ENGG3000 Department of Engineering, Macquarie University.

Design Document

Massive Marvellous Multiple Marble Machine

Kamakura Team - Communication Division (T1_C6)

Date of Issue: 1st August, 2023

I. Table Of Contents

III.Glossary of Definitions and Abbreviated Terms

Term	Definition
Con	Prefix for constraint
ReqF	Prefix for functional requirements
ReqN	Prefix for non-functional requirement
ReqL	Prefix for interfacing requirement
AS	Australian Standards
ReqM	Prefix for measurable performance requirement
TPM	Technical Performance Measure
MQ	Macquarie University

IV. Revision Log

Date	Revision	Author	Description
1/8/23	1.0	MQ Engineers	Initial Draft Design and Layout issued and shared via online platform
3/8/23	1.1	MQ Engineers	Requirements and problem definition updated after receiving tutor input
5/8/23	1.2	MQ Engineers	Requirements and TPMs updated
8/8/23	1.3	MQ Engineers	Deliverables updated and finalised
14/8/23	2.0	MQ Engineers	Draft scoping document submitted for buddy review via iLearn
15/8/23	2.1	MQ Engineers	Feedback received from buddy marking changes made to scoping document
18/8/23	3.0	MQ Engineers	Scoping Document Completed
25/8/23	3.1	MQ Engineers	Scoping Document Amended based on Feedback from Client
23/8/23	4.0	MQ Engineers	Structure and layout for Design Document created
26/8/23	4.1	MQ Engineers	Conceptual Design draft completed
30/8/23	4.2	MQ Engineers	Design Alternatives draft completed
1/9/23	4.3	MQ Engineers	Final Design draft completed
3/9/23	5.0	MQ Engineers	Design Traceability Integrated
5/9/23	5.1	MQ Engineers	Conceptual Design Finalised
7/9/23	5.2	MQ Engineers	Design Alternatives and Detail Design Updated and Finalised
10/9/23	5.3	MQ Engineers	Final Proposal Review Completed

V. Contributors

Members	Student ID	Email	Workload Acknowledgement
Aaron Chakerian (Team Lead)	43255752	aaron.chakerian@students.mq.edu.au	Problem Definition, Constraints, Assumptions, Functional Requirements, Non-Functional Requirements, Interfacing Requirements, TPMs
Luca Gonzalez	45959897	luca.gonzalez@studnets.mq.edu.au	Project Schedule, Problem Definition, User Stories, Functional Requirements, TPMs, Appendix A
Hugo Cawsey	47293241	hugo.cawsey@students.mq.edu.au	Problem Definition, Justifications, Inclusions / Exclusions, Assumptions, Constraints, Subsystems
Andrei Falls	47205482	andrei.falls@students.mq.edu.au	Problem Definition, Constraints, Inclusion / Exclusions, Preliminary Concept, DSM, Project Schedule
Filip Zurawski	47268077	filip.zurawski@students.mq.edu.au	Consultation Findings, User Stories, Constraints, Deliverables, Systems and Subsystems, Requirements, TPMs

1 Introduction

1.1 Scoping Document Overview

The scoping document has been created to show the project planning process that outlines deadlines, deliverables and budgets for the "Massive Marvellous Multiple Marble Machine" project. Before any work commences the project scope needs to be agreed upon by the client (MQ Engineering Faculty Staff) and the Kamakura Team.

This document intends to provide the necessary details to ensure clear expectations are set between client and project team. To ensure efficiency across the project development, expectations are set at this early juncture to ensure satisfaction for both parties upon project completion.

The scope of the project is clearly defined within this document to ensure Kamakura project team and the client come to agreement prior to development ensuring deadlines and budget stay on track. This allows clear expectations to be set by the client, and outlines what the project team will deliver.

As prospective engineers aspects of this project have been provided considerably clearer compared to in industry. This is likely due to our developing experience as engineers and the timeframes in which the project and documentation needs to be completed.

While the entire sculpture has been delegated to teams across the ENGG2000/3000 disciplines, the Kamakura team has been tasked with designing a single cube that meets the clients needs. The Comms team is a team within Kamakura that is responsible for delivering the logic systems within and across boxes. Monitoring the motion of the ball, motors, timings and lighting within the system to fulfil the clients requirements. An additional objective of across box communication has also been requested and may be considered during later design stages.

1.2 Consultation Findings

Preliminary research and discussions procured vital insights to increase efficiency and mitigate risk. Gathering data from the client and teaching staff, enhanced the team's ability to preemptively identify scenarios that were not initially considered. This gave our team critical information to plan for the project, increasing the likelihood of the timeframes for deliverables.

Moreover, informal interaction with other teams and the teaching staff equipped us with intricate details crucial for the successful completion of this project. The key areas of focus stem from the issues encountered by the cohort of 2022 and their suboptimal project completion rate. Furthermore, aspects related to documentation, accountability, team synergy and dynamics played a paramount role in shaping our scoping document.

The scoping document has taken these findings into account, adjusting appropriate elements to ensure system functionality and reliability. This makes the consultation phase crucial, as utilising this information allows the team to better align with the client’s vision and ensure the successful realisation of the project.

Consultation Finding ID	Consultation Summary
CoF1	The Marble machine must be designed with an always-ready acceptance mechanism, ensuring that it is receptive to marbles according to the requirements without disruptions.
CoF2	The machine should not return or reject any marbles, regardless of the influx rate or potential internal or external discrepancies. This implies that a heavily efficient and fault tolerant design is required to ensure consistent processing and accommodation of marbles.
CoF3	Our client stocks some of the components that could be used to fulfil the functional requirements of the machine. Notably, Arduino Uno microcontrollers and NeoPixel-type LED strips. They’ve graciously extended an offer to provide these to the project team at a subsidised rate presenting an opportunity for budget optimization.
CoF4	The available power supply for the system will be limited to 5V1A and 12V1A with a common ground terminated with a Mini-Fit Jr female connector. This introduces a challenge in terms of how much power can be drawn at any given time,

	especially if multiple components require power simultaneously. Given these constraints, power management may play a pivotal role not just at the hardware level, but also at the software level. Software-driven power management could be indispensable to optimise the consumption and distribution of power among various project components, ensuring smooth and uninterrupted operation. Special consideration should be given to ensuring that the system’s operations, lighting functionalities, and other active components are powered and prioritised to ensure uninterrupted operation.
CoF5	The client requested to consider options for allowing the cube to interact and communicate with the greater sculpture installation. This interaction should further extend to receiving commands from external entities or systems. Such commands may direct the cube to modify the lighting profile or even transition it to a different operational mode.
CoF6	The software embedded within the cube should be receptive, allowing for instantaneous adjustments based on received commands. The capability to respond in real-time to dynamic needs of the overall installation could prove vital in ensuring reliability and managing energy use of the greater system.
CoF7	The software architecture shall allow for easy addition of communication capabilities should the client request that feature. Such capabilities would consist of a robust protocol, ensuring that command transmissions are accurate and instantaneous. Proper error handling and feedback mechanisms will be crucial to confirm successful reception and execution of commands.

Table1.2a - Consultation Findings

1.3 Problem Definition

The client, Macquarie University Faculty of Engineering representatives, is seeking a creative and eye-catching engineering marvel at the entrance to their building. This should be aesthetically pleasing and display to academics, students and visitors, while also exhibiting impactful learning and student journey.

The “Massive Marvellous Multiple Marble Machine” is to be a kinetic sculpture made up of cubes designed and developed by teams of students. Teams have been designated a standard interface, each cube to be incorporated into a single, collective, kinetic sculpture. This interface allows marbles to be transferred across the different teams' cube creations and throughout the entire sculpture.

After individual teams design and develop their cubes, the client should then be able to arrange individual cubes into the sculpture as they please. The ability to change the position of individual cubes within the sculpture, will allow the client to change the overall effect of the sculpture.

The Kamakura team, sectioned into Structure, Motions and Comms groups, has been tasked with designing one of these cubes to be incorporated into the larger sculpture. The Comms team is a team within Kamakura that is responsible for delivering the logic systems within and across boxes. Designing and developing systems that monitor the motion of the ball, motors, timings and lighting within the box. An additional objective of across box communication has also been requested and may be considered during later design stages.

Additionally, if needed, the Comms group may need to develop a system in which the Kamakura Box is able to communicate with other Team Boxes in the sculpture, for a more organised structure. This will require Comms to carry out efficient integration during a specified period, allowing for testing and improvements to be made.

1.3.1 Subsystems

Developing a comprehensive understanding of the issue is essential to meeting deliverables in the desired timeframe. A high level overview of the subsystems is used to identify interactions between the required components of the system. The subsystems can be segmented into distinct functionalities allowing parallel development while efficiently allocating resources and managing project challenges.

Below, Figures 1.3.1 and 1.3.2 present an elegant solution providing flexibility surrounding hardware decisions while allowing for subsystems parallel development. The modular nature of the system allows upgrades and repairs without significant modifications. (Refer to Appendix A3.1, Appendix A3.2)

1.3.2 Justifications

The Faculty of Engineering at Macquarie University is interested in acquiring additional modules for a kinetic sculpture installation destined to be installed by the entrance of their new building by the year 2025 at the latest.

The project was started in the year 2022 and remains incomplete due to the previous cohort running into issues with reliability, durability and completion rate. Particularly, the existing installation is missing the necessary return mechanism, which would move marbles from the bottom of the sculpture back to the top, and distribute them between columns.

Continuing work on the kinetic sculpture installation will not only complete the legacy of the 2022 cohort, ensuring that their hard work does not go unrecognised, but also uphold the Faculty’s reputation as an institution which values sustainability, adaptability and problem solving. It will prevent the already spent materials from being sent to the landfill, could serve as a physical display of the students’ ability to solve problems and work around constraints.

The sculpture could also communicate to the future students that their hard work will be noticed, inspiring them to overcome their challenges in new, creative ways. It could showcase that any challenge, no matter how difficult, can be overcome with determination and collaborative effort.

1.3.3 User Stories

As a viewer I would like an interesting sculpture to look at.

As a viewer I would like for the statue to be running when I next visit so that I can show my friends.

As a viewer I would like a consistent theme between boxes so that the colours between boxes make sense and do not clash.

As a viewer who is not a current engineering student. I want a technically complex and impressive looking statue so that I may be inspired to pursue engineering.

As a viewer who is a parent who is visiting the engineering department, I would like a showcase of what the abilities are of students who attend this university.

As a viewer who is just passing by I would like to take a moment to stop and enjoy the playful nature of the statue, which adds a unique touch to the building foyer.

As a viewer who is an Alumni of the engineering department I want a statue that I can be proud of so that I can show off the calibre of students who study at Macquarie University.

As a viewer who is a professor from another department. I want a statue that incites a feeling of enlightenment, leading me to reflect on the interdisciplinary nature of innovation and the potential collaborations between my field and some field of engineering.

As a viewer who is an up and coming field reporter for a major news outlet I would like a backdrop that exudes a strong sense of pedigree and aptitude when reporting on one of the many Macquarie University game changing endeavours. So that viewers can easily display the significance of the endeavours.

As a person with special needs, I would like an installation with a light display which does not promote epileptic seizures.

Maintenance Worker

As a maintenance worker I want clear documentation so that I can quickly diagnose and repair the cube when it breaks.

As a maintenance worker I want an easy way to reset the machine so that if an error occurs it can be the software can be reset.

As a maintenance worker I want to be able to easily access the information on the serial port so that I can diagnose errors.

As a maintenance worker I want a reliable system so that I can minimise the time I spend repairing the system.

1.4 Constraints & Assumptions

1.4.1 Constraints (personnel, time and regulatory standards)

Constraint ID	Description
CON 1.0	Skills are limited to the knowledge and experience of project team members.
CON 2.0	Project's timeframe is 13 weeks, from consulting to final integration / deployment
CON 3.0	$100 budget to be shared with other groups in the project
CON 4.0	Compatibility with the client's existing installation
CON 5.0	AS3000 Wiring Rules 2018 [1]
CON 6.0	Coding standards and practices [2]
CON 7.0	Coding Style Guide [2]

Table 1.4.1 Constraints

1.4.2 Assumptions

Assumption ID	Assumption
A01	Deliverables & FInalised Requirements by the client will not change during construction of the project.
A02	Power supply is constant and stable whilst the box is in use.
A03	All programmable components are compatible with an Arduino Uno.
A04	Potential for each box in sculpture to communicate
A05	The environment that the box is stored in will not significantly affect the box operations.
A06	Skills developed through design and construction will be sufficient enough to successfully complete the project.
A07	No counter occurrence plan to deal with indirect failures such as fire, water etc.
A08	Any hardware will perform as expected.

A09	Marbles will enter the box in a deliberate controlled manner
A10	Materials will be delivered in a timely manner
A11	Costs will remain reasonable throughout product development
A12	Workspace provided for construction and testing of box will remain in a condition that is usable and safe
A13	Utilities will remain within operational tolerances to ensure adequate testing and development

Table 1.4.2 Assumptions

2. Scope of Works

2.1 Deliverables

2.1.1 Inclusions and Exclusions (delivered in Scoping Document)

Agreement of action items and services to be provided or omitted in the project works for the client.

2.1.2 Requirements (delivered in Scoping Document)

A document containing all the necessary requirements imposed on the project by the client, considering all the necessary items to signify the solution successful.

2.1.3 Scoping Document (delivered by 20/08/2023)

This document establishes the contractual basis of works and defines perimeters of the project across multiple dimensions. It contains items such as the problem statement, constraints, deliverables, detailed requirements and justifications for the project. It also includes a testing schedule and timeline of works. The document excludes any proprietary information.

2.1.4 Design Review Document (delivered by 29/08/2023)

This document discusses the selected design at a high level. It is created to ensure that the selected solution does not possess any inherent deficiencies that could result in failure to meet the requirements.

2.1.5 Design Document (delivered by 05/09/2023)

A document containing the full requirements, constraints, testing criteria, diagrams, interfaces and communication protocols for the final proposed solution. Alternate versions may be presented together with their own justifications. Overall, the design document will communicate the final design of the product and allow the client to verify whether or not the proposed solution meets their requirements.

2.1.6 Testing Document (delivered by 29/10/2023)

This document will present a comprehensive testing methodology with success/failure criteria, used to measure actual performances of the designed solution, then evaluated against the Technical Performance Measurement (TPM) values that were initially defined as requirements. It discusses the initial analysis, calculations and comparisons of realistic design alternatives. A review of the chosen design against requirements will be included, together with any possible future improvements.

2.1.7 Statement of Work (delivered by 29/10/2023)

This document will provide a description of the project requirements. It is used to ensure that the stakeholders are informed about deadlines, scope of work, and project expectations which will allow everyone to stay aligned towards project goals and objectives.

2.1.8 Project Presentation (delivered by 07/11/2023)

The presentation will discuss the quality, functionality, and any issues of the communications subsystem. This is to determine if the functionality of the subsystem is sufficient, along with any deficiencies in quality, or integrability. The overall functionality of the whole system will be discussed in terms of meeting project goals and requirements.

2.1.9 The Cube (delivered by 22/10/2023)

The cube itself will be designed and delivered to the client, meeting all the requirements set out in this document.

2.2 Inclusions and Exclusions

2.2.1 Inclusions

Inclusion ID	Inclusion Name	Explanation
INC 1.01	Software Design & Development	● Develop the software that monitors the marble's movement inside the cube. ● Ensure that the software can track or reasonably estimate the marble's position, and provides debugging tools for this purpose.
INC 1.02	Visual Feedback System	● Design and integrate software controls for the LEDs or other approved lighting systems, providing a visual indication corresponding to the function being performed by the marble.
INC 1.03	System Troubleshooting & Diagnostics	● Develop diagnostic tools and procedures to identify issues within the software or its interfacing with the hardware. ● Create a debugging and troubleshooting system to detect anomalies, view state of the machine, or allow control and tuning of functional modules of the machine. ● Create alert mechanisms for any software failures or discrepancies.
INC 1.04	Software Testing	● Rigorously test the software to ensure it meets all requirements and can effectively interface with the hardware components. ● Design test cases to simulate potential problems and validate the troubleshooting mechanisms.
INC 1.05	Integration	● Collaborate with the motions and structure teams to ensure the software integrates seamlessly with the cube's physical mechanisms.
INC 1.06	Documentation	● Produce comprehensive software specifications, design decisions, troubleshooting guides, and other relevant information.

INC 1.07	Compliance	● Ensure the software adheres strictly to the provided requirements, especially those concerning visual feedback and interfacing

Table 2.2a - Inclusions

2.2.2 Exclusions

Exclusion ID	Exclusion Name	Explanation	Group Responsible
EXC 1.01	Physical Construction and Design	● Dimension: Physical dimensions relating to comms components not included. ● Materials: Material choices for comms components are outside our scope ● Arrangement: Physical arrangement of comms components are not the responsibility of our group	Structures
EXC 1.02	Motion Control	● The mechanical movement/motion of the marble.	Motions
EXC 1.03	Electrical Wiring	● Wire routing and circuit connections of components.	Structures
EXC 1.04	Hardware Troubleshooting	● Physical diagnostics and hardware failure identifications of components.	Structures
EXC 1.05	BOM for hardware	● Bill of Materials breakdown for hardware / electronic components	Structures, Motions, Communications
EXC 1.06	Mounting Rack	● The rack in which the final design of the box will be fit to	Structures

Table 2.2b - Exclusions

2.3 Project Schedule

2.3.1 Key User Stories:

1) As a viewer I would like an interesting sculpture to look at.
2) As a viewer I would like a consistent theme between boxes so that the colours between boxes make sense and do not clash.

Key User Story 1	**Key User Story 2**
Design and implement Lighting control	Design and implement Lighting control
Design and implement Power Management	Design and implement Power Management
Design and implement Track and movement control	Design and implement Track and movement control
	Design and implement inter-box communications

Table 2.3a - Breakdown of key user stories

2.3.2 List of Tasks

- Design and implement lighting control
- Design and implement power management
- Design and implement track and movement control
- Design and implement inter-box communications

To be completed by 5/9/2023

Tasks	**Wk5**	**Wk6**	**Wk7**	**BR1**	**BR2**	**Wk8**	**Wk9**	**Wk10**	**Wk11**	**Wk12**
Design and implement lighting control	▨	▨	▨	▨	▨	▨				
Design and implement power management			▨	▨	▨	▨	▨			
Design and implement track and movement control			▨	▨	▨	▨	▨			
Design and implement inter-box communications	▨	▨	▨	▨	▨	▨				
Integration							▨	▨	▨	▨

Figure 2.3b - Project Schedule

3 Requirements

All the requirements that are laid out below are specific to the Communications responsibilities for the project. The requirements are split into different sections Functional, Performance and Interfacing allowing streamlined collaboration across the different teams (Structure, Motion and Communication). These requirements will be used throughout the documentation ensuring sufficient detail and traceability is documented from initial, to final design phases.

Separating the different groups the Functional Requirements will outline the mandatory behaviours for the system to meet the client's requests. The Non-Functional Requirements are critical for the overall success of this project. Considerations during project design and development are crucial to meet the clients needs. The interfacing requirements are those that need to be resolved between Motions, Structures and Communications in the Kamakura team.

3.1 Functional Requirements

Table 3.1.1 Defining the projects Functional Requirements endorsed by the Client

REQUIREMENT ID	What is the Requirement?
ReqF01	The cube must have the external dimensions of 250x250x250 mm
ReqF02	The cube must mount to the sculpture using only the four mounting positions specified in figure 3.
ReqF03	The cube must have a visual indication using lights displaying the function being performed. Fit Requirement measured by (TPM_F03a):
ReqF04	The cube must have a single 'service port' to access data and power (specified in Appendix A Figure 3)
ReqF05	Cube location on the backing board shall be in the middle of a square 265mm horizontally by 265mm vertically. This requirement implies that there is a 15mm distance between cubes.
ReqF06	Cube must be designed to accept marbles that are a solid steel sphere with a diameter of 16mm.
ReqF07	Cube must accept marbles that are spaced no closer than 1

	second apart through the bottom opening port. Fit Criteria measured by (TPM_F07)
ReqF08	Cube must be manufactured from 3mm MDF for the back and side pieces, with the additional requirement that the front surface must be transparent.
ReqF09	The front surface must not be structural, and must be easily removable for maintenance purposes for the cube.
ReqF10	The only forces provided to a cube are gravity and electrical power. The cube must not utilise any liquids or compressed gases in its operation.
ReqF11	The cube must accept a marble through the bottom surface at the location specified in figure 1. The marble being accepted is assumed to be moving upwards at a trajectory that will peak at 50mm into the cube.
ReqF12	Each cube must expel all marbles out from the TOP surface such that its trajectory will peak at 50mm, at designated location (specified Appendix A Figure 2)
ReqF13	Cubes shall not deliberately expel marbles at a speed significantly higher than that required. Approximately 2.5-2.6 m/s for a standard 16mm ball bearing launched from a point approximately 10 cm below the exit port.
ReqF14	The marble makes a full rotation around an imaginary line that MUST NOT intersect the marble, drawn through the cube. In other words the marble's direction must change in at least two dimensions by the size of the marble.
ReqF15	The marble is held stationary in all physical dimensions for a minimum of 0.5 seconds in time.
ReqF16	The marble moves upward against gravity for a minimum of 100mm in one action.

3.2 Non-Functional Requirements

Table 3.2.1 Defining the projects Non-Functional Requirements

REQUIREMENT ID	What is the Requirement?
ReqN01.1	Reliability: Ensuring the design is reliable and marble can traverse the box repeatedly, this project is designed for repeated and extended use. Fit Criteria measured by (TPM_P01.1)
ReqN01.2	Reliability: The system should be able to function in the event of hardware and/or software failures.
ReqN01.3	Reliability: The system should have a 99.9% uptime. Over the course of 30 days there should be no more than 43.2 minutes of downtime
ReqN02.1	Manufacturability: Equipment and materials should be easily accessible to ensure the product can be designed, manufactured and tested before the project deadline
ReqN02.2	Manufacturability: Ensure available materials. To allow design, manufacturing and testing before the project deadline
ReqN03.1	Maintainability: Clear and concise code with necessary comments throughout. Modular system design to ensure ease of repairs and upgrades
ReqN03.2	Maintainability: The code will follow the style guide and best coding practices
ReqN03.3	Maintainability: The system should have an automated deployment process
ReqN04.1	Assembly: Allowing appropriate functionality while removing unnecessary complexity through best coding practices, after unexpected events i.e. power failure. The system can reset and resume operation without a programmer present. Fit Criteria measured by (TPM_P03.2b)
ReqN05.1	Modularity: The box should be able to be moved to any fixture

REQUIREMENT ID	What is the Requirement?
	point around the sculpture and begin/resume operation once connected to power.
ReqN06.1	Robustness: Ensuring the machine has a life expectancy greater than one year.
ReqN06.2	Robustness: The system should handle errors such that they are logged with relevant information.
ReqN06.3	Robustness: The system should be designed with fault tolerances to allow operation during hardware and software failure.
ReqN07.1	Sustainability: Considering material choices and waste management is vital for long-term environmental considerations, but it may have a less immediate impact on the project's success within a 13-week timeframe.
ReqN08	Human-centred designs: While important for user experience, clear briefing statements, and team recognition, this aspect can be prioritised lower for specific engineering projects, mainly when time is limited. However, consider this your platform to showcase your work to future employers.
ReqN09	Bill of materials: must total to less than AUD$100.00.

3.4 Interfacing Requirements

Interfacing requirements define how different components, systems, or disciplines will communicate, collaborate, and interact to ensure seamless integration and functionality.

REQUIREMENT ID	What is the Requirement	Sign off
ReqI01	Power and Electronics: Safe and reliable power distribution across all electrical components in the project. Ensuring peak power consumption stays within acceptable ranges throughout the system. Including 12v power and 5v USB (data and power) supply.	Motions Group Team Lead- Adrian Zhang
ReqI02	Electrical-Mechanical: Precise dimensions, tolerances and mounting mechanisms for attaching electrical components. Proper alignment and secure attachment of hardware. Items such as wiring(power and data), sensors, servo motors and LEDs.	Structures Group Team Lead- Hrithik Barua
ReqI03	Software-Hardware Interfacing: Synchronisation mechanisms between software and hardware such as a microcontroller, actuators, servo motors and sensors.	Motions Group Team Lead- Adrian Zhang
ReqI04	Communication standards: This includes documentation, terminology, language and cross disciplinary checks.	All Teams Signed off Adrian Zhang Hrithik Barua Aaron Chakerian

Table 3.4a - Requirements

4 Technical Performance Measures and Testing Schedule

4.1 TPM Summary

TPM Code	TPM Name
TPM_F03a	Lighting System Presence
TPM_F03b	Lighting System Responsiveness
TPM_F03c	Marble Detection Reliability
TPM_F07	Marble Spacing
TPM_F10	Energy Sources
TPM_F11b	Marble Acceptance Capabilities
TPM_F12b	Marble Expulsion Energy
TPM_N01.1	Continuous Duty Time Without Failure
TPM_N01.2	System Fault Tolerance
TPM_N03.1	Code Maintainability
TPM_N03.2a	Code Readability
TPM_N03_2b	Code Documentation
TPM_N04.1	Boot Time
TPM_I01	Power Management

Table 4.1a - Technical Performance Measures Overview

4.2 Technical Performance Measures

TPM ID	TPM_F03a
TPM Name	Lighting System Presence
TPM Purpose	Ensure that the structure is equipped with a lighting system
Source Requirement	ReqF03
Risk Level	Low
What should be measured?	Presence of lighting system
How should it be measured?	Presence of light sources, wiring and sensors should be verified
Measure of Success	The cube contains the specified equipment
Measure of Failure	Cube is missing any or all of the components required for the lighting system to output light
Possible Causes of Failure	Inadequate care taken during design and/or assembly

TPM ID	TPM_F03b
TPM Name	Lighting System Responsiveness
TPM Purpose	Ensure that the system can detect and react to a passing marble within a reasonable time, so that the animation is synchronised with its movement
Source Requirement	ReqF03
Risk Level	Medium
What should be measured?	The delay between detection of the marble and the first LED on the path of the marble reaching maximum designed brightness; The number of failed activations
How should it be measured?	1. Enable debugging mode in the software configuration

	2. Collect logs sent by the cube over 5 minutes of nominal operation 3. Find the first detection event and the nearest corresponding "maximum brightness reached" event. Subtract the time of detection from time of reaching maximum brightness. 4. Log the difference and classify whether it passed or not 5. Repeat for all events 6. Divide the number of failed tests by the number of passed tests
Measure of Success	99.9% of animation sequences started within 500 ms of the marble being detected
Measure of Failure	In > 0.1% of cases, the animation sequence failed to start, or started with a delay longer than 500 ms
Possible Causes of Failure	- Improper care taken during system configuration or integration - Errors in the debugging/logging algorithm - Error in the lighting animation script - Error in another subsystem of software

TPM ID	TPM_F03c
TPM Name	Marble Detection Reliability
TPM Purpose	Ensure that the marble detection system is reliable. That is, it not only detects close to all marbles passing, but is not susceptible to false activations when no marble is present.
Source Requirement	ReqF03, Affects TPM_F03b
Risk Level	Low
What should be measured?	The number of sensor triggers; Number of marbles entering the cube
How should it be	1. Enable debugging mode in the software

measured?	configuration 2. Collect logs sent by the cube over 5 minutes of nominal operation 3. Count the number of marbles entering the cube over the same 5 minute period using a clicker, tally or another preferred method 4. Count the number of marble detection events in the logs for each sensor 5. Divide the number of detection events by the number of marbles entering the cube 6. Repeat step 5 for each sensor
Measure of Success	Every sensor detects more than 99.99% of marbles passing and no more than 100.001%
Measure of Failure	Any of the sensors installed detects less than 99.99% or more than 100.001% of marbles passing through its detection zone
Possible Causes of Failure	- Improper system configuration values related to sensor sensitivity - Improper electrical connection of the sensor - Misalignment between the sensor and detection zone - Sensors mounted directly in line leading to crosstalk

TPM ID	TPM_F07
TPM Name	Marble Spacing
TPM Purpose	Ensure that the system is capable of handling marbles at the required rate while maintaining required performance
Source Requirement	ReqF07, Affects all TPM_F** and all TPM_N**
Risk Level	Medium
What should be measured?	Rate of marble acceptance and expulsion; Peak expulsion height; Marble detection reliability; Lighting

	System Responsiveness
How should it be measured?	1. Enable debugging mode in the software configuration 2. Assign one person to counting marbles entering the cube 3. Install a marker for minimum and maximum expulsion height in the cube above at 63 and 67 mm above the exit hole of the tested cube 4. Assign one person to counting marbles expelled by the cube peaking between markers in the cube above 5. Start all measurements 6. After 5 minutes stop all measurements 7. Inspect the cube for the presence of any marbles that have gone off track and record the figure as Loss. Divide by number of marbles entering the cube to get L% 8. Subtract the number of marbles expelled within tolerance from the number marbles entering the cube. Record this figure as Abnormal Expulsion (AE). Divide Abnormal Expulsion by number of marbles entering the cube to get AE% 9. Follow testing procedures for TPM_F03b and TPM_F03c and ensure the tests pass
Measure of Success	1. Tests for TPM_F03b pass 2. Tests for TPM_F03c pass 3. L% <= 0.003%. I. e. No marbles may exit the designed track in a 5 minute period. 4. AE% <= 0.01% I.e. no more than 3 marbles are expelled with an out of specification speed.
Measure of Failure	- Major failure resulting in failure to complete any of the tests - Tests for TPM_F03b, TPM_F03c, TPM_F11b, TPM_F12a or TPM_F12b fail
Possible Causes of Failure	- Improper configuration - Improper integration

	- Improper assembly - Design not fit for task

TPM ID	TPM_F10
TPM Name	Energy Sources
TPM Purpose	Ensure that the system only uses energy sources approved by the customer
Source Requirement	ReqF10
Risk Level	Low
What should be measured?	Presence of systems requiring liquids, gases, plasma or any other source of energy that is not electricity of gravity
How should it be measured?	Visual inspection of elements
Measure of Success	The only sources of energy are electricity and gravity
Measure of Failure	The system requires gases or liquids to function
Possible Causes of Failure	Improper care taken during design stage

TPM ID	TPM_F11b
TPM Name	Marble Acceptance Capabilities
TPM Purpose	Ensure that the marble input is fit for task of accepting marbles
Source Requirement	ReqF11
Risk Level	Low
What should be measured?	Number of marbles entering the cube, Number of marbles correctly captured onto the internal track and into the first functional module
How should it be measured?	1. Count the number of marbles entering the cube (E), and marbles correctly entering the first functional module after capture (C) for 5 minutes

	2. Subtract C from E and divide the result by E. This is your Failure rate (F%)
Measure of Success	F% < 0.003. That is, all marbles entering the cube over a 5 minute period are captured and passed onto the first functional module correctly
Measure of Failure	F% > 0.003. Not all marbles are captured correctly over a 5 minute period
Possible Causes of Failure	- Improper design of the capture mechanism - Improper design of the track - Failure of the first functional module - Improper configuration - Improper integration - Improper assembly

TPM ID	TPM_F12b
TPM Name	Marble Expulsion Energy
TPM Purpose	This is to ensure that the marbles are expelled into the cube above so that they can be captured correctly
Source Requirement	ReqF12
Risk Level	Medium
What should be measured?	Total number of marbles expelled into the cube above; Number of marbles that peak below the specified height
How should it be measured?	1. In the cube above, mark lines at 63 and 73 mm above the top wall of the cube being tested 2. Count the total number of marbles expelled by the cube (N), and number of marbles that peak below or above the marked lines (X) within a 5 minute period 3. Subtract X from N and divide the result by N to get the percentage of marbles expelled with energy that significantly differs from specification (X%)
Measure of Success	X% < 0.003. That is, over a 5 minute period at the

	maximum rate, all marbles are expelled to a height that is within specification
Measure of Failure	X% >= 0.003. That is, any marble expelled during the 5 minute testing period does not fit the criteria
Possible Causes of Failure	- Improper configuration - Improper integration - Improper track/outlet geometry - Improper system design - Chosen components not fit for purpose

TPM ID	TPM_N01.1
TPM Name	Continuous Duty Time Without Failure
TPM Purpose	Ensure that the system meets the reliability criteria set by the customer
Source Requirement	ReqN01.1, ReqN06.1
Risk Level	Medium
What should be measured?	Unplanned downtime over the testing period
How should it be measured?	1. Enable debugging mode in the software and connect a storage device (optional) 2. Leave the system operating for 7 days 3. If not notified of failures over the period, return to the machine 4. Verify that all functional modules are operating and there are no visible failures 5. Inspect the cube for any marbles that may have fallen off the track
Measure of Success	The machine performs nominally and did not suffer any failures requiring maintenance in the time period. No more than 1 marble had fallen off the track.
Measure of Failure	Occurrence of any mechanical, electromechanical or software fault prevented the machine from

	functioning, or more than 1 functional module suffered loss of configuration during normal operation in the 7 day period
Possible Causes of Failure	- Manufacturing defects - Design defects - Improper configuration - Improper integration - Software bugs - Hardware bugs - Components not fit for purpose

TPM ID	TPM_N01.2
TPM Name	System Fault Tolerance
TPM Purpose	Ensure that minor failures do not propagate through the system or affect core functions of the system
Source Requirement	ReqN01.2, ReqN06.1
Risk Level	Moderate
What should be measured?	Number of subsystems affected by failures in non-critical subsystems
How should it be measured?	1. Introduce a non-conductive, light marble into the system. Verify that it passes correctly, log the result and repeat 10 times 2. Disconnect the sensors from the controller. Ensure that the software runs correctly at a reduced capability level and that the cube is still able to accept and pass marbles. Log the result and reconnect the sensors. 3. Disconnect the LEDs from the controller. Ensure that no other capabilities of the system are affected and that the cube can still pass marbles. Log the result and reconnect the lights. 4. If installed, disconnect the communication bus leading out of the cube. Verify that the lighting remains the assigned colour and no other capabilities

	of the system are affected. Log the result and reconnect the communications bus.
Measure of Success	Disconnection of non-critical modules does not prevent marbles from passing
Measure of Failure	Disconnection of any of the modules results in loss of configuration, or causes failures in other modules of the system
Possible Causes of Failure	- Improper configuration - Improper integration - Poor choice of components - Software bugs - Hardware bugs

TPM ID	TPM_N03.1
TPM Name	Code Maintainability
TPM Purpose	Ensure that the code supplied as a part of this project will be easily maintainable, allowing the customer to extend and reuse the software as needed
Source Requirement	ReqN03.1
Risk Level	Low
What should be measured?	Levels of indentation, Number of functional modules per file
How should it be measured?	1. For each file, inspect the number of levels of indentation 2. If it is more than 5, a part of that block should be split into a separate function 3. Assert separation of concerns between files. If some functions are better suited as a part of another functional module, they should be moved to that file
Measure of Success	Maximum of 5 levels of indentations (Tab = 4 spaces). One functional module per file.

Measure of Failure	More than 5 levels of indentation in any file; Functions for one functional module located in a file for another
Possible Causes of Failure	User error

TPM ID	TPM_N03.2a
TPM Name	Code Readability
TPM Purpose	Ensure that the code can easily be read and understood by a person from outside the development team
Source Requirement	ReqN03.2a
Risk Level	Low
What should be measured?	Number of function, variable or module names that do not communicate their purpose, Presence of comments containing documentation above functions
How should it be measured?	1. Count the total number of functions, variables and modules 2. Judge and record whether the names communicate their function
Measure of Success	99.5% or more of the code base is considered readable
Measure of Failure	Less than 99.5% of the codebase considered readable, more than 0.1% of codebase containing single letter variable names
Possible Causes of Failure	User error

TPM ID	TPM_N03.2b
TPM Name	Code Documentation
TPM Purpose	Ensure that the code base is easily readable and possible to be understood by a person outside of the project team

Source Requirement	ReqN03.2b
Risk Level	Low
What should be measured?	Number of functions, percentage of functions that lack documentation
How should it be measured?	1. Count the total number of functions 2. Record the number of comments above functions that communicate the purpose and meaning of inputs, outputs and limitations, as well as purpose of the function
Measure of Success	99.5% or more of the code base well documented
Measure of Failure	Less than 99.5% well document or more than 0.1% of documentation blocks missing some of the descriptions specified above
Possible Causes of Failure	User error

TPM ID	TPM_N04.1
TPM Name	Boot Time
TPM Purpose	Ensure that the system is able to recover quickly in event of loss of power
Source Requirement	ReqN04.1
Risk Level	Moderate
What should be measured?	The time it takes from connecting the power source to readiness of all functional modules of the cube
How should it be measured?	1. Start the timer and turn on the power supply 2. Await a “ready” signal from the system, inspect that all motors are at a constant, operational speed and system is ready to accept marbles 3. Stop the timer 4. Record result and repeat 5 times
Measure of Success	Full operational capability achieved within 30 seconds

	after enabling power supply on each of the 5 cycles. Discovery of perpetual motion after disconnection of the power supply
Measure of Failure	One or more attempts results in power up time greater than 30s or failure to achieve full operational capability
Possible Causes of Failure	- User error - Improper component choice - Design defects - Manufacturing defects - Software bugs

TPM ID	TPM_I01
TPM Name	Power Management
TPM Purpose	Ensure that the power consumption of the system does not exceed the budget
Source Requirement	ReqI01
Risk Level	Moderate
What should be measured?	Total maximum power draw of all installed components
How should it be measured?	1. Record maximum power draw values from component specification sheets 2. Multiply values by the number of units for each of the components 3. Calculate the total power draw for each power bus supplied to the cube 4. Compare values
Measure of Success	Maximum power draw on the 5V bus does not exceed 5W and 12W is not exceeded on the 12V bus OR power management strategies are implemented to limit power draw at any time to within the mentioned limits

Measure of Failure	Power draw on one or both of the buses is exceeded at any time, for any amount of time
Possible Causes of Failure	- User error - Improper component choice - Design defects - Manufacturing defects - Software bugs

5 Conceptual Design

5.1 Initial Analysis and Calculations

Initial analysis of the problem definition allowed the team to develop high level system models to ensure the end result meets the customers specifications. The team set up TPMs to ensure the requirements would be met while coming up with the final design.

A high level overview of the system started to materialise with different options and configurations for the subsystems. These were further narrowed down to specific components, practices and integrations to ensure the system would meet the customer requirements. Below the subsystems, design alternatives and final design is elaborated on to detail the design of the Marble Machine.

5.1.1 Micro-Controller Considerations

Perhaps the most critical component of the communications teams system was microcontroller choice. A microcontroller needs to be used to designate logic of the system. The microcontroller is required as the system to meet a number of requirements (incl ReqF03, ReqF02). The selected microcontroller should be able to manage digital and analog inputs allowing us to interact with a variety of sensors, lighting and electronic components.

The client is supplying a 12v and 5v power input which can be utilised to power the microcontroller and other electrical components. To assist with development it is ideal if the microcontroller has widespread community support and available libraries will aid code production and maintenance. The numerous features of the microcontroller should include a reset button, onboard LED indicators, and expansion shields, all which will aid in the development and maintenance of the machine.

Considerations by the Kamakura team for the function and purpose of this device are to manage the lighting(ReqF03), sensors, communication and motors(ReqF11, ReqF12) throughout the Marble Machine. This requires the microcontroller to send and receive simultaneously, or at least at an interval that can meet the requirements set out above.

The team also reviewed the hardware and corresponding software specifications of each device to ensure it has the capability to manage the logical workload of the system. This alongside minimising the cost(ReqN09) of the device to allow flexibility for the Motions and Structures team.

5.1.2 Communication Considerations

A number of considerations need to be made when designing and implementing a communication system involving multiple microcontrollers. Note this is not a requirement of the project but will improve the overall aesthetics of the entire sculpture.

The two most important considerations for communication is that it doesn't impede the software that operates the lighting, sensors or motors, and that it doesn’t use necessary hardware for lighting, sensors or motors. Additional considerations that the team determined to be fundamentally important to meet non-functional requirements included, data framing method(ReqN04.1), latency and responsiveness, error reporting(ReqN06.2), power consumption(ReqI01), robustness(ReqN06.2) and scalability.

The communication aspect of the microcontroller needs to be simple to implement and modular, such that it can be easily integrated into additional Marble Machine projects, such that current and future teams working on their Marble Machines can easily implement.

5.1.3 Marble Movement and Control Considerations

Communications group designated a logic subsystem to manage the marbles movement throughout the system(ReqF06, ReqF12). The subsystem was further split off into subcategories, Controller the Motor and Driving the Motor. This subsystem requires sophisticated software to ensure precise, synchronised, and dynamic movement of marbles (ReqF06) throughout the system.

The primary challenge involves developing software that can orchestrate these variables through coordinating with the movement of various mechanical components, readings from sensors (ReqF03) and power needed to steadily run components whilst achieving previously specified requirements. Key considerations when generating conceptual designs for this subsystem include:

- Timing: Coordinating between sensor output data and motor movement controls to ensure optimal marble throughout the machine (e.g. Motors calibrated for the 1 marble per second input).
- Sensor Integration: Ensuring each sensor is efficiently used and properly mapped in the software to correctly output data to movement components related to a function / action (e.g. Sensor recognises marble entered stationary event (LC2) and activates movement after specified time period).
- Track Control: Motors directing the movement of the marble around the track should be programmed to output correct amounts of momentum where marbles will not dislodge or obstruct from the track.
- Power budget: Ensure efficient power usage of motor, as unnecessary power affects the overall performance of the machine.

In response to these challenges, the subsystems have been split into two sections, driving the motor and managing motor RPM. A number of design alternatives have been created offering diverse strategies for tackling the track and movement control subsystem.

5.1.4 Runtime Considerations

The runtime is the heart of the software system as such the choice of runtime is paramount to the success and efficiency of which we can achieve the overall project. The choice in runtime is either to use an off the shelf real time operating system such as FreeRTOS or to create a bare metal algorithm where issues such as timing and task execution are controlled by the main superloop (ReqN02). The main trade off that occurs when considering whether to use a real time operating system or not for this project comes down to a consideration of memory and processing power.

Adding a real time operating system does increase the need for memory and adds overheads in terms of processing power. The hardware provided in the Arduino Uno, should be able to support the added memory and processing overheads for FreeRTOS as the tasks we have to use are not massively complex or memory intensive. This allows us to gain the many benefits of using a real time operating system such as increases in speed of development due to having more known working software and less to debug as well as more efficient scheduling of tasks and access to what amounts to multithreading on the single core of the Arduino Uno.

5.1.5 Lighting Considerations

The project aims to incorporate a lighting system into the module that has two functions; illuminating the contents inside the Marble Machine and providing visual cues in line with the system's state. A number of considerations have been put forward to gauge the appropriateness of the different lighting technologies for the final design. Metrics include, power consumption, power requirements(i.e. constant voltage, constant current), cost, visual aesthetics (ReqF03), brightness, hue, saturation, colour range and contrast compared with the final box design.

Additional considerations include, having the lighting solution run directly off the microcontroller for the final system to require less components and compatibility with a simple, well documented, and easily utilisable software library. This will allow for seamless integration with the microcontroller reducing hardware components and associated cost.

These considerations will ensure the team meets the designated non-functional requirements, (specifically ReqN01, ReqN03, ReqN08). The goal is to enhance user engagement and improve user experience.

5.1.6 Hardware Interface Considerations

Working with a microprocessor and various peripherals requires considerations on how they will interface and work together as a single system. The Communication team had key considerations when evaluating appropriateness of the hardware interfaces. This included voltage compatibility, current limitations, microcontroller compatibility, communication protocols required and necessary documentation, to name few.

The Hardware Interface is a core component of the system, its reliability (ReqN01) is critical for the project. As an abstraction layer between the hardware components and core functional logic. The hardware employs a limited number of external interrupt interfaces and therefore a sensor polling routine is required to meet project requirements (ReqF03, ReqF12, ReqF13).

The hardware interrupt capabilities should be reserved for timing critical operations like support for motor encoders and must be composed of as little instructions as possible to allow for execution of tasks supporting lighting and motions.

5.1.7 Sensor Considerations

The functionality of the sensors in the overall system was to indicate the position of a single marble at some location of the system. Marbles passing through the system can be made of a number of different materials, however they need to be a sphere 16mm in diameter and weigh 16 grams. The client has not stated that the marble material has not stated that the material cannot be transparent.

With these details in mind the Communication group had additional thoughts on the sensor to ensure appropriate functionality. This included easy integration into the microprocessor software (ReqN03 and ReqN04), minimising a non-detection event (ReqN01) and that it met power and budgetary requirements (ReqN07).

5.1.8 System-Wide Considerations

The subcomponents of the system and the respective considerations have been listed above. An additional consideration is required to ensure all the components integrate and function appropriately together. The client has imposed requirements of the system which need to be addressed as an accumulation of these subsystems:

- Power consumption - the power budget of the entire system is limited to 18 watts. Considerations need to be made such that the power budget of the entire system remains under 18 watts. Reducing the power consumption of individual components is a priority (after meeting system requirements) such that the power budget can be best utilised for motors
- Size - the size of each component should be minimised to allow for the Motions and Structures group to meet the physical requirements, (specifically ReqF01, ReqF02, ReqF04
- Software interfacing - all components should be able to be controlled directly off the microcontroller, OR an small inexpensive module/shield that connects to the microcontroller. The components should be easily controlled through the software running on the microcontroller, AND each component be able to be used simultaneously.

- Power delivery - having all the components run off the microcontroller will allow the system to fewer hardware devices, which reduces cost and complexity. Ensuring the power controller stays within nominal power and heat ranges during normal operation is advised.
- Inter-subsystem connections - managing the various different connections between the logic components. Wiring, placement of required elements and standardising the different types of connections.

Each option has its features and potential drawbacks, using this the Communication team conducted a comprehensive review of each subsystem and the entire system, weighing up the advantages and disadvantages of the combinations of components that make up the entire system. With these considerations in mind for the whole system, the team has provided documentation of the system that aligns with the requirements, constraints and assumptions.

5.2 Design Alternatives

Tables for the respective alternatives are made available in section 5.3 providing an overview into the research and considerations made by the Kamakura Communications group. The Communications team narrowed this down to a number of viable options, the factors influencing the suitability are listed in the tables below. This narrowed our decision making to a smaller number of component configurations, allowing the team to make a more informed decision for the final design.

5.2.1 Micro-Controller Alternatives

Over the last 50 years there have been countless microcontrollers developed all with a variety of uses and specialisations. Narrowing down the alternatives based on functionality and cost the team discussed the Arduino Uno R3, the ESP32 and the Beaglebone Black. Considerations listed above have filtered these candidate microprocessors to ensure the team meets the requirements of the system.

5.2.2 Communication Alternatives

The communication protocols commonly used on the Arduino are Serial Communication(UART), I2C(Inter-Integrated circuit), SPI(Serial Peripheral Interface), CAN(Controller Area Network), ModBus and LoRa. Interfacing between a number of components requires the team to utilise a number of different communication protocols.

5.2.3 Marble Movement and Control Alternatives

To ensure the necessary movement of the marble throughout the system to meet the requirements, a number of different options need to be considered. Separating these into two sections, driving the motor and controlling motor RPM.

Potential methods for driving the motor included, manually toggling the motor’s power source, N Channel Mosfet - P30N06LE (signalling a transistor to regulate motor power supply), and the L293D Motor Driver with PWM (Pulse Width Modulation) capabilities.

Alternatives for controlling the motor RPM included, setting constant voltage and current for static motor speed, Proportional Integral Differential(PID) Tuning and Proportional adjusting.

Proportional adjusting uses a feedback control system using current error against the desired setpoint to generate a control signal for the motor. PID tuning adds to this fundamental idea such that it accumulates the errors using an integral and calculates the control signal using the derivative. Past errors are accumulated allowing the system to adjust for any constant biases, and better management of voltage overcorrection for output speed relative to the setpoint.

5.2.4 Runtime Alternatives

For the runtime there are two choices which largely affect the overall design of the program. Either a bare metal design which utilises a superloop or a lightweight operating system namely freeRTOS.

5.2.5 Lighting Alternatives

A vast number of lighting technologies and products are available and can be used in this project. The Communication group decided to investigate further into products available for these different technologies. The potentials that have been considered include WS2812b LEDs Strip Light (directly addressable LEDs), RGB LED Strip Light(not directly addressable LEDs) and Luce Bella Incandescent Salt Lamp 12W 12V Globe. As many technologies have similar specifications the alternatives that have been reviewed showed the best value in their respective technologies.

5.2.6 Hardware Interface Alternatives

As an abstraction layer, alternative designs for the Hardware Interface revolved around its responsibilities and ‘position’ in the software stack. The project team explored many implementations including enclosing the Hardware Interface in a separate library, and various configurations of dedicated tasks. Those were later assessed against the requirements and TPMs to ensure best project outcome.

As computing resources of the Arduino are limited, and interrupts have to reach multiple functional modules at a time, creating a separate library was deemed unsuitable due to the vastly increased number of input polls. The input and output monitoring functionalities were not separated into additional tasks due to the added complexity, which incurs increased memory usage.

5.2.7 Sensor Alternatives

The types of sensors that could work in this system are vast, IR, motion, proximity and pressure sensors to name a few. The team had to narrow down the choices to that which would serve for the greatest functionality. The potential sensors under consideration are a 3mm IR beam break sensor, and a metal proximity switch.

5.2.8 System-Wide Alternatives

Collaboration between the Structures, Motion and Communication groups requires a dynamic approach to fulfilling the clients requirements. Therefore the Communication group has come up with alternatives to ensure each group in the Kamakura Team has some flexibility in their design. Notably there are a number of physical layouts that can work within the box to satisfy the logic requirements of the Communication group.

Notably the placement of the microcontroller can be anywhere inside the box. The first option for microcontroller placement is behind a false back, placing all the logic components behind this will aid the aesthetics of the Marble Machine. The position can be anywhere behind the false back, as long as ports run through the false back to supply power and control to the various peripherals.

Another option is to place the microcontroller and wiring on a side wall where little or no structural elements are. This will be out of the way but visible to the client, which may impact the aesthetics of the system.

As there will be a number of components that run on the 12v power supply, the wiring and connections need to be considered. Soldering the wires together then wrapping them with heat shrink will allow strength and rigidity (ReqN01) however this limits flexibility and reduces the modular nature of the system. An alternative is to use a small conductive bolt as a hub style connection that each component can be connected to and the bolt tightened to secure the wires. This makes the system modular, however can be problematic if too many wires are connected or poor installation.

The connections to the Arduino for the various communication and power requirements needs to be considered. The Arduino has female connections for all digital, analog and power pins allowing a male Dupont connector for all aspects. The female connections can be removed from the Arduino giving access to the connections directly on the board. Cabling can then be semi-permanently attached using solder. This however leaves conductive material exposed which may reduce the reliability and maintainability of the project (ReqN01, ReqN03).

5.3 Comparison of Alternatives

5.3.1 Micro-Controller

Microcontroller	Description	Advantages	Disadvantages	Suitability
Arduino Uno R3	Open-source microcontroller platform designed for ease of use, designed for both beginners and experienced programmers	- Simplicity - Ease of use - Open source nature - Community and documentation - Low cost	- Limited processing - - Power consumption - Limited memory - Limited connectivity	High
BeagleBoard Black REV C	Single-board computers, high-performance processors and versatile connectivity options. Well-suited for a wide range of embedded and IoT applications.	- Connectivity - High performance - Community and documentation	- Complexity - Power consumption - Cost	Low
ESP32 Development Board (ESP32 – WROOM-32 – DevKitC)	A versatile microcontroller, built-in Wi-Fi and Bluetooth capabilities. Ideal for IoT and wireless communication projects.	- Connectivity - Low cost - Low power modes	- Limited processing - - Power consumption - Limited memory - Limited GPIO pins	Low

5.3.2 Communication Protocols

Communication Protocols	Description	Advantages	Disadvantages	Suitability
Serial Communication Universal Asynchronous Receiver-Transmitter (UART)	Data is exchanged between devices facilitated by UART using transmit and receive pins. Typically used between two microcontrollers and also bluetooth modules.	- Simple - Widely supported - Low pin count - Real time debugging	- Limited transmission distance	High
I2C Inter-Integrated Circuit	A multi-device bus communication framework. Typical usage includes sensors, displays and EEPROMs	- Multi-device communication - Low pin count - Standardised protocol	- Limited distance - Slower transmission speed	Medium
SPI(Serial Peripheral Interface)	A high speed full duplex synchronous communication method.	- High speed communication - FullDuplex - Flexible	- More pins required - Complex wiring	Low
CAN (Controller Area Network)	Supports multi-node networks with high resilience to support data integrity. Typical usage includes automotive diagnostics and industrial automation.	- Robust communication - Fault tolerant - Multi-node networks	- Complex protocol - Higher pin count	Medium

Wireless Communication Protocols	Wireless communications allow communications without physical connections. Applications include IoT projects, remote control and wireless sensor networks	- Wireless - Flexible - IoT integration	- Interference - Power consumption	Low

5.3.3 Driving the Motor

Driving the Motor Method	Description	Advantages	Disadvantages	Suitability
Manually toggling the motor’s power source	Control is achieved by manually switching the motor’s power supply on or off.	- Simple - Low complexity	- Limited control - Lack of feedback	Low
N Channel Mosfet - P30N06LE	The microcontroller sends a signal to the transistor, which regulates the motor’s power supply.	- Voltage control - Power efficiency	- Moderate complexity - No advanced features	Low
L293D Motor Driver	Motor driver using PWM (Pulse Width Modulation), enabling fine-tuned speed adjustments and dynamic direction changes.	- High precision control - PWM capabilities - Advanced features	- Higher complexity - Cost consideration	High

5.3.4 Managing Motor RPM

Motor RPM	Description	Advantages	Disadvantages	Suitability
Constant Speed	The motor is set to a specific RPM value, and no adjustments are made during runtime. This approach is suitable for applications where speed variation is not a critical requirement.	- Simple - No external devices	- Zero adaptability - Low consistency - Precision problem	Low
PID Tuning (Proportional, Integral, Differential)	A PID control loop is implemented to maintain a constant motor RPM. The PID controller adjusts the motor's power supply based on the difference between the desired RPM and the actual RPM, using proportional, integral, and derivative terms for control.	- Precise RPM control - High adaptability - Higher reliability	- Complex implementation - High tuning effort - Advanced PID tuning knowledge	High
Proportional Adjusting	This design involves periodically measuring the motor's RPM and adjusting the power supply proportionally to maintain the desired RPM. Rather than using a PID loop, the system makes coarse adjustments based on the RPM error.	- Simplicity - Moderate adaptability - Moderate RPM control - Reliability	- Limited precision - Static adjustments - Low tuning - Availability	Moderate

5.3.5 Lighting

Lighting method	Description	Advantages	Disadvantages	Suitability
WS2812b LEDs Strip Light	Light Emitting Diodes are solid state sources that work by passing a current through a semiconductor material. Each LED in the strip can be addressed and provide unique output	- Lower energy consumption - Considerable lifespan - Compact form factor - Highly dynamic colour manipulation	- High initial investment - Heat sensitivity - Higher programming complexity	High
RGB LED Strip Light (not directly addressable)	Light Emitting Diodes are solid state sources that work by passing a current through a semiconductor material. All LEDs in the strip have unified colour, saturation and hue. Single LEDs in strip are not addressable	- Lower energy consumption - Considerable lifespan - Compact form factor - Less programming complexity	- Medium initial investment - Heat sensitivity - Medium programming complexity - Less dynamic colour scheme	High
Luce Bella Incandescent Salt Lamp 12W 12V Globe.	Incandescent light bulbs work by passing an electric current through a wire filament that heats up and glows to produce light.	- Cost effective - Warm lighting - Simple implementation	- High energy consumption - Shorter operational life - Large heat production - Static colour scheme	Low

5.3.6 Hardware Interface

Architectural Placement (?)	Description	Advantages	Disadvantages	Suitability
Individualised hardware interfacing schemes for every functional module	Hardware interfacing code implemented ad-hoc in every section of software that utilises it	- Easy to read and understand; Bugs in one module may not affect the other; - Sensor polling rate different for each module	- Lower maintainability due to inherent code duplication (ReqN03.1, ReqN03.2), - Sensor polled by multiple modules leading to increased resource use	Low
Abstraction Layer with separate Input and Output interface tasks	Hardware Interfacing code implemented in one specialised module composed of 3 tasks, working asynchronously	- Respects differing timing needs of sensors and outputs; - Core Hardware Interface only active when there are pending updates, resulting in lower CPU usage	- Increased memory usage due to presence of additional tasks; - Hardware Interface would run and stop frequently due to some functions requiring constant updates; - Increased complexity may result in lower reliability (ReqN01.*)	Moderate

Abstraction Layer with one task	Hardware Interfacing code runs in a separate, single task, with additional	- Tighter integration results in lower memory usage; Timing of the Hardware Interface can be tied to external interrupts	- Increased CPU usage due to additional code included in the module;	High

5.3.7 Sensors

Sensor Type	Description	Advantages	Disadvantages	Suitability
IR Break Beam Sensor - 3mm LEDs	An infrared sensor with send and receive components	- Very low power consumption - High precision - High detection	- Accurate mounting required - Two hardware components required	High
5V Metal Proximity Switch	Uses ultrasound to detect distance. If marble crosses the sensors path the distance changes	- Quick response time - Low power consumption	- Cost - Inaccurate readings	Low

5.3.8 Digital Switch

Sensor Type	Description	Advantages	Disadvantages	Suitability
N-Channel MOSFET 60V 30A (P30N06LE)	Digital switch, that can also manage output voltage	- Very low resistance - Control voltage - Handles up to 30A at 60v	- Cost - Possible heat production	High
TIP120 Power Darlington Transistors	Powerful little electronic switch	- Low resistance - Handles up to 5A at 20v	- Cost - Temperature sensitive - Slow high/low switching	Medium

5.3.9 Runtime Environment

Runtime	Description	Advantages	Disadvantages	Suitability
Bare Metal	Using a bare metal design, the software would be designed such that there is one central loop that runs forever (superloop) this would make sequential calls to functions. For functions that require actions at any time such as the input/output manager functions then would need to be implemented with service interrupts.	- Memory efficient - Predictable performance - Low latency - Ease of debugging	-Sequential execution -Complexity in development -Lack of abstraction -Lack of multi threading	medium-high
FreeRTOS	FreeRTOS is an open source and almost industry de facto runtime operating system. A runtime operating system is a lightweight operating system that offers some basic functions that you would expect from an operating system such as: Schedule background tasks, Manage virtual folders and Management of device drivers. For this project's use case the most	- Multithreaded like execution of code - Well documented and tested - Allows for faster development time overall - Actively Open Source - Ease of task scheduling - Relatively lightweight (low overhead)	- Processing and memory overheads - Added complexity - Harder Debugging - Unpredictable performance - Higher latency than bare metal	High

	useful is the scheduling of background tasks. This essentially allows for something that sort of resembles multithreading on the single core Arduino uno. As such the implementation of each routine (function) as its own task allows the OS to schedule the task.			

6 Detailed Design

6.1 Specific Design Details

In this section the particulars of the project the Kamakura Communication group will be discussed at depth. Sufficient detail of each component will be discussed to give both particular component detail, sub-system and total system final design. This will also link the design back to the original requirements set out by the client to display how all requirements are being met.

6.1.1 Micro-Controller and Connections

The microcontroller most suitable to meet the clients requirements is the Arduino Uno R3. The board offers excellent operating ranges, the main processor runs at 20 MHz, adequate number of digital GPIO pins, operating temperatures between -45℃ and 80℃, and excellent documentation, libraries and time tested applications, this microcontroller can manage all logic required of the system. Power consumption of the Arduino with all components operational is calculated as a maximum at 1000 milliwatts.

The Arduino can be placed in the back of the box with a ‘false back’ to reduce the negative visual aesthetics that the cables and connections will impose. This allows all cabling and electronic components to be visually minimised and increases the aesthetics of the box. Majority of components the cabling has been minimised through appropriate placement and orientation.

The next paragraphs go into detail about the wiring schematic for the Arduino. External connections are provided through the service port(ReqF04), the Arduino will be powered through the 12v power connection supplied by the client(schematics in Appendix A). Male connections for the interface have been ordered through the Core Electronics Supplier and can be found in the finalised Bill Of Materials (located in section 6.4.1). The Arduino will have an additional 20 AWG cable running through the service port for use as a serial communication receiver for the Marble Machine. The single wire along with the negative 12v power cable will allow inter box communication.

Lighting for the Marble Machine will be controlled directly from the digital pins of the Arduino, but powered from a node connecting the power rail from the service port. The digital pins 5 and 6 will be used to control four 200mm LED strips, each digital pin can be set to send a signal and provide up to 40mA of current. This is sufficient to manipulate the ‘digital in’ of each LED ws2812b strip and directly address each individual LED in each strip.

Sensors for the Marble Machine will run directly from the Arduino, as the IR beam break sensor will indicate a marble entering the system. This sensor requires minimal power (approx 0.1 Watt) which can be run directly off the Arduino, digital pin 8 will be set to write, which can update the Arduino state.

Motors are required to provide potential and kinetic energy to each marble. The

Archimedes screw will provide the potential energy such that the requirements ReqF10, ReqF11, ReqF12. The energy requirement of these subsystems will exceed that which can be provided through the Arduino, therefore a mosfet will be required. The N- Channel mosfet that will be used allows a small current from the Arduino to act as a switch and voltage regulator to manage the speed of the motors.

6.1.2 Inter Box Communication

The communication protocol that has been chosen by the team is the UART Serial Communication method. UART has been chosen to communicate the unified aesthetic across all the boxes. The project does not require high bandwidth, with sensor data being transmitted to the 'master' Arduino requiring only a single byte, colour, hue, saturation and brightness requiring less than two bytes. This allows the communication protocol to remain simple, allowing each team the ability to standardise bitrates and connection protocols across all Marble Machines.

The UART communication method allows intercommunication between all boxes. The cabling across the greater sculpture will remain static allowing for box position to reflect a particular digital pin on the master Arduino, allowing for individual addressable boxes. The Marble Machines will all utilise the same receiver pin but the master arduino will have pins set for the different positions in the greater sculpture. This allows the overall entire structure to create synchronised lighting patterns.

This is an optional addition of all teams to have synchronisation across the greater sculpture. Adhering to the UART standards and specifications, are the considerations adopted in the design for optimal operation laid out below.

High level overview shows each 'slave' Arduino communicating using UART using a single communication line, the data is sent across the communication line which can be received by any or all of the individual marble machines. This allows the master Arduino to dictate the overall aesthetics of the greater sculpture. Each marble machine shall contain a 'slave' Arduino that receives serial data from the 'master'. Only the 'master' Arduino will be able to send data to all the boxes. The master will be separate and not contained in any individual marble machine, but still a part of the larger sculpture. Specifications of the Open System Interconnection(OSI) model(Alani 2014) each layer and its role in this are outlined here.

The Physical Layer will manage the physical connection between devices, allowing raw binary data to be transferred over the physical medium. The connection required for the Arduino to connect to the overall sculpture will be soldered using 20 Awg (Appendix B) to the Arduino pins D0(RX) and D1(TX) and connected to a 2 Pin Female Polarized Header Connector. This includes the TX and RX pins, common ground and their respective cabling for transferring the data. Using a two wire balanced signalling scheme and bus topology connecting the ‘master’ Arduino to all the slaves, allows quick and easy connection between the greater sculpture for maintenance and diagnostics.

The bitrate is set for this protocol at 9600 bits/s, sufficient speed for the colour variables to be updated, it also aligns with the typical bitrate set on other teams' Arduinos. The physical layer of this communication method will be following the RS232 standards(O’Reilly 2019), this is the most commonly used and defined by the Serial Communication Standards.

The Data Link Layer will handle data framing, error detection and flow control. In this design the serial communication protocol will be managed by the SoftwareSerial package. Including start and stop bits for framing and error checking.

The Network, Transport, Session and Presentation Layers functionalities are handled at the Application layer. The functionality of data being sent to slave Arduino’s requires no session management, no formatting/encryption or compression, no routing or error handling. All devices receive all communication sent by the master and choose to disregard or receive, allowing many of the OSI model layers to use the Application layer and its functionalities.

The Application Layer will handle the user interface of updating and maintaining the system and any updates coming from the ‘master’ Arduino. Any custom implementations of the LEDs including time of day settings or lighting levels. The protocol implementations between the master and slave Arduino for data formatting and how data is structured. The ability to translate the received data into a behavioural change on the LED colours and logic. As a ball passes a sensor in the box it must incorporate a lighting pattern and the received colour values. If no message has been received from the master in a predetermined amount of time, a default lighting sequence will be initiated.

This will be used for communication between the box being designed by our team and the greater sculpture, allowing all cubes participating to coordinate the lighting aesthetic. Its ease of integration, inexpensive additional hardware requirements and simple message structure makes it ideal for this application. Modifications to this design will allow two way communication, no error handling simplifying communication of the overall structure. In this iteration of the project only colour and brightness values will be transmitted to each box, in further iterations of this design numerous characteristics can be monitored with two way communication introduced to the overall sculpture. This will however require some additional hardware components

This will not be used in initial implementations of this communication method but may be easily upgraded to deliver two way communication from the ‘slave’ (individual boxes) to the ‘master’. The wiring diagram to connect these devices can be seen in Appendix A.3. The system has been designed to be able to accept input data from sensors at all times. The communication methods being used for the input data allows for sensor data to be processed back to the master Arduino

6.1.3 Motor and Modulation

The machine requires precise marble control throughout the marble machine. A requirement of the machine is to expel every marble 500mm out the top of the box. Precise control over the momentum and speed of the marble is required to achieve this requirement. Therefore the motor in control of each marble's exit speed must be running at a desired RPM. Keeping that RPM is essential to meeting the clients requirements and will be achieved using precise motor tuning.

Therefore, we have chosen to use a motor driver and PID tuning in order to produce the desired behaviours of the system for managing power budget keeping the motor at a constant RPM. This approach involves utilising a dedicated motor driver with PWM capabilities, allowing the microprocessor to manage the motor's speed.
PWM refers to controlling the speed of the motor by switching the motor on in a series of pulses. A Motor driver with PWM abilities will control the motor speed by modulating the width of the pulses that supply power to the motor.
In order to achieve this control using PWM we will be using the L293D Motor Driver. This L293D is a suitable design for our project as it is designed for lower current applications, making it a great fit for our low power budget. Additionally, the L293D design is small and simplistic, it has a simpler pin configuration which makes it easier to work with than other modules.

In order to enable the PWM capabilities on the L293D motor driver we will need to establish the connections between the arduino, motor and the motor driver module. We will first need to connect the motors positive and negative terminal to the L293D's OUT1 and OUT2 pins respectively, the connections may be swapped around if the direction of the motor is backwards. The ENA pin on the L293D will then be connected to the D10 pin on the Arduino, this will enable our program to control the speed of the motor using PWM.

By interfacing the motor driver with the Arduino, communication is established on a channel that facilities the dynamic adjustment of motor performance.
The control loop is used by specifying the desired motor speed using PID tuning. Subsequently, the Arduino creates a PWM signal corresponding to the desired speed, this signal is then communicated to the motor driver to govern the power supply to the motor. This capability provides us with fine-grained control over our motors.

6.1.4 Lighting

Lighting The main focus is to incorporate a NeoPixel LED strip, for lighting purposes and was chosen due to its efficiency and ability to display a multitude of colours. Measuring 1 metre long it will strategically cut and positioned within the cube to effectively illuminate its contents and create a visually cohesive connection with the systems operations

What sets this LED strip apart is that each LED can be controlled individually allowing a captivating experience that combines both aesthetic appeal and functional efficiency. With this control through the arduino uno, a range of colours can be displayed while also having the ability to adjust brightness levels resulting in an engaging and visually pleasing setup.

To connect the LED strip to the Arduino Uno board, pins 5 and 6 will be used. These pins are known for their input/output capabilities and support for PWM output making it easy to control the LED strip. This choice of pins simplifies the coding process seamlessly integrating with the setup.

In terms of power consumption each LED on the strip typically uses around 60 milliwatts (or 0.06 watts) when displaying light. Considering that a typical strip can have from 30 to 60 LEDs the total power consumption of the strip could range from 1.8 to 3.6 watts. However the actual power usage would depend on the number of LEDs used after cutting the strip and their individual brightness settings while in operation.

As the project moves forward careful attention is needed to ensure that the cut and NeoPixel strip is positioned in a way that aligns perfectly with the project's goals and requirements. The aim is to strike a balance between cost, performance and visual appeal.

6.1.5 Hardware Interface

A high level overview of the functions the Hardware Interface performs are shown in Appendix C.5.1. The proposed sequence of events allows for priority handling of functionalities affecting TPMs TPM_F03b, TPM_F03c, TPM_F07, TPM_F11b, TPM_F12b and TPM_P01.2. Functional blocks are only entered when the global state is updated.

The motor encoder connected to pin 2 has 6 poles and the software interrupts will be configured to trigger on the falling edge of the signal and will be responsible for sending an event to the core Hardware Interface task over the xEncoderISRQueue and waking the task if it is currently inactive. The Hardware Interface then takes a timestamp of the event and calculates the RPM based on the following formula:

RPM = (1/(activation1 - activation0) * 60) / n_poles

With activation1 being the previous activation, and activation0 being the most recent activation event. The result of the calculation is then stored in a global variable protected by a semaphore. If the change in RPM differs by more than a specified sensitivity threshold, an event is sent to the xMotorRPMEvent queue using xQueueSend().

Additionally, the Hardware Interface stores a timestamp of the last RPM update, to assist in case of accumulation of events in the queue due to delays in other tasks. This way, the average RPM can be calculated based on the number of accumulated interrupts. Refer to Appendix C.5.2 for a flow chart of the behaviour.

Arduino Uno only supports external interrupts on GPIO pins 2 and 3. As there are more than 2 sensors, and adding a multiplexer is not possible due to budget constraints, the remaining sensors have to be polled for detection of trigger events. This is accomplished by storing the last known state of the sensors in a global array, and periodically scanning the relevant input pins for transition from ‘high to low’ or ‘low to high’ depending on the sensor type and desired activation delay.

Additionally, to prevent ghost activations there will be an additional array containing timestamps of last activation events for any given sensor. This way, events that occur at a rate higher than expected can be considered an error and disregarded. The activation events will be fed to the xMovementEvent and xLightingEvent queues to aid in fulfilling requirements ReqF03, ReqF07, ReqF11, ReqF12 and ReqF13. Refer to Appendix C.5.3 for the sensor polling routine.

The motor used in the marble lifting spiral will be controlled using a PWM signal off pin 6 through a transistor. The maximum duty/cycle durations will have an imposed maximum value tailored to the motor’s voltage requirements so as to not damage the equipment. The Hardware interface receives update events over the xMotorCtl queue which prompts it to check values stored in the global motorPowerRequest floating point variable. Only the most recent event will be used and the remaining will be disregarded. Refer to Appendix C.5.4 for the motor control flow chart.

The servomechanism will be controlled using the Servo library. Setpoints are stored in a global variable as an integer between 0 and 180. Hardware Interface is notified about updates via the xServoCtl queue, always taking the most recent event and clearing remaining updates to prevent use of stale data.

The NeoPixel LED strip contains WS2812B LEDs that are well supported by the AdaFruit NeoPixel library. The module will expect an array of int16_t of length n, with n being the number of LEDs installed in the system. This is then passed onto the Adafruit NeoPixel Strip object to be sent to the strip. The last message is kept as state and overridden by more recent data. Refer to Appendix C.5.5 for the functional flow diagram.

On startup, the Hardware Interface will first initialise the lighting by creating a Strip object from the NeoPixel library and displaying the pre-loaded “boot up” pattern which signals to the user that the system is initialising. The solenoid is initialised first by checking whether there is a marble and ejecting it, then toggling the solenoid expecting the extension and retraction to be reflected on the sensors. Any behaviour indicating a fault would result in a “failure” event being sent to the lighting module to request a change of the lighting pattern. If the solenoid initialisation is successful, the Hardware Interface will then start the motors to a predefined default power level and expect the RPM to rise to a value defined during integration in 1 second. In the event of failure, the startup sequence will be aborted, a failure event will be propagated to relevant modules, and power will be cut to all modules. Startup would be re-attempted after 30 seconds. Upon completion, a “ready” event is sent to the relevant tasks and the Hardware Interface moves to a normal operation.

6.1.6 Sensor

For the system to work as intended, the machine must give an indication of a marble passing a point(ReqF03) in the Marble Machine. To fulfil this requirement a sensor is needed to provide the microprocessor an input indicating the marble has moved through this point. The sensor selected to perform this task is the IR Break Beam Sensor - 3mm LEDs.

This sensor will be mounted where the marbles are accepted into the box. The transmitter and receiver will be on opposite sides of the entrance. Once the marble passes through the entrance the beam breaks and changes the values being received by the microcontroller. Each beam break will be connected to the 3.3v power supply off the Arduino, with the receiver having an additional digital connection to indicate if a marble has broken the beam. In this configuration there is a small 9mA draw from the sensors and well within the Arduino power tolerances.

Although two parts are required (receiver and transmitter), this sensor allows faster and better control of detecting the marble passing through. Within the program two variables will be used throughout, in order to store the state of the sensor and previous sensor state. This is to ensure that only a single change to the system occurs when the marble passes through.

6.1.7 Runtime

The best course of action was determined to be the use of freeRTOS due to the complexity involved in developing our own system and the extended time required for testing and debugging. This was done to facilitate the Software interfacing requirements. By using freeRTOS it allows for only a single microcontroller to more easily handle all the “concurrent” tasks required of it.

Most of the work done in FreeRTOS is writing tasks. Think of tasks as functions in most common programming languages, however these functions must never return anything or else the whole program will crash. To stop this from happening a common way to do this is to put a for loop with three semicolons as parameters i.e. for(;;){}. Further note that once you have finished your task you may use delay() function. This will not lock up the microcontroller in the same way that it does on bare metal as delay() simply moves the task back into a queue until the delay has passed. Allowing for other tasks to be executed in the interim. Examples can be found in Appendix C

As stated in the conceptual design the goal of the runtime is to facilitate the Software interfacing requirements. As such the runtime is to be set with the following Tasks: Lighting, HardwareInterface, Communication and MotionControl.

The purpose of the lighting function is to generate the lighting pattern and make the appropriate call to the HardwareInterface function that controls the lights.

The purpose of the Hardware Interface is to act as an intermediary layer between the custom functions created for the library and the hardware itself. For example in the Lighting Task led 1 needs to be activated. Depending on the hardware used the specific function call to do this may be different. The Hardware Interface may have a stub function called turnOnLED1(). This function would be used in the Lighting task and any hardware specific idiosyncrasies can be changed there and easily replaced should we decide to change any of the hardware components at a later date.

The communications task will be used to send and receive messages from a master controller arduino on the sculpture that will control the hue-range and brightness of the LED's. This communication standard is currently to be determined with consultation with other teams ongoing.

The purpose of MotionControl is specifically for controlling any hardware components that create some sort of motion in the system. This may include PID's for any motors that require it as well as the code of when and how to run the hardware. Specifics can be found in Appendix C (C.4.2)

6.1.8 Final Design

This section will follow the marble through the teams Marble Machine design, highlighting where each requirement is satisfied. The cube will have external dimensions of 250mm for each side(ReqF01), it will be made of 3mm MDF for all sides but the front (ReqF08). The thin walls will be kept together using wood glue along the edges and screws where the larger piece of wood is in contact(ReqF05).

The front will be attached using a hinge on the right hand side wall and be made of a transparent perspex material which allows viewers to see inside the cube (ReqF09). The cube will have additional wooden pieces on the inside of the cube that will serve as structural support for any shear force and overall cube structure through the mounting holes (ReqF02).

A 30mm diameter hole will be made in the centre bottom on the backside of the box to serve as the service port for utilities (ReqF04). At the bottom and top of the box there will be a 20mm hole where the marbles will be accepted(ReqF11) and expelled(ReqF12), the measurement requirements are in Appendix A.1 (ReqF06). This will allow the marble machine to accept marbles every second (ReqF07).

Once the marble enters from the bottom of the cube, the beam break sensor will indicate to the arduino that a marble has entered the system and will indicate this through the LEDs (ReqF03). The marble will hit a physical barrier which deflects the marble trajectory onto a wire track. The marble traverses through the cube in between lengths of galvanised wire that is 15mm apart. The track will lead the ball to the Achmedies Spiral (refer to Appendix A.4). The Achmedies spiral raises the marble to the top of the box that takes around five seconds(ReqF16). Every five seconds the motor which drives the Achmedies Spiral will be toggled off for 0.75 seconds (ReqF15). At the top of the spiral the marble will transfer to another wire track which will guide the marble around the box, changing the direction of the marble in two dimensions (ReqF14).

The marble travels around the box until it reaches the other side of the Achmedies spiral where it will again be lifted to the top of the box. It will then travel along a track to the ejection mechanism. The ejection mechanism is a rubber wheel attached to a motor. The speed is predefined speed (ReqF13) such that the marble meets appropriate speed requirements. This speed is managed using PID through a control on the Arduino. When the marble passes underneath the rubber wheel, momentum is transferred from the wheels circular momentum to the marble such that the marble has sufficient momentum to travel up an arc, through the expulsion hole and into the next box (ReqF12). Refer to Appendix A.5.

This allows the Marble Machine to meet all requirements without the use of compressed gas or liquids (ReqF10)

6.1.9 Arduino Pin usage

Arduino Pin	Peripheral	Set to Read/Write
D0 (RX)	Interbox communication receive	Read
D1	Interbox communication transmit	Write
D2	L293D motor driver for Ejection motor	Write
D3 (PWM)		
D4		
D5 (PWM)	LED light strip control 2	Read
D6 (PWM)	LED light strip control 3	Read
D7		
D8	IR break beam sensor	Write
D9 (PWM)	Mosfet for Archimedes screw	Write
D10 (PWM)		
D11 (PWM)		
D12		
D13	Arduino Light to display errors	Write

6.2 Assumptions

Adding to the assumptions that were previously made in the scoping document (1.4.1)

Assumption ID	Assumption
A14	Electricity will remain within tolerances of all components of the marble machine and no spikes in voltage or current will be incurred.
A15	Heat and pressure systems will remain in nominal ranges for appropriate operation of motors.
A16	Assume no interference from other teams, including but not limited to, moving, disconnecting, tampering or any adjustments to the Marble Machine or greater structure.
A17	Heat generated will dissipate such that temperature inside box will remain at nominal ranges
A18	System security is not in design considerations and may affect the marble exit speed if unauthorised access to microcontrollers is permitted.
A19	After the project is completed the client maintains access to the design code of each box to reimage any marble machine that is behaving incorrectly.
A20	Boxes will be stored and operated in a dry undercover area and will not endure any extreme weather events. Operation of boxes will be within appropriate temperature tolerances such that silicon based chipsets maintain operational viability and do not throw errors.
A21	Communication cabling to greater structure will not be attached to external power sources and will be within designated tolerances. The standards of the 'master' Arduino will remain the same, any communication protocols will affect the overall performance of the greater sculpture
A22	Manufactured products from suppliers are reliable and have long operating life under normal use.

6.4 Hardware Components

6.4.1 Bill of Materials

QTY	Item	Unit Cost	Supplier	Total Cost
1	Arduino Uno R3	$10	Core Electronics Australia	$10
1	LED WS2812b Lighting Strip	$5	Client supplied Core Electronics Australia	$5
1	IR Break Beam Sensor - 3mm LEDs	$7.55	Core Electronics Australia	$7.55
2	N-Channel MOSFET 60V 30A (P30N06LE)	$2.35	Core Electronics Australia	$4.70
1	250mm x 1000mm x 3mm MDF	$10	Client supplied	$10
1	Spiral Motor	$13.40	Core Electronics Australia	$13.40
1	Galvanised Tie Wire 2.00mm x 15mtr	$11.49	Bunnings Australia	$11.49
1	Jumper Wire 20cm Ribbon	$3.95	Core Electronics Australia	$3.95
1	12V 8,100 RPM DC Electric Motor	$11.95	12V 8,100 RPM DC Electric Motor | Jaycar Electronics	$11.95
Total				$78.04

References

[1] Standards Australia / Standards NZ, “Wiring Rules AS/NZS 3000:2018”, [Online]. Available:
https://infostore.saiglobal.com/preview/as/as3000/3000/3000-2018_v6.pdf?sku=1974289
Accessed: August 2023

[2] S. Bose, “Coding Standards and Best Practices to Follow”, Apr 2023, [Online]. Available: https://www.browserstack.com/guide/coding-standards-best-practices Accessed: August 2023

[3] "23S2_ENGG2K3Kv1.3 - 2023 SPINE Engineering Project ENGG2000/ENGG3000 - Massive Marvellous Multiple Marble Machine", ENGG2000/ENGG3000 (MQ), 2023. [Online]. Available: https://ilearn.mq.edu.au/mod/resource/view.php?id=7833939 Accessed: August 2023.

[4] Arduino Docs, “Arduino Style Guide for Writing Content”, Aug 2023, [Online]. Available: https://docs.Arduino.cc/learn/contributions/Arduino-writing-style-guide
Accessed: August 2023

[5] Arduino Uno R3 Documentation, N/D, [Online] Available:
UNO R3 | Arduino Documentation (accessed September 2023)

[6] iLearn, “Questions for the customer - Q&A”, Aug 2023, [Online]. Available: https://ilearn.mq.edu.au/mod/forum/discuss.php?d=2258420

[7] O’Reilly, Serial Communication Standards, September 2023[Online]. Available at 4. Serial Communications | Arduino Cookbook (oreilly.com) (accessed September 2023)

[8] Alani, Mohammed M., and Mohammed M. Alani. "OSI model." Guide to OSI and TCP/IP Models (2014): 5-17.

[9] 4qd, “What is PWM Motor Control”, N/D, [Online]. Available:
https://www.4qd.co.uk/docs/what-is-pwm/ (accessed September 2023)

[10] Arduino Language Reference, [Online]. Available:
Arduino Reference - Arduino Reference (accessed September 2023)

[11] Project Kamakura GitHub, [Online]. Available:
https://github.com/HugoCawsey/ENGG2000_3000-Project-Kamakura

[12] LastMinuteEngineers, Control DC Motors with L293D Motor Driver IC & Arduino, [Online]. Available: https://lastminuteengineers.com/l293d-dc-motor-arduino-tutorial/

Appendix A - Schematics

A.1 Marble Machine Physical Requirements

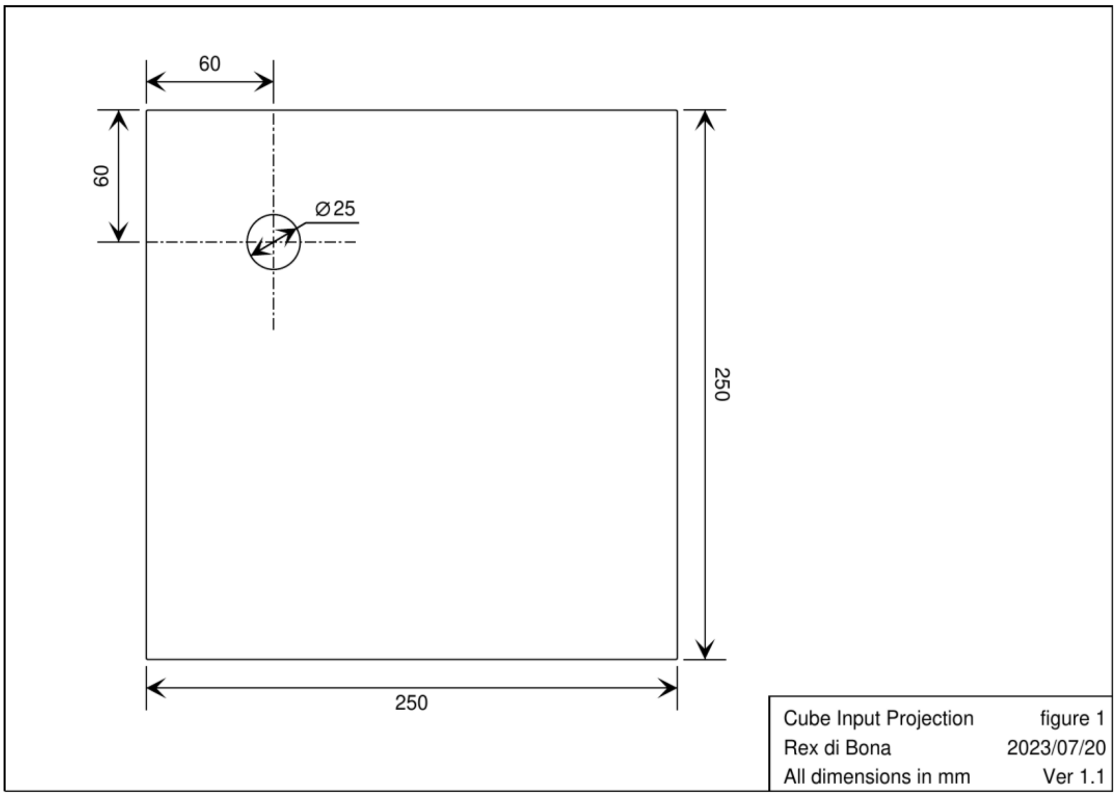


Figure 1. Input location for High Distinction

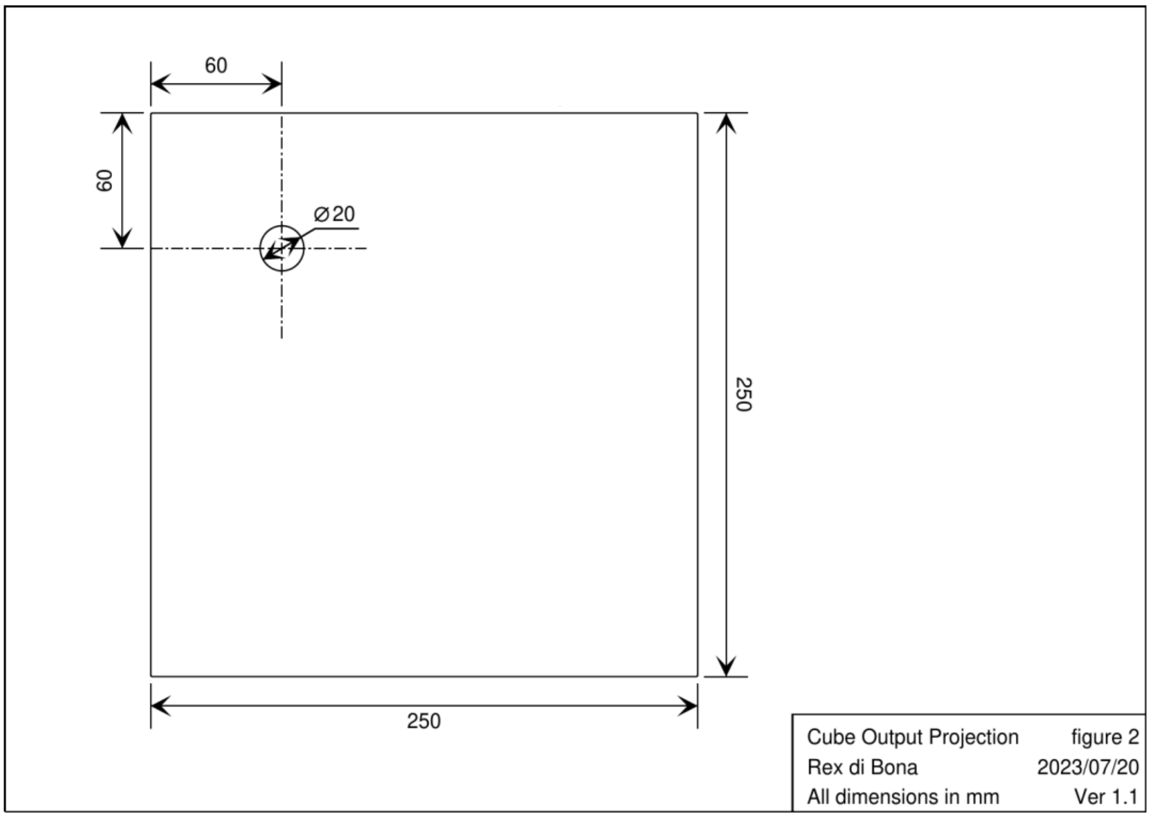


Figure 2. Cube output location for High Distinction

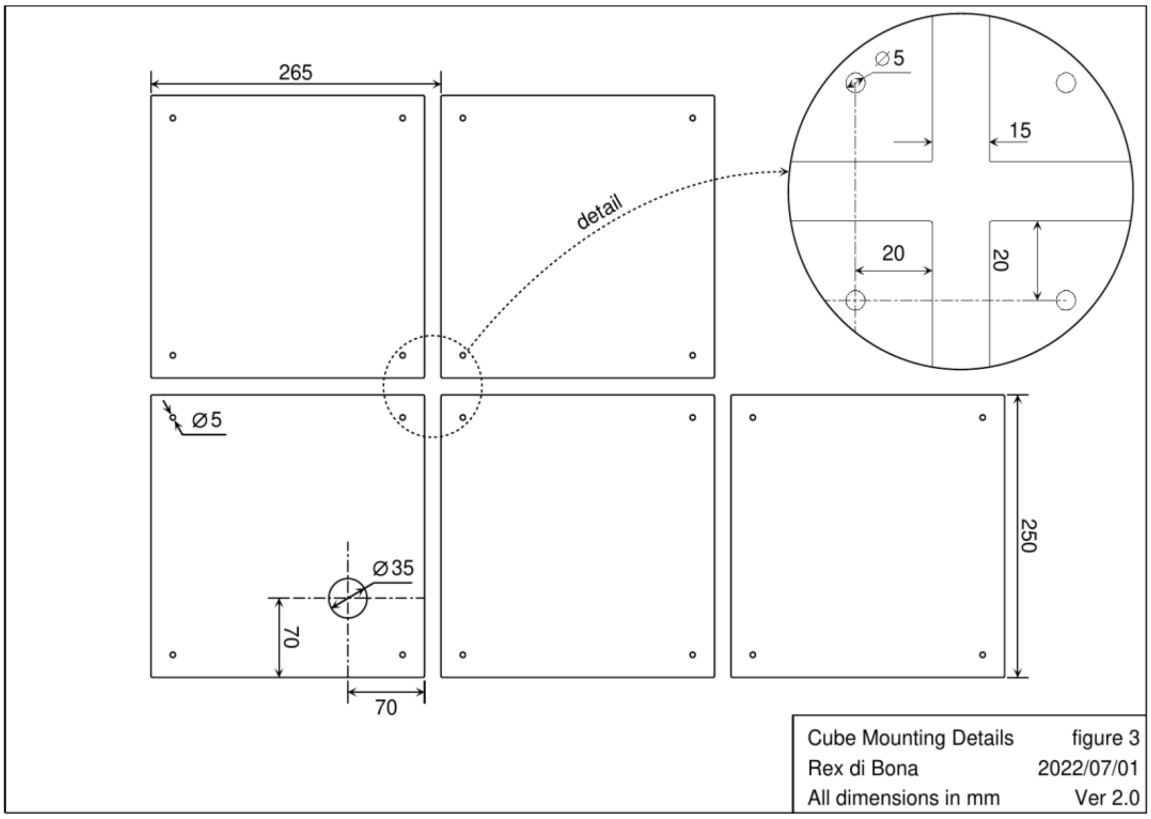
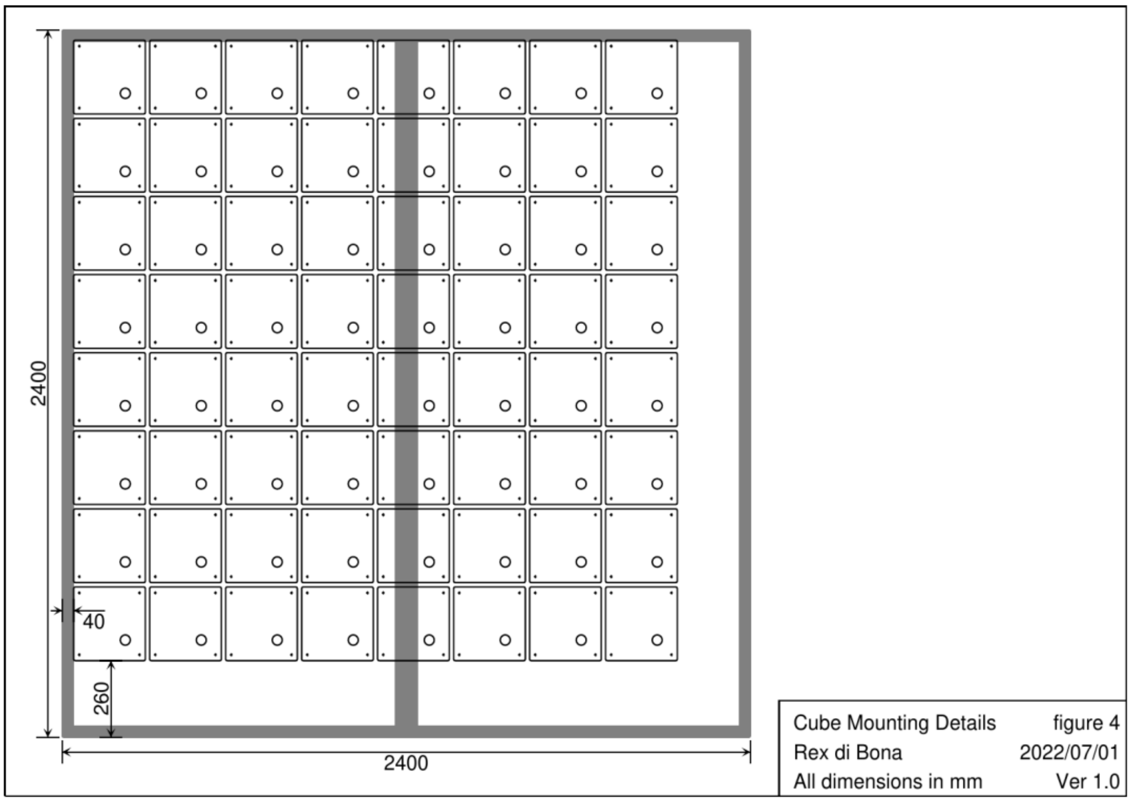


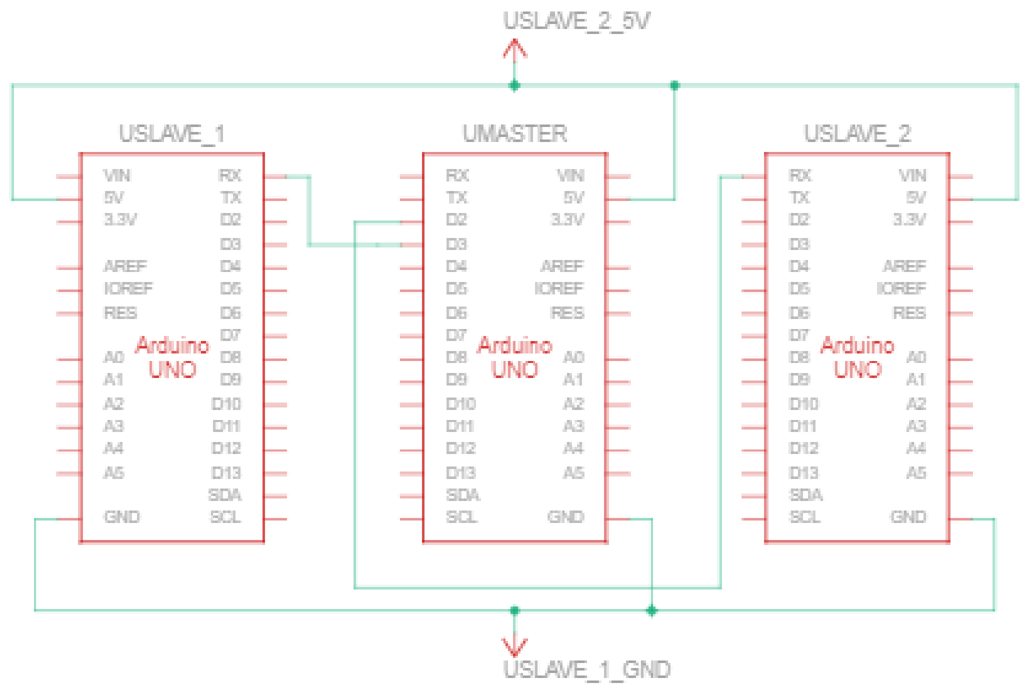
Figure 3. Cube Mounting Locations and Services Hole



Cube Mounting Details	figure 4
Rex di Bona	2022/07/01
All dimensions in mm	Ver 1.0

Figure 4. Full Mounting Frame description

A.2 Wiring Diagram for inter-box communication



A.3 High Level Subsystem

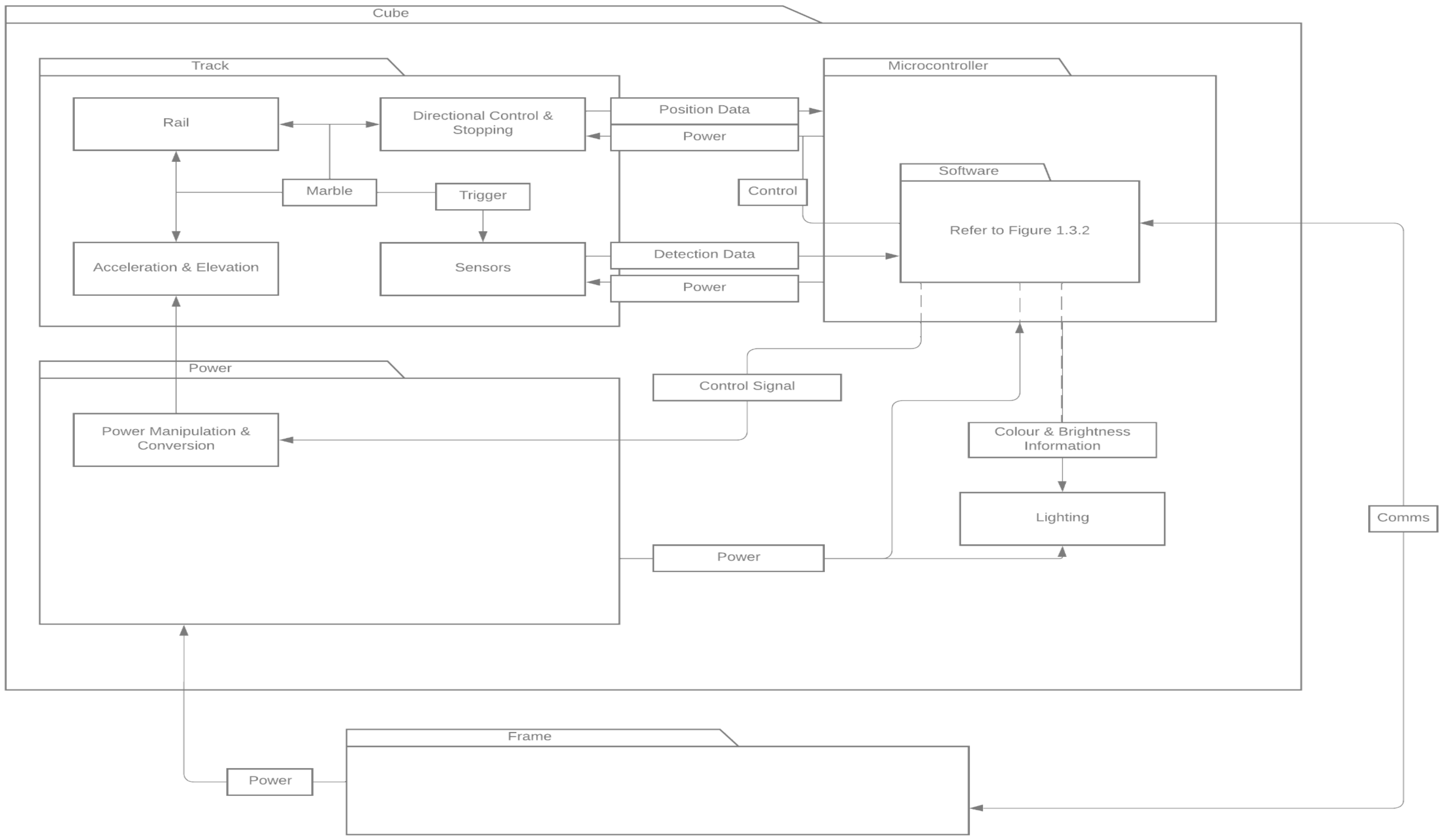


Figure 1.3.1 - High level Subsystems

A.3.2 Figure 1.3.2 - Software subsystem

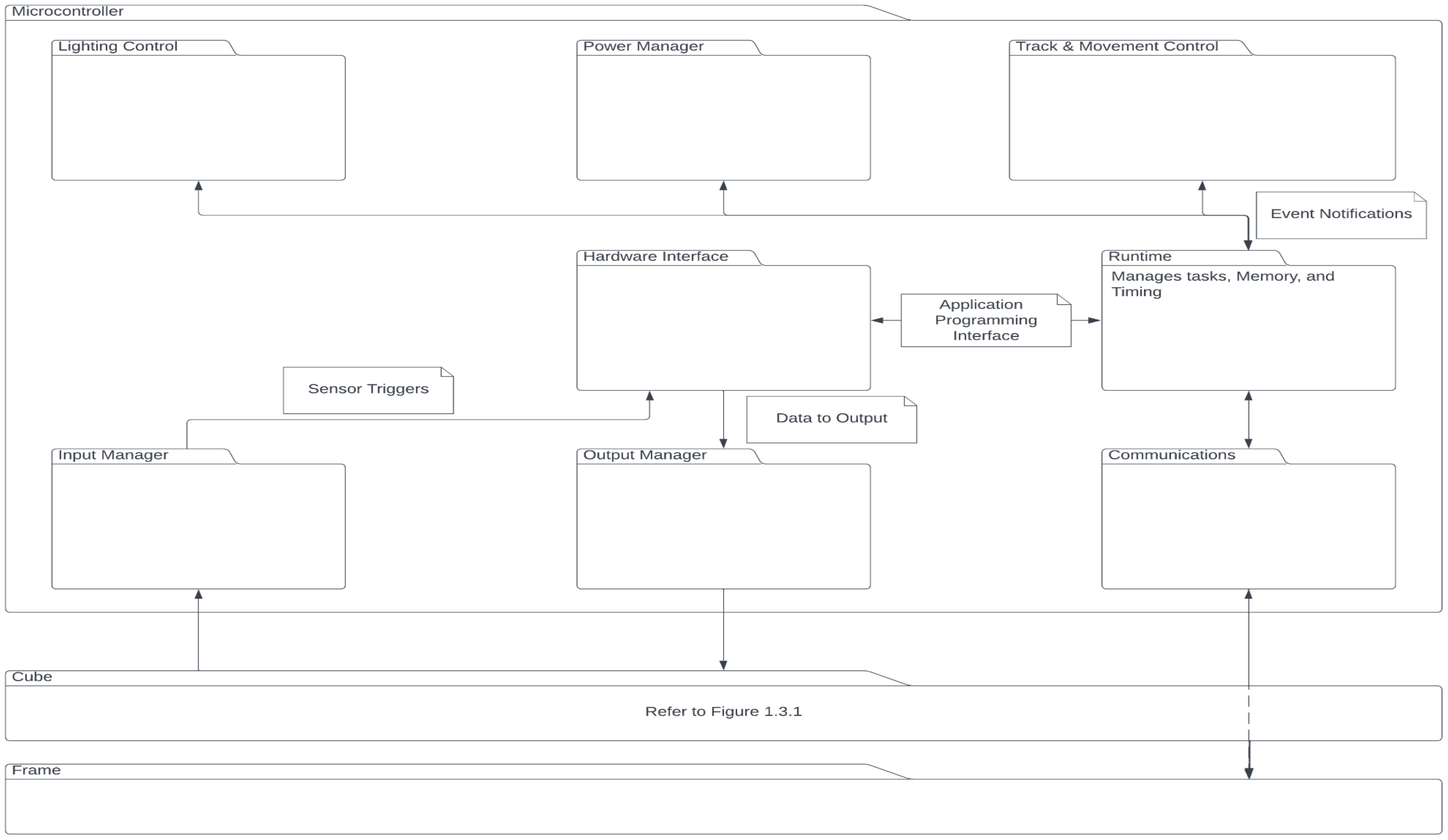
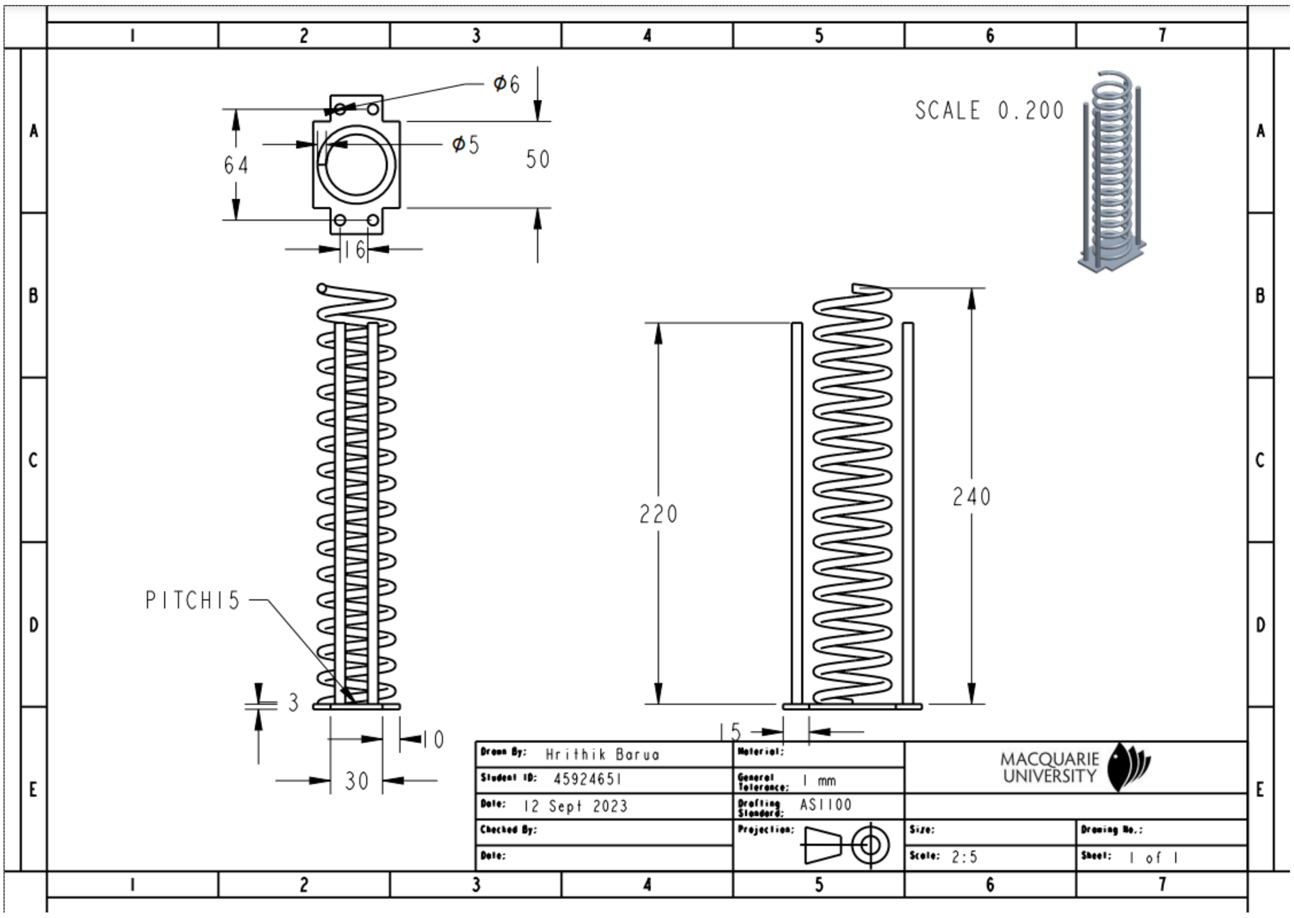


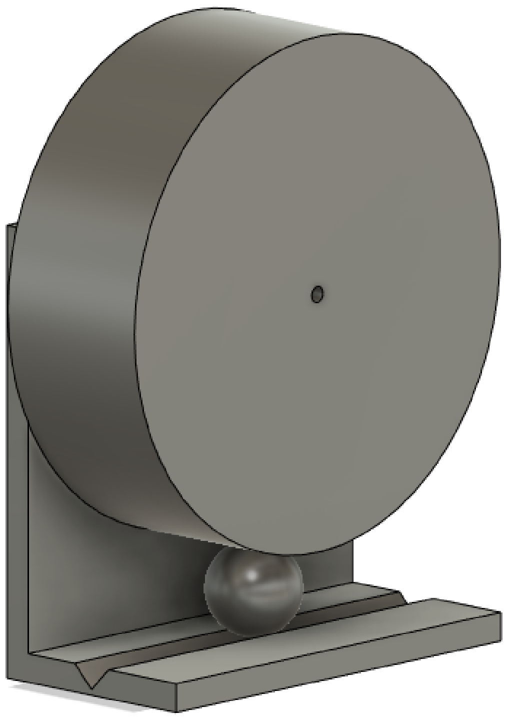
Figure 1.3.1 - Software Subsystems

A.4 Achmedies Spiral

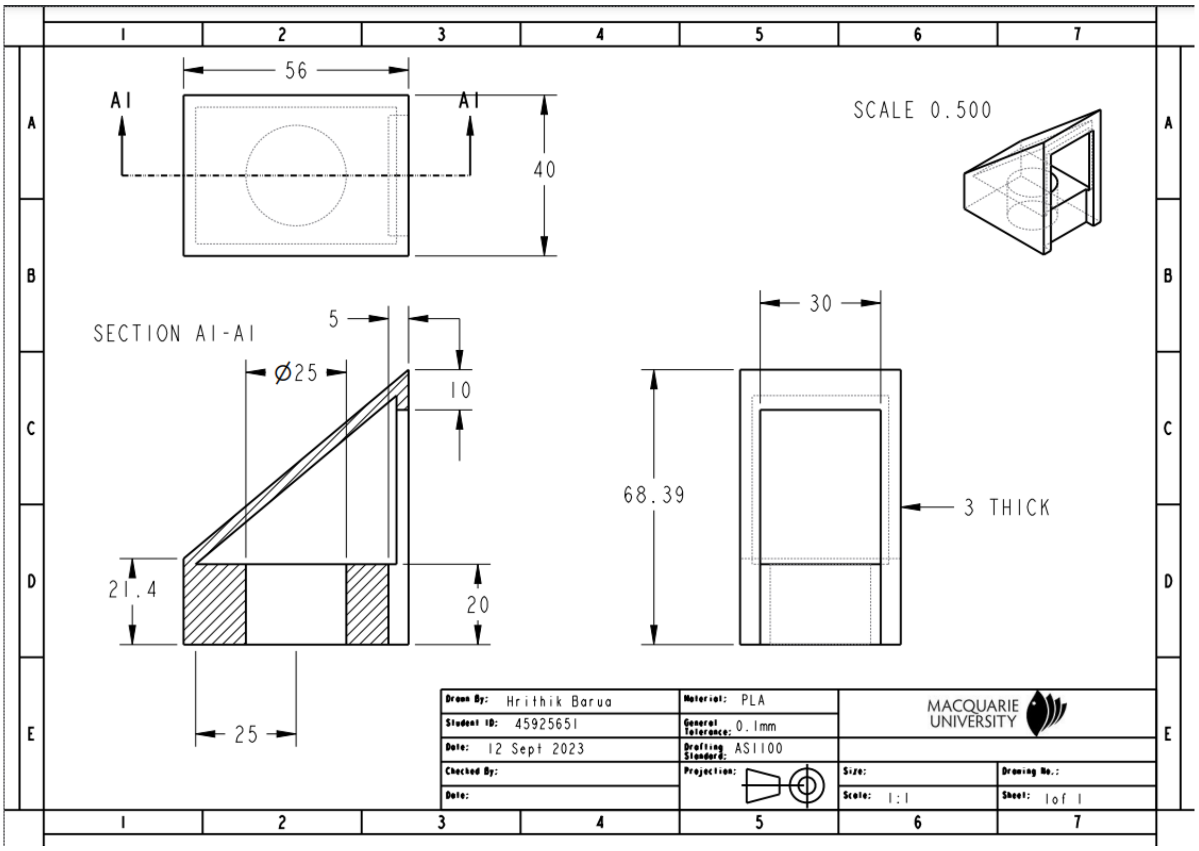


Drawn By:	Hrithik Barua	Material:			
Student ID:	45924651	General Tolerance:	1 mm		
Date:	12 Sept 2023	Drafting Standard:	AS1100		
Checked By:		Projection:		Size:	Drawing No.:
Date:				Scale: 2:5	Sheet: 1 of 1

A.5 Ejection Device



A. 5 Input Ramp Manufacturing Drawing



Appendix B - Project Design

B.1 DSM

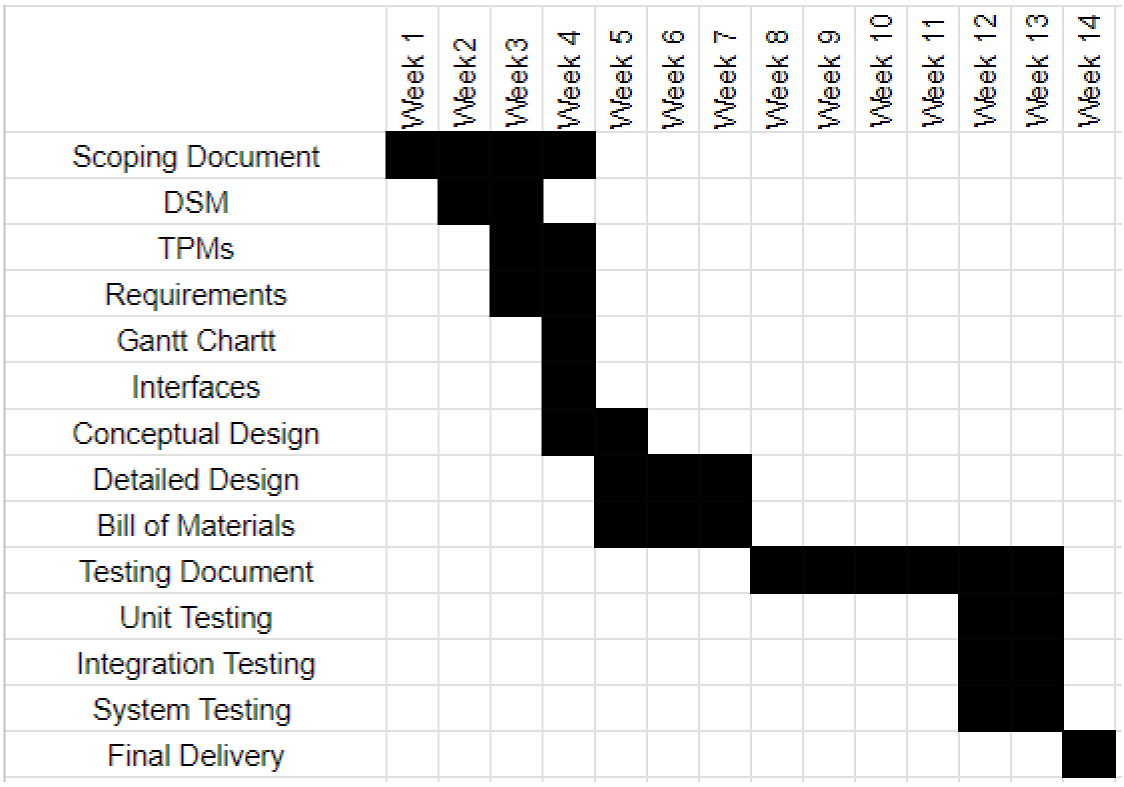
	Sensors	Wiring Sensors	Lights	Wiring Lights	Box Structure	Mounting for Sub-systems	Intake Mechanism	Raising Mechanism	Transfer Mechanism	Stopping Mechanism	Rotating Around Axis	Launching Mechanism	Controller	Motors	Wiring Motors	Static Lighting (RC3)	Triggers (sensors/latches)	Design (LED Display etc)	Dynamic Lighting (RC3)	Ball Control Flow (motors)
Sensors	■	x			x	x	x						x							
Wiring Sensors	x	■			x	x	x													
Lights	x		■	x	x	x	x						x			x	x	x	x	
Wiring Lights			x	■	x	x														
Box Structure					■															
Mounting for Sub-systems					x	■														
Intake Mechanism					x	x	■													
Raising Mechanism					x	x		■	x											
Transfer Mechanism					x	x			■											
Stopping Mechanism					x	x				■				x	x					x
Rotating Around Axis					x	x					■									
Launching Mechanism					x	x						■	x	x	x					
Controller					x	x							■							
Motors	x				x	x							x	■						
Wiring Motors	x				x	x							x	x	■					
Static Lighting (RC3)	x			x	x	x										■				
Triggers (sensors/latches)	x	x			x	x											■	x	x	
UI Design (LED Display etc)				x	x	x										x		■		
Dynamic Lighting (RC3)	x	x		x	x	x											x		■	
Ball Control Flow (motors)	x	x			x	x		x	x	x		x	x	x	x					■

Structure Team

Motions Team

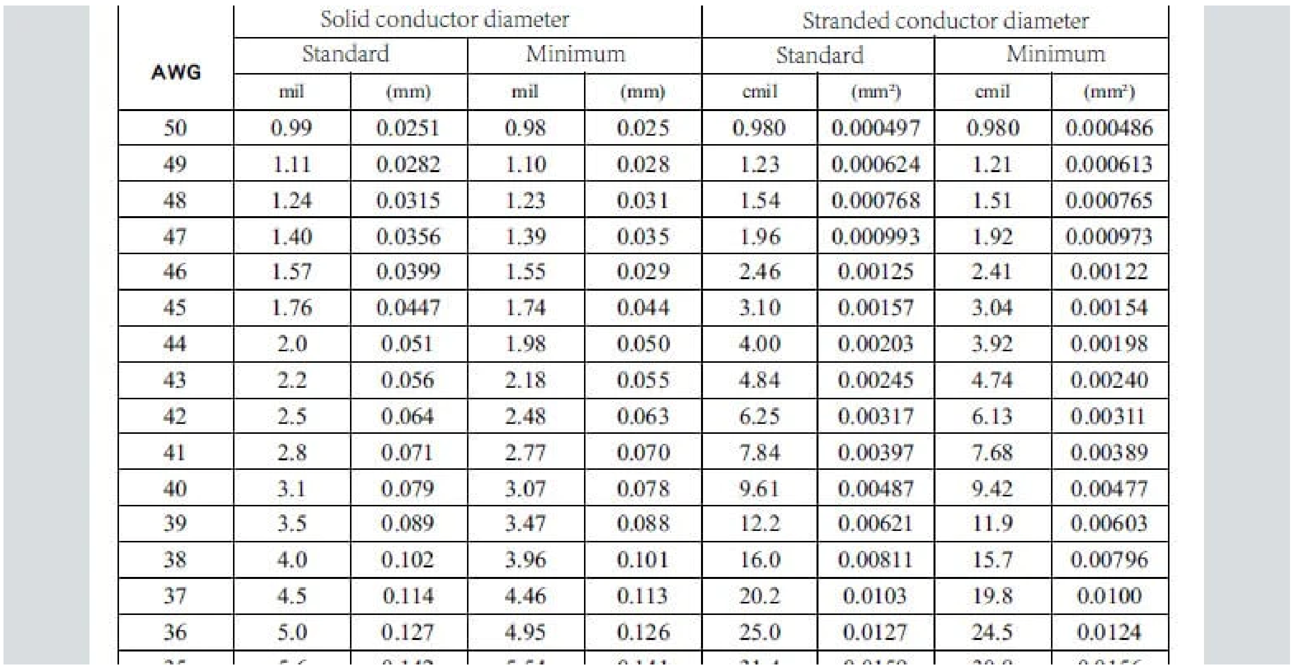
Comms Team

B.2 Gantt Chart



	Week 1	Week2	Week3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14
Scoping Document	■	■	■	■										
DSM		■	■											
TPMs			■	■										
Requirements			■	■										
Gantt Chartt				■										
Interfaces				■										
Conceptual Design				■	■									
Detailed Design					■	■	■							
Bill of Materials					■	■	■							
Testing Document								■	■	■	■	■	■	
Unit Testing												■	■	
Integration Testing												■	■	
System Testing												■	■	
Final Delivery														■

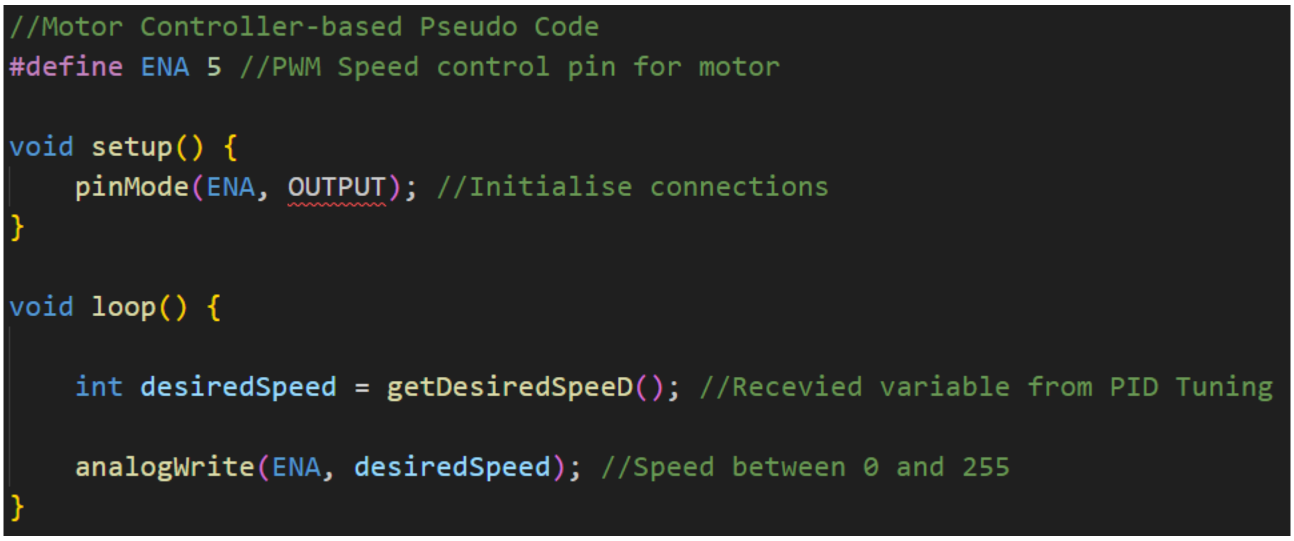
B.3 Wire Gauge Specification



AWG	Solid conductor diameter				Stranded conductor diameter			
	Standard		Minimum		Standard		Minimum	
	mil	(mm)	mil	(mm)	cmil	(mm²)	cmil	(mm²)
50	0.99	0.0251	0.98	0.025	0.980	0.000497	0.980	0.000486
49	1.11	0.0282	1.10	0.028	1.23	0.000624	1.21	0.000613
48	1.24	0.0315	1.23	0.031	1.54	0.000768	1.51	0.000765
47	1.40	0.0356	1.39	0.035	1.96	0.000993	1.92	0.000973
46	1.57	0.0399	1.55	0.029	2.46	0.00125	2.41	0.00122
45	1.76	0.0447	1.74	0.044	3.10	0.00157	3.04	0.00154
44	2.0	0.051	1.98	0.050	4.00	0.00203	3.92	0.00198
43	2.2	0.056	2.18	0.055	4.84	0.00245	4.74	0.00240
42	2.5	0.064	2.48	0.063	6.25	0.00317	6.13	0.00311
41	2.8	0.071	2.77	0.070	7.84	0.00397	7.68	0.00389
40	3.1	0.079	3.07	0.078	9.61	0.00487	9.42	0.00477
39	3.5	0.089	3.47	0.088	12.2	0.00621	11.9	0.00603
38	4.0	0.102	3.96	0.101	16.0	0.00811	15.7	0.00796
37	4.5	0.114	4.46	0.113	20.2	0.0103	19.8	0.0100
36	5.0	0.127	4.95	0.126	25.0	0.0127	24.5	0.0124

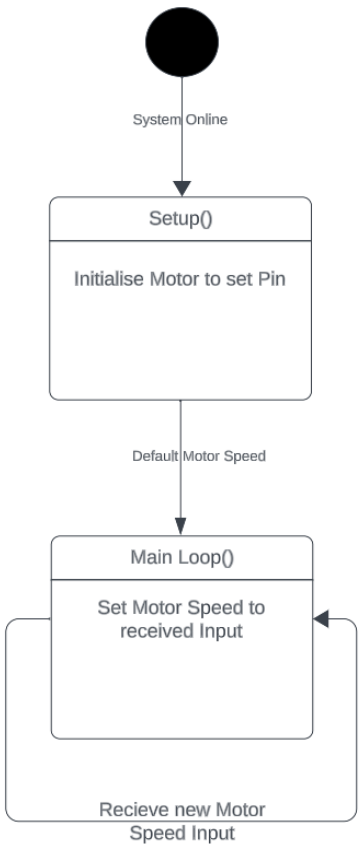
Appendix C - Code Design

C.1 Motor Driver Code

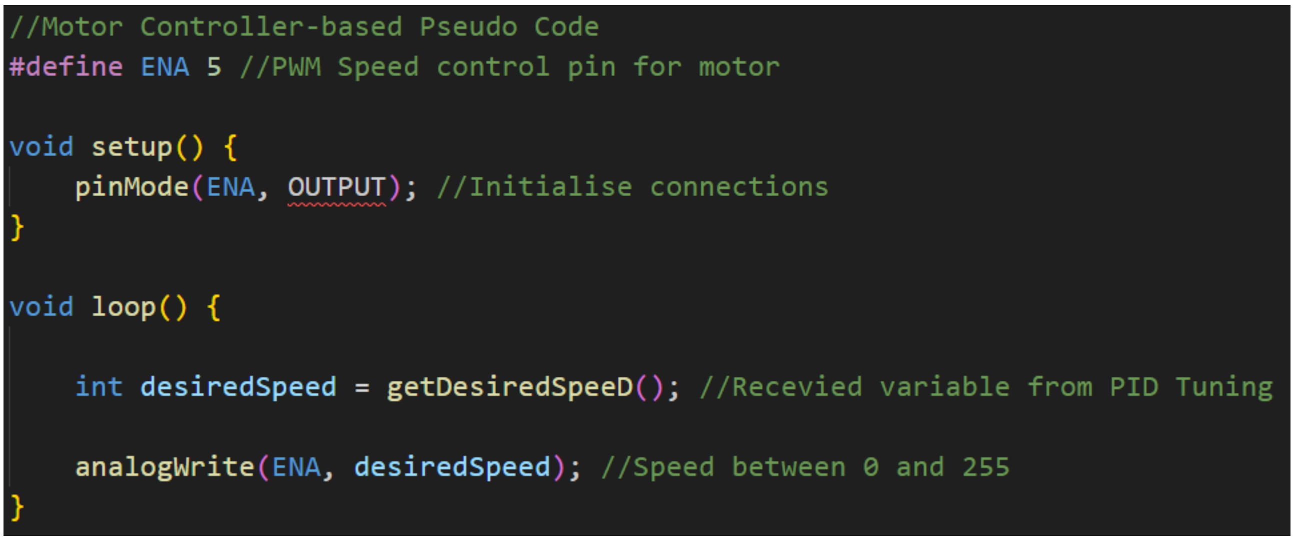


```
//Motor Controller-based Pseudo Code
#define ENA 5 //PWM Speed control pin for motor

void setup() {
    pinMode(ENA, OUTPUT); //Initialise connections
}

void loop() {

    int desiredSpeed = getDesiredSpeeD(); //Recevied variable from PID Tuning

    analogWrite(ENA, desiredSpeed); //Speed between 0 and 255
}
```

C.2 High-Level Motor Driver Implementation

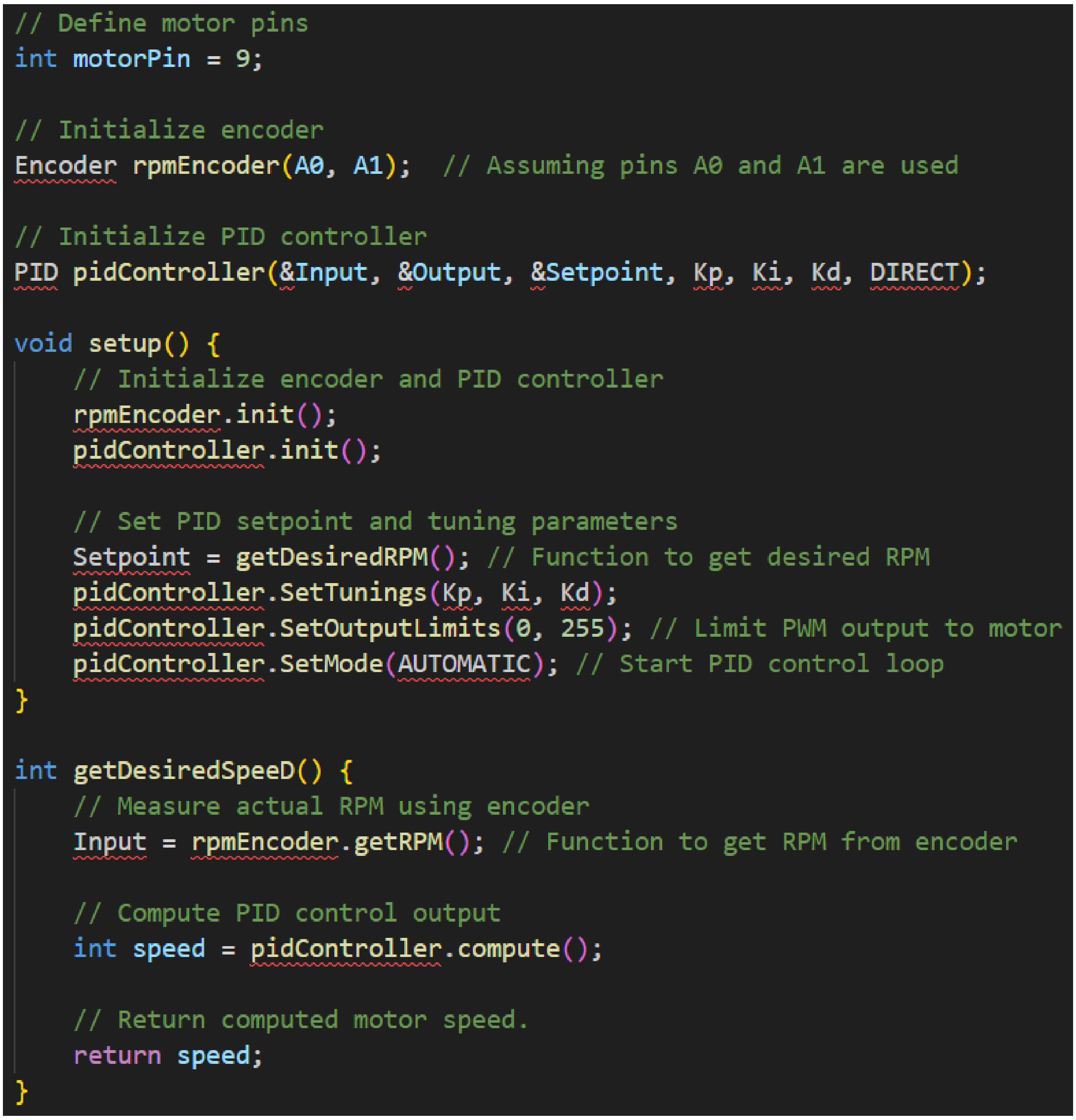

System Online
Setup()
Initialise Motor to set Pin
Default Motor Speed
Main Loop()
Set Motor Speed to received Input
Recieve new Motor Speed Input

C.3.1 Motor Driver setup



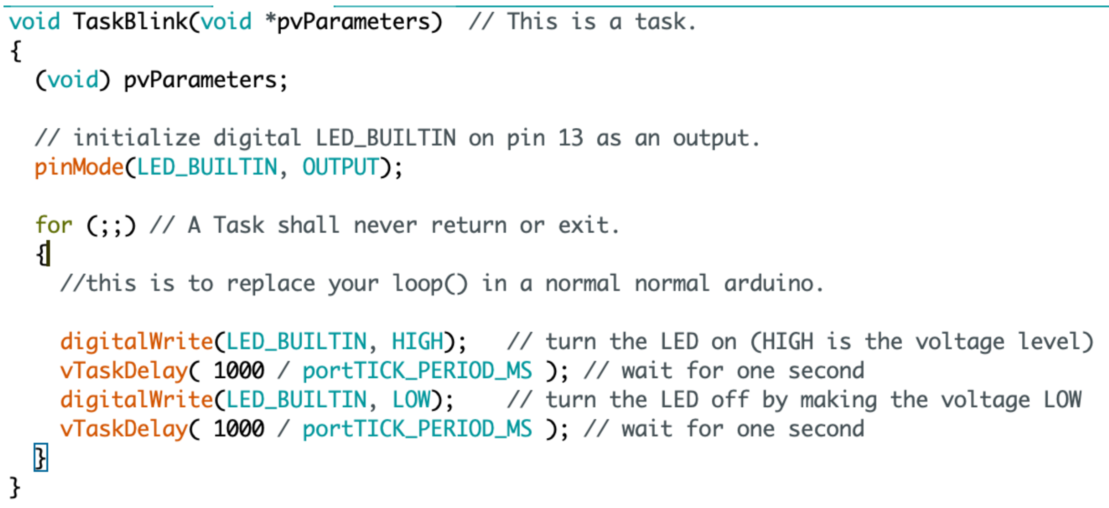
```
//Motor Controller-based Pseudo Code
#define ENA 5 //PWM Speed control pin for motor

void setup() {
    pinMode(ENA, OUTPUT); //Initialise connections
}

void loop() {

    int desiredSpeed = getDesiredSpeeD(); //Recevied variable from PID Tuning

    analogWrite(ENA, desiredSpeed); //Speed between 0 and 255
}
```

C.3.2 PID Tuning Setup



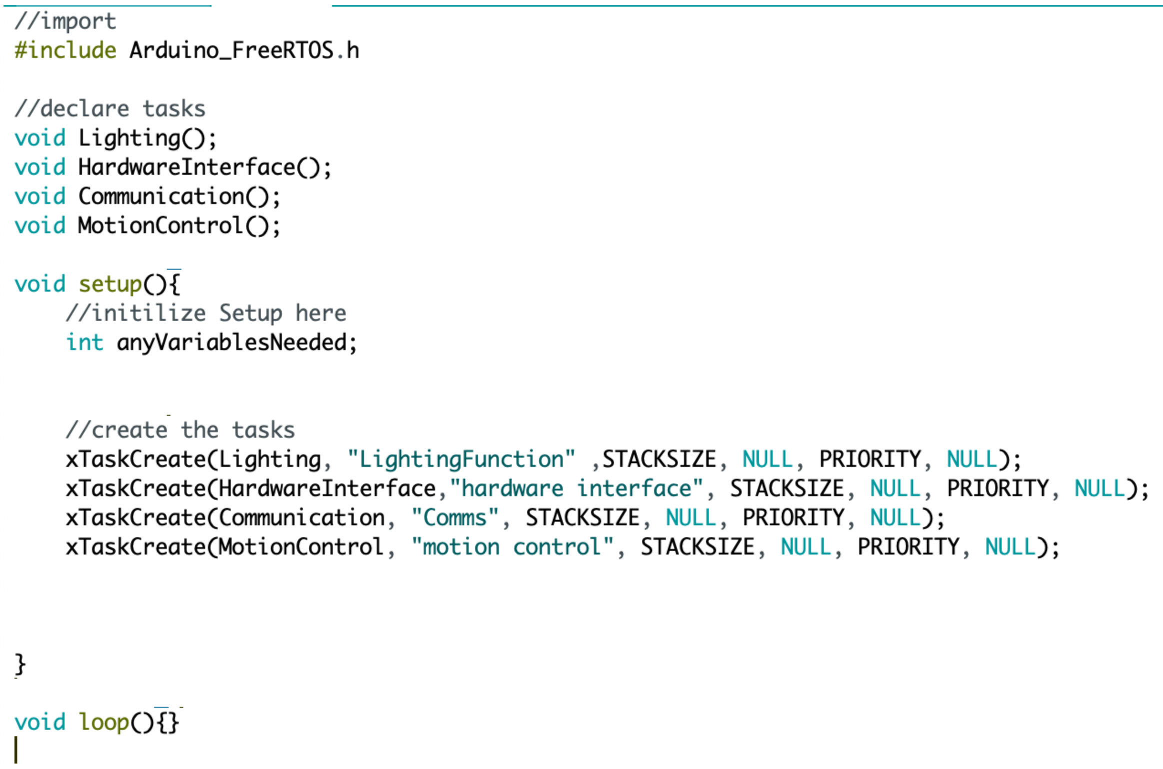
```
// Define motor pins
int motorPin = 9;

// Initialize encoder
Encoder rpmEncoder(A0, A1);  // Assuming pins A0 and A1 are used

// Initialize PID controller
PID pidController(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);

void setup() {
    // Initialize encoder and PID controller
    rpmEncoder.init();
    pidController.init();

    // Set PID setpoint and tuning parameters
    Setpoint = getDesiredRPM(); // Function to get desired RPM
    pidController.SetTunings(Kp, Ki, Kd);
    pidController.SetOutputLimits(0, 255); // Limit PWM output to motor
    pidController.SetMode(AUTOMATIC); // Start PID control loop
}

int getDesiredSpeeD() {
    // Measure actual RPM using encoder
    Input = rpmEncoder.getRPM(); // Function to get RPM from encoder

    // Compute PID control output
    int speed = pidController.compute();

    // Return computed motor speed.
    return speed;
}
```

C.4.1 Task setup



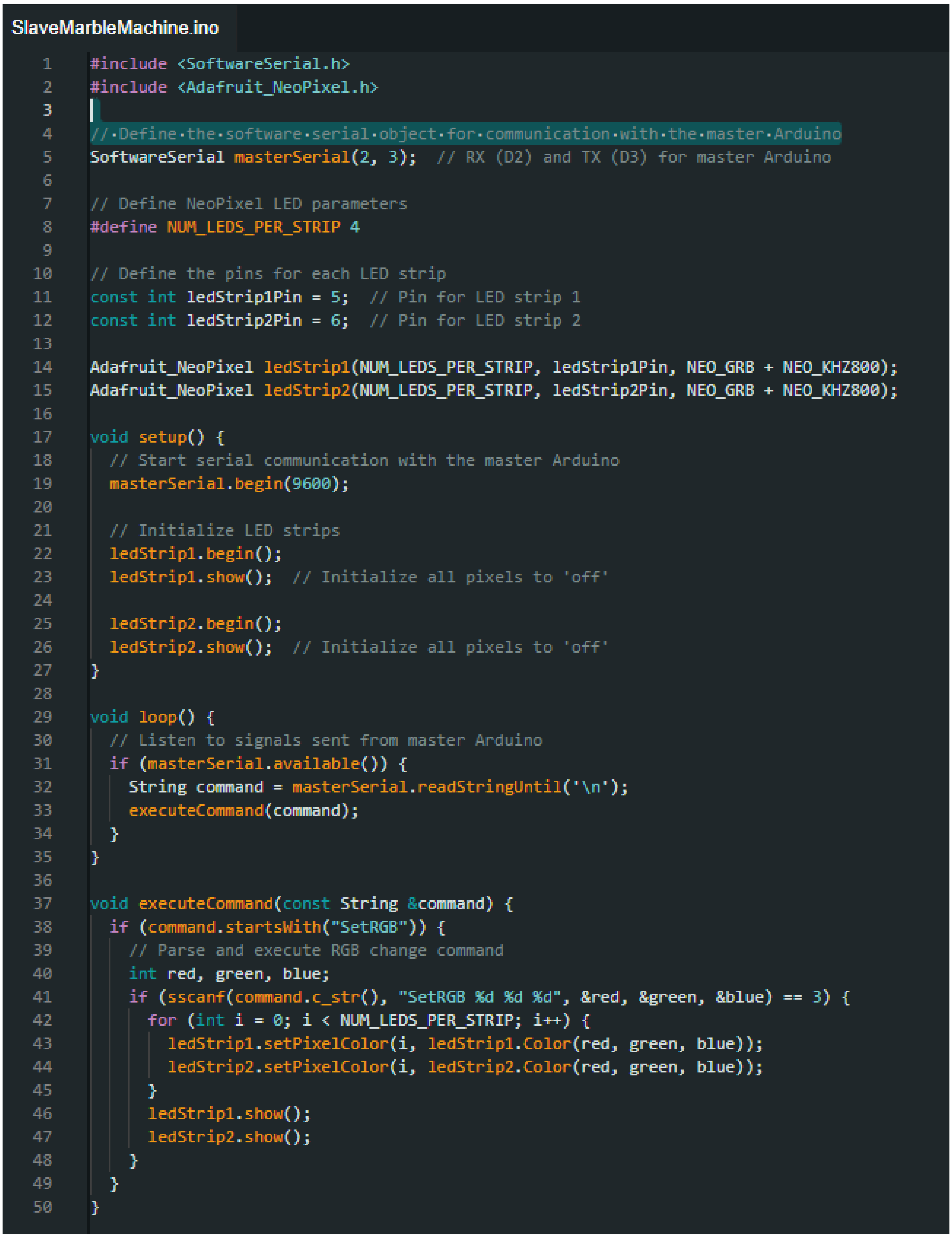
```
void TaskBlink(void *pvParameters)  // This is a task.
{
  (void) pvParameters;

  // initialize digital LED_BUILTIN on pin 13 as an output.
  pinMode(LED_BUILTIN, OUTPUT);

  for (;;) // A Task shall never return or exit.
  {
    //this is to replace your loop() in a normal normal arduino.

    digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
    vTaskDelay( 1000 / portTICK_PERIOD_MS ); // wait for one second
    digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
    vTaskDelay( 1000 / portTICK_PERIOD_MS ); // wait for one second
  }
}
```

C.4.2 FreeRTOS



```
//import
#include Arduino_FreeRTOS.h

//declare tasks
void Lighting();
void HardwareInterface();
void Communication();
void MotionControl();

void setup(){
    //initilize Setup here
    int anyVariablesNeeded;


    //create the tasks
    xTaskCreate(Lighting, "LightingFunction" ,STACKSIZE, NULL, PRIORITY, NULL);
    xTaskCreate(HardwareInterface,"hardware interface", STACKSIZE, NULL, PRIORITY, NULL);
    xTaskCreate(Communication, "Comms", STACKSIZE, NULL, PRIORITY, NULL);
    xTaskCreate(MotionControl, "motion control", STACKSIZE, NULL, PRIORITY, NULL);



}

void loop(){}
```

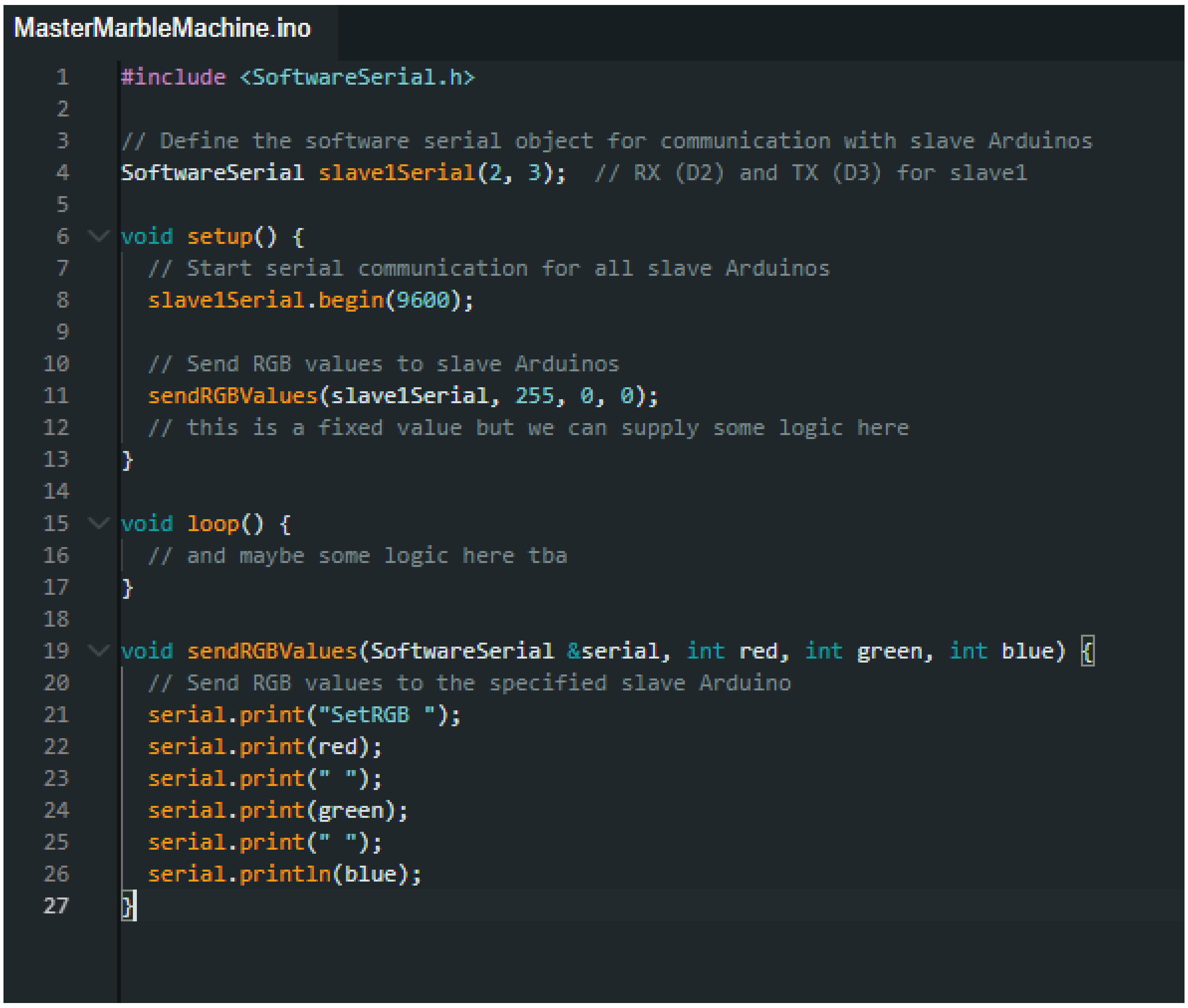
C.4.3 Marble Machine ‘Slave’ Communication and Lighting



SlaveMarbleMachine.ino

```
#include <SoftwareSerial.h>
#include <Adafruit_NeoPixel.h>

// Define the software serial object for communication with the master Arduino
SoftwareSerial masterSerial(2, 3);  // RX (D2) and TX (D3) for master Arduino

// Define NeoPixel LED parameters
#define NUM_LEDS_PER_STRIP 4

// Define the pins for each LED strip
const int ledStrip1Pin = 5;  // Pin for LED strip 1
const int ledStrip2Pin = 6;  // Pin for LED strip 2

Adafruit_NeoPixel ledStrip1(NUM_LEDS_PER_STRIP, ledStrip1Pin, NEO_GRB + NEO_KHZ800);
Adafruit_NeoPixel ledStrip2(NUM_LEDS_PER_STRIP, ledStrip2Pin, NEO_GRB + NEO_KHZ800);

void setup() {
  // Start serial communication with the master Arduino
  masterSerial.begin(9600);

  // Initialize LED strips
  ledStrip1.begin();
  ledStrip1.show();  // Initialize all pixels to 'off'

  ledStrip2.begin();
  ledStrip2.show();  // Initialize all pixels to 'off'
}

void loop() {
  // Listen to signals sent from master Arduino
  if (masterSerial.available()) {
    String command = masterSerial.readStringUntil('\n');
    executeCommand(command);
  }
}

void executeCommand(const String &command) {
  if (command.startsWith("SetRGB")) {
    // Parse and execute RGB change command
    int red, green, blue;
    if (sscanf(command.c_str(), "SetRGB %d %d %d", &red, &green, &blue) == 3) {
      for (int i = 0; i < NUM_LEDS_PER_STRIP; i++) {
        ledStrip1.setPixelColor(i, ledStrip1.Color(red, green, blue));
        ledStrip2.setPixelColor(i, ledStrip2.Color(red, green, blue));
      }
      ledStrip1.show();
      ledStrip2.show();
    }
  }
}
```

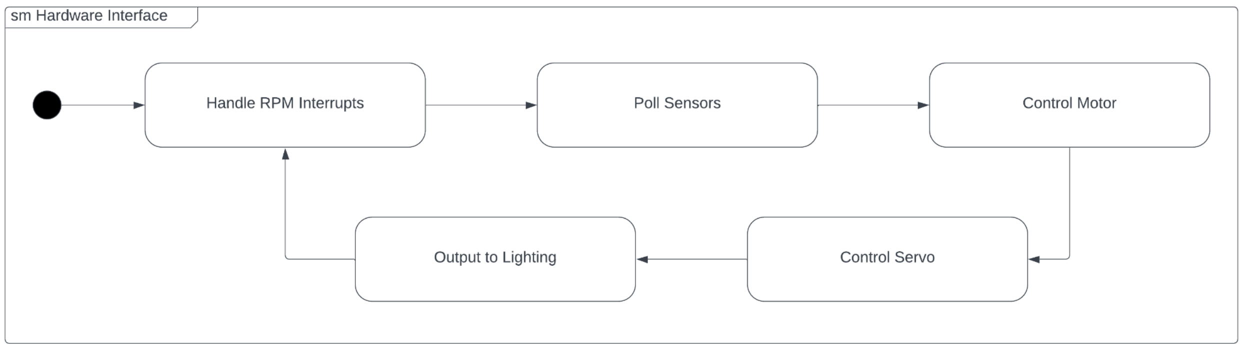
C.4.4 Marble Machine ‘Master’ Communication and Lighting



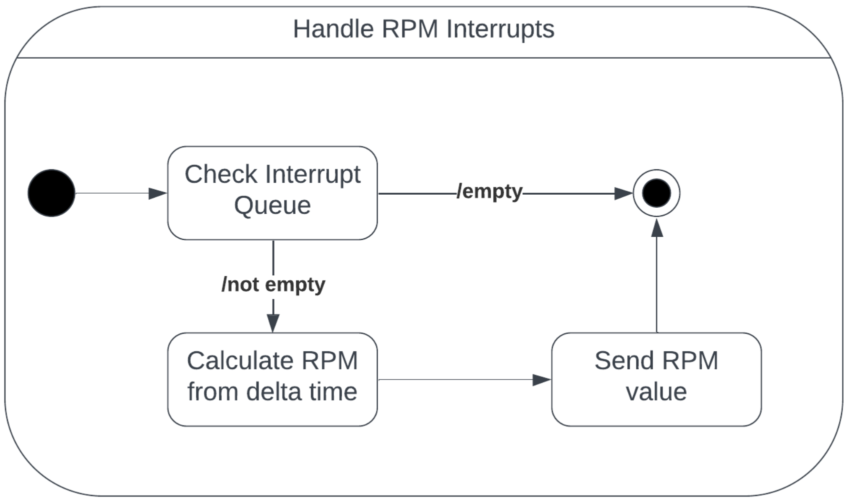
MasterMarbleMachine.ino

```
#include <SoftwareSerial.h>

// Define the software serial object for communication with slave Arduinos
SoftwareSerial slave1Serial(2, 3);  // RX (D2) and TX (D3) for slave1

void setup() {
  // Start serial communication for all slave Arduinos
  slave1Serial.begin(9600);

  // Send RGB values to slave Arduinos
  sendRGBValues(slave1Serial, 255, 0, 0);
  // this is a fixed value but we can supply some logic here
}

void loop() {
  // and maybe some logic here tba
}

void sendRGBValues(SoftwareSerial &serial, int red, int green, int blue) {
  // Send RGB values to the specified slave Arduino
  serial.print("SetRGB ");
  serial.print(red);
  serial.print(" ");
  serial.print(green);
  serial.print(" ");
  serial.println(blue);
}
```

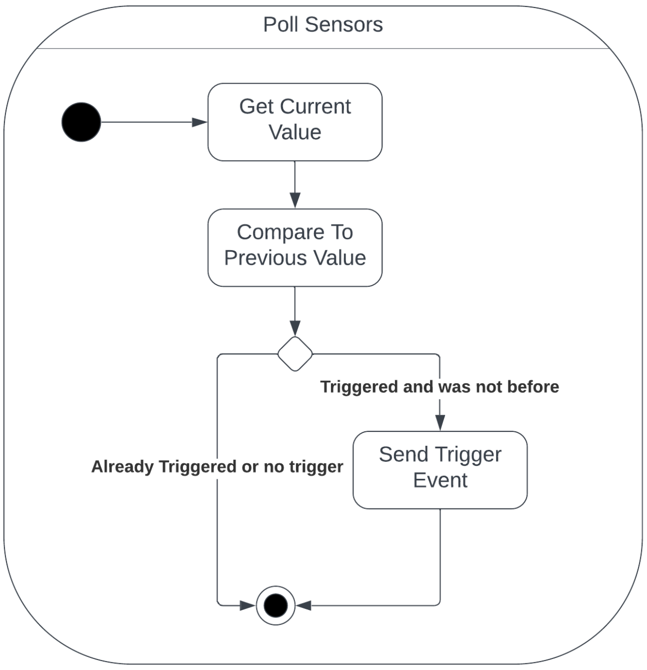
C.5.1 Hardware Interface - Functional Loop



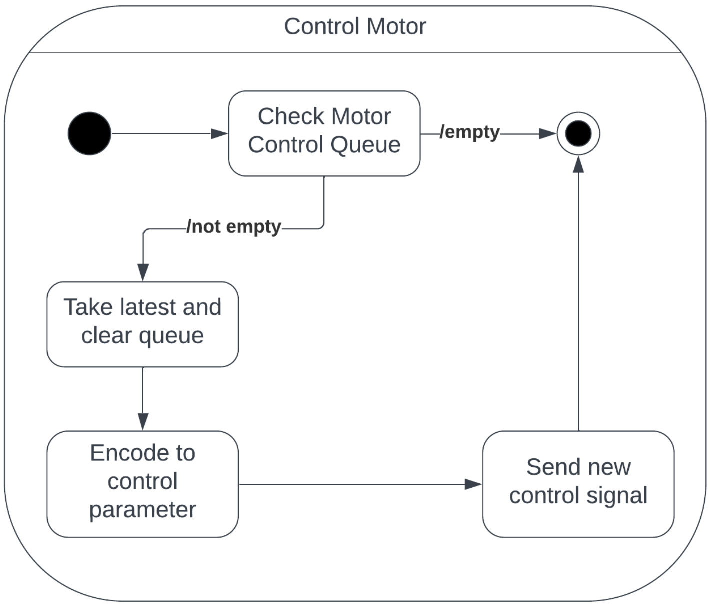
C.5.2 Encoder Interrupt Handling



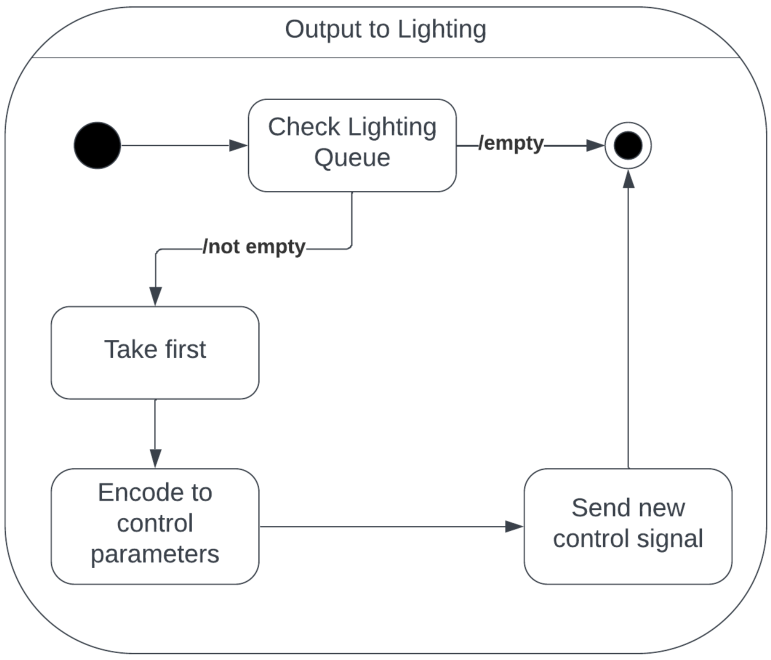
C.5.3 Sensor Polling


Poll Sensors
Get Current Value
Compare To Previous Value
Triggered and was not before
Send Trigger Event
Already Triggered or no trigger

C.5.4 Motor Control


Control Motor
Check Motor Control Queue
/empty
/not empty
Take latest and clear queue
Encode to control parameter
Send new control signal

C.5.5 Lighting Functional Flow


Output to Lighting
Check Lighting Queue
/empty
/not empty
Take first
Encode to control parameters
Send new control signal