

T1_COMMS_1 TESTING DOCUMENT



MACQUARIE
University

Team Name: T1_Comms_1

ENGG2000/ENGG3000

Date Submitted: 05/11/2023

PREFACE

The following engineering document was put together by members of the T1_Comms_1 group as a deliverable for the project that was given to them from their client. This document is intended to guide the clients and stakeholders through the thirteen weeks of the project's management and development.

This project that was given involved designing and assembling the "Massive Marvellous Multiple Marble Machine". In short, this machine was expected to be made up of many small boxes that guide numerous marbles throughout each of them while performing interesting actions on the marbles inside each box. This task was challenging and complex, requiring knowledge from multiple disciplines of engineering, which gave rise to the need to establish a multidisciplinary development team to successfully complete the project.

There were three smaller engineering groups inside the larger multidisciplinary team, where each team was responsible for their own clearly defined section of the designing and implementation of the project. One of the teams was responsible for the structure of the project, another for the motions aspect, and another for the communications component. The team "T1_Comms_1" was formed to plan, design, and implement the electronic sub-components of Box 1 of the machine. This included LEDs, servos, and other components requiring programming.

REVISION HISTORY

Version	Date	Week	Description
1.0	25/07/2023	1	Created the scoping document
1.1	25/07/2023	1	Set up document structure including headings and subheadings
1.2	25/07/2023	1	Completed 'Preface'
1.3	25/07/2023	1	Completed 'Contributing Members'
1.4	26/07/2023	1	Completed 'Purpose and Scope'
1.5	26/07/2023	1	Completed 'Problem Statement'
1.6	26/07/2023	1	Completed 'Deliverables'
1.7	01/08/2023	2	Completed 'Subsystems'
1.8	01/08/2023	2	Completed 'Assumptions'
1.9	01/08/2023	2	Completed 'Constraints'
1.10	01/08/2023	2	Completed 'Functional Requirements'
1.11	01/08/2023	2	Completed 'Performance Requirements'
1.12	08/08/2023	3	Completed 'Inclusions and Exclusions'
1.13	08/08/2023	3	Completed 'Interface Requirements'
1.14	08/08/2023	3	Completed 'Interface Sign-Off'
1.15	14/08/2023	3	Completed scoping document draft
1.16	20/08/2023	4	Completed final scoping document and submitted
1.17	28/08/2023	6	Began the design document
1.18	30/08/2023	6	Completed 'Conceptual Design - Code'
1.19	30/08/2023	6	Completed 'Detailed Design - Code'
1.20	01/09/2023	6	Completed 'Conceptual Design - Motors'
1.21	01/09/2023	6	Completed 'Programming Choices - Programming Language'
1.22	01/09/2023	6	Completed 'Programming Choices - Style of Code'

1.23	03/09/2023	6	Completed 'Conceptual Design - LEDs'
1.24	04/09/2023	7	Completed 'Detailed Design - LEDs'
1.25	08/09/2023	7	Completed 'Detailed Design - Motors'
1.26	11/09/2023	Break	Completed 'Conceptual Design - Sensors'
1.27	13/09/2023	Break	Completed 'Programming Libraries Used'
1.28	14/09/2023	Break	Completed 'Design Traceability'
1.29	15/09/2023	Break	Completed final Design Document and submitted
1.30	27/09/2023	8	Completed 'Project Schedule'
1.31	27/09/2023	8	Rewrote 'Detailed Design – Code' to reflect changes
1.32	30/09/2023	8	Completed 'Test Cases'
1.33	03/10/2023	9	Completed 'Verification of Requirements'
1.34	06/10/2023	9	Completed 'Resources and Equipment'
1.35	06/10/2023	9	Completed 'Types of Testing'
1.36	11/10/2023	10	Completed 'Testing Modes'
1.37	11/10/2023	10	Completed 'Testing Environments'
1.38	15/10/2023	10	Completed 'General Testing Procedure'
1.39	18/10/2023	11	Completed 'Specific Testing Procedure'
1.40	21/10/2023	11	Completed 'Review'
1.41	22/10/2023	11	Completed 'Further Improvements'
1.42	28/10/2023	12	Made various 'professionalism' changes
1.43	29/10/2023	12	Rewrote 'Problem Definition' according to feedback
1.44	29/10/2023	12	Rewrote 'Design Traceability' according to feedback
1.45	31/10/2023	13	Made various changes to the document structure according to feedback
1.46	02/10/2023	13	Rewrote 'Requirements' and 'Constraints' according to feedback
1.47	03/10/2023	13	Rewrote 'Conceptual Design' according to feedback
1.48	03/10/2023	13	Rewrote 'Detailed Design' according to feedback

1.49	05/11/2023	13	Completed the final Testing Document and submitted
------	------------	----	--

Scoping Document signed off by Ben Cavanagh

TABLE OF CONTENTS

PREFACE	2
REVISION HISTORY	3
LIST OF FIGURES	7
LIST OF TABLES.....	8
1.0 - INTRODUCTION AND OVERVIEW	9
1.1 - Purpose and Scope	9
1.2 - Glossary of Terms	10
1.3 - Contributing Members	11
1.4 - Problem Definition	11
1.4.1 Objective	11
1.4.2 Client Requirements.....	11
1.4.3 Challenges	12
1.4.4 Success Criteria	12
1.4.5 Scope.....	12
1.5 - Deliverables	12
1.6 - Subsystems.....	13
1.7 - Assumptions.....	14
1.8 - Constraints	15
2.0 - REQUIREMENTS	17
2.1 - Functional Requirements.....	17
2.2 - Performance Requirements.....	18
2.3 - Interface Requirements	19
2.4 - Requirements Sign-Off	20
3.0 - INCLUSIONS AND EXCLUSIONS.....	22

3.1 - Inclusions	22
3.2 - Exclusions	23
4.0 - CONCEPTUAL DESIGN	23
4.1 - Conceptual Design - Code	23
4.2 - Conceptual Design - LEDs	24
4.3 - Conceptual Design - Sensors	26
4.4 - Conceptual Design - Motors	27
5.1 - Detailed Design - Code	28
5.2 - Detailed Design - LEDs	29
5.3 - Detailed Design - Motors	30
5.4 - Programming Choices	31
5.4.1 - Programming Choices - Style of Code	31
5.4.2 - Programming Choices - Programming Language	31
5.5 - Programming Libraries Used	33
6.0 - DESIGN TRACEABILITY	34
7.0 - PROJECT SCHEDULE	38
8.0 - TESTING	38
8.1 - Technical Performance Measures (TPMs)	38
8.2 - Test Cases	41
8.3 - Verification of Requirements	43
8.4 - Testing Environments	45
8.5.1 - Digital	46
8.5.2 - Physical	46
8.6 - Testing Modes	46
8.7 - Types of Testing	47
8.8 - General Testing Procedure	48
8.9 - Specific Testing Procedures	49
9.0 - DESIGN REVIEW AND CONCLUSION	51
9.1 - Review	51
9.2 - Further Improvements	52
9.2.1 - Introduction	52
9.2.2 - Code Structure - Motor Class Improvements	52
9.2.3 - Code Structure - LED Implementation	52

9.2.4 - Time System	52
9.2.5 - Error Handling and Monitoring - Error Detection	53
9.2.6 - Error Handling and Monitoring - Real Time Monitoring	53
9.2.7 - Hardware Improvements - Power Efficiency	53
9.2.8 - Hardware Improvements - Additional Hardware	53
9.2.9 - Conclusion	53
REFERENCES	54
APPENDICES	55

LIST OF FIGURES

Figure 1 - Subsystems and Interconnections	55
Figure 2 - Marble Machine CAD model (by Joshua Ramos)	56
Figure 3 - Electronic Component Overview	56
Figure 4 - Team Gantt Chart	57
Figure 5 - House of Quality	58
Figure 6 - Conceptual Code Design Structure	59
Figure 7 - Conceptual Design Pseudocode	60
Figure 8 - Final Code Design Structure	61
Figure 9 - Position of the Staircase motor inside of the larger Staircase subsystem	61
Figure 10 - Position of the Ferris Wheel motor inside of the larger Ferris Wheel subsystem	62
Figure 11 - Sample Code for a LED Pattern	62
Figure 12 - a Class Used to Manage the States Across the Two Motors	63
Figure 13 - Final Code	63

LIST OF TABLES

Table 1 - Glossary of Terms.....	10
Table 2 - List of Team Members	11
Table 3 - List of Project Deliverables.....	12
Table 4 - Project Subsystems	13
Table 5 - List of Project Assumptions	14
Table 6 - List of Project Constraints	15
Table 7 - List of Project Functional Requirements	17
Table 8 - List of Project Performance Requirements	18
Table 9 - List of Project Interface Requirements.....	19
Table 10 - Sign-Off for Interface Requirements.....	20
Table 11 - List of Inclusions.....	22
Table 12 - List of Exclusions	23
Table 13 - Table of Algorithm Functions.....	28
Table 14 - Design Traceability Table.....	34
Table 15 - Project Document Schedule.....	38
Table 16 - Technical Performance Measures.....	38
Table 17 - Test Cases.....	41
Table 18 - Verification of Requirements Table.....	43
Table 19 - Comms Team DSM.....	55

1.0 - INTRODUCTION AND OVERVIEW

1.1 - Purpose and Scope

The purpose of the Scoping and Requirements Document is to meet all the following criteria:

- Define the engineering problem for the project.
- Describe all the deliverables that the client will receive upon completion of the project.
- Define all the subsystems of the project, including those from each team (Comms, Motions and Structures) so that the connections between each are considered and planned.
- Describe all assumptions made prior to the construction of the project.
- Describe all constraints in relation to the project.
- Describe all the functional, performance, and interface requirements for the project.
- Describe all the inclusions and exclusions in relation to each of the subsystems of the Comms team.

1.2 - Glossary of Terms

Table 1 - Glossary of Terms

Term	Definition
Arduino Uno	The Arduino Uno is an open-source microcontroller board designed by Arduino.cc that is based on the Microchip ATmega328P microprocessor.
Assumptions	Anything that is assumed not directly stated.
Comms Team	Engineers responsible for the functionality of all the software and correctness of technological components of the project.
Constraints	Internal and external factors imposed on the project.
Deliverables	The result of the project will be delivered to the client.
Functional Requirement	A requirement which relates directly to functional elements.
Interface Requirement	A set of criteria which explain the requirements for common functional or physical barriers between two systems.
Motions Team	Engineers responsible for all moving parts of the project, especially how marbles are guided through the box and any electrical components.
Performance Requirement	A set of criteria which stipulate how things should perform.
Structures Team	Engineers responsible for the structure of the project, specifically with relation to the materials used and how the parts are assembled together.
Subsystem	A self-contained system within a larger system.

1.3 - Contributing Members

The following is a list of all members in T1_Comms_1.

Table 2 - List of Team Members

Team Member	Cohort	Student ID	Role	Contribution
Ben Cavanagh	ENGG3000	46975373	Team Leader	14.29%
Quoc Huy Pham	ENGG3000	46294175	Team Leader	14.29%
Vikil Chandrapati	ENGG2000	47189320	Member	14.29%
Eugene Tshien	ENGG2000	47407417	Member	14.29%
Thomas Mikic	ENGG2000	47287632	Member	14.29%
Irvin Hansen	ENGG2000	46990704	Member	14.29%
Rumeth Herath	ENGG3000	47087854	Member	14.29%
TOTAL				100%

1.4 - Problem Definition

1.4.1 Objective

Our objective, T1_Comms_1 is to engineer electronic/software components within the box to meet client requirements and ensure seamless integration with the "Massive Marvellous Multiple Marble Machine.".

1.4.2 Client Requirements

- The box **MUST** provide some visual indication using LEDs or other lighting systems approved by the customer as to the function being performed. (RC3).
- The box **MUST** access external services, such as power or telecommunications, only through the service port shown. (RC4).
- The box should have a remote control within the machine. (RC23).

1.4.3 Challenges

These challenges include designing an efficient and reliable control system, seamlessly integrating electronic components, and ensuring compatibility with the work of other teams contributing to the overall functionality of the machine.

1.4.4 Success Criteria

The successful execution of T1_Comms_1's engineering tasks involves achieving all client requirements. This will result in the box performing its unique marble manipulation functions while ensuring smooth communication (e.g., LEDs) with the remaining components of the machine.

1.4.5 Scope

This document primarily revolves around the engineering problems in T1_Comms_1. Our primary focus lies in programming the electronic components within the box and to ensure all components compatibility with each other to meet client requirements and integrate seamlessly within the larger machine.

1.5 - Deliverables

The following table presents a list of every item that the client will be given prior to or upon the completion of the project.

Table 3 - List of Project Deliverables

ID	Item	Description of Item	Due Date
T1_C1_D01	Scoping and Requirements Document	A document containing project requirements and the scope (what is included and not included).	20/08/2023
T1_C1_D02	Design Document	A document explaining the initial and final designs of Box 1, including the decisions and trade-offs made.	10/09/2023
T1_C1_D03	Testing Document	A document containing detailed testing procedures to be conducted on Box 1, including all information from previous documents.	29/10/2023
T1_C1_D04	Statement of Work	A document explaining the changes to the project document that were made after receiving feedback.	29/10/2023
T1_C1_D05	Project Presentation	A video presentation of Box 1 successfully working as intended.	07/11/2023

T1_C1_D06	Arduino Code for Box 1	The written code allowing the electronic components to operate in Box 1.	07/11/2023
T1_C1_D07	Student Logbooks	Every team member's logbook with entries from the beginning of the project to its end.	07/11/2023
T1_C1_D08	Box One	The first box to be given to the client to become part of the Massive Marvellous Multiple Marble Machine.	07/11/2023

1.6 - Subsystems

Box One was required to effectively use several subsystems to fulfil the requirements. A diagram showing the connections between the different subsystems is shown in *Figure 1*. While certain groups were responsible for their own components, some groups were required to share the responsibility of certain subsystems. Each subsystem as well as the division of responsibility is described below:

Table 4 - Project Subsystems

ID	Subsystem Name	Description	Team
T1_Subsystem_A	Code	C++ code that is used to control Arduino Uno components, including servos and LED Strips.	Comms
T1_Subsystem_B	Servos	An electronic device that rotates to a degree specified by code.	Comms, Motions
T1_Subsystem_C	LED Strips	A flexible strip of small LEDs.	Comms
T1_Subsystem_D	Staircase	A rotating staircase designed to take marbles in an upwards direction inside the box.	Motions, Structs
T1_Subsystem_E	Marble Tracks	Regular tracks designed to control the movement of marbles to a desired location inside the box.	Structs
T1_Subsystem_F	Ferris Wheel	A system designed to rotate and move marbles in an upwards direction inside the box.	Motions, Structs
T1_Subsystem_G	Marble Queue	A system designed to hold marbles stationary while they wait before moving to the staircase section of the box.	Structs

1.7 - Assumptions

The following table presents a list of assumptions for the project that were made prior to its commencement.

Table 5 - List of Project Assumptions

ID	Assumption
T1_C1_A01	The performance of other boxes in the machine will not affect the performance of Box 1.
T1_C1_A02	The requirements provided from the client do not change through the construction of the box.
T1_C1_A03	Members of all teams (structures, motions, comms) have sufficient technical knowledge and skills to successfully construct the box.
T1_C1_A04	Members of all teams have sufficient teamwork skills to successfully construct the box.
T1_C1_A05	The box will be housed in such a way that the environment does not significantly affect the box.
T1_C1_A06	The box has sufficient structural integrity.
T1_C1_A07	The electronic components used are compatible with an Arduino Uno.
T1_C1_A08	All physical components to be implemented are fully functional with no defects.
T1_C1_A09	Power will be provided at any time marbles are passing through the box.
T1_C1_A10	All Comms components (LED, sensors, motors, etc.) can be successfully controlled by the Arduino software.

1.8 - Constraints

The following table presents a list of constraints for the project.

Table 6 - List of Project Constraints

ID	Type	Constraint
T1_C1_C01	Skills and knowledge	The quality of work that can be carried out to complete the project is limited to the current skills and knowledge of everyone in the team.
T1_C1_C02	Programming language	The Arduino Uno can only accept C++ as a programming language.
T1_C1_C03	Timeframe	The amount of time allowed to complete the project is limited to only thirteen weeks, starting from 25/07/2023 (plus two weeks for the mid-semester break).
T1_C1_C04	Suppliers	Any component used inside the box can only be purchased from the following five suppliers: Bunnings, RS Online, Core Electronics, Hobby King, and Jaycar.
T1_C1_C05	Modularity	The final box must not interfere with the performance of modules surrounding it and will seamlessly integrate with other systems.
T1_C1_C06	Budget	The total cost of all components cannot exceed \$100 AUD (Excluding the Arduino).
T1_C1_C07	Memory Limit	The binary sketch size must be below 32Kb to be able to be supported on the Arduino Uno.
T1_C1_C08	Size	All components must be able to fit inside the box.
T1_C1_C09	Dimensions	The external dimensions of the box must be 250mm x 250mm x 250mm.
T1_C1_C10	Mounting	Each cube must use only the four mounting points to attach to the sculpture (as described by the client).
T1_C1_C11	Visual Indicator	The box must provide some form of visual indicator, such as LEDs or another lighting system permitted by the customer.
T1_C1_C12	Communication with components	The box must only access external services, such as power or telecommunications, via the service port.
T1_C1_C13	Location on sculpture	Each cube location on the backing board must be in the centre of a 265mm horizontally by 265mm vertically square.

T1_C1_C14	Marble	Each cube must be built to take marbles with a diameter of 16mm made of solid steel.
T1_C1_C15	Marble speed	Each cube must receive marbles no more than 1 second apart through the entrance port.
T1_C1_C16	Cube materials	Cubes must be made of 3mm MDF for the back and side parts, with the addition that the front surface must be transparent.
T1_C1_C17	Maintainability	The cube's front surface must be easily detachable for maintenance needs.
T1_C1_C18	Mechanical limitations	Gravity and electrical power are the two forces available to a cube. A cube must not use any liquids or compressed air or other gases in its operation.
T1_C1_C19	Marble entrance	Each cube must receive a marble through an input aperture on a surface other than the cube's top surface. The customer must agree to this location.
T1_C1_C20	Marble exit	Each cube must expel all marbles through an exit aperture on a surface other than the one where the marble entered.
T1_C1_C21	Marble expelling rate	Cubes must not expel marbles at a rate that is much faster than that attributed to gravity alone.
T1_C1_C22	Marble exit hole position	Each cube must direct the marble so that the bottom of the exit hole is at least 100 mm higher than the top of the input opening.
T1_C1_C23	Marble positioning	The marble is held immobile in all physical dimensions for at least 0.5 seconds.

2.0 - REQUIREMENTS

The following requirements will be only relevant to the Comms team and does not include requirements that the Structures and Motions team are responsible for.

2.1 - Functional Requirements

The following table presents a list of functional unit and subsystem requirements for the project.

Table 7 - List of Project Functional Requirements

ID	Functional Requirement	Description
T1_C1_FR01	Cube functionality	The cube must be able to utilise mechanisms to interact with the marble kinetically.
T1_C1_FR02	Marble entry and exit	The cube must allow marbles to enter and exit with the interval of 1 second.
T1_C1_FR04	Marble movement	Marbles moving through the box must be able to stop moving in all physical dimensions for a short period of time.
T1_C1_FR05	Mechanical controller	Arduino Uno will be utilised to control all mechanical, sensors and lighting components inside the cube. In this sense, the uploaded code must control the lighting, Ferris wheel and the staircase such that they work appropriately.
T1_C1_FR06	Lighting	The cube will have LED strips that display patterns. In particular, the strip will turn on when the box is running, displaying patterns, and behaving appropriately.
T1_C1_FR07	Ferris wheel	The Ferris wheel will be operated by a dc motor that turns a gear to spin the wheel at the right torque and speed. This wheel will spin continuously, having marbles feed in when a slot is available. The marble will then spin around to the top, where a stopper will force it to roll out.
T1_C1_FR08	Staircase	The cube will have a Staircase operated by a DC motor to elevate the marbles. A circular wheel will spin in an orbit such that it pushes the staircase up and down in a vertical manner.

2.2 - Performance Requirements

The following table presents a list of non-functional (performance) requirements for the project, considering both individual units and subsystems.

Table 8 - List of Project Performance Requirements

ID	Performance Requirement	Description
T1_C1_PR01	Safety	No electronic component should be able to injure or harm any person observing it.
T1_C1_PR02	Latency	Latency between all components must be minimal.
T1_C1_PR03	Brightness	The LEDs should shine brightly.
T1_C1_PR04	Accuracy / Correctness	The LEDs should shine in patterns as expected from the written code.
T1_C1_PR05	Efficiency/Time Complexity	The time complexity must be efficient to ensure high speed, smoothness, and minimal delay.
T1_C1_PR06	Memory Usage	Memory must be efficiently used in the code.
T1_C1_PR07	Maintainability	Code must be modular in design, being able to easily, add, edit, and remove parts.
T1_C1_PR08	Flexibility	Being able to modify existing code easily to work in many scenarios without influencing other code segments.
T1_C1_PR09	Robustness	The code will work through repeated uses without bugs or issues throughout the lifecycle.
T1_C1_PR10	Ferris Wheel speed	The default spinning speed for the Ferris wheel that should keep marbles travel smoothly inside the machine.
T1_C1_PR11	Staircase oscillator speed	The default oscillation for the staircase that should move the marbles upward.

2.3 - Interface Requirements

The following table presents a list of interface requirements for the project.

Table 9 - List of Project Interface Requirements

ID	Interface Requirement	Description
T1_C1_IR01	Placement of components	The placement of components inside the box must not affect the movement of marbles in any way that forces it off track.
T1_C1_IR02	Cable Routing	Any wires and cables need to be arranged in a way as not to affect the movement of the marbles.
T1_C1_IR04	Ferris wheel DC motor	The Arduino shall be connected to a DC motor that rotates a Ferris wheel. Rotation speed is based on T1_C1_PR10.
T1_C1_IR05	Staircase DC motor	The Arduino shall be connected to a second DC motor that rotates the Staircase. Rotation speed is based on T1_C1_PR11.
T1_C1_IR06	LED strips	The Arduino shall be connected to 2 light strips. The first light strip is fixated around the ferris wheel and set with a looping rainbow colour. The second light strip is set up alongside the main ramp and lights up if the sensor detects marbles on the track.

2.4 - Requirements Sign-Off

The following table presents the requirements sign-offs for the construction of Box 1 between each team (Structures, Motions, and Comms). The sign-offs were completed on the 18th of August 2023.

Table 10 - Sign-Off for Interface Requirements

Combined Interface Sign-off				
Group	Requirement Number	Requirement Description	Team Leader Sign-off	Date
Structures 3	RC1	The cube MUST have the external dimensions of 250 mm by 250mm by 250mm.	Justin Mallouk	18-08-2023
Structures 3	RC2	The cube MUST mount to the sculpture using only the four mounting positions specified by the customer.	Joshua Ramos	18-08-2023
Comms 1	RC3	The cube MUST provide some visual indication using LEDs or other lighting system approved by the customer as to the function being performed.	Quoc Huy Pham	18-08-2023
Comms 1	RC4	The cube MUST access external services, such as power or telecommunications, only through the service port.	Ben Cavanagh	18-08-2023
Comms 1	<i>Additional Requirement</i>	The cube MUST be able to communicate with other cubes to perform actions.	Ben Cavanagh	18-08-2023
Structures 3	RC5	The cube's 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.	Justin Mallouk	18-08-2023

Structures 3	RC6	The cube MUST be designed to accept marbles that are a solid steel sphere with a diameter of 16mm. The cube SHOULD also accept marbles of the same diameter that are made of lighter materials. It is RECOMMENDED that lighter marbles still pass through the cube. (AND INTERNALLY).	Justin Mallouk	18-08-2023
Motions 6	RC7	The cube MUST accept marbles that are spaced no closer than 1 second apart through the opening port.	Neil Quisumbing	18-08-2023
Motions 6	RC8	The cube MUST be manufactured from 3mm MDF for the back and side pieces, with the additional requirement that the front surface MUST be transparent.	Neil Quisumbing	18-08-2023
Structures 3	RC9	The front surface MUST NOT be structural and MUST be easily removable for maintenance purposes for the cube.	Joshua Ramos	18-08-2023
Motions 6	RC10	The only forces provided to the cube are gravity and electrical power. The cube MUST NOT use any liquid in its operation and MUST NOT use compressed air or other gases.	Ivan Schipper	18-08-2023
All Groups	RC11	The cube MUST have a bill of materials, and each bill of materials MUST total to less than AUD\$100.00.	Neil Quisumbing	18-08-2023
Motions 6	RC20	The cube MUST accept a marble through an input opening on a surface other than the top surface of the cube. This location must be agreed to by the customer.	Neil Quisumbing	18-08-2023
Motions 6	RC21	The cube MUST expel all marbles out through an exit opening on a different surface to that where the marble entered.	Ivan Schipper	18-08-2023
Motions 6	RC22	The cube SHALL NOT deliberately expel marbles at a speed	Ivan Schipper	18-08-2023

		significantly higher than that attributable to gravity alone.		
Motions 6	RC23	The cube MUST control the marble in such a fashion that the bottom of the exit opening is at least 100 mm higher than the top of the input opening.	Ivan Schipper	18-08-2023
Motions 6	RC24	The cube MUST control the marble in such a fashion that satisfies ONE of the LEVEL conditions LC1 to LC3.	Neil Quisumbing	18-08-2023

3.0 - INCLUSIONS AND EXCLUSIONS

3.1 - Inclusions

The following table presents a list of inclusions for the project.

Table 11 - List of Inclusions

ID	Inclusion	Description
T1_C1_IN01	Scoping and Requirements Document	The project document that contains the requirements and scope of the project (the inclusions and exclusions).
T1_C1_IN02	Design Document	The project document that contains extensive information in relation to the conceptual design and detailed design of the project.
T1_C1_IN03	Testing Document	The project document that details specific testing procedures to test the Box One.
T1_C1_IN04	Statement of Work	The project document that contains an overview of all changes made to documentation after receiving feedback.
T1_C1_IN05	C++ Code	The code to control the electronic components inside the box.
T1_C1_IN06	LEDs	All the LEDs placed inside Box One. This will be two strips of 5mm LEDs (one placed around the Ferris wheel subsystem and one on the right-wall above the staircase subsystem).
T1_C1_IN07	Motors	The two DC motors used inside Box One (where one motor will be placed on the Ferris wheel subsystem and the other will be placed above the staircase subsystem).

3.2 - Exclusions

The following table presents a list of exclusions for the project.

Table 12 - List of Exclusions

ID	Exclusion	Description
T1_C1_EX01	Motions subsystem components	Any components or technical information related to how marbles will physically move through the box. This responsibility is for the Motions team.
T1_C1_EX02	Structures subsystem components	Any components or technical information on the materials or construction of the box. This responsibility is for the Structures team.
T1_C1_EX03	Mounting Rack	The final wall that all the boxes get mounted onto.
T1_C1_EX04	Telecommunications and power	The power system and Bluetooth adapters that will be provided for the box.
T1_C1_EX05	Cable Routing	The arrangement of any wires and cables in a way so that it is aesthetically pleasing.
T1_C1_EX06	Circuit Design	The layout of the circuit and wiring needed to run the system.

4.0 - CONCEPTUAL DESIGN

The following section of the document discusses the conceptual design of the Comms team concerning the design of the code, and the placement of LEDs, sensors, and motors inside the box. This section is intended to describe the initial ideas and concepts that were detailed later in development, as well as some alternative design choices that were excluded in order to proceed with the final design.

4.1 - Conceptual Design - Code

After the scope of the project was clearly defined and written, the conceptual design was required to be established. The T1_Comms_1 team decided that the best way to make this happen was through the use of diagrams and pseudocode. Both are effective at demonstrating an overview of how the various components of the design should interact with one another, and how this will achieve the planned goals that were previously written in the Scoping and Requirements Document.

Figure 6 in the appendix shows the general structure of the Comms team's conceptual design, and *Figure 7* shows the pseudocode that would guide the team through the development of the code. It can be seen from these two figures that the conceptual design was specifically designed so that different individuals could work on different functions of the program simultaneously, with the intention of saving time for implementing the solution.

The code begins with a function called "setup", which runs only one time, allowing variables to be initialised and libraries to be imported before proceeding with the loop function. The "loop" function runs continually until the program is stopped by the user. This function makes calls to three other functions called DetectRemote, FerrisWheelMotor, and StaircaseMotor. Both motor-related functions are used to run the motor for their respective subsystem (the ferris wheel and the staircase). The "DetectRemote" function was planned to constantly check whether the customer's remote was activated or not. If so, the function would then make a call to the "DisplayGroupBoxPattern" function, displaying a type of LED pattern consistent with the other boxes in the Massive Marvellous Multiple Marble Machine. If the remote is not activated, the function was intended to make a call to the "DisplayOurBoxPattern" function, displaying a LED pattern unique to Box One. Each of these functions will be explained in greater detail in the following section (*Detailed Design*).

4.2 - Conceptual Design - LEDs

LEDs served as a core component of the final box due to the added functionality. They are a multifunctional system that can be used for feedback, status, troubleshooting and overall added flair. Several decisions were made surrounding the LEDs regarding their functionality and implementation which were discussed in full.

Initial Plans

The initial plans were to have a traffic light system to indicate when marbles were moving around the track, but this was deemed redundant as the marbles would be in constant motion. Another plan involved having the LEDs act as status lights to inform viewers when a motor was malfunctioning or inoperative for an undefined reason which would allow for clear identification of problems and narrow down problem-solving steps in the event of a mechanical failure. The last plan is to also have LED patterns sync up between different marble boxes, the idea was conceived in unison with other computer engineering groups and would make the final product look smoother between the transitions.

Final Decision

Based on the team's initial plans, it was decided that the LED's functions must achieve this including a status LED system and synchronised lighting between boxes.

To accomplish the requirements that were set out for the LEDs, the team had three methods of implementation:

1. Individual LEDs that would be wired together.
2. A LED strip.
3. A hybrid of the two implementations which was immediately disregarded due to added costs.

The decisions made by T1_Comms_1 was founded on two primary considerations:

1. **Physical Implementation and Cost:** Utilising singular LEDs would involve hiding the wiring to make the final box look cleaner, but if it is accomplished properly, it could ultimately lead to a much cleaner appearance and potential saving since we don't purchase any unnecessary LEDs. An LED strip could allow for a simpler installation by comparison as the entire strip could be run across the box with relative ease and a smoother transition between boxes as the strip would likely be more synced due to the innate nature of same length wiring and latency sync across the LEDs, all of these benefits come with the downside of the added cost for the inflexibility but upon performing a brief cost benefit analysis it was that the amount of manual labour that was saved compared to the added cost was worth spending the extra money for the LED strip. This was further backed by websites such as Jaycar electronics selling the full W2812B LED strip for only \$34 at a length that was greater then what was required providing much more flexibility
2. **Software Implementation:** Individual LEDs would require a custom implementation of an existing RGB control library in order to control the individual LEDs unless added effort was made to wire them in parallel which would lead to increased cable management and added labour and a potential downside for the final product. The individual LEDs were also not supported by the FastLED or NeoPixel library that we were planning to utilise for the LED control which the W2812B supported naturally out of the box.

Final Decision

Between the 2 viable choices, (individual LEDs and an LED strip), based on intending functionality and compatibility with our pre-existing requirements it was decided to buy an LED strip. The team bought the BTF-Lighting W2812B due to its compatibility with the already existing arduino library Adafruit Neopixel and cheaper overall cost coming in at \$34.

4.3 - Conceptual Design - Sensors

Sensors were planned to be one of the crucial sub-systems for Box One. It was intended to be used as a simple marble detection system that other subsystems (like the track splitter, LEDs, Ferris wheel, etc.) can use to adjust their properties.

Initial Plans

The initial plan for the sensors was that the team was going to use it to determine the number of marbles in the system, then use this information to control the sub-systems such as the Ferris wheel and splitter so that congestions were minimal. The team had two options available at the time: ultrasonic sensors and infrared sensors. Both of types of sensors had their pros and cons:

- Ultrasonic sensors had great detection accuracy, but the data type returned could give some difficulties during the implementation process.
- Infrared sensors, in contrast, provide boolean data type which made it easy to implement into our system, but it had problems with detection stability.

The team has since decided that ultrasonic is the better option due to its detection accuracy.

Final decision

After the design review that took place with the customer, the team concluded that some subsystems needed to be removed to optimise the flow of marbles inside the machine, prevent unnecessary complications and improve stability. Unfortunately, the sensors were one of those subsystems to be excluded. The splitter subsystem had been removed. The Ferris wheel had been revamped so that it can run at constant speed whilst minimising congestion. Error indication now used voltage as the detection threshold, thus removing the need for the existence of sensors.

4.4 - Conceptual Design - Motors

Motors are crucial to Box One, as they provide the necessary motion to move the marbles in the specific ways that are required. The motors serve one main function, and several considerations and alternatives were considered regarding their implementation.

Motor Uses

The initial plans for Box One from the perspective of motors was that three motors were required for three separate sub-subsystems. One motor was to be used to rotate the ferris wheel sub-system, which is required to spin at a constant speed. Another motor was to be used in the staircase subsystem to rotate a gear such that the staircase oscillates up and down when the motor spins. The final motor was to be used in the splitter subsystem, which diverts the marbles to one of two tracks depending on sensor input.

Alternatives

The options for motors that we considered were stepper motors or DC motors. These were the main options considered as they both can fulfil the requirements. The stepper motor allows for precise control of the motor spin such that the degree of rotation can be controlled. The stepper motor, however, does not return whether it has successfully spun like a DC motor would. The DC motor continuously spins at a consistent speed and could be achieved after allowing for some startup time. The DC motor requires more voltage to operate, and with three motors connected, it would not be able to function properly. Additionally, the DC motor has relatively higher cost compared to the stepper motor.

Initial plans

In the end, the motors chosen for the marble machine were stepper motors because of the cost and ability to specifically control rotation. The continuous torque provided was also a positive factor as the team's main worry was the rate of input. Additionally, the DC motor was not chosen because of its higher cost and need for more voltage.

Final Decision

The team's final decision for the type of motor was ultimately changed due to problems that were raised in the design review. This caused the overall design to change as large sections of Box One were changed such that three motors were no longer required. The splitter subsystem was removed as the second track was removed; thus, our voltage usage was greatly lowered. This additionally reduced the cost allowing for the alternative of DC motors to be used. This means that the DC motor became a much better choice as it provides continuous rotation to two subsystems that should be spinning at a constant speed. The DC motor also allows for the Arduino to detect if the motor is not spinning, thus, allowing it to determine if the motors have stopped and display as such through the LEDs.

5.0 - DETAILED DESIGN

The following section of the document discusses the final detailed design of the Comms team. The intention of this section is to describe the role of each subsystem component in greater detail than the conceptual design, explaining certain programming choices, the types of motors and LEDs, and all the functions used in the code.

5.1 - Detailed Design - Code

The following table describes all the functions present in the final code, including their name and purpose in the code.

Table 13 - Table of Algorithm Functions

ID	Function Name	Class	Input	Description
T1_C1_F01	stop	Motor	None	Stops the motor by setting its speed to zero.
T1_C1_F02	getStatus	Motor	None	Gets the status of the respective motor.
T1_C1_F03	setSpeed	Motor	Speed	Sets the speed of the respective motor to a specific value.
T1_C1_F04	forward	Motor	Speed	Sets the motor to move forwards at a specific speed value.
T1_C1_F05	backward	Motor	Speed	Sets the motor to move backwards at a specific speed value.
T1_C1_F06	offset	Motor	None	Gets the trigger offset value for the respective motor.
T1_C1_F07	calcNextTriggerTime	Motor	lastTriggerTime	Uses the current time and trigger offset to calculate the next trigger time.
T1_C1_F08	setup	Main	None	The standard Arduino function for initialising and setting up variables.
T1_C1_F09	loop	Main	None	The standard Arduino function for continually executing code.
T1_C1_F10	traffic	Main	State	To indicate an error (by turning all LEDs to red) when a motor stops working as expected.
T1_C1_F11	setRed	Main	None	Sets the LED colour to red for the

				error LED strip.
T1_C1_F12	setGreen	Main	None	Sets the LED colour to green for the error LED strip.
T1_C1_F13	setError	Main	None	Checks whether the motors are working correctly and sets the colour of the LED strip accordingly.
T1_C1_F14	incrementPattern	Main	None	Runs an incrementing LED pattern on the main LED strip.
T1_C1_F15	backForthMult	Main	Pix	Runs a back-and-forth LED pattern on the main LED strip up to a specific pixel distance.
T1_C1_F16	oddSwap	Main	None	Flashes LED colours in an odd-even type of pattern on the main LED strip.
T1_C1_F17	loopSingle	Main	None	Runs a single LED loop on the main LED strip.
T1_C1_F18	stackMult	Main	Pix	Runs a “stacking” LED pattern on the main LED strip up to a specific pixel distance.

5.2 - Detailed Design - LEDs

Ultimately, the team ended up using the 5 metre RGB LED strip controlled directly through the Arduino Uno. No subsystem can directly interact with the LED component or vice versa without the use of the Arduino as a main control source, which can be seen in *Figure 1*. For the purposes of error handling, the signal sent from the motor to the Arduino will be directly related to the code responsible for the LEDs where a colour change to red will occur if any anomalous signal is detected. The following list outlines the locations in the Box One where the LED will be placed strategically to be aesthetically pleasing:

- **Ferris Wheel:** A LED strip will be placed on the outer facing circular surface and will display aesthetic patterns depending on the movement of the wheel. A potential idea is to have the LEDs do a circular loop in sync with the Ferris wheel.
- **Staircase:** A LED strip will be placed on the right wall of the box in the same angle and direction as the staircase module. A potential idea for this module is to have the LEDs increment one by one as the steel ball moves up the staircase, and then have a flashing animation appear, indicating a marble has reached the top of the staircase.

If the box was to interact in unison with other boxes using a Bluetooth module, separate code for the LEDs would run where the selected group LED pattern would run.

Figure 11 shows code for a sample LED pattern. The timing system for the LEDs is calculated with the line of code:

```
if(time >= led_Patt_Lasttriggered + 50)
```

This effectively makes the next trigger of the LEDs occur 50 milliseconds later and resets the led_Patt_Lasttriggered time to the current time.

5.3 - Detailed Design - Motors

The motors that will be implemented are 6V 9,000 RPM DC Electric Motor [2] which will be controlled or operated through the arduino component. If needed, the motor will also be connected to an external power source in the case where the motor is not receiving sufficient voltage. The Arduino Uno acts as the “middleman” for Box One to communicate through code (e.g., speed, direction) with the motors. The motors are also connected to a motor shield [6] where they can tap into the motor’s voltage and current spike, which then will be used for error detection, where it can detect if the hardware motor is moving or not. The motors will be attached to the staircase, and the Ferris wheel components.

- **Staircase Motor:** The motor in the staircase component will turn to move the staircase through gears in a manner where the steel balls will climb up the staircase.
- **Ferris Wheel Motor:** The motor in the Ferris wheel component will rotate at a constant RPM (revolutions per minute). This will create a smooth motion with the gears for the steel balls to be carried up to the spiral component.

The motors that will be implemented will be placed in their respective sections (as shown in *Figure 9* and *Figure 10*).

5.4 - Programming Choices

5.4.1 - Programming Choices - Style of Code

T1_Comms_1 will follow the correct coding standard that has been used for various programming languages (such as C++) to keep the codebase readable, maintainable, and robust. The following are the standards that the team will use:

- Use *camelCase* format to name variables.
- Use meaningful and easy-to-understand variable/function names.
- Use the standard operator spacing and indentation.
- Commenting in detail to explain non-obvious parts of the codebase.
- Use block comments at the start of a logical code block.
- Function structure has a clear and visible usability to ensure modularity: No complicated/multiple purposes functions.
- Minimise the use of magic numbers.
- Handle all errors and edge cases.
- Remove the usage of `delay()` and utilise `millis()` to ensure precision and ensure the “event-driven” approach is followed.

5.4.2 - Programming Choices - Programming Language

To produce the proposed project, the components of Box One must be controlled in an orderly manner. Many microcontrollers exist for this purpose. These microcontrollers house small computers that can be programmed to perform many functions, each having their own advantages and disadvantages. To choose, a list of options and their details was made regarding multiple researched microcontrollers:

1. NodeMCU:

- Small
- Cheap
- Uses Arduino architecture.
- Can program in Lua, Arduino or microPython.
- Wi-Fi.
- Low power consumption.

2. Teensy 4:

- Small.
- Computational power.
- Micro-SD card slot.
- Cheap.
- No headers or breadboard.

3. MSP430 Launchpad

- Very low power consumption.
- Cheap.
- IDE options.
- Low memory.
- Large.

4. Arduino Uno

- C or C++ only.
- Easy to use/beginner friendly.
- Relatively cheap.
- Active user community.
- Cross platform support.
- Libraries.
- Lacks multitasking.
- Not performance oriented.

Although other options were available and briefly looked at, the ones stated above catered to the project and team experience the most. While each has their strengths and weaknesses, the Arduino Uno stood out due to its popular nature and the collective experience team members already have with it, while the other options may perform better computationally or with power, which could help a lot with running multiple components. The Arduino Uno's active community, libraries and shared knowledge make learning and understanding its functionality simple and accessible. If other options were to be explored, time that is not available would need to be spent on learning its functionality.

As per the language, the Arduino Uno offers its own programming language, which is based on C++. The only available language to choose from is Arduino's C++ programming language.

5.5 - Programming Libraries Used

To integrate the necessary hardware components (such as the motors and LEDs) into the project, several code bases and programming libraries were required. These libraries act as “middlemen” that play a crucial role in simplifying the interaction between the code and hardware components. The list below describes the libraries that the team has used their code:

1. <Adafruit_NeoPixel.h>

- **Functionality:** This library is designed for controlling single wire based Adafruit NeoPixel LED strips, which are RGB LEDs that can be individually addressed and controlled.
- **Common Functions:**
 - i. **show():** Used to update the NeoPixel strip with colour and brightness values. After making changes to the LED colour 'show()' makes them visible.
 - ii. **fill():** It allows all LEDs to be set to a specific colour upon request.
 - iii. **clear():** Turn off all the LEDs in the strip.

2. <AFMotor.h>

- **Functionality:** This library is part of the Adafruit Motor Shield library which is used for controlling DC and stepper motors using the Adafruit Motor Shield (add-on board for Arduino).
- **Common Functions:**
 - i. **Controlling DC motors:** Functions such as 'setSpeed()' and 'run()' are used to set the motor speed and direction.
 - ii. **Controlling stepper motors:** Functions such as 'step()' are used to control the number of steps the motor takes.

3. <Servo.h>

- **Functionality:** This library is used for controlling servo motors (motors that can be precisely positioned at various angles).
- **Common Functions:**
 - i. **attach():** Used to attach a servo object to a specific pin on the Arduino board.
 - ii. **write():** Used to set the angle at which the servo should be positioned, a value between 0 - 180 degrees is taken as input.
 - iii. **read():** Reads the current position of the servo motor in degrees.

6.0 - DESIGN TRACEABILITY

Table 14 - Design Traceability Table

Requirement ID	Description	Design Criteria to Satisfy Requirement
T1_C1_FR01	One of the primary requirements is that the cube must be able to interact with marbles in a kinetic manner. This means the cube should be designed with mechanisms that allow it to move, rotate or otherwise manipulate the marble.	The cube incorporates motor-driven mechanisms to manipulate the marble's behaviour.
T1_C1_FR02	The cube should enable marbles to enter and exit at regular one-second intervals.	The use of the staircase kinetic module ensures the rate at which a marble exits is one second, where it is assumed that the marbles will enter the cube with a required minimum interval of a second also.
T1_C1_FR05	An Arduino Uno will need to be utilised to control all mechanical components in the box including motors and the LED lighting.	C++ code will be utilised in the Arduino IDE to control all mechanical and lighting components within the cube.
T1_C1_FR06	The cube will have some kind of aesthetically pleasing lighting effects that will interact with the rest of the components.	High quality LED strips have been incorporated into the cube where they are controlled using the Arduino Uno. Several pleasing patterns have been created in the code.
T1_C1_FR07	The cube will house a kinetic module that resembles a ferris wheel. It will prove to be aesthetically pleasing and provide a way for the marble to be manipulated.	The ferris wheel is powered by a high torque DC motor. The speed at which the motor spins is determined by the code in charge of the subsystems involving motor-driven mechanisms.

T1_C1_FR08	To accompany the ferris wheel kinetic feature, the cube will incorporate a staircase feature that will also be added which will oscillate bringing the marbles moving them up the cube to the exit.	The staircase is another kinetic subsystem which will be powered by a high torque DC motor. Again, the speed and power of the motor is controlled via the code in charge of the motor driven subsystems. The motor will be used in a way that it oscillates marble causing them to 'climb' the stairs.
T1_C1_PR01	All electronic components in the cube must be kept tidy and safe from any risks. In no circumstance should any injury or harm be caused to anyone who is observing it.	The electronic components are all hidden under custom built shrouds to reduce the chances of injury and there are fuses in place when the motor and electronics are impeded.
T1_C1_PR02	Latency between the Arduino Uno and dependent components (LEDS, motors) must be kept to a minimum.	The Arduino code that will be responsible for the necessary subsystems is efficient as possible. It is ensured that no input mechanisms that will bottleneck the whole system are used and that all commands are run through a unified timing system. To ensure no hardware bottlenecks, wiring runs will be kept as short as possible.
T1_C1_PR03	The LEDs in the cube must shine brightly at a level where it is clearly visible to the observer without the LEDs being too dim inside the cube.	The maximum available voltage (5V) of the Arduino Uno will run through the LED strip and in the code the brightness of LED strip is set to max.
T1_C1_PR04	It is expected that the LEDs should shine in the intended patterns that exist in the code.	Integration testing will be done rigorously on all LED patterns ensuring correct functionality. The testing process will also be testing the LEDs together with the motors to ensure that there is no dependency issues.

T1_C1_PR05	The code that controls all the components in the cube must have a desirable time complexity that must be efficient enough to ensure high, smoothness and minimum delay.	Each function of the code is run on a single timing loop with no additional loops. This ensures a time complexity of $O(n)$ due to sequential execution.
T1_C1_PR06	The code must be memory efficient whereby the max memory limit of the Arduino Uno is not used.	The code has minimum states and the states that do exist will be managed in form of single byte Boolean flags. This ensures minimal memory complexity and usage.
T1_C1_PR07	Code must be modular in design, being able to easily add, edit and remove parts improving maintainability.	Each subsystem in the code is managed by their own states and functions ensuring modality whereby modifications are easily added, and the code is easy to maintain.
T1_C1_PR08	The code should be able to be modified to work in many scenarios without influencing other code segments.	The modularity characteristic of the already existing code ensures that any modifications made will not affect the already existing functions.
T1_C1_PR09	The code should be sustainable where through repeated uses it will work without bugs throughout its life cycle.	The code will be tested rigorously both on paper and during execution ironing out any deficiencies making it free from all bugs. The code testing will be automated by unit tests that will be conducted before and after integration testing.
T1_C1_IR01	The placement of components inside the box must not affect the movement of marbles in any way that alter the intended movement or forces it off track.	All the components of the cube is placed in a way that separates each component into a specific part of the cube where no components move or shake the cube or any other components unnecessarily. To ensure the placement of the components a CAD rendering of the cube with all the components had

		been done making the actual building process straightforward.
T1_C1_IR02	Any wiring must be managed in a way as not to affect any other components or hinder the movement of the marbles.	All protruding cables are securely tucked away along specified locations in a way that they are at least 1cm away from any components.
T1_C1_IR04	The Arduino shall be connected to a DC motor that rotates the Ferris wheel component. Rotation speed is based on T1_C1_PR10.	The Ferris wheel spins appropriately, performing at a rate that can keep up with the marble input.
T1_C1_IR05	The Arduino shall be connected to a second DC motor that rotates the Staircase. Rotation speed is based on T1_C1_PR11.	The staircase oscillates appropriately, performing at a rate that can keep up with the marble input.
T1_C1_IR06	The Arduino shall be connected to two light strips. The first light strip is fixated around the ferris wheel and set with a random pattern from the code. The second light strip is set up alongside the staircase and while showing a random pattern should also light up red when an error with the motors is detected.	To ensure the correct positioning of the LEDs, they are secured at their designated locations in a way where it will not come loose. The code works well where it shows the correct patterns where all normal patterns are shown unless an error occurs where the LEDs then will show the error pattern.

7.0 - PROJECT SCHEDULE

The team planned the following documentation schedule to clarify when documents must be completed.

Table 15 - Project Document Schedule

Name of Document	Date
Scoping and Requirements Document	20/08/2023
Design Document	15/09/2023
Testing Document	05/11/2023
Statement of Work	05/11/2023

8.0 - TESTING

8.1 - Technical Performance Measures (TPMs)

TPMs (Technical Performance Measures) are a way of measuring if the project satisfactorily meets its requirements. The following table presents a list of technical performance measures, where each corresponds to every functional and performance requirement previously listed.

Table 16 - Technical Performance Measures

TPM ID	Requirement ID	TPM Description
T1_C1_TPM01	T1_C1_FR01	All the mechanisms inside the box must function correctly and accurately in 100% of the planned test cases.
T1_C1_TPM02	T1_C1_FR02	The box must be able to function correctly when marbles are placed through the entrance in 1-second intervals.
T1_C1_TPM03	T1_C1_FR03	The sensors can successfully detect the presence of marbles in one section of its path in 95% of planned test cases.

T1_C1_TPM04	T1_C1_FR04	Marbles travelling throughout the box must be physically stationary for 0.5 seconds.
T1_C1_TPM05	T1_C1_FR05	The Arduino Uno must control 100% of the electronic components inside the box.
T1_C1_TPM06	T1_C1_FR06	75% of the testers of the box must find the LEDs pleasing to them in some way.
T1_C1_TPM07	T1_C1_FR07	The motor controlling the Ferris wheel must function consistently and correctly in 100% of the test cases.
T1_C1_TPM08	T1_C1_FR08	The stepper motor controlling the movement of the staircase must function consistently and correctly in 100% of the test cases.
T1_C1_TPM09	T1_C1_FR09	The marble track splitter component must be able to function consistently and correctly in 100% of the test cases.
T1_C1_TPM10	T1_C1_PR01	During the testing period, 100% of project testers must not be injured by moving parts when simply observing the box in operation.
T1_C1_TPM11	T1_C1_PR02	There must be less than 500 milliseconds of delay between sensor detection and component execution.
T1_C1_TPM12	T1_C1_PR03	All sets of LEDs must use a 20mA current.
T1_C1_TPM13	T1_C1_PR04	The code must function 100% correctly in 99.95% of test cases.
T1_C1_TPM14	T1_C1_PR05	The time complexity of the final code must be less than or equal to $O(n)$.
T1_C1_TPM15	T1_C1_PR06	The amount of memory used by the code must be less than or equal to 32kb, as this is the memory limit of an Arduino Uno.

T1_C1_TPM16	T1_C1_PR07	The code must be written in the standard C++ style with correct indentation and commenting.
T1_C1_TPM17	T1_C1_PR08	The average time to make an adjustment must be less than 5 minutes.
T1_C1_TPM18	T1_C1_PR09	The box must be able to fully operate without stopping for two weeks.
T1_C1_TPM19	T1_C1_PR10	The rotation speed of the Ferris wheel must be 1 second multiplied by the number of marble slots in the Ferris wheel, and the rotation speed must be constant.
T1_C1_TPM20	T1_C1_PR11	The staircase oscillation must be around 20 rpm. Further figures will be calculated by the Motions team.
T1_C1_TPM21	T1_C1_IR01	In 100% of the planned test cases, all marbles remain on the track and are completely unaffected by other components.
T1_C1_TPM22	T1_C1_IR02	In 100% of the planned test cases, all marbles remain on the track and are completely unaffected by wires and cables.
T1_C1_TPM23	T1_C1_IR03	The ultrasonic sensors can successfully detect the presence of marbles before they reach the track splitter and the screw in 95% of planned test cases.
T1_C1_TPM24	T1_C1_IR04	The rotation speed of the Ferris wheel must be 1 second multiplied by the number of marble slots in the Ferris wheel, and the rotation speed must be constant.
T1_C1_TPM25	T1_C1_IR05	The staircase oscillation must be approximately 20 rotations per minute.
T1_C1_TPM26	T1_C1_IR06	The interaction between the light strip and sensor is functional in 95% of the planned test cases.
T1_C1_TPM27	T1_C1_IR07	The colour-changing traffic light LED has a delay of less than 500 milliseconds and the colour is correct in 95% of the planned test cases.

8.2 - Test Cases

The following table presents a series of test cases that have been planned to be carried out in the testing phase of the project's development. Each individual test case has a unique identifier, a short description about its purpose/goal, its input if there is one, and the expected output upon successful completion of the test. This section does not suggest that the tests will be conducted in a specific order, but rather shows the types of tests that may be conducted at any time before it is completed and presented.

Table 17 - Test Cases

Test Case ID	Test Case Description	Input (if any)	Output
TC01	Verify that Box One can successfully use the previously described subsystems to kinetically interact with marbles passing through the box.	No input	The box successfully handles marbles kinetically through its inner subsystems.
TC02	Verify that the cube can handle marbles passing through at one-per-second (or longer) intervals.	Marbles	The box (including all its subsystems) is able to accept marbles at a rate of one-per-second or longer without any issues.
TC03	Verify that the incrementPatt pattern functionally operates as intended.	No input	The LED strip slowly lights up, with one single LED at a time. Each one has a randomised colour.
TC04	Verify that the backForthMult pattern functionally operates as intended.	No input	The LED strip gives the appearance of a few LEDs moving along the strip back and forth from the right side to the left and so on without failure.
TC05	Verify that the oddSwap pattern functionally operates as intended.	No input	The LED strip continually flashes all the odd-numbered LEDs in the strip followed by the even LEDs in the strip repeatedly without failure.

TC06	Verify that the loopSingle pattern functionally operates as intended.	No input	The LED strip gives the appearance of one single LED moving from the left side to the right in a continuous loop.
TC07	Verify that the stackMult pattern functionally operates as intended.	No input	The LED strip continually lights up a new LED at the end of the strip until the whole strip is full of lit LEDs.
TC08	Verify that the DC motor can run the Ferris wheel module. The motor and its wires do not affect the movement of marbles.	No input	The Ferris wheel module is able to rotate as expected with no issues and having no impact on the movement of marbles inside the box.
TC09	Verify that the speed of the Ferris wheel DC motor is accurate to the value set in the code.	The desired motor speed	The speed of the Ferris wheel's own DC motor falls within an acceptable margin with the given input value.
TC10	Verify that the DC motor can run the staircase module. The motor and its wires do not affect the movement of marbles.	No input	The staircase module can oscillate as expected with no issues and having no impact on the movement of marbles inside the box.
TC11	Verify that the speed of the staircase DC motor is accurate to the value set in the code.	The desired motor speed	The speed of the Ferris wheel's own DC motor falls within an acceptable margin with the given input value.
TC12	Verify that all components that have been implemented are safe and have no risk of injuring any person watching.	No input	Not one person watching the box operate is physically injured by moving parts.
TC13	Verify that all computational instructions for all components are with sufficient speed.	No input	There is no noticeable delay when the code is executed.
TC14	Verify that the LEDs on the LED strip are as bright as they were programmed to be.	No input	The brightness of the LEDs on the LED strip falls within a narrow acceptable range.

TC15	Verify that the memory usage at any given time in operation is always less than or equal to 32kb.	No input	The amount of memory used is never more than 32kb.
TC16	Verify the quality of the code design, ensuring that it is easy to edit, add, or delete units of code.	No input	Making changes to a specific function does not break the whole algorithm.
TC17	Verify the reliability of the code by running the solution many times.	Marbles	After many successful trials, the solution runs successfully.
TC18	Verify that the Arduino Uno that is used can successfully control the LED strip, and both DC motors from their respective modules.	No input	The LEDs shine when connected to an Arduino Uno.
TC19	Verify that when either of the two motors stops moving entirely, the LEDs on the LED strip shine red.	No input	All the LEDs on the LED strip shine red, indicating there is a problem with the DC motor.

8.3 - Verification of Requirements

The purpose of this section is to show how every current requirement is verified by one or more specific test cases. It must be stated that it was necessary to slightly modify or remove certain initial requirements during the development of the project. If this was the case, it has been clearly indicated for the specific requirement in the table below.

Table 18 - Verification of Requirements Table

Requirement ID	Description	Test Case to Verify Requirement
T1_C1_FR01	The cube must be able to utilise mechanisms to interact with the marble kinetically.	TC01
T1_C1_FR02	The cube must allow marbles to enter and exit with the interval of 1 second.	TC02
T1_C1_FR03	Sensors set up inside the box must be able to detect marbles at different areas of the track.	N/A - requirement removed
T1_C1_FR04	Marbles moving through the box must be able to stop moving in all physical dimensions for a short period of time.	N/A - requirement removed

T1_C1_FR05	Arduino Uno will be utilised to control all mechanical, sensors and lighting components inside the cube.	TC18
T1_C1_FR06	The cube will have LED strips and traffic lights as indicators light with pleasing effects.	TC18, TC03, TC04, TC05, TC06, TC07
T1_C1_FR07	The cube will have a Ferris wheel operated by a DC motor to position the marbles inside the box.	TC08
T1_C1_FR08	The cube will have a Staircase operated by a DC motor to elevate the marbles.	TC10
T1_C1_FR09	The cube will have a splitter operated by a servo that splits the marbles to different tracks inside the box.	N/A - requirement removed
T1_C1_PR01	No electronic component should be able to injure or harm any person observing it.	TC12
T1_C1_PR02	Latency between sensors and other components (servos, lighting, and motors) must be minimal.	TC13
T1_C1_PR03	The LEDs should shine brightly.	TC14
T1_C1_PR04	The LEDs should shine in patterns as expected from the written code.	TC03, TC04, TC05, TC06, TC07
T1_C1_PR05	The time complexity must be efficient to ensure high speed, smoothness, and minimal delay.	TC13
T1_C1_PR06	Memory must be efficiently used in the code.	TC15
T1_C1_PR07	Code must be modular in design, being able to easily, add, edit, and remove parts.	TC16
T1_C1_PR08	Being able to modify existing code easily to work in many scenarios without influencing other code segments.	TC16
T1_C1_PR09	The code will work through repeated uses without bugs or issues throughout the lifecycle.	TC17
T1_C1_PR10	The default spinning speed for the Ferris wheel that should keep marbles travel smoothly inside the machine.	TC09
T1_C1_PR11	The default oscillation for the staircase that should move the marbles upward.	TC11

T1_C1_IR01	The placement of components inside the box must not affect the movement of marbles in any way that forces it off track.	TC08, TC10
T1_C1_IR02	Any wires and cables need to be arranged in a way as not to affect the movement of the marbles.	TC08, TC10
T1_C1_IR03	The cable will use ultrasonic sensors to detect marbles before arriving at the splitter and the screw.	N/A - requirement removed
T1_C1_IR04	The Arduino shall be connected to a DC motor that rotates a Ferris wheel. Rotation speed is based on T1_C1_PR10.	TC08, TC09
T1_C1_IR05	The Arduino shall be connected to a second DC motor that rotates the Staircase. Rotation speed is based on T1_C1_PR11.	TC10, TC11
T1_C1_IR06	The Arduino shall be connected to 2 light strips. The first light strip is fixated around the ferris wheel and set with a looping rainbow colour. The second light strip is set up alongside the main ramp and lights up if the sensor detects marbles on the track.	TC18
T1_C1_IR07	The Arduino shall be connected to another traffic light LED fixated to the splitter. The light will change colour based on the rotation of the splitter servo.	TC18, TC19

8.4 - Testing Environments

The testing environment being used is a controlled environment with an early prototype of the Marble Machine Box with all its subsystems assembled and connected to the power. The testing environment includes:

- The physical structure of the Marble Machine Box.
- The necessary electronics required for the Box to function properly.
- A reliable surface for the Box to rest on with zero physical effects on the box.
- All members of the responsible team that work on the Box.
- The staff of the MQ Engineering faculty.

The test environment is expected to simulate the installation requirements (height and orientation) provided by the client.

8.5 - Resources and Equipment

All resources and equipment that were used throughout the testing phase provided by T1_C1 and T1_M6 are:

8.5.1 - Digital

- Test Arduino file (.ino).
- Relevant software and documentation such as Arduino IDE, Visual Studio Code, Github, Arduino and Arduino libraries manual pages.

8.5.2 - Physical

- Two laptops: one will be used as the primary machine for testing and the other will be used as a backup device.
- A type-B USB cable to connect the laptop to the Box.
- The prototype Marble Machine Box.
- Five marbles.
- Spare parts such as wiring, Arduino and other physical components related to the Marble Machine Box.

8.6 - Testing Modes

It was described that the most efficient mode of testing would be running the system automatically while multiple testers record the outputs produced during the testing. After running the machine, there should be no interaction from the user with the machine.

Testing will be done by both physically examining the behaviour of the box and by conducting code verification tests. The testing of the code for the motors and LED subsystems will include both these methods, physically observing the movement of DC motors along with the patterns produced by the LED strips. Then automated tests will run on the code to make sure no bugs are represented as well as any bottlenecks.

8.7 - Types of Testing

8.7.1 - Unit Testing

After all components are fully developed, each one will undergo unit testing where they will be tested in various ways. For example, the LED component will be tested with a white box testing technique where the component will be given an input, which will then be given to the internal code and check if the outcomes are equivalent to the expected outcomes. This test ensures that a component's functionality works by itself.

8.7.2 - Integration Testing

After all components have successfully passed all unit tests, we can then move on to integration testing, where we test if each component will work well together concurrently. We will have the motor and LED components attached to the half-built box for visibility and test if both components work well together. This phase of testing determines the compatibility between the components.

8.7.3 – Performance Testing

With all the components completing the unit and integration testing phases, our final testing phase will be performance testing, where we test the reliability of the box. We will turn the box on with the components active and wait for an extended amount of time (approximately 2 hours). Through this phase of testing, we will determine its reliability.

8.8 - General Testing Procedure

Log:

- Record the results acquired from executing each of the tests in the logbook.
- Record the date, time, and execution details, such as the run number, running time etc.

Setup:

- Gather all the necessary resources and equipment together. This is specified in the 'resources and equipment' section of this document where both physical and digital resources are utilised.

Start:

- Power on all necessary equipment to start conducting the tests.
- Have spare equipment on standby in case of a failure.

Proceed:

- Run all the test cases specified for each day following the testing schedule.
- Monitor for any failures and repairs that need to be done.

Measure:

- Record the results and findings from each test case and compare to the expected results.
- If expected results are not met, design a framework to come up with a solution.

Wrap-Up:

- Power off all the equipment used for testing.

Shut Down:

- Return the equipment to their designated storage locations.

8.9 - Specific Testing Procedures

Test Case	T_00 Unified Time System
Environment	Digital
Success Condition	If all elements of the box activate at their specified trigger time
Failure Condition	If the time is out of sync with real time
Method	<ol style="list-style-type: none">1. Set a trigger time for a component such as LEDs2. Begin the program3. Begin a second counter at roughly the same time4. Given the known component trigger time, the component should trigger at roughly the same time the second counter reaches the trigger time
Alternative Flow	<ol style="list-style-type: none">1. Component does not trigger2. Indicates error in time system

Test Case	T_01 Motor Class
Environment	Digital
Success Condition	Successfully create a motor object that holds the needed values and can be used to run the motor
Failure Condition	Motor object cannot be used to run the motor
Method	<ol style="list-style-type: none">1. Instantiate motor object with values expected to be use2. Pass the motor object into the turn motor function to run the motor3. Test if the motor spins
Alternative Flow	<ol style="list-style-type: none">1. Motor object fails to initialise or be run2. Stop the system3. Error in the motor class or motor

Test Case	T_02 LED system
Environment	Physical
Success Condition	LEDs turn on
Failure Condition	LEDs do not turn on
Method	<ol style="list-style-type: none"> 1. Wire LEDs to arduino in a test environment 2. Activate a simple colour pattern to be shown on the LEDs 3. The LEDs should turn on matching with the colour pattern
Alternative Flow	<ol style="list-style-type: none"> 1. If the LED doesn't work 2. Utilise

Test Case	T_03 LED pattern
Environment	Physical
Success Condition	LEDs show instructed pattern
Failure Condition	LEDs do not show instructed pattern
Method	<ol style="list-style-type: none"> 1. Wire LEDs to arduino in a test environment 2. Activate a random pattern that is to be shown on the LEDs 3. The LEDs should then match the pattern being printed
Alternative Flow	<ol style="list-style-type: none"> 3. If the LED doesn't work or the pattern doesn't match 4. Utilise

Test Case	T_04 Motors
Environment	Physical
Success Condition	Motors exhibit appropriate spinning when component activation is ran (E.g, spin / not spin)
Failure Condition	If the motors do not react to the code implementation at all
Method	<ol style="list-style-type: none"> 1. Instantiate the code and components 2. Start the spin motor code 3. Start the stop motor code
Alternative Flow	<ol style="list-style-type: none"> 1. Motor does not spin at all 2. Error within the motor or code

9.0 - DESIGN REVIEW AND CONCLUSION

9.1 - Review

In the final design, there are four major sections in the code: time, motor, error detection, and LED. These sections all underwent minor and major changes, which ultimately depended on how the team felt the requirements would be best fulfilled based on the implementation of the motors and structures teams.

The time subsystem underwent two major upheavals. In the initial design, all the code for the marble machine went into one main "for loop", thus the architecture for the code ended up being largely procedural. After the design review, the code was changed to become event-driven, so the code can trigger at precise times. To do this, the code now reads the current "millis()" time at the top of the loop, and all other subsystems determine if they should trigger based on a modulus calculation. This timing system, however, is often likely to fail because the "millis()" function may not read the time sequentially. As a result, the time system needed to be changed again so that it focuses on reliability and robustness over precision. It does this by triggering at a particular point in time, allowing for a larger range of time. This guarantees that all events will always trigger, even if it is a couple of milliseconds off the intended trigger time.

The motor subsystem had multiple minor changes occur. In the initial design, we determined that using classes would be the best architecture because we have two motors that both require the same methods. Throughout the prototyping phase, the motor driver changed, and the motor library we used also changed, but the changes were only very minor. Ultimately, Motions decided to use the Adafruit motor shield as a motor driver, so it was necessary to alter the code to fulfil the requirements set by Motions.

In the motor subsystem, the team also needed to implement error detection to fulfil a requirement of determining when the motors fail. This is done by reading the current and determining if the value is above a certain threshold. This section of code underwent some minor changes as the current changed; in the end however, no major changes occurred to this subsystem.

In the initial motor subsystem design, it additionally had an error detection system which determined when a motor was failing based on the current. This system ultimately did not work, as it would short circuit the whole system. Thus, the error detection subsystem was ultimately dropped.

In the LED subsystem, the initial design ended up being the final design because of its ease of use and versatility. Throughout the production of the LED subsystem, the only changes made were to the pattern displayed.

9.2 - Further Improvements

9.2.1 - Introduction

The final design of the marble machine's programming showed a good integration of both software and hardware components to achieve the planned functionality. The core of the design was an event driven paradigm that was facilitated by a custom integration of a time system. This system played a key role in the synchronisation of the various components such as the LEDs and motor operations. Though the final system was well designed, like any engineered system, there were various enhancements that could have been made to further improve the design, its robustness and by extending its capabilities.

9.2.2 - Code Structure - Motor Class Improvements

The motor class was well designed and was instrumental in managing the various motor operations and conditions. It covered the necessary grounds that were initially planned but could have benefited from a more succinct implementation and further data encapsulation through techniques such as enumerators and additional functions to enhance efficiency and maintainability in the future.

9.2.3 - Code Structure - LED Implementation

The final LED control flow though sufficient and accurately displaying patterns could be better optimised through a clearer decomposition of the function and further data encapsulation. This would streamline the code, contribute to a cleaner codebase, make it easier to manage, and expand to add more patterns in the future.

9.2.4 - Time System

The custom time tracking implementation served as the foundation to the team's event driven program. It performs its job well and ensures that the system conforms to the tight operational requirements that were imposed. It could be improved further through the use of Real time clock (RTC) chips that could be integrated in the Arduino. The addition of this chip would ensure a much more consistent and precise time system as well as providing a fallback for the system in the event of a power failure, decreasing inconsistencies and improving the overall system performance.

9.2.5 - Error Handling and Monitoring - Error Detection

The current system has no real implementation of error detection, with a limited scope provided by internal tracking systems and does not have an external method of detecting when a component or module is inactive though many attempts were made. The component integration was in a manner that did not allow for it. Provided more time, a proper system would be implemented and would prove a necessary addition to the program.

9.2.6 - Error Handling and Monitoring - Real Time Monitoring

Coupled with the improvements that should be made to error detection would shed light on a necessary monitoring system, especially tracking states and fields such as current draw will provide value insight into the system's current state and prove paramount in proactive maintenance and troubleshooting as well as ensuring the longevity of the machine.

9.2.7 - Hardware Improvements - Power Efficiency

The current machine enlists the continuous operation of the motors and components regardless of their necessity. In the long run, this would drastically reduce power efficiency and poses the opportunity for implementing methods of prevention such as the use of sensors and triggers reducing power consumption.

9.2.8 - Hardware Improvements - Additional Hardware

Alongside the implementation of sensors and triggers, we would gain access to more dynamic functionality and behaviours such as trigger patterns for motors and would improve user experience and the functionality of the marble machine.

9.2.9 - Conclusion

Throughout the conception and actualisation of the marble machine, we faced many challenges and learnt new techniques. The final design is a well designed and implemented system that is well coordinated and operates in a satisfactory manner. Regardless of the time constraints and operational challenges there were improvements and enhancements that could be made in the mentioned areas that would concrete a smooth operation and increasingly efficient implementation of the marble machine.

REFERENCES

[1] "23S2_ENGG2K3Kv1.3 - 2023 SPINE Engineering Project ENGG2000/ENGG3000 - Massive Marvellous Multiple Marble Machine", Available:

<https://ilearn.mq.edu.au/mod/resource/view.php?id=7833939>. [Accessed: Aug. 15, 2023]

[2] "6v-9-000-rpm-dc-electric-motor", Available:

<https://www.jaycar.com.au/6v-9-000-rpm-dc-electric-motor/p/YM2712>. [Accessed: Sep 6, 2023]

[3] HACKADAY, "NEW TEENSY 4.0 BLOWS AWAY BENCHMARKS, IMPLEMENTS SELF-RECOVERY, RETURNS TO SMALLER FORM", Ted Yapo, August 7 2019, Available :

<https://hackaday.com/2019/08/07/new-tensy-4-0-blows-away-benchmarks-implements-self-recovery-returns-to-smaller-form/> [Accessed: Sep. 12, 2023]

[4] robocraze, "ARDUINO VS NODEMCU", Robocraze, March 15 2022, Available:

<https://robocraze.com/blogs/post/arduino-vs-nodemcu> [Accessed: Sep, 13, 2023]

[5] Embedded Computing, "LaunchPad MSP430G2", Available:

<https://embeddedcomputing.weebly.com/launchpad-msp430g2.html> [Accessed: Sep, 8, 2023]

[6] Arduino, "Arduino Motor Shield Rev3", Available:

<https://store.arduino.cc/products/arduino-motor-shield-rev3> [Accessed: Sep 15, 2023]

APPENDICES

Figure 1 - Subsystems and Interconnections

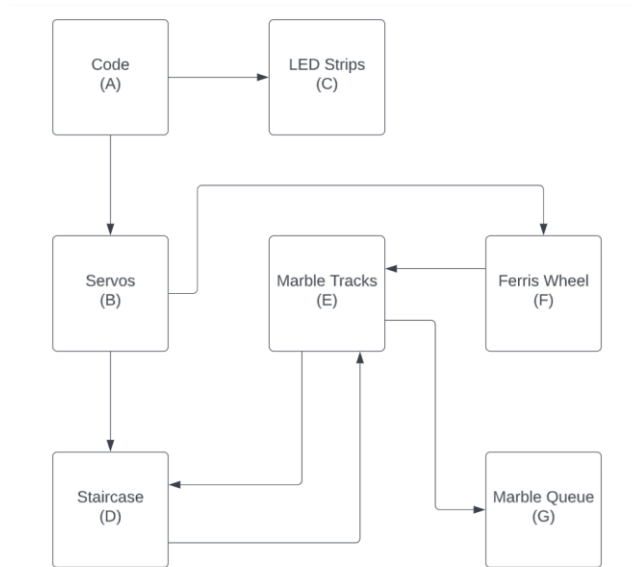


Table 19 - Comms Team DSM

	LED	Sensor	Motor	Arduino	Wiring	Code
LED		x		x	x	x
Sensor	x				x	
Motor		x		x	x	x
Arduino					x	x
Wiring				x		
Code				x		

Figure 2 - Marble Machine CAD model (by Joshua Ramos)

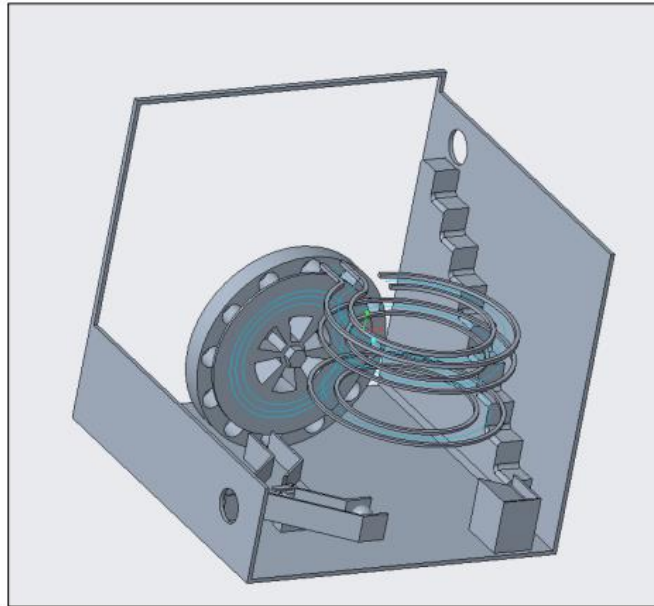


Figure 3 - Electronic Component Overview

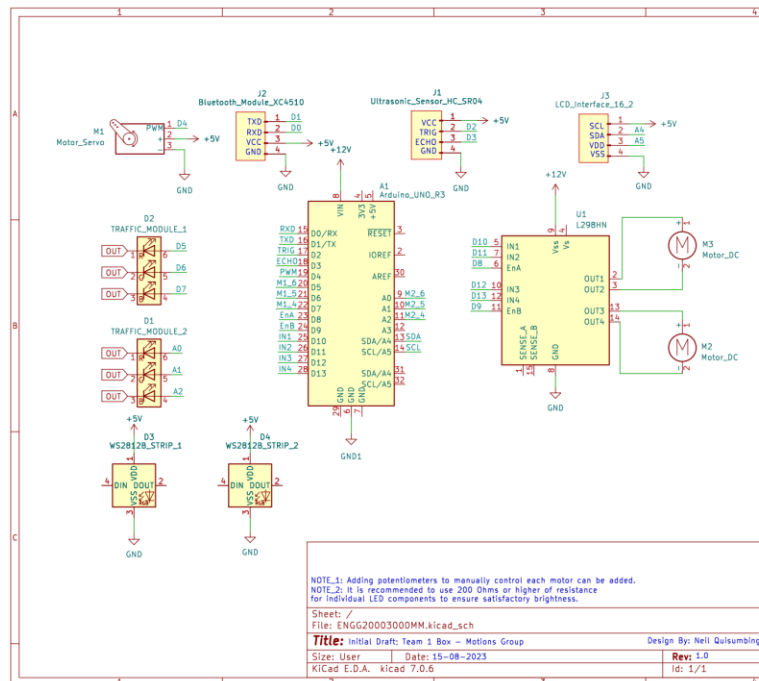


Figure 4 - Team Gantt Chart

Project Gantt Chart

PROJECT TITLE	Massive Marvelous Multiple Marble Machine
TEAM	Team 1: Motions 6, Communications 1 & Structures 3
CUSTOMER	Rex Di Bona
DATE	Friday, 18 August 2023

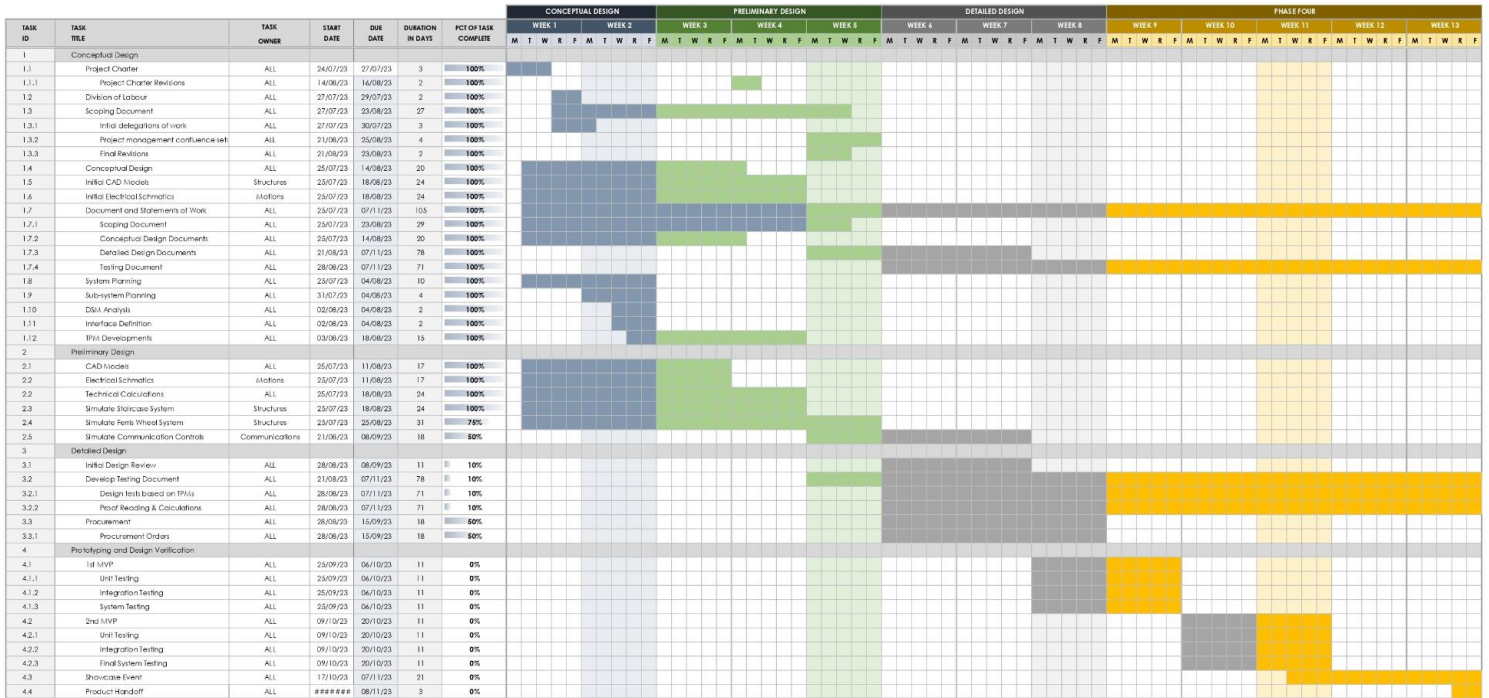


Figure 5 - House of Quality

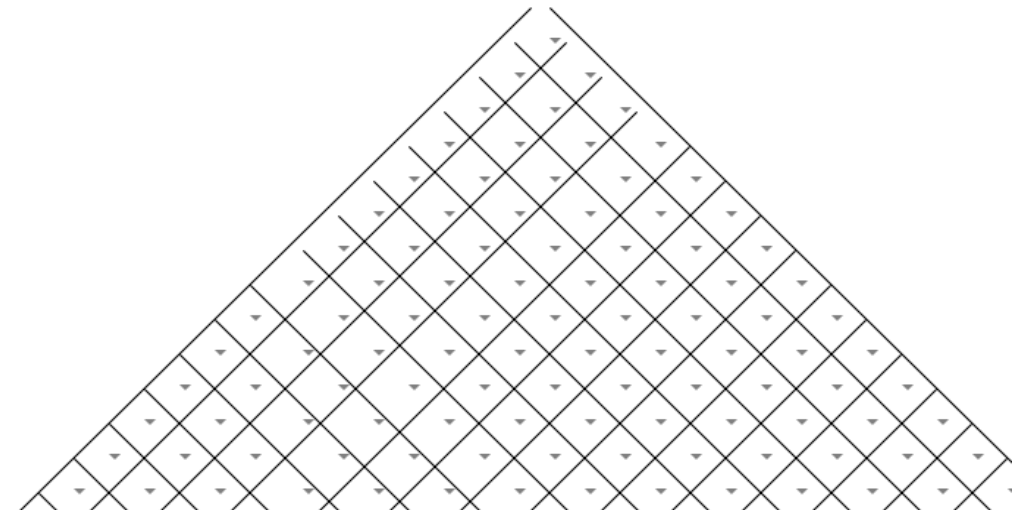
																
		Column #	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Row #	Demanded Quality (a.k.a. "Customer Requirements" or "Whats")	Quality Characteristics (a.k.a. "Functional Requirements" or "Hows")														
1	Have the external dimension of 250mm x 250mm x 250mm	Design the cube with external dimensions precisely measuring 250mm x 250mm x 250mm.	⊙				▲			⊙	⊙				⊙	⊙
2	Must be able to be mounted to the main sculpture using 4 mounting points	Ensure the cube is able to be mounted to the main sculpture using 4 mounting points	⊙	⊙			▲			⊙			▲	▲		
3	Provide visual indication using Led or any other lighting system approved by customers	Implement a lighting system, such as LEDs, approved by the customers, to provide clear and visually distinctive indications.			⊙	⊙										
4	External services (power, telecommunications, etc.) must be done through a service port	Design a service port through which external services like power and telecommunications can be routed into the cube while maintaining a secure and weather-resistant connection.			▲	⊙										
5	Distance between other cubes on the sculpture must be 15mm	Ensure a minimum gap of 15mm between the cube and any adjacent cubes on the sculpture	⊙				⊙			▲						
6	Must be able to accept steel or other lighter material marbles with diameter of 16mm	Design the cube's marble acceptance mechanism to accommodate steel or lighter material marbles with a diameter of 16mm						⊙	▲				▲		⊙	▲
7	Must accept marbles that are spaced no closer than 1 second apart through the opening port	Design the cube so that it accepts marbles spaced at least 1 second apart through the opening port						▲	⊙							
8	Must be manufactured from 3mm MDF for the back and side pieces, with the additional requirement that the front surface must be transparent.	Construct the cube's back and side pieces from 3mm MDF (Medium Density Fiberboard), ensuring the front surface is transparent	⊙				▲			⊙	⊙					
9	The front side of the cube must be able to remove for maintenance purposes	Design the cube with a removable front side to facilitate easy maintenance access	▲							⊙	⊙					
10	The cube must not use any form liquid or gas in its operation	Ensure the cube's functionality does not involve the use of any liquids or gases									⊙					
11	Must accept a marble on through an input opening on a surface other than the top surface of the cube.	Create an input opening on a surface other than the top						▲	▲				⊙	▲	▲	⊙
12	Must expel all marbles out through an exit opening on a different surface to that where the marble entered	Create an exit opening on a different surface to the input											⊙	⊙	⊙	⊙
13	Must not deliberately expel marbles at a speed significantly higher than that attributable to gravity alone	Design the cube in such a way that expels marbles at a controlled speed, avoiding speeds significantly higher than gravity's influence												⊙	⊙	⊙
14	Must control the marble in such a fashion that the bottom of the exit opening is at least 100mm higher than the top of the input opening	Ensure the bottom of the exit opening is at least 100mm higher than the top of the input opening	▲										▲	⊙	⊙	⊙

Figure 6 - Conceptual Code Design Structure

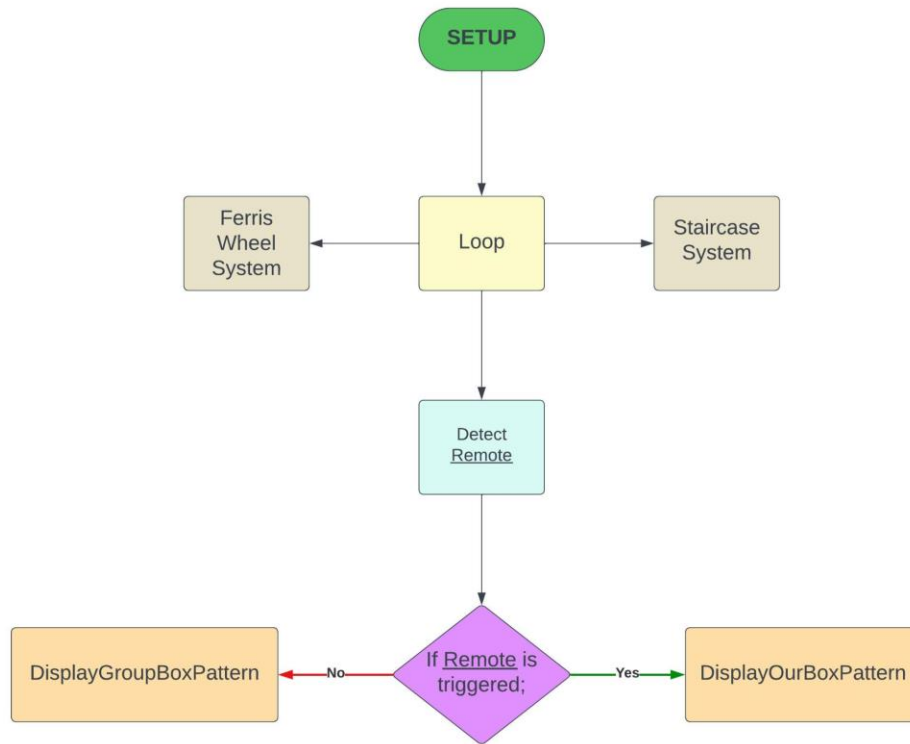


Figure 7 - Conceptual Design Pseudocode

```
int ferrisWheelSpeed
int minimumFerrisWheelSpeed

int staircaseSpeed
int minimumStaircaseSpeed

setup:
  import Arduino libraries
  import other libraries
  initialise pins

loop:
  DetectRemote() // Ben and Quoc
  FerrisWheelMotor() // Irvin and Thomas
  StaircaseMotor() // Irvin and Thomas

DetectRemote: // Ben and Quoc
  if(remotePressed):
    DisplayGroupBoxPattern() // Rumeth, Vikil and Eugene
  else:
    DisplayOurBoxPattern() // Rumeth, Vikil and Eugene

FerrisWheelMotor: // Irvin and Thomas
  run ferrisWheelMotor
  ferrisWheelSpeed = motorSpeed

StaircaseMotor: // Irvin and Thomas
  run staircaseMotor
  staircaseSpeed = motorSpeed

DisplayOurBoxPattern: // Rumeth, Vikil and Eugene
  if(ferrisWheelSpeed < minimumFerrisWheelSpeed):
    display red lights on ferris wheel
  else:
    display repeating interesting patterns
  if(staircaseSpeed < minimumStaircaseSpeed):
    display red lights on staircase
  else:
    display repeating interesting patterns

DisplayGroupBoxPattern: // Rumeth, Vikil and Eugene
  if(ferrisWheelSpeed < minimumFerrisWheelSpeed):
    display red lights on ferris wheel
  else:
    display group box pattern on ferris wheel
  if(staircaseSpeed < minimumStaircaseSpeed):
    display red lights on staircase
  else:
    display group box pattern on staircase
```

Figure 8 - Final Code Design Structure

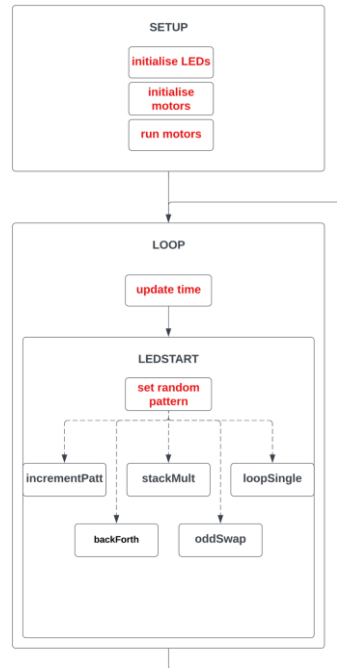


Figure 9 - Position of the Staircase motor inside of the larger Staircase subsystem

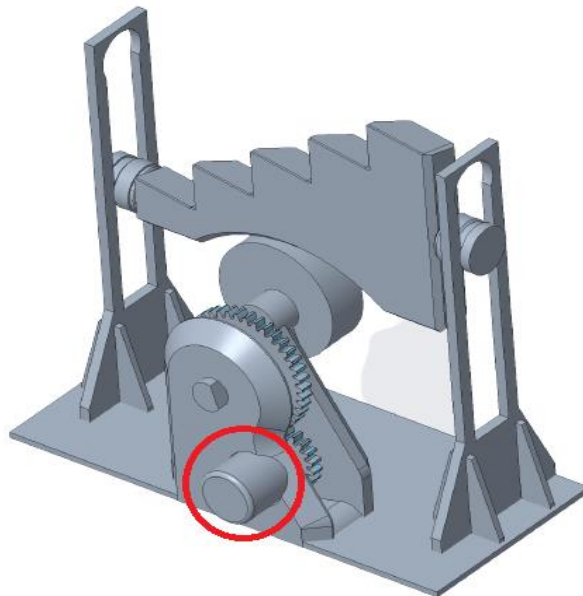


Figure 10 - Position of the Ferris Wheel motor inside of the larger Ferris Wheel subsystem

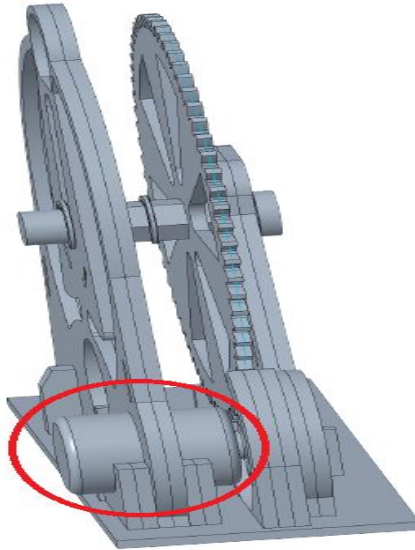


Figure 11 - Sample Code for a LED Pattern

```
void incrementPattern() {  
    if (time >= led_Patt_Lasttriggered + 50) {  
        leds[led_Patt_IncrementPattIndex] = CRGB(random(25, 200), random(25, 200), random(25,  
200));  
        FastLED.show();  
        if(led_Patt_IncrementPattIndex == numLeds-1){  
            fill_solid(leds, numLeds, CRGB(0, 0,0));  
            led_Patt_IncrementPattIndex = -1;  
        }  
        led_Patt_IncrementPattIndex++;  
        led_Patt_Lasttriggered = time;  
    }  
}
```

Figure 12 - a Class Used to Manage the States Across the Two Motors

```
class Motor {
private:
    int motorNumber;
    int speed;
    String status;
    AF_DCMotor motor;
    int triggerOffset;

public:
    int lastTriggerTime;
    int nextTriggerTime;
    int triggerTime;

    Motor(int motorNumber, int triggerTime) : motorNumber(motorNumber), motor(motorNumber),
    triggerTime(triggerTime)
    {
        speed = 0;
        stop();
    }
}
```

Figure 13 - Final Code

```
#include <FastLED.h>
#include <AFMotor.h>

//Roof
#define DATA_PIN 2
#define NUM_LEDS 23

unsigned long time = 0;
int index = 0;
boolean forwards = true;

CRGB ledsD[NUM_LEDS];

class Motor {
private:
    int motorNumber;
    int speed;
    String status;
    AF_DCMotor motor;
```

```

    int triggerOffset;

public:
    int lastTriggerTime;
    int nextTriggerTime;
    int triggerTime;

    Motor(int motorNumber, int triggerTime) : motorNumber(motorNumber), motor(motorNumber),
    triggerTime(triggerTime)
    {
        speed = 0;
        stop();
    }

    // Function to stop a specified motor
    void stop()
    {
        motor.run(RELEASE);
        status = "STOPPED";
    }

    // Function to get the status of a specified motor
    String getStatus()
    {
        return status;
    }

    // Function to set the speed of a a specified motor
    void setSpeed(int speed)
    {
        motor.setSpeed(speed);
    }

    // Function to set a specified motor to rotate forwards at a specified speed
    void forward()
    {
        motor.run(FORWARD);
        status = "MOVING";
    }

    // Function to set a specified motor to rotate backwards at a specified speed
    void backward()
    {
        motor.run(BACKWARD);
    }

```



```

    status = "MOVING";
}

// Function to get the trigger offset value of a specified motor
int offset()
{
    return triggerOffset;
}

// Function to calculate the next trigger time of a specified motor
void calcNextTriggerTime(int lastTriggerTime)
{
    nextTriggerTime = lastTriggerTime + triggerOffset;
}

int getMotorNumber()
{
    return motorNumber;
}
};

class LEDStrip {
public:
    LEDStrip(int dataPin, int numLeds, CRGB* led) : dataPin(dataPin), numLeds(numLeds) {
        leds = led;
    }

    void incrementPattern() {
        if (time >= led_Patt_Lasttriggered + 75) {
            leds[led_Patt_IncrementPattIndex] = CRGB(random(25, 100), random(175, 220), random(25,
100));
            FastLED.show();
            if(led_Patt_IncrementPattIndex == numLeds-1){
                fill_solid(leds, numLeds, CRGB(0, 0,0));
                led_Patt_IncrementPattIndex = -1;
            }
            led_Patt_IncrementPattIndex++;
            led_Patt_Lasttriggered = time;
        }
    }

    void forth(int pix) {
        if (time >= led_Patt_Lasttriggered + 75) {
            fill_solid(leds, numLeds, CRGB(0, 0,0));

```

```

    for(int i=0 + led_Patt_IncrementPattIndex ; i < pix + led_Patt_IncrementPattIndex; i++){
        leds[(led_Patt_IncrementPattIndex+i)%10] = CRGB(random(25, 200), random(25, 200),
random(25, 200));
    }
    if(led_Patt_IncrementPattIndex == numLeds-1){
        fill_solid(leds, numLeds, CRGB(0, 0,0));
        led_Patt_IncrementPattIndex = -1;
    }
    led_Patt_IncrementPattIndex++;
    led_Patt_Lasttriggered = time;
}
}

void incrementReversePattern() {
    if (time >= led_Patt_Lasttriggered + 75) {
        leds[numLeds-1-led_Patt_IncrementPattIndex] = CRGB(random(25, 200), random(25, 200),
random(25, 200));
        FastLED.show();
        if(led_Patt_IncrementPattIndex == numLeds-1){
            fill_solid(leds, numLeds, CRGB(0, 0,0));
            led_Patt_IncrementPattIndex = -1;
        }
        led_Patt_IncrementPattIndex++;
        led_Patt_Lasttriggered = time;
    }
}

void oddSwap(){
    if (time >= led_Patt_Lasttriggered + 500) {
        fill_solid(leds, numLeds, CRGB(0, 0,0));
        if(oddSwapFlag){
            for(int i=0; i < numLeds ; i += 2){
                leds[i] = CRGB(random(25, 200), random(25, 200), random(25, 200));
            }
            FastLED.show();
            oddSwapFlag = false;
        }
        else if(!oddSwapFlag){
            for(int i=1; i < numLeds ; i += 2){
                leds[i] = CRGB(random(25, 200), random(25, 200), random(25, 200));
            }
            FastLED.show();
        }
    }
}

```

```

        oddSwapFlag = true;
    }
    led_Patt_Lasttriggered = time;
}
}

void backForth(int pix){
    if (time >= led_Patt_Lasttriggered + 100) {
        fill_solid(leds, numLeds, CRGB(0, 0,0));
        for(int i =0 ; i < pix; i++){
            leds[led_Patt_IncrementPattIndex+i] = CRGB(random(25, 200), random(25, 200),
random(25, 200));
        }
        FastLED.show();
        led_Patt_Lasttriggered = time;
        if(hasReachEndBF(pix)){
            forwards = false;
        }
        else if (hasReachStartBF(pix)){
            forwards = true;
        }
        if (forwards == true) {
            led_Patt_IncrementPattIndex++;
        }
        else {
            led_Patt_IncrementPattIndex--;
        }
    }
}

void backForthTest(int pix){
    if (time >= led_Patt_Lasttriggered + 100) {
        fill_solid(leds, numLeds, CRGB(0, 0,0));
        for(int i =0 ; i < pix; i++){
            leds[led_Patt_IncrementPattIndex+i] = CRGB(random(25, 200), random(25, 200),
random(25, 200));
        }
        FastLED.show();
        led_Patt_Lasttriggered = time;
        if(hasReachEndBF(pix)){
            forwards = false;
        }
    }
}

```

```

bool hasReachEndBF(int pix){
  if(led_Patt_IncrementPattIndex == (numLeds - pix)){
    return true;
  }
  else {
    return false;
  }
}

```

```

bool hasReachStartBF(int pix){
  if(led_Patt_IncrementPattIndex == 0){
    return true;
  }
  else {
    return false;
  }
}

```

private:

```

int dataPin;
int numLeds;
int cycleCount;
bool oddSwapFlag = false;
CRGB* leds;
unsigned long led_Patt_Lasttriggered = 0;
unsigned long led_Patt_IncrementPattIndex = 0;
unsigned long LEDLastTrigger = 0;
};

```

```

LEDStrip stripD(0, NUM_LEDSD, ledsD);

```

```

Motor m1(1, 500);

```

```

Motor m2(2, 500);

```

```

void setup() {

```

```

  Serial.begin(9600);

```

```

  index = 0;

```

```

  FastLED.addLeds<WS2812, DATA_PIN, GRB>(ledsD, NUM_LEDSD);

```

```

  FastLED.setBrightness(100);

```

```

  FastLED.setMaxPowerInVoltsAndMilliamps(2, 200);

```

```
m1.setSpeed(100);  
m1.forward();  
  
m2.setSpeed(100);  
m2.backward();  
}  
  
void loop() {  
  time = millis();  
  
  stripD.incrementPattern();  
}
```