Stasis Engine

Final Report
PDE3413
Systems Engineering In Robotics
Lecturer: Mr Stephen J. Hall
22/05/24

Author: Abdur Rehman Nasir
M00xxxxxx

1 Abstract

The ‘Stasis Engine’ is at its base, a deceleration system akin to ‘automatic braking’, with a specific focus on optimality of comfort and wear. Its uses extend from the subsection of automating braking in AI-assisted driving, to swift and non-lethal deceleration of a fleeing vehicle in an emergency situation. As a concept, it aims to gradually decelerate the velocity of an object in motion to zero, simulating a surface with zero restitution. For the purposes of this project, this will be a ball travelling in a straight line, but proof of concept holds. The boundaries of this eventual reduction can be visualised by common real-life archetypes from which we can easily draw coherent parallels. The boundary in terms of braking force can be understood by comparison to common vehicular brake pads. The first boundary can be recorded with comparison to the case of extreme vehicular braking. In the case that this ‘extreme braking’ occurs, then so too does extreme, unusual wear on the brake pad and related parts. This is to be avoided in normal driving behaviour, and therefore serves as a hardline boundary to this project. Likewise, the opposite boundary can be recorded with reference to a uncontrolled vehicular collision. One can imagine the type and severity of damage to not only the vehicle, but any inhabitants or otherwise unsecured loads within it, keeping in mind ‘crumple zones’ and sudden changes in inertia. This is to be avoided in decelerative habits, and any deviation from this principle would nullify the purpose of the Stasis Engine. The most suitable build would find the optimal balance between speed of deceleration and instant application of force to the object in motion, which at the most basic overview lies directly in the middle of the two hardline boundaries of extreme braking and uncontrolled collision. At this median, there lies a point of assured optimality, at which point its visible effect can be seen: zero restitution. If an object in motion is received with a force variably increasing dependent on its deceleration, then the result will be that of zero restitution – a visible symptom of success.

As such, the Stasis Engine becomes a tool for optimal deceleration of an object in motion, such that when correctly received, the result is both zero restitution and the non-lethal deceleration of the object including any inhabitant or unsecured internal loads.

Table of Contents

2 Introduction

This project aims to prototype a ‘Stasis Engine’, a robotic mechanism designed to gradually decelerate a moving object's velocity to zero, simulating a surface with zero restitution. The force returned opposite, should be variably assessed and updated per microsecond to ensure that there is notably minimal rebound on contact. This report aims to conjure a detailed and pedantic perspective on the design and build of the Stasis Engine, starting from its objectives and functions, run concurrently alongside overviews of the test cases to use during its conception. Moving to its physical architecture, including a 3D model and circuit structure, the tangibility of the robot is discussed. At the climax of the report, the build list is discussed comprehensively, including any considerations made towards component pathways, their physical dimensions and price, and feasibility of attachment to the robot. It follows into the build process of these components, the alternatives sought from the original design and workarounds made available due to lack of material. The final decision consists of a stepper motor, lead screw/nut combo and a master/slave cylinder system implemented syringes connected by tubing. The report ends with other possible directions the Stasis Engine could have gone.

3 Project Aim

Aim: 'Catch' rolling objects with variable deceleration, such that no restitution occurs.

3.1 Explanation by Analogy

The 'catch' function mentioned, is treated much like finding the optimal balance between braking and riding neutral when coming to a stop at a red traffic light. Within this scenario, the initial pressure you apply to the brakes is dependent on the distance at which you spotted the traffic light, or any similar obstacle requiring deceleration (Lee et al., 1976). Subsequently, it is less obvious that there are several methods to approach it while causing *minimal* physical disruption to the driving experience of internal passengers – and concurrently, less brake wear, less injurious risk in rear-end accidents (Hynes and Dickey, 2008), less time spent at suboptimal fuel efficiency etc.. As such, a less experienced driver may begin braking straight away, while a veteran driver would recognise that there is a calculable 'sweet spot' that serves as the point of greatest efficiency for both passenger and car. This culminates in a ballpark of 20-30% shorter braking distance for those experienced behind the wheel (Greibe, P., 2008). While this is dependent on distance as well as driver skill, and the earlier you react, the easier and less resistive the stop is for an internal passenger. Here, *the rate of change of* acceleration is taken into account. This metric is commonly called a 'jerk' in machinery and vehicle design (Schot, 1978), and dominates engineering considerations where smooth changes in acceleration are necessary (Hayati et al., 2010). Sudden changes in acceleration (high jerk) can lead to mechanical stress and discomfort for passengers, and general risk to unsecured internal loads (Baumann et al., 2005). As this is an unintuitive term for the general reader, and in fact most people outside of niche vehicle engineering (Hayati et al., 2010), this report will defer to using the definition instead. Subsequently, the *rate of change of acceleration* is a fundamental underpinning the Stasis Engine, with the ability to manipulate it variably based on data collected is akin to computerising oscillatory damping.

3.1.1 High-Speed Vehicles

Compare this to an industrial-level, house-sized Stasis Engine, deployed as an emergency vehicle to stop a high-speed vehicle with minimal damage to internal passengers (such that they can be correctly processed). There would need to be an adequately large pneumatic cylinder rod to maximally slow the vehicle while minimally affecting the driver. Too abrupt of a stop would certainly cause the driver destructive, life-changing or fatal damage, and too mild of a stop wouldn't slow the vehicle enough to prevent it. Preventative measure in involuntary scenarios heavily rely on reducing vehicle acceleration (Isaksson-Hellman and Lindman, 2016) – which meets the purpose of the Stasis Engine. The use case in itself may come with considerable complications, but one can imagine a successful execution will display the oft monotonous laws of physics within an unusually surreal event.

3.1.2 Trains

In a similar scenario that remains one of the inspirations for this project, locomotive buffers work as a last resort for rail vehicles that have failed to stop, and are positioned on rails between areas of high human population density (stations, depots) and oncoming trains. Their use of the more reliable, well-lubricated oleo-pneumatic shock absorbers has allowed for a secondary purpose for decades – the use of the strong, non-restitutive rebound effect to push the train into reversal without straining the engine to move the enormous mass of a potentially loaded locomotive from static. This also ensures the safety of internal passengers due to the *variable* resistive forces.

3.2 Method Conceptualisation

Inherent in the design of the Stasis Engine is a substantial mathematical component, with heavy reliance on physics. In a natural experiment, the only two possible cases for the object's motion towards the Stasis Engine are highlighted in Section 3.2.1 and Section 3.2.2. Additionally, it is explained in the former why acceleration is not only unrealistic, but unobservable for the purposes of this endeavour.

3.2.1 Acceleration

When struck, a static object will rapidly accelerate, resulting in an increase to both its momentum and velocity. In a basic, uninformed visualisation, once leaving the contact surface of the device used to strike, the object's acceleration would rapidly increase till atmospheric and terrestrial resistances overtake the accelerative force. Hence, *Function 1.2* was written to allow for enough space between the object and Stasis Engine to ensure acceleration returns to net negative by the time of contact. However, this entire visual is not quite accurate.

"*When two bodies collide, they exert large forces on one another (during the time of the collision) called* ***impulsive*** *forces*" - (Alrasheed, 2019)

At the moment of impact, between a striking surface and an object, all accelerative potential and momentum is applied during the duration of the contact and no further. Assuming a non-isolated, open system, deceleration is instant after the two bodies cease contact. During contact, the collision parameter is non-zero between initial contact and cessation of it thereafter (Mirtitch and Canny, 2002), implying that the force deliverance occurs entirely between these two boundaries. Where an object is not materially robust enough to withstand this force transfer, it shatters (Roylance, 2001). This case remains so for the purposes of this report, unless the striking surface delivers constant energy by remaining in contact with the object, which is not currently within the scope of the Stasis Engine's purpose, but is referenced in *Section 12 – Future Developments*.

This means that as soon as the object leaves the striking surface, it begins decelerating – meaning it will always reach the Stasis Engine in a decelerative state, resulting in the continual observance of *Section 3.2.3*.

3.2.3 Deceleration

As mentioned in *Section 3.2.2*, an object will always reach the Stasis Engine in a decelerative state. Therefore, beyond any alterations to testing, no other cases need to be accounted for. This is detailed in *Section 5: Objectives 3-5*.

4 Objectives and Functions

The objectives of the Stasis Engine are limited to the five motivations of its engineering:

1) *Portability*
2) *Accelerative assessment*
3) *Incoming force estimation*
4) *Outgoing force generation*
5) *Optimal reciprocatory force reduction*

From which the following hierarchy can be prescribed.

Overview

1. Locate the object, and move into a suitable position.

- **1.1** Estimate the location of the object.
  - **1.1.1** Rotate to search.
  - **1.1.2** Identify object.
- **1.2** Move backwards to allow kinetic transfer into object.
  - **1.2.1** Check for space using rear distance sensor.
  - **1.2.2** Move back till object is out of sight of primary distance sensor.

2. Calculate the velocity of the object.

- **2.1** Use sensory mechanisms to measure speed at different points.
- **2.2** Find average speed and rate of acceleration.
  - **2.2.1** Use formula ‘s=d/t’ to find speed.
  - **2.2.2** Calculate rate of change between each speed to measure rate of acceleration.

3. Output an appropriate force required to prevent initial bounce.

- **3.1** Estimate time of contact with contact surface.
- **3.2** Estimate force required to prevent restitution.
  - **3.2.1** Calculate instant retraction speed and distance required.

3.2.2 Instruct actuator to execute at time of contact.

3.3 Extend actuator before time of contact.

3.4 Before time of contact, employ instant retraction speed to ‘catch’ the object.

4. Variably decrease this force in the next instant.

4.1 Calculate a force reduction curve based on the data collected, to predict force required.

4.2 Calculate the new appropriate force at the next fastest actuator update time.

5. Continue to variably decrease force till zero restitution achieved.

5.1 Use the force reduction curve calculated to send pre-processed results to the actuator for each interval.

5.2. As object nears zero velocity, adopt a predictable force reduction curve for the final end, since bounces may exceed microsecond oscillations.

In total, the Stasis Engine is designed with:

- 5 Objectives
- 12 Functions
- 8 Sub-functions

For a total of 25 uniquely titled segments underpinning the aim of the project.

4.1 *Objective 1:* Locate the object, and move into a suitable position.

In order to tackle real-world scenarios, the Stasis Engine must in fact be **portable**, in the same vein as other emergency vehicles are portable. In order to gage the direction from which the object is travelling, a distance sensor is required. Subsequently, moving the robot to match the central contact surface to the trajectory of the moving object requires programmable movement. These are detailed in several functions below.

4.1.1 Function 1.1: Estimate the location of the object.

Within the theme of portability, the Stasis Engine should be able to estimate the location of an object placed within its range of vision. For this implementation, we’d require 4 metres of free space extending from each plane of the robot, on a reasonably smooth surface, such that the presence of a rogue object would be within the detective capabilities of the primary HC-SR04 sensors.

Function 1.1.1: Rotate to search.

Scan surroundings. This is a basic implementation of the ability to search a state space. The installation of omnidirectional wheels will provide the ability to remain spatially static, and allows the privilege of a permanently affixed distance sensor, while still allowing a 360 degree view range.

Function 1.1.2: Identify object.

Detect anomaly. Where an object is placed within the vision range of the robot, the distance sensors will receive an abnormal reading via the ECHO pin. The ultrasonic wave would return much faster than previously, indicating the presence of an object.

4.1.2 Function 1.2: Move backwards to allow kinetic transfer into object.

Allow space for the object to receive sufficient kinetic energy to induce a strong enough linear path to contact the robot with reasonable force. Since the Stasis Engine is designed for stopping "moving" objects, it should be able to work purely off of speed metrics collected at different distances. This means the shape of the object is unnecessary, as is its size or mass (keeping reasonable dimensions). As a result, data should be processed and acted upon with microsecond intervals.

Function 1.2.1: Check for space using rear distance sensor.

Adhering to the 400cm max range of the HC-SRF05, the robot should assess that the nearest anomaly to the rear is at maximum a reasonable distance – a minimum of 30cm is a very safe estimation.

Function 1.2.2: Move back till object is out of sight of primary distance sensor.

Working in interplay with *Function 1.2.1*, the robot should accept a range of distances to move back to, lower bounded at 'the object is just out of sight of the primary sensor' & 'the structure to the rear is under 50cm away'. This should work with a beeper sound, ascending in frequency as the Stasis Engine approaches a structure to the rear, emulating both a parking sensor and the reversal of a large vehicle.

Test Cases

Test No.	Function	Name	Description	Method	Expected Outcome
1	1.1	Turn on	Push a button to turn the robot on.	Power state turns to active, LED turns to green.	The power state should change to 'active', indicated by the LED turning to green.
2	1.1.1	Rotate	Rotate the robot around its central axis.	The DC motors power the omnidirectional wheels to allow static rotation without lateral movement.	The robot should rotate clockwise, or anticlockwise dependent on programming. The chassis should remain static during this movement.
3	1.1.2	Identify	The robot detects an	The robot registers this	The robot should turn the LED

		Object	anomalous distance reading within its range of vision.	as the object required to act upon.	for ‘object detected’ on.
4	1.2.1	Reverse Check	Use the rear distance sensor to gage whether there is adequate distance to reverse, and if there is, distance available to reverse within limitation explained in *Test Case 5*.	The HC-SRF05 will send an analog response to the Arduino, clarifying reversal eligibility and distance.	The HC-SRF05 should perform its operation, to receive sensory data that describes the distance between itself and any rear structure that may impede the robot.
5	1.2.2	Reverse Till Adequate	Move the robot backwards, till either a sufficient distance has been covered, or the nearest structure to the rear is under 50cm away. Output regular beeps, increasing in frequency as the robot approaches a structure to the rear.	DC motors power omnidirectional wheels to allow longitudinal movement. A beeper can simulate a parking sensor.	The DC motors should actuate the wheels such that the robot reverses. The robot should reverse till the distance limitation is met, at which point it should stop shy of collision. A beep should sound at regular intervals, increasing in frequency till a continuous beep when the robot is too close to a rear structure.

4.2 *Objective 2:* Calculate the velocity of the object.

At this stage, the object will be moving linearly towards the robot at a linear speed. In order to judge speed, and subsequent decelerative force required at the contact surface, there are three points of speed assessment available. A HC-SR04 is attached to the top-front (primary sensor), bottom-left (secondary sensor) and bottom-right (tertiary sensor) of the robot. These measure distance at different points, and combining their data will allow for a comprehensive measure of both average speed, and object acceleration (which can be negative).

4.2.1 Function 2.1: Use sensory mechanisms to measure speed at different points.

Use all three sensors to sequentially measure distance at point of convergence with sensor vision. The primary sensor will provide the first point of measure, then the secondary, and finally the tertiary. As such, this function creates fertile grounds for *multi-sensor data integration.*

4.2.2 Function 2.2: Find average speed and rate of acceleration.

Some simple mathematical formulae will be used alongside the sensor outputs. This must take place in microsecond timing, or the actuator won’t receive instructions in time.

Function 2.2.1: Find average speed.

Use formula ‘s=d/t’ to find the speed at all three points, and take an average of the results.

Function 2.2.2: Find rate of acceleration.
Calculate rate of change between each speed to measure rate of acceleration.

Test Cases

Test No.	Function	Name	Description	Method	Expected Outcome
6	2.1	Primary Distance Check	Establish a distance and time checkpoint at the first point of convergence with the primary sensor's FoV (field of view).	Continuously engage the TRIGGER pin on the *primary* HC-SR04 (top-front of the robot) until an anomalous output is received via the ECHO pin.	The *primary* HC-SR04 should actuate, receiving *primary* distance data from the motion of the object.
7	2.1	Secondary Distance Check	Establish a distance and time checkpoint at the first point of convergence with the secondary sensor's FoV. Follows ***Test Case 6*.**	Continuously engage the TRIGGER pin on the *secondary* HC-SR04 (bottom-left of the robot) until an anomalous output is received via the ECHO pin.	The *secondary* HC-SR04 should actuate, receiving *secondary* distance data from the motion of the object.
8	2.1	Tertiary Distance Check	Establish a distance and time checkpoint at the first point of convergence with the tertiary sensor's FoV. Follows ***Test Case 7*.**	Continuously engage the TRIGGER pin on the *tertiary* HC-SR04 (bottom-right of the robot) until an anomalous output is received via the ECHO pin.	The *tertiary* HC-SR04 should actuate, receiving *tertiary* distance data from the motion of the object.
9	2.2	Data Check	Verify that all the data was correctly received – also finds a vital use case in test failure scenarios.	Activate an LED that verifies that the robot is primed with all the external sensor data required to actuate.	The robot should successfully receive all data from the three HC-SR04s, then turn on an LED signifying data collection has been successful.
10	2.2.1	Average Speed Calculation	Calculate (*and store for **Test Case 11***) the speed of the object at each of the three checkpoints, using object distance and time. Collate into an	Use the distances acquired, and the time of their acquisition, offset against the travel distance of the ultrasonic wave, to find	The Arduino should prepare an average calculation using the data collected.

			average speed metric.	the speed for each checkpoint via ‘s=d/t’ (speed = distance/time). Follow with a calculation of average.	
11	2.2.2	Acceleration Calculation	Calculate (and store for ***Test Case 13***) the rate of change of speed between each checkpoint, in order to instruct the actuator to deliver an opposite, similarly decelerative force.	Use the speeds calculated to find two acceleration metrics between the primary & secondary checkpoints, and the secondary & tertiary checkpoints.	The Arduino should prepare two values for the rate of change of acceleration between the three checkpoints.

4.3 *Objective 3*: Output an appropriate force required to prevent initial bounce.

This objective introduces the entire purpose of the Stasis Engine. In the successful case of all three checkpoints being correctly queried and processed, the actuator is primed for extension. At this point, the final calculations occur, and the contact surface is extended to meet the object in its path.

4.3.1 Function 3.1: Estimate time of contact with contact surface.

Since the contact surface does not have a contact sensor (detailed in the Future Developments section here), the point of contact can only be estimated. This requires microsecond precision, but abiding by minute uncertainty principles can in best case simulate ‘visual’ zero restitution. This uncertainty is also accounted for in a very general way in *Function 3.2.1*.

4.3.2 Function 3.2: Estimate force required to prevent restitution.

Using the data collected on the travel of the object between checkpoints, we can initialise an immediate actuator response post-contact. This entails the calculation of a sufficient decelerative force to completely prevent restitution. Because of the estimative design of the robot, this means the contact surface will be in motion **before** contact, and therefore will travel at the speed required for a small, predefined distance in order to account for uncertainty in time of contact estimation.

Function 3.2.1: Calculate instant retraction speed and distance required.

Using the data on the acceleration of the object between the three sensor checkpoints, the Arduino can estimate the speed, and *rate of change of* acceleration of the object at the time of contact. Further, we can match the speed of the object at or slightly before the time of contact for a preset distance to account for uncertainty in time of contact measurement.

Function 3.2.2: Instruct actuator to execute before time of contact.

The relevant function in the Arduino is suspended till the estimated time of contact occurs, and then it's fed all the data in order to correctly variably retract the pneumatic cylinder.

4.3.3 Function 3.3: Extend actuator before time of contact.

Where the actuator may be at a partially retracted position, this function ensures that the condition of full extension is met before the object is met.

4.3.4 Function 3.4: Before time of contact, employ instant retraction speed to 'catch' the object.

Several milliseconds before the estimated time of contact, the pneumatic cylinder should begin retraction at the speed calculated in *Function 3.2.1*. This will ensure the object doesn't immediately suffer the effect of restitution, since two objects in contact, travelling at identical speeds, are essentially independent of each other in terms of forces imposed by either, on either.

Test Cases

Test No.	Function	Name	Description	Method	Expected Outcome
12	3.1	Time of Contact Estimation	Using the data collected from the sensors, estimate the time whence the object will meet the contact surface.	Use average speed, rate of acceleration, distance from point of convergence of final sensor to contact point to estimate and store time of contact.	The robot should prepare an estimation for time of contact, and store it for future use.
13	3.2.1	Instant Retraction Calculation	Estimate the actuator speed required to match the object's speed at time of contact, taking into account *rate of change of* acceleration at contact time.	Use the data collected in ***Test Cases 10, 11, 12*** to form an estimation of speed to retract the pneumatic rod at, just before time of contact.	The robot should prepare a speed and acceleration required at the time of contact.
14	3.2.1	Initial Retraction Distance	Use a small, predefined distance within which the actuator's retractive speed remains constant for up to 100 milliseconds (subject to change), to allow for any uncertainty in previous calculations introducing a flaw in the time of contact.	Allows a short distance for the actuator to match the speed of the object – replicating a condition where no forces act upon it.	The robot should prepare a short distance to be able to apply fully the force reduction curve mentioned in ***Test Case 15***.

4.4 *Objective 4*: Variably decrease this force in the next instant.

As the sensor data collected turns into variable, predictive actuation for the first time, the microcontroller must work at or near minimal query time, such that subsequent predictive calculations and decelerative adjustment of the object is as fast and smooth as possible – thereby replicating the Stasis Engine's purpose of zero restitution.

4.4.1 Function 4.1: Calculate a force reduction curve based on the data collected, to predict force required.

Where two checkpoints provide a measure of *rate of change of* acceleration, this is sufficient data to produce a set of predictions for the next *n* number of checkpoints, given data on atmospheric resistance. However, the purpose of the Stasis Engine remains to decrease the speed of an object – with sequentially decreasing *rate of change of* acceleration. Therefore, a force *reduction* curve can be modelled, that remains *just* above the object's speed at any sample taken on the curve, such that the object slows, without restitution. This function ensures the creation of the first decelerative force is with reference to the sensor data collected in *Function 2.1* and *Function 2.2*.

4.4.2 Function 4.2: Calculate the new appropriate force at the next fastest actuator update time.

To begin the physical slowing of the object, the actuator requires an instruction following its motion at constant speed. The next update given to it will be a variable change in speed and *rate of change of* acceleration, and will be the first stage at which the object's velocity is notably changed by the actuator.

Test Cases

Test No.	Function	Name	Description	Method	Outcome
15	4.1	Force Reduction Curve Creation	Using the data collected in *Function 2.1* and *Function 2.2*, alongside a general formula for a force reduction curve with a low 'jerk' (smooth *rate of change of* acceleration)(Hayati et al., 2010), produce a force reduction curve.	Creates a force reduction curve for most of the remainder of the object's motion, whereby, paired with *Function 5.2*, no further real-time processing is required.	The robot should prepare a sufficiently populated force reduction curve, with at least 10 entries for speed and acceleration required at different time samples.
16	3.2.1	Instant Retraction Calculation	Estimate the actuator speed required to match the object's speed at time of contact, taking	Use the data collected in *Test Cases 10, 11, 12* to form an estimation of speed to retract the	The robot should prepare an actuator speed of retraction congruent to the values calculated in *Test Case 13*.

			into account *rate of change of* acceleration at contact time.	pneumatic rod at, just before time of contact.	

4.3 *Objective 5*: Continue to variably decrease force till zero restitution achieved.

As the object remains in contact with the contact surface, the force reduction curve modelled in *Function 4.1* can allow for preprocessed results to be sent to the actuator to further retract the pneumatic rod with variable speed, at the quickest intervals that the actuator would accept such commands. Nearer the end of motion, when the interval updates are no longer relevant to the speed at which the object first met the contact surface, a generic force reduction curve can overtake the one created specifically for the object's movement criteria.

4.5.1 Function 5.1: Use the force reduction curve calculated to send pre-processed results to the actuator for each interval.

Within this function, the actuator is updated as quickly as programmable to ensure the smoothest deceleration possible.

4.5.2 Function 5.2: As the object nears zero velocity, adopt a predictable force reduction curve for the final end, since bounces may exceed microsecond oscillations.

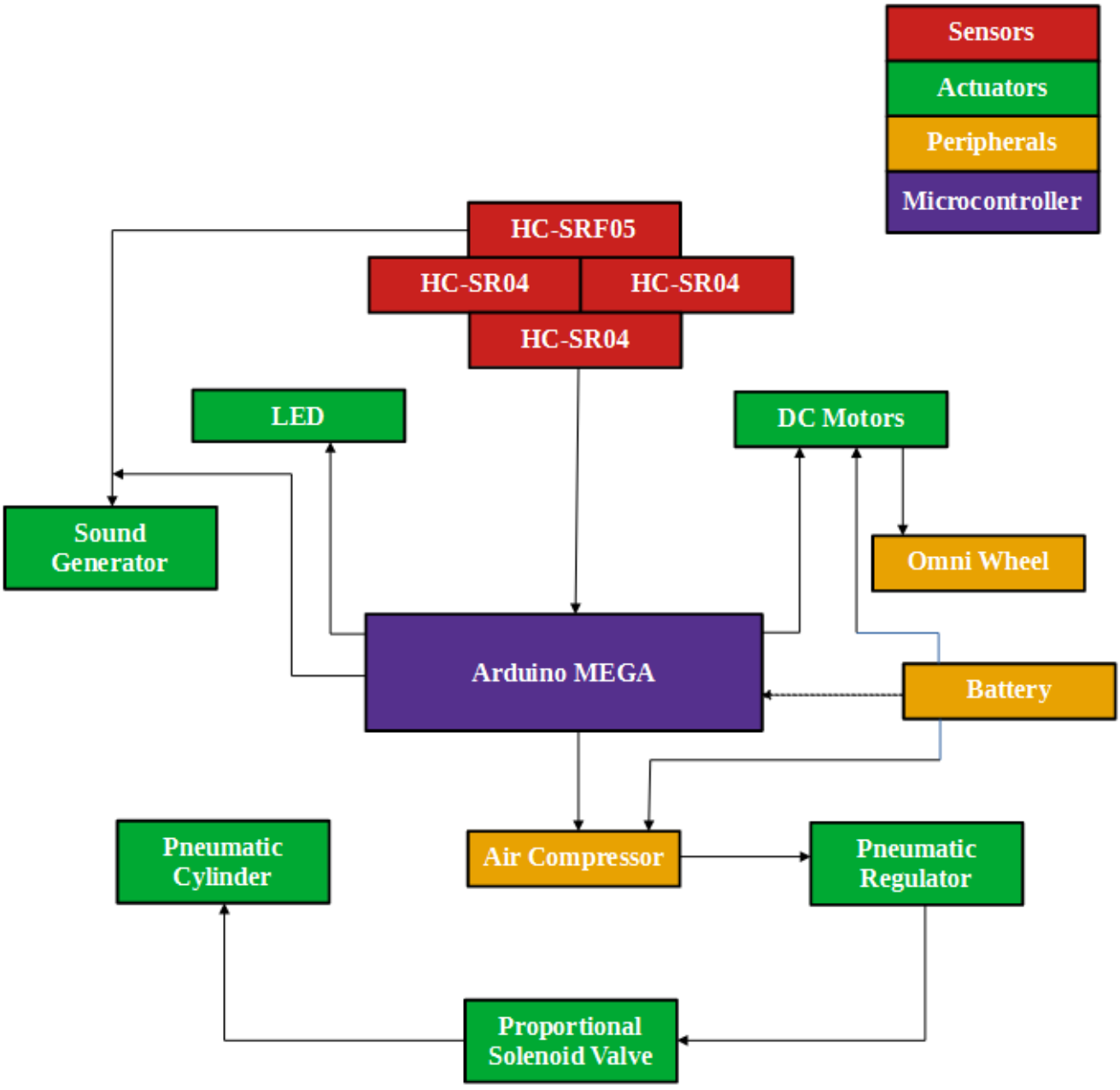
As the final portion of deceleration is reached, the potential oscillations of the object may exceed the actuator update intervals. Simultaneously, the object will reach a generic level of speed that no longer requires a specific force reduction curve, and so a preset alternative will be used instead, till the object comes to rest.

Test Cases

Test No.	Function	Name	Description	Method	Expected Outcome
17	5.1	Actuator Update	Using the force reduction curve created in *Function 4.1*, the actuator will continue to smoothly slow the object till a generic speed is reached, where eligibility for *Function 5.2* is unlocked.	As the object remains in contact with the contact surface, the actuator's *rate of change of* acceleration is variably updated.	The robot should variably slow the pneumatic cylinder as to prevent restitution.
18	5.2	Generic Force Reduction Curve	As the contact surface retraction slows to a predefined minimum, the specific force reduction curve is abandoned in favour of a generic alternate, since	The actuator is updated based on a preset generic force reduction curve till the object comes to rest.	The robot should finally slow the pneumatic cylinder according to a predefined force reduction curve.

			deceleration at slow speed does not require specificity in force.		

5 Logical System Diagram



6 Physical System Diagram

This section provides an overview of the final design of the robot, conjured in the planning stage with no real regard as to feasibility apart from that which was obviously improbable.

6.1 Main View

The Stasis Engine is built as a wide-based cubical system. *Figure 1* provides the first visual conception for the physical component of the robot. The upper lip of the chassis reveals the implementation of a lid. Within this central cavity lies the components necessary to the function of the robot. As a whole, the robot remains aesthetically dull. However, the frame sits low enough alongside the heavier components in the pneumatic cylinder and air compressor, to practicably fulfil its purpose without risk of toppling or being overcome.

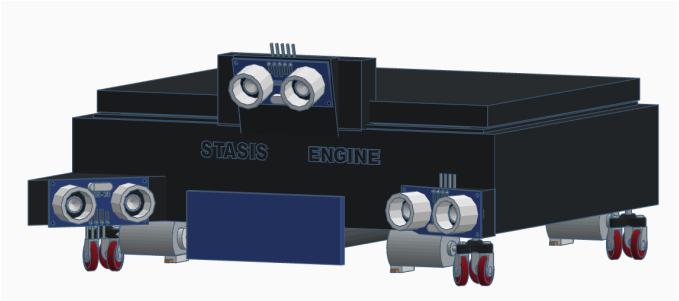


Figure 1: Main View

6.2 Front View

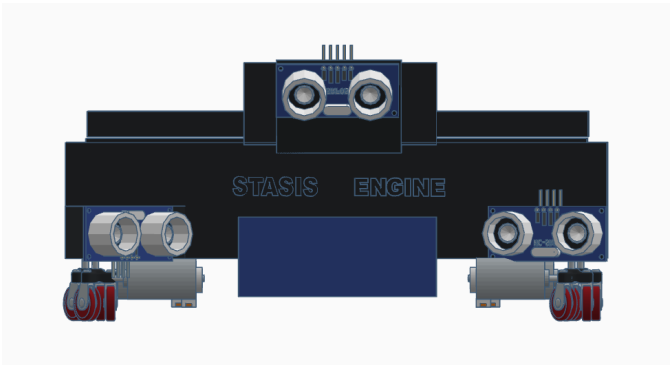


Figure 2: Front View

Figure 2 becomes the perspective of the object travelling to towards the robot. From this position, it's simple to see the progression with which the Stasis Engine receives distance readings. Of the three HC-SR04s, the primary is placed at the top, the secondary on the right, and the tertiary on the left.

6.3 Side View (Left)

Both the underside and rear of the Stasis Engine are available to view. The large grey cylindrical object at the underside is the pneumatic cylinder, from which protrudes a smaller grey shaft, the pneumatic rod. This will be explained in more detail with reference to *Figure 7(a)***.** Notice how all three frontal sensors are available to view in *Figure 3's* perspective.

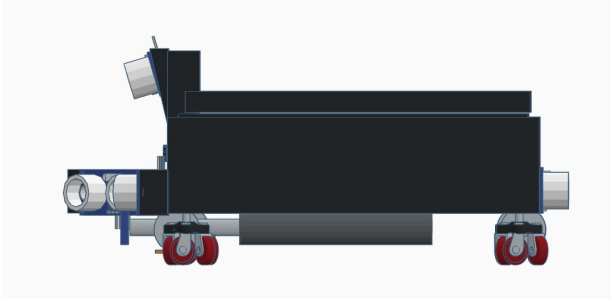


Figure 3: Side View (Left)

6.4 Side View (Right)

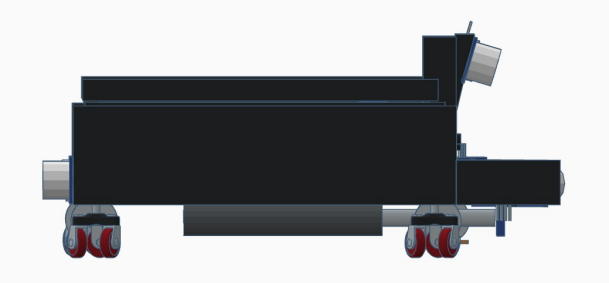


Figure 4: Side View (Right)

In *Figure 4*, it's immediately apparent that the pitch of the wedge hosting the right (*tertiary*) HC-SR04 is much greater than the left (*secondary*) HC-SR04. This means the path of convergence with the object will be at a much closer distance than with either of the two sensors.

6.5 Rear View

Figure 5 lays bare the Stasis Engine's rear, which hosts a single HC-SRF05 sensor, dedicated solely for *Function 1.2.1*. This is supplemented by the clear revelation of 4 DC motors attached to each wheel, which is explained in *Figure 7(a)*.

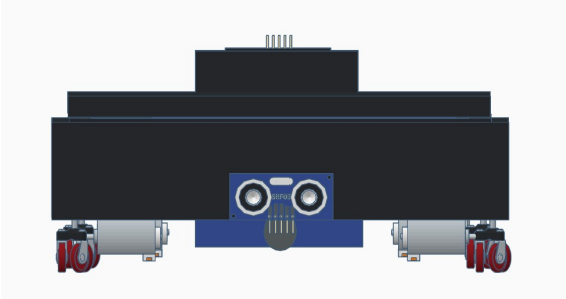


Figure 5: Rear View

6.6 Bottom View

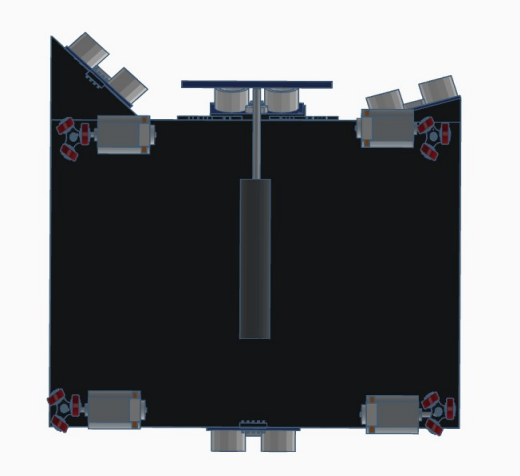


Figure 6: Bottom View

Figure 6 reveals the significant stroke of the pneumatic cylinder, which remains partially extended. This is a potential development flaw discussed later in. At this point, the pneumatic cylinder is neither fully extended nor retracted.

6.7 Wheel View

As we inspect *Figure 7(a)*, we can see the unconventional nature of the wheel – using a trio of standard wheels, connected to a DC motor (introduced in *Figure 5*) despite being incompatible with it within the scope of the components available in the 3D modelling tool.

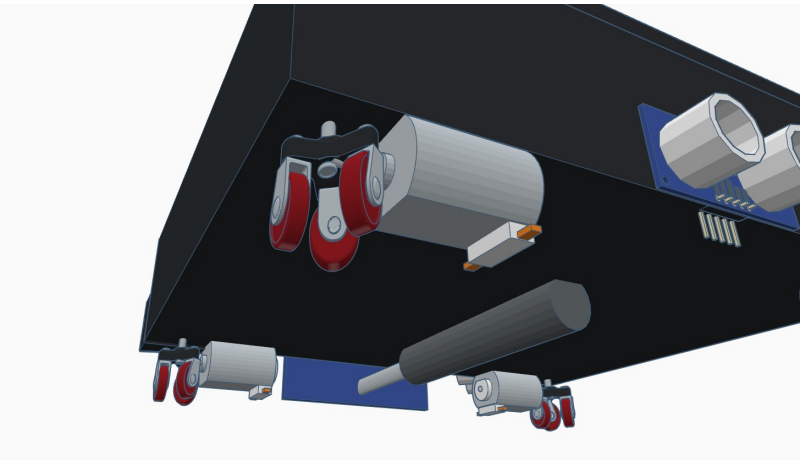


Figure 7(a): Wheel View

This is because Tinkercad lacks the default 3D model for omnidirectional wheels. This was the closest thing available in the library that replicates them. The correct rendition is viewable in *Figure 7(b)* – and this model is indeed compatible with individual motors to provide complete omnidirectional movement while the chassis remains static. No function of the Stasis Engine supports, or indeed requires complete omni-directionality.

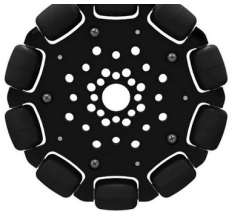


Figure 7(b):

4" Omni Wheel - https://uk.robotshop.com/products/4-omni-wheel

6.8 Pneumatic Cylinder View

As mentioned in *Figure 3*, the partial retraction is viewable here in *Figure 8*. I've left this in, as allowing the design to fully extend (where the maximum stroke for this particular pneumatic cylinder is 20cm) will require a potentially unnecessary rearrangement of the distance sensors. In all reasonable cases, a decelerative distance of 10cm will be likely be far more than enough to slow a reasonably weighted object to a stop – however this remains to be thoroughly tested for practicality. A detailed breakdown of the object's parametric considerations can be found in *Section 8.*

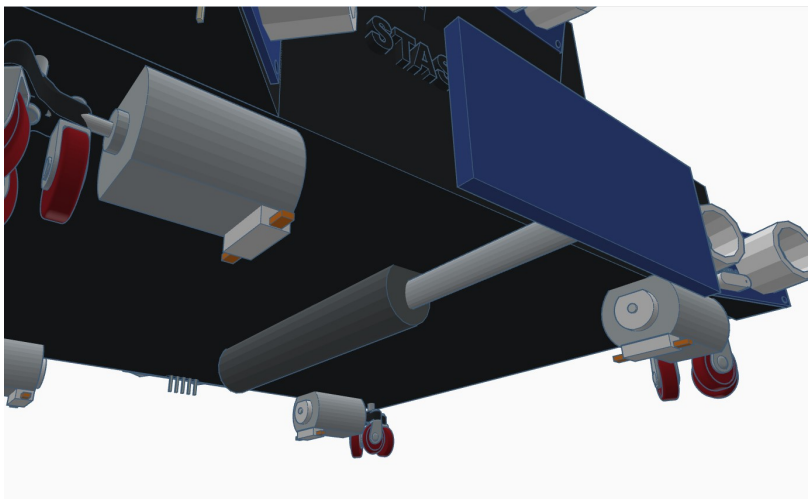


Figure 8: Pneumatic Cylinder View

The behaviour of the pneumatic piston will be controlled by the Arduino Uno R3 and a solenoid, wired into a breadboard alongside an air compressor or pressurised tank, and a 12V battery(?).
The distance measurements will feed into the Arduino, and a mathematical formula will decide what speed the solenoid should allow airflow into, and thus the piston should be retracted in order to constitute the property of zero restitution without the chance of multiple contacts, or too stiff and therefore restitutes.

6.9 Top View (Lidded)

Figure 9 focuses on the lidded property of the chassis chosen, first mentioned in *Figure 1*. This not only allows easy access during developmental stages, but also covers the valuable electronics within the central cavity.

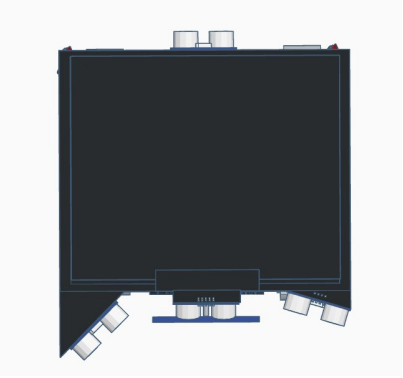


Figure 9: Top View (Lidded)

6.10 Top View (Unlidded)

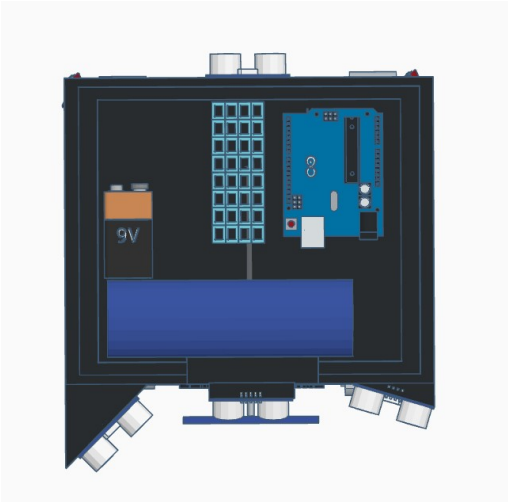


Figure 10: Top View (Unlidded)

In *Figure 10*, all the components within the central cavity are visible. Aside from the obvious, the large blue cylindrical shape resembles the portable air compressor required for this implementation. Previously, this was a CO^2 gas canister, however the idea was abandoned due to weight and safety considerations (CO^2 gas is often stored at 860psi, and a 3.15kg total of CO^2 requires a 10kg container gross weight (BOConline, 2023)).

7 Implementation

This section documents my build process, including only the most vital of failures, and disregarding all wiring, coding or functional issues except those which significantly disrupted. It also disregards learning and build time, which are briefly outlined in *Section 7.8*.

7.1 Build (v1)

I started with a disassembled 4WD Mecanum Robot Kit and an assortment of components, shown in *Figure 11*.

Figure 11: Disassembled Chassis

7.1.1 Chassis

The chassis was assembled with little difficulty. The end result would be a plastic base connected to four DC motors and wheels, extended vertically by another plastic base.

7.1.2 DC Motors

Despite aligning as much as possible, tolerances were quite generous considering the learner's design of the chassis. As a result, the affixations individually turned out slightly misaligned in several planes, which doesn't affect the basic function of the motors, but is worth noting for any possible result inconsistency that doesn't bear direct mechanical liability on the actuation system.

7.1.3 Omniwheels

The omniwheels were attached to the servo rod by way of screw friction. No threading exists on the rod, and therefore two small flat-point screws were inserted into designated holes on the wheel hub, torqued to an unmeasured but roughly identical specification for all eight screws on the chassis. These screws were positioned equidistant from a singular screw on the receiving port, thereby giving visual indication of a lack in torque, failure of screw threading (spinning) and therefore friction, or overloaded chassis (dead load).

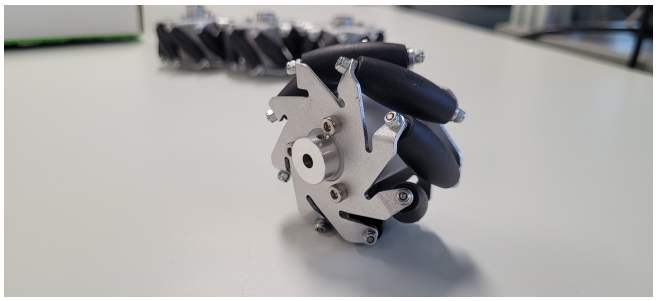


Figure 12: An example omniwheel, showing the protruding hub.

7.1.4 DC Motor Drivers

Noting that the general system was yet to be added, I opted to keep the L298N drivers on the topside of the lower floor, between either set of wheels. While the wiring style bore no actual effect on the function of the wheels, I chose to use an AWD configuration – the left and right set were paired to a single driver each. This forced me to explore a twist in the red and black wires of all four DC motors, which I later found was beneficial for structural integrity and reliability advantages

(Yang et. al., 2018, Mahmoud and Abdallah, 2008), and also allowed me to impermanently shorten the length of these cables for a tighter fit without resorting to removing wire sections. Interestingly, once both DC motors were attached to one driver after looping the standoff, a small tensile tension ensured stability of the driver.

7.1.5 Distance Sensors

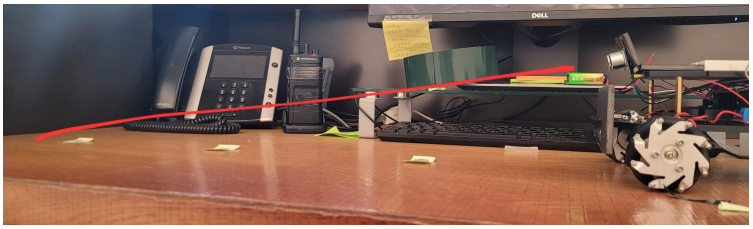


Figure 13: Red line marking the furthest distance detected (45cm)

As mentioned in *Function 2.1*, three stages of distance data were required. To implement this, the design shown in *Section 7.1* was followed stringently, and the foam board and chassis slits were used in tandem to provide a rigid surface for the distance sensors to attach, as in the design concepts. Then came the task of offsetting each distance sensor by a different angle. In this first iteration, no measurements were taken as to which point the sensor might detect an object in the line of its travel. As shown in *Figure 13*, an approximation was made as to the distance for which each sensor would be made responsible.

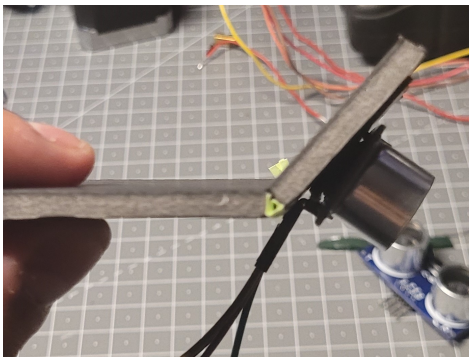


Figure 14: Post it usage to maintain angle

Approximations were taken by visual consideration of the angles and likely intersection points, and then a piece of well-measured foam board and adhesive was used to jut the distal edges out enough to form angles. This occurred at either lateral edge on the lower floor distance sensors. The upper floor distance sensor used a piece of foam board, slit at a certain point, to create a joint angle on which the distance sensor could stick. I added a post-it note rolled into the shape of a prism to keep the angle, and the weight of the distance sensor kept it from sliding out.

7.1.6 Stepper

I purchased a small 28BYJ-48 stepper from an electronics store while I awaited the delivery of my larger stepper motor. This was positioned on the upper floor, and attached to the lead screw using a shafter coupler, which also used friction identical to the flat-point screw design mentioned in *Section 7.1.3*. This was positioned vertically for testing as seen in *Figure 15*, but unfortunately didn't survive. This is mentioned later in *Section 7.1.10*.

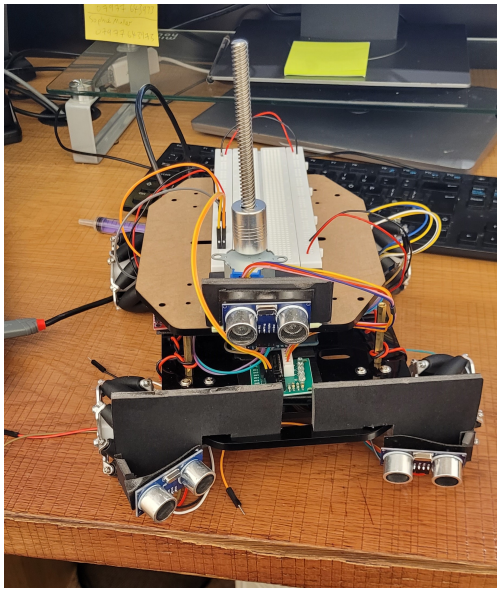


Figure 15: Vertical stepper

7.1.7 Breadboard and Arduino

Considering that this is a robotics project instead of a commercial product, I chose not to hide the wiring. Instead, I think the visible wiring adds to the aesthetic. Therefore, the breadboard occupied the upper floor, behind the upper distance sensor, while the Arduino was affixed between the two L298N drivers, viewable in *Figure 16*.

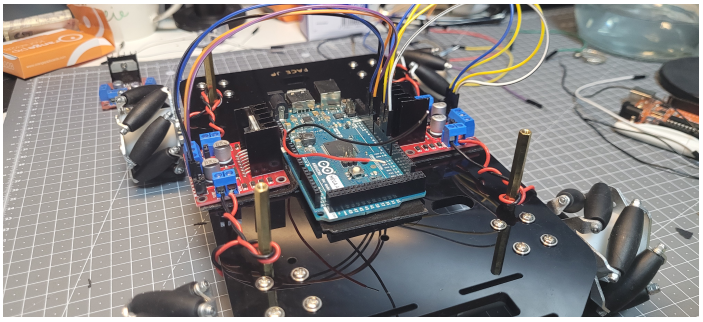


Figure 16: Breadboard placement between L298Ns

7.1.8 Master/Slave Cylinder System

Two 10ml syringes were available to me as partial alternatives for the pneumatic cylinder mentioned in the original design (*Section 14.1.2*). Silicone tubing was used to attach the syringes to each other. An advantage of this tubing is the innate damping property of silicone, which in this use case will reduce or completely negate reverberating vibrations on contact, which turned out quite useful for this application.

A basic master/slave cylinder system was implemented with one syringe on the topside of the lower floor, and the other on the bottomside of the lower floor. The bottom syringe was placed between the two L298N drivers on the lower floor, with its rod running perfectly between the two lower floor distance sensor apparatus, with the stroke tested such that it would neither collide nor interfere with the action of the HC-SR04s.

7.1.9 Wiring

I used basic jumper cables to connect the components to the breadboards, and in turn to power/gnd/pins. Where the jumpers were not long enough, I used basic wire to either extend a male-female connection or completely replace male-male wires.

The powered rails on the breadboard were both 5v, running off my laptop's USB port.
The ground rails on the breadboard were connected by a single black wire between them.
All components were connected at 5v. This was again due to awaiting a 12v battery in shipment.

7.1.10 Testing

I learned several things from this iteration, including the appealing visions of neatly connected wiring turning out to be an unsightly, jumbled mess without serious effort, circuit wiring conventions, the relative ease of joining/extending conductive metal and the dangers of overloading an actuator. However, I also weaned valuable lesson in the constantly shifting nature of a practical implementation. With no field experience, implementation will find itself straying further and further from design till it seats itself appropriately in practicality.

Some unusual wiring had to be added with respect to the limited equipment I had. The unusual requirements began with the distance I had chosen between the L298N drivers having to reach their power and ground wires all the way to the upper floor. I thought this might look like ‘ribs’, adding to the aesthetic of the robot. However, these turned out to be significantly problematic. Not only did the wiring look absurd, as shown in *Figure 17*, ote that the upper floor is detachable by unscrewing, and that this is a student project requiring comprehensive demonstration and frequent alteration. The addition of these wires from lower floor to upper floor meant that every time I needed to work on the master/slave cylinder system, I’d had to disconnect these wires from both sides, and also the interwoven HC-SR04 wires.

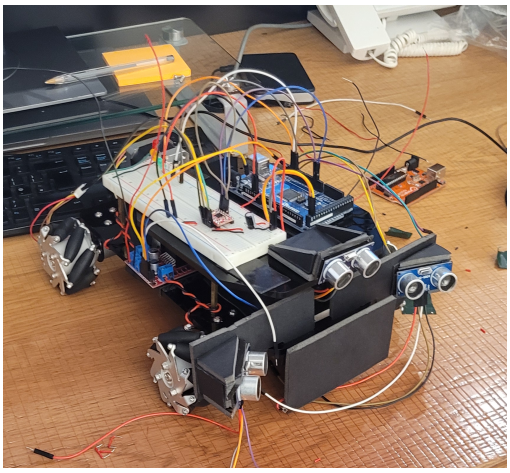


Figure 17: Messy wiring

Similarly, the HC-SR04s required female jumpers, which don’t keep their shape when deformed, as normal wires do. This led to an underhang which in my mind again added to the aesthetic, but in reality looked supremely messy. This came in tandem with another issue. These female connectors required a sizeable portion of the wire to maintain its rigidity via a plastic sleeve, and the further wire couldn’t be tensioned to hold any useful angle right out of the sleeve without risking:

- Damage to the sleeve
- Damage to the HC-SR04 male port
- Detachment from the adhesive due to tensile force
- A suitable point at which tension could be maintained further down the wire.

For these reasons, despite relatively meticulous assurance that there would be no impedance from the wheels or ground (a generous clearance was given), and that I did not take into account wire rigidity, the jumper wires laid on the floor for the majority of this setup, sometimes interfering with the wheels even at idle, and obviously during testing phases too. This obstructive looseness can be seen in *Figure 18.*

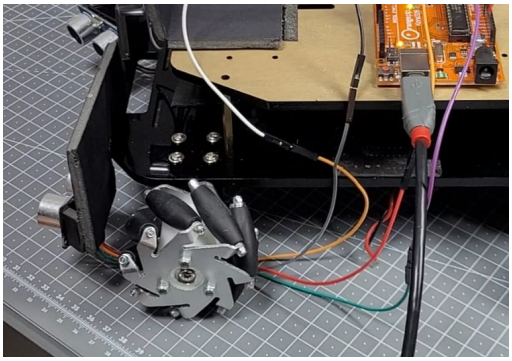


Figure 18: Loose, obstructive wiring from HC-SR04 to Arduino and breadboard

Once wiring was done, noting all improvements that would have to be made, I tested all components. The LEDs worked with resistor connected to ground. The HC-SR04s worked with TRIG and ECHO pins to begin filling out the Arduino R3 Uno, and I received good results on the serial monitor. The stepper worked, albeit too slowly for the application – a full rotation once every three seconds at full speed wouldn’t scratch the surface for how reactive counter-motion actuation would have to be.

The wheels also caused a critical concern at this juncture. The power supplied to them lacked either voltage or stability, as the wheels would spin at different speeds, often delayed, and often turning on and off throughout connection. When I purchased a 9v battery, both of these issues ceased to be.

At the final section of testing, I attached the shaft coupler and lead screw to the stepper. This proved to be too much for the tiny stepper, and it blew up under the load, which was an interesting experience.

7.2 Build (v2)

Learning from the mistakes of the first iteration, and with several ideas on how to produce a significantly better prototype, I tore down most of the components, leaving just the wheels and DC motors attached. I then re-evaluated the build to include a neater, tighter design while maintaining space for continuous work.

7.2.1 DC Motor Drivers

Despite temporarily testing switching the position of the L298N drivers to the bottomside of the lower floor, I found that there was still enough space after construction of the support structure mentioned in *Section 7.2.3* that I could maintain the visual aesthetic appeal of the drivers on the topside.

7.2.2 Stepper

As the component with the most significant individual weight, I felt it best suited at the rear of the lower floor, to provide a solid base from which the robot could accept an incoming force. Unfortunately, its addition in that position turned out disruptive to the placement of the upper floor. This was discovered after its addition with adhesive. Essentially, the rear end of the upper floor just slightly overlapped with the front edge of the stepper, causing a lift and non-flat angle of the upper floor. This wouldn't generally be an issue, but the standoffs between chassis floors required four screws. Why I couldn't immediately remove and reposition the stepper more appropriately is a consequence of the poor quality of the adhesive, explained in *Section 7.4.*

7.2.3 Master/Slave Cylinder System

The lower floor underside syringe was returned to its normal position between the lower two HC-SR04s. The lower floor topside syringe faced some major changes due to the addition of the new stepper.

Because of the weight and build quality of this stepper, I managed to attach the shaft coupler and lead screw while the stepper rod was horizontal, therefore allowing an actuation movement parallel to both floors. However, this would require the master syringe to attach to the lead nut through:

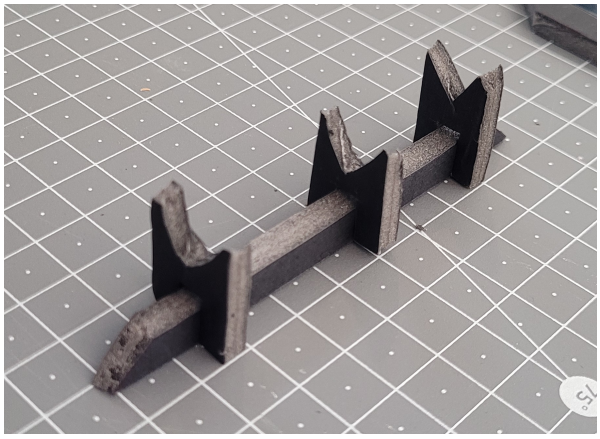


Figure 19: Supporting structure for syringe

1) a raised platform that would bring the master syringe adjacent to the lead screw.
2) some sort of improvised connecting rod.

Because of my lack of *strong* materials to work with (foam board and poor adhesive), this very quickly became a gargantuan task that was the main focus of the next iteration in *Section 7.3*, but I made an attempt during this build.

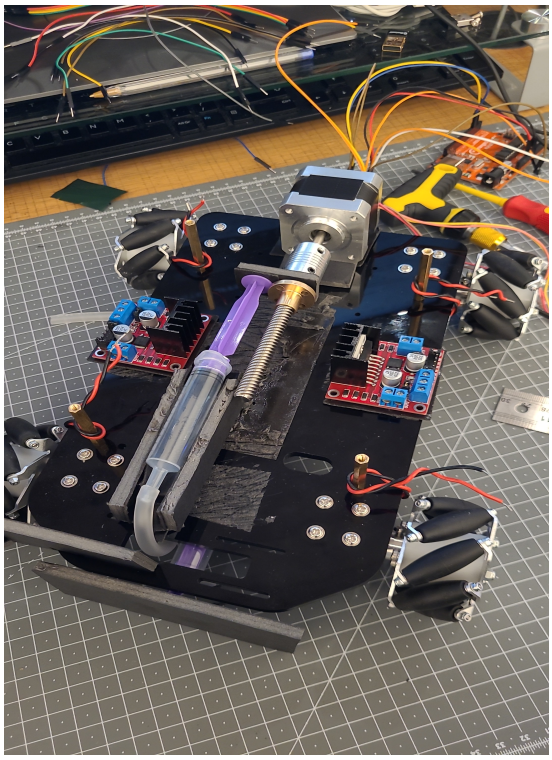


Figure 20: New master/slave system with stepper

I began with the construction of a strut intended to hold up the syringe. Three pieces of foam board were cut out, matching the height required to hold the 2cm diameter syringe to a height matching the lead screw. They were carved to fit the circular barrel of the syringe at three points, maintaining a maximum of 1cm depth at the lowest point of the carve. A longer supporting 'beam' was threaded through the lower middle of every piece, which maintained a height of 1cm through the new structure till tapering off at either end with a triangle. This structure can be seen in *Figure 19*.

Two larger pieces of foam board were added along the sides of the structure to increase strength further. Adhesive was added at the bottom, and in the crescents of the foam pieces to hold the syringe. I then measured the diameter of the lead nut's boss, and carved a hole into a new connecting rod between the syringe head and lead nut. This was secured with adhesive on both sides. The result can be seen in *Figure 20*.

7.2.4 Distance Sensors

These were returned to their normal positions with significant improvements.
The lower floor sensors suffered some adhesive issues since the last iteration, and the upper floor sensor did not keep a stable angle. Both designs were completely rebuilt.

At this point, I re-established the requirement of three points of detection.

Stage 1: 45cm
Stage 2: 30cm
Stage 3: 15cm

Though arbitrary, these fit well within the scope of the distance sensor's ranges, and assured a constant was available in distance-time calculations. The proximity of the final measurement would ensure a precise response was available to be given.

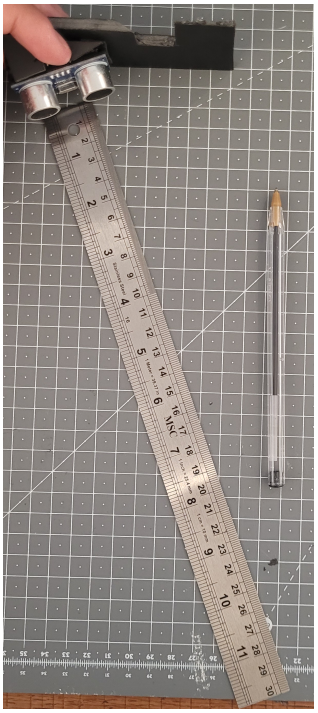


Figure 21: Stage 2 HC-SR04 showing detection at 30cm

To achieve these joint angles, and keep them this time, I re-measured the intersections and cut a more precisely finished series of pieces.
On the lower two sensors, these were slanted on one edge to match the angle of the distance sensor relative to its support surface. Then a triangular roof and bottom panel, matching the specification of the structure to significantly increase strength, and stuck them all together with adhesive matching the shape of the pieces, resulting in a clean, strong finish, shown in *Figure 22*.

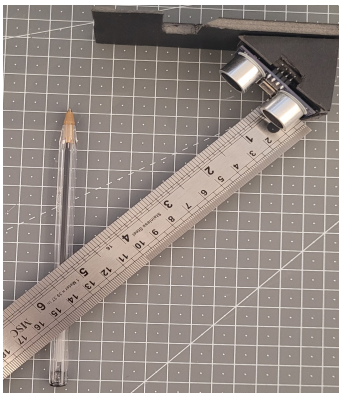


Figure 22: Stage 3 HC-SR04 showing detection at 15cm

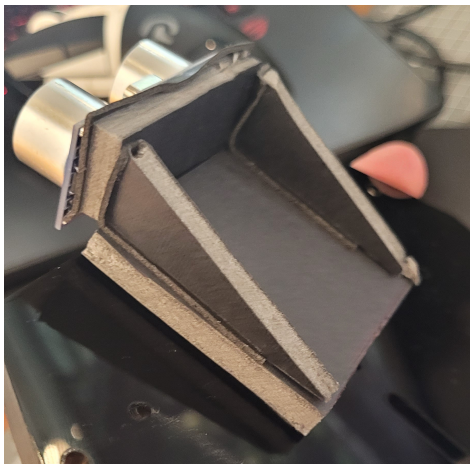


Figure 23: Anchors creating a fixed angle on upper floor HC-SR04

The upper floor sensor required a fixed angle, and I achieved this by affixing two precisely cut triangular anchors behind the foam board, shown in *Figure 23*. The end result was a far stronger structure on both upper and lower sensors.

7.2.5 Testing

This was the first stage in which I actually tested the entire range of the stepper, including speed and microstepping abilities. Though it was a success, the structure visibly moved during each extension and retraction of the syringe, which was a dangerous sign of weakness. One particular trial testing which default position was required tore through the structure supporting the syringe. The range accidentally driven to led the lead nut entirely out of the lead screw, and since both were made of metallic alloy, and the supporting structure was made of foam board & adhesive, the latter quickly gave way.

Unfortunately this failure would be true of any sort of structure attempt I'd make purely out of foam board and adhesive, but the catastrophic damage that occurred in this iteration warranted another complete rebuild.

7.3 Build (v3)

For the final iteration within this project, I redesigned the supporting structure again to support a much greater (relatively) force, with emphasis on structural integrity throughout the movement. I also redesigned the master/slave system to include an additional syringe. To accommodate this new design, I increased the gap between the lower and upper floor. Finally, I swapped out the wiring for a more flexible design.
Everything else remained the same.

7.3.1 Stepper

Noting the issues with the previous position interfering with the upper floor, I added an extension to the rear of the chassis to accommodate a portion of the stepper, with a weight distribution ratio of 60:40 favouring the extension. This was of course supplemented by the adhesive, and the extension stretches well into the lower floor topside, with a generous application of adhesive acting as a counterweight. This came with the auspicious side effect of more space on an increasingly limited working area.

Figure 24: Stepper extension

7.3.2 Master/Slave System

After the catastrophic failure of the previous design, and the addition of an extra syringe, I thought pretty hard about the final design. This involved two stages:

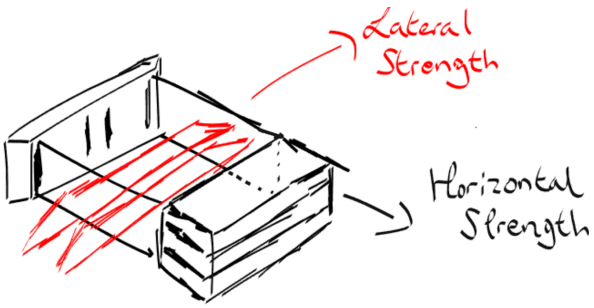


Figure 25: Primary design

The first design consisted of a piece of foam board clamping down on the two syringe barrels, connected by longer pieces to a reinforced stack of foam board piece layers. As shown in *Figure 25*, the middle would have two supporting 'beams' threading through to either side, tapering off into a triangle just as in *Section 7.2.3*.

The second design was more comprehensive, and though it removed the lateral strength aspect, it doubly reinforced the rigidity of the structure over all. Interestingly, this design far more resembles the 'train buffer' visual given in *Section 3.1*. This consisted of two anchors just as in *Section 7.2.4*'s upper floor distance sensor, affixed to a vertical stack of foam board pieces, then a final horizontal stack of the same, thereby significantly increasing resistance to movement, while allowing ample space for the syringes. This is shown in *Figure 26.*

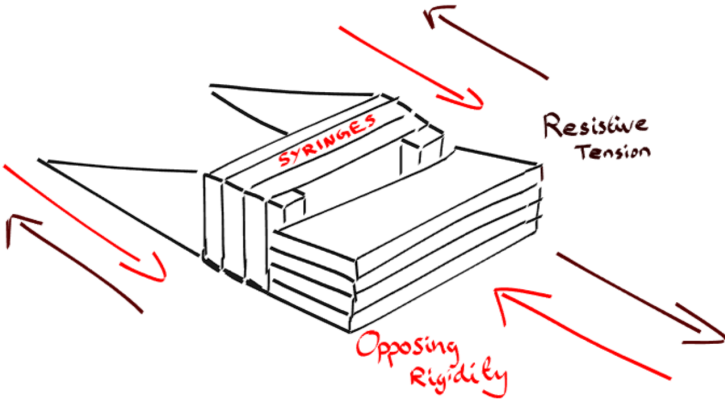


Figure 26: Chosen Design

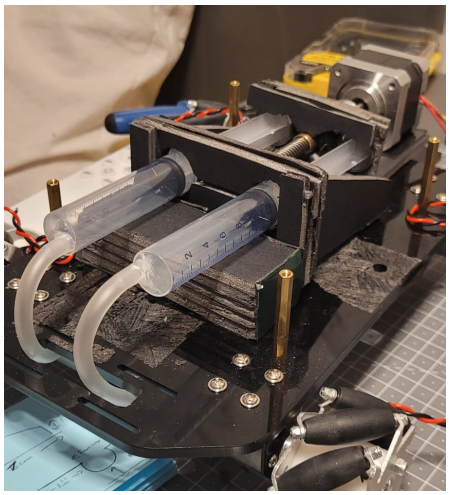


Figure 27: Final build on lower floor topside

On implementation, I found that the weaknesses of the adhesive and foam board shone brilliantly through any design I tried to lay down – while nothing moved as in the previous iteration, over the days of building this the adhesive failed in many places, causing me to add generous amounts just to compensate for the failure of the old strips. While I don't deny my own design could have been poor, I feel as if replacing the adhesive with simple superglue or even normal duct tape might've salvaged much of the time I spent repairing. In fact, by the end of the build I had applied superglue in several places, and much to my satisfaction, they stuck! The final build is shown in *Figure 27*.

7.3.3 Wiring

As I proceeded into the final connections, I discovered several tips for maintaining the all-important aesthetic. This culminated in my usage of colour-coding as follows:

Blue – **Data/Pin**
Orange – **Binary**
Red – **Power**
Ground – **Black**
Yellow – **Miscellaneous**

Combined with purely visually appealing techniques that reduced the jumbled mess seen in *Figure 17* to something resembling wire management, seen in *Figure 28*.

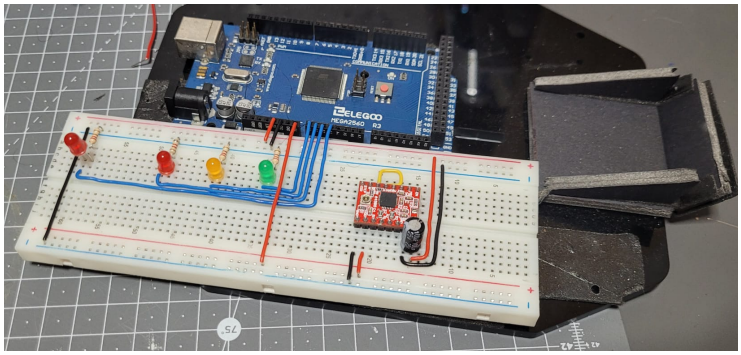


Figure 28: Colour-coded, neat wiring

7.4 Failures

One of the major reasons for the iterations that I neglected to mention, was the unavailability of the components requested in Jan 2024. Extra time was wasted on trialling inferior components, and the final build today contains a rough, improvised version of the robot I'd planned to build. This is mentioned later in *Future Developments*.

Two of the most vital parts of this build were the foam board, that maintained structure and kept components securely in place, and the adhesive, which bound the components to the foam board, and the foam board to the ground. These were both bought at my expense. However, they both turned out incredibly lacklustre, and left me wishing I hadn't maintained optimism in their use on purchase.

Despite switching from penknife, to Stanley knife, to scalpel, to hacksaw, carving through the foam board was consistently an unpleasant affair. If any of my blades were to move in a line that was non-straight, the surface of the foam board would bundle thickly beneath it within centimetres, causing a nasty finish and generally ruining the piece altogether. Cutting through all the way required several tries to get the pressure right, and then it would risk the working surface below. I ended up using a reverse grip with the blade pointing away to keep heavy, even pressure without losing precision. Then it would be a 50/50 game of whether a wad of foam would be ripped out alongside your stroke, rendering the piece as useful as a piece of paper.

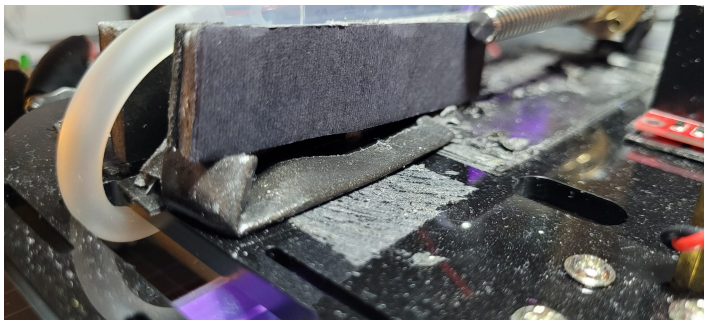


Figure 29: Adhesive failed due to time away

The adhesive tape is subject to a similar review, where a blade could only cut it if it were sliced through the covered back. On application, it held, but my builds consistently, and frustratingly failed *every few days* (*Figure 29*). The adhesive somehow lost most of its strength over a very short period if applied to an uneven surface like the back of a component. However, when applied to a clean smooth surface like the chassis, it stuck on as if it had come with the chassis (*Figure 30*). No amount of soapy water, friction, scratching, or nail polish remover did the trick, and therefore I will have to present with an… unpresentable chassis.

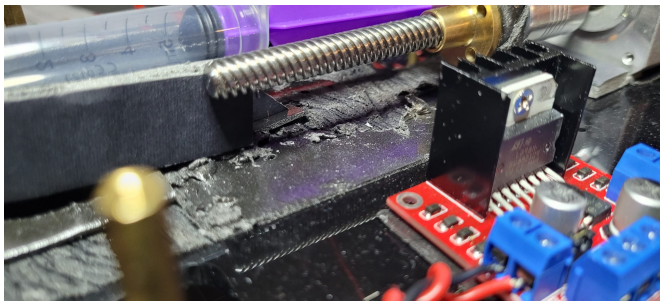


Figure 30: Adhesive stuck irreparably to chassis

Beyond the obvious, there was catastrophic damage to the 28BYJ-48 mentioned in *Section 7.1.6* caused by overloading, and I suffered a failure on one L298N driver and the original breadboard (the metal backing came out when prying it from the chassis).

7.5 Objective 1 Removal

Another major failure came in my inability to implement objective 1. Working within limited, congested space leaves very little room for a distance sensor to accurately detect an object. Further, looking at the recommended surface area for the object to be detected (0.5m^2), it resembles much more a common piece of furniture or bed sideboard than a palm-sized ball. This objective was created with the intent of portability, and a foray into a potential future where the Stasis Engine would be required to work with dynamically positioned objects. However, realism is also lost whereby I've no complex mathematics to work out how exactly to move the Stasis Engine such that it'll fall in line with the object path unless it's already positioned so perfectly that a simple rotation would allow the robot's contact surface to be in line.

7.6 Component Removals

The initial design of the Stasis Engine involved a complex array of components including a hydraulic cylinder and several related parts (*Appendix Section 14.1.2*). This was cleared as a design choice, but I was later suggested an alternative better suited for a student project, which came in the form of a master/slave system using syringes. Therefore, most of the components that I've labelled as removed in implementation were intentional, and were done irrespective of further plan or failure in execution.

7.7 Version Control

Throughout this project, I used a version control system to keep track of all reports, documents, notes and lesson overviews.

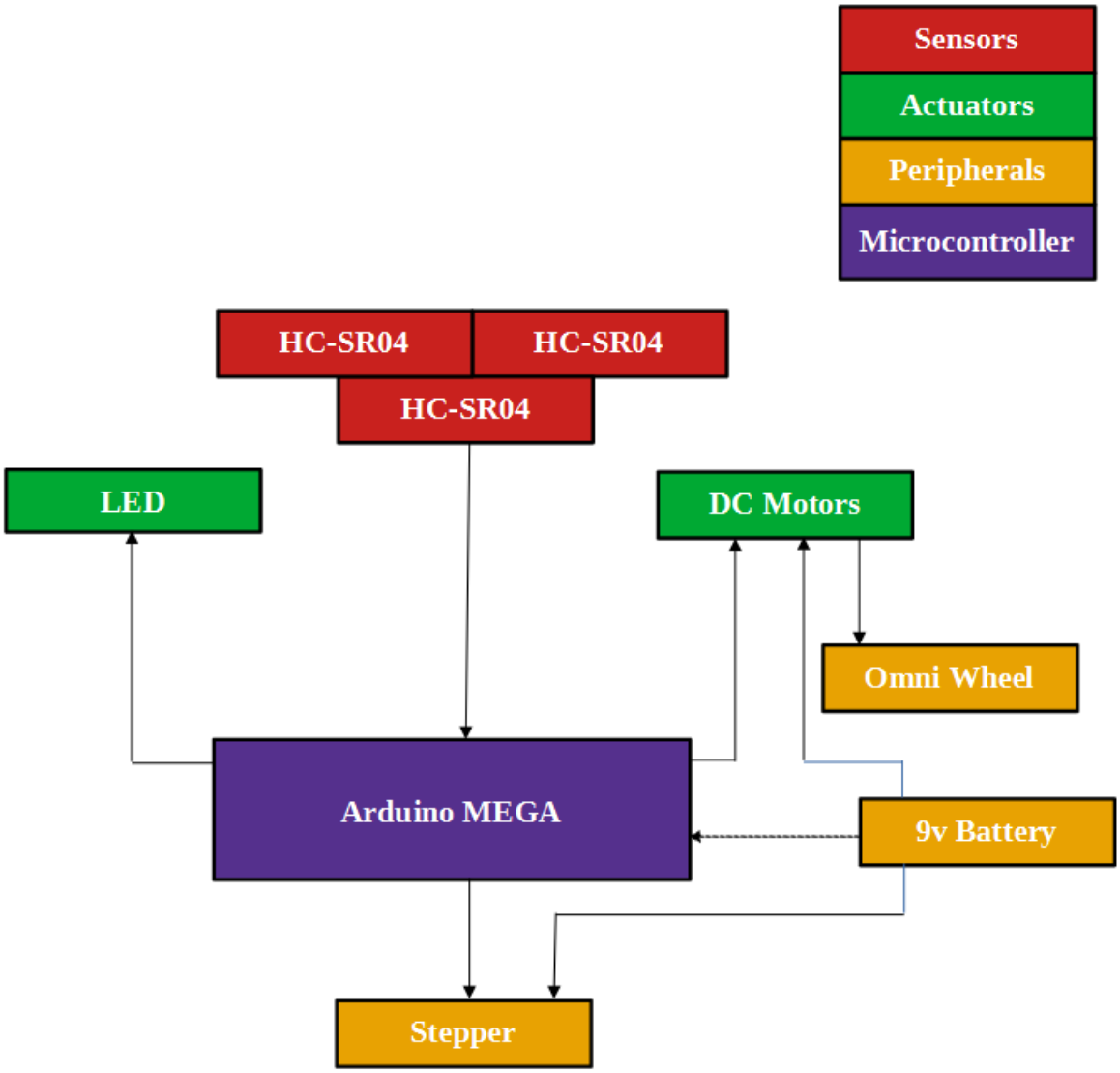
This was cygin64 for the Windows environment, paired with git, and uploaded to bitbucket. As of now (22/05/24), there have been 94 commits since Jan 2024.

7.8 Video Footage

I began recording the work sessions for PDE3413 on 27th March 2024. The total recorded length of video I have is 1433 minutes, or 23.8 hours. This is specifically work on the physical robot, and does not include any work done before 27th March 2024 (2 months prior to now), nor report-writing time, which was not recorded.

This footage captures my real-time learning process, and the slow but inevitable construction of one of the many ideas that sprung to mind at the behest of Mr. Hall many months ago, which I think is a fascinating concept that deserves a small section in this report.

7.8 Physical System Diagram



8 Results

I wasn't able to implement the Stasis Engine's design – it proved too difficult. However, this report serves as a guide to a certain point not far off its inception.

The main issue I found at end state testing was the lack of pressure in the master/slave system. When actuating the stepper, the larger slave cylinder always produced an unpredictable delay as the air was physically pushed through the master barrels and tubing. This flexibility had no place in the application required, which was intended to work with millisecond timing.

As such, I could never get around to the complex math, because the variables in displacement speed threw off every calculation I tried.

Test Cases

Test No.	Function	Name	Description	Expected Outcome	Actual Outcome
6	2.1	Primary Distance Check	Establish a distance and time checkpoint at the first point of convergence with the primary sensor's FoV (field of view).	The *primary* HC-SR04 should actuate, receiving *primary* distance data from the motion of the object.	The *primary* HC-SR04 receives distance data, marked by a green LED.
7	2.1	Secondary Distance Check	Establish a distance and time checkpoint at the first point of convergence with the secondary sensor's FoV. Follows ***Test Case 6*.**	The *secondary* HC-SR04 should actuate, receiving *secondary* distance data from the motion of the object.	The *secondary* HC-SR04 receives distance data, marked by an orange LED.
8	2.1	Tertiary Distance Check	Establish a distance and time checkpoint at the first point of convergence with the tertiary sensor's FoV. Follows ***Test Case 7*.**	The *tertiary* HC-SR04 should actuate, receiving *tertiary* distance data from the motion of the object.	The *tertiary* HC-SR04 receives distance data, marked by a final red LED.
9	2.2	Data Check	Verify that all the data was correctly received – also finds a vital use case in test failure scenarios.	The robot should successfully receive all data from the three HC-SR04s, then turn on an LED signifying data collection has been	The robot should successfully receive all data from the three HC-SR04s, then turn on an LED signifying data collection has been successful.

				successful.	
10	2.2.1	Average Speed Calculation	Calculate (*and store for **Test Case 11***) the speed of the object at each of the three checkpoints, using object distance and time. Collate into an average speed metric.	The Arduino should prepare an average calculation using the data collected.	The Arduino should prepare an average calculation using the data collected.
11	2.2.2	Acceleration Calculation	Calculate (and store for ***Test Case 13***) the rate of change of speed between each checkpoint, in order to instruct the actuator to deliver an opposite, similarly decelerative force.	The Arduino should prepare two values for the rate of change of acceleration between the three checkpoints.	The Arduino should prepare two values for the rate of change of acceleration between the three checkpoints.

Test No.	Function	Name	Description	Expected Outcome	Actual Outcome
12	3.1	Time of Contact Estimation	Using the data collected from the sensors, estimate the time whence the object will meet the contact surface.	The robot should prepare an estimation for time of contact, and store it for future use.	
13	3.2.1	Instant Retraction Calculation	Estimate the actuator speed required to match the object's speed at time of contact, taking into account *rate of change of* acceleration at contact time.	The robot should prepare a speed and acceleration required at the time of contact.	
14	3.2.1	Initial Retraction Distance	Use a small, predefined distance within which the actuator's retractive speed remains constant for up to 100 milliseconds (subject to change), to allow for any uncertainty in previous calculations introducing a flaw in the	The robot should prepare a short distance to be able to apply fully the force reduction curve mentioned in ***Test Case 15***.	

			time of contact.		

Test No.	Function	Name	Description	Expected Outcome	Actual Outcome
15	4.1	Force Reduction Curve Creation	Using the data collected in *Function 2.1* and *Function 2.2*, alongside a general formula for a force reduction curve with a low ‘jerk’ (smooth *rate of change of* acceleration)(Hayati et al., 2010), produce a force reduction curve.	The robot should prepare a sufficiently populated force reduction curve, with at least 10 entries for speed and acceleration required at different time samples.	
16	3.2.1	Instant Retraction Calculation	Estimate the actuator speed required to match the object’s speed at time of contact, taking into account *rate of change of* acceleration at contact time.	The robot should prepare an actuator speed of retraction congruent to the values calculated in *Test Case 13*.	

Test No.	Function	Name	Description	Expected Outcome	Actual Outcome
17	5.1	Actuator Update	Using the force reduction curve created in *Function 4.1*, the actuator will continue to smoothly slow the object till a generic speed is reached, where eligibility for *Function 5.2* is unlocked.	The robot should variably slow the pneumatic cylinder as to prevent restitution.	
18	5.2	Generic Force Reduction Curve	As the contact surface retraction slows to a predefined minimum, the specific force reduction curve is abandoned in favour of a generic alternate, since deceleration at slow speed does not require specificity in force.	The robot should finally slow the pneumatic cylinder according to a predefined force reduction curve.	

9 Bill of Materials

Item No.	Model	Link	Quantity	Price	Total
1	HC-SR04 Distance Sensor	https://handsontec.com/index.php/product/hc-sr04-ultrasonic-ranging-module/	3	£0.71*	£2.13
2	HC-SRF05 Distance Sensor	https://www.ebay.co.uk/itm/152354614958	1	£3.85	£3.85
3	4” Omni Wheel	https://uk.robotshop.com/products/4-omni-wheel	4	£10.82	£43.28
4	TR20X60S Pneumatic Cylinder	https://britishpneumatics.co.uk/collections/pneumatic-cylinders/products/airtac-tr20x60s-guided-pneumatic-cylinder	1	£53.60	£53.60
5	PS-1270 12V Battery	https://batteryinthecloud.com/products/ps-1270	1	£19.85*	£19.85
6	Arduino Mega 2560 Rev3 Microcontroller	https://uk.robotshop.com/products/arduino-mega-2560-microcontroller-rev3	1	£44.68	£44.68
7	Single Breadboard	https://uk.robotshop.com/products/breadboard-single-panel	1	£4.95	£4.95
8	VIAIR 12V 120psi Air Compressor	https://uk.robotshop.com/products/viair-12v-120-psi-air-compressor	1	£105.00	£105.00
9	RS PRO G ¼ Regulator	https://uk.rs-online.com/web/p/pneumatic-regulators/2351139	1	£29.54	£29.54
1	G209A405S0X00H1 Proportional Solenoid Valve	https://www.emerson.com/en-gb/catalog/proportional-flow-control-valves/asco-sku-g209a405s0x00h1-en-gb	1	£171.09	£171.09
11	9904-120-18105 DC Motor	https://uk.farnell.com/allied-motion-premotec/9904-120-18105/servo-motor-12vdc-3840rpm/dp/147875	4	£96.67 (exc. VAT)	£386.68
12	PIEZO Sound Generator	https://uk.robotshop.com/products/piezo-sound-generators	1	£1.95	£1.95
13	Triple Output LED RGB - SMD	https://www.sparkfun.com/products/7844	3	£0.44*	£1.32

14	AWG24 – 3m Black Silicon Wire	https://uk.robotshop.com/products/black-silicon-wire-awg24-3m	1	£1.21	£1.21
15	AWG24 – 3m Red Silicon Wire	https://uk.robotshop.com/products/red-silicon-wire-awg24-3m	1	£1.21	£1.21
16	AWG24 – 3m Blue Silicon Wire	https://uk.robotshop.com/products/blue-silicon-wire-awg24-3m	1	£1.21	£1.21
					871.55

*USD to GBP conversions accurate at time of creation.

Comments:
This is certainly prohibitively expensive, however some more expensive components were chosen in favour of more technical datasheets. The realistic cost of this report's interpretation of the Stasis Engine is likely *less than* half this figure. Following my research, I leave it to the lecturer to confirm or deny specific component choices and configurations.

10 Project Plan

Green tasks are easy.

Orange tasks are moderately difficult.

Red tasks will require most of my attention.

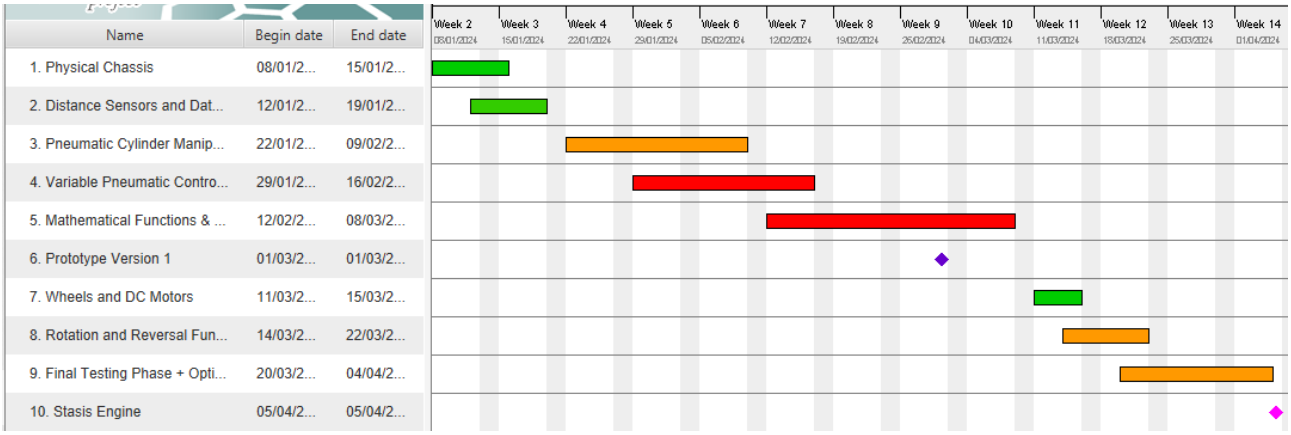


Figure 31: January 2024 Gantt Chart

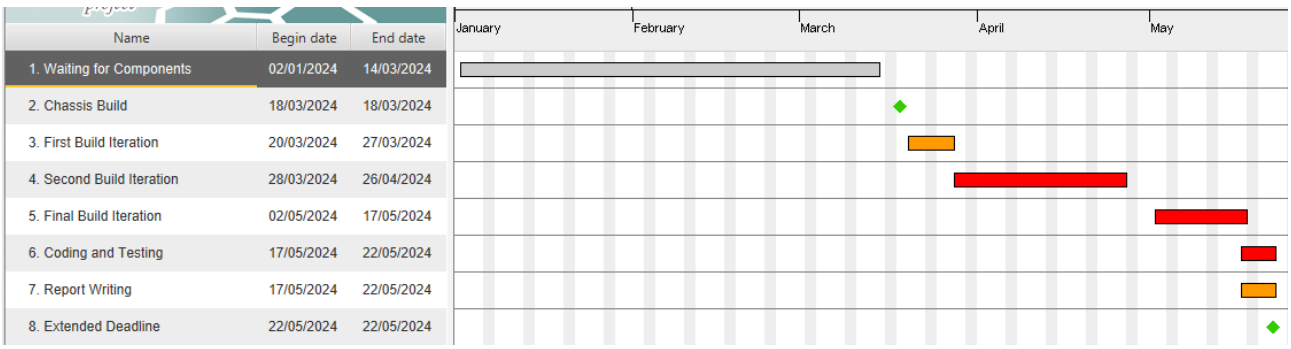


Figure 32: May 2024 Gantt Chart

11 Future Developments

Several developments that remain strong candidates for addition in future applications are highlighted here. Among these are some considerations that were closer to inclusion in the project than others, but were discarded due to budgetary, academic or algorithmic limitations. Others are purely conceptual.

Mechanical tilt sensor:
This sensor would find exceptional application in direct vertical free-fall, as a naturally falling object rarely ever remains static after exerting all its gravitational potential energy. This was part of the original design of the Stasis Engine, but was discarded due to the application of smooth motion reduction of *vehicles* being generally incompatible with vertical motion.

Retractable contact surface:
Currently, the contact surface is designed such that only a limited amount of air can leave the pneumatic cylinder cavity, after which the contact surface will collide with the robot chassis. An option to remedy this can be attaching the pneumatic cylinder further ahead underneath the robot, such that full extension and retraction is allowed. However, the full extension will interfere with the function of the distance sensors. This is a developmental flaw, but I've decided to keep it as it remains a trade-off between a more complex and obviously messy configuration where the sensors are extended far enough ahead that the contact surface won't interact with them, and preventing any further loss of aesthetic. Following, a further remedy that requires no repositioning of sensor nor actuator, is a retractable contact surface. In this case, some sort of actuator or structural component that allows 90-> 180 degree bend and re-erection would be suitable.

Rear braces/feet:
For the larger, commercial applications of this robot, accepting enormous amounts of mass in an effort to slow, would inevitably cause movement regardless of the brakes installed. A simple remedy s a pair or trio of extendable feet, which correctly moves away from brake improvement into focus on load-bearing limitations. This would increase overall stopping power through a rigid connection to the ground, overriding the brakes and would be a crucial addition to this sort of application, since although the portability of wheels are a requirement in road-work, the total mass of the Stasis Engine could still be overridden due to the property of wheels to assist movement. Lowering the chassis centrally and retracting the wheels entirely could create an incredibly powerful and capable vehicle. Much like a tornado research van, that roots itself in one spot to maximally negate the excessive forces acting upon it, the worst case with this type of addition would be destructive damage to the Stasis Engine itself, sacrificing its integrity for the safety of those behind it – which when considered from a human perspective, is not all that bad.

Oleo-pneumaticism:
In industrial-use, daily application, oleo-pneumatic cylinders have the added advantage of constant lubrication of the moving parts within the sole purposeful actuator of the robot. This remains a fundamental within existing mechanisms that serve a similar purpose to the Stasis Engine – most notably the landing gear for aeroplanes, and the locomotive buffers present at every major train station.

Lowered robot with interior pneumatic piston:
Furthering the industrial-use application, removing the wheels as the primary load-bearing structures, and replacing them with a greater contact point over the whole area of the chassis would be of even greater benefit to the purpose of the robot. Sequentially, moving the pneumatic piston from its attachment at the underside of the robot, to the inside of the robot would not only benefit aesthetic design, but it would allow the entire robot chassis to be lowered to the floor, with space allocation and sufficient motors for the wheels to be accepted into the robot's central cavity.

Installing a frictioned material on the lip of the bottom edges of the robot would provide far greater stopping and resistive power than the wheels, and also allow for a faster piston extension without affecting the position of the robot. Combined with rear braces/feet, the robot gains powerful application in many scenarios where the mass or density of the object may be greater than the robot's.

Sway bar:
Part of mechanical oscillatory damping is dealing with extraneous forces in tandem with the one most affecting the robot. This can come in forms of spin, unwanted lateral movement, or the delayed, unpredictable force applied with active internal loads that aren't attached rigidly to the system. These are necessary to deal with to induce minimal restitution, and I believe the implementation of a sway bar commonly used in car suspensions may work (in tandem with the pneumatic cylinder) to reduce the effect of some of these forces.

Contact surface curvature:
On a deformed surface (somewhat resembling a curved roofing tile), hysteretic damping deals with oscillations just due to the property of its curvature.

Predictive programming:
Through the integration of a machine learning model, the model could more reliably and accurately predict the force reduction curve.

Distance sensor-object offset identification:
The HC-SR04 used for the identification of an object is unfortunately incapable of identifying offsets between the central point of its field of vision, and any objects that lie on its edges. This implementation was previously made, defined and integrated into the robot's development ideal. However, at the test case stage, I discovered this inconsistency in my conceptualisation vs the hardware available. This led to the painstaking removal of a 7-part function under Objective 1.1.
Within a future design, a more pragmatic and detailed sensor, such as a LiDAR, will be almost obligatory for the use cases put forward in real-life applications. Identifying and negating offsets between the position of an object and the central section of the contact surface is a fundamental operation to maximally prevent damage to the robot, its environment, and any unsecured internal loads within the object.
The unused implementation is available for view below.

Unused Implementation:

5.1.2 Function 1.2: Align the contact surface with the object

While an advanced application would necessitate the movement of the Stasis Engine alongside the moving object, this introduces too many obstacles to development. Therefore, for this implementation, we'd require the object to be static before execution.

In order to ensure a linear path to the central point of the contact surface, potential offsets must be taken into account. This is especially necessary for future applications, where the offset would be variably assessed, much like acceleration.

Function 1.2.1: Estimate distance to object.
Use the primary sensor at the top-front of the robot to check and store the distance to the object.

Function 1.2.2: Identify offset between object trajectory and centre of contact surface.
Since the HC-SR04 uses conical vision with a Field of View (FoV) of 15-20 degrees, it could potentially identify offsets where the object doesn't line up with the centre of its vision. Where the edge of the ultrasonic wave is noticeable, in similar capacity to a LiDAR sensor, one can identify both direction and magnitude of offset in one numerical value.

Function 1.2.2.1: Identify direction of offset.
If an offset is detected, it must be offset. Like velocity and acceleration, direction can be negative. This function assesses whether the robot requires a slight left or right movement in order to negate the offset.

Function 1.2.2.2: Identify magnitude of offset.
Assess the distance required to travel to complete negation – this is specifically named magnitude in order to remove the variability of negative numerics.

Function 1.2.2.3: Compare to preset distance between sensor vision source and centre of contact surface.
Use euclidean distance to find the distance between the object and contact surface. Embedded into the Arduino program, should be the human-measured and pre-set distance between the source of the sensor vision and the central point of the contact surface.

Function 1.2.3: Move in respect to offset to allow a linear path.
Following the distance assessment, the robot should move laterally to negate the offset. This requires specific and detailed drive of the omnidirectional wheels, subject to only centimetre-based uncertainty.

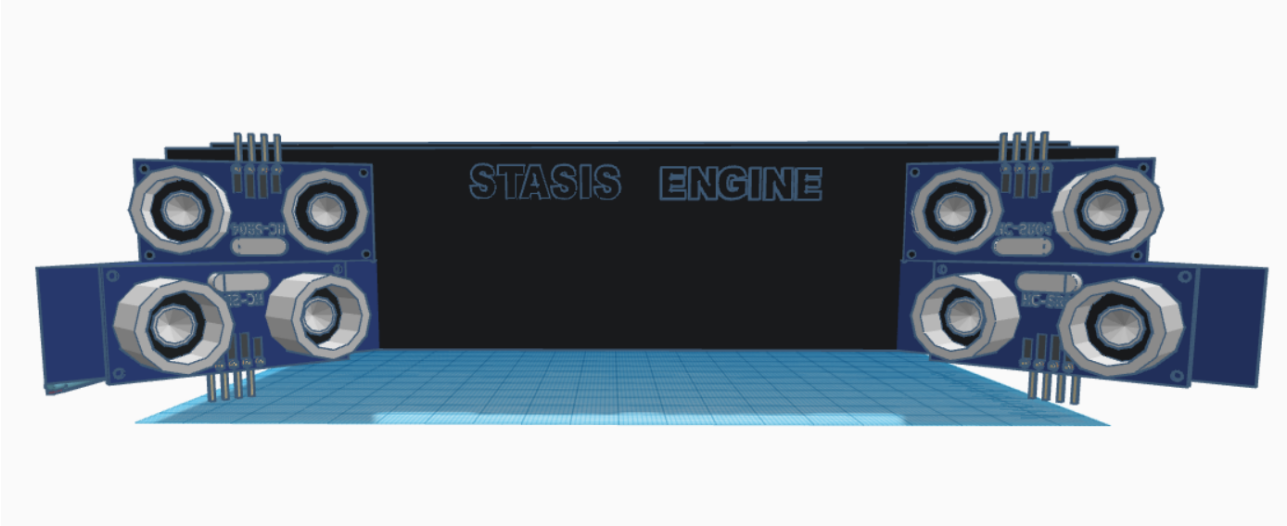
Multiple Distance Sensors:
Multi-sensor data fusion is a strong and reliable concept – if done right. With this pathway of development, I focused two major strengths, with a tertiary benefit appended:

- **Redundancy**
  Where failure is concerned, the future use cases of the Stasis Engine have no leniency. Therefore, the pairing of distance sensors would ensure that it could do its job with minimal risk of a destructive outcome for both robot, and vehicle, due to sensor failure.

- **Complementarity**
  If an uncertainty or offset is found in the data received over both sensors, the average can be taken for the most accurate depiction possible.

- **Aesthetic**
  Design is arguably more important in higher tiers of development, but the attractiveness of a symmetric aesthetic remains an important feature in commercialisation.

An early stage of development:



Motion detection on either side of contact surface:
Have a digital value and discard estimation of the contact in favour of binary data from an actual sensor, in order to decrease uncertainty within the system. This is particularly important for **Function 3.2.1**.

Pressure sensors on the contact surface:
To empirically verify that there was no physical restitution at all, as an advancement to visual verification. FSR (Force-Sensitive Resistors) could be applied on the contact surface to allow for an accurate measurement of whether zero restitution is achieved. Another strength of this addition is that even objects with very little mass (that I'm likely to be using in testing for the Stasis Engine),

exert very little pressure when not assisted by gravity. At this stage, the Stasis Engine also likely lacks the processing power and actuation speed to handle an object with 'greater than reasonable' testing force.

Return function for empty CO2 tank:
Using a barometric sensor paired with the valve regulator, the Stasis Engine could practicably assess whether the availability of pressurised gas within the canister is insufficient for further use. Following, it would return to the nearest refill station (for the purposes of testing, identify the closest structure using rotation and reverse into it, stopping just before) where it could refill its gas using a larger air compressor.

Re-extension function:
For smaller, non-serious pathways of development, the Stasis Engine (or a more appropriately named subtype) could return the object (assumed ball) to the trajectory from whence it came. The implementation of this function is simple, and the applications extend from golf to football to pet-toy-robot interaction.

Pulse Width Modulation-based Gas Regulation:
As opposed to a proportional solenoid valve, PWM-controlled programming can allow for a normal regulator to present similar (but jittery) results. However, for the purposes of slowing an object as smooth as possible, I chose to stick with a dedicated solenoid manufactured for precision.

Soldering:
This was a skill learned in the module, and I would have loved to add it to the project. However, I was torn between the increasingly attractive breadboard wiring aesthetic and the use of a solder board on an already compact working area. Coupled with the constantly changing requirements of the project, a solution as permanent as soldering just wouldn’t have fit the current ‘learn as you go’ type of project.

Gyroscope:
To better fit the surface area recommendations given by the manufacturer of the HC-SR04, one could look into implementing a gryroscopic apparatus complete with gimbals and a protruding, flat >0.5m^2 surface area. The Stasis Engine’s contact surface could then be offset appropriately to detect and decelerate the ball with significantly increased precision and accuracy.

Pneumatic Fluid:
Since the air within the master/slave system silicone tubing was unpressurised, small movements weren’t enough to displace any significant volume of air. I’d ideally have either a pressurised system, or a liquid that would respond to slight movements given the criticality of accuracy in this application.

Higher Resolution Components:
A 2mm lead screw as opposed to 8mm would allow me 4x as much accuracy, and the use of microstepping would’ve increased this to incredible heights. However, this level of accuracy would be lost on this project considering the limitations in its current design.

13 Conclusion

As the design report comes to a close, I'd like to re-establish the base concepts from which the Stasis Engine was borne. Where the possibility of an existing locomotive buffer slowing a train in this capacity is likely to be impossible, the Stasis Engine preferentially accepts loads smaller than itself. This is likely to be an inefficient use of the design, but I'd like to see a project where a similar robot could tackle something of its own mass, or even greater mass, and use pure physics to bring it to the least resistive stop possible.

Over the course of this project, I learned a great many things about physical design, fundamentals of robotics, and how a project feels like it has all the possibility in the world. Despite this, the greatest thing I'll take away is that robotic design is one of the most infinitely expandable planning stages I've ever witnessed. If one wishes, they can examine the implementation of their idea (which could be borne entirely from their own neurons!) at a realistic, microscopic level, making decisions on microsecond levels, altering response times to optimise performance that is practicably invisible to the human eye.

In conclusion, the Stasis Engine is a foray into a realm of safety that relies on the pure and oft monotonous laws of physics to deliver predictable, repeatable results. If upscaled, the robot could see wide use in emergency vehicles, locomotive accident prevention, or even some form of electronic projectile reflection system in the far future. At this point, if prototyped with a reasonably low failure rate, I hope to see surreal results, with great potential for scalability and repeatability in larger systems.

14 Appendix

14.1 Build List

The process of building the specification of the robot was first implemented in Tinkercad. The strengths of this application include ‘locked scaling’, in that no component that was derived from existing, real-world components could change in size. This meant that the chassis and space allocation could be accurate enough to real-world implementation that a vivid conceptualisation could be produced with little difficulty.

14.1.1 Sensors

Only distance sensors are required for the Stasis Engine, within the bounds of my knowledge at the time of writing. Multiple more options are highlighted in the Future Developments section at the end of this document. At this stage, I’ve committed to the use of only three **HC-SR04**s and a single **HY-SRF05**.

HC-SR04

This distance sensor serves as the primary input device of the system. A trio are used to collect sufficient data for further functionality. Their placements on the Stasis Engine ensure that three unique readings of distance, and time of measurement are taken.

The HC-SR04 was chosen for this task for its robust application regardless of luminance, as opposed to an alternate like an IR sensor. Considering the future application of the Stasis Engine, many emergency situations are likely to occur in poor and variable lighting conditions.

This particular sensor uses two pins: a TRIGGER and ECHO pins.

- **TRIGGER**:
Accepts a 10µs pulse to initiate the emittance of 8 bursts of ultrasound at 40KHz.

- **ECHO**:
Measures the time taken for the echo to return using a specified formula.

Type	**Distance Sensor**
Make	Handsontec
Model	HC-SR04
Operating Range	2cm – 400cm
Input Voltage	3.3v
Effective Angle	15 degrees (conical)
Operating Frequency	40KHz
Sound Pressure	122dB

Connector	4-pins header with 2.54mm pitch
Minimum Sample Rate	60ms
Weight	9g
Dimensions	45mm x 20mm x 15mm
Datasheet	https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

HY-SRF05 (not included in implementation)

This component finds a singular use in the reverse function, **Function 1.2.1**. Named as 'HC-SRF05' on Tinkercad, at first glance this sensor presents a promising increase in general capability. The largest notable improvement is the use of a single pin for input/output as opposed to a separate TRIGGER and ECHO pin like on the *HC-SR04*.

However, I did not choose this component since the promised 8-year production lead over the HC-SR04 delivered lacklustre results for the purposes of this project. Finding both a supplier and a sufficiently detailed datasheet was a task in itself, suggesting the use of the *HC-SR04* still domineers over its younger variant. Following, the supply voltage remains higher at 4.5V to 5.5V as opposed to the Handsontec's *HC-SR04*'s meagre 3.3V requirement.

Type	**Distance Sensor**
Make	Unbranded/Generic
Model	HY-SRF05
Operating Range	2cm – 450cm
Input Voltage	4.5v – 5.5v
Effective Angle	15 degrees (conical)
Operating Frequency	40KHz
Connector	5-pins header
Minimum Sample Rate	40Hz = 25ms
Weight	8g
Dimensions	45mm x 20mm x 15mm
Datasheet	https://www.micros.com.pl/mediaserver/M_HY-SRF05_0003.pdf

14.1.2 Actuators

The original actuation system has been changed from a variably controlled pneumatic cylinder system backed by an air compressor/CO^2 tank to a simple master/slave hydraulic cylinder system using plastic syringes.

The reasons for this change lay mostly in the academic purpose and therefore budget of this project, leading to a change from the notably more expensive former system design, to a build assisted by common household/pharmaceutical inventory. The original design has been kept, but highlighted, for archival purposes.

17HS1910-P4170 (Stepper Motor)

Despite not being its primary purpose, the stepper motor introduced me to the powerful conversion of rotational motion to linear motion. Paired with a lead screw and nut, under the mobility allowed it by the microcontroller, this remains the main fundamental change from the previous iteration of the project. This particular stepper was designed for Arduino application (alongside 3D printing, linear actuation and other DIY applications), noting the low current required for its operation.

The variable control is actionable by the decelerative calculation of the microcontroller, based on the intervalled measurements accrued by the HC-SR04s. The lead nut attachment to the flat head of the syringe rod ensures this variability transfers forth to the master/slave system through the stepper-assisted exchange of air between syringe barrels.

Type	**Actuator – Stepper Motor**
Make	Handsontec
Model	17HS1910-P4170
Type	Nema17 Bipolar
Phases	2
Step Angle	1.8°
Phase Current	1.7A
Shaft Diameter	5mm
Holding Torque	0.55Nm
Weight	0.38kg
Dimensions	48mm ± 0.5mm x 42.3mm x 31mm ± 0.1mm**
Datasheet	http://handsontec.com/dataspecs/17HS1910.pdf

Full Color Chip LEDs

For this project, I require LEDs to communicate the successful state change of 3 functions:

- Robot Power State (On/Off) **[Function 1.1]**
- Object Status (Found / Not Found) **[Function 1.1.2]**
- Input Verification (Collected/Failed) **[Function 2.2]**

Type	**Actuator – LED**
Make	Sparkfun
Model	Full Color Chip LEDs
Reverse Voltage	5v
Weight	<1g
Dimensions	3.2mm x 2.6mm
Datasheet	https://cdn.sparkfun.com/datasheets/ Components/LED/LEDdatasheet.pdf

TR20X60S (Pneumatic Cylinder) (not included in implementation)

This pneumatic cylinder finds standard use in many force-reduction applications, mentioned across this design report. In order to personalise it to my own configuration, which includes the methodology of ‘computerised damping’, I will have to forcibly exceed the speed range given in the datasheet for the chosen pneumatic cylinder, through the use of the *Proportional Solenoid Valve*. This is not a concern for safety, as the speed required goes below the *lower* bound of 30mm/s, as opposed to exceeding the higher bound. This is to ensure the object comes to a complete stop without restitution.

Type	**Actuator – Pneumatic Cylinder**
Make	Airtac
Model	TR20X60S
Bore	20mm
Stroke	60mm
Maximum Stroke	200mm
Acting Type	Double-Acting
Fluid	Air
Operating Pressure	14 – 145psi
Speed Range	30 – 500mm/s
Weight	1.36kg*

Dimensions	114mm (A + Stroke) x 64mm (B) x 25mm (C)**
Datasheet	https://cdn.shopify.com/s/files/ 1/0068/3235/7429/files/ Airtac_TR_Trimantec2016.pdf

* https://www.radwell.co.uk/Buy/AIRTAC/AIRTAC/TR20X60S?redirect=true
**these values were distilled from the datasheet.

RS PRO G ¼ (Pneumatic Regulator) (not included in implementation)

Compressed gas canisters arrive with extreme levels of air pressure within, that may cause destructive damage to both a pneumatic cylinder and the system within which its implemented. For this reason, a pressure regulator will be used to ensure consistent and steady pressure, available for the pneumatic cylinder to use. Nothing special is required here, since the *Proportional Solenoid Valve* handles the variability of the airflow to the *Pneumatic Cylinder*.

Type	**Actuator – Pneumatic Regulator**
Make	RS PRO
Model	G 1/4
Adjustable Pressure Range	0.5 – 8.5bar (7.3 – 123.3psi)
Max Working Pressure	10bar (145psi)
Port Size	1/4”
Weight	0.27kg
Dimensions	86.8mm (F + E) x 40mm (A) x 95mm (B)
Datasheet	https://docs.rs-online.com/b7f0/ A700000006783557.pdf

G209A405S0X00H1 (Proportional Solenoid Valve) (not included in implementation)

Used in conjunction with the *Pneumatic Regulator*, this valve will control the flow rate of compressed air to the pneumatic cylinder, and thus the speed of the pneumatic rod. The quality and availability of this component directly influences the quality of deceleration, given a similarly robust pneumatic cylinder. The standard solenoid valves do not provide this precise control flow. The model chosen has an identical port size to the *Pneumatic Regulator,* and has extreme sensitivity values with very low hysteresis.

Type	**Actuator – Proportional Solenoid Valve**
Make	ASCO
Model	G209A405S0X00H1

Adjustable Pressure Range	0 – 8bar (0 – 116psi)
Max Working Pressure	10bar (145psi)
Port Size	1/4”
Voltage Regulation	0 – 24v or 0 – 12v DC
Weight	0.28kg
Dimensions	65mm (G) x 34.5mm (B) x 32mm (E)
Datasheet	https://www.valves-direct.com/wp-content/uploads/2023/05/ASCO-Series-209-Proportional-Valves.pdf

Piezo Sound Generator (Buzzer) (not included in implementation)

The buzzer is an optional but weighty addition considering any factor of commercialisation. The sound during reversal will instantly engage semantic recall. Its only usage is within *Function 1.2.2*.

Type	**Actuator – Buzzer**
Make	PARALLAX
Model	Piezo Sound Generator
Operating Voltage	3 – 16v
Resonant Frequency	4000±500Hz
Weight	1g
Dimensions	13.8mm (diameter) x 7.5mm (height) (cylindrical)
Datasheet	https://www.mouser.com/datasheet/2/400/ef532_ps-13444.pdf

9904-120-18105 (DC Motor) (not included in implementation)

A set of 4 DC motors are required to fulfil the rotational ability of the omnidirectional wheels, specified in *Function 1.1.1*. The motor model chosen is prohibitively expensive for the purposes of a student project, however a similar design is likely available elsewhere. I chose this model because of the incredibly detailed datasheet, and the compatibility with my choice of 12V battery.

Type	**Actuator – DC Motor**
Make	ALLIED MOTION PREMOTEC

Model	9904-120-18105
Nominal Voltage	12v
Nominal Speed	2440rpm
Weight	200g
Dimensions	29mm (diameter) x 55mm (height) (cylindrical)
Datasheet	https://www.farnell.com/datasheets/1685513.pdf

14.1.3 Microcontroller

Arduino Mega 2560 Rev3 (Microcontroller)

This board houses significant processing ability, catering slightly better to the modest requirements of this project than a board like the Arduino R3 Uno. The DC motors as a base, in tandem with the omnidirectional wheels, require 8 PWM-capable pins in order to successfully turn forwards and backwards.

Another advantage of including a board that isn't positively stuffed with pins, is that any future implementation that draws either interest or obligation, is accessible within the framework of the current design. This includes the period between the submittal of the design report and the completion of the Stasis Engine (~12 weeks).

Type	**Microcontroller**
Make	Arduino
Model	MEGA 2560 Rev3
I/O Pins	54 Digital 16 Analog 15 PWM output
Input Voltage (from VIN pad/DC jack)	7 – 12v
Input Voltage (from USB)	4.8 – 5.5v
Chip	ATmega2560 @ 16MHz
Dimensions	53mm x 99mm
Datasheet	https://www.arduino.cc/en/uploads/Main/ arduino-mega2560_R3-schematic.pdf

14.1.4 Peripherals

1604AU (9v Battery)

Despite not matching with either my previous setup iterations nor this one, I purchased this battery on last-minute basis with accessibility and ease of construction in mind. This battery powers a single rail on the breadboard, leaving the other rail the responsibility of the Arduino's 5v out.

The components receiving power from this battery include the stepper motor, 4x DC motors through 2x L298N drivers and the microcontroller itself, thus allowing it a degree of limited portability, driving towards the general intention in the now defunct *Objective 1*.

Type	**Battery**
Make	GP
Model	Ultra G-Tech
Type	Alkaline
Voltage	9v
Dimensions	25.5 mm x 16.5 mm x 47.5 mm
Weight	48g
Datasheet	https://www.mouser.com/catalog/specsheets/gp1604au_ds.pdf

L298N (Motor Driver)

Two motor drivers were required to drive the four DC motors attached to the chassis. Though the functionality involving these is defunct, they remain a vital part of the construction, and provide a degree of visual complexity that adds significantly to the visual appeal of the build.

Type	**Motor Driver**
Make	Handsontec
Model	L298N
Type	Dual H-Bridge
Input Voltage	3.2v - 40v
Peak Current	2A
Control Signal Input Voltage Range	Low: -0.3V ≤ Vin ≤ 1.5V High: 2.3V ≤ Vin ≤ Vss
Enable Signal Input Voltage Range	Low: -0.3 ≤ Vin ≤ 1.5V

	High: 2.3V ≤ Vin ≤ Vss
Dimensions	34mm x 43mm x 27mm
Datasheet	https://www.alldatasheet.com/datasheet-pdf/pdf/22440/STMICROELECTRONICS/L298N.html

A4988 (Stepper Driver)

I managed to secure this driver despite my intentions being that of the DRV8825 driver. The latter's unavailability resulted in this selection. Alongside the undervolted stepper motor, it surprisingly allowed for much of the force production and speed required for the principles of this project.

At its base, the A4988 allows for microstepping bipolar stepper motors like the 17HS1910-P4170 used in this project, with a sizeable voltage and current capacity.

Type	**Motor Driver**
Make	Allegro Microsystems LLC.
Model	A4988
Type	Microstepping Driver with Translator
Load Supply Voltage Range	8v - 35v
Step Modes	Full, ½, ¼, 1/8, 1/16
Dimensions	5mm x 5mm x 0.9mm
Datasheet	https://www.allegromicro.com/~/media/Files/Datasheets/A4988-Datasheet.ashx

PS-1270 (12v Battery)

This was chosen for the specification of this project. All components are verifiably at or below the voltage that this battery provides. However, several components, including the *HC-SR04* will require a step-down solution for voltage, which may include a breadboard rail solely powered by the Arduino's 5v out.

Type	**12v Battery**
Make	POWER-SONIC
Model	PS-1270
Nominal Capacity	20-hr. (350mA to 10.50v) – 7.00AH 10-hr. (650mA to 10.50 volts) – 6.50AH

	5-hr. (1.2A to 10.20 volts) – 6.00AH 1-hr. (4.5A to 9.00 volts) – 4.50AH
Internal Resistance	23milliohms
Weight	2.18kg
Dimensions	151mm x 65mm x 94mm
Datasheet	https://www.power-sonic.com/wp-content/uploads/2018/12/PS-1270%20technical%20specifications.pdf

4” Omni Wheels:

These were chosen for their ability to traverse multiple directions with ease. This could come useful in future applications, but the functionality remains limited to rotation (**Function 1.1.1**) and reversal (**Function 1.2.2**) at the time of this design report.

Type	**Peripherals – Wheels**
Make	Actobotics
Model	4” Omni Wheels
Central Hole Diameter	1/2”
Rubber Rollers	10
Weight	105g
Dimensions	101.6mm x 11.1mm
Datasheet	N/A Further data available here: https://www.servocity.com/4-omni-wheel/

Breadboard:

A required piece of electronics equipment. It allows neat, accessible configuration without tampering too much with the microcontroller. Also allows you to extend the power/ground connections to an entire strip.

Type	**Peripherals – Breadboard**
Make	Solarbotics
Model	Single Panel
Bus Strips	2
Terminal Strips	1
Total Contact Points	830
Weight	0.11kg
Dimensions	215mm x 95mm x 18mm

Datasheet	N/A Limited data available here: https://www.solarbotics.com/product/21020/

VIAIR 90C (Air Compressor) (not included in implementation)

This was an extremely late stage change from a pressurised gas canister. In fact, it was changed during the creation of this build list! Realistically, where gas canisters of the type I was eyeballing can go up to 860psi of pressure, this is far too much of a threat to both the system and safety of anyone nearby. In the case of failure, the only component preventing the unrestricted outflow of 860psi is the *pneumatic regulator*. Therefore, the air compressor remains a far better option. It even matches the 12V rating of the battery!

Type	**Peripherals – Air Compressor**
Make	VIAIR
Model	90C
Input Voltage	12v
Output Pressure	120psi
Total Contact Points	830
Weight	1.08kg
Dimensions	150.9mm x 115.1mm x 53.65mm
Datasheet	https://www.chiefdelphi.com/uploads/default/original/3X/3/1/312a205c85a54041193e0b321f6544895006fcde.pdf

15 References

1. Lee, D.N., Bootsma, R.J., Land, M., Regan, D. and Gray, R. (2009). Lee's 1976 Paper. *Perception*, 38(6), pp.837–858. doi:https://doi.org/10.1068/pmklee.

2. Greibe, P. (n.d.). *Poul Greibe 1 DETERMINATION OF BRAKING DISTANCE AND DRIVER BEHAVIOUR BASED ON BRAKING TRIALS*. [online] Available at: https://www.trafitec.dk.linux20.wannafindserver.dk/sites/default/files/publications/braking%20distance%20trb2008.pdf.

3. Hayati, H., Eager, D., Pendrill, A.-M. and Alberg, H. (2020). Jerk within the Context of Science and Engineering—A Systematic Review. *Vibration*, 3(4), pp.371–409. doi:https://doi.org/10.3390/vibration3040025.

4. Baumann, J., Torkzadeh, D.D., Axel Ramstein, Uwe Kiencke and Schlegl, T. (2006). Model-based predictive anti-jerk control. *Control Engineering Practice*, 14(3), pp.259–266. doi:https://doi.org/10.1016/j.conengprac.2005.03.026.

5. Isaksson-Hellman, I. and Lindman, M. (2016). Evaluation of the crash mitigation effect of low-speed automated emergency braking systems based on insurance claims data. *Traffic Injury Prevention*, 17(sup1), pp.42–47. doi:https://doi.org/10.1080/15389588.2016.1186802.

6. Tijtgat, P., Mazyn, L., De Laey, C. and Lenoir, M. (2008). The contribution of stereo vision to the control of braking. *Accident Analysis & Prevention*, 40(2), pp.719–724. doi:https://doi.org/10.1016/j.aap.2007.09.009.

7. *Official BOC UK Online | Industrial Gases | Products & Solutions | BOConline UK* (2023). https://www.boconline.co.uk/shop/en/uk/gas-a-z/carbon-dioxide/suregas-cylinder-569.

8. Schot, S.H. (1978). Jerk: The time rate of change of acceleration. *American Journal of Physics*, 46(11), pp.1090–1094. doi:https://doi.org/10.1119/1.11504.

9. Hynes, L.M. and Dickey, J.P. (2008). The rate of change of acceleration: Implications to head kinematics during rear-end impacts. *Accident Analysis & Prevention*, 40(3), pp.1063–1068. doi:https://doi.org/10.1016/j.aap.2007.11.012.

10. Alrasheed, S. (2019). Impulse, Momentum, and Collisions. *Principles of Mechanics*, pp.73–85. doi:https://doi.org/10.1007/978-3-030-15195-9_5.

11. Mirtich, B. and Canny, J. (2002). *Impulse-based Dynamic Simulation*. [online] Available at: https://people.eecs.berkeley.edu/~jfc/papers/94/ibds94.pdf [Accessed 14 Dec. 2023].

12. Roylance, D. (2001). *STRESS-STRAIN CURVES*. [online] Available at: https://resources.saylor.org/wwwresources/archived/site/wp-content/uploads/2012/09/ME1022.2.4.pdf.

13. Yang, M., Sun, Y., Pan, X., Wan, H. and Lu, X. (2018). Development and Prospect of Twisted Pair Cables. IOP Conference Series: Earth and Environmental Science, 170, p.042126. doi:https://doi.org/10.1088/1755-1315/170/4/042126.

14. Mahmoud, A.F. and Abdallah, M.I. (2008). Performance Testing of Twisted Pair Cables. *Journal of Computer Systems, Networks, and Communications*, 2008, pp.1–8. doi:https://doi.org/10.1155/2008/586427.