ENGG2000 and ENGG3000 SPINE Engineering Project Massive Marvellous Multiple Marble Machine T4C5 - T4 Box 4 Marble Machine Software - Communications team

By: Daniel Johnston, Eli Mana, Mohammad Akhlaquzzaman Ferdous, Quoc Huy Pham & Zarraf Khan

30 October 2022

Contents

List of Tables

List of Figures

List of source codes

Revision History

Table 0.1: Document Revision History.

Version	**Date**	**Comment**	**Name**
4.0	30/10/2022	Testing final document	Group
3.0	09/10/2022	Design final document	Group
2.0	04/09/2022	Scoping and Requirements final document	Group
1.0	24/08/2022	Scoping and Requirements draft document	Group
0.0	03/08/2022	Created document	Eli Mana

Division of Labour and Workload Acknowledgement

Table 0.2: Division of Labour and Overall Contribution

Member Name	**SID**	**Cohort**	**Sections responsible**	**Overall**
Daniel Johnston	47095733	ENGG2000	• Terms & Definitions • Introduction • Scope • Deliverables • Lighting • Integration Tests	20%
Eli Mana	44907311	ENGG3000	• Professionalism • Interface Requirements • Sign Offs • Inclusions & Exclusions • Stepper Motor • Diagram Management • General Test Plan	20%
Mohammad Akhlaquzzaman Ferdous	46150927	ENGG3000	• Problem Definition • Assumptions • Conceptual Design • Programming Decisions • Unit Tests	20%
Quoc Huy Pham	46293175	ENGG2000	• Constraints • Detection • Servo Motors • Subsystem Management • System Tests	20%
Zarraf Khan	45949344	ENGG3000	• Functional Requirements • Performance Requirements • Table of Variables • Table of Functions • Design Review	20%

Terms, Definitions, and Abbreviated Terms

Table 0.3: List of Terms, Definitions, and Abbreviated Terms

Terms and Abbreviation	**Definition**
Arduino	Single-board microcontrollers and microcontroller kits for building digital devices.
Arduino Uno	The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. See Appendix A.
Arduino IDE	Software that allows programming of Arduino microcontrollers, including Arduino Uno, Arduino Nano, ESP32 controllers.
C++	General purpose C based language with class support.
Communications	The teams working on the code and communication for the State Machine and Control Box, comprised predominantly of Software Engineers.
Client	Macquarie University Faculty of Science and Engineering.
IEEE	Institute of Electrical and Electronics Engineers.
Interaction Diagram	Emphasize the flow of control and data among the things in the system being modeled.
Motions	The teams working on the subsystems comprising of the motors, sensors and controls of the robot.
MVP	Minimum Viable Product. The bare minimum product to be expected by the client.
Subsystem	A self-contained system within a larger system.
Servo	Short for servomechanism; a mechanical device that outputs increases motion or force from the input; high precision low torque addressable motor.
Structures	The teams working subsystems comprising of the physical structure of the robot.
T4	The entire team that is working on the project as a whole, Team 4.
T4B4, Marble Machine Box, The Cube	The Massive Marvellous Multiple Marble Machine individual box being constructed by T4S5, T4M5, and T4C5 groups.
T4C5	Team 4 Communications 5 subsystem Team. Accountable for the Software Design and programming of T4B4.
T4M5	Team 4 Motions 5 subsystem team. Accountable for the hardware and electronics for T4B4 including the controller, motors, and the placement of the sensors and wiring.
T4S5	Team 4 Structures 5 subsystem team. Accountable for the design, manufacturing and installation of the structural elements of T4B4.

1 Introduction

1.1 Preface

This report has been written by the members of T4C5 (Team 4 Communications 5) as part of an ongoing thirteen week project, with the purpose of informing the engineers, client, and other relevant stakeholders of the scoping and requirements of the Marble Machine Box 4 (T4B4) Software. See Appendix B for the project timeline.

1.2 Problem Introduction

Developing and designing a marble machine is a complex and intricate procedure, requiring the combined efforts of multiple teams spanning various engineering disciplines. For this engineering challenge a structural team, motions team, and communications team were required. The structural team looks after the design and manufacture of the structural elements of the box. The motions team is responsible for the design and manufacture of the mechanically and electronically actuated areas of the box. The communications team is responsible for the communication of the box to other surrounding boxes, as well as the programming of any electronically controlled elements implemented by the motions team. The following document follows the standard IEEE29148:2018 format [1] and details the requirements for the design and manufacture of a marble machine box.

The function of the box is to accept marbles and manipulate them as part of a larger kinematic sculpture in accordance to the given design specifications and constraints. This document details the scope, subsystems, constraints, assumptions, requirements, design specification, and testing of the box. This document also contains previously captured constraints, requirements, and assumptions, as well as relevant diagrams and codebases.

1.3 Scope

This document:

- defines engineering problems and provides an overview of the subsystems;
- specifies the functional requirements of the sculpture;
- specifies interface requirements between the communications subsystems and the other subsystems;
- provides sign-offs by the various engineering teams for the agreed upon subsystem interfaces;
- provides a set of constraints and assumptions for the project;
- specifies the inclusions and exclusions for the subsystem scope;
- outlines deliverables, including the subsystem and software to be developed and relevant documentation;
- provides a conceptual design in the form of a list of functions, flow charts and analysis, including a comparison to alternatives;
- proposes a detailed design outlining specifications in a table of methods, outlining the actors and objects in UML Sequence diagrams and discusses the flow and details of the methods with reference to code;
- provides a test plan, incorporating procedures for testing state machine and overall robot integration, that account for TPM and Requirement traceability.

This document is applicable to:

- the structural engineering team;
- the motions engineering team;
- communications engineering team;
- the client;
- product testers;
- product operators;
- all other relevant stakeholders for the marble machine box.

2 Problem Definition

This Engineering project is to build a marble machine as per the client's requirements. A sculpture will be incorporated with multiple boxes, up to 64, that work together to collect marbles and produce a pleasing kinetic effect. The boxes are to be mounted in specific points of the sculpture and there will be a feeder positioned above that will return the marbles at certain interval.

The project in this documentation is comprised of different subsystems, with several teams working together on their respective subsystems to build the machine. The main goal of this project is to build a marble machine encased in a 250mm*250mm*250mm box that will accept marbles from an opening on the upper surface and expel it from the lower surface of the cube. It will also have a pleasing kinetic rotation with the marbles incorporated with a synchronised LED effect. Structures, Motions and Communications teams are collaborating to design and construct the machine.

The cube with a dimension of 250mm*250mm*250mm will accept the marble with its upper surface and expel it from the lower surface with a minimum interval of 1 second which will be mounted on a specified sculpture. There will be other similar cubes built by other teams mounted on the sculpture and each cube will be able to communicate with other cubes. For the cube managed by T4 Communications 5, T4B4, the communication will be internal and independent. However, future work may include communication with the main grid LED controller. The cube must hold the marble stationary for a minimum of 1 second. Additionally, the marble will go through a pleasing kinetic rotation inside the cube. The cube must be able to detect marbles, and all the marbles must retrace their path for at least 50mm. There will be a feeder positioned above the cube that will return the marble at a certain interval.

The budget provided for this project is AUD $100. All the electronic communication will be controlled by a single Arduino Uno. For the rotation effect we will use Servomechanism and LED lights for visual effect synchronised with the rotation. The cube will detect the marbles through sensors. All the electronics are programmed and controlled by the Arduino. The equipment must be bought from the allowed suppliers only.

The diagram in Figure 2.1 illustrates the subsystems and tasks managed by each team.

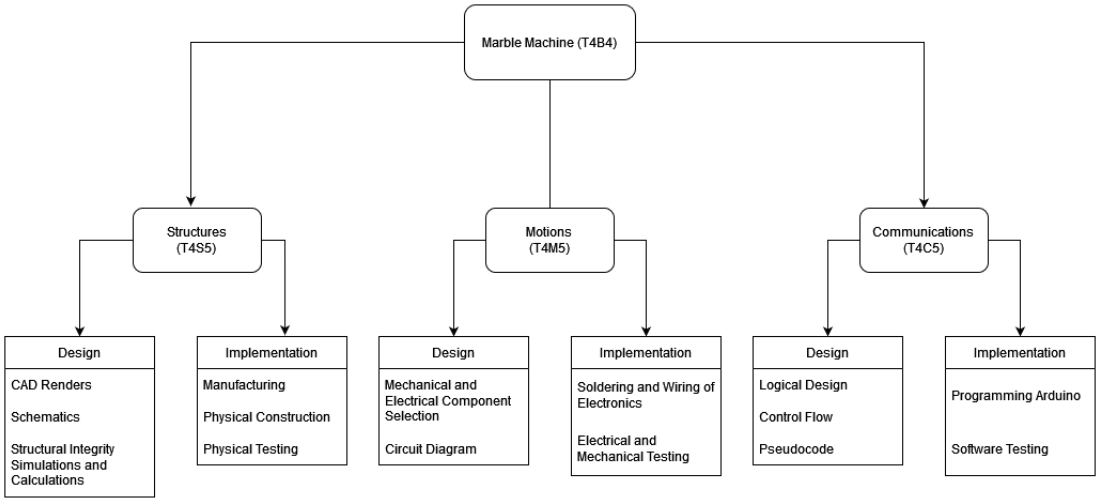


Figure 2.1: Diagram of group subsystems and responsibilities for Communications, Motions and Structures

2.1 Subsystems

To implement this problem, the project is divided among 3 teams with each team responsible for their own subsystems. These teams are Structures, Motions, and Communications.

For T4B4 the structure team T4 Structures 5, motion team T4 Motions 5 and communication team T4 Communications 5 are collaborating together on their respective subsystems to achieve successful project results.

2.1.1 Structures

The Structures team T4 Structures 5 is responsible for looking after the physical design of the cube as well as manufacturing the structural elements of the cube. They are assigned to build the physical box i.e., the cube and all the structural components of the cube.

2.1.2 Motions

The motions team will design and build the mechanical and electronic components of the cube. The motions team T4 Motions 5 will design the mechanical subsystem of the cube. They are responsible for the implementation of motors and sensors and wiring layout. They will also work on the layout design of all the electronic and mechanical parts.

2.1.3 Communications

The communications team are responsible for the software implementations and communications of the cube. T4 Communications 5 will program all the electronic components of the cube. The motors, sensors, LEDs, etc. used in the cube are all controlled by the Arduino Uno and T4 Communications 5 is responsible for programming the components according to the system requirements.

T4 Communications 5 will be responsible for programming the control logic for 3 elements of the Marble Machine Box.

- Marble Detection: Register marbles upon arrival, which will signal the appropriate servomechanisms and lighting animations.
- Movement: Marbles will be expelled along ramp structures and servomechanisms in a satisfying manner.
- Lighting: The cube will integrate dynamic lighting animations to coincide with marble movement.

3 Deliverables

A complete list of deliverables for the project is outlined in Table 3.1, including a list of documents, plans, and any other items that will be delivered. Also included is the final output of the project when completed.

Table 3.1: Deliverables

Deliverable	**Description**	**Format**	**Due Date**
Marble Machine Box (T4B4)	This is the final product. It will comprise of a single box which will house internal mechanisms.	Physical Machine	02/11/2022
Marble Machine Box (T4B4) Code	T8CA will write code for the Marble Machine Box (T4B4) to program the Arduino Uno.	C/C++/INO	30/10/2022
Final Documentation & Compiled Scoping	Requirements, Design, and Testing Documents that track all project progress.	PDF	30/10/2022
Scoping and Requirements Document	This document will illustrate the expected outcome of the project, as well as the main requirments.	PDF	04/09/2022
Design Document	This document will cover the alternative conceptual designs, as well as the finalised design.	PDF	9/10/2022
Testing Document	This document will be comprised of the performance metrics and the appropriate test cases for each.	PDF	30/10/2022
Statement of Work	This extended letter will collate all formative feedback which will be received throughout the course of this project.	PDF	30/10/2022
Project Presentation	A short presentation video (approximately 5 min) presenting the project achievement over the course of the semester.	Video	06/11/2022

4 Constraints and Assumptions

4.1 Constraints

Some constraints need to be considered throughout the process of building the Marble Machine Box, listed in Table 4.1. These constraints apply to the Marble Machine Box and the Marble Return Mechanism based on the given specifications documents [2].

Table 4.1: Constraints

Constraint ID	**Constraint**
T4C5_C01	The overall budget of the Cube must not exceed AUD $100.
T4C5_C02	The external dimension of the Cube must be 250mm x 250mm x 250mm.
T4C5_C03	The cube must accept marbles on the upper surface and output all marbles from the lower surface at the location given.
T4C5_C04	The marble's output rate must not be higher than the attribute of gravity, or in a direction that is neither vertical downward or through a location that is not stated.
T4C5_C05	The cube must be mounted to the structure using four mounting position stated.
T4C5_C06	The cube must provide visual indication using LEDs or other customer-approved lighting systems.
T4C5_C07	External services to the cube such as power or telecommunication must go through a stated service port.
T4C5_C08	The cube's location on the structure must be in the middle of a 265mm x 265mm square, creating a 15mm gap between it and other cubes.
T4C5_C09	The back and sides of the cube must be build using 3mm MDF sheets with the front surface being transparent.
T4C5_C10	The front surface of the cube must not be build upon and easy to remove.
T4C5_C11	The cube must not use any form of liquid, compressed air nor any other form of gasses during its operation.
T4C5_C12	The cube must be designed to accept marbles that are solid steel spheres with the diameter of 16mm.

4.2 Assumptions

Assumptions made for this project are listed in Table 4.2 below.

Table 4.2: Assumptions

Assumption ID	**Assumption**	**Description**
T4C5_A01	All electronics can be controlled by an Arduino	The provided motors, sensors, LED lights, and LCD display can all be controlled and programmed by the Arduino software.
T4C5_A02	Structural integrity	All ramps, funnels, wheels as well as the housing cube itself will have sound structural integrity.
T4C5_A03	Marble timed interval at arrival	Marbles are expected to arrive spaced apart at an interval no less than 1 second.
T4C5_A04	Uniform marble size	All marbles are expected to have the same diameter.

5 Requirements

5.1 Functional Requirements

Table 5.1 lists the functional requirements for the Marble Machine Box.

Table 5.1: Functional Requirements

ID	Dependencies	Requirement Name	Functional Requirement	MVP
T4C5_F01	-	Cube Function	The cube must coordinate mechanisms to produce a pleasing kinetic effect with the marble.	Yes
T4C5_F02	-	Marble Entry and Exit	The cube must allow all steel marbles to pass through the designated openings, with 1 second intervals between each.	Yes
T4C5_F03	-	Marble Traversal	The cube must not obstruct any marbles from exiting it.	Yes
T4C5_F04	-	Change in Direction	All marbles must experience a change in direction in at least 2 dimensions.	Yes
T4C5_F05	-	Marble Detection	The cube will be able to detect marbles at the ends of the tracks.	Yes
T4C5_F06	-	Marble Path Retrace	All marbles must retrace its path for at least 50mm.	Yes
T4C5_F07	-	External Power	External cabling must route through the designated service opening.	Yes
T4C5_F08	-	Main Controller	The cube will utilise an Arduino Uno to function as the control board	Yes
T4C5_F09	T4C5_F04, T4C5_F05	Stationary Marble	All marbles must be held completely stationary for at least 1 second.	Yes
T4C5_F10	T4C5_F04	Marble Count Display	In accordance with T4C5_F04, the cube will incorporate a marble counting feature, and display the results on an LCD.	No
T4C5_F11	-	Marble Count Alarm	A buzzer will play a sound once marble count reaches a predetermined parameter.	No
T4C5_F12	-	Cube Lighting	The cube will utilise LED rings to produce a pleasing light effect.	Yes

Table 5.1 continued from previous page

ID	Dependencies	Requirement Name	Functional Requirement	MVP
T4C5_F13	T4C5_F09	Still Lighting Effect	Lighting theme will coincide with stationary marbles.	Yes
T4C5_F14	-	Marble Retrace	Marbles will be launched 50mm along a ramp after being held stationary for 1 second.	Yes
T4C5_F15	-	Slotted Wheel	Marbles will traverse through into a rotating double-slotted wheel mechanism.	Yes
T4C5_F16	T4C5_F15	Wheel Light Effect	Lighting theme will coincide with rotating wheel.	Yes

5.2 Performance Requirements

Table 5.2 lists the performance requirements and Technical Performance Measures (TPMs) for the Marble Machine Box.

Table 5.2: Performance Requirements

ID	**Dependencies**	**Requirement Name**	**Performance Requirement**	**MVP**
T4C5_P01	TPM_03	Marble Detection	Optical sensors should have a accurate detection rate of 99.95%	Yes
T4C5_P02	T4C5_I03, TPM_01	Wheel Rotational Speed	The motor moving the slotted wheel must maintain the correct revolution rate (per 1 second multiplied by number of slots)	Yes
T4C5_P03	T4C5_I07, TPM_04	Cube Noise Level	Maximum cube noise level must not exceed 60dB	No
T4C5_P04	-	Minimum Stationary Time	Marble must be held stationary by servo for at least 1 second	Yes
T4C5_P05	-	Marble Intervals	Marbles must maintain an interval of 1 second upon exit.	Yes
T4C5_P06	T4C5_P02, TPM_01	LED Visual Accuracy	Synchronisation effect of the LEDs must visually coincide with slotted wheel rotation.	Yes

5.2.1 Technical Performance Measures

Table 5.3 lists the technical performance measures (TPMs) that T4C5 has for the Marble Machine Box.

Table 5.3: Technical Performance Measures (TPMs).

ID	**TPM**	**Value**	**Unit**	**Success Parameter**
TPM_01	Speed of slotted wheel rotation	15rpm	Revolutions per minutes	Slotted wheel rotates at 15rpm.
TPM_02	Slotted Wheel Reliability	99.95%	Percentage (Successful marbles run/unsuccessful marbles run)	Slotted wheel successfully inputs and outputs a marble greater than 99.95% of the time.
TPM_03	Marble detection rate	99.95%	Percentage (Marbles detected / Marbles missed)	Laser successfully detects a marble greater than 99.95% of the time.
TPM_04	Cube Noise Volume	60dB	Decibels	Cube never exceeds 60dB volume during operation.
TPM_05	Memory requirement	2KB	Kilobytes	Must use less than 2K of memory at any one time.
TPM_06	Latency	100ms	Miliseconds	Must operate at less than 100ms latency at all times.
TPM_07	Lifespan	1 month	Months	Must last under constant operation for at least 1 month.
TPM_08	Servo reliability	99.95%	Percentage (actuation success/actuation failed)	Must actuate greater than 99.95% of the time.
TPM_09	LED accuracy	99.95%	Percentage (lighting coincides/ lighting does not coincide)	Lighting must coincide with rotation of the slotted wheel greater than 99.95% of the time.
TPM_10	Runtime	15 minutes	Minutes	The system must be stable under load conditions for the specified period of time.

5.3 Interface Requirements

Table 5.4 lists the interface requirements that T4C5 has for the Marble Machine Box.

Table 5.4: Interface Requirements

Interface ID	**Requirements Name**	**Interface Requirement**	**MVP**
T4C5_I01	Marble IR Detection	The Arduino Uno shall be connected to two IR detection sensors to detect when a marble has arrived in front of each of the two respective servo motors.	Yes
T4C5_I02	Marble Launching Servo Motors	The Arduino Uno shall be connected to two servo motors that will launch individual marbles backwards along a track.	Yes
T4C5_I03	Wheel Rotating Motor	The Arduino Uno shall be connected to a stepper motor that rotates a slotted wheel with marbles at a rate in accordance with T4C5_P02.	Yes
T4C5_I04	Individually Addressable Led Strips	The Arduino Uno shall be connected to 4 individually addressable LED strips that can be controlled through both the Arduino FastLED library and the Adafruit NeoPixel library.	Yes
T4C5_I05	Marble Counter IR Detection	The Arduino Uno shall be connected to an IR detection sensor to detect when a marble has entered through the box opening and count it.	No
T4C5_I06	Marble Counter LCD	The Arduino Uno shall be connected to an LCD segment display to display the running count of mables that have entered the box.	No
T4C5_I07	Marble Counter Buzzer	The Controller shall be connected to a piezo buzzer that plays a sound when a user specified number of marbles are counted entering the box, after which the counter is reset.	No
T4C5_I08	Inter-Box LED Grid Control	The Arduino Uno shall send a signal with a grid location and a command string asynchronously whenever the box wants to update the behaviour of the grid LED strips.	No
T4C5_I09	Physical Access to Arduino Uno	The Arduino Uno shall have ease of physical access without the need to remove permanent structures, for the purposes of programming and maintenance.	Yes

5.3.1 Interface Sign-offs

Sign-offs by the collaborating Communications, Motions and Structures groups on shared interface requirements. Signature in Table 5.5 indicates acceptance of requirements.

Table 5.5: Interface Sign-Offs with Contact, Signature and Date.

Interface ID(s)	**Group**	**Name**	**Contact Details**	**Date Signed**
T4C5_I08	T4C2	Nader Awad	nader.awad@students.mq.edu.au	17/08/2022
T4C5_I01 - T4C5_I07	T4M5	Daniel Hunnam	daniel.hunnam@students.mq.edu.au	24/08/2022
T4C5_I09	T4S5	Alexey Korolkov	alexey.korolkov@students.mq.edu.au	24/08/2022

6 Inclusions and Exclusions

6.1 Inclusions

Table 6.1 shows the items of the project to be included in the scope of T4C5.

Table 6.1: Inclusions

Inclusion ID	Inclusion	Description	MVP
T4C5_IN_01	Arduino Uno	Use Arduino Uno to control the marble machine electronics.	Yes
T4C5_IN_02	Incoming Marble Detection	Incoming marbles will be detected and tracked by the marble machine box system.	No
T4C5_IN_03	Dynamic Lighting	Lighting will operate dynamically in accordance with the current operation and state of the marble machine box T4B4.	Yes

6.2 Exclusions

Table 6.2 shows the items of the project to be excluded in the scope of T4C5.

Table 6.2: Exclusions

Exclusion ID	Exclusion	Description
T4C5_EX_01	Software security	Security of the controller logic software will not be guaranteed.
T4C5_EX_02	Software updates	Future software updates will not be guaranteed.
T4C5_EX_03	Physical structure	T4C5 will not be responsible for the design or building of the physical structure of the marble machine box and its physical components.
T4C5_EX_04	Electronics	T4C5 will not be responsible for the design and assembly of the electronic systems of the marble machine box.
T4C5_EX_05	Inter-box communication	There will be no communication between the box T4B4 and other boxes created by groups from T4 or other teams. All communications created by T4C5 for the box T4B4 will been internal, and the box T4B4 will behave independently. The box T4B4 will interact with adjacent boxes in a modular fashion, as outlined by the project specifications and interface requirement T4C5_I08.

7 Conceptual Design

The conceptual design for this subsystem includes building a marble machine and designing the machine according to the requirements and constraints of the client.

Figure 7.1: CAD Render of Physical Box Design (Provided by T4M5 and T4S5)

As seen in Figure 7.2, the Marble Machine Box control code includes several movement, lighting, and detection elements. These elements update and run in real-time and operate concurrently. These elements, as seen in Figure 7.1, include stepper motor control for the marble wheel, LED animation control for the lighting, IR sensor reading for the marble detection, and servo control for the marble hold and release track.

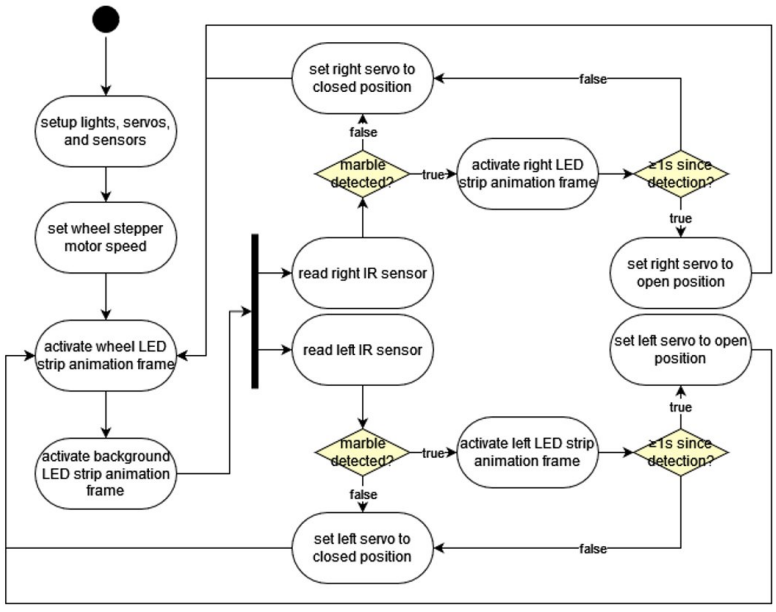


Figure 7.2: Activity diagram of main control flow

7.1 Design Alternatives

While designing the subsystems, they needed to be modified and revised a number of times to both maintain the constraints and requirements and optimise the system. The alternative designs are narrated below:

7.1.1 Alternative Design 1 - Light Sensor and Ultrasonic Sensor

In the initial design proposed for marble detection, light sensors and ultrasonic sensors were thought to be used. Instead, Infrared (IR) break beam sensors are used to detect the marbles. According to the requirements, the marbles must remain stationary for 1-second interval and have them retrace along the track path, and the cube must detect the marble at the end of the track. 2 LED IR break beam sensors are used to detect the marble as the emitter break beam sensor emits IR beam light and the receiver break beam is sensitive to that light beam. When any thing passes through the sensors the light beam breaks down and will notify about it. Thus the marbles will be detected. If instead of IR beam sensor light sensors and ultrasonic sensors were used then it would not have precisely detect the marble. The light sensors and ultrasonic sensors will detect any light rays and return the signal. It may or may not be a marble. Similarly, the ultrasonic sensor detects any obstacles and then return the signal which also can not give the marble count precisely. The IR break beam sensor is able to precisely detect the marble and is also budget friendly which is why The IR break beam sensors were used instead of the light and ultrasonic sensors.

7.1.2 Alternative 2 - Continuous Rotation Servomotor

An earlier alternative for the marble wheel rotation was a continuous rotation servomotor to meet T4C5_F15. This would be controlled through the Arduino Servo Library to rotate the marble wheel at a precise speed to drop the marbles at a 1-second interval. However, budget-friendly continuous rotation servomotors are not designed for long-term non-stop rotation. Ultimately, they were replaced with a stepper motor and stepper motor driver controlled through the Arduino Stepper Library. A stepper motor was more suitable due to it being more robust for long-term continuous operation and it could also be controlled more precisely through discrete steps.

7.1.3 Alternative 3 - Timing Using Delay()

An early Arduino programming design option was to use delay() calls as a means of timing various control operations in the code. For example, the servo opening and closing logic would be timed with a 1 second delay between actions using delay(1000) to satisfy **??**. However using delay() in the code introduces several problems. It interrupts other portions of the code from executing, therefore preventing concurrent execution of different functions. This prevents real-time detection from IR sensors during a delay(). Therefore the millis() function was utilised instead. This was used along with a comparison against timestamps for timing. The way this is implemented for the IR sensors and servos can be seen in Figure 8.2.

8 Detailed Design

8.1 List of Variables

Table 8.1 outlines the significant global variables predefined in the final design.

Table 8.1: Significant Global Variables

Variable	**Data Type**	**Description**
stepsPerRevolution	const int	Motor steps for one revolution (64)
stepsToDropMarble	const int	Steps per revolution / number of wheel slots
stepFrequency	const int	revolution frequency / steps to drop marble
currMillis	unsigned long	Saves current time
lastStepTime	unsigned long	Saves last time Stepper motor rotates
lastLEDUpdate	unsigned long	Saves last time strip is updated
lastMillisRing	unsigned long	Saves last time LED ring is updated
lastMillisExterior	unsigned long	Saves last time T4C5_FUN_12 is called
lastMillisTraffic	unsigned long	Saves last time T4C5_FUN_12 is called
marble1Detected	unsigned long	IR Sensor 1 status
marble2Detected	unsigned long	IR Sensor 2 status
servo1	Adafruit_TiCoServo	An object to control the first servomotor.
servo2	Adafruit_TiCoServo	An object to control the second servomotor.
exteriorStrip	Adafruit_NeoPixel	An Adafruit_NeoPixel object to control the LED strip on the exterior of the box.
trafficStrip1	Adafruit_NeoPixel	An Adafruit_NeoPixel object to control the LED strip for the first traffic light.
trafficStrip2	Adafruit_NeoPixel	An Adafruit_NeoPixel object to control the LED strip for the second traffic light.
ringStrip	Adafruit_NeoPixel	An Adafruit_NeoPixel object to control the LED strip for the ring above the marble wheel.

8.2 List of Functions

Table 8.2 lists all of the functions in the final design.

Table 8.2: Functions

Function ID	Signature	Description	Requirements	Input	Output
T4C5_FUN_01	void rotateMarbleWheel()	Rotates the marble wheel 1 step	T4C5_F04, T4C5_F15, T4C5_F16, T4C5_C03, T4C5_C12	Current timestamp and frequency interval between steps	Sets wheel motor to rotate 1 step, and updates the current timestamp
T4C5_FUN_02	bool detectIRSensor1()	Checks if marble is detected, and sets marble1Detected to the time it takes to be detected	T4C5_F05, T4C5_F09, T4C5_F13	Data read from break beam sensor pin, and preset interval time for marble to reach servos	Boolean indicator if marble is detected, and updates the current timestamp
T4C5_FUN_03	bool detectIRSensor2()	Checks if marble is detected, and sets marble2Detected to the time it takes to be detected	T4C5_F05, T4C5_F09, T4C5_F13	Data read from break beam sensor pin, and preset interval time for marble to reach servos	Boolean indicator if marble is detected, and updates the current timestamp
T4C5_FUN_04	void servo1Swing()	Sets servo on open or close position	T4C5_F04, T4C5_F06, T4C5_F09, T4C5_F13, T4C5_C03, T4C5_C12	Sensor status, and preset interval time for marble to reach servos	Writes angle values to the servo to set the mechanism accordingly
T4C5_FUN_05	void servo2Swing()	Sets servo on open or close position	T4C5_F04, T4C5_F06, T4C5_F09, T4C5_F13, T4C5_C03, T4C5_C12	Sensor status, and preset interval time for marble to reach servos	Writes angle values to the servo to set the mechanism accordingly
T4C5_FUN_06	void LEDInit()	Initialises the LED strip data pins	T4C5_C06, T4C5_F12	N/A	Sets data pins for Neopixel output, and initialises all pixels as off
T4C5_FUN_07	void LEDRingUpdate()	Updates animation frame for the ring LED strip	T4C5_C06, T4C5_F12, T4C5_F16	Current timestamp	Sets ring lights to red configured with each interval

Table 8.2 continued from previous page

Function ID	**Signature**	**Description**	**Requirements**	**Input**	**Output**
T4C5_FUN_08	void LEDTrafficUpdate(int stage, int light)	Updates traffic light animation based on the current frame and strip used	T4C5_C06, T4C5_F12, T4C5_F16	Current stage of lights (stage), and which LED strip to control (light)	Shifts the colour based on the current stage, and updates stage.
T4C5_FUN_09	void setTrafficDist (uint32_t color, int n, int light)	Triggers T4C5_FUN_10, or T4C5_FUN_11, depending on strip	T4C5_C06, T4C5_F12, T4C5_F16	Preset uint32_t colour value (color), stage status (n), and LED strip	Calls setTraffic1 or setTraffic2 depending on input
T4C5_FUN_10	void setTraffic1 (uint32_t color, int n)	Sets pixel colour for trafficStrip1	T4C5_C06, T4C5_F12, T4C5_F16	Preset uint32_t colour value (color), stage status (n)	Sets new colour value to LED's and send data to strip
T4C5_FUN_11	void setTraffic2 (uint32_t color, int n)	Sets pixel colour for trafficStrip2	T4C5_C06, T4C5_F12, T4C5_F16	Preset uint32_t colour value (color), stage status (n)	Sets new colour value to LED's and send data to strip
T4C5_FUN_12	void LEDExteriorUpdate()	Updates animation frame for the LED strip on the exterior	T4C5_C06, T4C5_F12	Current timestamp	Sets Exterior LED's to empty based and saves current timestamp

8.3 Marble Machine Box Control

T4C5 were responsible for designing the main control logic for the 3 elements as mentioned prior.

- Marble Detection
- Movement
- Lighting

Considering this aspect, the final design was comprised of numerous functions to manage the control logic for each element.

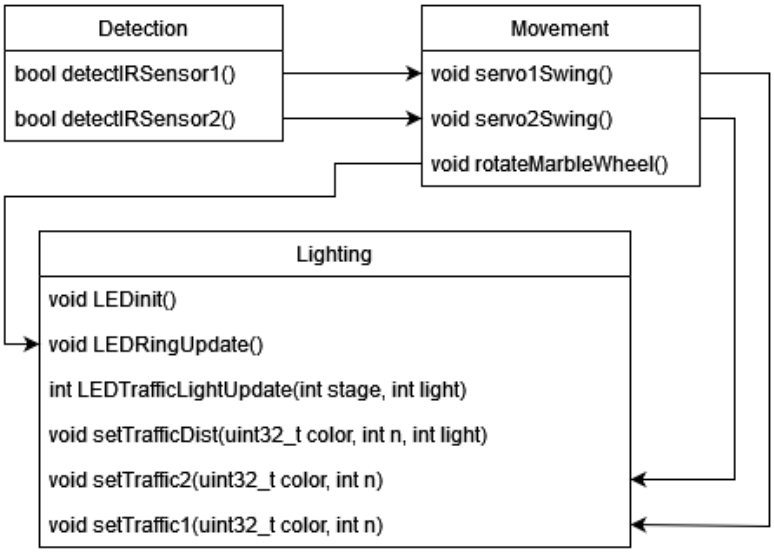


Figure 8.1: UML Class diagram of final solution

8.3.1 Marble Detection

There are two main detection elements in the Marble Machine Box. These detection elements are both 3mm LED IR Break Beam Sensors [3]. The circuit diagram for this can be seen in Appendix D. As of now, these satisfy the marble detection requirement (T4C5 F05), and through dependencies, marble kinetic effects as well as the corresponding lighting effects (T4C5_F09, T4C5_F13, T4C5_F14,T4C5_F16).

Infrared (IR) break-beam sensors are used to detect motion. They function using an IR emitter side and a receiver side sensitive to that light. When something intercepts the two, and breaks the beam, the receiver provides a digital signal.

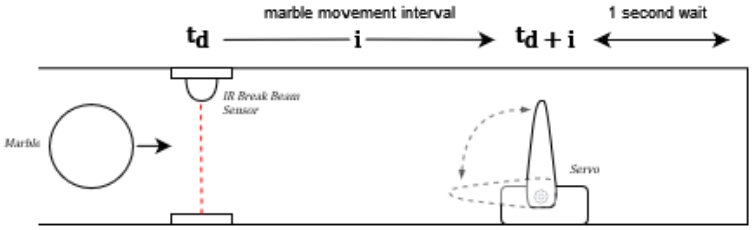


Figure 8.2: Timing diagram for marble detection and hold track

Each sensor is placed before its respective servomotor on each of the 2 tracks after the marble splitter at the opening of the Marble Machine Box. The functions detectIRSensor1() (T4C5_FUN_02) and detectIRSensor2() (T4C5_FUN_03) read the digital pins to check if the IR beam is intercepted, record the millisecond timestamp to marble1Detected and marble1Detected respectively, and then return a boolean value if the marble was recently detected within its travel interval to the servo. This timing process can be seen in Figure 8.2

8.3.2 Marble Movement

8.3.2.1 Servomechanism

In the Marble Machine Box, two servomechanisms are used; Both are 9g micro servo motors [4]. The circuit diagram for this can be seen in Appendix D. This logic involves the functions servo1Swing() (T4C5_FUN_04) and servo2Swing() (T4C5_FUN_05).

Each servo motor is placed respectively on each ramp structure after the splitter and the IR sensor, right before leading to the marble wheel at the bottom of the Marble Machine Box. Marble detection information retrieved from Infrared break-beam sensors is used to adjust the movement of each servo motor correspondingly as seen in Figure 8.2. Initially, the servo motors start at the close position. When the Infrared break-beam sensor corresponding to each servo motor detects marble motion, the servo motor will change its place from close to open to accept the marble. After an amount of time, the servo will then return to its close position. This satisfies the stationary marble and marble retrace requirements (T4C5 F09, T4C5 F14).

8.3.2.2 Stepper Motor Control

In the Marble Machine Box, one 5V geared stepper motor is used [5]). The circuit diagram for this can be seen in Appendix D. The stepper motor is placed at the bottom of the Marble Machine Box, fixed with the marble wheel as seen in Figure 8.3. The Stepper motor rotates the marble wheel at a stated speed synchronising with the marble drop frequency of the Marble Machine, which will lead marbles in the marble wheel to the exit of the Marble Machine Box. This ensures that all marbles are allowed to exit the designated opening. This adheres to the requirements (T4C5 F02 and T4C5 F15) and the constraint T4C5 C03.

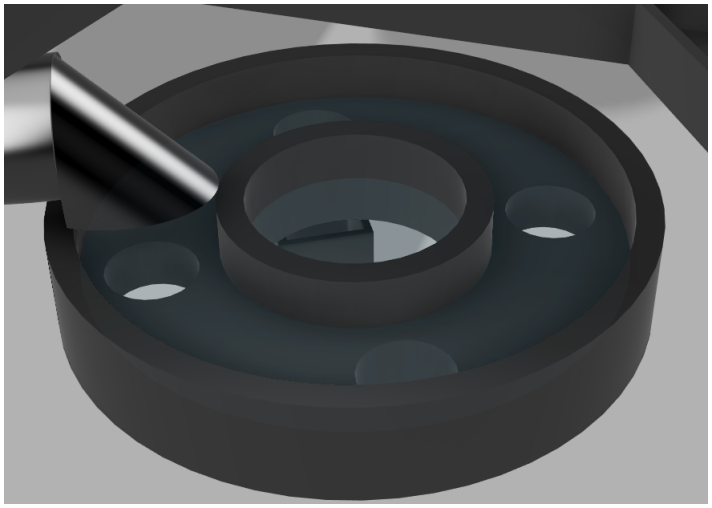


Figure 8.3: CAD Render of Physical Marble Wheel Design (Provided by T4M5 and T4S5)

8.3.3 Lighting

8.3.3.1 Neopixel Initialisation function

The NeoPixel LED strips are wired to the Arduino in 4 distinct strips. The circuit diagram for this can be seen in Appendix D. As such, four distinct Neopixel objects are defined as follows:

- a 12 led long strip on digital pin 10 called exteriorStrip.
- a 3 led long strip on digital pin 9 called trafficStrip1.
- a 3 led long strip on digital pin 8 called trafficStrip2.
- a 12 led long strip on digital pin 7 called ringStrip.
- begin() function starts data output to the strips.
- show() function sets the red, green, and blue values to 0 to prevent retained information on system restart.

8.3.3.2 Ring Lighting

The ring lighting blinks red every second as a placeholder animation, while further lighting animation can be created if required. It utilises the same real-time delay functionality as the Servo timing, using current runtime Millisecond, compared to a saved timing, to calculate the exact time between runs.

8.3.3.3 Traffic Light lighting

The function LEDTrafficLightUpdate() creates 32-bit int variables representing colours, which are created as required. The switch/case iterates through the three stages of the traffic light (Red, Orange, and Green) and returns what stage the traffic light is up to. setTrafficDist() distributes the values from LEDTrafficLightUpdate() into either setTraffic1() or setTraffic2(), and exists to reduce computational load to help prevent lag during concurrent operation with other real-time

functions. setTraffic1() and setTraffic2() are operationally identical, and push the colours from LEDTrafficLightUpdate() to the strip.

8.3.3.4 Exterior Lighting

The Exterior strip uses animation framework, and animation for the exterior strip may be controlled externally as part of a collection of multiple boxes or individually via set animation frames.

8.4 Programming Decisions

8.4.1 Language

Arduino IDE utilises C/C++ syntax, so this was the primary language used to program the functions. As such, T4C5 used a combination of VS Code and the Arduino IDE to write and compile the current Arduino sketch.

8.4.2 Code Style

Overall, T4C5 had decided to house the current implementation under a single Arduino sketch, and refactor the code into several functions to promote readability and manageability. This point was specifically brought attention to, as it would significantly aid the testing process. Similarly, all constant values were predetermined at the top as global definitions. As such, the code standards included:

- 2 space indentation
- Limit lines to 85 characters
- Meaningful variable names
- Descriptive global constants (no “magic numbers”)
- Descriptive code commenting

8.4.3 Libraries

Table 8.3: Utilised libraries

Library	**Purpose**
Adafruit_TiCoServo.h	Modified servo control library that allows for both servo motors and Neopixels to be used on the same Arduino.
CheapStepper	A non-blocking Arduino library for controlling 28BYJ-48 stepper motors [5].
Adafruit_NeoPixel.h	Allows control of Neopixel individually addressable LED’s

9 Testing

9.1 Marble Machine Box Test Case Specification

Table 9.1: Traceability Table for Marble Machine Box Tests

Test Case ID	**TPMs Tested**	**Requirements Tested**	**Test Description**	**Input Specifications**	**Expected Output**
T4C5_T01	TPM_03 TPM_08 TPM_09	T4C5_F12 T4C5_F13 T4C5_F16 T4C5_P06 T4C5_I08	Test whether LED animation functions work together.	N/A	All LED Animations function correctly during operation
T4C5_T02	TPM_01 TPM_02 TPM_03 TPM_08 TPM_09	T4C5_F05 T4C5_F12 T4C5_F13 T4C5_F16 T4C5_P01 T4C5_P02 T4C5_P06 T4C5_I01 T4C5_I02 T4C5_I03 T4C5_I04	Test all temporal subroutines operation	N/A	All functions trigger correctly
T4C5_T03	TPM_01 TPM_02 TPM_03 TPM_08 TPM_09	T4C5_F01 T4C5_F02 T4C5_F05 T4C5_F06 T4C5_I01 T4C5_I02 T4C5_I03 T4C5_I04 T4C5_I08 T4C5_I09	Test functions's ability to operate concurrency	N/A	Test all functions operate concurrently correctly
T4C5_T04	TPM_05 TPM_06 TPM_10	T4C5_F01 T4C5_F01 T4C5_F04 T4C5_F08 T4C5_F12 T4C5_F14 T4C5_F15 T4C5_I01 T4C5_I02 T4C5_I03 T4C5_I08	Test the machine's usability under heavy usage for 3 minutes. The software is run to test processing, memories and latency constraints.	N/A	The machine operates without any errors and the program returns a successful debug log.

Table 9.1 continued from previous page

Test Case ID	**TPMs Tested**	**Requirements Tested**	**Test Description**	**Input Specifications**	**Expected Output**
T4C5_T05	TPM_01 TPM_02 TPM_03 TPM_04 TPM_07 TPM_08 TPM_09	T4C5_F01 T4C5_F02 T4C5_F03 T4C5_F04 T4C5_F05 T4C5_F06 T4C5_F09 T4C5_F12 T4C5_F14 T4C5_F15 T4C5_F16 T4C5_P01 T4C5_P02 T4C5_P03 T4C5_P04 T4C5_P04 T4C5_P05 T4C5_P06 T4C5_I01 T4C5_I03 T4C5_I04 T4C5_I05 T4C5_I08 T4C5_I09	The system must be able to operate at least an 1 month of constant runtime load before maintenance	N/A	The machine is able to withstand load for at least 1 month of constant runtime.
T4C5_T06	TPM_03	T4C5_F02 T4C5_F05	Test whether IR break beam sensors can be powered.	N/A	The IR break beam sensors are showing active status in the serial monitor
T4C5_T07	TPM_03	T4C5_F02 T4C5_F05	Test whether the response of the sensors fluctuate or not	N/A	Same serial monitor commands when nothing passes and when any obstacle passes through the sensors serial monitor commands shows the status
T4C5_T08	TPM_08	T4C5_F09 T4C5_P04 T4C5_P05	Test servo's timing interval	N/A	The servo works after 1 second interval when marble passes through the sensor

Table 9.1 continued from previous page

Test Case ID	TPMs Tested	Requirements Tested	Test Description	Input Specifications	Expected Output
T4C5_T09	TPM_08	T4C5_F09 T4C5_P04 T4C5_P05	Test wether he servo motor returns to its position after certain interval	N/A	The servo returns to its position after certain interval
T4C5_T10	TPM_01 TPM_02	T4C5_F15 T4C5_P02	Test whether the stepper motor controls the speed of the marble machine	N/A	Rotate the wheel of the marble machine with synchronizing speed of the marble drop frequency
T4C5_T11	TPM_09	T4C5_F12 T4C5_F13 T4C5_F16 T4C5_P06	Test whether LED strips are powered and initialised	N/A	LED strip are initialised with Neopixel function and showing synchronized effect with marble machine

9.2 Test Environment

A controlled environment containing the prototype Marble Machine Box fitted with electronics and connected to power. This includes:

- the complete physical structure of the box.
- connected and working electronics.
- access to the Arduino Uno USB A programming port.

The test environment will simulate the installation orientation and height of the final Marble Machine Box provided to the client.

9.2.1 Personnel

For satisfying precautionary safety measures, only members of the teams responsible for the testing of the T4B4 Marble Machine Box as well as MQ Engineering faculty staff and the client(s) will be authorized access to the testing environment, including operation of testing equipment as well as the Marble Machine Box.

Training for operating the Marble Machine Box and diagnosing any problems discovered as a result of testing will be required for any and all personnel present in the testing environment. This includes:

- Operation of the Marble Machine Box

- Any physical interaction with wiring
- Any physical interaction with the power supply
- Uploading code to the Arduino Uno
- Familiarity with all of the test procedures documented

9.2.2 Equipment

Equipment to be present at the testing environment provided by T4_Comms5 throughout the period of testing will be;

- a Laptop;
- a USB cable (Type-B);
- Test script files (.ino file);
- a stopwatch;
- relevant Arduino software (IDE's, handbooks and manuals);
- T4B4 Marble Machine Box;
- Marbles

Each individual test outlines from this list which equipment is required.

9.3 Test Modes

The test modes of this document are the set of tests that will be conducted with the assistance, supervision and aid of the entirety of Team 4 as well as those performed solely by at least one T4C5 Student. Automated configurations may be used in some test cases to marble input.

The tests performed with all groups are the Integration and System tests. These tests are listed and discussed in further detail in Section 9.5 Marble Machine Specific Test Case Procedures. of this document. These tests require members of groups beyond those in T4C5 to perform due to the nature of their required equipment, personnel and training.

The tests performed by T4C5 are listed and discussed in further detail in Section 9.5 Marble Machine Specific Test Case Procedures. Automated scripts may be created for certain tests to simulate the marble input to box. No equipment beyond the T4B4 Control Code for which T4C5 is responsible for, as well as any item of equipment listed above in Equipment will be required for or provided by T4C5.

9.4 General Test Procedure

The general test procedure for T4C5 specific tests is as follows:

9.4.0.1 Log

Results of the test execution will be logged on a plain text log file, within the testing machines memory.

9.4.0.2 Setup

The Marble Machine Box T4B4 is connected to power. The Arduino Uno is then connected to a laptop via a USB cable (Type-B) and the Arduino IDE is opened.

9.4.0.3 Proceed

Complete specific test procedures outlined in the testing tables below.

9.4.0.4 Start

Initialize respective test(s) script(s), power on the robot.

9.4.0.5 Measure

Record the results after completing the respective test(s) and compare to expected results.

9.4.0.6 Stop

After finalizing the test procedure, stop all running scripts interfacing with the Arduino Uno and disconnect from the laptop.

9.4.0.7 Shutdown

Power off the Marble Machine Box and quit the Arduino IDE.

9.4.0.8 Contingencies

Power to both the electronics and Arduino are to be shut off immediately. Personnel should be assessed and treated for any injury. Equipment should then be inspected for any damage.

9.5 Marble Machine Specific Test Case Procedures

Table 9.2: Test Case T4C5_T01

Test Case	**T4C5_T01**	
Type of Test	Integration test	
Goal	Test LED animation functions work together.	
Equipment	Marble Machine Box	
Personnel and Training	T4 Comms 5	
Success Condition	• All LED Animations function correctly during operation	
Failure Condition/s	• LED Animations do not operate correctly	
Method	**Step**	**Action**
	1.	Start Box electronics operation and drop a marble.
	2.	Trigger IR Servo Gate Motors and ensure traffic light objects operate correctly.
	3.	Ensure Spiral LED strip animates correctly.
	4.	Ensure External LED Strip animates correctly.

Table 9.3: Test Case T4C5_T02

Test Case	**T4C5_T02**	
Type of Test	Integration test	
Goal	Test all temporal subroutines operate correctly.	
Equipment	Marble Machine Box	
Personnel and Training	T4 Comms 5	
Success Condition	• All functions trigger correctly	
Failure Condition/s	• Functions do not trigger correctly	
Method	**Step**	**Action**
	1.	Begin box operation and drop a marble.
	2.	Ensure Marble gate servos and traffic lights operate simultaneously.
	3.	Ensure Stepper motor continues to rotate.
	4.	Ensure all LED strips continue to update.

Table 9.4: Test Case T4C5_T03

Test Case	**T4C5_T03**	
Type of Test	Integration test	
Goal	Test all functions operate concurrently correctly.	
Equipment	Marble Machine Box	
Personnel and Training	T4 Comms 5	
Success Condition	• All functions operate correctly	
Failure Condition/s	• Functions do not operate correctly	
Method	**Step**	**Action**
	1.	Begin box operation and drop a marble.
	2.	Ensure Stepper motor and all led strips are updating.
	3.	Ensure marble gates functions without interrupting stepper motor or led strips.

Table 9.5: Test Case T4C5_T04

Test Case	**T4C5_T04**	
Type of Test	System test - stress testing	
Goal	Test the machine's usability under heavy usage for 3 minutes. The software is run to test processing, memories and latency constraints.	
Equipment	Marble Machine Box, PC/Laptop with USB connection to Arduino	
Personnel and Training	T4 Comms 5	
Success Condition	• The machine operates without any errors and the program returns a successful debug log	
Failure Condition/s	• Freezing • High latency/delay • Failure debug log	
Method	**Step**	**Action**
	1.	Modify constants related to stepper motor, LED and servo motors into variables.
	2.	Create a function that randomises variables without going over limits set by hardware.
	3.	Integrate the random variables function so the function starts up at the same time with the system.
	4.	Add an exit function that exits the program after 3 minutes.
	5.	Run software.
	6.	Software passes the test, result is recorded.

Table 9.5 continued from previous page

Test Case	**T4C5_T04**	
Alternative Flows	**Step**	**Action**
	5.a	Software fails the test, analyse result.
	6.a	Code review to identify errors/imperfections.

Table 9.6: Test Case T4C5_T05

Test Case	**T4C5_T05**	
Type of Test	System test - load testing	
Goal	The system must be able to operate at least an 1 month of constant runtime load before maintenance.	
Equipment	Marble Machine Box, PC/Laptop with USB connection to Arduino	
Personnel and Training	T4 Comms 5	
Success Condition	• The machine is able to withstand load for at least 1 month of constant runtime.	
Failure Condition/s	• The machine fails before 1 months of constant runtime. • Suspicious, unexpected or abnormal behaviours occurs at any time during operation.	
Method	**Step**	**Action**
	1.	Execute program and let the run.
	2.	Observing machine's behaviours.
	3.	Passes, record result.
Alternative Flows	**Step**	**Action**
	2.a	Machine loses power.
	2.a.1	Check power connection to the Marble Machine.
	2.b	Machine shows signs of failure.
	3.a	Fails, analyse result and identify errors.

Table 9.7: Test Case T4C5_T06

Test Case	**T4C5_T06**
Type of Test	Unit Test
Goal	IR break beam sensor must be powered.
Equipment	Marble Machine Box, PC/Laptop with USB connection to Arduino
Personnel and Training	T4 Comms 5

Table 9.7 continued from previous page

Test Case	**T4C5_T06**	
Success Condition	• The IR break beam sensors are showing active status in the serial monitor.	
Failure Condition/s	• Active status either unable to be seen on Serial Monitor return error status.	
Method	**Step**	**Action**
	1.	Execute and run the program.
	2.	Open the Serial Monitor.
	3.	Observe the IR break beam sensor status.
	4.	Passes, record result.
Alternative Flows	**Step**	**Action**
	1.a	Check machine power connection.
	2.a	Serial Monitor status fluctuates.
	3.a	Identify errors.

Table 9.8: Test Case T4C5_T07

Test Case	**T4C5_T07**	
Type of Test	Unit Test	
Goal	Response of the IR break beam sensors does not fluctuate.	
Equipment	Marble Machine Box, PC/Laptop with USB connection to Arduino	
Personnel and Training	T4 Comms 5	
Success Condition	• Same serial monitor commands when nothing passes and when any obstacle passes through the sensors serial monitor commands shows the status.	
Failure Condition/s	• Responses from the sensors fluctuate or show errors.	
Method	**Step**	**Action**
	1.	Execute and run the program.
	2.	Open the Serial Monitor.
	3.	Observe the IR break beam sensor status.
	4.	Pass a obstacle through the sensor.
	5.	Observe the sensor while passing the obstacle.
	6.	Record the reading.
Alternative Flows	**Step**	**Action**
	1.a	Check machine power connection.
	2.a	Check the reading on Serial Monitor.

Table 9.8 continued from previous page

Test Case	**T4C5_T07**	
	3.a	If it fluctuates identify the error.

Table 9.9: Test Case T4C5_T08

Test Case	**T4C5_T08**	
Type of Test	Unit Test	
Goal	Servo works after 1 second interval when marble passes through the sensors.	
Equipment	Marble Machine Box, PC/Laptop with USB connection to Arduino	
Personnel and Training	T4 Comms 5	
Success Condition	• The servo works after 1 second interval when marble passes through the sensors.	
Failure Condition/s	• The response from the IR sensors fluctuates or it does not detect the marble.	
Method	**Step**	**Action**
	1.	Run the program.
	2.	Pass the marble through the IR beam sensor.
	3.	Observe and record the timing when the servo motor turns.
	4.	Passes the test if the time interval is 1s.
Alternative Flows	**Step**	**Action**
	1.a	Check machine power connection.
	2.a	If the time interval is not 1s check the code.
	3.a	Identify the error and change the value for the delay accordingly.

Table 9.10: Test Case T4C5_T09

Test Case	**T4C5_T09**
Type of Test	Unit Test
Goal	The servo motor returns to its position after certain interval.
Equipment	Marble Machine Box, PC/Laptop with USB connection to Arduino
Personnel and Training	T4 Comms 5
Success Condition	• Servo returns to its position after certain interval.
Failure Condition/s	• Servo does not return to its position or the serial monitor commands fluctuates.

Table 9.10 continued from previous page

Test Case	**T4C5_T09**	
Method	**Step**	**Action**
	1.	Run the program.
	2.	Pass the marble through the machine.
	3.	The servo motor turns after 1s interval.
	4.	Observe the serial monitor.
	5.	Passes the test if the servo motor returns to its position after certain interval.
Alternative Flows	**Step**	**Action**
	1.a	Identify the error if servo not returning to its position.
	2.a	Check the code and change it accordingly and try again.
	3.a	Restart the program.

Table 9.11: Test Case T4C5_T10

Test Case	**T4C5_T10**	
Type of Test	Unit Test	
Goal	The stepper motor controls the speed of the marble machine.	
Equipment	Marble Machine Box, PC/Laptop with USB connection to Arduino	
Personnel and Training	T4 Comms 5	
Success Condition	• Rotate the wheel of the marble machine with synchronizing speed of the marble drop frequency.	
Failure Condition/s	• Speed of the wheel does not match with the marble drop frequency.	
Method	**Step**	**Action**
	1.	Run the program.
	2.	Observe and record the speed of the wheels.
	3.	Passes if the speed aligns with the marble drop frequency.
Alternative Flows	**Step**	**Action**
	1.a	Identify the error if servo not returning to its position.
	2.a	If fails check the code and identify the problem.

Table 9.12: Test Case T4C5_T10

Test Case	**T4C5_T10**	
Type of Test	Unit Test	
Goal	The stepper motor controls the speed of the marble machine.	
Equipment	Marble Machine Box, PC/Laptop with USB connection to Arduino	
Personnel and Training	T4 Comms 5	
Success Condition	• Rotate the wheel of the marble machine with synchronizing speed of the marble drop frequency.	
Failure Condition/s	• Speed of the wheel does not match with the marble drop frequency.	
Method	**Step**	**Action**
	1.	Run the program.
	2.	Observe and record the speed of the wheels.
	3.	Passes if the speed aligns with the marble drop frequency.
Alternative Flows	**Step**	**Action**
	1.a	Identify the error if servo not returning to its position.
	2.a	If fails check the code and identify the problem.

Table 9.13: Test Case T4C5_T11

Test Case	**T4C5_T11**	
Type of Test	Unit Test	
Goal	LED strips are powered and initialised.	
Equipment	Marble Machine Box, PC/Laptop with USB connection to Arduino	
Personnel and Training	T4 Comms 5	
Success Condition	• LED strip are initialised with Neopixel function and showing synchronized effect with marble machine.	
Failure Condition/s	• LED is not working. • LED effect not synchronized with box.	
Method	**Step**	**Action**
	1.	Run the program.
	2.	Observe LED animation effect.
	3.	Observe the Exterior lighting effect.
	4.	Observe the Ring lightning effect.
	5.	Observe the Traffic lighting effect.

Table 9.13 continued from previous page

Test Case	**T4C5_T11**	
	6.	Passes the test if all the lighting effects are synchronized with the marble machine.
Alternative Flows	**Step**	**Action**
	1.a	Restart the program.
	2.a	If fails check the code and identify the problem.

10 Design Review

This section presents an overview on the current design on a subsystem basis. This includes any improvements that were made to achieve the working solution.

10.0.1 Marble Detection

During the initial unit testing phase, the IR Break-beam sensors worked successfully. The components were tested using the Arduino Serial monitor, which would generate some arbitrary text if the beam was broken proceeding a predetermined delay.

10.0.2 Servomechanism

After re-examining the TicoServo library documentation, both servomotors were attached to pins 9 and 10 respectively. As with this change, the servomechanism was able to rotate to the specified positions. Additionally, this worked successfully when integrated with the marble detection and delay logic.

10.0.3 Stepper Motor Control

This component has proven to be problematic when testing the original design. The initial logic implemented the Stepper library. Through testing, it was discovered that the motor had a maximum allowable step per revolution of 19. After inconsistent results, with most being unsuccessful, the motor driver was replaced with the ULN2003 driver. As such, the CheapStepper library had replaced the Stepper library due to it being compatible with the motor-driver setup, as well as it offering non-blocking functionality.

10.0.4 Lighting

This entire component has been very successful in its implementation. Additional changes to the pin layout were made to cater for the TicoServo library requisites. The "traffic" lighting functions are now tethered to the IR Break-beam logic, as opposed to it being under a separate clock. This ensures its synchronous behaviour upon integration and system checks.

10.1 End of life

This entire marble machine sculpture is presented to be able to operate for 2 years. Generally, maintenance can occur once or twice a month. An effective methodology for this would be to follow the general test plan as to review its performance after operating for lengthy periods of time.

10.2 Future Implementations

For non-MVP design elements, additional features were discussed. An LCD and piezoelectric buzzer component could be installed onto the box. This would expand upon the marble detection logic, as the display would present the marble count. The buzzer would run synchronously operate with these components and vibrate after a certain fixed number of marbles have passed through.

Additionally, more external lights can be added to follow the tracks. These features would further expand on the requirement of designing a kinetically entertaining sculpture surrounding the given box. If provided additional time for this project, this can be implemented fairly easily. Similar tests would be used on each component to ensure they work well with the IR sensors. Overall, this would be a natural follow up for the current product.

11 Conclusion

Developing this Marble Machine throughout the course of this project has required a systematic approach to the design. The majority of this project has involved the integration and communication between multidisciplinary teams. Due to this aspect, discussing design alternatives was a consistently occurring step. As for the project schedule, we initially started late due to various complications. The team was also comprised of 3 people which later grew into 5. Despite this, T4C5 remained on time with its deliverable components.

12 References

[1] IEEE, "ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering," *ISO/IEC/IEEE 29148:2018(E)*, pp. 1–104, 2018.

[2] R. di Bona, "2022 SPINE Engineering Project ENGG2000/ENGG3000: Massive Marvellous Multiple Marble Machine v1.2," 2022. [Online]. Available: https://ilearn.mq.edu.au/mod/resource/view.php?id=7260367

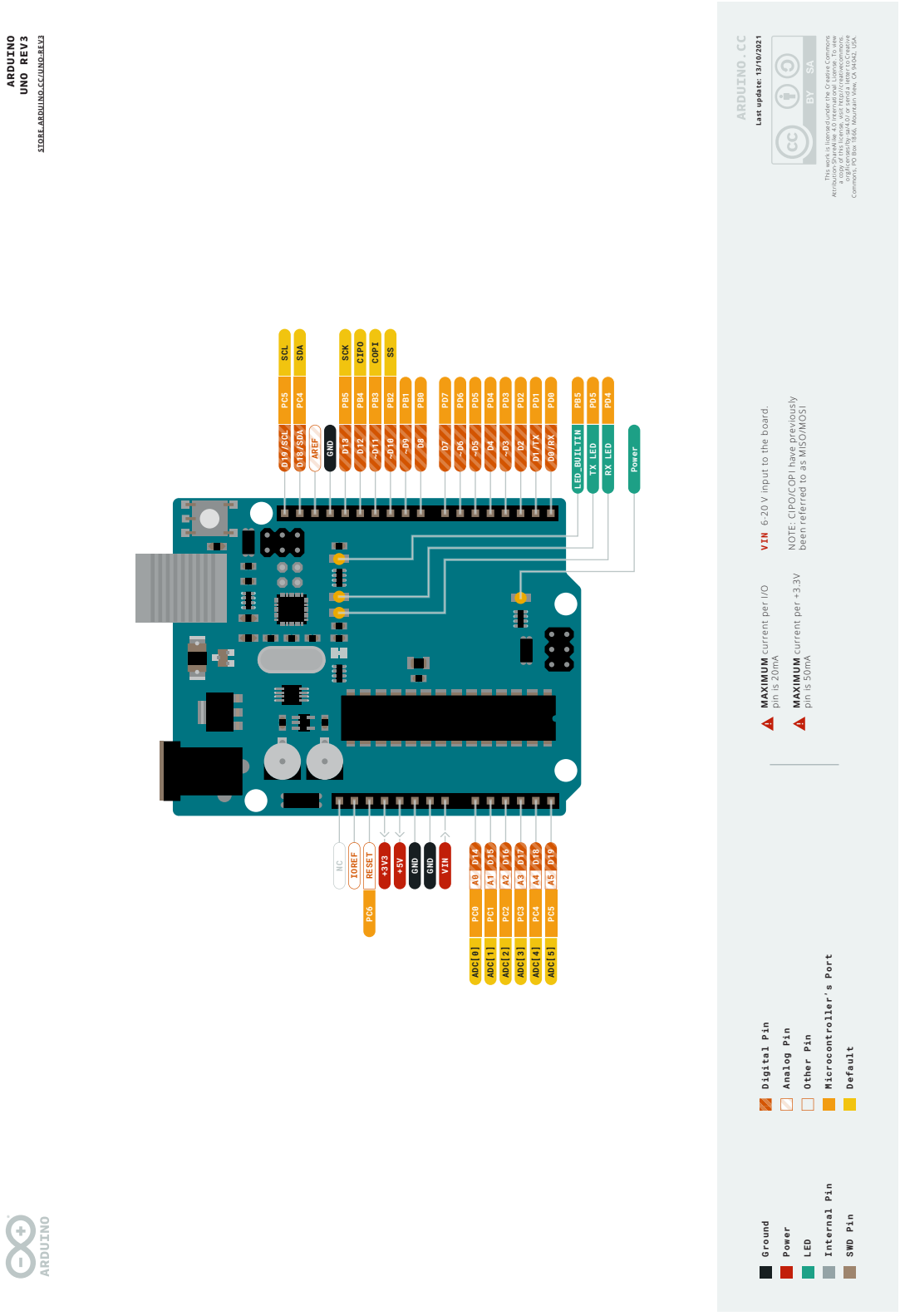
[3] "IR Break Beam Sensor - 3mm LEDs," 2022. [Online]. Available: https://core-electronics.com.au/ir-break-beam-sensor-3mm-leds.html

[4] "9g micro servo (1.6kg)," 2022. [Online]. Available: https://core-electronics.com.au/9g-micro-servo-1-6kg.html

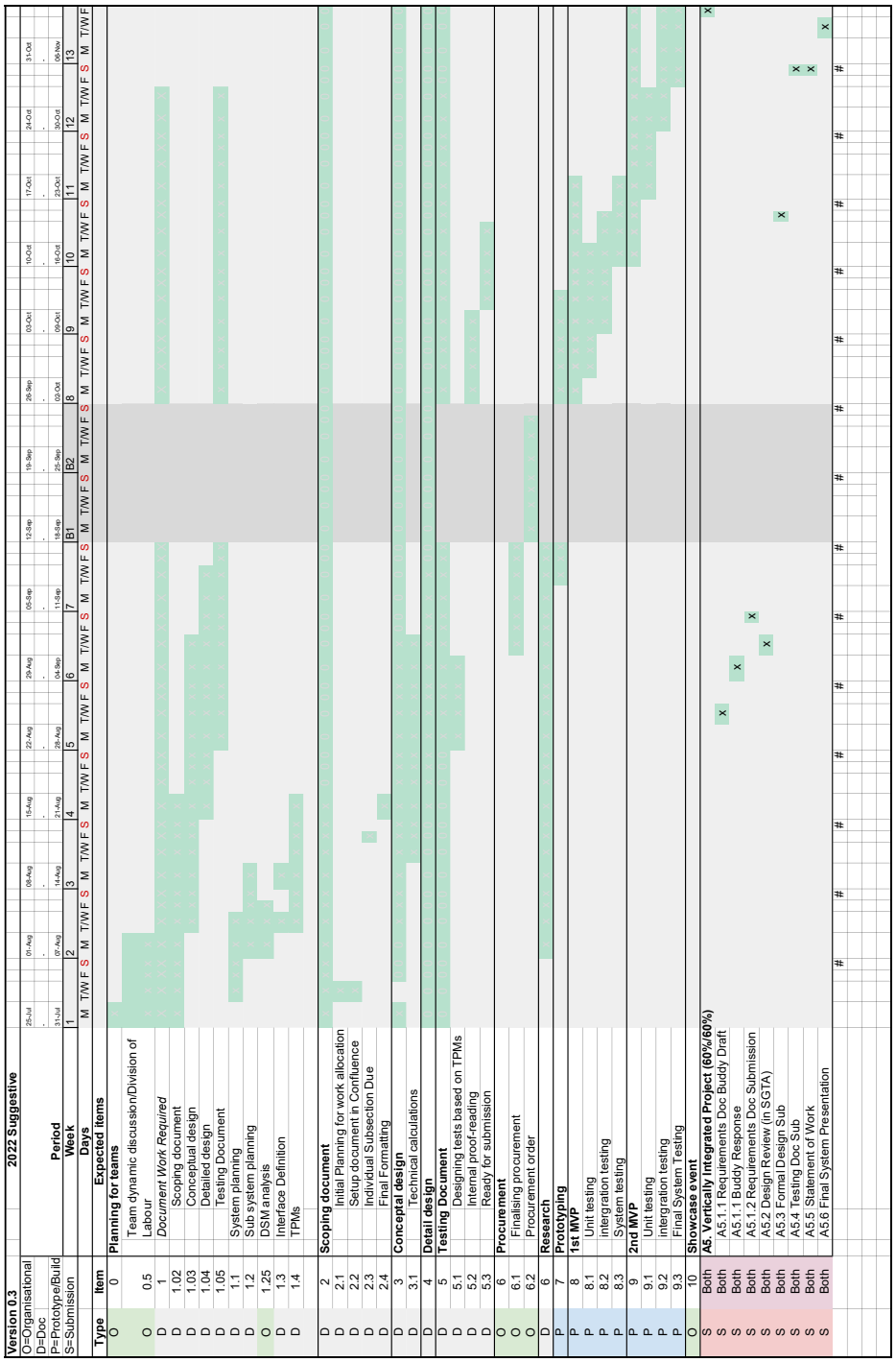
[5] "5V Geared Stepper Motor," 2022. [Online]. Available: https://core-electronics.com.au/5v-geared-stepper-motor.html

13 Appendix

A Arduino Uno Pinout



B Gantt Chart



C Marble Machine Box Control Code

```
#include <Adafruit_TiCoServo.h>
#include <Stepper.h>
#include <Adafruit_NeoPixel.h>

// Parameters for marble wheel rotation
#define REVOLUTION_FREQ_MS 1000
#define WHEEL_SLOTS 8

// Motor steps for 1 revolution
const int stepsPerRevolution = 64;
// Number of motor steps between each drop of a marble
const int stepsToDropMarble = (stepsPerRevolution/WHEEL_SLOTS);
// Frequency between steps so that marbles are dropped
// at the configured frequency (1 second)
const int stepFrequency = REVOLUTION_FREQ_MS/stepsToDropMarble;
unsigned long lastStepTime = 0;

// Motor driver pins, Model: Makerverse 2 Channel Motor Driver
#define STEPPER_DIR_B 3
#define STEPPER_DIR_A 4
#define STEPPER_PWM_B 5
#define STEPPER_PWM_A 6

// Configure the stepper motor
Stepper wheelMotor(stepsPerRevolution, STEPPER_DIR_B, STEPPER_DIR_A,\
                   STEPPER_PWM_B, STEPPER_PWM_A);


//LED CONFIGS (change as required)
#define LED_INTERVAL 500 //1/2 sec

unsigned long lastLEDUpdate = 0;


#define IRPIN1 13 //IR1 pin
#define IRPIN2 4  //IR2 pin
// Interval time for marble to travel from IR sensor to servo
// used to time the closing time
// of the servo after it is detected by the IR sensor
#define MARBLE_TRAVEL_INTERVAL 10
#define MARBLE_STATIONARY_TIME 1000
unsigned long marble1Detected = 0;  //IR1 status
unsigned long marble2Detected = 0;  //IR2 status

Adafruit_TiCoServo servo1;  //creating test servo 1
Adafruit_TiCoServo servo2;  //creating test servo 2

#define SERVO_OPEN 0
#define SERVO_CLOSE 90

int TrafficStage1;
int TrafficStage2;

uint32_t ringLEDs[12];
int ringHue[12];

Adafruit_NeoPixel exteriorStrip = Adafruit_NeoPixel(12, 8, NEO_GRB + NEO_KHZ800);
Adafruit_NeoPixel trafficStrip1 = Adafruit_NeoPixel(3, 7, NEO_GRB + NEO_KHZ800);
Adafruit_NeoPixel trafficStrip2 = Adafruit_NeoPixel(3, 6, NEO_GRB + NEO_KHZ800);
Adafruit_NeoPixel ringStrip = Adafruit_NeoPixel(12, 5, NEO_GRB + NEO_KHZ800);


void setup() {
  servo1.attach(9);  //attach servo1
  servo2.attach(10);  //attach servo2
  pinMode(IRPIN1, INPUT);  //setup IR1
  pinMode(IRPIN2, INPUT); //setup IR2
  LEDinit();
```

```
}

void loop() {
  rotateMarbleWheel();

  detectIRSensor1();
  detectIRSensor2();

  servo1Swing();
  servo2Swing();
  LEDRingUpdate();
  LEDExteriorUpdate();
}

// Rotates the marble wheel 1 step in tune
// with the configured frequency of marble movement
void rotateMarbleWheel() {
  // Rotate the motor 1 step if the next frequency interval is reached
  unsigned long currentMillis = millis();
  if (currentMillis != lastStepTime && currentMillis%stepFrequency == 0) {
    wheelMotor.step(1);
    lastStepTime = currentMillis;
  }

}

//Returns boolean indicator if marble is detected,
//and updates the current timestamp
bool detectIRSensor1(){
  unsigned long currMillis = millis();
  if (digitalRead(IRPIN1) == HIGH) {
      marble1Detected = currMillis;
  }

  if (marble1Detected >= currMillis - MARBLE_TRAVEL_INTERVAL) {
    return true;
  } else {
    return false;
  }
}

//same as above
bool detectIRSensor2(){
  unsigned long currMillis = millis();
  if (digitalRead(IRPIN2) == HIGH) {
      marble2Detected = currMillis;
  }

  if (marble2Detected >= currMillis - MARBLE_TRAVEL_INTERVAL) {
    return true;
  } else {
    return false;
  }
}

//Writes angle values to the servo to set the mechanism accordingly
void servo1Swing() {
  unsigned long currMillis = millis();
  if(marble1Detected + MARBLE_TRAVEL_INTERVAL + MARBLE_STATIONARY_TIME <= currMillis){
    servo1.write(SERVO_OPEN);
    TrafficStage1 = LEDTrafficLightUpdate(1, 1);
  } else if(marble1Detected + MARBLE_TRAVEL_INTERVAL <= currMillis){
    servo1.write(SERVO_CLOSE);
    TrafficStage1 = LEDTrafficLightUpdate(2, 1);

  }
}

//same as above
void servo2Swing(){
```

```
  unsigned long currMillis = millis();
  if(marble2Detected + MARBLE_TRAVEL_INTERVAL + MARBLE_STATIONARY_TIME <= currMillis){
    servo2.write(SERVO_OPEN);
    TrafficStage2 = LEDTrafficLightUpdate(1, 2);

  } else if(marble2Detected + MARBLE_TRAVEL_INTERVAL <= currMillis){
    servo2.write(SERVO_CLOSE);
    TrafficStage2 = LEDTrafficLightUpdate(2, 2);

  }
}

//Sets data pins for Neopixel output, and initialises all pixels as 'off'
void LEDinit() {

  exteriorStrip.begin(); //start data out
  trafficStrip1.begin();
  trafficStrip2.begin();
  ringStrip.begin();
  exteriorStrip.show(); //init 0,0,0
  trafficStrip1.show();
  trafficStrip2.show();
  ringStrip.show();
  exteriorStrip.setBrightness(128); //halve brightness
  trafficStrip1.setBrightness(128);
  trafficStrip2.setBrightness(128);
  ringStrip.setBrightness(128);

  for (int i = 0; i < 12; i++) {
    ringHue[i] = (5461/2)*i;
  }
}

//Sets ring lights to red configured with each interval
void LEDRingUpdate() {
  for (int i = 0; i < 12; i++) {
    if (ringHue[i] > 65535) ringHue[i] == 0;
    ringLEDs[i] = ringStrip.gamma32(ringStrip.ColorHSV(ringHue[i]));
    ringHue[i] += 25;
    ringStrip.setPixelColor(i, ringLEDs[i]);
  }
  ringStrip.show();
}

//takes in current stage of lights (stage) and which led strip to control (light)
//return new stage light set to
int LEDTrafficLightUpdate(int stage, int light) {
  uint32_t red = exteriorStrip.Color(255, 0, 0);
  uint32_t orange = exteriorStrip.Color(255, 100, 0);
  uint32_t green = exteriorStrip.Color(0, 255, 0);

  switch (stage) {
    case 0: //light is red
      setTrafficDist(orange, 1, light);
      stage++;
    break;
    case 1: //light is orange
      setTrafficDist(green, 2, light);
      stage++;
    break;
    case 2: //light is green
      setTrafficDist(red, 0, light);
      stage = 0;
    break;
  }

  return stage;
}

//triggers setTraffic1 or setTraffic2 depending on int light
```

```
209 void setTrafficDist(uint32_t color, int n, int light) {
210   if (light == 1) setTraffic1(color, n);
211   if (light == 2) setTraffic2(color, n);
212 }
213
214 //Sets new colour value to LED's and send data to strip
215 void setTraffic1(uint32_t color, int n) {
216   uint32_t empty = exteriorStrip.Color(0, 0, 0);
217   trafficStrip1.fill(empty, 0, 3); //blank all 3 leds
218   trafficStrip1.setPixelColor(n, color); //set correct led to color
219   trafficStrip1.show(); //send data to strip
220 }
221
222 //same as above
223 void setTraffic2(uint32_t color, int n) {
224   uint32_t empty = exteriorStrip.Color(0, 0, 0);
225   trafficStrip2.fill(empty, 0, 3);
226   trafficStrip2.setPixelColor(n, color);
227   trafficStrip2.show();
228 }
229
230 //Sets Exterior LED's to empty based and saves current timestamp
231 void LEDExteriorUpdate() {
232   uint32_t empty = exteriorStrip.Color(0, 0, 0);
233 }
```

Listing 1: Marble Machine Box Control Code - main.ino

D Circuit Diagram (Provided by T4M5)