ASHESI

ASHESI UNIVERSITY

WALL CLIMBING ROBOT:

MODEL DESIGN, LOGIC CONTROL AND MODEL SIMULATION

CAPSTONE PROJECT

B.Sc. Mechanical Engineering

David Edem Doamekpor

2019

ASHESI UNIVERSITY

Wall Climbing Robot:
Model Design, Logic Control and Model Simulation

CAPSTONE PROJECT

Capstone Project submitted to the Department of Engineering, Ashesi University in partial fulfilment of the requirement for the award of Bachelor of Science degree in Mechanical Engineering

David Edem Doamekpor

2019

DECLARATION

I hereby declare that this capstone is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

.....

Candidate's Name:

.....

Date:

.....

I hereby declare that preparation and presentation of this capstone were supervised in accordance with the guidelines on supervision of capstone laid down by Ashesi University.

Supervisor's Signature:

.....

Supervisor's Name:

.....

Date:

.....

Acknowledgements

The saying that “it is easier said than done” was demonstrated in its quintessence during this applied project. I am therefore eternally grateful for all the people who played key roles in helping me overcome the hurdles in the undertaking of this project.

Firstly, I would like to thank the almighty God for his continuous guidance and wisdom to continue in the face of frustrations. I would also like to thank my supervisor, Dr Heather Beem for her guidance and support.

Secondly, I would like to thank my family for the mental and financial support that helped kickstart this applied project.

Finally, I would like to acknowledge my other lectures and students who shared their ideas on the implementation of this project.

Abstract

Industrial accidents are on the rise. This fact is even more prevalent in developing countries where the countries lack the safety standards required to ensure workplace safety in the pursuit of economic development. As land in prime locations becomes increasingly scarce, there is an argument for building skyscrapers. But this just creates a new workforce that are subject to the horrific dangers of cleaning glass panels at heights well above the ground.

In this applied project, I designed a robot, Wall-C to take the pain of climbing to unfavourable heights and undertaking the incredibly dangerous tasks of cleaning sky-scrappers. Wall-C is a simple and affordable robot that works via linear actuation and vacuum pressure. Using a path planning algorithm that I designed and implemented, the robot climbs the glass wall, traverses the entire surface area with its wipers and descends once the glass is clean.

Testing of Wall-C was two-fold. First, FEA analysis was performed on Wall-C and this revealed an entirely stable frame with a factor of safety of 8. The second assessment, a cleaning efficiency test, proved the robot could clean at a rate of $1731mm^2$/s (or $6.228m^2$/h).

Table of Contents

List of Tables

List of Figures

List of Abbreviations

FOS – Factor of Safety

ARS – Advanced Robotic Systems

PLC – Programmable Logic Control

DOF – Degree of Freedom

CAD – Computer Aided Design

GPIO – General Purpose Input and Output

This page intentionally left blank

Chapter 1: Introduction

1.1 Background

In 2014 alone, there were 38 workplace-related deaths per 100,000 employees in the United States [1]. Out of this lot, one area where the toll keeps increasing is the cleaning of skyscrapers. Needless to say, Ghana does not have as many skyscrapers as the United States, so the death toll is lower. However, it is important to note that as the realization of the scarcity of land hits most countries, more stringent measures have been put in place to ensure that land is used efficiently. One of such standards is the requirement for skyscrapers with a stipulated minimum height in the city centres. This policy for Ghana implies that eventually, the city centre would be a hub of skyscrapers.

As stated above, there is indeed an argument for a revised workflow as far as the maintenance of these skyscrapers is concerned to prevent accidents. This paper's primary focus shall be on the cleaning of the exterior glass panels of towers.

Currently, there are two significant ways most management companies execute the task of cleaning the exterior glass panels.

- Hoisting the cleaner up the side of the building using a window cleaning cradle; Although this approach is relatively quick, it does a poor job of providing adequate safety for the employed worker who has to manually clean each square inch of the glass panel at that height.
- Creating a secure scaffold with a protective net all around the platform and a secondary fail-safe net below – This approach is, in fact, the safest practice currently in the industry. The

problem here, however, is that the process of setting up the scaffold, cleaning the glass panels and disassembling the scaffold is a pervasive process that trades time for safety.

1.2 Problem definition

The traditional approach to skyscraper window cleaning is a twofold problem that poses a great risk to the cleaner and reduces the efficiency with which glass panels of skyscrapers are cleaned.

1.3 Objectives of the project work

The objectives of this project work are to design and fabricate a low-cost robot that can efficiently and safely clean glass panels on skyscrapers. These objectives are detailed below as features of the robotic cleaner.

- The robotic cleaner would have four (4) suction supports (can be thought of as legs)
- Additionally, the robot's design should include a set of eight (8) suction cups. This would facilitate the climbing of extremely high skyscrapers with a relatively high standard of safety.
- The robot is expected to be robust and durable.
- Furthermore, the robot is expected to be a low-cost alternative to other robot glass panel cleaners.
- The robot is also expected to clean each glass panel with maximum efficiency. This objective would be addressed with an algorithm to ensure that resources such as windshield washer fluid and time are kept at the barest minimum.

1.4 Expected outcome of the project work

It is expected that the project would provide the industry with a device that increases the efficiency of cleaning and safety. Another primary expectation is that the implementation of this device would require input from the worker who would otherwise have had to climb the skyscraper. This feature is to be implemented to ensure that this innovative device does not increase the size of the unemployment pool.

Chapter 2: Literature Review

A fair number of people have undertaken the challenging task of creating a wall-climbing robot that can function in a hazardous environment, thereby preserving human life. Of the many well-written papers, we shall first take a critical look at "Effective Pneumatic Scheme and Control Strategy of a Climbing Robot for Class Wall Cleaning on High-rise Buildings" published by Houxiang et al., in the International Journal of Advanced Robotic Systems (ARS).

The first major section of the paper is the mechanical build-up of the wall climbing/cleaning robot. The robot features a pair of rodless linear actuators that it employed in moving along both the X and Y axis. These actuators were coupled to 14 suction cups which gave it the capability of carrying up to 60kg. Additionally, the robot was fitted with a pendulum cylinder that connected the linear actuators described above. The pendulum cylinder's primary role was to provide an axis for rotation when the robot needs to be turned around. The paper stated that the pendulum cylinder had a rotation capability of 2° per step. The final mechanical system discussed is the cleaning component. An adaptive cleaning head is designed for effective cleaning. This cleaning head uses four brushes and water (that is circulated from a ground support unit).

The second section of Houxiang's paper describes the control system of the robot. The robot's programming is done via the use of a Programmable Logic Controller (PLC). According to the authors the PLC was selected because of its high stability and modularity. The PLC coordinates the activities of the robot by controlling the actuator pneumatic systems, relays and valves. By employing the use of ultrasonic sensors, the robot can navigate around obstacles on the glass panels. Additionally, the robot uses a closed-loop feedback control approach with its

vacuum sensors to determine the vacuum condition of the suction cups and to confirm the stability of the entire robot.

The next few sections describe the cleaning trajectory and the pneumatic configuration of the linear actuators. In each case, the objective is to best optimize each system without incurring too much cost. In the example of the cleaning trajectory, few tests were conducted to find what the fastest cleaning rate was (measured as area/time). The maximum efficiency achieved was $125\ m^2/hour$. The paper is very instrumental in understanding the available options in solving the prevalent problem of non-linearity in pneumatic actuators – through the use of some algorithmic feedback control system; *bang-bang control* in this case.

A major limitation of Houxiang's robot was its price. In light of the robot's sensor suite and overall complexity, its price tag, assuming it were to be used commercially, would invariably be high.

Another paper that is key on the subject of wall climbing robots is "ROBUG II: AN INTELLIGENT WALL CLIMBING ROBOT" by Luk et al, from Portsmouth Polytechnic [2]. The paper described the legged wall climbing robot to a fair amount of detail.

For adhesion, the robot relied on rubber sucker pads driven by air vacuum ejector pumps. These were able to provide a pull off force of close to 80% of atmospheric pressure. Another impressive feature of this legged wall-climbing robot was its speed, pegged at one meter per minute.

Despite all the above, ROBUG II's major limitation for the fact that it had no real purpose. The robot's build followed a modularized design to ensure that it could be modified based on the intended application. Currently, all ROBUG II can do is climb walls, albeit very well.

The final paper that shall be taken into consideration under the subject of wall climbing is “Development of Small-Size Window Cleaning Robot By Wall Climbing Mechanism” by Miyake and Ishihara [7]. In this paper, the authors explain the creating and testing of a portable wall-climbing robot.

Per the designers’ objectives, the robot was made extremely small and lightweight to aid in its portability – dimensions of 300mm x 300mm x 100m and a weight of 3kg. Also, a deliberate attempt was made at ensuring that the edge of glass panels was cleaned anytime the robot changes direction by 90°. Although the small-size robot had an impressive clean rate of 214.92 square meters per hour, it had a major issue; it lacked a feedback control system and was therefore unable to track errors. Thus, the robot could not precisely follow its locomotion algorithm.

Chapter 3: Design and Architecture

3.1 Design Criteria

The wall climbing robot – Wall-C, had a prime purpose of improving industry safety by taking on the task of climbing and cleaning glass panels. Based on this purpose, the elaborate design criteria in table 3.1 below was created.

Table 3.1: Design Criteria

DESIGN CRITERIA	DESCRIPTION
Durability	Stresses in the robot's build should be below the yield point for each material. Additionally, a factor of safety value of at least two (2) is required.
Cost	Robot should use cost-effective materials. Additionally, a complex sensor suite and logic controller should be avoided.
Weight	It should be possible for a single user to carry the robot onto the glass panel.
Mobile	Movement of the entire robot (including ground unit) should not require more than one person.
Ease of Use	Human-Machine interface should be intuitive and easy to use.

3.2 User Requirements

Being a project that directly addressed the pain points of users, there was the need to directly list the requirements as would be seen from a user perspective. The relevance of the development of a user requirement list was that it provided guidance on the features the robot should have. Thus, it was expected that

1. The robot would take in preliminary information such as entire glass height and width.
2. The robot would move and clean autonomously.

3. The robot would be user-friendly.
4. The robot would clean efficiently.
5. The robot would not destroy the surface of the glass panel.
6. The robot would have high durability.

3.3 System Requirements

To achieve the above user requirements, there was the need to adhere to precise specifications. These specifications are enlisted below.

1. The robot should be programmed in python; This would better facilitate the implementation of an algorithm to ensure automation. Moreover, the availability of powerful frameworks would ensure the interface was user-friendly.
2. The robot's wipers should have a rubber cleaning edge. This would guarantee that the glass surface being cleaned does not get defaced during the cleaning process.

3.2 Constraints

It was realised early in the project's phase that the robot's build would be subject to some constraints that would alter the perfect design and execution. These constraints ranged from funding, lack of readily available materials, and most importantly – time.

3.3 Principal Design Decisions

3.3.1 Material Selection

From the Pugh matrix in Table 1.0 below, aluminium was selected as the most optimum material to be used in creating the frame of the robot. The most critical constraints in material selection were the durability and weight of the material. At the robot's typical operating height, it would undoubtedly be at the mercy of the natural elements. Durability was once again required because of the amount of machining that the material was going to be subjected to in

the creation of each part. Moreover, because the weight of the overall robot is highly dependent on the material selected, it would require significantly less energy to scale the walls if, by design, it is made relatively lighter.

Table 3.3a: Material selection Pugh matrix

DESCRIPTION CRITERIA	ANNEALED STEEL	ALUMINUM ALLOY (1060)	PLASTIC (PLA)
Total Weight	-3	0	+2
Yield Strength	-1	0	-4
Tensile Strength	+2	+1	0
Cost	-2	0	+1
Manufacturability	+1	+2	-1
Availability	+2	+1	0
Net Score	**-1**	**+4**	**-2**

3.3.2 Pump Sizing

In light of the fact that the robot's adhesion to glass panels was based on vacuum technology, there was the need to appropriately size the vacuum pump to be procured. An overly large vacuum pump would cause energy inefficiency whereas a smaller pump may not be able to counteract the forces due to the weight of the robot.

Table 3.3b: Table Estimating Robot Weight

Quantity	Part Reference	Mass (kg)	Mass x Quantity (kg)
4	Guiding Rod	0.21755	0.8702
2	Placer	0.37775	0.7555
2	Governing Block	0.36833	0.73666
4	Suction Support	0.71341	2.85364
2	Extension	0.48524	0.48524
2	Actuator	1.53	3.06
2	Pneumatic Valves	0.345	0.690
	Miscellaneous		0.6
	TOTAL:		10.05124

The first step in sizing the vacuum pump to be used was assessing the weight of the robot. The net weight-estimate from the above was found to be about 10kg. As a safety precaution, this mass was increased to 15kg in the subsequent calculations to guarantee the selected vacuum pump's capability of handling the weight of the wall climbing robot.

$$\rho(air) = 1.225\ kg/m^3$$

$$\rho(water) = 1.225\ kg/m^3$$

$$suction\ cup\ radius = 1.5cm$$

$$\therefore suction\ cup\ area = \pi \times (1.5cm)^2 \ = 7.0686cm^2$$

$$Net\ area = \ 8 \ \times \ 7.0686cm^2 = 56.544cm^2\ (as\ suction\ cup\ number \ = \ 8)$$

$$Actuator\ Speed \ = \ 5.7mm/s$$

$$Flow\ rate\ (Q) \ = \ V \ \times \ A \ = \ (0.0007068) \ \times \ (0.0057) \ = \ 4.02876 \ \times \ 10^{-6} m^3/s$$

$$Mass\ Flow\ Rate\ of\ Air \ = \ 1.225(4.02876 \ \times \ 10^{-6}\ m^3/s) \ = \ 4.935\ x\ 10^{-6}\ kg/s$$

$$\therefore in\ one\ minute, Mass\ (air) = 2.961 \ \times 10^{-4}\ kg$$

$$For\ water,$$

$$Q = 240L/h$$

$$Q = 6.67 \times 10^{-5}\, m^3/s$$

$$Mass\ flow\ rate = 1000(6.67 \times 10^{-5}\, m^3/s) = 0.0667kg/s$$

$$\therefore in\ one\ minute, Mass\ (water) = 4.002kg$$

$$\therefore Total\ mass\ of\ Wall - C = 4.002kg + 2.961 \times 10^{-4}\ kg + 7kg$$

$$= 11kg$$

$$\underline{Assuming\ a\ mass\ of\ 15kg:}$$

$$F = m \times g$$

$$F = 8.3268 \times 9.81$$

$$F = 81.686\ N$$

$$Pressure = Force/Area$$

$$Pressure = 81.686/5.6544\ x\ 10^{-3}$$

$$Pressure = 14446.44878\ Pa$$

3.3.3 Linear Actuator Choice

The actuator of choice for the robot was meant to be a double acting pneumatic linear actuator. The reason for this was that the pneumatic actuator had a significantly higher force-to-weight ratio than for an electric actuator of the same size. Furthermore, the pneumatic actuator posed more than twice the rated speed of the electric linear actuators. Unfortunately, however, there was the need to switch to electric actuators for the simple fact that the pneumatic actuators were not readily available in Accra.

3.3.4 Logic Controller

The selected logic controller from the onset was the Raspberry Pi 3. The Arduino microcontroller was considered but was disregarded because of the following reasons. One, the Raspberry Pi had a considerably higher number of General-Purpose Input and Output pins (GPIO). Because of the many components under control, a good number of GPIO were needed. Two, although subjective, python seemed a friendlier language for use in the implementation of a path-planning algorithm; The Raspberry Pi could be programmed in python. Finally, the Raspberry Pi, because of its onboard network capabilities could interface with a laptop easier than the Arduino would have.

3.3.5 Tubing Network

The tubing network for the suction system employed the use of special grade plastic tubes. These tubes were capable of withstanding pressures of up to 10 bar. The entire pneumatic system (tubes, Tee connectors, etc.) were based on the G1/8 standard. This standard is a simple reference to the following dimensions. [3]

Table 3.3c: G1/8 Thread Dimension

Thread Designation	Diameter (inch)	Outside Diameter (mm)	Diameter nut (mm)	Diameter Tap hole (mm)	Threads per inch
G1/8”	1/8	9.73	8.85	8.80	28

3.3.6 Fixtures

When metal parts are involved, the fixture of choice is usually welding, riveting, bolting, or some other traditional method. For Wall-C however, there was the need to seek alternatives.

It was realized that an attempt at riveting/bolting would fail because of the thickness of most aluminium parts. Welding was taken into consideration as well but discarded because of the unavailability of DC welding rods needed in welding aluminium. Epoxy was therefore settled on as the primary bonding agent for over 90% of Wall-C's needs. This was because of both the high bond strength epoxy offered as well as its ready availability in the Ghanaian market.

3.4 Wall-C Overview

3.4.0 Iterations

Wall-C saw two major iterations in its design phase. Both the first and second design share the same working principle, in that its two degrees-of-freedom (DOFs) was achieved by two linear actuators.

In the first design, however, the actuator's (and other components') design reflected the size of a particular pneumatic actuator that was to be used. Due to the difficulty in the pneumatic actuator procurement, it was replaced with an electric actuator. The design was therefore revised to take into the account the larger actuator.

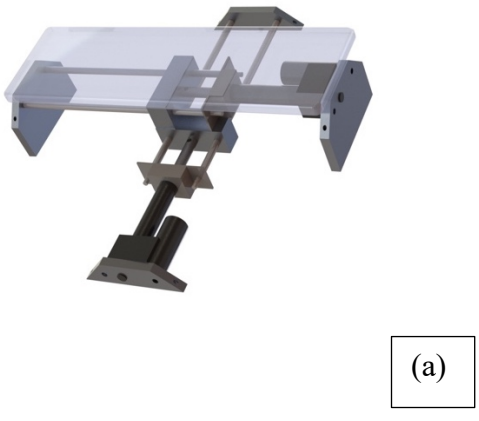


Figure 3.1a: First iteration of the wall climbing robot

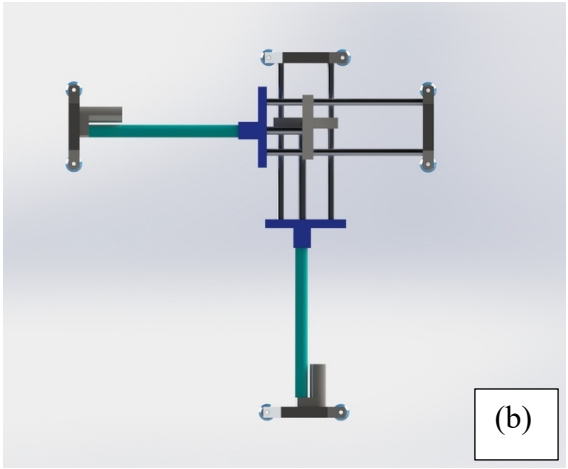


Figure 3.1b: Final 3D wall climbing robot model – Wall-C

While the electric actuator was easier to manipulate, it was rather slow and therefore limited the robot's motion to a maximum speed of 5.7mm/s.

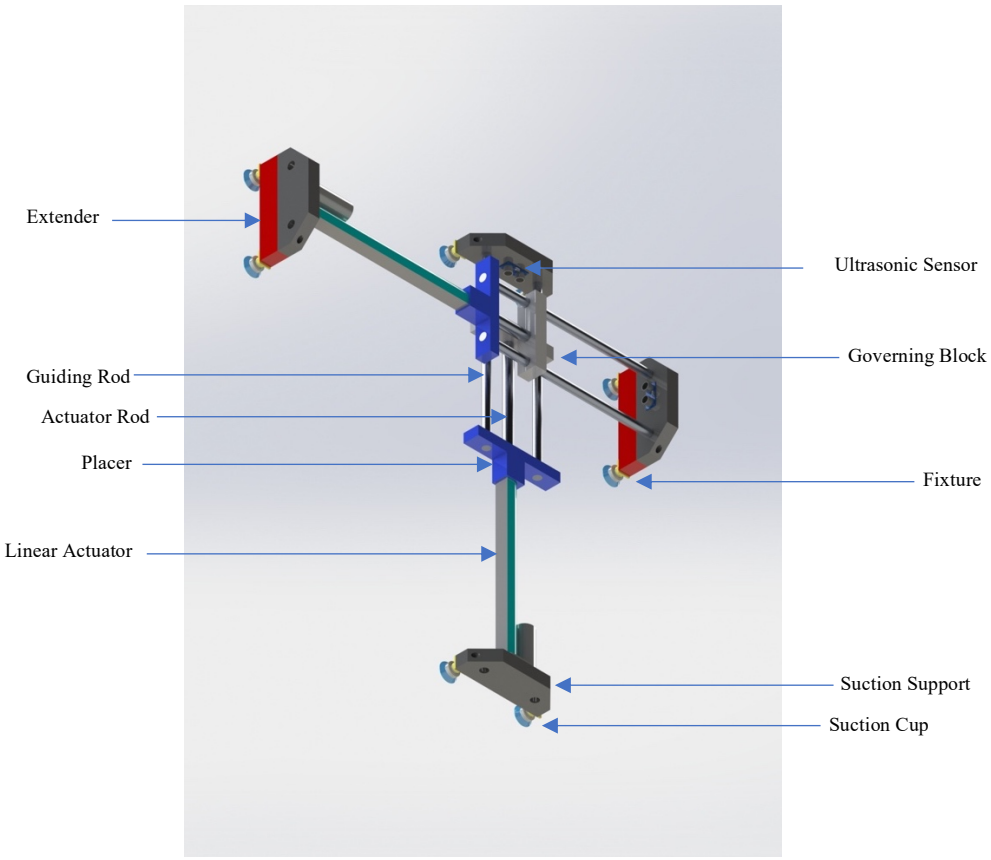


Figure 3.2: Labelled parts of Wall-C

3.4.1 Objective

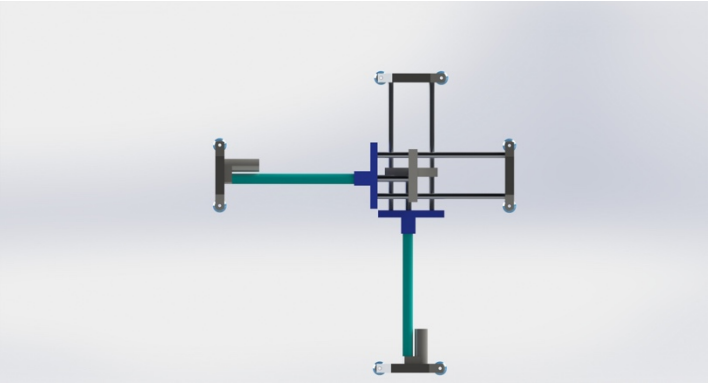
This applied project sought to build a low-cost robot that could effectively climb and clean glass panels of skyscrapers. The robot's weight was not to exceed 15kg. Furthermore, it was expected that the robot's cleaning rate be above 10 square meters per hour. This statistic

meant that for a skyscraper such as the "Premier Towers" in Accra (39m x 9.5m) [10], one of its four sides could be cleaned in less than two days.

3.4.2 Approach

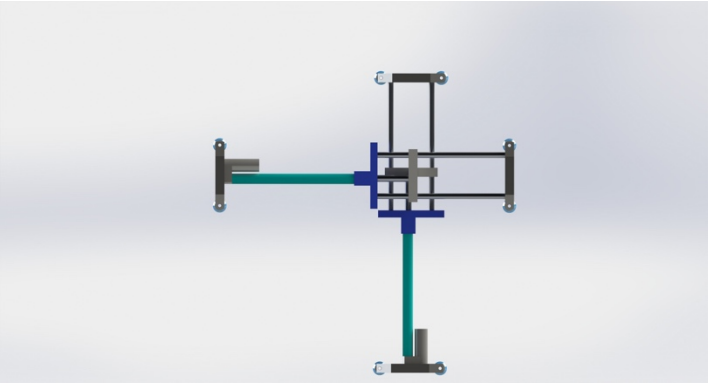
The rigid frame of the robot involved two linear actuators. These linear actuators provided the robot with two degrees-of-freedom (DOFs): vertical (y-direction) and horizontal (x-direction). A well-defined sequence of actuator extension and retraction and suction cup action were used to achieve any desired motion.

In the sequence below, the robot attempts to move up a vertical distance of 300mm - the entire stroke length of the actuator in use.

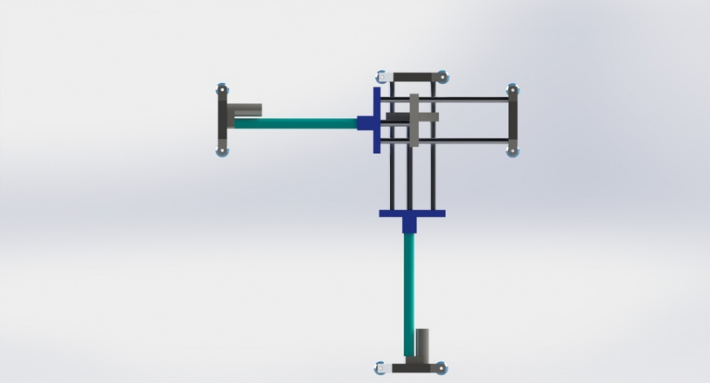


Initial position of Wall-C

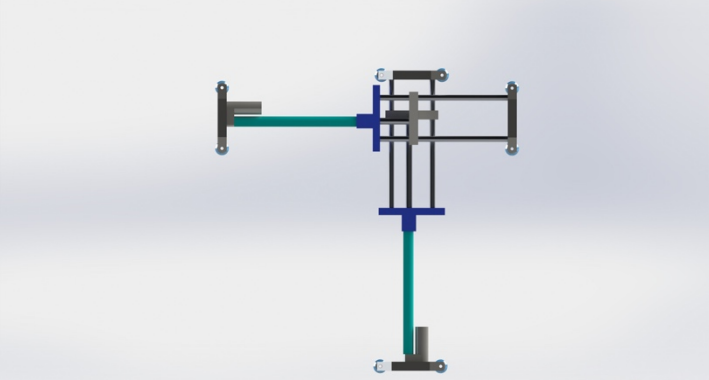
- Vertical Suction Cups "grip" in effect
- Horizontal Suction Cups "grip" in effect



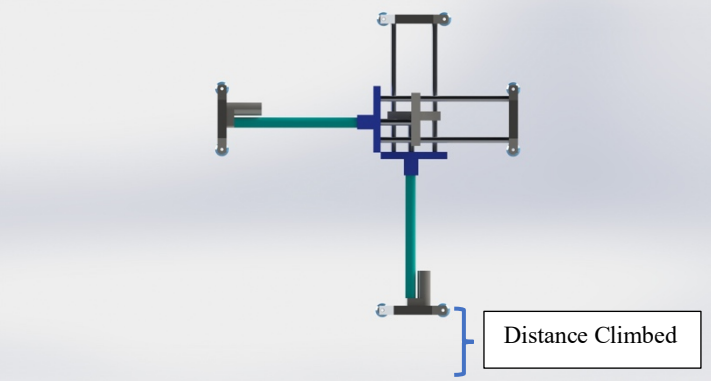
- Vertical Suction Cups "grip" in effect
- Horizontal Suction Cups "release" in effect



- Vertical Actuator Extended – pushing the entire Horizontal arm upwards.



- Horizontal Suction Cups “grip” in effect
- Vertical Suction Cups “release” in effect



- Vertical Actuator Retracted – Pulling the entire vertical arm upwards.

Figure 3.3 - Wall climbing sequence

The final picture in the sequence above shows the distance covered by Wall-C in its vertical climb. Horizontal movement follows the same general principle as described above.

3.4.3 Component List and Brief Description

- Raspberry Pi 3: The control architecture of the robot was governed by the Raspberry Pi 3 micro controller. The main role played by the micro controller was activating and deactivating a normally closed solenoid valve and a vacuum pump. By activating the solenoid valve, a region of low pressure was created beneath the connected suction cups, causing the needed adhesion.

- Eight (8) Thin Layered Male Fibre Suction Cups: A pair of these suction cups were attached to each contact patch of the robot to serve as the support system. The suction cups provided the grip needed to climb the glass panel. The inlets of the suction cups were of the G1/8 industry standard.

- Six (6) Tee connectors: The Tee connectors are simple fixtures used in routing the pressure from the vacuum pump to the various suction cups. The inlet and outlet diameters of the Tee connectors were of the G1/8 industry standard.

- Pneumatic tube: The pneumatic tube provided a transportation path for fluid (air) within the robot. This tube was of the G1/8 industry standard.

- 150 Psi Vacuum Pump (110 VAC): The Vacuum Pump is a device that provided a region of negative pressure at an inlet through the creation of a vacuum. This negative pressure

region then drew air in from the inlet (connected to the suction cup) and created the adhesion effect. The vacuum pump's outlet is of the G1/8 industry standard.

- Four (4) Normally Closed Pneumatic Valves operating at 12VDC: The solenoid valves are devices that restrict or allow the flow of a fluid based on an electrical signal. These normally closed valves can withstand pressures of up to 10 bar making them suitable for the robot's application. Both the inlet and outlet are of the G1/8 industry standard.

- 12 VDC Water Pump: This rather small water pump was used to feed water into the sprinkler system of the robot to ensure the smooth passage of the wipers during the cleaning action.

- Plastic Wipers: These plastic wipers were responsible for cleaning the glass panel as the robot traversed the surface of the glass panel.

- Four (4) 12V 20AH Battery: Owing to the power-hungry nature of the robot, four (4) batteries were connected in parallel to ensure that 80 AH was available for the undisturbed motion of the linear actuators.

- Thin aluminium metal rods (Hollow): These guiding rods were lubricated to serve as tracks on which either robot axis can slide during their movement. Hollow rods were selected for this application in a bid to shed some weight off of Wall-C.

- Four (4) Metal 25mm diameter wheels: These four wheels were attached to the ground (support) unit. It served the purpose of improving the ease of mobility of the unit.

- Two (2) 12 VDC linear electric actuators: The electric actuators extended and contracted in an iterative fashion and were primarily responsible for the robot's locomotion. In actuality, these electric actuators were selected as a compromise to the pneumatic actuators. With a speed of less than 50% that of the pneumatic actuators, the only reason the electric actuators were selected was because of their ready availability.

- Two (2) Ultrasonic Sensors: Two HC-SR04 ultrasonic sensors were coupled to the Raspberry Pi and worked based on the time of flight principle. The time of flight principle essentially is a method for checking distance by targeting an ultrasound at an object and measuring how long it took to hit the target object.

3.4.4 Primary Design

The robot's cross design was fabricated in Solidworks® with the supporting parts found in Appendix A. The entire frame was modelled around the idea of one actuator being able to completely carry the weight of another. For this purpose, it was necessary that the electric actuator selected had the capability of carrying a load of 4.32 kg. The 4.32 kg figure, generated from Solidworks, was representative of the support parts in each axis (wing).

Table 3.4a: Table Showing Individual and Cumulative Weight of Parts on One Axis

Quantity	Part Reference	Mass (kg)	Mass x Quantity (kg)
2	Guiding Rod	0.21755	0.4351
2	Placer	0.37775	0.7555
2	Governing Block	0.36833	0.73666
2	Suction Support	0.71341	1.42682
2	Extension	0.48524	0.97048
	TOTAL:		4.32456

The selected 12VDC linear actuator has a rated load of 1500N. This rating means the actuator can carry a load of 150kg. From the table above, the load bearing application requires an actuator with a capacity of at least 44N. The selected actuator, therefore, had a safety factor of over 34.

3.4.5 Sprinkler system

The Sprinkler system adopted by this robot cleaning device consists mainly of a 15L water reservoir housed in the supporting (ground) unit, four (4) nozzles that function as a water outlet and a water pump with a head of 300cm. The combined flow rate of the nozzles is 1.512L/hr.

3.2 Ground Unit Overview

3.2.1 Objective

The purpose of the ground unit was to work in tandem with Wall-C by providing Wall-C with power from the battery bank, window washer solution from the reservoir, the low pressure required for suction from the vacuum pump, and finally the logic from the Raspberry Pi that governs Wall-C's operation.

3.2.2 Approach

The Ground Unit's design started with the creation of a dimensionally accurate representation of the deep cycle batteries. The placement of the four batteries was crucial in ensuring that one, there was enough space for other components and two, the unit itself was in static equilibrium. The justification for using the batteries to achieve weight balance in the ground unit is its relatively heavy weight.

Once again, using Solidworks®, the specifications of the ground unit were determined as shown in Figure A-1 in Appendix A. The first iteration of the design included a top cover, but due to the following reasons the top cover was taken off.

- Ventilation: The continued use of the robot is bound to generate some heat in the battery bank. For this reason, a decision was made to leave the ground unit lidless for ventilation.

- Ease of umbilical connection to Wall-C: The ground unit's link to Wall-C is through the use of an umbilical. Umbilical refers to a bonded set of all the cables needed by Wall-C to function effectively. A topless design for the ground unit ensures that connections are easy to carry out.

- Oil vapour escapes from the vacuum pump: During the vacuum pump's operation, a little oil was discharged in the form of gas from the vacuum pump's exhaust. This discharge needed to escape into the atmosphere.

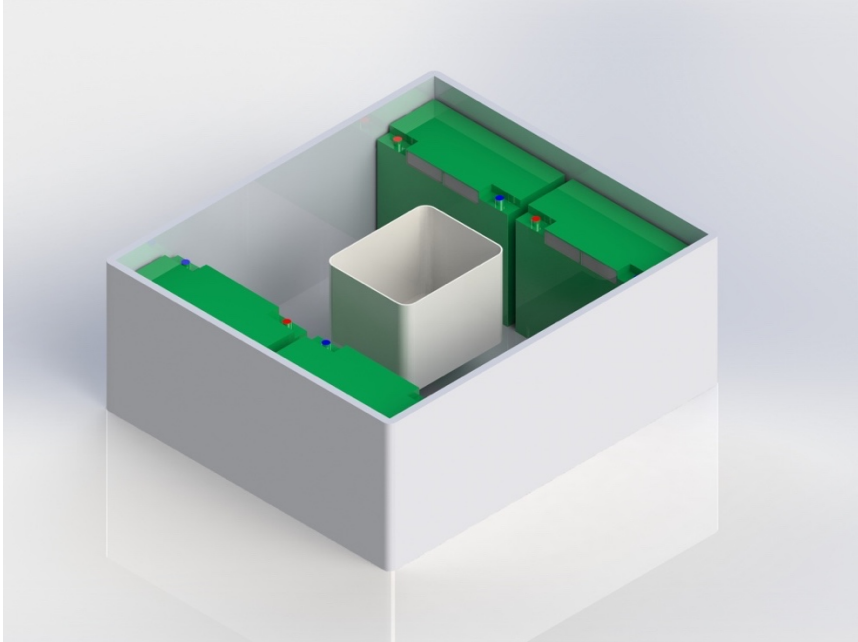


Figure 3.4: 3D model of the ground (support) unit

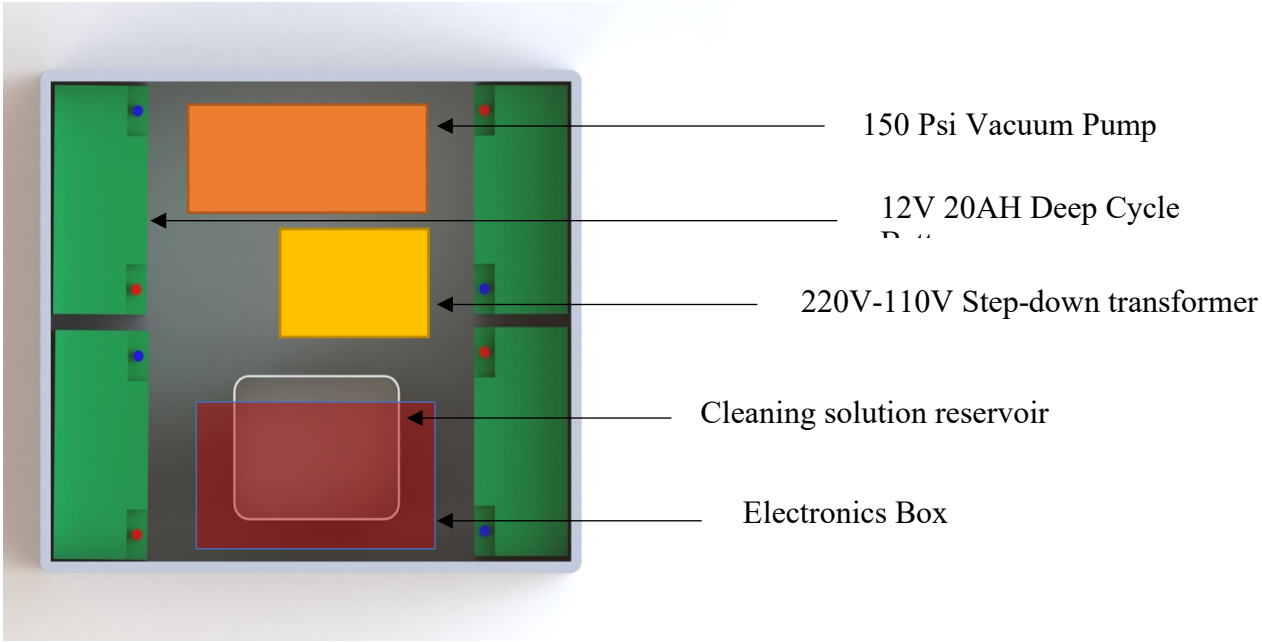


Figure 3.5: Layout of the ground (support) unit

Chapter 4: Methodology

4.2 Electrical Connections and relevant schematics

4.2.1 Power Supply system

The power requirements for the robot are 110VAC, 12VDC and 5VDC. All the major components such as the linear actuators and valves require 12VDC. The electronic components and the Raspberry Pi require 12VDC. The power requirement of the 1-stage vacuum pump is 110 VAC. Finally, the Raspberry Pi itself requires 5VDC.

Housed within the ground unit is a 220V-to-110V step-down transformer, whose purpose is to meet the power needs of the vacuum pump. The input voltage is therefore taken directly from a 220VAC 50Hz wall socket, and the 220V-to-110V step-down transformer is used to convert the voltage.

The power requirement of the Raspberry Pi system is supplied using a voltage regulator circuit. The LM2596 buck converter, in particular, was used in this project. The buck converter took the 12V source from the battery bank and was able to step it down to the 5V as required by the Raspberry Pi and the connected electronic components.

4.2.2 Relay circuits

The electrical architecture of Wall-C is such that it relies a lot on relays to provide actuation for the linear actuators, to regulate the pneumatic valves and to finally drive the water pump. For simplicity sake, two (2) and four (4) channel relay boards are used, but to develop a better appreciation of the operation of the relay board, I shall briefly explain the relay's mode of operation using the schematic below.

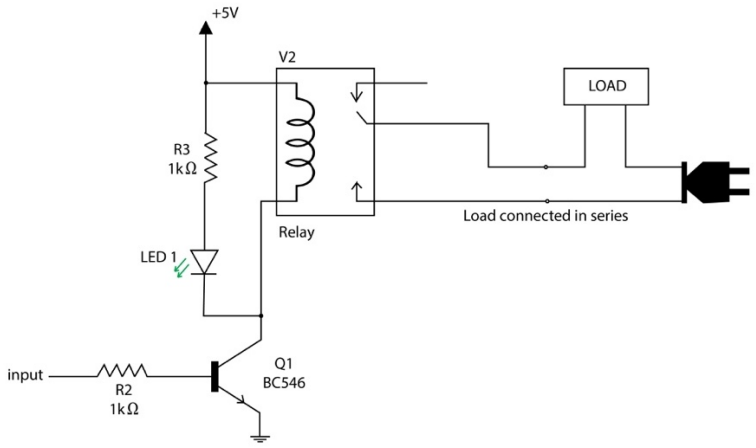


Figure 4.1 – Single relay connection schematic

At the heart of the relay module's operation is the relay itself. A relay is a device that contains an inductive coil that functions as an electromagnet. Relays have a low voltage side and a high voltage side (load). One of the coil's leads on the low voltage side is connected to a voltage source (5V) and the other is connected through a transistor to ground.

Triggering the relay therefore involves sending a current to the base of the transistor. This then connects the collector to the emitter (ground) and closes the circuit for the inductive coil. With the coil energized, it acts as an electromagnet and attracts a contactor to close the circuit on the high voltage side of the relay, thus providing power to the load.

4.2.2.1 Pump/valve relay connection

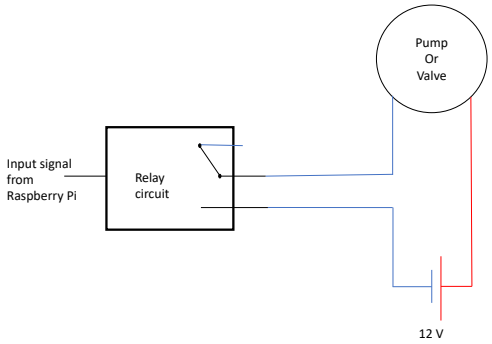


Figure 4.2 – Simple load-relay connection schematic

With the exception of the linear actuators, each of the relay connected devices follow the form as shown in the schematic XYZ above. The ground cable of the device to be connected (load) is routed through the normally open relay. "Normally open" means that until triggered, the relay (which is functioning as a switch) keeps the circuit open and prevents the flow of current.

To trigger the relay with the raspberry pi, using the *pigpio* library, a method of the form *pi.write(pin_location, HIGH)* is used. Upon issuing this command, 5V is sent to the designated GPIO pin. This (high) voltage causes the coils within the relay to become energized and closes the electromechanical magnetic switch. The closed circuit then turns the device on. The code written above can be seen in its entirety in Appendix B.

4.2.2.2 Linear actuator relay connection (H-bridge)

Like many other motors, clockwise and anticlockwise rotation is accomplished by reversing the polarity of the power supply cable. To ensure that both actuators can be extended and retracted, an H-bridge circuit was designed and implemented.

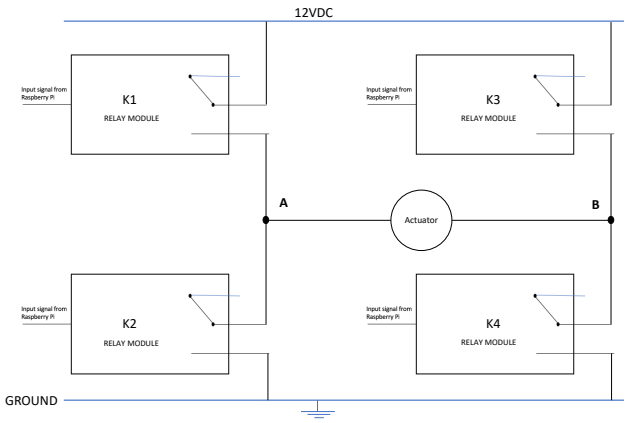


Figure 4.3 – H-bridge schematic for actuator (motor) control

The operation of the H-bridge circuit displayed above is as follows.

- For extension of the actuator, only relays K1 and K4 are closed. This action causes current to flow from the 12V DC source into relay K1, through node A, then node B and finally to ground through relay K4.
- For retraction of the actuator, only relays K2 and K3 are closed. This action causes current to flow from the 12V DC source into relay K3, through node B, then node A and finally to ground through relay K2.

It would be observed that the current flow in both configurations described above are the exact inverse of each other. This reversal in current is precisely what is needed to reverse motor direction.

4.3 Pneumatic Circuit

Wall-C alternates between two sets of four suction cups to facilitate its movement on the glass panel. Two sets of two valves (four in total) are used ingeniously to accomplish the suction cups' grip and release.

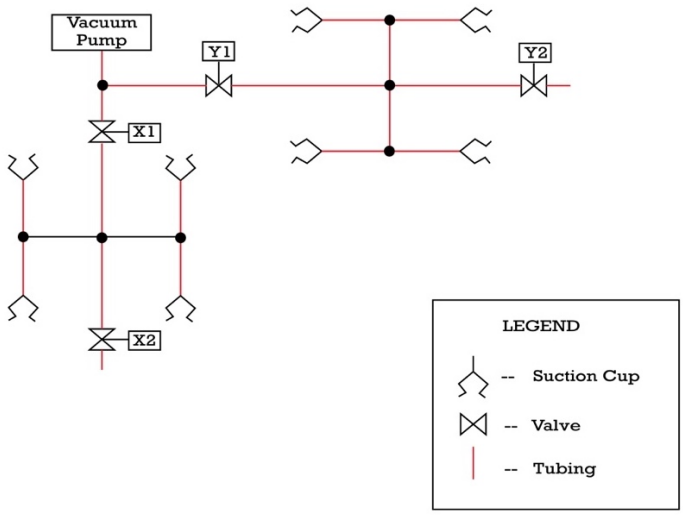


Figure 4.4 – Pneumatic schematic for Wall-C's suction system

4.3.1 Providing grip to the suction cups

The explanation below uses the horizontal arm (X) for demonstration purposes. Grip in the four attached suction cups connected to the horizontal arm is accomplished by closing the pneumatic valve X2 and opening the valve X1. The above creates a direct connection between the suction cups and the vacuum pump. By creating a vacuum or a region of reduced pressure beneath the suction cups, atmospheric pressure pushes the exterior of the suction cups against the glass panel and creates the adhesion.

4.3.2 Losing grip from the suction cups

Once again, using the horizontal arm illustratively, the principle governing the loss of grip from the suction cup can be explained. To lose grip (suction), valve X1 is closed and valve

X2 is opened. Closing valve X1 severs the connection to the vacuum pump. It is worth noting at this point that even with the vacuum pump connection severed, the area of reduced pressure beneath the suction cups still exists. It is for this reason that the valve X2 is open. Opening X2 introduces atmospheric pressure into the tubing network. A volume of air rushes into the region of low pressure beneath the suction cups. This balances the atmospheric pressure acting on the exterior surface of the suction cups thus releasing the grip on the glass surface.

4.4 Raspberry Pi Programming Using Python

4.4.1 Overview

Programming of Wall-C is accomplished in Python. Because control involves a physical connection to the Raspberry Pi's input and output pins, a GPIO library was selected. For this application the library selected was *pigpio*. By importing this library, Wall-C's movement library could then be created.

4.4.2 Wall-C's Movement Library

Before the writing of a path planning algorithm, Wall-C's movement library had to be created. The robot had to be equipped with the capability of moving in all four cardinal point direction. This was accomplished by creating functions for each direction, with each function being a set of sequences controlling pneumatic valves and actuator motors (as demonstrated in section 3.4.2 above.

Precision is a fundamental necessity in the area of Robotics. Robot designers therefore often use stepper motors because of the ability to move these motors through very discrete steps – called encoder ticks. An encoder tick is the smallest movement the motor can execute on command. The linear actuators used by Wall-C, however, are run by continuous 12V DC

motors. Had these motors been stepper motors, a simple mathematical model would have been used to find the number of encoder ticks required to move the motor (and thus the actuator) a fixed distance.

In the absence of stepper motors, Wall-C used a feedback control system to ensure that motor commands are discharged as expected. This was done through the use of two strategically placed ultrasonic sensors. Whenever an "extend" command, for instance, was passed to one of the actuators, the associated ultrasonic sensor continuously read the distance between the *governing block* and the *suction support*. Wall-C was then able to confirm indeed that the actuator was extended and that the extension distance was as expected.

4.4.3 Path Planning Algorithm

To ensure the efficient movement of Wall-C along the glass panel which it is tasked to clean, the following path-planning algorithm was created. The two prime objectives of the algorithm were high-efficiency cleaning and to facilitate exteroception.

Exteroception

The concept of exteroception is defined by The Robotics Primer as the process through which the robot gains information about its surroundings and itself. [5] The relevance of exteroception in such a project cannot be overemphasized. In tracking cleaning efficiency, for instance, it is crucial that the exact location of the robot on the glass panel is known. It is for the above-stated reason that a mapping system for Wall-C was built using a cartesian coordinate system. At any given time, the robot was aware of its cell coordinates, and thus position on the wall.

High-efficiency Cleaning

Wall-C's cleaning mechanism consists of 4 wipers and is directly linked to the robot's locomotion. In other words, as the robot moves along the glass panel, it sprays water, and the wipers clean the surface.

It was realized early that if the robot wiped in a top-to-bottom fashion (from cell (0,0) to cell (14,0)), it would be unable to clean the next column (14,1) to (0,1) because residue water from the sprinkler system would ruin the already cleaned sections as it makes its ascent.

y

x

(0,0)	**(0,1)**	**(0,2)**	**(0,3)**	**(0,4)**	**(0,5)**	**(0,6)**	**(0,7)**	**(0,8)**	**(0,9)**	**(0,10)**	**(0,11)**	**(0,12)**	**(0,13)**	**(0,14)**
(1,0)	(1,1)													(1,14)
(2,0)		(2,2)												(2,14)
(3,0)			(3,3)											(3,14)
(4,0)				(4,4)										(4,14)
(5,0)					(5,5)									(5,14)
(6,0)						(6,6)								(6,14)
(7,0)							(7,7)							(7,14)
(8,0)								(8,8)						(8,14)
(9,0)									(9,9)					(9,14)
(10,0)										(10,10)				(10,14)
(11,0)											(11,11)			(11,14)
(12,0)												(12,12)		(12,14)
(13,0)													(13,13)	(13,14)
(14,0)	(14,1)	(14,2)	(14,3)	(14,4)	(14,5)	(14,6)	(14,7)	(14,8)	(14,9)	(14,10)	(14,11)	(14,12)	(14,13)	(14,14)

Figure 4.5 – Sample grid cells generated by Wall-C for path-planning

The solution was therefore to clean in a horizontal pattern. From the origin (0,0), Wall-C moves to the end of the row (0,14), moves down one cell and repeats the process in the reverse direction.

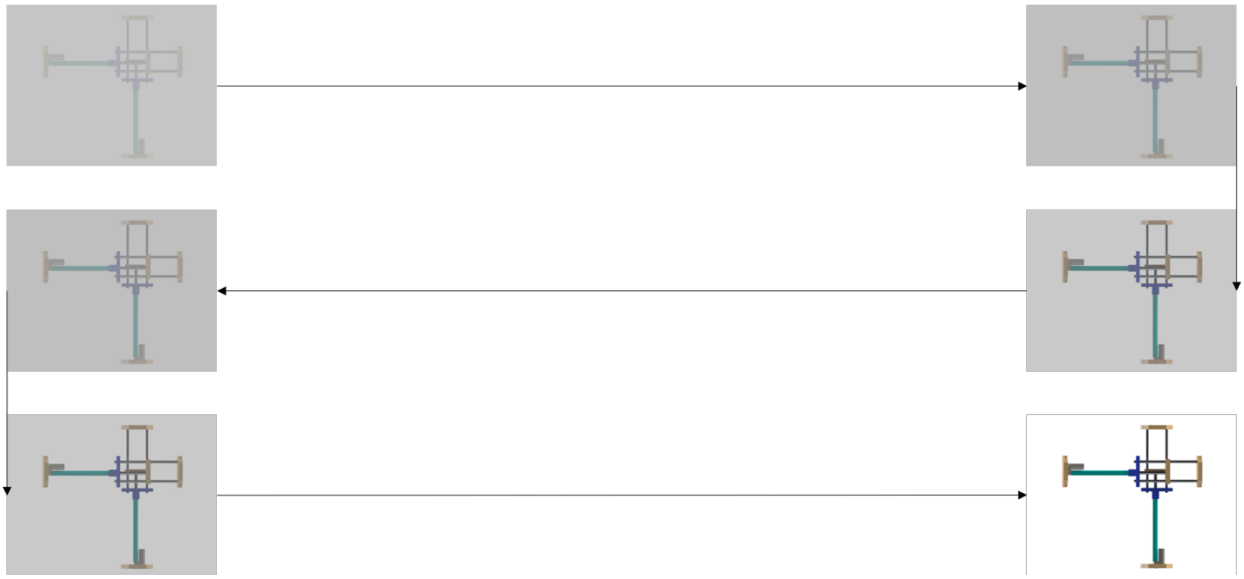


Figure 4.6 – Wall-C's cleaning trajectory

Algorithm implementation

The user specifies the height and width of the glass panel to be cleaned. Based on the pre-specified stroke length, the program divides the glass panel into a grid of cells. From the cleaning sequence described above, one would notice that once the row of x is even, e.g. (2,14), Wall-C's trajectory is in increasing values of y. When the x row is odd, as in the case of (1,14), the robot moves in decreasing values of y.

With this information, an array is created, and each cell coordinate is placed in the order as would be executed from the (0,0) reference position. With the path complete, the next step is traversing the path.

Wall-C's movement library allows two controllable DOFs. This means that the robot is capable of moving upwards, downwards and to either side. To traverse the path generated by the function discussed above, Wall-C compares the coordinate of the next cell from the array to its current cell.

If for instance that next cell was (0,1) and the current cell was (0,0) the x value of *current cell* is equal to the x values for *next cell*. The y value for *next cell* however is higher than that of *current cell*. With this information, Wall-C is able to tell that the next cell is to the right relative to its current cell. A direction integer representative of the intended direction is assigned. Finally, Wall-C determines the function to run based on the assigned direction integer. That function per this example is the *sweepright* function. A similar comparison is done for the other three (3) directions.

4.4 Simulations

From section 3.1.1 above, G1/8 was preselected as the working diameter for all tube components and all tubing fixtures. Even though much care was taken in material selection, a mismatch problem was encountered - there was a mismatch between the suction cup inlet and the internal diameter of the pneumatic tube. This meant that some form of a fixture was needed to bridge the divide. One such fixture was found online but owing to the time constraint imposed on the execution of this project, a workaround to the problem was found. A 3D printing option was selected using PLA.

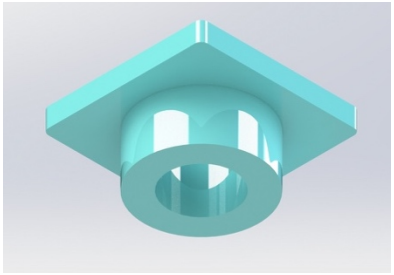


Figure 4.7 – 3D printed adapter for suction cup connection

In addition to bridging the divide, the fixture had another purpose of ensuring there was a wide enough surface area for adhesion to the suction support. The load bearing application of the fixture meant that the point of adhesion could very easily be a point for failure. The possibility of failure informed the decision to make one end of the fixture as flat and wide as possible for maximum epoxy application. Moreover, to ensure the adapter could serve the intended purpose without failure, a pressure vessel simulation was designed.

4.4.4 Adapter Simulation

A custom material was first defined. The material that was to be used in 3D printing, PLA was not readily available in Solidworks®. Its material properties were therefore looked up online and inserted into the CAD tool – Solidworks®.

4.4.4.1 Fixture Geometry

Two parts of the fixture were assigned fixed geometries. This was to mimic the forces at boundary conditions that the fixture would be subject to - one, the force at the suction cup end of the 3D printed fixture and two, the weight of the robot.

4.4.4.2 Loads - Forces

Although the robot has eight (8) suction cups coupled to it, the entire robot's weight would mostly be held by four (4) suction cups. The reason being that with one arm moving, only one arm would be stationary. This arm, with four (4) suction cups would hold Wall-C in place on the glass panel as the other actuator drags the other four (4) suction cups across the glass surface.

Following the above reasoning, a load of $36.7875\ N$ was placed at one end of the fixture.

$$15kg \times 9.81m^2/s = 147.15\ N$$

$$147.15\ N\ /\ 4 = 36.7875\ N$$

This load is what each fixture would be subjected to during its operation and was therefore applied to the fixture (purple arrows).

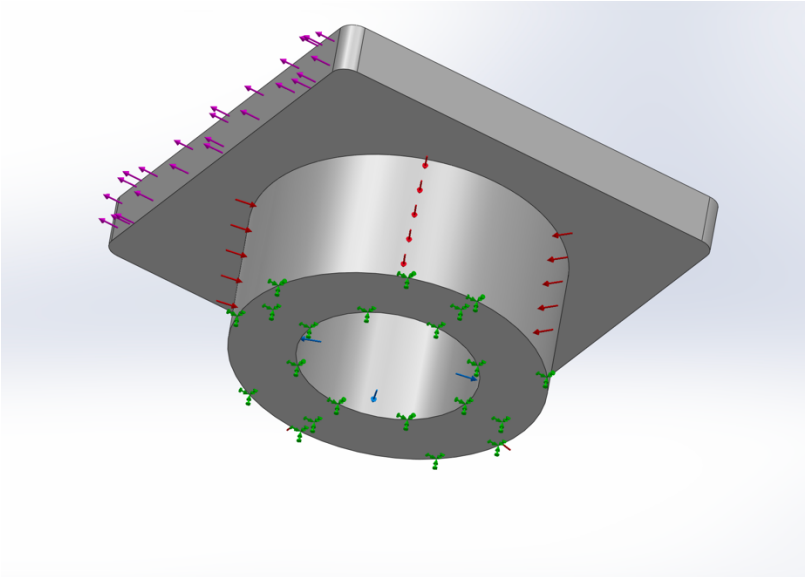


Figure 4.8 – Adapter image showing loads and fixture geometry

4.4.4.2 Loads – Pressures

Given that the interior cylindrical surface of the fixture would be subjected to the 0.133 kPa pressure from the vacuum pump, a pressure setup was created in the simulation software with that same value. This pressure is visualized in the above image as blue arrows.

On the exterior cylindrical surface of the fixture a pressure value of 101.325 kPa was used. This pressure was to simulate the atmospheric pressure that would be at play in Wall-C's operation. This pressure is visualized in the above image as red arrows.

4.4.4.3 Simulation Results

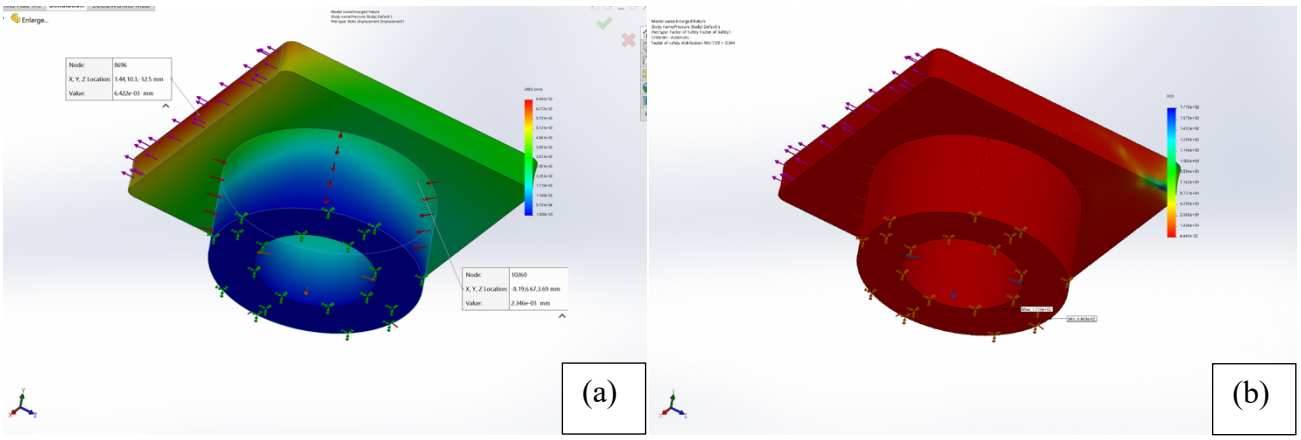


Figure 4.8 – FEA of adapter showing (a) Von Misses Stress and (b) Factor of Safety plot

The completed simulations provided very useful information regarding how the fixture would fare under loading applications. In Figure 4.8, very high stresses could be found in the area of the fixture where it would be attached to Wall-C.

More importantly from the simulation is the Factor of Safety (FOS) plot. The Factor of Safety plot provides critical information on how much load the fixture can bear before failure. A value of 2 for instance indicates that the part is able to handle two times the forces of the intended application. In the above simulation, the FOS values range from 0.4445 to 1718, with over 70 % of the model at or below 0.5. This value proves that the material is unable to sustain the forces and pressures involved in its application.

4.4.5 Wall-C's operation sequence

Wall-C operates through a very fixed sequence. Detailed below are the discreet steps taken by the robot to ensure its safe operation.

- The user specifies the height and width of the glass panel to be cleaned.
- The GPIO pins are turned off – this step is essential in ensuring that any pin state from a previous session is returned to being off before a new session is started.
- Wall-C self-calibration – Once again, from a previous session there is the possibility that the actuator position was not returned to the default. This calibration program ensures that both arms are retracted to the natural resting position at the start of the new session.
- Next, Wall-C prompts the user through the use of printed messages to the terminal screen to hold the robot against the glass panel to be cleaned for five seconds. Within these five seconds, Wall-C then opens the pneumatic valves creating adhesion in the suction cups.
- Finally, the robot begins its climb to the top left corner of the glass panel to be cleaned, position (0,0), after which the path planning algorithm is executed.

Figure 4.9 – Wall-C moving to the origin/start point. (0,0)

Chapter 5: Results

A holistic evaluation of Wall-C can be categorized under two broad headings. The first is the robot's durability by virtue of its material selection and build process. The second is its cleaning efficiency based on its pneumatic system, electrical system, and path planning algorithm.

5.1 Simulation Test and Results

The first test performed on Wall-C was virtual – in that it was performed on the Solidworks® model of Wall-C. This finite element analysis (FEA) was set up to display and study the forces and stresses bound to come into play during the robot's natural operation. The setup and results of the FEA are detailed in the subsequent sections.

5.1.1 Simulation Setup

The simulation was setup with a thorough check of the materials specification of each part. Although the material – aluminum alloy 1060 had been selected during the part modelling phase, care was taken to ensure that the materials were correct for a higher simulation accuracy.

Additionally, the minor components such as the suction cups and ultrasonic sensors were suppressed for a quicker simulation run time. Because these components played no role in the structural properties of Wall-C, they could easily be removed without any significant deviation from the expected results.

5.1.1.1 Fixture Geometry

To simulate the stresses on Wall-C, an assumption was made that the robot was climbing up vertically. For a vertical climb, the suction cups hold the robot's vertical axis steadily to the glass panel while the actuator rod pushes the horizontal axis up a distance. On that basis, the two suction supports on the vertical axis were fixed in the FEA software as shown in Figure 5.1 below.

5.1.1.2 Loads – Forces

During Wall-C's simulated climb, there were two principal loads it experienced. First was the robot's weight. For the FEA, this was represented by introducing gravitational acceleration as a force at play. The second load was the load experienced by one axis by virtue of the actuator action on the other. Once again, using an illustrative vertical climb, the actuator on the vertical arm pushed the entire horizontal arm a distance. This force exerted by the vertical arm's actuator during this action was captured by the application of a 1500N force, shown in the Figure below as purple vertical lines.

5.1.2 – Simulation Results and Interpretation

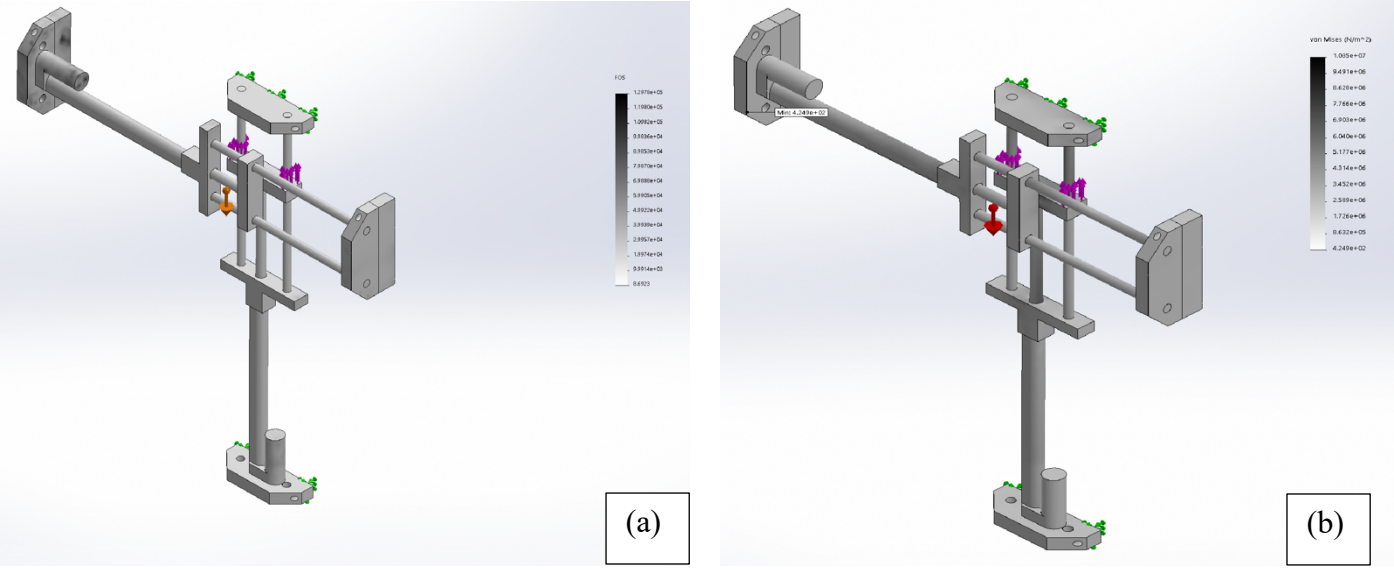


Figure 5.1 – FEA of Wall-C showing (a) Factor of Safety plot and (b) Von Misses Stress

Figure 5.1a is a factor of safety plot per the operating loading conditions on Wall-C. The factor of safety (FOS) plot is a method of detecting device failure. Because the iterative solver used in Solidworks® first divides the model into nodes, a colour graph is displayed beside the model for each result. This colour graph provides information for every node on a continuum plot.

From this colour plot, we could see both the minimum and maximum FOS values in the entire model were 8.69 and 129780 respectively. The interpretation of FOS is such that a value of 1, meant that the node in question (or area) was only capable of carrying the applied load. Thus, any additional load would result in stresses above the material's yield strength. With this reasoning, a minimum FOS value found to be less than 1 was indicative of the failure of the device at a particular point. A FOS value of approximately eight (8) therefore meant that the node in question was capable of carrying 8 times its current loading configuration.

Figure 5.1b was a static nodal stress plot. It displayed the stress conditions that would be present in Wall-C's operation. The stress plot was also instrumental in determining the success (or lack thereof) of the device from a mechanical structure point of view. Once again, the accompanying colour graph was displayed as a continuum plot. The stress results were exactly as expected. In Figure window 5.1b, areas of relatively high stresses were shown with a darker tone.

It is important to note that the presence of stresses in the model in no way means that the robot failed. Stresses are inherent in every loading application. What matters most, therefore, was the fact that for each material the stress did not exceed the yield strength (as confirmed by the FOS plot). The stress plot could, however, be used to make significant improvements to future iterations of Wall-C.

5.2 Efficiency Test and Results

5.2.1 Cleaning Area

It could be argued that Wall-C's cleaning is on the lower end of the efficiency scale. This is owed directly to the robot's locomotion mechanism. As has been stated above, the actuators used had a speed of 5.77mm/s. Based on this speed and the measured length of the 300mm wipers, the cleaning rate (area/time) could be calculated.

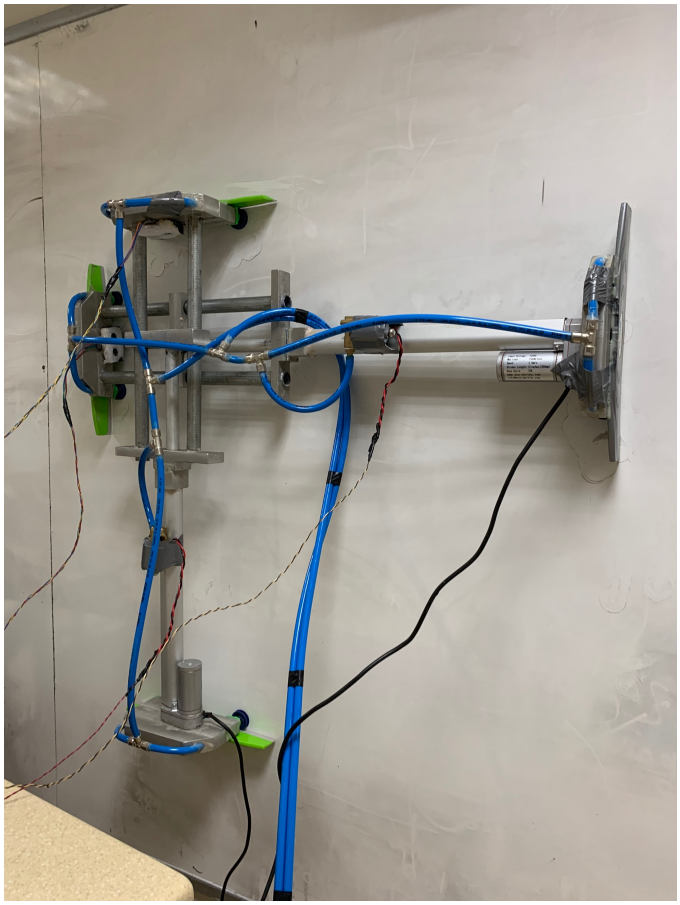


Figure 5.2 – Wall-C demo on a smooth surface

Cleaning Rate $= 5.7mm/s \times 300mm = 1710\ mm^2/s$

For better appreciation, a conversion of the value above was performed and found to be $6.156m^2/h$. Evidently, this cleaning rate is considerably less than the $10m^2/h$ cleaning rate stated in the objectives section in chapter 1.

Once again, for a building such as "Premier Towers" in Accra (39m x 9.5m), one of its sides with an area of 370.5 m^2 would be cleaned in 59.5 hours. This rather long cleaning period forces one to consider the subject of the batteries' longevity.

5.2.2 Battery Life

During the robot's testing, it was found that within an hour, the voltage of the battery bank had dropped from 11.89V to 11.36V. This introduced the realization that in Wall-C's current configuration (using the 5.7mm/s actuators) power supply would be inadequate in cleaning the entire height of an average storey building in Accra. This was further confirmed theoretically.

Amperage per battery = 20AH

Number of batteries = 4

Net amperage = 80AH

In an hour, all four pneumatic valves were active a total of 120 times - each time for 2 seconds.

$\text{Time} = 120 \times 2 \text{ seconds} = 240 \text{ seconds or } 0.0666667 \text{ hours.}$

Current per pneumatic valve = 1A

Pneumatic Valve Amperage = 0.0666667AH

For 55 out of those 60 minutes, an actuator was either extending or retracting.

Current per linear actuator = 1A

Linear Actuator Amperage = 55AH

In those 60 minutes, the small water pump run 27 times – each time for 5 seconds.

Time = 27×5 seconds = 135 seconds or 0.0375 hours.

Current for water pump = 0.3A

Water Pump Amperage = 0.01125AH

Amp Hours Remaining; 80AH - 0.0666667AH - 0.01125AH - 55AH = **24.92175AH**

From the above, it is evident that the entire system would be unable to function for up to two hours continuously.

Chapter 6: Conclusion

6.1 Discussion

Overall, despite the robot's languid speed, it was considered an enormous success. This was because of the number of issues addressed by the robot. With the development of Wall-C came a movement library and path planning algorithm that can be executed by any other wall climbing robot of the same form. Also, unlike Miyake and Ishihara's paper [7], this wall climbing robot can precisely follow the path that it generates.

Furthermore, using inexpensive materials for the build the robot was made comparatively cheap. The inexpensive materials, such as plastic wipers also ensured that the surface being cleaned did not get defaced during the cleaning activity.

6.2 Limitations

Wall-C solves some critical pain points surrounding the cleaning of glass panels but suffices to say the robot still has a fair number of limitations. Documented in the subsequent sections are a few limitations that impede the robot's maximum efficiency.

Firstly, Wall-C cannot work at heights above 300cm. Because Wall-C's sprinkler system depends heavily on the submersible water pump, the robot, theoretically can only work at height either at or below the head of the pump. The water pump used in this case has a head of 300cm or 3m. This statistic means that Wall-C can only be used for demonstrational purposes and will not be able to clean glass panels at extremely high heights.

Secondly, Wall-C has a short life span. The lifespan of a device can often be thought of in terms of the lifespan of the component with the shortest lifecycle. Once again, from a theoretical standpoint, the component with the shortest lifecycle is the 3D printed fixture discussed extensively in the fourth chapter. From the simulation, PLA lacks the material

properties needed to repeatedly withstand the forces and pressures at play in the robot's operation.

Next, is the speed of the robot. As stated above, Wall-C has a rather slow cleaning speed, and this is owed directly to the selected actuators. The linear actuators which were selected based on ready availability are incredibly slow at the rated voltage. Wall-C is rather sluggish because the entire robot's locomotion is dependent on actuator movement.

Another limitation that Wall-C has is its inability to avoid obstacles. Since the robot has no outward facing camera/sensor, it is completely unable to detect where there is a gap and/or obstacle on the surface of the glass panel. For this reason, care is taken to ensure that all panels used for demonstration purposes are plain and free from obstacles.

6.3 Future Work(s)

A design element to be addressed if a revised version of this robot were to be made is the use of metal to create the adapter fixture. A metal such as aluminium possess the metallurgical properties needed to ensure that the fixture would survive stresses associated with repeated loading conditions – fatigue as well as static stresses.

Also, to improve Wall-C's speed, the electric linear actuator would be replaced by a double acting pneumatic actuator. Pneumatic actuators by design are naturally faster than electric actuators. Additionally, they have a higher power-to-weight ratio relative to electric actuators.

Finally, Wall-C would be equipped with cameras to provide it with the ability to navigate around obstacles.

References

[1] A. Albagul, A. Asseni, O. Jomah, M. Omer and B. Farge, "Design and fabrication of an automa," *Research Gate,* pp. 208-212, 2014.

[2] B. L. Luk, A. A. Collie and J. Billingsley, "Robug 11: An Intelligent Wall Climbing Robot," IEEE, 1991.

[3] FESTO. (2013, June) *Threads in pneumatics*. [Online] Available: https://www.festo.com/wiki/en/Threads_in_pneumatics

[4] M. Eleftheria, "Safety requirements for standardization of a window-cleaning robot," Researchgate, 2015.

[5] M. Matarić, "What's going on? Sensors," in *The robotics primer*. 2nd ed. Cambridge, Mass: The MIT Press, 2008.

[6] N. Mir-Nasiri, H. Siswoyo J and H. Ali, "Portable Autonomous Window Cleaning Robot," ScienceDirect, p. 197–204, 2018.

[7] T. MIYAKE and H. ISHIHARA, "DEVELOPMENT OF SMALL-SIZE WINDOW CLEANING ROBOT BY WALL CLIMBING MECHANISM," *International Symposium on Automation and Robotics in Construction*, 2006.

[8] V. Ashish. (2016). *How Is The Safety Of Window Washers Working on Skyscrapers Ensured?*. Available: https://www.scienceabc.com/humans/clean-skyscrapers-windows-washers-equipment-height-bosun-chair-boom-carriage.html

[9] Y.-H. Choi and K.-M. Jung, "WINDORO : The World's First Commercialized Window Cleaning Robot for Domestic Use," *Applied Technology Division,* pp. 1-5, 2011.

[10] Z. Gracia. (2018, June) *What is the tallest building in Ghana*. *YEN.* [Online]. Available: https://yen.com.gh/111380-what-tallest-building-ghana.html#111380

Appendix A: Cad Drawing of Select Parts

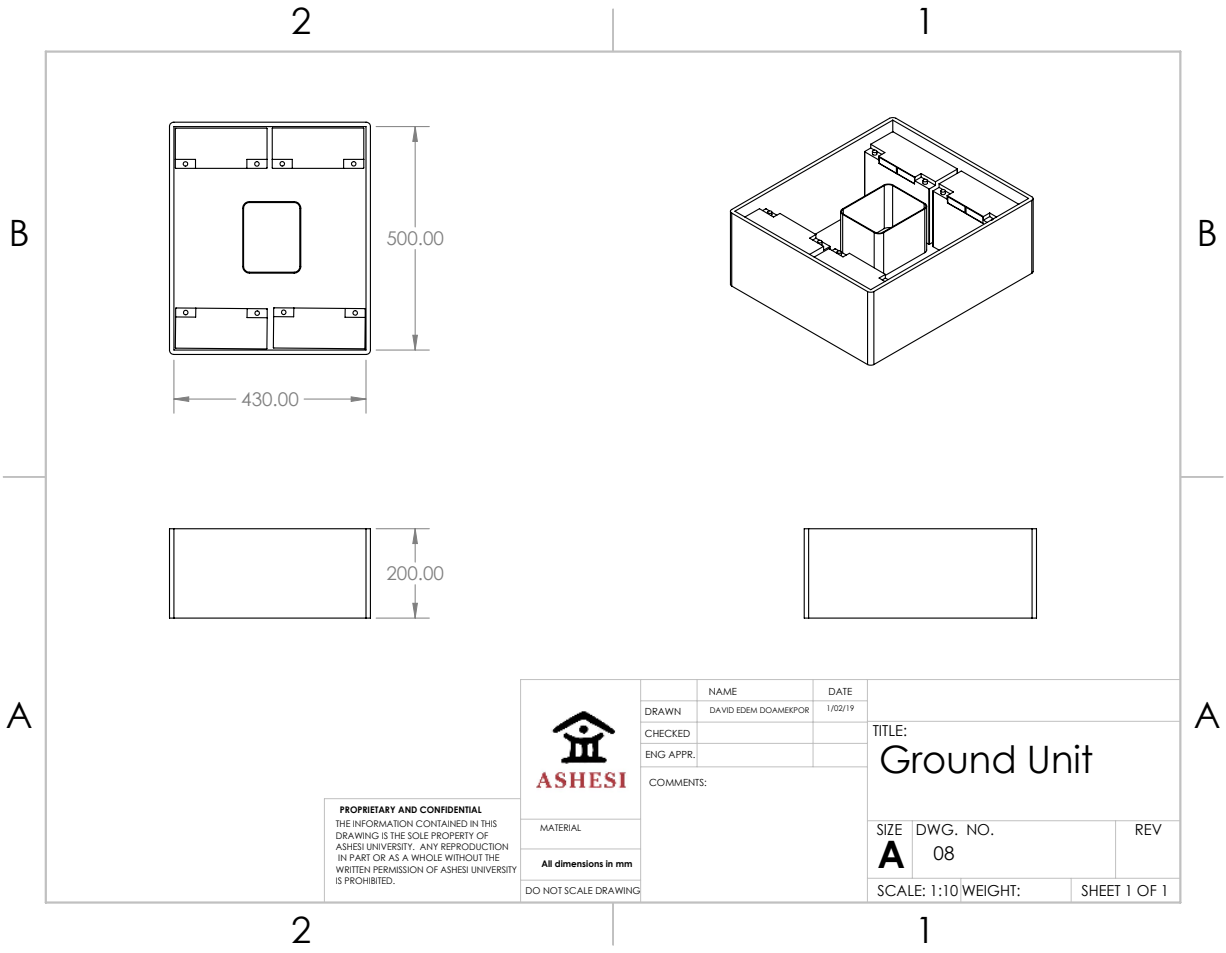


Figure A-1: Technical Drawing - Ground Unit

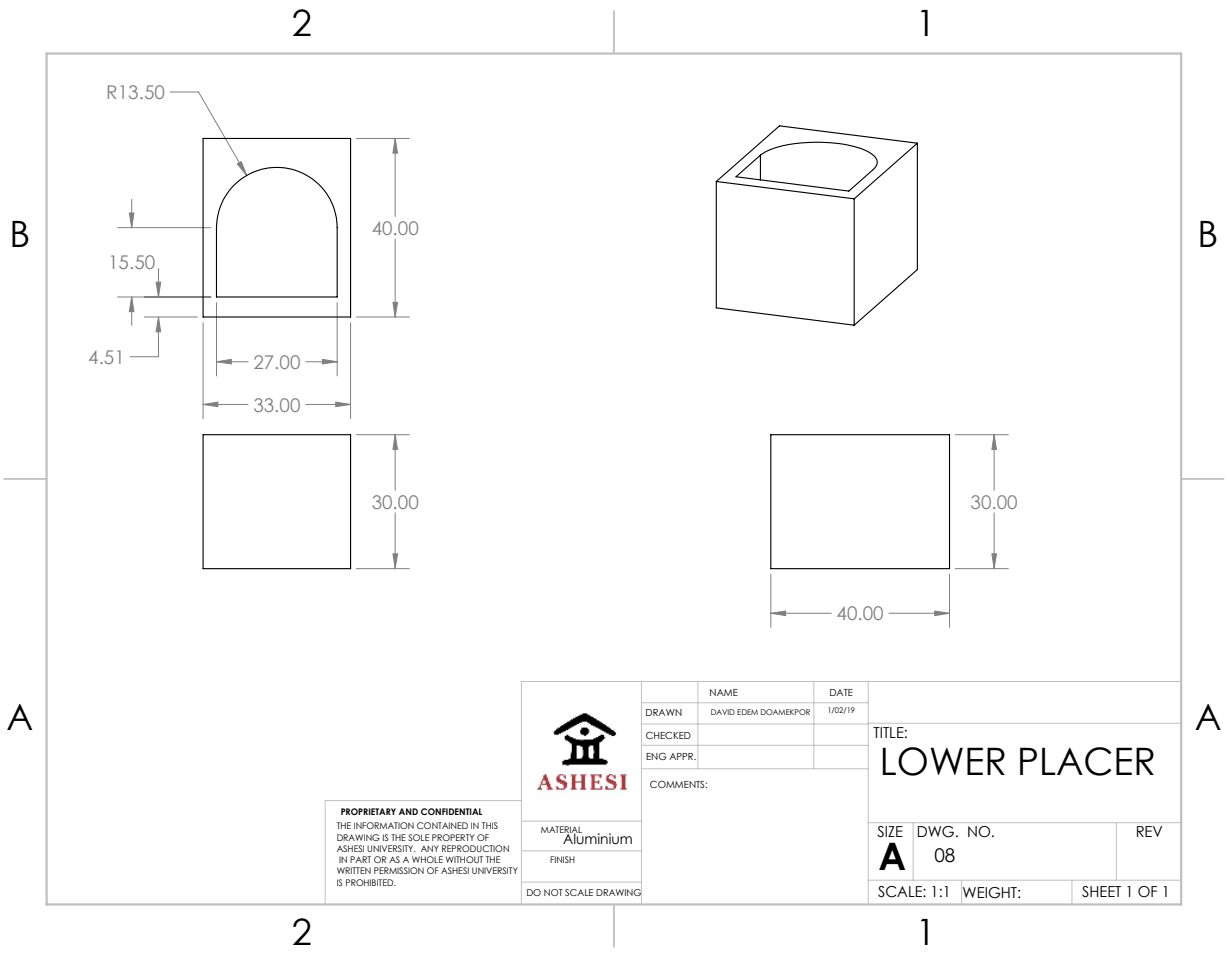


Figure A-2: Technical Drawing - Lower Placer

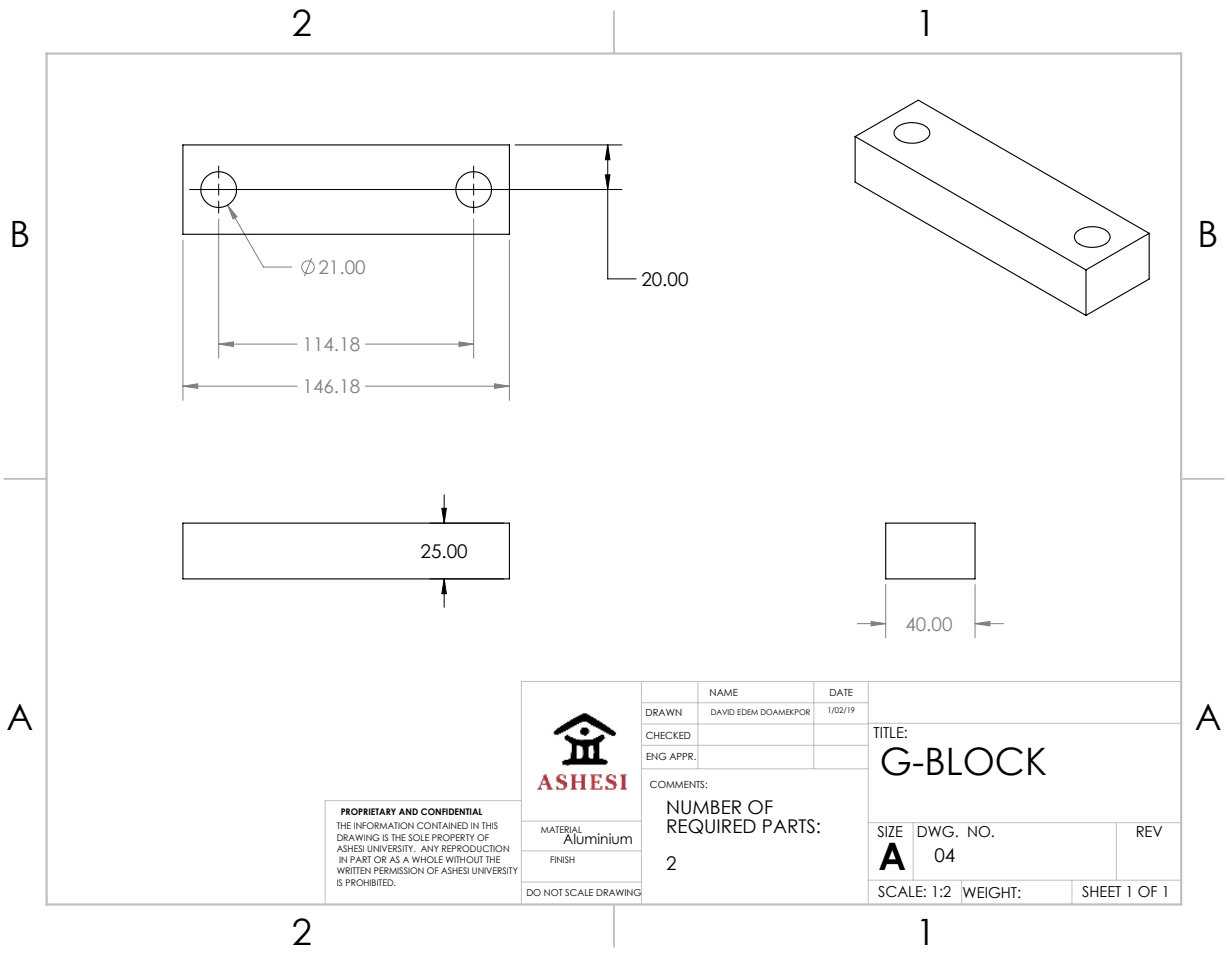


Figure A-3: Technical Drawing – Governing Block

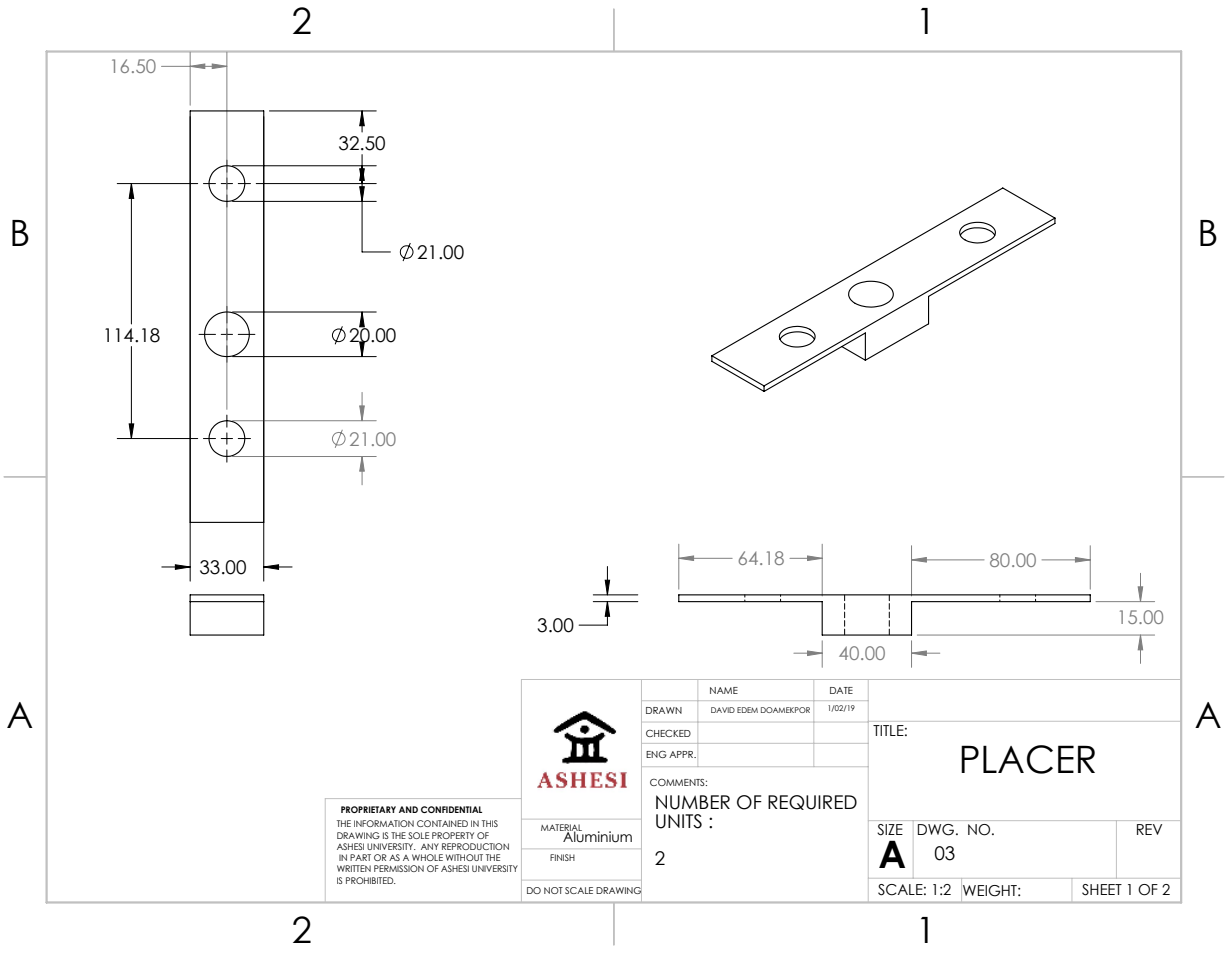


Figure A-4: Technical Drawing – Placer

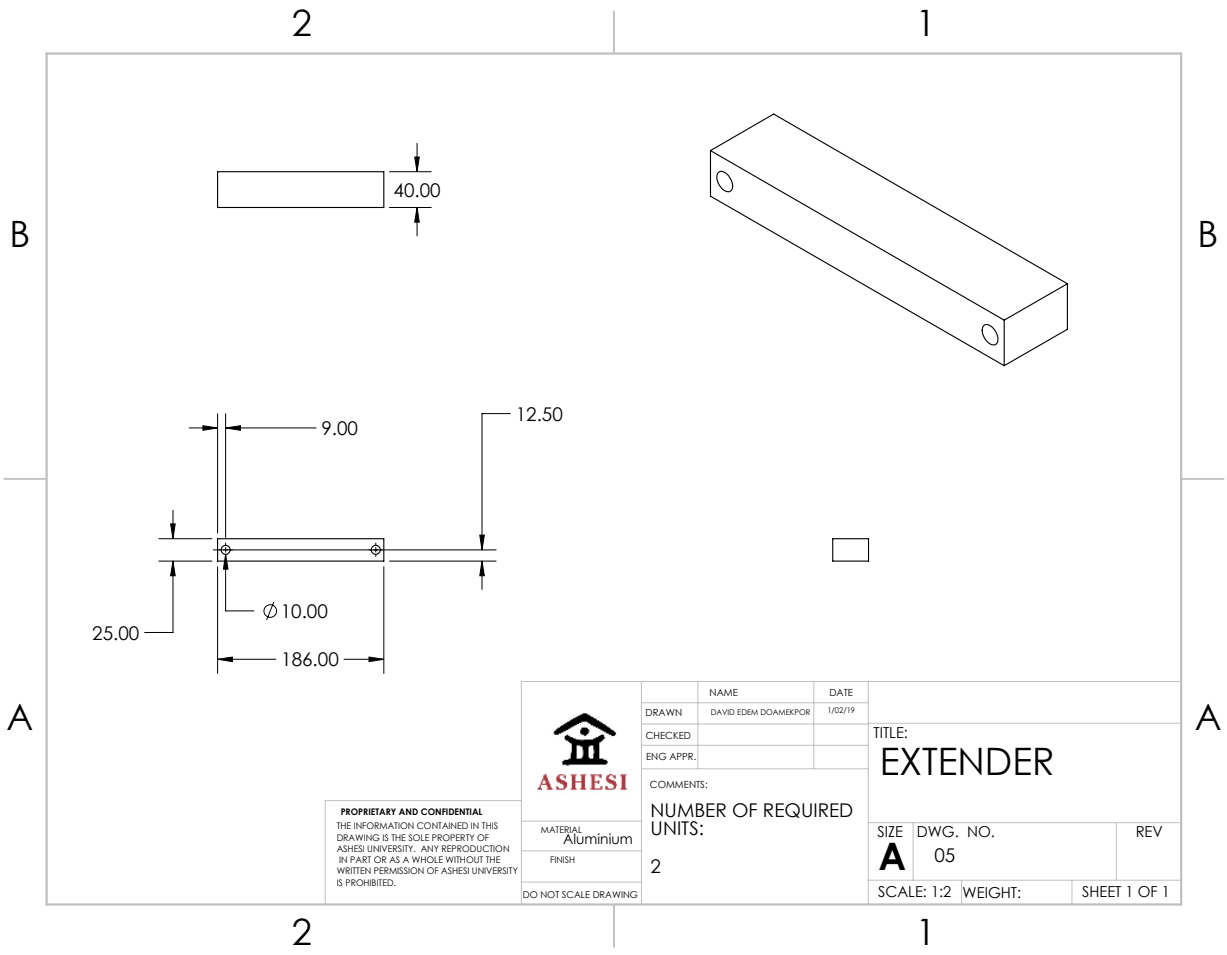


Figure A-5: Technical Drawing – Extender

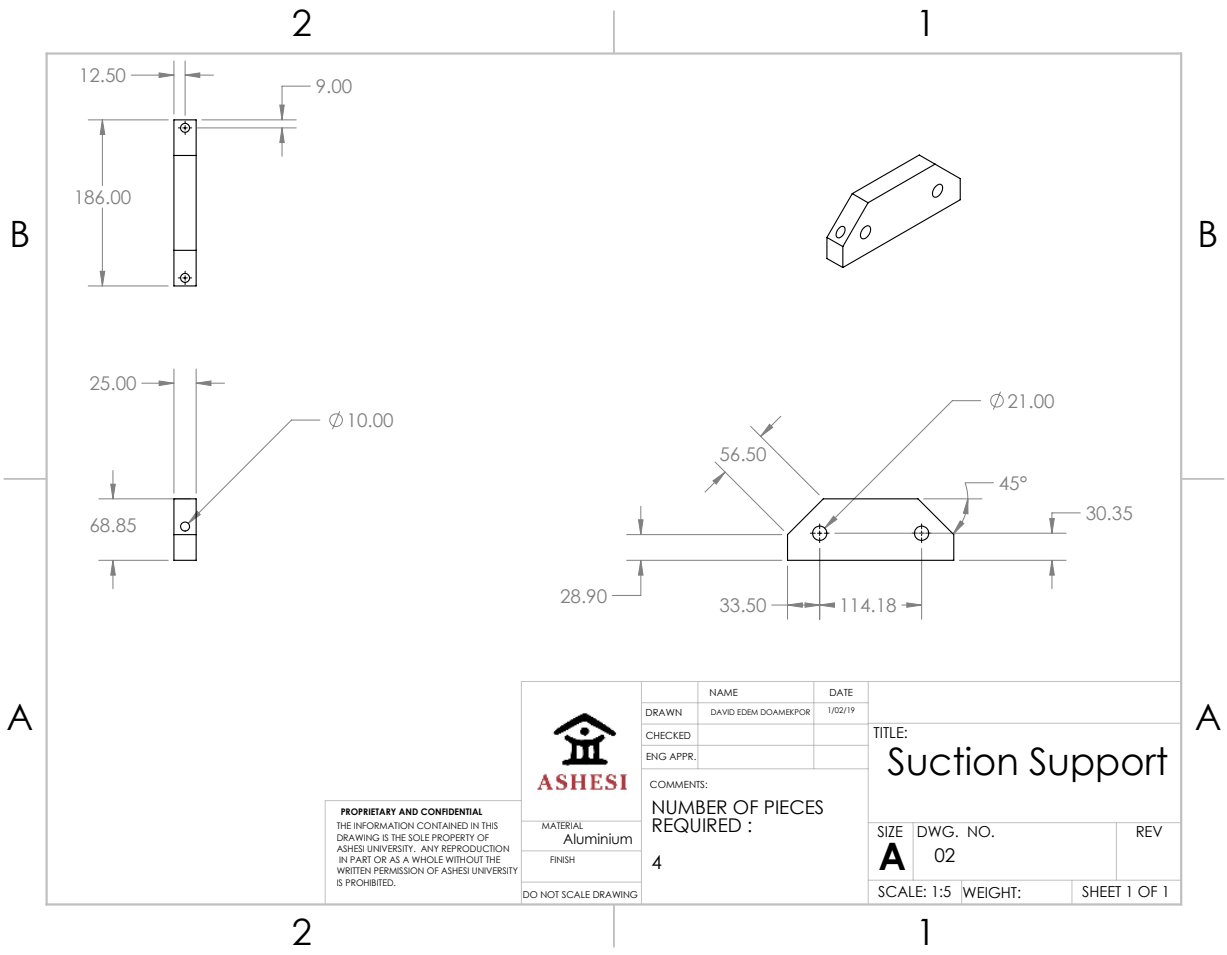


Figure A-6: Technical Drawing – Suction Support

Appendix B: Python Programs

RobotClean.py

```
# Control Code for the wall climbing robot
# Makes use of the robot library
# Simply run and follow the prommpts

#!/usr/bin/python
from time import *
from math import *
from RobotLib import *

def Clean (Wall_width, Wall_height): #unit in mm
    EverythingOff()
    Alist = [ ]
    path = [ ]

    YCells = Wall_height // stroke_len
    XCells = Wall_width // stroke_len

    for a in range(YCells):
            Alist.append([])

    for b in Alist:
            for c in range(XCells):
                    b.append(0)


    # Both actuators must be returned to a default state
    CalibrateX()
    CalibrateY()

    print("Welcome User! Please place robot on the left bottom
corner of the glass panel!")
    sleep(5) #this is to ensure the user is holding the robot
against the wall correctly.

    #robot grips the wall now with all 8 suction cups
    HorizontalGripper("grip")
    VerticalGripper("grip")

    ##Climbing the wall to the start point
    for i in range(Wall_height//stroke_len):
        VerticalClimb()
        print("initiation phase: Currently climbing to the top")
    print("Climb finally complete")

    for A in range(XCells):
        for B in range(YCells):
            if (A/2 == A//2):
```

```
                path.append((A,B))
            else:
                path.append((A,YCells-1-B))

    path.pop(0)
    print("the working path is ...", path)
    CurrentCell = (0,0)
    prevCell = CurrentCell

    for i in range ((XCells * YCells) - 1):
        nextCell = path.pop(0)

        if nextCell[0] == prevCell[0] and nextCell[1] >
prevCell[1]:
            Dir = 0
            SprayWater()
            LeftRightSweep()
            print("Robot moving east")

        elif nextCell[0] > prevCell[0] and nextCell[1] ==
prevCell[1]:
            Dir = 3
            SprayWater()
            VerticalDescent()
            print("Robot moving south")

        elif nextCell[0] == prevCell[0] and nextCell[1] <
prevCell[1]:
            Dir = 2
            SprayWater()
            RightLeftSweep()
            print("Robot moving west")

        elif nextCell[0] < prevCell[0] and nextCell[1] ==
prevCell[1]:
            Dir = 1
            SprayWater()
            VerticalClimb()
            print("Robot moving north")

        else:
            print("we seem to have a problem")

        prevCell = nextCell

    return

Clean(3000, 3000)
```

RobotLib.py

```
#!/usr/bin/python
import pigpio
from time import *
from math import *
from RobotLib import *
from PiDist import *

pi = pigpio.pi() #connecting to local pi

K1 = 4
K2 = 17
K3 = 27
K4 = 22 ## GPIO pins for the X actuator
Q1 = 16
Q2 = 26
Q3 = 20
Q4 = 21 ## GPIO pins for the Y actuator
W4 = 19 ## Water pump
Y1 = 25
Y2 = 18
X1 = 5
X2 = 6
TrigA = 23
EchoA = 24
TrigB = 12
EchoB = 13

stroke_len = 100
pi.set_mode(K1, pigpio.OUTPUT)
pi.set_mode(K2, pigpio.OUTPUT)
pi.set_mode(K3, pigpio.OUTPUT)
pi.set_mode(K4, pigpio.OUTPUT)
pi.set_mode(Q1, pigpio.OUTPUT)
pi.set_mode(Q2, pigpio.OUTPUT)
pi.set_mode(Q3, pigpio.OUTPUT)
pi.set_mode(Q4, pigpio.OUTPUT)
pi.set_mode(W4, pigpio.OUTPUT)
pi.set_mode(Y1, pigpio.OUTPUT)
pi.set_mode(Y2, pigpio.OUTPUT)
pi.set_mode(TrigA, pigpio.OUTPUT)
pi.set_mode(EchoA, pigpio.INPUT)
pi.set_mode(TrigB, pigpio.OUTPUT)
pi.set_mode(EchoB, pigpio.INPUT)

def EverythingOff():
    pi.write(K1,1)
    pi.write(K2,1)
    pi.write(K3,1)
    pi.write(K4,1)
    pi.write(Q1,1)
```

```
    pi.write(Q2,1)
    pi.write(Q3,1)
    pi.write(Q4,1)
    pi.write(W4,1)
    pi.write(X1,1)
    pi.write(X2,1)
    pi.write(Y1,1)
    pi.write(Y2,1)

def SprayWater ():
    pi.write(W4, 0)
    sleep(4)
    pi.write(W4, 1)

def VerticalGripper (state):
    if state == "grip":
        pi.write(Y1, 1)
        pi.write(Y2, 0)
        sleep(0.01)
    elif state == "release":
        pi.write(Y1, 0)
        pi.write(Y2, 1)
        sleep(0.01)
    else:
        print("the y input is invalid")

def HorizontalGripper (state):
    if state == "grip":
        pi.write(X1, 1)
        pi.write(X2, 0)
        sleep(0.01)
    elif state == "release":
        pi.write(X1, 0)
        pi.write(X2, 1)
        sleep(0.01)
    else:
        print("the x input is invalid")

def CalibrateX():
    SafeXDistance = 5 # distance the governing block should keep
from the suction support, subject to modif.
    X_distance = DRead(TrigA, EchoA)
    while (X_distance > SafeXDistance):
        XActuator("extend")
        X_distance = DRead(TrigA, EchoA)
        sleep(0.0001)
    else:
        XActuator("stop")

def CalibrateY():
    SafeYDistance = 5 # distance the governing block should keep
from the suction cups, subject to modif.
    Y_distance = DRead(TrigB, EchoB)
```

```
    while (Y_distance > SafeYDistance):
        YActuator("extend")
        Y_distance = DRead(TrigB, EchoB)
        sleep(0.0001)
    else:
        YActuator("stop")

def XActuator (direction):
    if direction == "extend":
        pi.write(K1, 0)
        pi.write(K4, 0)

    elif direction == "retract":
        pi.write(K2, 0)
        pi.write(K3, 0)

    elif direction == "stop":
        pi.write(K1, 1)
        pi.write(K2, 1)
        pi.write(K3, 1)
        pi.write(K4, 1)
    else:
        print("the input is invalid")

def YActuator (direction):
    if direction == "extend":
        pi.write(Q1, 0)
        pi.write(Q4, 0)

    elif direction == "retract":
        pi.write(Q2, 0)
        pi.write(Q3, 0)

    elif direction == "stop":
        pi.write(Q1, 1)
        pi.write(Q2, 1)
        pi.write(Q3, 1)
        pi.write(Q4, 1)

    else:
        print("the input is invalid")

def VerticalClimb ():
    VerticalGripper("grip")
    HorizontalGripper("release")
    YExtension = 6 #this distance is the allowable extension for
the y actuator, subject to modif.
    Y_distance = DRead(TrigB, EchoB)

    while (Y_distance > YExtension):
        YActuator("extend")
        Y_distance = DRead(TrigB, EchoB)
        sleep(0.0001)
```

```
    else:
        YActuator("stop")

    HorizontalGripper("grip")
    VerticalGripper("release")
    YRetraction = 20 #this distance is the allowable retraction for
the y actuator, subject to modif.
        YActuator("retract")
        Y_distance = DRead(TrigB, EchoB)
        sleep(0.0001)
    else:
        YActuator("stop")

def VerticalDescent ():
    VerticalGripper("grip")
    HorizontalGripper("release")
    YRetraction = 20 #this distance is the allowable retraction for
the y actuator, subject to modif.
    Y_distance = DRead(TrigB, EchoB)
    while (Y_distance > YRetraction):
        YActuator("retract")
        Y_distance = DRead(TrigB, EchoB)
        sleep(0.0001)
    else:
        YActuator("stop")
    HorizontalGripper("grip")
    VerticalGripper("release")
    YExtension = 6 #this distance is the allowable extension for
the y actuator, subject to modif.
    Y_distance = DRead(TrigB, EchoB)
    while (Y_distance > YExtension):
        YActuator("extend")
        Y_distance = DRead(TrigB, EchoB)
        sleep(0.0001)
    else:
        YActuator("stop")

def LeftRightSweep():
    VerticalGripper("release")
    HorizontalGripper("grip")
    XExtension = 6 #this distance is the allowable extension for
the x actuator, subject to modif.
    X_distance = DRead(TrigA, EchoA)
    while (X_distance > XExtension):
        XActuator("extend")
        X_distance = DRead(TrigA, EchoA)
        sleep(0.0001)
    else:
        YActuator("stop")
    VerticalGripper("grip")
    HorizontalGripper("release")
    XRetraction = 20 #this distance is the allowable retraction for
the x actuator, subject to modif.
```

```
    X_distance = DRead(TrigA, EchoA)
    while (X_distance > XRetraction):
        XActuator("retract")
        X_distance = DRead(TrigA, EchoA)
        sleep(0.0001)
    else:
        XActuator("stop")

def RightLeftSweep ():
    VerticalGripper("release")
    HorizontalGripper("grip")
    XExtension = 6 #this distance is the allowable extension for
the y actuator (state it)
    X_distance = DRead(TrigA, EchoA)
    while (X_distance > XExtension):
        XActuator("extend")
        X_distance = DRead(TrigA, EchoA)
        sleep(0.0001)
    else:
        XActuator("stop")
    VerticalGripper("grip")
    HorizontalGripper("release")
    XRetraction = 20 #this distance is the allowable retraction for
the x actuator, subject to modif.
    X_distance = DRead(TrigA, EchoA)
    while (X_distance > XRetraction):
        XActuator("retract")
        X_distance = DRead(TrigA, EchoA)
        sleep(0.0001)
    else:
        XActuator("stop")
```