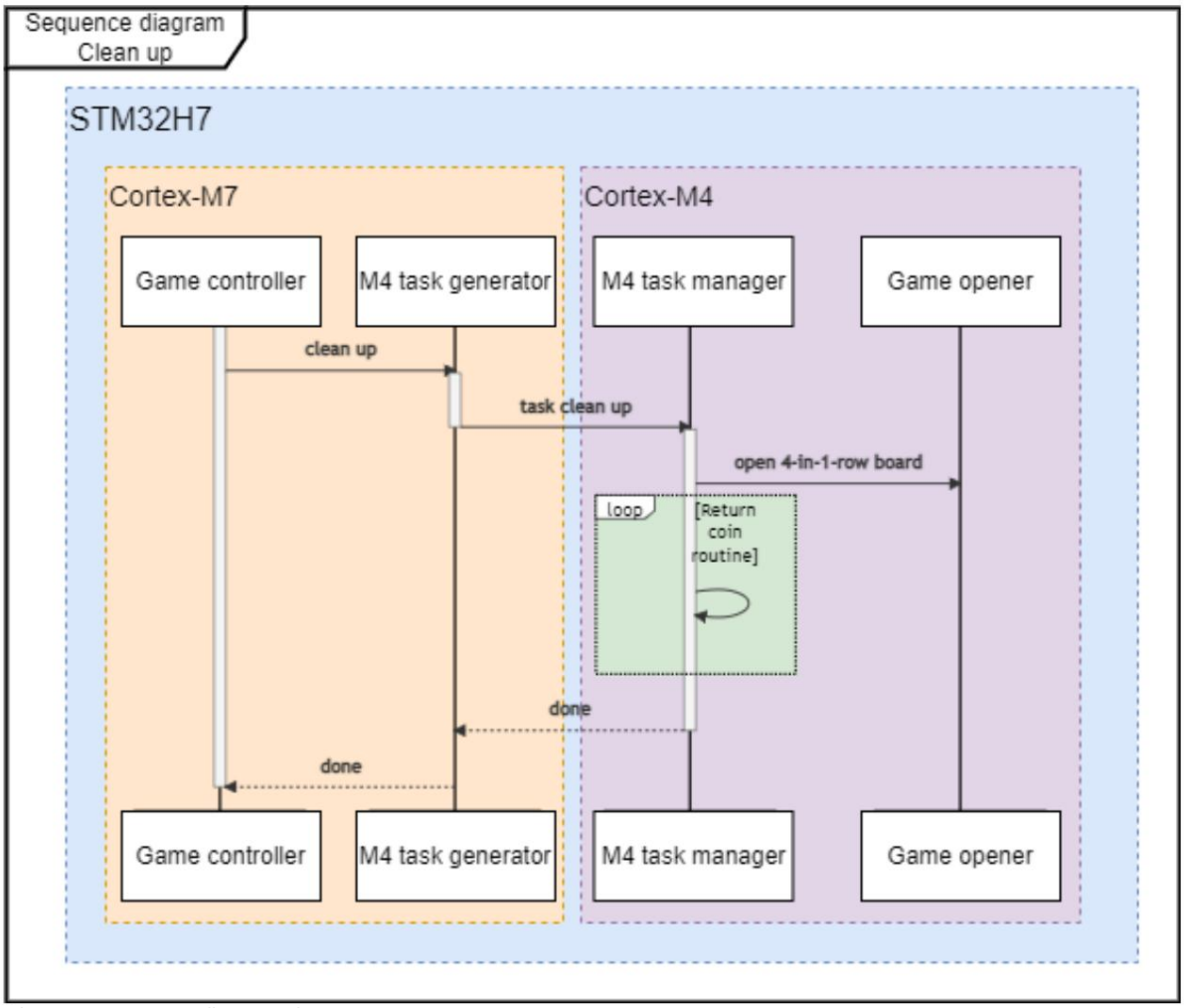

Figure 17 Sequence diagram Clean-up

The “Clean up” in Figure 17 begins by signaling that the game is over and the cleanup routine can begin at 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-in-1 board and starts the “Return coin routine” which runs until all chips are cleared. Once that is done, the M4 task manager tells 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.

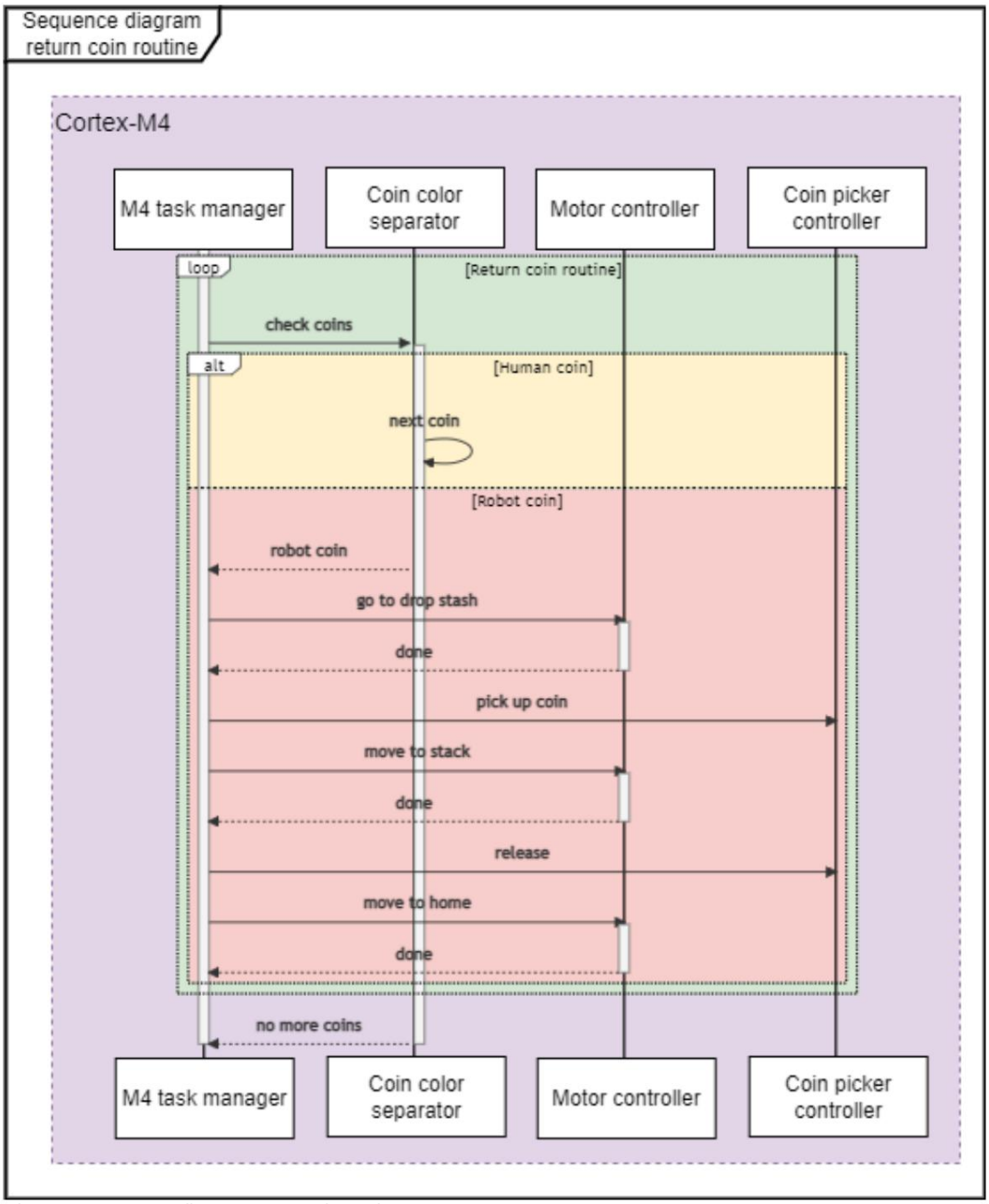

Figure 18 Sequence diagram return coin routine

The “Return coin routine” in Figure 18 is part of the “Clean up” in 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 chip is detected, the Coin color separator will check which color this chip has. If the chip belongs to the player, it is shot by a flipper at the player's tray. If the token belongs to the robot, the M4 task manager is informed that it is a token from the robot. The M4 task manager directs the Motor controller to go to the file. Then the token is picked up and the robot moves to its own chip storage. Here the token is released again and the robot moves to its “home” position. This loop continues until all tokens have been cleared.

7 Deployment View

7.1 NUCLEO-H755ZI-Q

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

Figure 4. Nucleo-144 board top layout

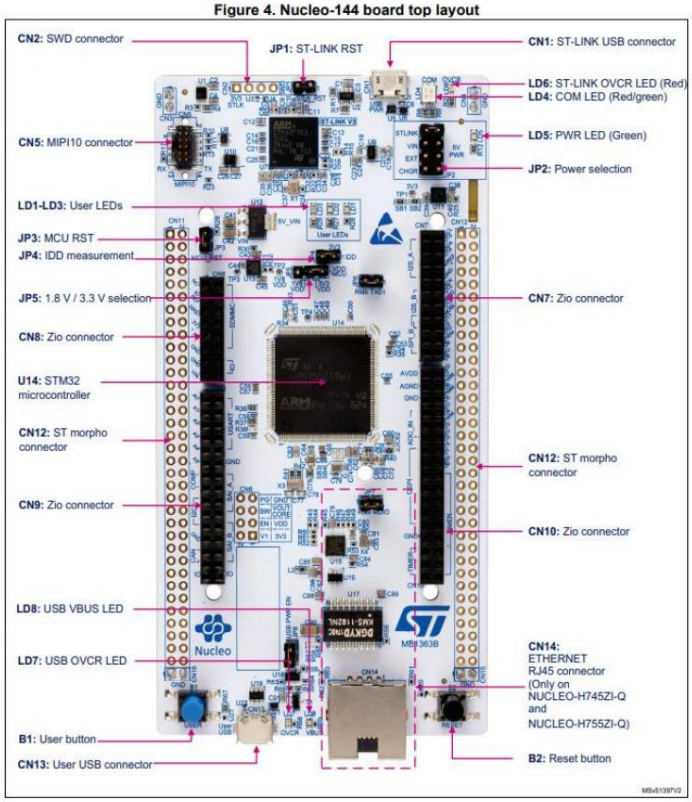

Figure 19 NUCLEO-H755ZI-Q

7.2 Pinout NUCLEO-H755ZI-Q

To realize all the functions of the 4-in-1 row, the pinout in Figure 20 has been composed. This pin out takes into account the function of each pin and what the pin should facilitate. Table 14 lists the function 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 established for Ethernet. Ethernet is a feature that may be used in the future but is beyond 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.

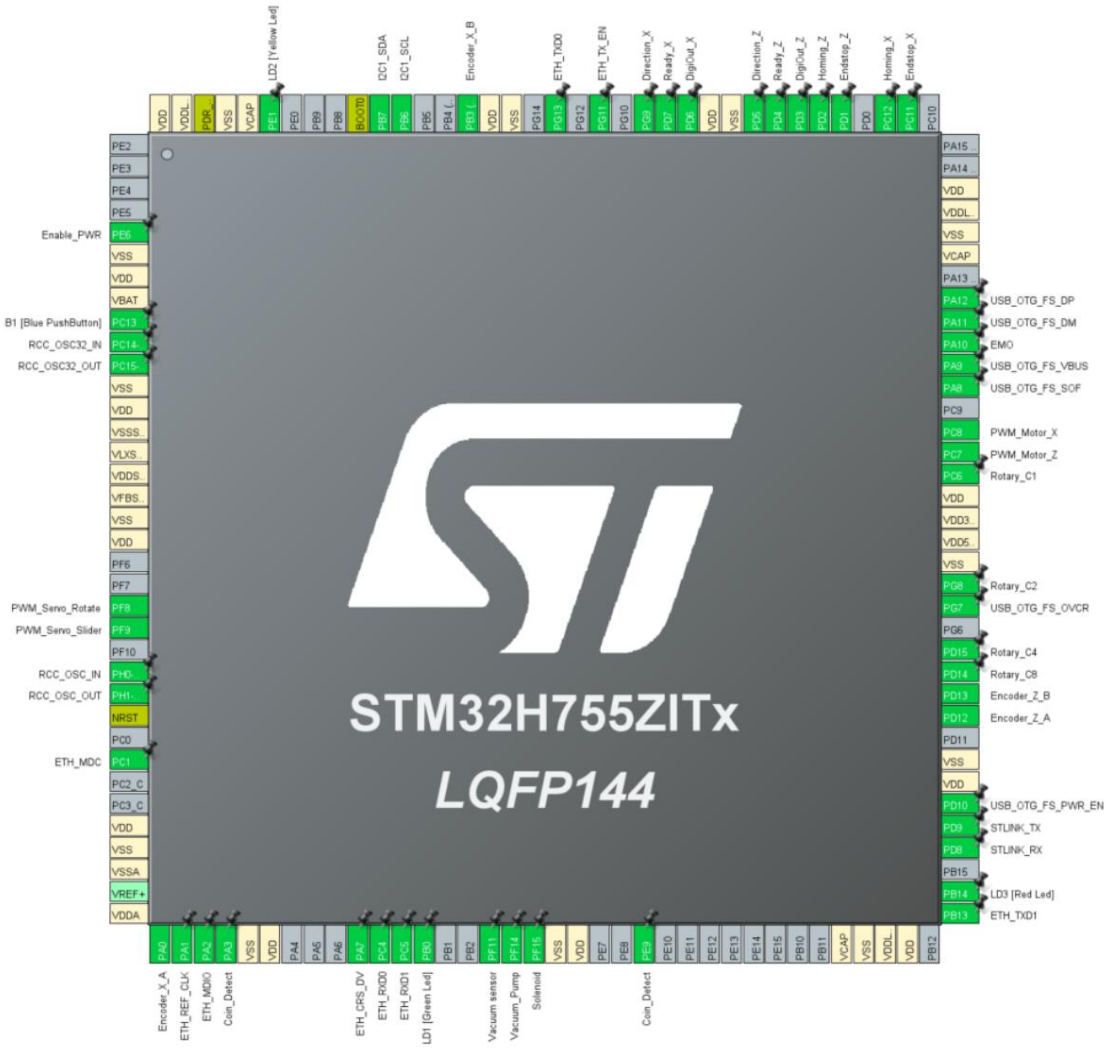

Figure 20 pinout STM32H755ZITx

Table 14 pinout 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	
DigiOut 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

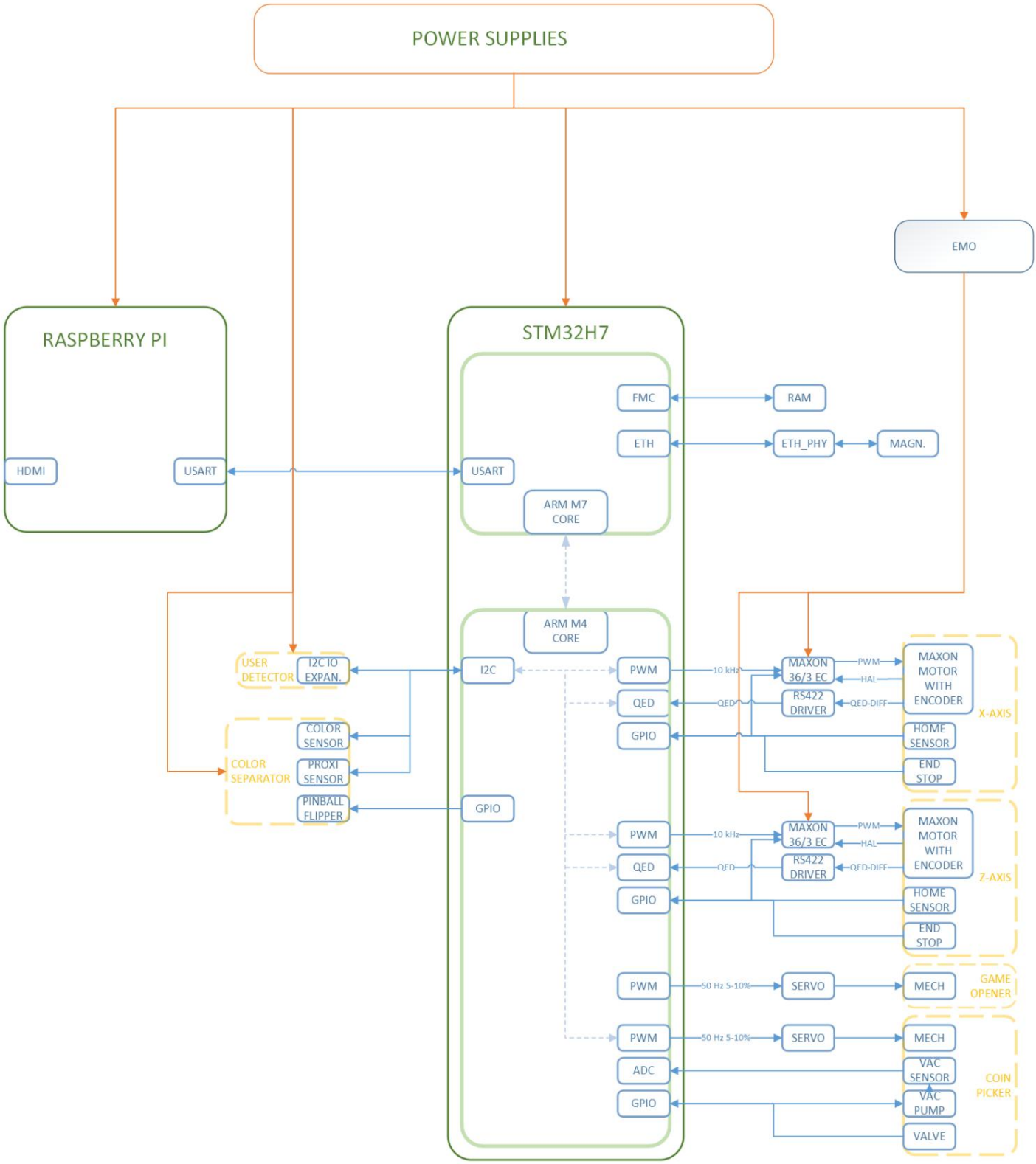

Figure 21 4-in-1-row hardware layout with STM32H7

7.4 Master-Minion relationship

Figure 22 has been drawn up to indicate how the processors relate in a Master-Minion relationship. This shows that the Cortex-M7 is the top master of the system. The game flow runs on the Cortex-M7 and therefore determines the entire game course. The Raspberry Pi and Cortex-M4 are a minion of the Cortex-M7 and perform tasks when the Cortex-M7 asks. The Cortex-M4 is also a master for the hardware. All hardware systems of the 4-in-1 row are a minion of the Cortext-M4 and perform tasks when the Cortex-M4 asks.

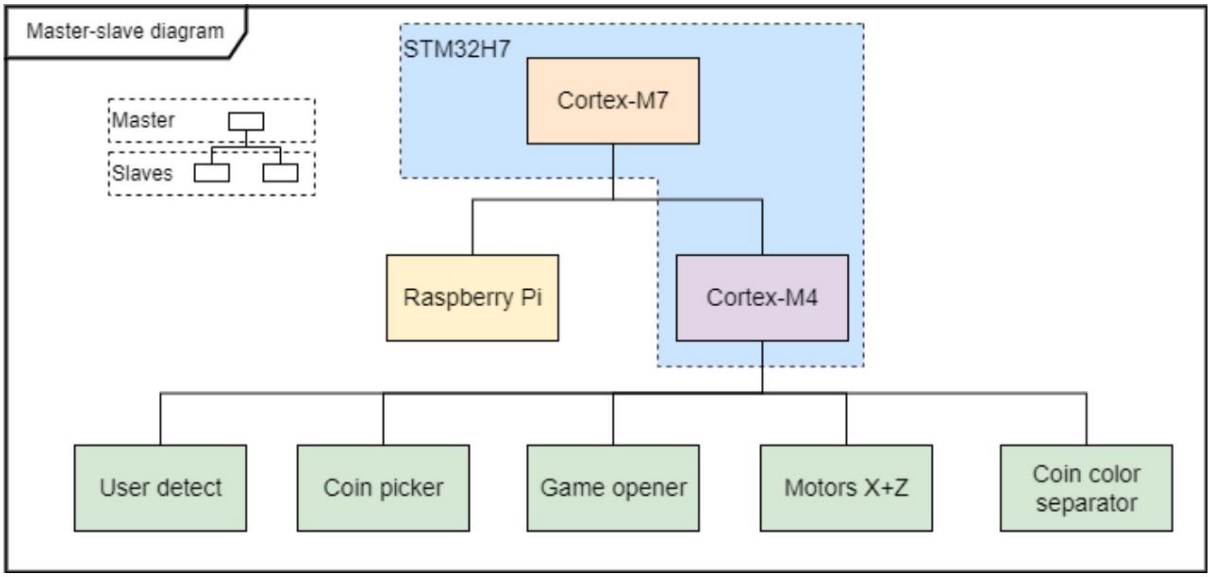

Figure 22 Master-minion relationship

8 Modular software implementation

(Chapter 5) shows which software blocks must be present to implement a fully functional 4-in-1-row robot. It is not shown here 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

In Figure 8 all software blocks of the Cortex-M7 core are shown. To show how the blocks relate to the hardware, the diagram in Figure 23 has been drawn up.

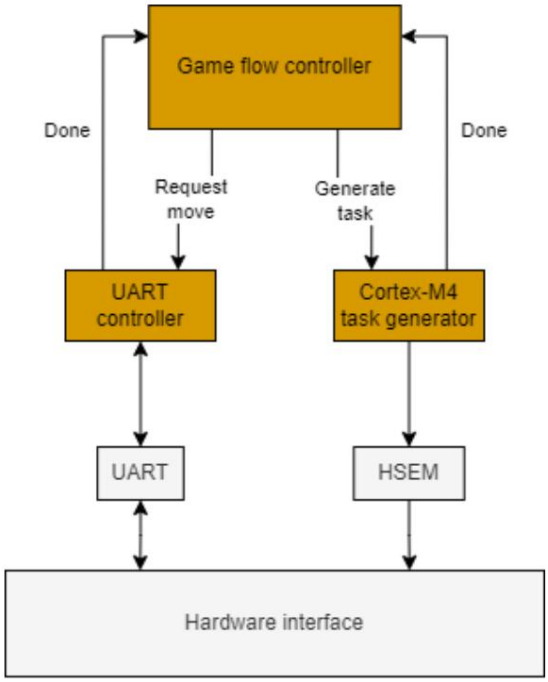

Figure 23 IBD game controller

8.2 Cortex-M4 core

In Figure 9 all software blocks of the Cortex-M4 core are shown. To show how the blocks relate to the hardware, all diagrams have been drawn up in the subchapters.

8.2.1 Engine controller

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

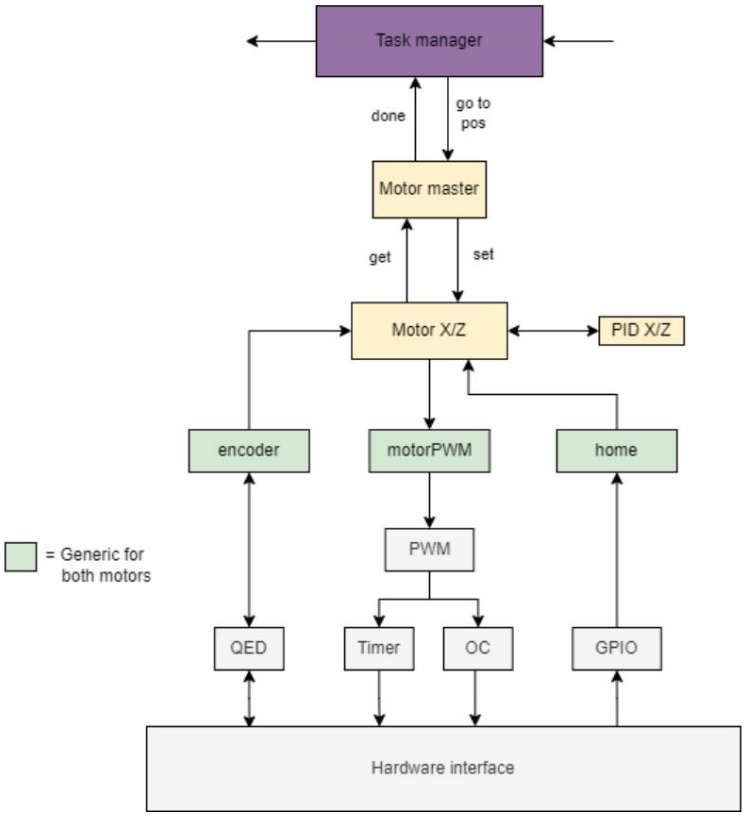

Figure 24 IBD Motor controller

8.2.2 Coin color separator

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

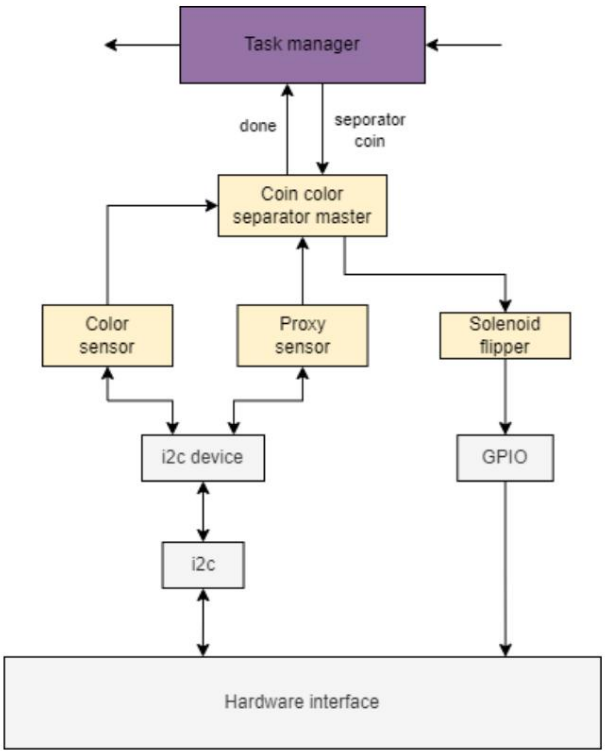

Figure 25 IBD Coin color separator

8.2.3 Coin picker

Figure 26 shows the diagram for the Coin picker.

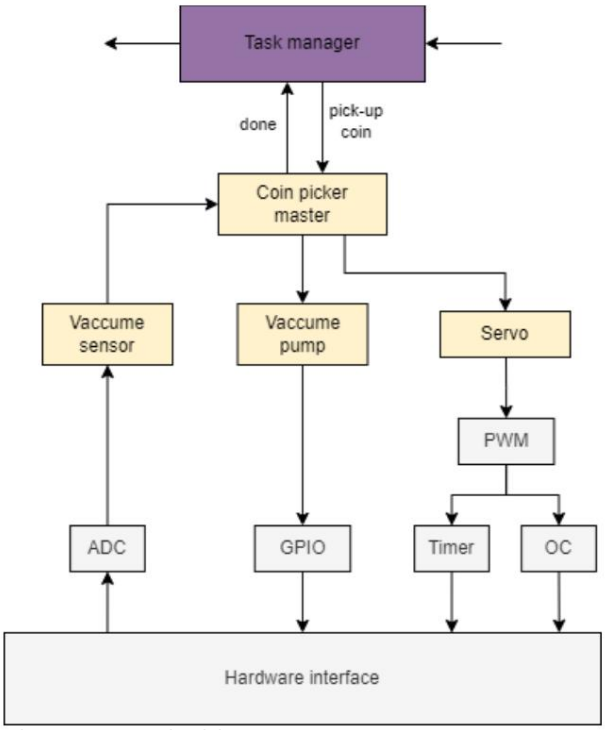

Figure 26 IBD Coin picker

8.2.4 Board opener

Figure 27 shows the diagram for the Board opener.

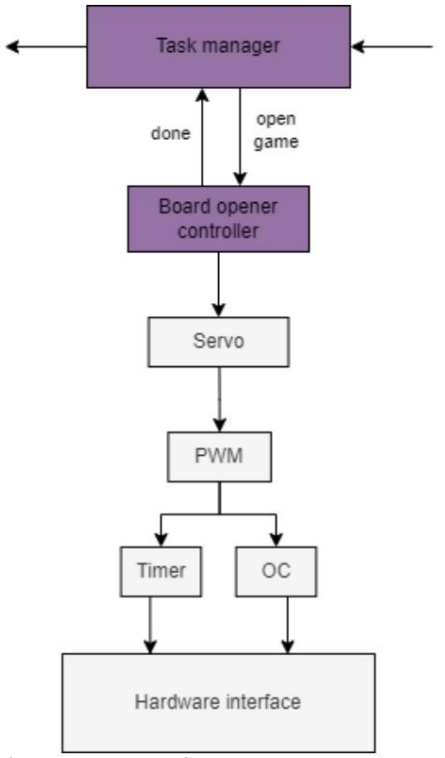

Figure 27 IBD Board opener

8.2.5 User detect

Figure 28 shows the diagram for the User detect.

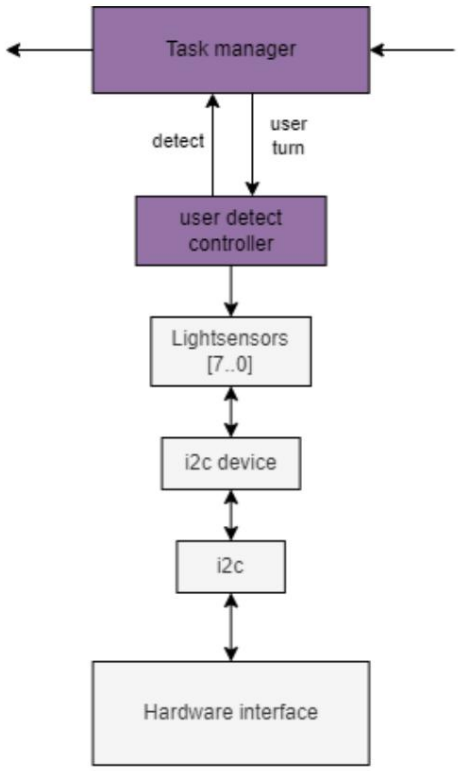

Figure 28 IBD User detect

9 Error handling According

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

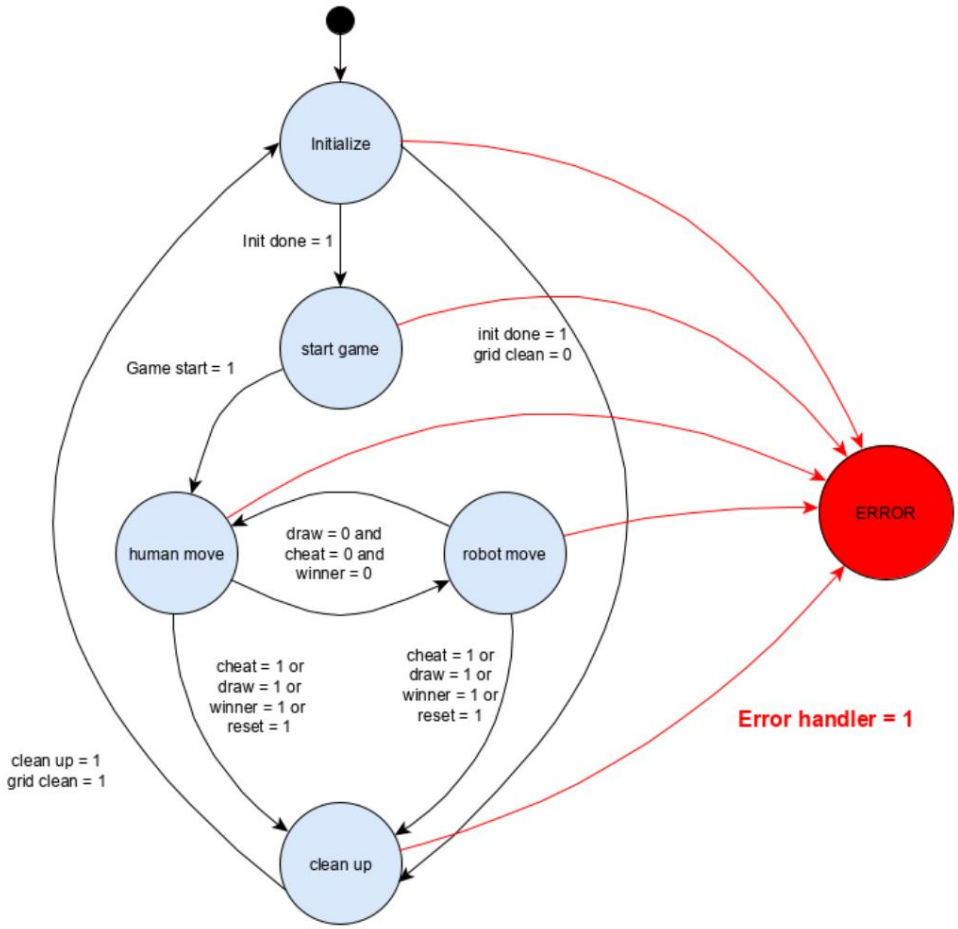

Figure 29 State machine with error handler In the “ERROR” state, all motors and other hardware components within the two cores of the STM32H7 are turned off and the system remains in this state until reset.