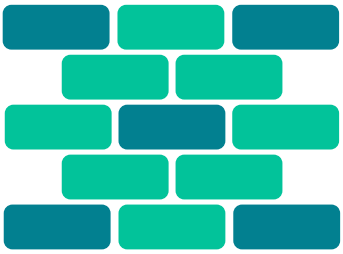
AutoArtisan

Jointing robot

WB3700 Robotics Minor Project

SMB GEVELTECHNIEK



AutoArtisan

Design Report

Prepared by:

Max van 't Hart
Neel Lodha
Tristan Little
Melihcan Balasar
Lili Sitányi
Rob Krüger

Date: September 4, 2026

Preface

This report was prepared by a group of six students from diverse academic backgrounds, including Computer Science, Electrical Engineering, and Mechanical Engineering, at Delft University of Technology. As part of the Robotics Minor and in collaboration with our client, SMB Geveltechniek, the project's objective was to design and develop a Jointing Robot capable of assisting operators in jointing brick walls on construction sites, thereby enhancing efficiency and precision in construction tasks.

The report provides a comprehensive overview of the robot's design, the engineering challenges encountered, and the implementation strategies employed. It is intended for robotics enthusiasts, engineers, and stakeholders interested in the project's technical and practical aspects.

We would like to express our gratitude to Maikel Schouw, our client; Martin Klomp, the Robotics Minor coordinator; Rob Bouwmeester, our teaching assistant; and the project sponsors, Bouwend Nederland and REBRICK, for their support and guidance throughout this project.

Summary

The purpose of this project was to build a robot that is able to perform both rake and flush joints on walls with various brick patterns, whilst meeting our client's standards for speed and quality. In this way, the robot will help alleviate the physical strain that construction workers experience as a result of jointing walls. This report focuses on finalizing the robot's detailed design and justifying the main decisions we made in the process.

This robot helps mitigate challenges such as the scarcity of skilled human jointers and the physically strain caused by masonry work. By collecting data from a depth camera, encoder feedback, and other sensors, the robot detects and fills joints with precision, improving both the speed of the process and the uniformity of the joints filled.

In the long run, this robot has the potential to streamline construction projects, reduce the frequency and severity of worker injuries, and address labor shortages by automating a hitherto manual task. In the future, the robot could have a pivotal role in making construction sites safer and more productive, while also benefiting workers and furthering innovation in the industry.

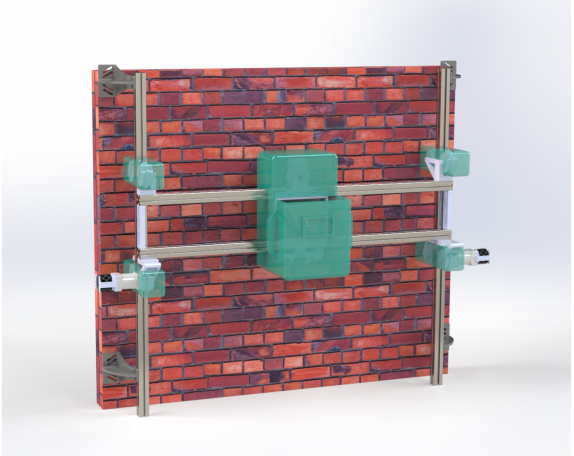


Figure 1: Final design with the casing

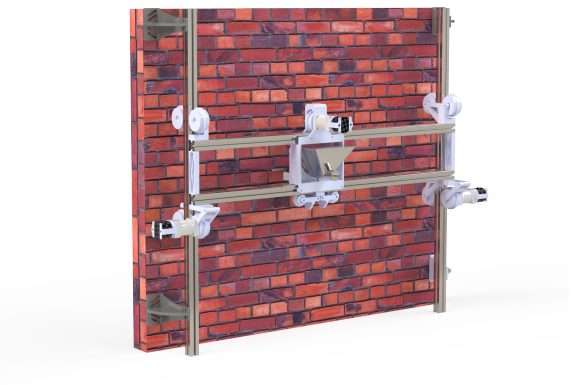


Figure 2: Final design without the casing

Table of Contents

1. Introduction

1.1 Design brief

The aim of this project is to create an autonomous robot to help construction workers joint walls. The robot will joint on straight, vertical, brick walls. These walls are at minimum 1.2 meters wide and 2 meters high and can be larger. The bricks can be arranged in any standard masonry pattern/bond. These bonds can be stretcher, Flemish, cross, tile, freestyle or others. The operator will be able to choose the type of joint that the robot will execute. Joint styles referring to flush-, rake-, tuck-, shadow joints or more. To execute an authentic masonry joint, the robot must extrude the mortar into the gaps between bricks, press the mortar smooth, brush off rough edges and have the joints moistened. For the robot to be competitive with a human mason, it needs to joint at least 30 m^2 of wall in a day. A future rendition of this robot might be able to chisel out old joints as well as joint on corners and window casings.

1.2 Refined requirements

This section contains all the requirements for the brick-jointing robot. The requirements are written using SMART notation, where:

- S - Specific: Requirement targets a specific area for improvement
- M - Measurable: Requirement has a quantified or a suggested indicator of progress.
- A - Assignable: Requirement is assigned to a specific discipline/team member
- R - Realistic: Requirement must be realistic to achieve given the available resources
- T - Time-bound: Specifies a time frame for expected results

Requirements were all given an ID that helps identify its type, the system, and the sub-system that the requirement belongs to. The first part of the ID stands for the project Mason, and the second part stands for the system (JOI: Jointing, GAN: Gantry, DMA: Data Manipulation, and USC: User Control). If a requirement refers to multiple systems then the system section of the ID is assigned GLB (global). The third part of the ID stands for the subsystem (MOB: Mobility, EXT: Mortar extrusion, SMT: Smoothing tools, BRS: Brush, DIG: Digital, ANL: Analog, DPR: Digital Processing). For requirements assigned to the global system, the subsystems are ENV: Environment. If a requirement refers to more than one subsystem, it is assigned ALL. The fourth part of the ID refers to the requirement's type (FUN: Functional, PER: Performance, DES: Design, SAF: Safety, and COS: Cost).

Element	Requirement ID	Requirement	Value	Verification Technique
Primary	MAS_GAN_STB_FUN_01	Gantry Shall be stable and rigid relative to the wall		Demonstration
Primary	MAS_GAN_MOB_FUN_01	System shall independently control horizontal movement of the gantry	1.2 m	Demonstration
Primary	MAS_GAN_MOB_FUN_02	System shall independently control vertical movement of the gantry	>1.2 m	Demonstration
Primary	MAS_JOI_EXT_FUN_01	Robot shall extrude mortar out of its nozzle	1 - 10 L/min*	Test
Primary	MAS_JOI_SMT_FUN_01	System shall smooth joints with a consistently uniform finish	Max. joint surface deviation 0.1mm	Inspection
Primary	MAS_JOI_SMT_FUN_02	Tools shall smooth out the joints in the horizontal direction		Demonstration
Primary	MAS_JOI_SMT_FUN_03	Tools shall smooth out the joints in the vertical direction		Demonstration
Primary	MAS_JOI_SMT_FUN_04	Tools shall be replaceable in case of wear		Demonstration
Primary	MAS_JOI_SMT_PER_01	Smoothing tools shall joint a minimum area before breaking	100 m^2	Inspection, Analogy
Primary	MAS_JOI_BRS_FUN_01	Joints shall be brushed after smoothing		Demonstration
Primary	MAS_JOI_ALL_FUN_01	System shall control activation of tool heads independently		Demonstration, Test
Primary	MAS_DMA_DPR_FUN_01	Reservoir volume shall be monitored		Inspection, Demonstration
Primary	MAS_DMA_DPR_FUN_02	System shall keep track of its current position		Inspection, Test
Primary	MAS_DMA_DPR_FUN_03	System shall establish a coordinate system		Inspection, Test
Primary	MAS_DMA_DPR_FUN_04	System Shall detect and differentiate between bricks and joints		Demonstration, Test
Primary	MAS_DMA_DPR_FUN_05	System shall create a path for the tools to navigate along		Demonstration, Test
Primary	MAS_USC_DIG_FUN_01	System shall keep track of how many square metres of wall it has already jointed		Inspection, Test
Primary	MAS_USC_DIG_FUN_02	System shall issue operator alerts when an error occurs or maintenance is required		Demonstration, Test
Primary	MAS_USC_DIG_FUN_03	System shall coordinate the filling of multiple joint types (flush, raked)	At least 1 joint type	Demonstration, Test
Primary	MAS_USC_DIG_FUN_04	System shall issue warning messages when mortar in the reservoir is running low	20%	Demonstration, Test
Primary	MAS_USC_DIG_DES_01	System shall show different states and errors on a digital/interactive interface.		Inspection
Primary	MAS_USC_ANL_PER_01	System shall joint a minimum area in a day	30m^2	Inspection, Analysis
Primary	MAS_GLB_ENV_PER_01	System shall operate consistently on vertical straight walls of a minimum height	>1.2 m	Analysis, Demonstration
Primary	MAS_GLB_ALL_FUN_01	Components shall have built-in cushioning against heavy handling	Stiffness, k = 300 N/cm	Demonstration, Test
Primary	MAS_GLB_ALL_DES_01	Components shall be water resistant to a standard level	IP65	Inspection
Primary	MAS_GLB_ALL_DES_02	Components shall be dust resistant to a standard level	IP65	Inspection
Primary	MAS_GLB_ALL_DES_03	Components shall each have a maximum mass	20kg	Inspection
Primary	MAS_GLB_ALL_DES_04	Robot shall fit within a 'standard' van	350x175x190 cm	Inspection
Primary	MAS_GLB_ALL_SAF_01	Robot shall have synchronized emergency shutoff capability at the push of a button		Demonstration, Test
Primary	MAS_GLB_ALL_SAF_02	Robot shall have an automatic brake if the power is cut		Demonstration, Test
Primary	MAS_GLB_ALL_COS_01	Robot shall be developed within a budget	€3,000	Inspection

1.3 System tree and risk assessment

1.3.1 System Tree

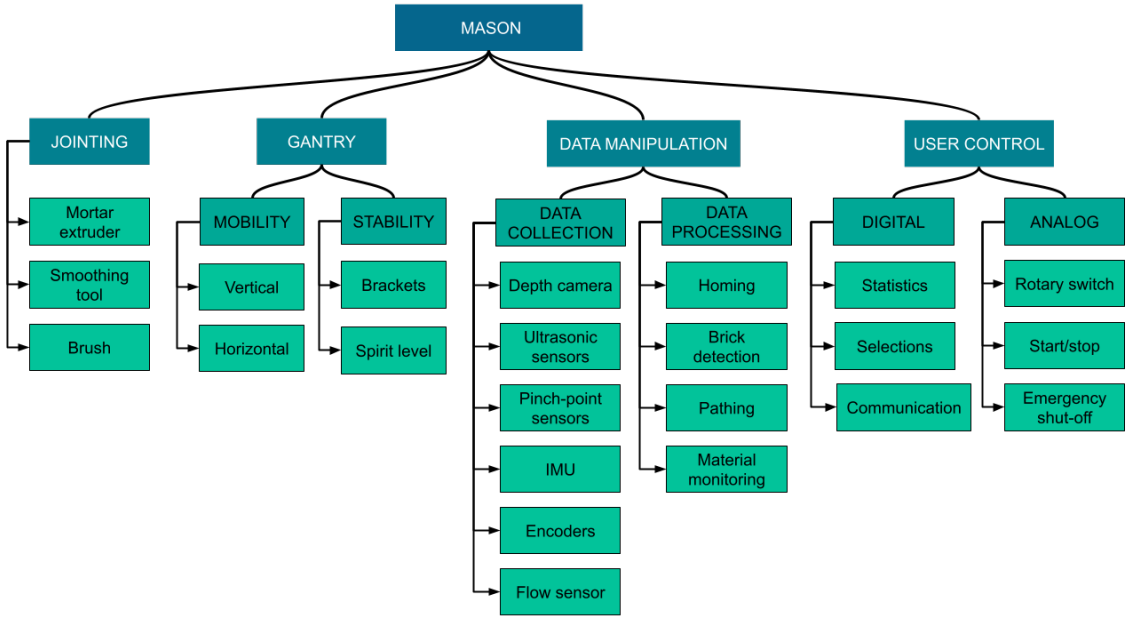


Figure 1.1: The System Tree for the jointing robot

The system tree is shown in Figure 1.1. The brick jointing robot (Mason) consists of four systems: Jointing, Gantry, Data Manipulation and User Control. The Jointing system consists of the tools needed for filling, smoothing and finishing joints (i.e., a brush, a smoothing and mortar extruder). The Gantry consists of the components responsible for holding the robot in place and providing structural support as well as moving the tool heads around. The tools need to be move both vertically and horizontally. To ensure that the robot is properly set up and aligned with the wall before operation, the gantry will have brackets and a spirit level. The Data Manipulation system is split into two subsystems: Data Collection and Data Processing. The Data Collection subsystem consists of the components used to obtain information about the environment and the status of the robot. These components include a depth camera to help collect information about the surface of the wall and map a path; an ultrasonic sensor to help monitor the mortar levels in the reservoir; a pinch point sensor to prevent injuries; a flow sensor to help monitor the flow rate of mortar through the extruder and an IMU so that the acceleration/orientation of the robot can be factored into calculations. Data Processing is a subsystem dedicated to using data collected from sensors to execute more complex functions, such as brick detection, mapping a path, homing and material monitoring. The User Control system refers to the components that the user and the robot use to communicate and the information exchanged between them. The Digital subsystem includes possible user inputs such as selecting a joint type. Communication refers to any error or maintenance messages that may be displayed on the GUI. Statistics refers to any metrics that can be viewed by the operator like square meters jointed in the current session, square meters jointed so far during the day, percentage of time spent refilling the mortar or dealing with maintenance messages, etc. The Analog subsystem refers to electronic components that the operator uses to navigate the GUI. This refers to a rotary switch that is turned to scroll through menu options and then pressed to select an option, a start/stop button, and an emergency

shutdown button.

1.3.2 Risk assessment table

The risk assessment table is split into four sections: Identification, Classification, Taking Action and Action Results. Identification splits the robot into subsystems, defines potential failures and their effects. The severity of these potential failures is given a score from 1 - 10.

Identify						
Item	Function	Requirements	Failure ID	Failure Mode	Effects of potential failure	Severity
Power source	Powering the system	Provide reliable and sufficient power to every electrical component in the system	1	External power is cut	Robot comes crashing down	8
Mortar extruder	Extruding mortar	Extrude mortar in a way that gives the joint a consistently uniform surface	2	Is misaligned	Mortar ends up outside the joint	2
Gantry	Holding the robot in place and providing structural support	Provide structural support to the robot and allow smooth movement	3	Frame falls from where it's mounted in the wall	Injury, Damage to robot	10
Chassis	Housing tool-heads, motors, gears, power supplies and other components of the robot	Safely housing toolheads.	4	Limb is pinched between moving parts	Injury, Damage to robot	7
Camera	Recording/viewing area of wall in front of the robot	The camera should be able to get accurate depth/color images of the wall	5	Damage	Errors while filling/detecting joints	3
Motors	Moving the robot along both X and Y axis	The robot should be able to move anywhere on the plane of the wall	6	Damage	Obstruction in movement, Injury	8
Wheels	Moving the robot along both X and Y axis	The wheels should give enough grip to have a reliable and safe power transfer from the motors to the gantry to enable movement	7	Damaged or worn out wheels	Obstruction in movement, incorrect movement, injury in case of fall	5

The classification table identifies the cause of the failures outlined in the 'Identification' section of the table, potential methods of detection and prevention. The failure is then assigned a risk priority number (RPN) of the current risk and its solution. This number is computed from the severity of the failure, the likelihood of the failure occurring and the effectiveness of the current detection/prevention method.

Classification							
Failure ID	Classification	Potential Cause of failure	Prevention	Occurrence Likelihood	Current Design Controls (Detection/ Prevention)	Effectiveness of best method of detection control	RPN (Risk priority no.)
1	Danger to life	Power outage, user disconnect, power supply failure, emergency stop, wire failure	Mechanical brake that is only released when power is provided	3	Alarm/Warning lights	5	120
2	Waste of materials	Improper set-up	Careful documentation of set-up procedure, gyroscope to detect alignment and prevent operation when misaligned	5	Slots in brackets	-	10
3	Danger to life	Erosion/cracks in mortar that the frame is mounted in	Frame fixed to mortar at multiple points	2	Frame fixed to mortar at multiple points	-	20
4	Danger to life	Lack of casing/protection or detection from the robot	Emergency stop button, pinch detection sensors and covering moving parts	5	Edge on the chassis to prevent external objects from sliding between the chassis and the wall	-	35
5	Waste of materials/time	Rough handling, frame falling from where it's mounted	Add casing that protects the camera and prevents accumulation of dust on the camera sensors	6	Protective casing/error messages when no joints are detected	3	54
6	Danger to life, Damage to robot	Operator is blocking the default movement of the robot	Add a pinch point sensor to prevent accidents caused by trapped limbs	5	Edge on the chassis to prevent external objects from sliding between the chassis and the wall	-	40
7	Damage to robot	Extensive use without maintenance	Enforce regular maintenance to attach new wheels, make wheels from the best material available	10	Scheduled maintenance notifications	2	100

The last two sections of the FMEA table outline an ideal detection solution for each failure, assign the implementation of said solution to a specific discipline, and then computes a new RPN based on the advantages of the recommended solution over the current detection solution.

3. Take Action			4. Results of taking recommended action			
Failure ID	Recommended action(s)	Responsibility	Severity	Occurrence likelihood	Efficacy of detection/prevention	New RPN
1	Mechanical brake that is only released when power is provided	ME	5	3	2	30
2	Accelerometer to detect alignment combined with clear set-up instructions	EE and CSE	2	4	3	24
3	An IMU/tilt sensor mounted to the gantry may detect when the gantry tilts past a safe angle. A vibration sensor could flag unexpected movements	EE and CSE	10	2	3	60
4	Emergency stop button and pinch point sensor	EE and CSE	7	5	2	70
5	Protective casing, monitor frequency of detection error messages - if there is a significant increase send a maintenance notification for the depth camera	ME and CSE	3	5	3	45
6	Use of pinch point sensor	EE and CSE	7	5	3	105
7	Scheduled maintenance notification for wheel replacement every 100,000 cycles of the wheel	CSE	5	8	2	80

2. Concept Analysis

2.1 Ethics

Our mission is to help lessen the workload on jointers, improving their productivity while decreasing the risk of health complications caused by the strenuous work. We will be doing this by making a jointing robot that does the most of the monotonous jointing work almost autonomously. At the same time, it is also our desire to preserve the traditional masonry craft. Our robot should not replace jointers or their jobs. Instead it should act as a tool to support and collaborate with them. With regards to the more external stakeholders such as residents near potential building sites, it is the team's mission to create a robot that produces minimal noise pollution and works as quickly as possible, to avoid disturbing local residents. Naturally, ensuring high quality joints which improve the structural integrity of the wall we work on is also of significant importance. We don't want to permanently damage the wall or leave it in a more dangerous state. AutoArtisan should also go for the more environmentally friendly options where the choice can reasonably be made. In addition, it is preferable for the design and operation of the robot to be straightforward, so that the robot is easy to work with for workers from any skill level.

This chapter dives deeper into ethical considerations in the development of a brick-jointing robot, focusing on the well-being of the aforementioned stakeholders.

2.1.1 Stakeholders & values

The primary stakeholder involved in using the robot is the operator: the operator is responsible for setting up the robot, configuring it to perform specific tasks and ensuring that it operates as intended. The operator's decisions directly impacts the safety and success of the robot's operation and is thus classified as a direct stakeholder.

The operator should prioritize their own safety above anything else. While the robot will include built-in safety features to help reduce the risk of injuries, it is essential that the operator remains aware of potential safety hazards. Being aware of these risks and taking appropriate precautions help to creating a safe working environment. The operator would likely need training and clear guidance to properly use the robot and might therefore also value transparent communication with the developers.

In addition to transparency and safety considerations, the operator might also value reliability and efficiency in the robot's design. A robot that works consistently without frequent malfunctions improves the operator's productivity and presumably results in less frustrations with the operator.

One layer below the direct stakeholders are the indirect stakeholders. These include the project team, the client and the sponsor. Although they do not directly interface with the robot, these actors make it possible for the robot to exist. At the root of the robot's development is the client. The client presents their vision and requirements to the project team. These requirements are shaped by the client's values, such as the desire for quality, cost-effectiveness or ease-of-use. The input of the client serves as the foundation upon which the entire project is built. The client also serves as the primary link between the industry and the technical team, keeping the technical team up-to-date with the latest industry demands.

The project team is responsible for translating this vision into reality. It brings together multiple disciplines to solve a wide range of challenges. But not only does their role involve satisfying the client's requirements, they also refine the robot so that it performs as reliable and efficient as possible.

Lastly, the sponsor provides the financial resources needed to see the project through. Their supports enables the project team to acquire the necessary materials and tools that will be included in the robot. Without their contribution, the robot would likely not make it past the conceptual stage. Just like the client, sponsors value cost-effectiveness.

Even more distant from the robot are the external stakeholders, among which are the building owners, residents, manufacturers and bystanders. Although they have very little to no interaction with the robot, the robot's operation can still significantly affect them in multiple ways. Building owners, for example, may value operating quality so that there is little maintenance needed after the robot has completed it's operation. For the same reason, residents could value reliable operation, which improves their quality of living. Bystanders on the other hand might be affected by the robot's presence in public spaces. They could experience discomfort from the robot's operation or, if the robot severely malfunctions, it may even injure a bystander. Therefore values that would suit the bystander include safety and comfort. Finally, manufacturers who are involved in producing components for the robot may value simplicity. A straightforward design allows for more efficient production, which could positively impact the robot's cost-effectiveness.

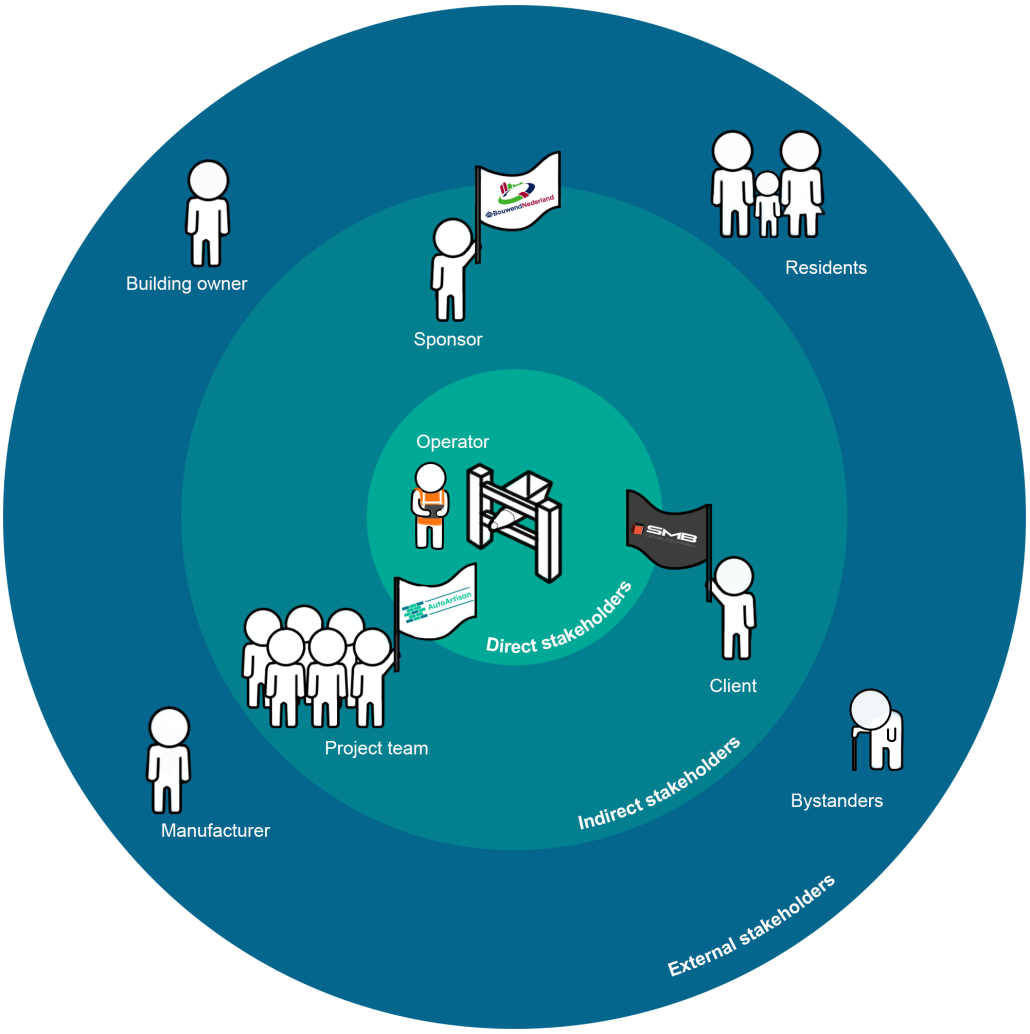


Figure 2.1: Stakeholder circle

2.1.2 Benefits & harms of the design

The main benefits and harms of the design for each of the stakeholders is tabulated below.

Stakeholder	Benefit(s)	Harm(s)
Operator	- Reduced physical strain - Operator can get more familiar with new technology step by step	- Risk of injury when robot malfunctions - Robot requires frequent monitoring
Developer	Hands-on experience in developing and integrating a full-scale system	Industry demand might change over time, which requires the design to be changed
Client	- Faster completion of jointing tasks - Reduced labor costs	- Uncertain outcome due to uniqueness of project - Result will likely have limited functionality
Sponsor	Benefits societal goals	Cost estimate is uncertain
Residents	Long term cost saving	Jointing quality is uncertain –>affects quality of living
Building owner	Shorter construction timelines	Jointing quality is uncertain –>affects maintenance costs
Bystanders	Not applicable	- Robot may cause injuries to bystanders in the case of a severe malfunction. - Robot may cause discomfort to bystanders
Manufacturer	Not applicable	Industry demand might change over time, which requires the design to be changed

2.1.3 Design for values

Values from the stakeholders mentioned before include:

- **Sustainability**: *refers to minimizing the environmental impact during production and use.* With regards to sustainability, a design requirement to improve in this area would be to design the robot such that it can easily be disassembled and recycled when it reaches the end of its lifespan. This approach is also called Design for Disassembly (DFD). One way to enhance the efficiency of the disassembly process is to take a modular approach in which components can easily be swapped out.

- **Safety**: *extent to which the user and bystanders are protected from harm.* To minimize the chance of injury caused by the robot, the design should include fail-safe mechanisms or an emergency button in case the power shuts down. Additionally, the toolbox should be shielded to prevent the user from accidentally reaching the tools while the robot is operating.

- **Simplicity**: *ease of use or manufacturing with minimal effort.* To make the robot easier to work with, we could provide the operator with an instruction manual or design stickers with visual cues or instructions on how to use the robot.

- **Quality**: *refers to a minimal amount of defects* In order to improve the quality of the joints, it might be convenient to constantly shake up the mortar mixture so that the consistency of the mixture stays relatively unchanged throughout the entire operation. This could be achieved using a vibration motor that is attached to the mortar extruder.

- **Comfort**: *absence of discomforting factors* In order to minimize noise pollution the design could include cushioning or suspension to absorb vibrational energy coming from the motors.

2.1.4 Value conflict

A possible value conflict arises between the operator and the sponsor: the operator prioritizes having many safety features implemented into the robot's design, while the sponsor

is focused on minimizing costs to keep the project within budget, which could inadvertently compromise the safety of the robot. One approach to resolve this conflict is to (re)design the components to withstand high forces before failure. Design-wise, this would mean incorporating fillets or ribs into the design that can enhance the durability of the components by redistributing excessive stresses and preventing areas that are prone to failure from breaking. Using Finite Element Methods (FEM) analyses, we could pinpoint areas within the components that lack strength. This prevents overengineering parts that are already strong enough. Another way that utilizes FEM-analyses which may improve the cost-effectiveness while maintaining the same strength is to simulate the same load scenario for different materials. For example, if, for a specific load scenario, an aluminum alloy appears to provide sufficient strength to withstand the load, then making the component out of steel would be redundant.

These possible solutions may lower the cost-estimate of the robot, whilst maintaining the same strength.

2.2 Use case and process

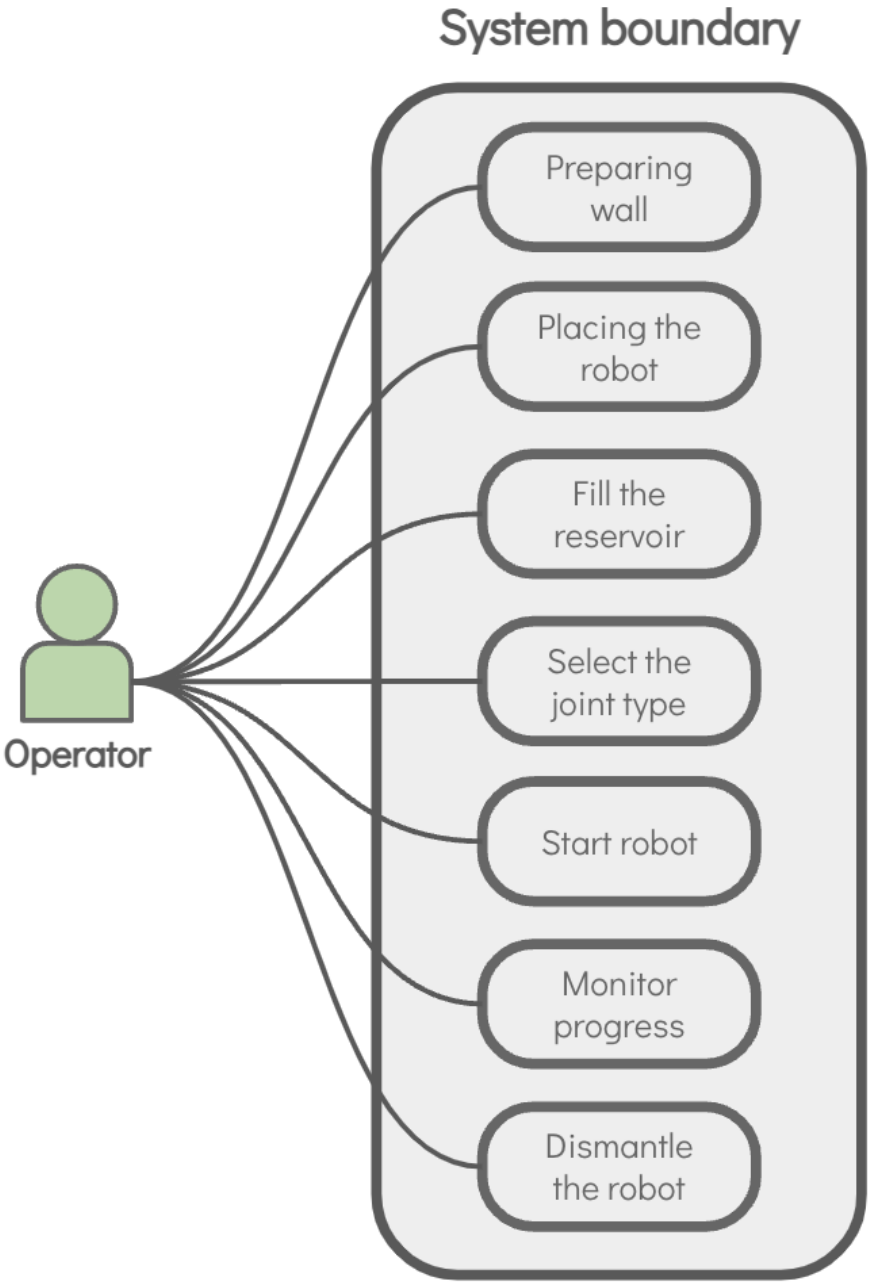


Figure 2.2: The use case diagram for the Jointing Robot

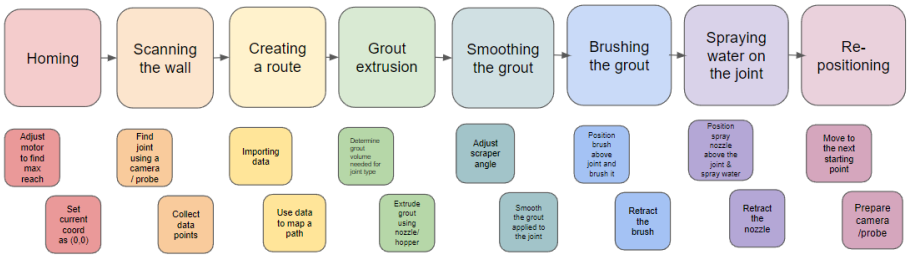


Figure 2.3: Steps and substeps

As can be seen from Figure 2.2, although the jointing robot has various stakeholders, the only stakeholder that will be directly interacting with the robot will be the operator. All the functions that the operator interacts with refer to different parts of the jointing process: manual set-up of the robot; choosing the settings; starting the robot and monitoring the robot. The 'Monitoring the robot' interaction refers to the operator responding to potential error messages that the robot alerts them of on the graphical user interface. Possible contents of error messages include: an empty reservoir; mortar build-up in the nozzle; significant variation in joint size; etc.) The operator may also need to manage uncaught errors (i.e., fixing improperly filled joints). The exact order of operation and the steps needed to accomplish the jointing task can be seen in Figure 2.3.

Our goal is to make the system as clear and intuitive as possible. From a CSE perspective, this goal can be achieved by adhering to the general principles of interaction design. The principles most relevant to our system design are Visibility of System Status; Helping Users Recognize, Diagnose, and Recover from Errors; User Control and Freedom and Documentation. Visibility of System Status refers to keeping the user informed about system operations (e.g., if an operation is underway there could be a progress bar displayed on the GUI, if an error has occurred there should be a message on the GUI alerting the user, etc.). To help the operator recognize and recover from errors, the error messages displayed should be as clear as possible, and suggest courses of action that can rectify the error, if applicable (e.g., "Error: Mortar cannot be extruded. Please clear the extruder nozzle"). User control and Freedom allows the user to recover from mistakes when interacting with the GUI - i.e., if they click a button to execute an operation, a window can appear asking them for confirmation that they would like to execute said operation. Documentation allows the operator to get more information about the system if necessary. For instance, on the statistics GUI, next to each metric, we could put an 'i' card that when clicked on, describes the implication of low/high values of that metric, and why that metric is relevant.

2.2.1 Human-robot interaction

Although an autonomous jointing robot makes jointing less burdensome for the operator, the operator remains an essential actor in the process of autonomous jointing. Before the robot is mounted onto the wall, the operator verifies that the two vertical bars are well-aligned, ensuring that both ends of the horizontal bar are leveled at the same height. Then, preferably with the help of two people, the operator secures the robot in place. The brackets along the vertical bars have slots instead of holes which means there is always the possibility of bolting into joints, which makes the mounting process easier for the operator.

When the robot is firmly fixed to the wall, the operator calibrates the origin by physically moving the robot such that its starting position is in the bottom left corner. Before extruding any mortar, the operator needs to set the necessary parameter values on the user interface of the robot. The user interface is discussed further in chapter 2.3.

The operator should always be near during the jointing operation to supply the robot with mortar when the reservoir runs low. When a failure occurs, the robot shows visual cues to alert the operator. The operation can then be paused from the UI. In case of an emergency, however, the operator will need to press the emergency button located on top of the toolbox, or on either vertical movement system.

When the jointing process is finished the robot needs to be shut down from the UI and unplugged from the power source. The operator then dismounts the robot from the wall and properly cleans the nozzle from leftover mortar. This ensures that the extruder

will not get clogged in the next operation. The remaining bolt holes in the joints can be filled manually. Finally, the operator safely stores away the robot until its next use.

2.3 Robomotive

The colour set and shape language used in our branding is consistent with that found in similar construction contexts. Features of construction machinery design that our robot will also incorporate includes bright primary colour, black highlights and in some cases exposed metal. Furthermore, dangerous elements or emergency mechanisms are often marked with red stripes, or retro reflective materials. Additionally, elements with high contrast often indicate direct user contact like buttons, handles and clasps. One example of how this can be integrated in the design of our robot is shown in
Figure 2.4:

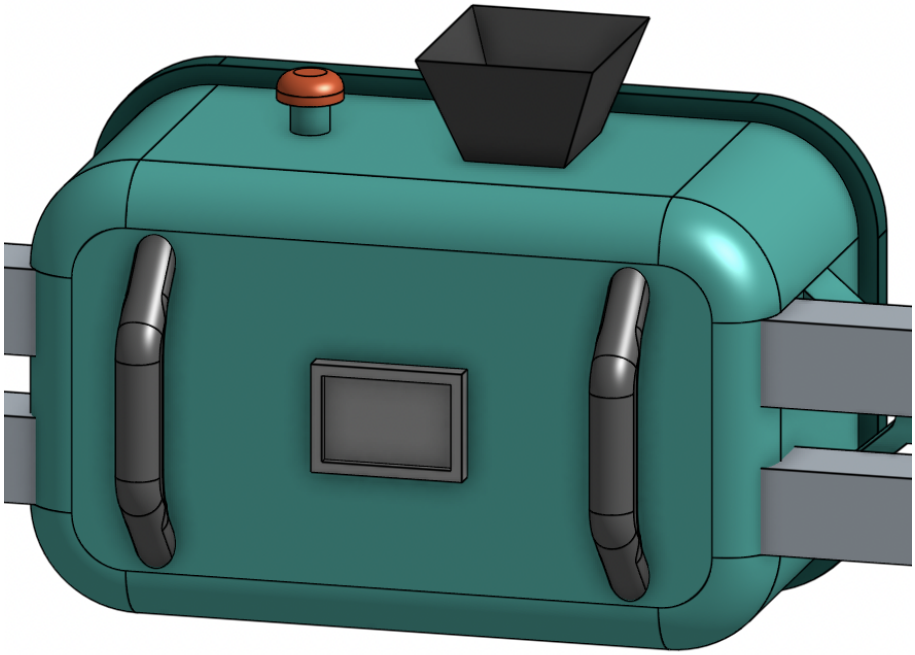


Figure 2.4: Ergonomics and robomotive design elements

The handles have high contrast relative to the rest of the chassis plus they are placed to promote safe and comfortable assembly of the robot. The mortar funnel is obvious and visually distinct from the rest of the robot body, to guide the user in accordance with proper operation. The user is shielded from the tool heads and moving parts by the chassis shell. The extra brim on the back edge reduces the chance of objects and or hands sliding between the wall and the tool heads. The emergency stop button is clearly visible, accessible and its purpose is communicated through design. The central digital display, which can be mirrored on a tablet or phone, is used for error messages and status update communication. A sample of how the user interface might look is shown in Figure 2.5.

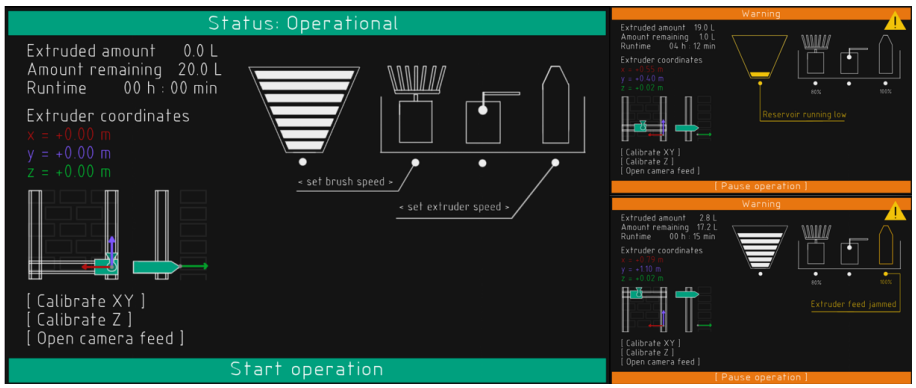


Figure 2.5: Sample user interface

3. Integrated blueprint

3.1 Mechanical and Electrical

3.1.1 Choice of motors

The robot will need powerful motors capable of lifting the horizontal section of the gantry up. This part will weigh about 20 kg. Additionally, these motors need to be able to be precisely controlled to achieve good jointing precision. The optimal solution therefore would be high-torque servo motors, but since these are expensive, we chose to use brushless DC motors (Flipsky 6354 140kv) controlled in closed loop with AS5047P encoders and a FSESC 6.6 controller. These motors are more difficult to set up and get working. Though, once they work, they offer more precision and power than an integrated servo motor at lower costs. To increase the torque, they are connected to a 20:1 planetary gearbox. Additionally, some smaller motors are required for mortar extrusion, brushing, and activating the smoothing tool. For brushing, a small 12V DC micromotor is used. Similarly, for activating the smoothing tool, a small servo is used which can provide enough torque to resist any unevenness in the joint. The motor for the mortar extrusion is not yet chosen, since the mortar extrusion system itself is not fully worked out yet. It will most likely, however, be a medium-sized BLDC motor.

3.2 Electrical and Software

3.2.1 Choice of sensors

`MAS_DMA_DPR_FUN_01, MAS_DMA_DPR_FUN_04`

To detect the wall and determine where the joints are, an appropriate sensor is needed. There are numerous valid choices, such as a normal camera, a laser distance sensor, or even an ultrasound sensor. These sensors could all work, but at varying accuracy and implementation intensity. Therefore, we chose to use a depth camera, namely the Intel RealSense D405. This camera provides both RGB data and depth data. This depth data is resilient to different lighting and weather conditions, which means the same algorithm to scan the wall can be used at all times. Additionally, an IMU will be integrated into the robot. This sensor can provide additional feedback on the movement of the robot which can improve positioning and will also help to make sure the different parts of the robot are properly aligned with each other during the setup by the operator. If the operator does not level the horizontal part of the robot properly, the robot might face issues, but with an IMU this can be detected and communicated to the operator.

To monitor the mortar reservoir level and make sure there is a proper flow of mortar out of the extruder, there are multiple solutions, of which the best is still to be decided. The most simple solution would be to measure the level of mortar using an ultrasound sensor mounted on top of the reservoir. Then, when the mortar decreases, the distance to the sensor will increase, and by taking the derivative you get the flow. This produces substation noise meaning a different solution might be necessary. Some examples are using multiple sensors or upgrading to a microwave-level sensor, which works on almost the same principle but is more expensive.

Finally, we have to think about safety. The mass of the robot combined with the power of the motors as explained in section 3.1.1 means there is a substantial risk of trapped limbs getting pinched between surrounding objects or floors and the robot. Therefore, there will be strips of anti-pinch sensors on all surfaces where there is a chance of getting pinched. These sensors will most likely be from the brand Mayser, which look like window insulation rubber strips. These

sensors work as pressure sensors, and as soon as they experience any resistance they will send a signal to stop and reverse the motors. Additionally, the torque experienced by the motors will be monitored, and if this is above an abnormal threshold, the motors are likely jammed and will shut off.

3.2.2 Choice of Microcontroller

The robot needs substantial processing power, capable of image processing as well as planning and controlling the movement of the robot, as compact as possible. After research, we chose the Raspberry Pi 5, 8GB version. This SBC offers a great performance-to-cost ratio and is one of the most popular choices in many non-industrial robotic projects. Its compact size allows easy integration into the design and it supports Ubuntu 24.04 to use ROS2. It might need to be upgraded with a CAN-hat for interfacing with the motors, but this is to be researched further. If we find later in the project that the RPi does not offer enough processing power, we have a Jetson Nano available to us for free with which we can offload the image processing to the GPU in that board. For now, we are trying to keep it at just the Raspberry Pi to minimize complexity.

Figure 3.1: Raspberry Pi 5

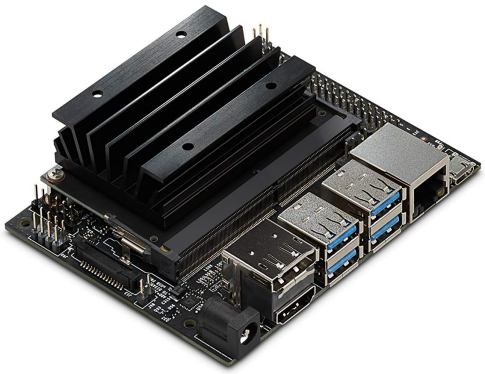


Figure 3.2: Jetson Nano

3.3 Software and Mechanical

3.3.1 Establishing a coordinate system & homing

MAS_DMA_DPR_FUN_03

During a homing sequence, Mason attempts to find the limits of its mobility and calibrate its position relative to static elements. At either end of the gantry movement range for both horizontal and vertical movement, there will be a physical buffer that the robot runs into. Upon collision with this buffer, the pinch point sensors are activated. The motors then stop trying to move in that direction and the current encoder value is stored as the new zero for the relevant coordinate. The X-axis points from the bottom left of the gantry to the right, horizontally along the wall. The Y-axis points up vertically along the wall from the same bottom left point of the gantry. The Z-axis is positive in the direction normal to the wall. Therefore, the point Mason attempts to describe as (0, 0, 0) is a point on the surface of the wall in the bottom left of the gantry.

3.3.2 Movement system; Motor control

MAS_JOI_SMT_FUN_01, MAS_USC_DIG_FUN_03

This section addresses the requirement MAS_JOI_SMT_FUN_01 When the robot is traversing the wall without extruding mortar, then its speed can be increased when the path contains fewer changes of direction. However, if the robot is moving along a path with lots of turns, then the speed should be kept at a more moderate level. However, if the robot is extruding mortar whilst moving, then speed should depend on the mortar levels left in the reservoir as well as the shape of the planned path. If mortar levels are higher, then the robot could comfortably operate at a higher speed without compromising the quality of the joints being filled (the more mortar there is in the reservoir, the more capable the robot is of maintaining a flow rate sufficient for filling joints at high speeds). However, when mortar levels are lower, the robot should operate at lower speeds, to avoid the operator having to fix low-quality joints with too little mortar applied. Thus when the robot is filling joints, the maximum speed decreases as mortar level decreases, which ensures a uniform finish.

Responding to user input

The current design for monitoring the robot's jointing (shown in Figure 2.5) includes an option to adjust extruder speed. To keep the finish of joints consistent, the speed of traversal should be somewhat proportional to extruder speed (i.e., if the user increases extruder speed, Mason should move faster). The operator can also select a joint type, which would involve at least a flush joint and a raked joint. This is done by changing the smoothing angle of the servo's. For a flush joint the smoothing tool is parallel to the wall and at the surface of the bricks. For a raked joint, the servos only need to be angled more such that the tip of the smoothing tool goes beyond the brick face ever so slightly. This results in a flush joint that is set back into the wall.

Potential failure detection

If a potential problem is detected (i.e., misalignment, IMU senses dangerous levels of tilting that indicate the robot may fall from the wall, etc.) the robot should automatically stop. Removing the need to manually stop the robot during erroneous operation, makes it easier for the operator to deal with whatever errors may arise.

3.3.3 Camera positioning

The camera has a ROS node /camera/camera/imu that can be subscribed to. This would allow the angle of the camera to be monitored and adjusted to make sure that the camera is always parallel with the brick wall (to ensure that the depth readings are as accurate and reliable as possible).

3.3.4 Controlling tools

MAS_JOI_ALL_FUN_01

A method of tool control could involve extending and angling the appropriate tool before the mortar nozzle starts extruding and the robot starts moving along the wall. This method would avoid potential damage done to the tool from trying to angle or extend it whilst the robot is in motion. This approach also minimizes the waste of mortar, as the tool is appropriately angled before extrusion, and the mortar has no time to fall out of the joint before it is smoothed. Tools will be retracted once inactive to not lock up against the bricks and inhibit motion.

4. Detailed Aspects

4.1 Software Design

4.1.1 Software Architecture

Robot Operating System

ROS2 Jazzy (Python) was chosen for the development of this project due to its support until 2029, ensuring future maintainability, as well as its extensive community package support. The ROS following packages are used:

1. **realsense-ros[4]:** A package for integrating Intel RealSense depth cameras with ROS2, providing access to depth, color, and infrared camera streams.
2. **ros2-control[2]:** A framework for controlling hardware interfaces and robots in ROS2, used for the management of actuators and sensors.
3. **gazebo-ros-pkgs[5]:** Enabling integration between ROS2 and the Gazebo simulation environment. *Used solely for testing and simulation*

Potential Python libraries include: numpy, opencv, scipy, open3d, Adafruit-GFX-Library[1]

ROS Nodes and Topics

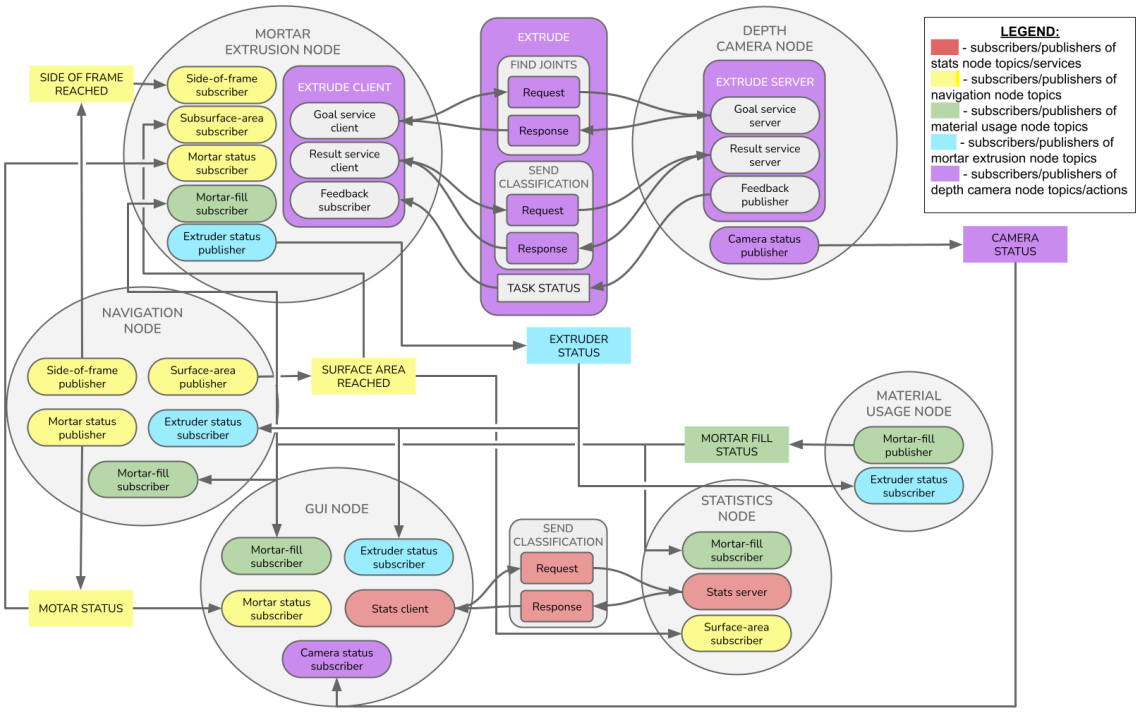


Figure 4.1: ROS Node layout

The node layout shown in Figure 4.1 was devised to make communication between the robot's subsystems as efficient as possible. As nodes should be responsible for a single modular purpose, there will be separate nodes for mortar extrusion, navigation, GUI, statistics, material

monitoring and the depth camera. In the current layout, the mortar extrusion node publishes to an extruder status topic; the navigation node publishes to motor status and frame boundary topics; the statistics node provides the 'compute stats' service; material monitoring publishes to the mortar fill status topic and the depth camera has the 'extrude' action and publishes to the camera status topic.
The GUI node is subscribed to all the status topics because it needs to display error and maintenance messages to the operator. The navigation node is subscribed to the mortar fill status and extruder status topics to prevent the robot from moving along joints when there is no mortar left (i.e., no unwanted gaps will be created). The reason the statistics node is subscribed to the mortar fill status topic is that it may be useful to measure the percentage of time spent refilling the mortar - such insight could help the operator determine how much of the time is being spent productively. The mortar extrusion node is subscribed to the motor status topic because if there is an error with the motor, the extruder should stop dispensing mortar.

User flow diagram

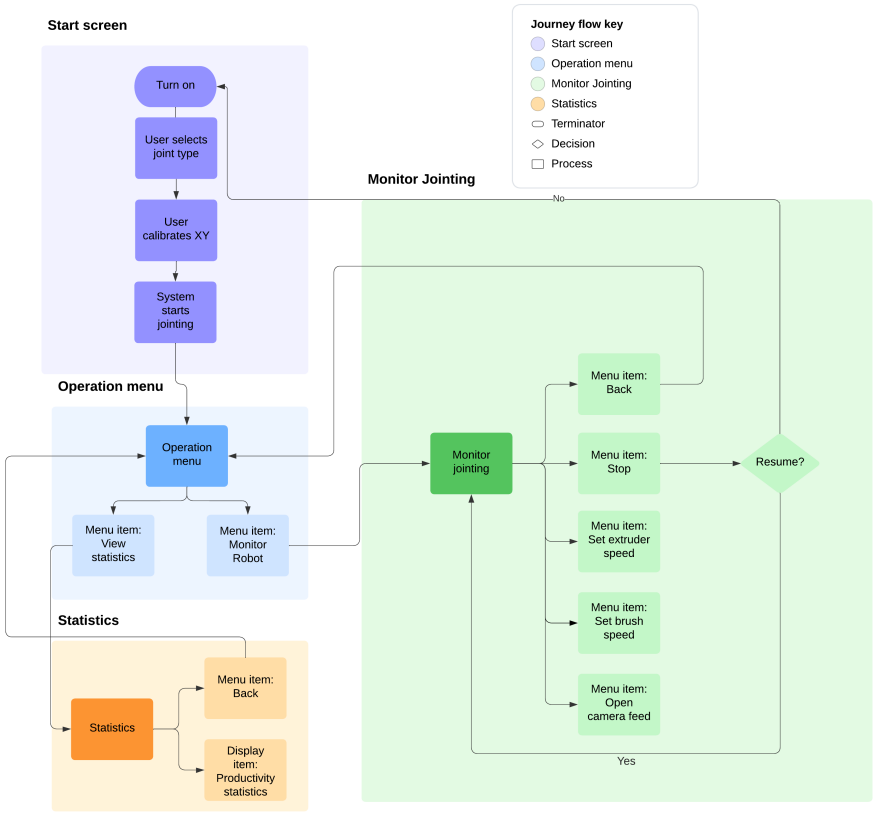


Figure 4.2: User flow diagram

The user flow diagram can be seen in Figure 4.2. When the robot is turned on, the operator is presented with a start screen, where they are prompted to calibrate the robot's XY coordinates and select a joint type. Until the operator does this, the robot will not start the jointing process. Once the jointing process starts, the operator will be able to interact with the operation menu. If the operator selects the 'Monitor robot' option, they are taken to the screen shown in Figure 2.5 where they can monitor the robot's jointing. If the operator stops the jointing session, a confirmation window will appear asking whether they want to resume the session. If they select 'Yes' then they are taken back to the Monitor Jointing screen. If they select 'No', then they are taken to the start screen, so they can set their preferences for the robot's next jointing session.

The operator can return to the Operation menu from the 'Monitor Jointing' screen. If the operator selects the 'View Statistics' option, then a screen that shows productivity statistics is displayed. These productivity statistics include percentage of time spent filling the mortar/dealing with maintenance alerts, the square meters of wall jointed so far in the day and the average jointing rate.

Simplified state diagram

A simplified state diagram of the jointing process is demonstrated by the Figure 4.3.

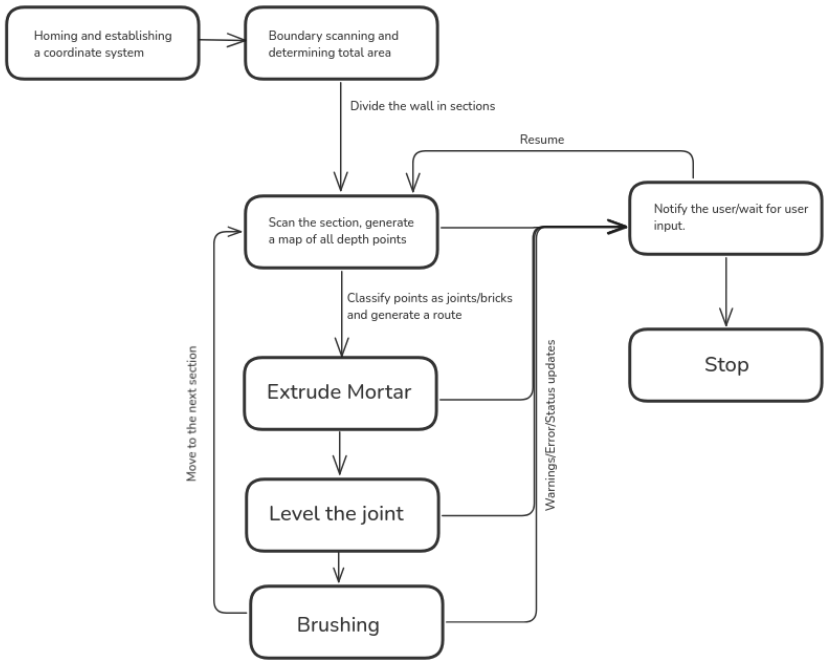


Figure 4.3: Simplified state diagram

4.1.2 Depth Processing

The following specifications and processing details are based on information from the Intel RealSense Camera 400 Series Product Family Datasheet [3]. The depth camera is capable of providing real-time depth images at various resolutions and frame rates (FPS). Below are important specifications of D405 sensor.

- **Resolution Options:**
  - **848 × 480** at **30/60/90 FPS**
  - **1280 × 720** at **15/30 FPS**
- **Field of View (FOV):** H-84° / V-58° / D: 92°
- **Other Specifications:**
- **Baseline:** *18 mm*
- **Camera distance from the wall:** 14 cm
- **Invalid Depth Band Ratio:** The depth data generated by the D4 Vision Processor utilizes the left imager as the reference for the stereo matching algorithm, resulting in a non-overlapped region in the camera's field of view. This non-overlapped region (at the left edge of the frame) contains no depth data. Based on calculations and testing, this non-overlapped region accounts for **4%**.

Resolution Selection and Processing

Since the maximum resolution of **1280 × 720** supports only up to **30 FPS**, it is optimal to use the resolution **848 × 480** with **90 FPS** to reduce motion blur. To reduce computational complexity and accurate depth values, the following post-processing filters are applied:

1. **Decimation Filter**
   Kernel size: `2 × 2`
   Reduces the depth scene complexity by down-sampling the image.

2. **Holes Filling Filter**
   Method: `nearest_from_around`
   Rectifies missing data by considering the four immediate pixel neighbors (up, down, left, and right) and selecting one based on a user-defined rule.

Final Image Pipeline

The image resolution evolves through the following steps:

$$\texttt{Initial Resolution:}\quad 848 \times 480 \xrightarrow{\texttt{Decimation (2 × 2)}} 424 \times 240 \xrightarrow{\texttt{Invalid Depth Band Cut}} 400 \times 240$$

4.1.3 Brick detection

`MAS_DMA_DPR_FUN_04`

Requirements

- Must not be too computationally intensive
- Must not depend on knowledge of the speed at which the camera is moving
- Must be able to clearly detect edges using data collected from the Intel Realsense D405 (i.e., color, depth data, IR sensor data, Point Cloud data, etc.)

In this sub-section, three solutions for brick detection have been tested and compared. This is because the efficiency and reliability of the robot's brick detection affects its ability to perform the rest of its tasks (i.e., if the robot cannot detect bricks efficiently, it cannot map a path or extrude mortar into joints).

Solution 1: Fourier Analysis

The edges detected using this method can be seen in Figure 4.4. Whilst this method is more efficient than iterating through each pixel of the frame, the edges detected are also fainter, and the image is more noisy in general.

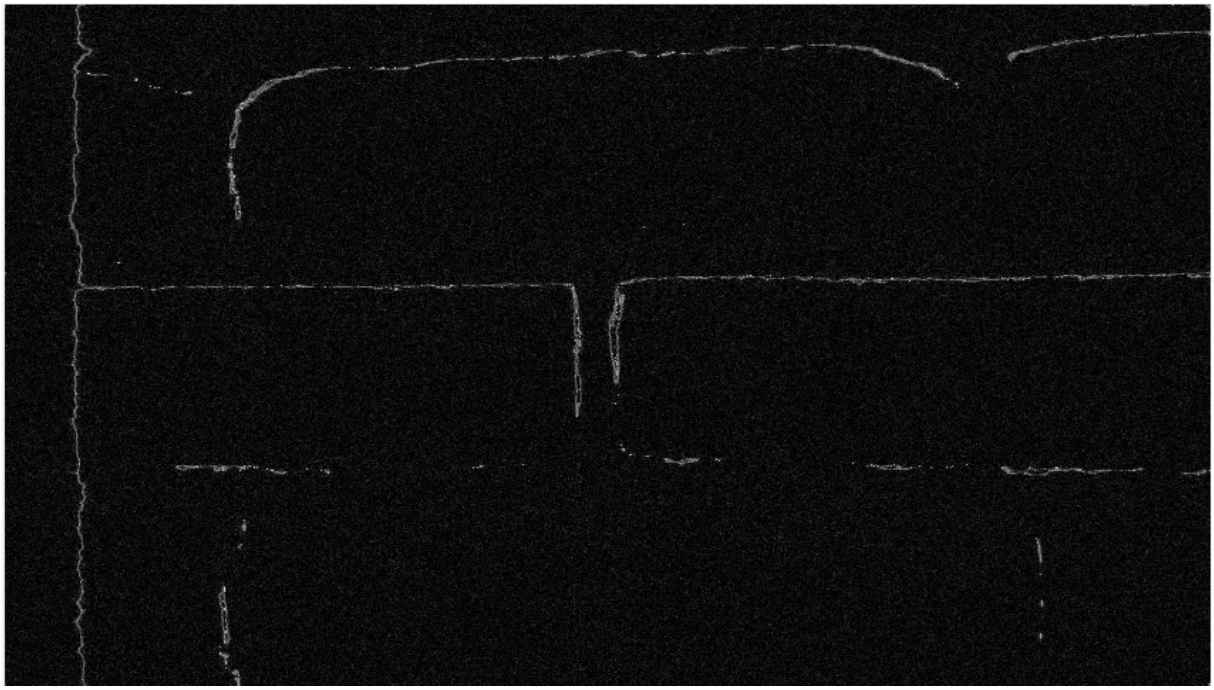


Figure 4.4: Edges detected via Fourier analysis

Unlike a straightforward band-pass circuit filter, the Fourier transform of a real image contains a more complicated set of frequencies that correspond to spatial patterns in said image. So unlike a circuit filter, the cut-off frequencies are more arbitrary than simply finding the points where the Log of the frequency is -3dB. Thus, there were two methods that were tested to find the low and high cut-off frequencies:

1. Cumulative Amplitude Threshold (SUMMARY): Calculate cumulative energy in the Fourier spectrum. Cut-offs are determined based on a certain percentage of the total energy. This approach preserves the most significant frequencies

2. Adaptive Threshold based on Image statistics (SUMMARY): various Image statistics were tested as thresholds (mean, median, quartiles), and the low cut-off was determined by finding the lowest index to exceed the threshold and the high cut-off was determined by finding the highest index to dip below the threshold. This approach was not more successful in edge detection than the Cumulative Amplitude approach. In fact, there was more flickering between frames. This was because changes in the image content caused shifts in amplitude and resulted in inconsistent cut-off frequencies.

Solution 2: Threads running edge detection on frame sub-areas

In this method the frame is split into a certain number of sub-areas and an edge mask is computed for each sub-area in parallel, then the edge-masks are combined to make an edge mask for the entire frame. A SubArea object was created with attributes for the infrared sensor readings for that region, depth camera readings for that region, an edge mask (initialized as an empty array) and an index value that indicates where in the frame the sub-area is located (as shown in Figure 4.5).

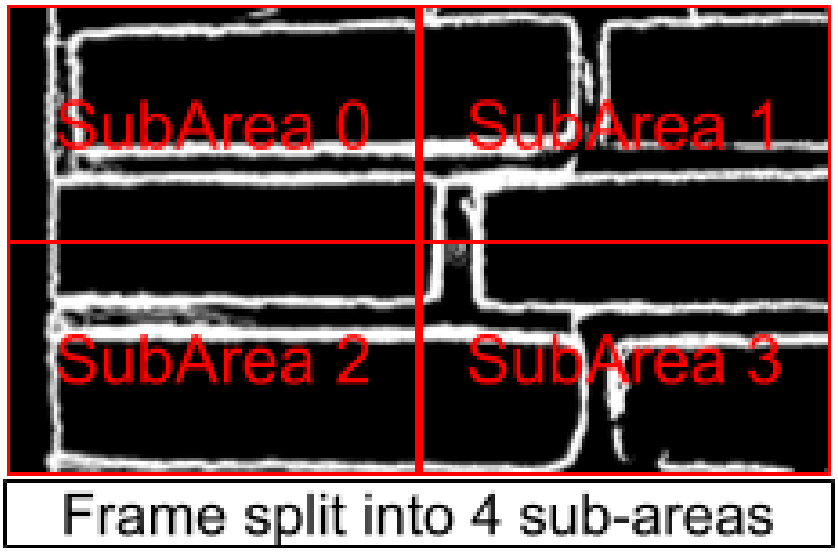


Figure 4.5: Frame split into different numbers of sub-areas

The edge mask is computed using Sobel operators, which combine Gaussian smoothing and differentiation which produces a result that is resistant to noise.
The edge mask is then used to compute the horizontal line and vertical line mask. The corner mask is the bit-wise and combination of the horizontal line and vertical line mask. The corners of the bricks are determined by computing the centroid of the corner mask's contours. The contours' centroids are computed using spatial moments in the following equations:

$$c_x = \frac{M['m10']}{M['m00']}$$

,

$$c_y = \frac{M['m01']}{M['m00']}$$

where $M['m00']$ is the contour's area, $M['m10']$ is the sum of the contour pixels' x coordinates and $M['m01']$ is the sum of the contour pixels' y coordinates.

Visual representations of these stages can be seen in Figure 4.6.

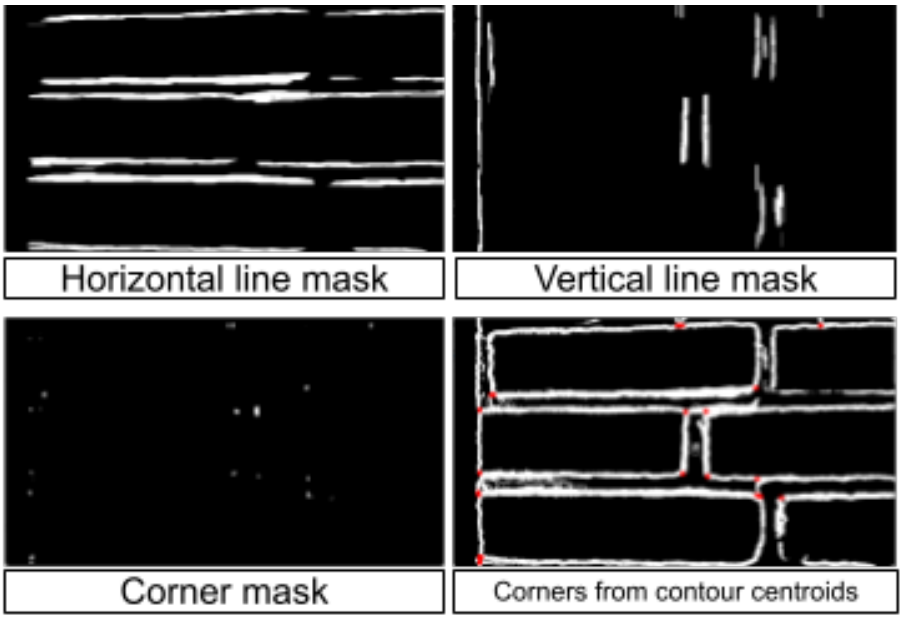


Figure 4.6: Stages of obtaining brick corners from edge mask

	Time taken per frame (in seconds)			
Subareas	1	2	3	Avg.
4	0.229543699	0.173789111	0.187760303	0.197031038
16	0.046610605	0.201309501	0.247681199	0.165200435
64	0.055736002	0.239249856	0.236576603	0.177187487

The benchmark table shown in Figure 4.1.3 is used to compare the performance time of the algorithm per frame for different numbers of sub-areas. From this table it seems that on average 16 subareas would be the optimal number to have, as after that, the time taken to concatenate the sub-areas starts to counteract the time saved by processing sub-areas in parallel.

Solution 3: Heuristic based approach

- **Depth Analysis:** Based on testing, the average distance along the z-axis between the joint and the brick is approximately 12 mm. The average minimum depth from the camera to the wall is 140 mm, and the maximum depth is 170 mm. To capture both the brick and joint pixels, a 40×40 (the minimum brick height/width in the frame is 50 mm, which corresponds to 30 pixels.) pixel window is created around each pixel.
- **Labeling Function:** The labeling function used to distinguish between the joint and the brick is defined as:

$$\text{Label} = \begin{cases} \text{Joint} & \text{if } \frac{\text{currentDepth} - \text{minDepth}}{\text{maxDepth} - \text{minDepth}} \geq 0.5 \\ \text{Brick} & \text{otherwise} \end{cases}$$

  Where minDepth and maxDepth are calculated within a square window of size 40×40 pixels around the current pixel. An example of the above function can be seen in the Figure 4.7.

- **Performance optimization**:
  - As each frame is processed independently, utilizing multi threading will speed up the processing.
  - A brute-force algorithm has a time complexity of $O(H \cdot W \cdot K^2)$. However, by employing a 2D sliding window algorithm, we can optimize the process, reducing the time complexity to $O(H \cdot W \cdot K)$.
    H: Height, W: Width, K: Window Length
- **Resolution:** The depth analysis is performed for a resolution of 424×240 pixels.

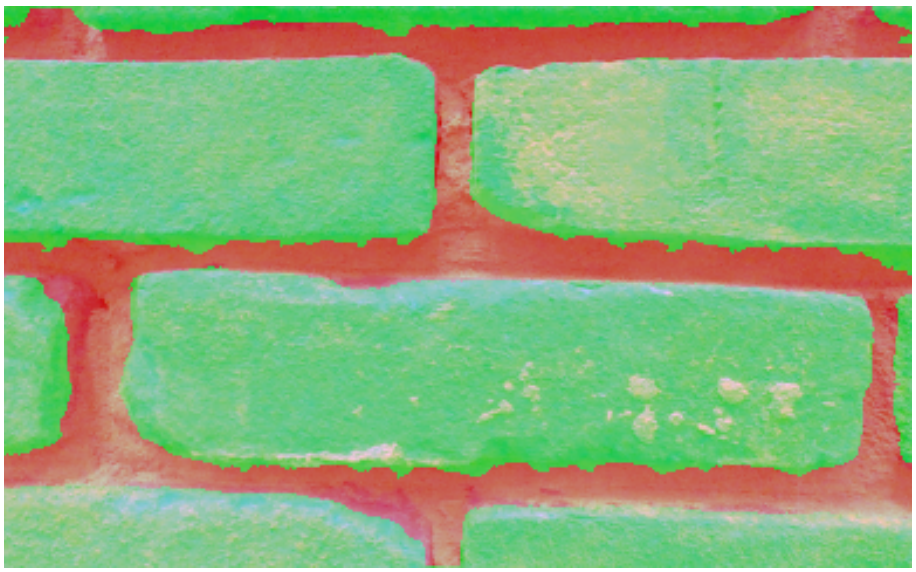


Figure 4.7: Labeled frame (Joint: Red, Brick: Green)

Conclusion

All three solutions are equally feasible in terms of complexity and performance. The final decision on one or more of these approaches will depend on testing with the actual robot to determine which aligns best with the other components and overall system requirements.

4.1.4 Relationship between pixels and coordinate system

As outlined in Section 3.3.1, the robot will first establish a coordinate system by performing a homing procedure. It will then traverse the wall in a zig-zag pattern, capturing depth frames along the way. The camera's origin will be determined using data from the IMU and encoders, enabling the calculation of pixel coordinates in the frame based on the depth information. Since each point on the wall will be captured by multiple frames it is ideal to normalize the pixel value for a specific point using values calculated from consecutive frames. Based on the following calculations, the total number of pixels for scanning the entire wall at once is approximately 27 million. Processing and storing such a large volume of data would be computationally expensive and memory-intensive. Therefore, it is more efficient to scan and process the wall in smaller sections first, rather than attempting to handle the entire wall in one go.

$$\text{Total pixels} = (H \times W) \times \frac{R}{F_H \times F_W}$$

Where:

- H is the height of the wall (1100 mm),
- W is the width of the wall (1000 mm),
- F_H is the height of a frame (15 mm),
- F_W is the width of a frame (24 mm),
- R is the resolution of the frame (400 x 240 pixels).

4.1.5 Track m^2 jointed

`MAS_USC_DIG_FUN_01`

By using data collected from encoders, it is possible for the robot to record the distance traveled in real time. By using the encoder resolution (i.e., ticks per revolution) and the wheel circumference, the distance traveled per tick can be calculated using the following formula:

$$Distance\ per\ tick = \frac{Wheel\ Circumference}{TPR}$$

The jointed area can be calculated in the following way:

$$Area\ jointed = Ticks\ counted\ \times\ Distance\ per\ tick\ \times\ Joint\ width$$

The tick count can be updated periodically when the flow rate detected by the flow sensor in the mortar extruder exceeds a certain threshold. Therefore, tick count is only updated when mortar is actually being extruded (by combining data collected by the flow sensor with data collected by the encoder, we ensure that we are tracking square meters of wall jointed as opposed to square meters of wall traversed).

If the operator stops the robot mid-session then the statistics node (the ROS node responsible for tracking the square meters jointed) will publish the square meters jointed for that session. Then, the square meters jointed in that current session would be saved to a log file along with the session start timestamp and the session duration. Then the square meters jointed is reset to 0 and the daily total for square meters jointed is updated. Then:

- If the operator resumes: the next session start timestamp will be when the operator resumes the session
- If the operator does not resume: the next session start timestamp will be when the operator starts a new session

By resetting the square meters jointed to zero at each pause and logging the previous session data, potential discrepancies between the timestamps and durations are eliminated, and the workflow stays consistent.

At the start of every new day (12:00 AM) the daily total should reset and the previous day's data should be archived.

Error handling

If a session is interrupted because of an error, the error should also be recorded in the logs, so that when the data is reviewed there is a distinction between an intentional session break and an unintentional one. In order to avoid losing data, partial session data should be saved in real time (e.g., if the IMU senses the robot tilting at a dangerous angle that indicates it might fall from the wall, current data should be logged immediately).

Reasons for choosing this approach

- Precise tracking: allows productivity analysis at finer levels. Such specific data can be used to analyze trends (peak productivity times, bottlenecks, jointing rate per session, etc.)
- Monitoring progress: allows the robot to keep track of how close it is to its daily jointing target regardless of how often a session is paused or a new session started
- Facilitates error recovery: If there is a system failure, because the data is logged, previous progress is easier to recover

4.1.6 User Interface and Operator alerts

MAS_USC_DIG_FUN_02, MAS_USC_DIG_DES_01, MAS_USC_DIG_FUN_04, MAS_USC_DIG_FUN_03, MAS_USC_DIG_FUN_01

A display will provide vital information about the robot's status, including warnings and error messages. In combination with hardware buttons, the operator will be able to navigate the user interface and interact with various settings or functionalities.

The operator may see a few different kinds of alerts such as:

1. A notification when they have reached the daily target of 30 m^2
2. Maintenance messages:
   2.1. Every 100,000 rotations of the robot's wheels, a maintenance message will be sent to the operator to inspect the wheels and ensure they are not too worn out to function.
   2.2. Similar to the wheels, there could be a scheduled maintenance message that prompts the operator to check whether the joint finishing tools need replacing. This notification will be displayed every $100m^2$ jointed
3. Error messages:
   3.1. If something is blocking the gantry, this alert will be displayed: "Obstruction detected. Initiating emergency shut-off"
   3.2. If there is some mortar build-up blocking the extrusion nozzle, this alert will be displayed: "Blockage detected in mortar extrusion nozzle, unable to extrude mortar"
   3.3. If the camera is on but no bricks are detected this alert will be displayed: "No bricks were detected. Please inspect the camera to ensure it is not obscured or damaged."
4. Warning message:
   4.1. IMU detects the robot tilting at an angle that indicates it might fall from the wall the following message will be displayed: "Robot is tilting at an unsafe angle, please attach the robot to the wall more securely."
   4.2. When mortar reservoir is below 20% capacity, the operator will get the following alert: "Mortar reservoir levels are low. Please refill it."
   4.3. If the frame is misaligned when the robot is set-up the following alert will be displayed: "Robot must be properly aligned before operation."

4.1.7 Routing

MAS_DMA_DPR_FUN_05

From the calculations in Section 4.1.4, it is ideal to divide the wall into sections. Given that there are more horizontal joints, it is preferable to divide the wall into horizontal sections. The robot can then joint the horizontal joints in each section first, followed by the vertical joints. In the Figure 4.8, wall is divided in subsections by the green boxes, and the robot will follow the white arrows and then the blue arrows.

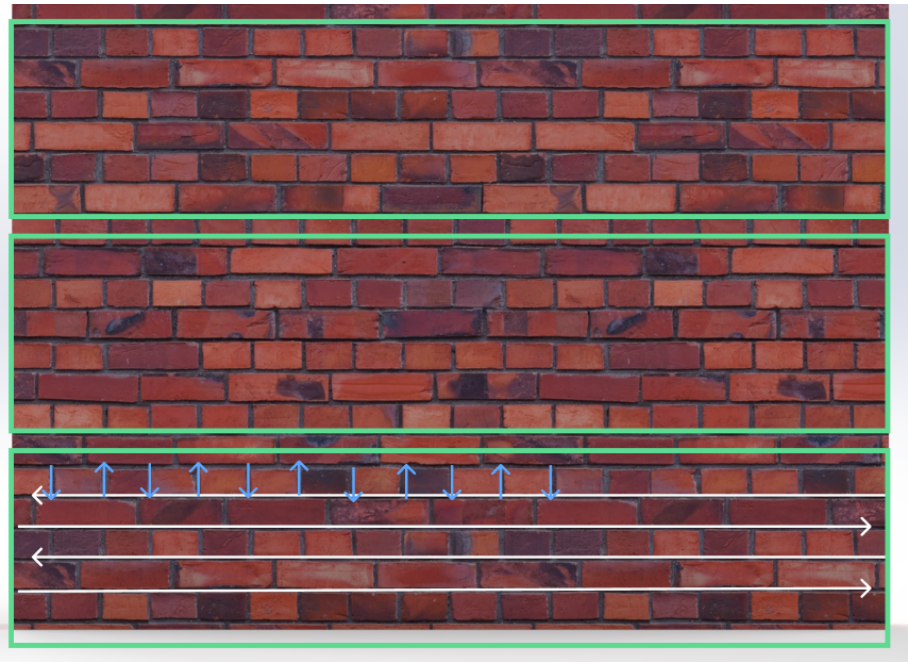


Figure 4.8: Example routing

4.2 Mechanical Design

4.2.1 Stability

MAS_GAN_STB_FUN_01

To make sure that requirement MAS_GAN_STB_FUN_01 is satisfied it is essential that the gantry can withstand the reaction force generated when the tools make contact with the wall. The gantry must maintain parallel alignment with the wall, or risk improper jointing or inhibited mobility. One way to do this is to fix the gantry to the wall. Mason currently includes brackets which are designed to attach to the vertical bars using M8 bolts and T-nuts. The brackets also feature M8-sized slots which allow for more spatial tolerance than M8-sized holes. The tolerance is necessary for the bolts to always have the option of mounting into joints (Figure 4.9). Drilling into the joints is preferred since the operator can plug the holes with mortar when done and leave no trace. The brackets hold the vertical gantry bars far enough from the wall to allow the wheels to fit in between.

In Mason's current design the bracket is 3D-printed, because it contains fairly complex geometry such as fillets and ribs. In the case that the gantry requires more rigidity, the brackets could also be manufactured from multiple sheet metal parts that are joined together using welding techniques. Alternatively, if welding is not feasible, the design could be modified to replace the rib with foldable lips, which would make fabrication easier without compromising the bracket's strength.

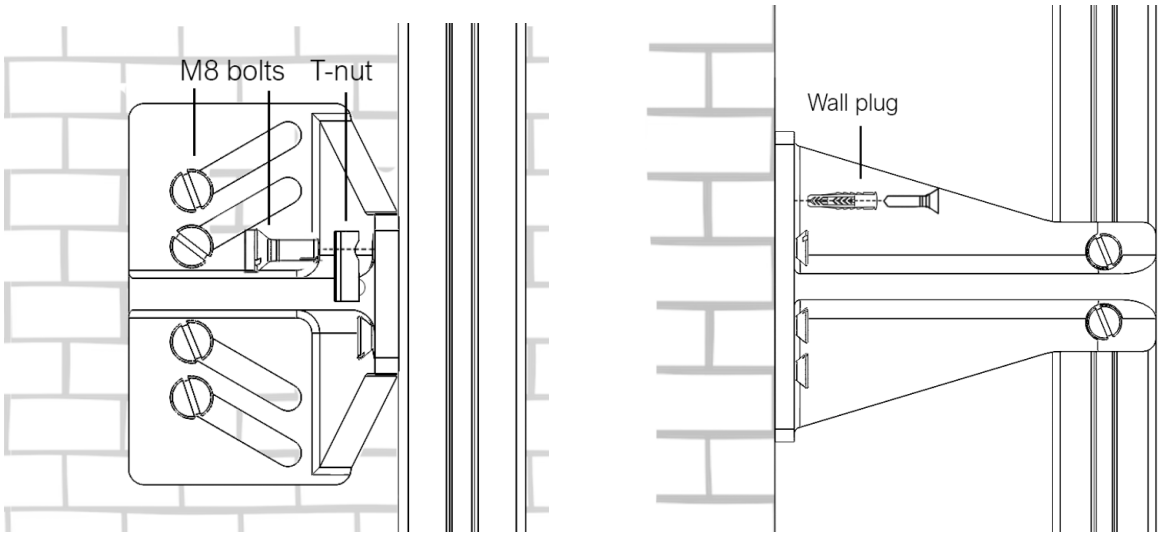


Figure 4.9: Front & side view of the 3D-printed bracket

On the profiles themselves, there are spirit levels also attached with T-nuts. These help the operator mount the gantry profiles vertically, and most importantly, parallel to one another.

4.2.2 Tools

MAS_JOI_EXT_FUN_01, MAS_JOI_SMT_FUN_2, MAS_JOI_SMT_FUN_3, MAS_JOI_BRS_FUN_01

To enable a correct jointing process, three tools are used. Namely, a mortar extruder, a smoothing tool, and a brush. These will all need to be able to be controlled individually. There are also several smoothing tools oriented differently to satisfy MAS_JOI_SMT_FUN_2 and MAS_JOI_SMT_FUN_3. The arrangement of tool heads and the depth sensor can be visualized with Figure 4.10.

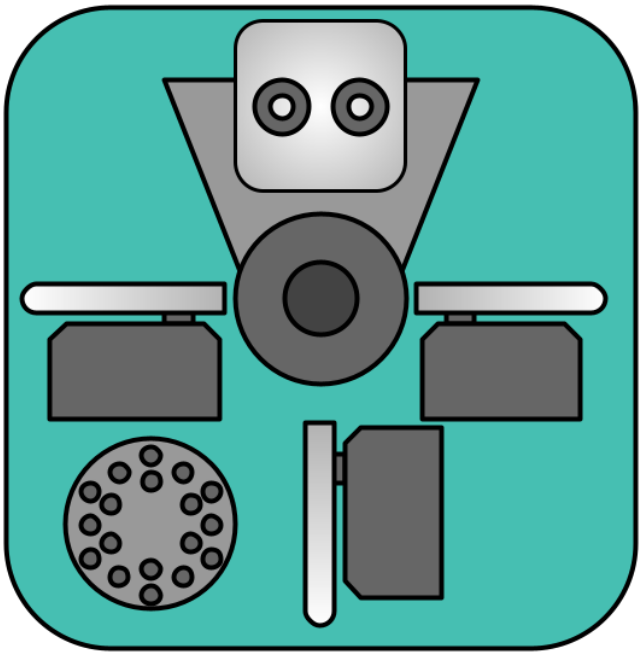


Figure 4.10: Tool head arrangement

There are three smoothing tools around a central extruder nozzle. This allows for smoothing directly after mortar extrusion when the chassis is moving up, left, or right. There is one brush mounted on the bottom instead of after every smoothing tool which reduces the space occupied by the tool heads as a whole and increases the effective jointing area with the same gantry setup. The mortar extruder is placed centrally under the depth camera so that everywhere the nozzle can extrude is visible to Mason.

For the mortar itself to get injected into the gaps between bricks, it is first loaded in a hopper behind the camera. This hopper leads down to a spiral screw. When the screw is rotated clockwise, continuous small amounts of mortar are dragged from the hopper and towards the nozzle. At the nozzle, the mortar gets compacted and ejected right into a joint. That is how Mason satisfies MAS_JOI_EXT_FUN_01: Robot shall extrude mortar.

Whilst the brush is not active in the same jointing path as the extrusion and smoothing, the operation does still happen in a later pass. Therefore Mason does still accomplish MAS_JOI_BRS_FUN_01: Joints shall be brushed after smoothing. The decision to have a larger jointing area instead of fewer passes over the same area is one informed by the specifics around the brushing effect. Brushing the joints does not require a slow, precise or methodical approach. It can be done in large sweeping motions, covering a general area. This can happen rapidly so the brush would be limited by needing to wait for the other tools to finish their slower parts. An extra pass does add more time overall but the addition is relatively tiny compared to the initial jointing pass. An extra jointed width of about 100 mm on a wall that might be as high as 10 meters or more could save the operator a whole square meter, per set up. This extra area consists of mostly slow vertical joints they would otherwise have to do by hand. Doing this with mason likely saves time in the end and certainly saves effort.

4.2.3 Movement

`MAS_GAN_MOB_FUN_01, MAS_GAN_MOB_FUN_02`

The tool heads move horizontally along two aluminium profiles. There is a single powered wheel riding on the top profile. The wheel engages with the profile through friction between the

soft 3D printed plastic wheel and a rubber coating on the profile. The wheel has a central spine which interfaces into the slot of the profile, helping with alignment and preventing movement in the Z axis. On the underside of the bottom profile, two similar underpowered wheels are pushed apart by a spring and into the profile, providing more pressure for the powered wheel. A visualization of this set up can be seen in Figure 4.11.

Figure 4.11: Horizontal gantry arrangement

Figure 4.12: Vertical gantry arrangement

As can be seen in Figure 4.12, the Y axis movement of the gantry works on similar principles. There is one powered wheel tensioned onto the profile relative to 3 other wheels. All wheels have the same central ridge to align with the profile direction and prevent lateral movement. What is different here is that the powered wheel is not on a rigid axle. It is mounted on a pivoting arm which has a torsion spring which presses it into the profile. Furthermore, the geometry is such that when the motor is static, the weight off the horizontal gantry presses the wheel further into the profile, providing more grip and preventing slippage. When the motor provides torque to move up, the wheel pushes the entire horizontal section upwards. The vertical movement arangement is duplicated for the other vertical extrusion and their movements are digitaly syncronised. The overall set up of the gantry can be seen in Figure 4.13. With this it becomes apparent that MAS_GAN_MOB_FUN_01 and MAS_GAN_MOB_FUN_02 regarding the independent horizontal and vertical movement of the gantry is satisfied by this design.

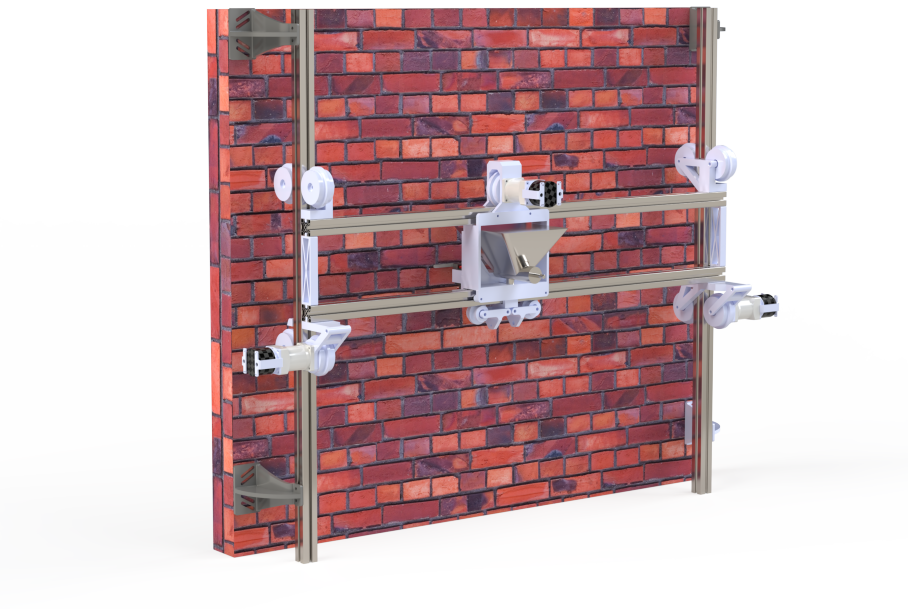


Figure 4.13: complete gantry arrangement

4.2.4 Tool maintenance

MAS_JOI_SMT_PER_01, MAS_JOI_SMT_FUN_04

Over time, the smoothing tools will abrade from hard contact with the coarse mortar and hard ceramic bricks. To make sure the smoothing tools last as long as they can, Mason uses the tips of standard commercial jointing irons made from 8mm x 3mm x 65mm spring steel. These are made to last moths on traditional masonry job sites, where they can be used for at least 30 m^2 a day and see heavy handling. Ours only need to last for about 3.34 full days of target jointing to satisfy MAS_JOI_SMT_PER_01.

Should the tool break and need replacing, the standard dimensions mean the operator can simply undo the two set screws on the servo adapter to release the old smoother, slot in a new one, and tighten, to fully replace the smoothing tool and no other part. This way, Mason also complies with requirement MAS_JOI_SMT_FUN_04.

4.2.5 Operation on standard wall

MAS_GLB_ALL_DES_04, MAS_GLB_ENV_PER_01

The test surfaces available to us are two walls of about 1.2 m high and wide. In most real life cases, Mason would be working on much larger single areas. In this design, those walls can be covered by remounting the setup built for the test surface over and over, or the extrusions could have another copy joined end to end, effectively expanding the gantry footprint in a modular fashion. Current aluminium profiles used for the gantry are 1.25 m long. In order to still be compliant with MAS_GLB_ALL_DES_04, which is a maximal component size requirement, the profiles can not simply be replaced with longer variants. With the current modular gantry, there is no limit on the horizontal or vertical size of walls that Mason can operate on in one set up apart from the length of wires and maximal mass movable by the motors. The smallest possible gantry set up fits on the test wall, and larger walls can be covered with multiple set ups of the same gantry or with expanded profiles. Therefore, requirement MAS_GLB_ENV_PER_01: System shall operate consistently on vertical straight walls ranging from 1.2 m tall to multi story is satisfied.

4.2.6 Water & dust resistance

MAS_GLB_ALL_DES_01, MAS_GLB_ALL_DES_02, MAS_GLB_ALL_FUN_01

Mason needing to be rated to IP65 means it is resistant to small particle intrusion meaning dust, and is capable of withstanding being sprayed with water. To accomplish this, the casing must enclose large sections. Bolts and fixtures connecting the casing to internal parts of the shell are equipped with rubberized seals. Parts where internal components must interact with the outside such as the tools will be partitioned by the casing. Mounting points and electrical elements are internal and the moving parts outside the casing will have a close fitting hole. This does mean that the casing has to be removed before maintenance of parts can happen, but without this separating casing, water and especially dust can seep into the machine unhindered. All external metal components are either aluminum or stainless steel. Otherwise, water exposure could accelerate corrosion. Internal structural elements of the casing are made from sheet metal which can itself be vulnerable to corrosion, especially in cases of welding or bending parts. For this reason the sheet metal parts will receive a protective coating. External elements of the casing will largely be 3D printed plastic in this prototype version. Production ready versions of Mason might replace those parts with injection molded components. In any case they will have a nonporous external structure, providing a physical barrier to water and dust. Lastly the structure of the casing strikes a compromise between cushioning, and rigidity. In accordance with MAS_GLB_ALL_FUN_01, The casing is sturdy enough that deliverance of a small force does not permanently deform is but also provides enough cushioning that the important internal components do not bear the damage.

4.2.7 Maximum mass & dimensions

MAS_GLB_ALL_DES_03, MAS_GLB_ALL_DES_04

In order to have the robot be easily carried and set up by a single person, each of the components of the robot should weigh no more than 20 kilograms. If this requirement is met, MAS_GLB_ALL_DES_03 will be satisfied. This weight limit also allows for safe and ergonomic lifting, which minimizes the risk of strain. Furthermore, to make the robot easily transportable it should fit comfortably within a 'standard' van. This means that the robot's dimensions should not exceed 350 cm in length, 175 cm in width and 190 cm in height, satisfying MAS_GLB_ALL_DES_04.

4.3 Electrical Design

The electrical system will act as the robot's nervous system, ensuring all components get power, the motors can be controlled, and sensors can be used by the controller. This section will dive deeper into the design choices for specific components and how they will work together.

4.3.1 Simplified System

The following figure shows all components and their connections. Some parts are simplified or still unclear as to how they are connected exactly.

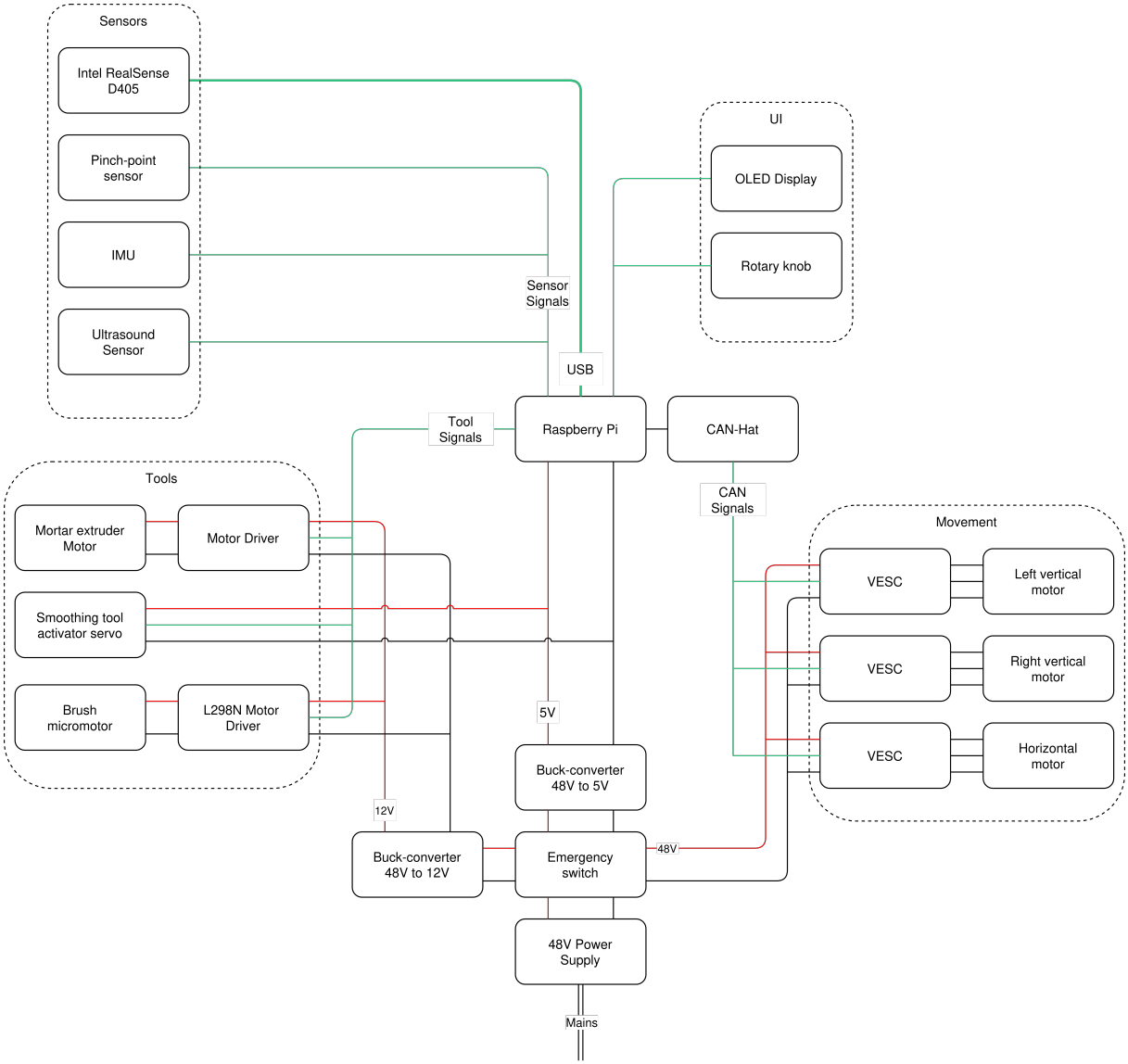


Figure 4.14: Electrical system

4.3.2 Power

For power, a 48V DC power supply will be used. Since the robot is static to the wall, a battery is not practical and will only add substantial extra weight, complexity, and safety concerns. Therefore, a DC power supply is the better option. The main motors and VESCs are rated for 12 to 60 volts and therefore 48 volts was chosen to minimize resistive losses ($P_{loss} = I^2R$) while still keeping some safety margins for voltage spikes. Not every component can work at 48 volts, however, so there need to be two step-down buck converters. One to step down to 12 volts, for the brush motor and the mortar extrusion motor and one to step down to 5 volts for the Raspberry Pi and all sensors. These buck converters will be slightly less efficient at 48 volts compared to 24 volts for example, but since the power for all other components than the main motors is only a fraction of the whole system, this is still the most efficient solution.

Power estimation

In the following table, all components are listed, and their average power. Some sensors are omitted because their power consumption is negligible.

Component	Voltage (V)	Average Power (W)	Quantity
Raspberry Pi	5	25	1
BLDC Motor	48	150	3
VESC	48	5	3
Micromotor	12	15	1
Mortar motor	12	50	1
Smoothing servo	5	15	3
Intel RealSense D405	5	1.5	1
OLED Display	3.3	0.1	1
Total		601.6	

Therefore a 48V DC Power supply capable of providing about 600 to 700 watts is required, but to have some margin, a power supply capable of providing 1000 to even 1500 watts is ideal. This is so that when the motors experience extra torque, they have the power needed to resist that torque.

4.3.3 Movement

As touched upon in section 3.1.1, 3 BLDC motors will be used to enable the movement of the toolhead around the wall. These motors will be controlled by FSESC 6.6s, based on the VESC project. VESCs are highly customizable motor controllers that can control BLDC motors for speed control, torque control, and position control. We will be using the latter, assisted by AS5047P encoders. Using the VESC, you can precisely control the rotor position in degrees, and interfacing with the Raspberry Pi can be done using the CAN protocol after connecting a CAN-hat.

4.3.4 Sensors

`MAS_DMA_DPR_FUN_01, MAS_DMA_DPR_FUN_02, MAS_DMA_DPR_FUN_04`
The robot needs sensors for the requirements MAS_DMA_DPR_FUN_01, MAS_DMA_DPR_FUN_02, and MAS_DMA_DPR_FUN_04 as well as general safety sensors. To solve the first requirement, an ultrasonic sensor is chosen. This offers a non-intrusive solution to the problem since the mortar reservoir can be closed with a lid on which an ultrasonic sensor is mounted and by monitoring the distance that the sensor reads, the volume of mortar remaining in the reservoir can be calculated. By taking the derivative of these readings, it can double as a primitive flow

sensor. For the second requirement, there will be multiple sensors working together. First, every motor will have an encoder with which the distance traveled from the homing point (that is, point (0, 0, 0)) can be determined. On top of that, an IMU will provide extra feedback and even the RealSense camera could provide positioning information if needed. That also brings us to the last requirement, detecting where the joints are in the wall. For that, a depth camera was chosen, particularly the Intel RealSense D405. Most stereo depth cameras have a high minimum distance to the scanned object for it to be accurate since the distance between the two cameras is high. Just like when you bring your finger close to your eye, you can not properly focus on your finger. Therefore, the distance between the cameras is lowered for the D405, providing accurate depth data on close range and thus the D405 is ideal for our problem, since the difference in depth between a brick and a joint is generally only about 1 to 2 centimeters.

4.3.5 Tools

The three tools the robot has also need to be able to be (de)activated. Therefore, there are some additional, smaller motors used in the toolhead. The smoothing tool is attached to a small servo, which can move the tool from and to the wall. The servo has 20 kg · cm stall torque, to resist unevenness in the joint when smoothing. Another servo is used to move the brush from and to the wall and a small DC micromotor then rotates the brush. For the last tool, the mortar extruder, a BLDC motor, about the size or slightly smaller than the movement motors, will be used. This motor does not require closed-loop feedback, but if the accuracy of the ultrasonic reservoir level sensors turns out to be low, this feedback can be added with an encoder to assist as a flow sensor.

4.3.6 User Interaction

`MAS_USC_DIG_DES_01`

The robot is powered using a power button. For statistics and control, there is an OLED display. The user can then navigate a set of menus and displayed messages using a pushable rotary knob. The dial is used to cycle through the options presented on the digital display. To select an option, the user presses the rotary knob. The digital display is present on the robot and a mirrored version will be displayed on the tablet or phone of the operator (with added controls for equivalent navigability).

4.3.7 Safety

`MAS_GLB_ALL_SAF_01, MAS_GLB_ALL_SAF_02`

Safety is one of the top priorities because there will be people working with the robot who do not necessarily have a lot of experience with the technology used in the robot. From an electrical engineering viewpoint, this means the power supply should be safe and shielded away from the user, the motors should be able to detect when something is trapped between two contact points and the operator should be easily able to safely turn the robot off in case of emergency. Also, in case anything in the circuit fails, different subsystems should be able to be isolated from each other with fuses to minimize damage.

There will be pressure sensors on the horizontal bars which should cut off the power, and the current flowing into the motors will be monitored. When the current suddenly spikes, the motors experience excessive resistance and should be powered off. If both safety systems fail, the user can manually cut off power by pressing one of the emergency buttons. In total, there will be three placed, one central on the toolhead, but also one near the wheels at each side. This ensures that even when the toolhead has fully moved to one of the sides, there is still a button within reach. This solves MAS_GLB_ALL_SAF_01.

Finally, we have requirement MAS_GLB_ALL_SAF_02. When the power is cut, the motors also lose their ability to resist gravity. BLDC motors do offer some resistance when not powered, and since they are connected to a 20:1 gearbox this resistance is further increased, but they will

still slide down. The exact speed with which this happens needs to be tested at a later stage and when this is found to be too much, a mechanical brake will be added which is normally activated and only deactivates when it is powered. In case of a sudden power loss, the brake will activate again and keep the robot in place. Additionally, an alarm powered by a small backup battery will sound along with flashing lights to alert the operator of danger.

5. Conclusion

The purpose of this report was to showcase the final detailed design that we intend to make and justify why it is the optimal design.

To help us decide on a final design, we looked into the process of brick jointing in the process of building our test wall. We also considered the environment the jointing robot would have to work, the weather condition it would have to withstand and the needs of our client it would need to adhere to.

- **Lack of a water nozzle**: Usually when a wall is jointed, the mortar is applied to the joint, the mortar is smoothed, brushed and then sprayed with water. Our current design does not include a water nozzle and so that task is still delegated to the operator. Future iterations could add a water nozzle to the set of tool heads
- **Rotation of smoothing tools**: In our current design, smoothing tools are only able to rotate along the Z-axis. This means that our design is limited to executing raked and flush joints. Future iterations of our design could rectify this by enabling smoothing tool rotation along the X axis, which would allow for the execution of struck and weathered joints.
- **Sophistication of maintenance alerts**: Currently, our scheduled maintenance alerts simply pop up after a specified period of time. Future iterations could have the maintenance messages appear as a task that the operator has to mark as 'done' or ignore. If the maintenance task is done then the timer for the task is reset and will appear as normal next time a part has to be inspected or replaced. However, if the task is ignored then the maintenance task will appear daily with increasing urgency level.
- **Sophistication of stored and displayed data**: Currently only daily session statistics are stored and computed. If the operator or construction site manager wants to compute more nuanced statistics where long-term productivity trends can be observed, a database could be designed to store material usage, square meters jointed, duration and starting timestamp of each session.
- **Versatility**: Our current design only executes joints on a standard brick wall with different brick patterns. Future iterations could be expanded to be able to joint concave corners of buildings or joint brick patterns around doors or windows.
- **Training a machine learning model**: Due to a lack of training data our current design uses depth data to differentiate between bricks and joints rather than using a machine learning model. Future iterations could employ a more sophisticated method of detecting bricks.

In summary, our current design meets the needs of our client and provides a decent foundation to expand from and execute more specialized masonry tasks.

Bibliography

[1] Adafruit. *Adafruit GFX Library*. 2024. URL: https://github.com/adafruit/Adafruit-GFX-Library.

[2] ROS Control. *ROS Control Documentation*. 2024. URL: https://control.ros.org/rolling/index.html.

[3] *Intel® RealSense™ Camera 400 Series Product Family Datasheet*. Intel Corporation. 2023. URL: https://www.intelrealsense.com.

[4] IntelRealSense. *realsense-ros*. 2024. URL: https://github.com/IntelRealSense/realsense-ros.

[5] ros-simulation. *gazebo_ros_pkgs*. 2024. URL: https://github.com/ros-simulation/gazebo_ros_pkgs.

Appendix A: Bill of materials

Component	Quantity	Price	Estimated Price	Total Price
Raspberry Pi 5	1	106.85		106.85
BLDC Motor	4	81.99		327.96
VESC	4		90	360
AS5047P encoders	4	6.99		27.96
Micromotor	1	6.99		6.99
Smoothing servo	3	13.50		40.50
Brush actuator servo	1			8
Intel RealSense D405	1	329.36		329.36
OLED Display	1	22.50		22.50
Power Supply	1		400	400
CAN-Hat	1	29.95		29.95
Ultrasound sensor	1	5.00		5.00
Pinch point sensor	4		10	40
IMU	1	28.86		28.86
Rotary Knob	1	5.95		5.95
Lights, alarm speakers and extra peripherals			30	30
Aluminium profile	4	16.31		65.24
EG24-G20-D10 gearbox	3	50.11		150.33
Bearings	19			9
Steel axles	560 mm			12
Nuts, bolts and fasteners	75			20
T-nuts	40			12
Smoothing tool	3			23.77
Brush	1	6.49		6.49
3D printed TPU parts	1 kg	30		30
3D printed PLA parts	5 kg	17		85
Sheet metal	2 m^2			40
Mortar gun	1	25		25
Total				2248.71

Appendix B: Fabrication

The fabrication of the robot will happen over the course of several weeks. Many components are already fabricated or will be made in tandem with other parts, but an approximate order of manufacturing is the following:

1. Gantry - horizontal movement
2. Gantry - Vertical movement
3. Tool heads
4. Electronics - computing and power modules
5. Electronics - sensors
6. Electronics - wiring
7. Software - pathing, control and detection algorithms
8. Software - integration with display
9. Protective casing

The most critical system to have up and running is the mobility of the gantry. If things don't move, there won't be any jointing. Next are the tool heads. The arrangement of tools and the camera can likely still be optimized, so visualizing and testing what works and what doesn't as soon as possible is important. Having functional tools and movement means we can begin with realistic tests evaluating joints that Mason makes. After that, the Raspberry Pi, motor controllers, and power supplies can be installed followed by the rest of the connecting electronics. The code base for interpreting sensor data and determining how to act will take a relatively long time to fully develop. The sooner that process starts, the better the result will be. Lastly, the protective casing which encloses dangerous and vulnerable parts will be made and installed. The casing protects the user from the components and the components from the environment as well as giving a professional finished look, but it doesn't provide a core function. Additionally, having the space to build, install, and test the previously mentioned components without a large casing getting in the way would also be favorable.

The gantry is made from standard purchased elements and custom-made, 3D-printed components when the required geometry is complex such as for the wheels or motor attachment points. AutoArtisan has in-house capabilities for multi-material printing, including flexible filaments, engineering filaments, and support filaments that dissolve in water. Therefore the challenging geometries of those components can still be made with high quality and low cost.

The tool heads are also largely a combination of purchased parts and 3D-printed pieces. The mortar gun that extrudes the mortar has for example been purchased, but the brush motor casing is 3D printed to accommodate the unique Z-direction actuation capabilities.

Electronic parts are naturally not manufactured by us, but the casings and holders for those parts will be 3D printed and some might be made from bent sheet metal where applicable and necessary as might be the case for the heavy power supply.

The protective casing has an underlayer of formed and laser-cut sheet metal for strength and for providing a base upon which the mostly 3D-printed cushioning outer layer can be mounted. The shell can not be made from massive prints since the manufacturing of 3D-printed parts is limited by bed size.

Unless absolutely necessary for strength or safety reasons, parts manufactured by AutoArtisan will not be milled or laithed. This position is held because machined parts take long, are

expensive, and are heavy. Furthermore, plastic parts are electronically insulating and resistant to corrosion.