Contents

1. Executive Summary

Efficient Multi-Agent Coverage Planning Systems can reduce the time and cost of cleaning large areas. Our Team will be presenting our solution for such a system.

Current autonomous cleaning robots used in households use bumper sensors and cover the floor in various zig-zag patterns(snake, random etc.) Furthermore for larger areas, we need to use multiple robots and hence the household cleaning robot approach is neither cost effective nor time efficient.

For efficient and effective cleaning of large areas, we need a planned path which equally divides the area to be covered among the total number of robots deployed. The paths must be planned well to avoid or minimize overlap of area covered by each robot. The optimal number of bots needed can be found out by plotting the variation of time and cost with the number of bots.

We have designed a system that plans an optimal well weighted path among all the bots deployed according to their initial positions. This planned path is then relayed to the robots who follow it. Sensors on the bot are used to ensure that the bots are on their assigned paths. An IMU is used to track the inertial measurements of the bots.

The equal distribution of area tries to distribute the work evenly. The DARP algorithm returns a closed loop path for each robot to follow.

Each designed robot approximately costs $400 - $1000 (Roomba range) with a $30 annual maintenance costs. This needs to be accompanied by a ground station which calculates and relays the optimized path according to the initial position of the cleaning robots. Time taken to clean and Cost vs number of bots data is attached later on, to decide an optimal number of robots for the task

2. Project Description

2.1. Introduction

The purpose of this project is to design a robot with appropriate sensors which will cover the area to the maximum extent. Along with the robot design, we had to design a multi-agent coverage planning system that will ensure that all the robots cover the area in the least amount of time.

Multiple robots are to be deployed in an environment. All the robots will be given the same map and path to clean.

Our task was to estimate the best minimum number of agents which can be used for a given map.

The Mechanical Design of the Robot was done taking into consideration factors such as:

- Turning Radius
- Electrical Equipment onboard
- Cleaning Equipment onboard
- Cleaning Efficiency

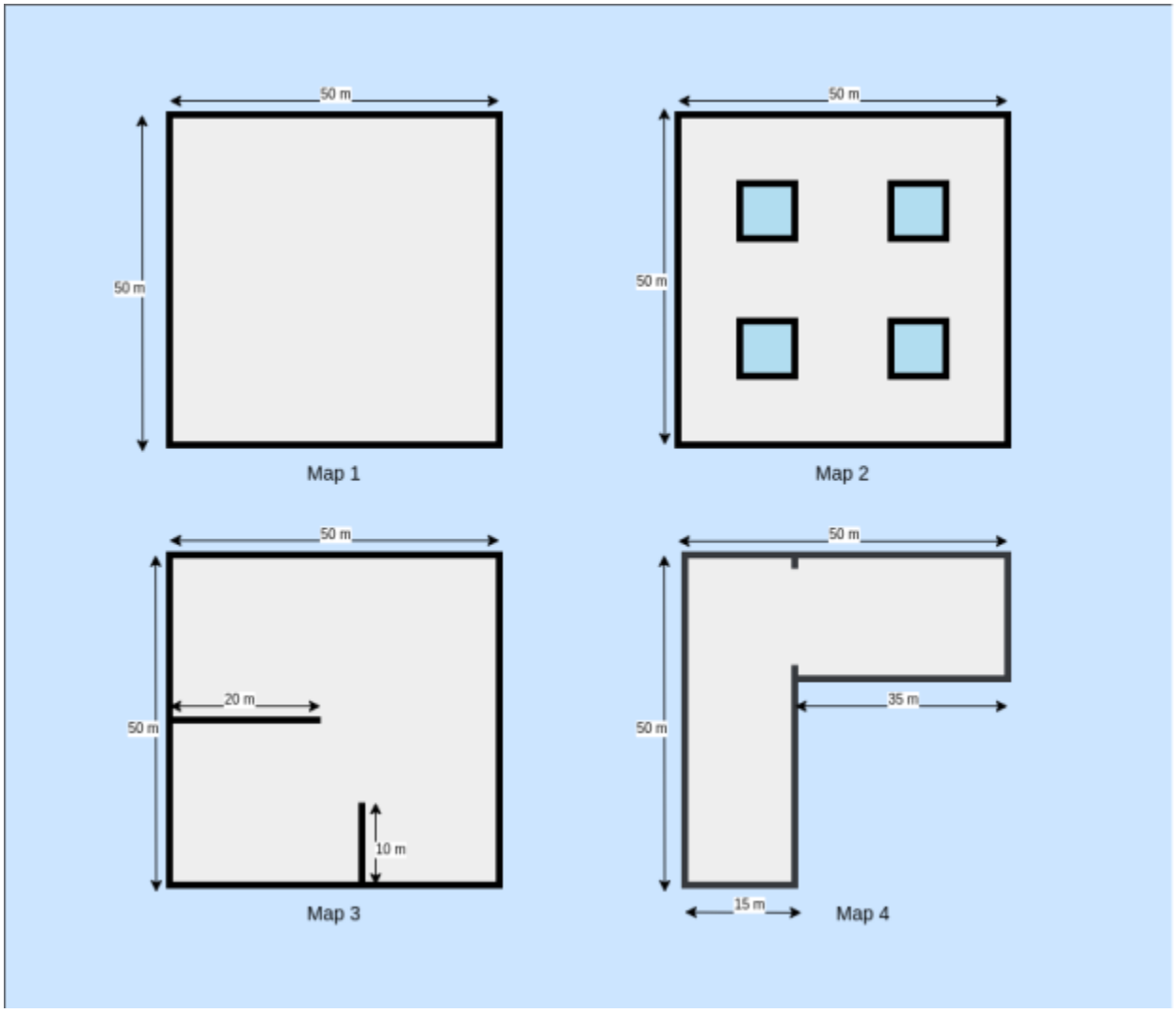
The Electrical System and Sensors are used to convey robot dynamic information to the Ground Station and correct any path deviation that may creep in while the bot follows the assigned path.

The Complete Coverage Path Planning for multiple agents has been carried out with the use of the DARP Algorithm. This algorithm ensures complete coverage of the map along with no overlapping and non back-tracking. These features ensure maximum coverage efficiency.

Cleaning Efficiency has also been calculated based on the effective cleaning area of the bot and the robot design has been enhanced from it's previous prototypes to deliver a cleaning efficiency of more than 90% everytime. Thus, this satisfies the minimum required cleaning efficiency parameter.

The time taken to clean the given maps reduces drastically as the number of bots are increase.The manufacturing cost of the robot (material) has also been mentioned. The optimal number of bots to be bought to clean can be found out using the value that is given for the time saved. Since this differs from person to person , the data has been presented and left to that.

The Solution that we propose has been tested on the maps provided to us.



All the results mentioned in this Proposal are for the above 4 maps and are referred to as the name mentioned below each map. The missing dimensions have been refined and the detailed maps along with the dimensions used have been attached in the Appendix Section.

2.1. Methodology

2.1.1. Mechanical Design

Shape

The main debate in deciding the shape of the base was between a square base and a circular base.

Circular Base

- Pros

The typical shape of a robotic vacuum cleaner is a disk. The reasons they are disk-shaped is because of mobility. They can maneuver through tight spaces and still clean effectively. When they bump into a wall or piece of furniture, since it is a circle, it can easily turn around and adjust its position and continue cleaning.

- Cons

The major problem with the vacuum being a circle is that it cannot clean the corners of rooms very well.

Square Base

- Pros

If you change the shape to a square, then the vacuum can get into the corners and clean better, but there are no square robot vacuums.

- Cons

As the vacuum is going along and cleaning it will bump into obstacles and then reposition itself. As the vacuum is re-positioning itself, the edges can come into contact with obstacles and will waste more time re-positioning itself instead of cleaning.

So considering the above facts to create a robot more applicable for the real life purposes a **Circular Base** robot was chosen for its **better maneuverability.**

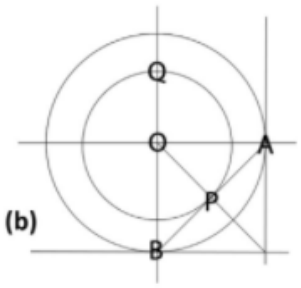
Wheel Placement and Driving System

The position of the wheels was considered by taking mainly two factors in mind.

1. To evenly distribute the load on the wheels system.
2. To accommodate large front sweeping brushes for better cleaning coverage.

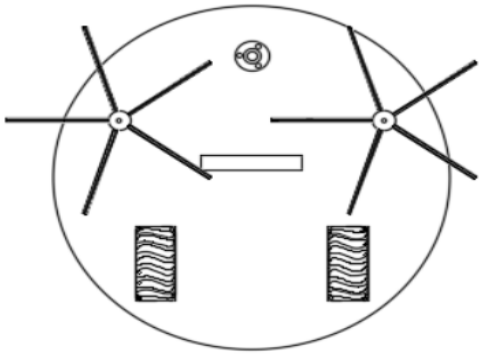
A three wheel system was carefully and systematically laid out to ensure low center of gravity and accurate motion of the robot. The wheel placement was decided by triangulation method to provide more room at the front for a higher sweeping area for the brushes. Two dc motor controlled wheels were at the back and a front ball type caster wheel for support.

A and B are the points on the base circle. Here P is the midpoint of the line AB . It is the point where the wheel will be placed. Second back wheel will be placed at the point that is the mirror of P about OB.



Q is the point where the front caster wheel is to be placed.

For the driving system, two individually controlled DC motors are used to provide desired rotational speed to each wheel. As a result the wheels can not only move forward but also turn by changing the speeds of the wheels.



Material and Component selection

Body

The material being used for the case and most of the robot is ABS because it is easily manufactured, cheap and it can be injection molded. Acrylonitrile butadiene styrene, or ABS, is a common thermoplastic used to make light, rigid, molded products such as pipes, golf club heads. The styrene gives the plastic a shiny, impervious surface. The butadiene, a rubbery substance, provides resilience even at low temperatures. ABS can be used between −25 °C and 60 °C.

Polymeric blend ratios for (PS/ABS)	Young's modulus MPa	Strain%	Tensile strength MPa
(100/0)	12.69	3.25	39.5
(90/10)	10.6	3.4	19.6
(80/20)	17.63	11.25	35.3
(70/30)	17.51	20	43.98
(60/40)	21.1	10.1	50.8
(50/50)	17.9	13.2	45.68
(40/60)	10.1	8.5	18.9
(20/80)	16.7	17	34.68
(0/100)	14.42	13	35.8

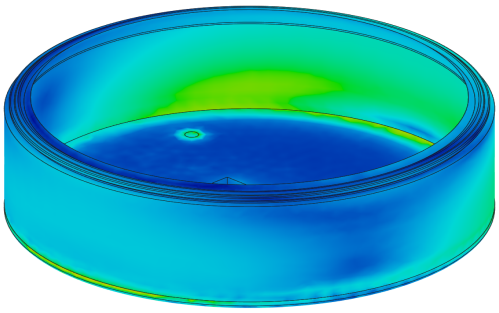
The outer case and the top are designed to be 5mm in thickness to provide a safe build and the base plate is kept to 10 mm in thickness to hold the weight. The weight of the internal components was around 2.5 kgs.

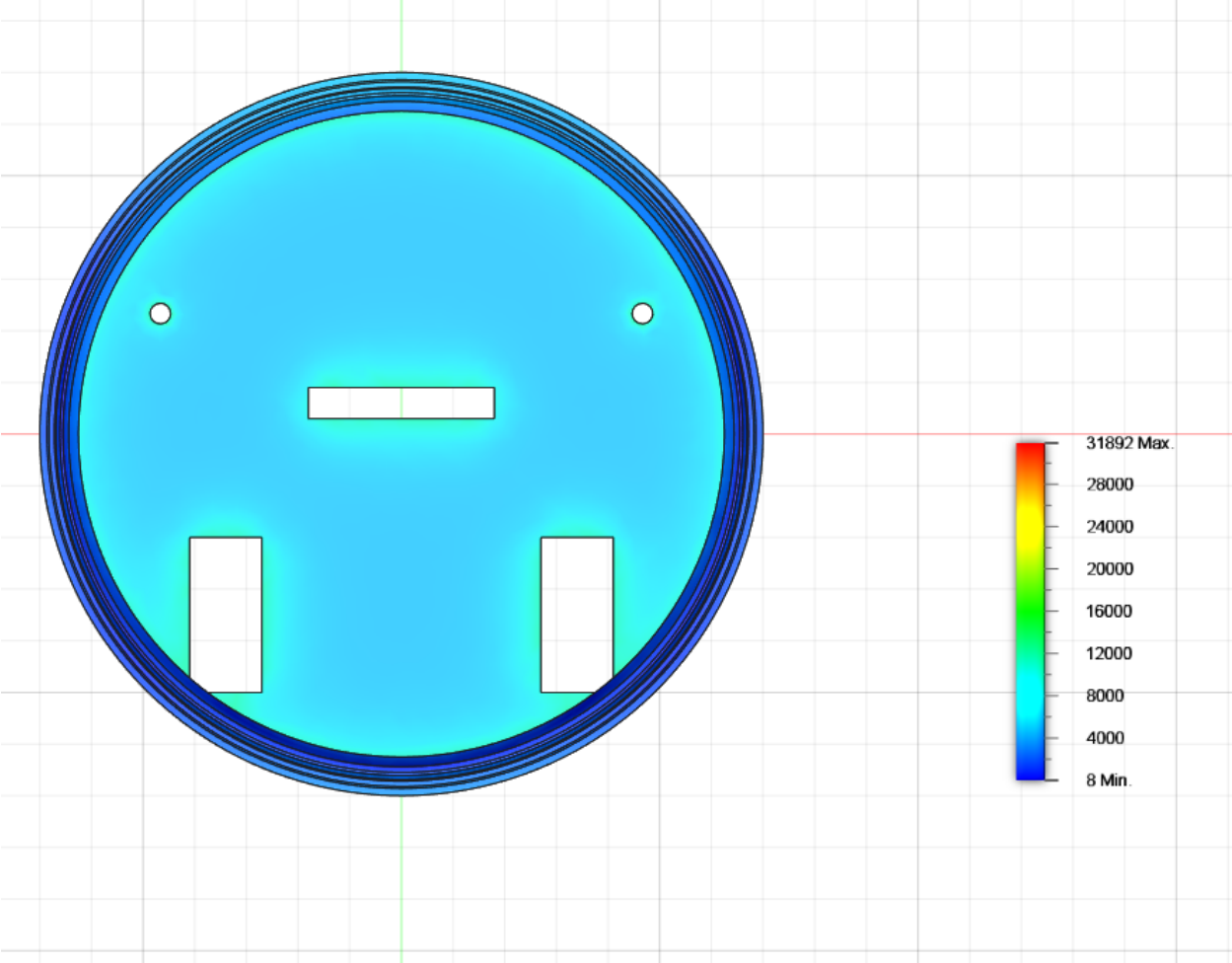
An approximate simulation for load design was done with an even distribution of 3 kgs over the base along with the outer case and the result gave a von Mises stress of 0.013 MPa.

So the base can handle a lot larger loads as compared to the one currently applied. Even half the thickness of the base can be sufficient for the load currently applied.

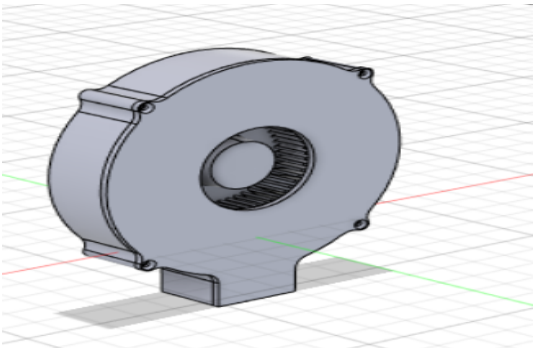
Currently the thickness is kept the same for need be arise for higher loads in the future

[MPa] 0.00001  0.01292



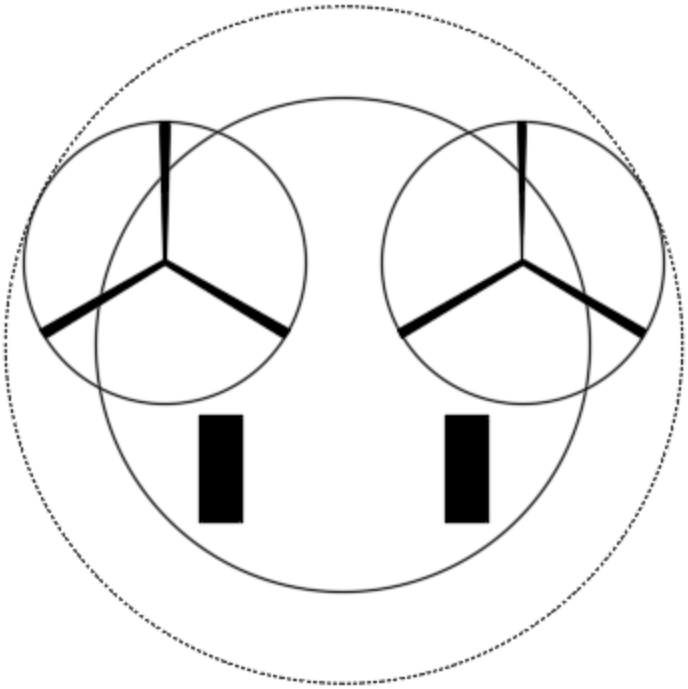


Fan



The design uses a single impeller fan and has a small suction hole at the bottom of the base to provide sufficient vacuum power to suck the dirt. It at maximum capacity can have a flow rate of 3 m3/ hr.

Brushes



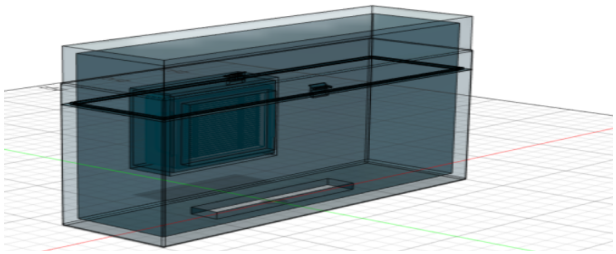
The brushes that will be used will be expected to rotate at 4500RPM . This transmitted centrifugal force will translate to a lot of stress on the rod. But as shown above the base plate made of ABS has high strength to handle the stresses.

The brushes are arranged in a circular pattern to provide complete all around coverage. The radius of the brush is taken to be 100mm to provide efficient coverage.

The outer circle depicts the effective area that the brushes can clean.

The brushes used can be of nylon to provide necessary strength and flexibility.

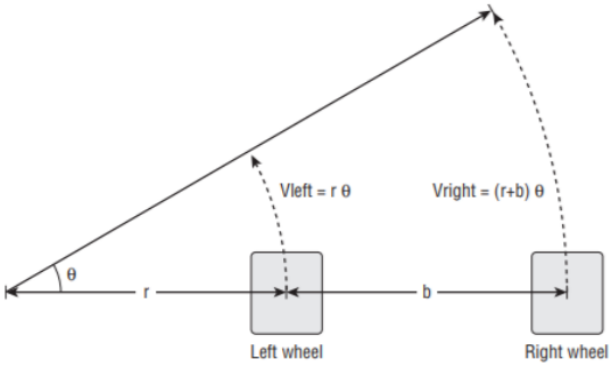
Bin



The bin to collect the dirt is also to be made of ABS for a sturdy build that does not wear with time. The bin has an internal volume of 712500 mm3 to hold the dirt.

Turning

The driving system as mentioned consists of 2 DC motors with individual differential control for adjusting the speed. The velocity value is specified in millimeters per second and describes the averaged velocity of the two drive wheels: ((Vleft + Vright)/2). From that equation you can see that a positive velocity makes the Robot go forward and a negative velocity makes it go backward. But since this is an average and Robot, like all real machines, takes time to come up to a speed and slow down from a speed, any command will result in positional error, and a series of commands with rapid starts and stops will accumulate position errors to an even greater degree. But for this project we will consider the motor to provide sufficient acceleration to make the time to reach max velocity negligible.



Vleft = rθ
Vright = (r+b)θ
V = (Vleft + Vright) / 2

By eliminating theta we get
Vleft = V(1-b/2r)
Vright = V(1+b/2r)

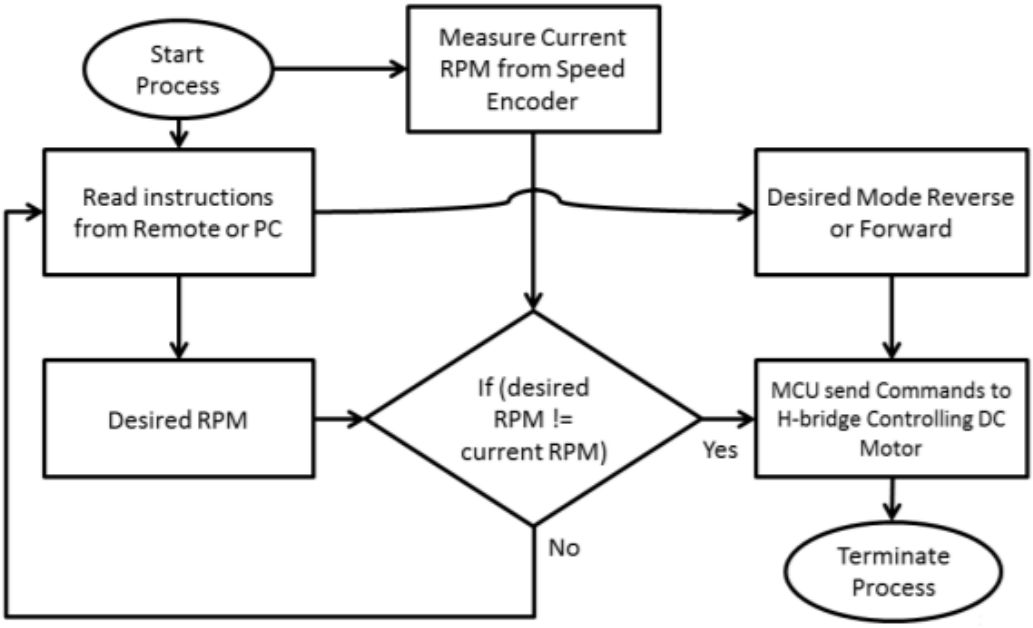
So we can adjust the speeds of each wheel in accordance with the necessary type of the turn that we need the robot to make.

Description	*Velocity*	*Radius*
Straight forward, medium speed	250	32768
Straight backward, slowly	-50	32768
Spin left in place	500	1
Spin right in place	500	-1
Turn left forward (left wheel stopped)	200	129
Turn right forward (right wheel stopped)	200	-129

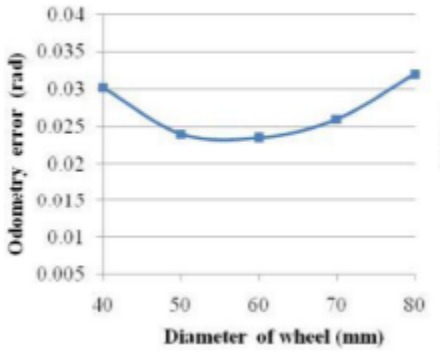
This table provides some examples of how the above equation can be used to perform turning calculations.

Slipping and Skidding

Skid and slip have been at large focus of researchers when considering the motion of the robot. The modern solution involves the placement of Rotary Incremental Motion Encoders to perform



the task of monitoring and controlling the slip and skid. Shown below is the control process.



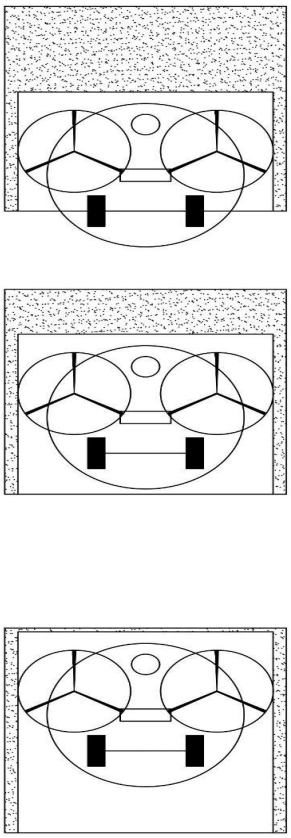
For our robot design we didn't have the resources to perform the above task. But the design and the geometric placement was done in such a way to minimize the above losses. The relation of slip with diameter is shown beside. As a result , considering the weight of the robot (approximately 4 kgs) and the required speed a good diameter size was about 70 mm as used in the robot. Even though the thickness of the wheel used by us is large and can increase slip, the larger thickness provides better grip at turning and the high powered motor used by us helps in negating the slip effect and helps in reducing the skid.

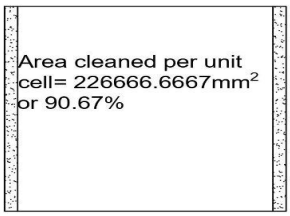
The wheel used is about 34 mm in thickness and 75 mm in diameter and has necessary threading to provide a good grip.

Cleaning Coverage

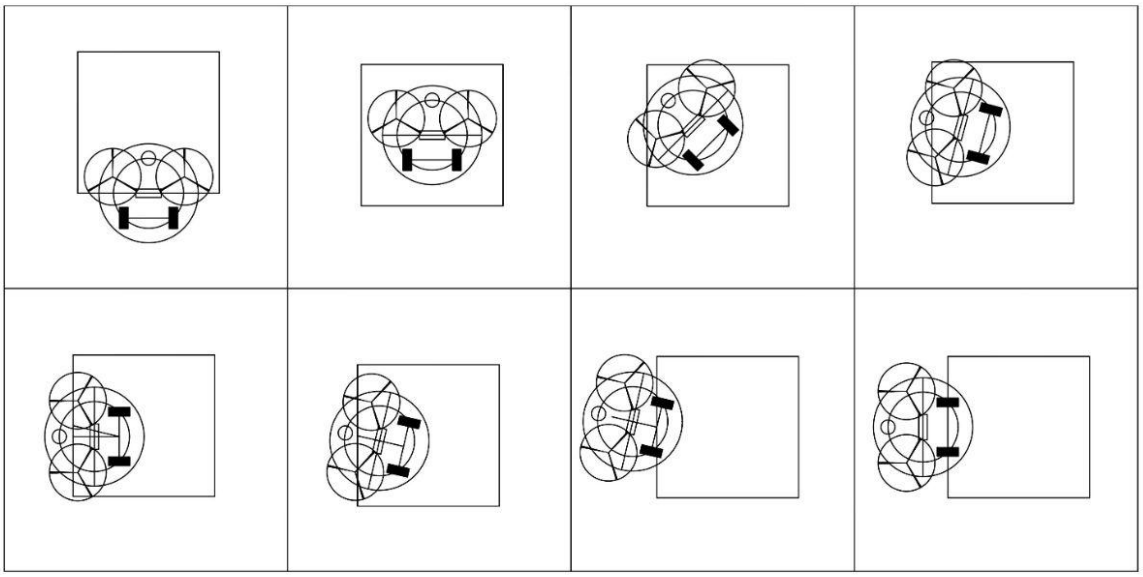
Straight path

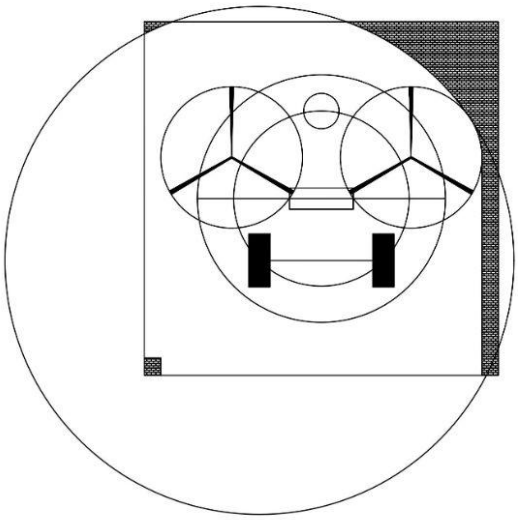
For all calculations a 0.5m X 0.5m unit cell is considered. The figure below shows the cleaning coverage of the robot on a straight path.





Turning Coverage





Area covered
=226950mm^2 or 90.78%

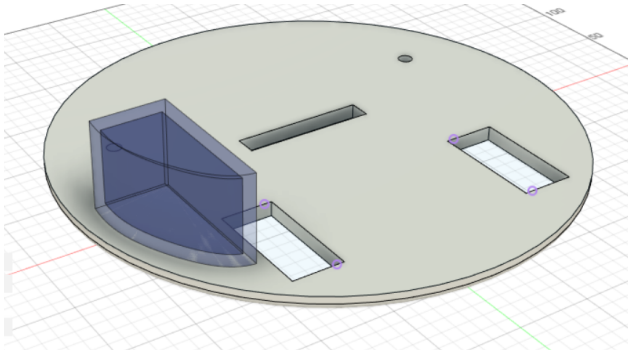
Turning Steps

1. The robot continues on the straight path till its geometric center aligns with the center of the unit cell.
2. The robot performs a 90° swing turn about one of its wheels(in the direction of the turn).
3. The robot is now not aligned with the center of the edge that it is facing, so to get aligned it turns about the center of the wheel axis for 13°.

   Then it moves straight for a 167.057 mm and now rotates in the opposite direction of initial 13° to realign the robot with the unit cell's edge.

Improvements

1. Better slip and skid control system
2. Can have a mopping function as well. The base plate has space on its sides to accommodate water tanks on both sides that can hold a combined 200 ml of water for cleaning purposes. The following design could not be added due to time constraints for design and power requirements of the pump.



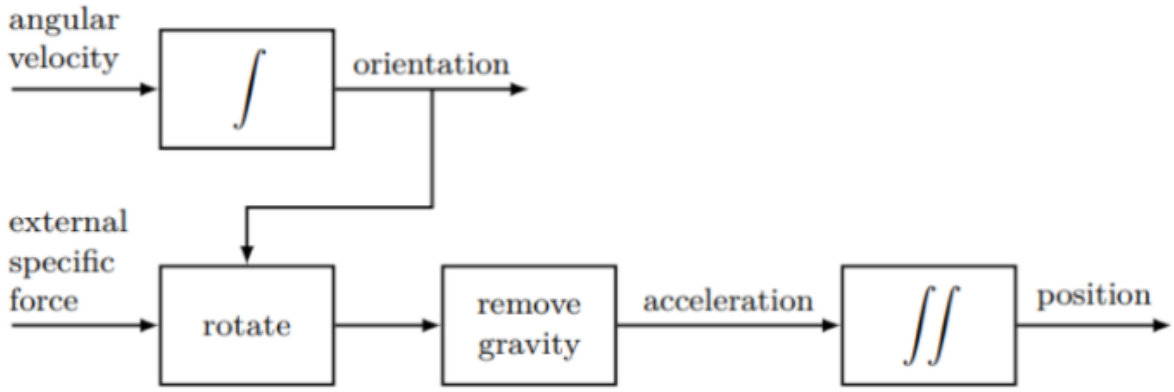
2.1.2. Electronics and Sensors

Sensors

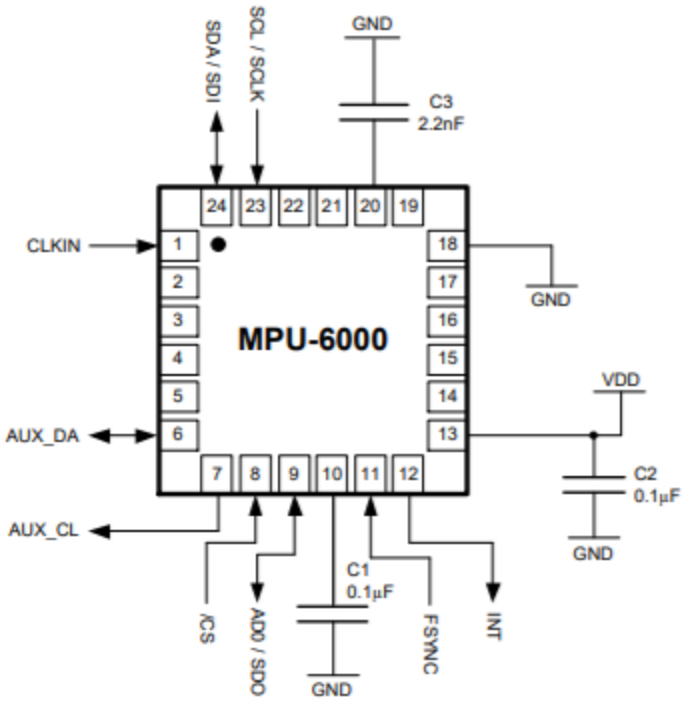
The sensors we have chosen to install in our bot and their selection process are explained below.

IMU

An Inertial measurement unit is fixed to the bot, so that we get the accurate position as well orientation of the bot at all times. This will be used in the navigation algorithm to provide commands to the bot. IMU consists of an acceleration sensor coupled with a gyroscope which gives us the angular velocity and external specific force acting on the body. This data is further processed to obtain the position and orientation.



After considering about 4-5 products, we chose the **MPU-6050** by **InvenSense** which has a triple-axis MEMS gyroscope and accelerometer, supporting both I2C and SPI protocols.



LIDAR

We utilize the Light Detection and Ranging (LIDAR) mechanism to map and understand the environment around the bot and thus proceed to navigate through it. The LIDAR module fit on top of the bot serves dual purposes of both understanding the environment and detecting obstacles if any. We also implement a sensor fusion algorithm to get a better accurate position of the bot with respect to the environment.

We chose the TF-LUNA Micro LiDAR for our purposes as it has a very good accuracy of about +/- 2% and works very well upto a range of 8m. Also it supports the I2C interface and hence can

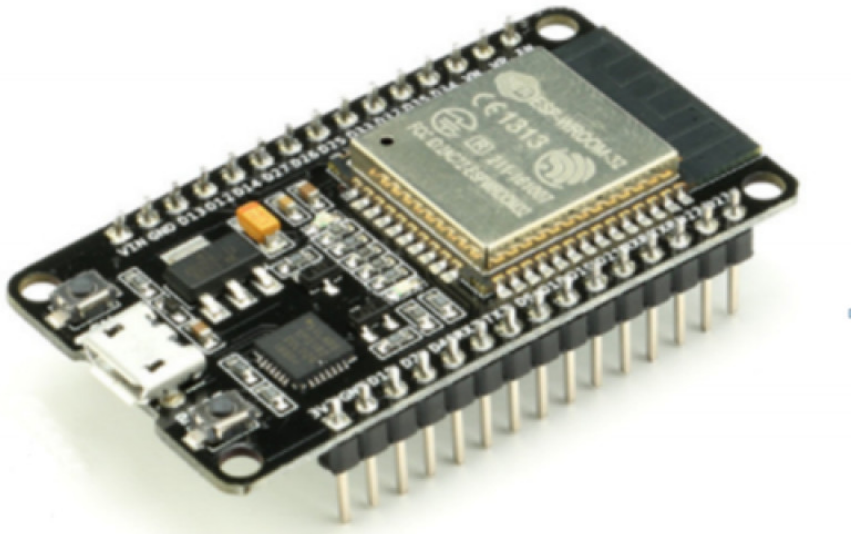
be easily integrated into the ESP32 system. It is built with algorithms adapted to various application environments and adopts multiple adjustable configurations and parameters so as to offer excellent distance measurement performances in complex application fields and scenarios.

Controller

Our Main controller, ESP32 WROOM board with a dual core processor, is a powerful, generic Wi-Fi+BT+BLE MCU module best suited for low-power sensor networks. It gives the best performance for electronic integration, range, power consumption, and connectivity. These are the hardware specifications :

Categories	Items	Specifications
Hardware	Module interfaces	SD card, UART, SPI, SDIO, I^2C, LED PWM, Motor PWM, I^2S, IR, pulse counter, GPIO, capacitive touch sensor, ADC, DAC, Two-Wire Automotive Interface (TWAI®), compatible with ISO11898-1 (CAN Specification 2.0)
	On-chip sensor	Hall sensor
	Integrated crystal	40 MHz crystal
	Integrated SPI flash	4 MB
	Operating voltage/Power supply	3.0 V ~ 3.6 V
	Operating current	Average: 80 mA
	Minimum current delivered by power supply	500 mA
	Recommended operating temperature range	–40 °C ~ +85 °C
	Package size	(18.00±0.10) mm × (25.50±0.10) mm × (3.10±0.10) mm
	Moisture sensitivity level (MSL)	Level 3

ESP32 Wroom Module



Navigation and Control

The data from IMU enables us to calculate the current location of the bot given its initial location. But since this is achieved via integration over time, it is prone to accumulative errors. Hence to get a reliable estimate of the location of the bot, we use the LiDAR to measure the distance from known wall boundaries. Now a noise-filtering algorithm such as a Kalman-filter is used to get precise information on the location and orientation of the bot.

Thus the position and orientation of the bot would be obtained by fusing the data from IMU and LIDAR and would be fed to the algorithm in the controller which decides the linear velocity and angular velocity of the bot at the next time step. The corresponding speeds of individual wheels in order to achieve this, is sent to the motors via the motor drivers. Thus a course-corrective feedback loop is set up to ensure that the bot follows the intended path.

Battery Pack

Crucial aspects we considered while choosing the power solution for the bot were :

1. Provide enough power to all components
2. Should last long enough for one clean without having to recharge in middle
3. Minimum calendar ageing

Based on our calculations, we decided that the battery pack must be of 14-18 V with a minimum capacity of 3000 mAh. It must last at least for an hour without having to recharge in the middle.

We chose to go with a battery pack consisting of **LG HG-2 18650 3000 mAh Li-ion batteries**, as they had superior energy density and longer calendar life.

Motors

We need two kinds of motors in total for our bot, One to drive the wheels and the other to drive the vacuum fan. We explain our selection for the same in this section.

DC Gear Motor

To drive the wheels, We have the RS-540SH-5045 with a 50 W power delivered to wheels. This is a 12V DC motor which can run till 18000 RPM.

DC Motor - Fan

We would need a DC motor to run the vacuum fan which should provide a reasonable RPM powerful enough to collect dust. We have chosen the 12V - 3000 RPM DC Motor for this purpose.

2.1.3. Multi-Agent Complete Coverage Path Planning

With the Map given to us along with its obstacles, we have a given area that we need to cover. Firstly we need to decide how many bots we will be using to clean the assigned area, and secondly, we need to decide where each bot starts.

According to the problem statement, we need to use a minimum of 2 bots and hence we have considered cases from 2 bots upto 7 bots. For maps 2 and 3, we have limited the maximum number of bots to 5 since there were many obstacles which caused the effective area to be reduced. Thus, even with a lesser number of bots, we were able to clean the area in a decent amount of time.

Next is the initial position. For planning the path, we have discretized the map into unit cells of 0.5mX0.5m each. The bots will initially start at the centre of these unit cells. Hence, we define the initial position by specifying the unit cell in which the bot starts. The manner in which they have to be defined is mentioned in the Github Description mentioned in the References.

Our Objective was to design robot paths that **completely cover** this area of interest in the **minimum possible time.**

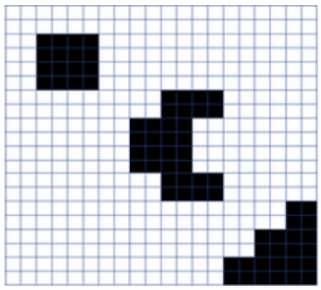
The key attributes that affect this performance are :

- Eliminate Backtracking
- Leave no area uncovered
- Equal Coverage Trajectory for Each Bot

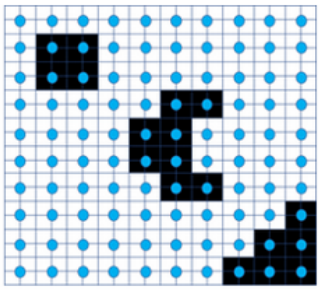
We have implemented the DARP Algorithm to do this and in term optimize the cleaning of the assigned area.

The above mentioned bullet points ensure optimal cleaning in the least amount of time. With DARP covering all the unit cells, the cleaning efficiency in terms of unit cells cleaned is always 100% and hence to compute the cleaning efficiency, we can simply use the cleaning efficiency per different unit cells (straights and turns) and calculate the weighted cleaning efficiency accordingly.

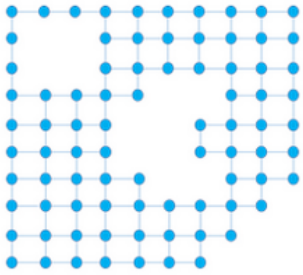
To ensure that the above bullet points are covered, DARP uses **Spanning-Tree Coverage (STC)**, which computes the optimal robot path under some mild discretization assumptions. In a nutshell, STC Algorithm :



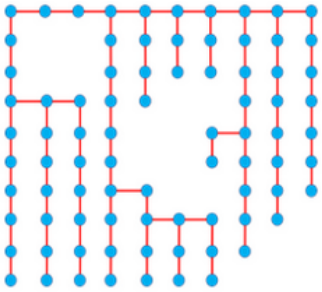
(1) Area discretization



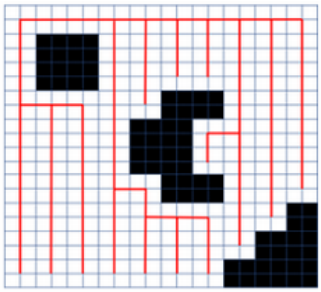
(2) Convert cells to nodes



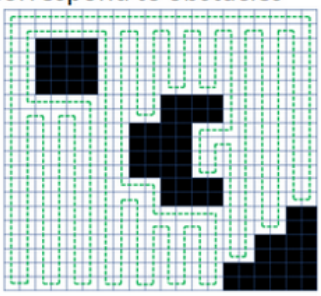
(3) Discard nodes and edges that correspond to obstacles



(4) Construct a Minimum Spanning Tree



(5) The resulting MST is "our guide" for the navigation



(6) The final path that circumnavigates the MST

DARP implements this Minimum Spanning Tree concept for multiple robots and thus we have a system which requires the following inputs :

- Map-size : size of the map which is being considered
- Obstacle positions : linear-grid locations of the obstacles
- Bot Initial positions : initial linear-grid locations of the cleaning bots (all bots)
- Trajectory coverage weightage : weightage of trajectory covered by each bot

For each combination of the above parameters, the portion to be covered by each bot is designated. This designated portion can be covered by the MST lengthwise or breadthwise whilst being connected at either of the sides. This gives us 4 possible modes. These modes have different amounts of cells to be covered straight and to be turned on.

Due to computational limitations, we can not find the time for all the cases, but we have taken 10,000 cases of bots positioned randomly, such that each bot is at least 10-15 meters away, mutually, from each other. For more bots, we have reduced this number to 1000 cases.

Furthermore, if a single case is to be calculated, the number of iterations after which the code shows that a solution can not be found is 80,000. To account for the computational complexity, we have reduced that number to 1024. Although if only a single simulation is to be run, we can increase that number.

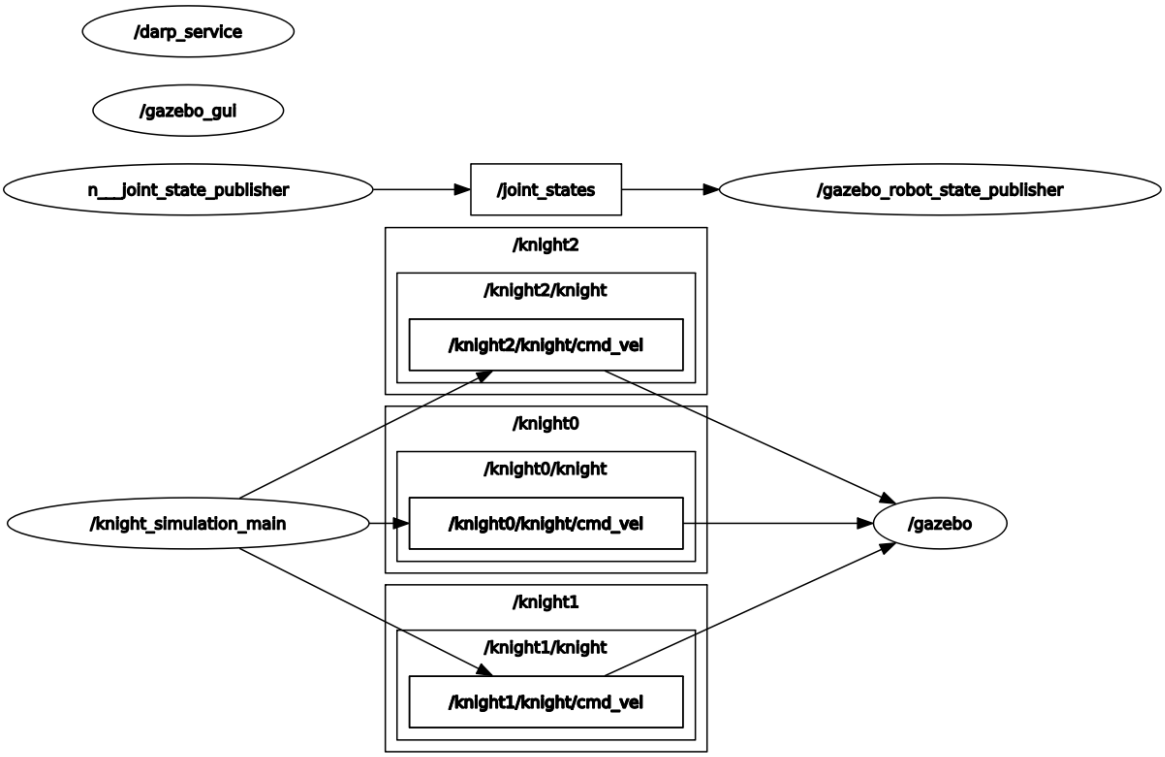
Thus, additional parameters which we have tweaked are :

- maxIter : number of iterations after which the code prints "solution can not be found"
- Mutual_dist : minimum distance between any 2 bots

We have set up a script to generate the random position configurations and then automated the running of the DARP algorithm for these locations.

The calculation and analysis of the output Data is explained in the Results section.

2.1.4. Simulation



Our current simulation setup allows us to dynamically change any of the parameters (including robots) without any recompilation or setup required, which makes testing and simulation workflow very easy. This is done with the help of the ROS parameter server.

The simulation also employs multithreading so that any of the bots don't have to wait for any other bot's communications.

The path is fetched from the /darp_service, which is very fast because it pre-calculates the path in the start.

The implementation is also not very heavy on the resources because at any point of time, the majority of the data quanta being communicated are less than 20 bytes in size.

2.3. Cost Estimation

Component	Qty	Cost
IMU	1	Rs. 250
LIDAR	1	Rs. 2000
Controller	1	Rs. 500
LG HG2 18650	4	Rs. 2000
DC Gear Motor	2	Rs. 1500
DC Motor	1	Rs. 90

Total Cost : Rs. 6340

Component	Approximate price
Body (ABS)	300 Rs (110 rs / kg ABS)
Brush	100 Rs (nylon bristles cost 340 rs/ 10 kg)
Wheel	400 Rs (200 per wheel)
Fan	200 Rs

Total = 1000 Rs

Total Approximate Cost per bot = Rs 7340

3. Results

3.1. Time Analysis

By running the DARP algorithm for various combinations of initial positions of bots, we get the number of straight line moves and number of turns for each bot on each map for each initial configuration. Also, we know the speed of the robot in a straight line and it's turning speed from the mechanical design section.

Using both the above mentioned facts we can find the time taken by each bot for a given configuration. Now, naturally the maximum out of the amount of the time taken by each bot to cover its path for a given configuration is the cleaning time of that configuration.

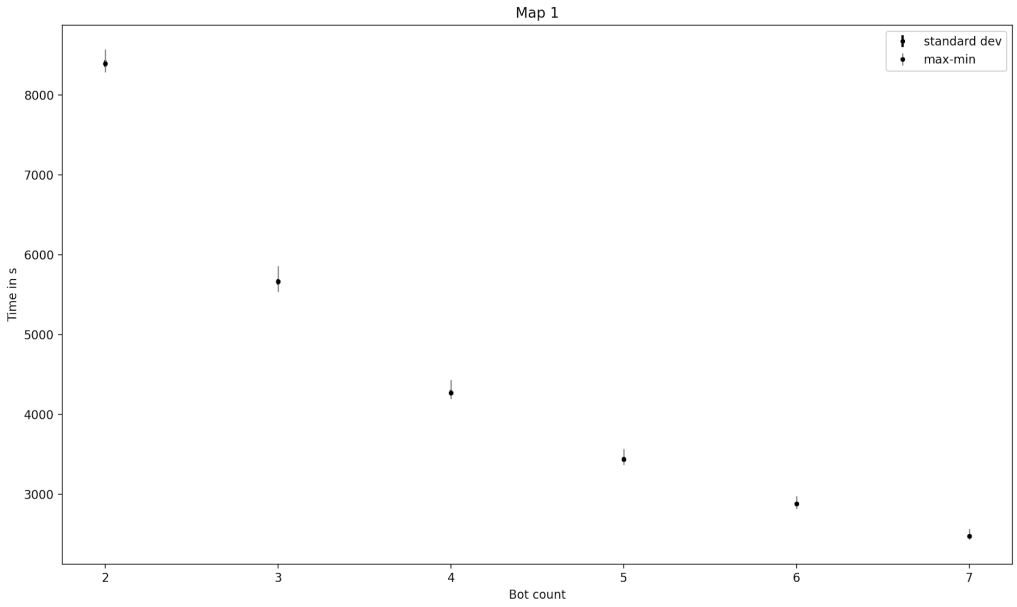
We have plotted the results below, wherein each graph contains timing data for a particular map for various bot counts. For each bot count we have plotted two error bars, one (thin) depicting the difference in the minimum and the maximum time for a configuration for that number of bots while the other (thick) denotes the standard deviation among the times calculated for various configurations. The dot at the centre represents the average time for that bot count.

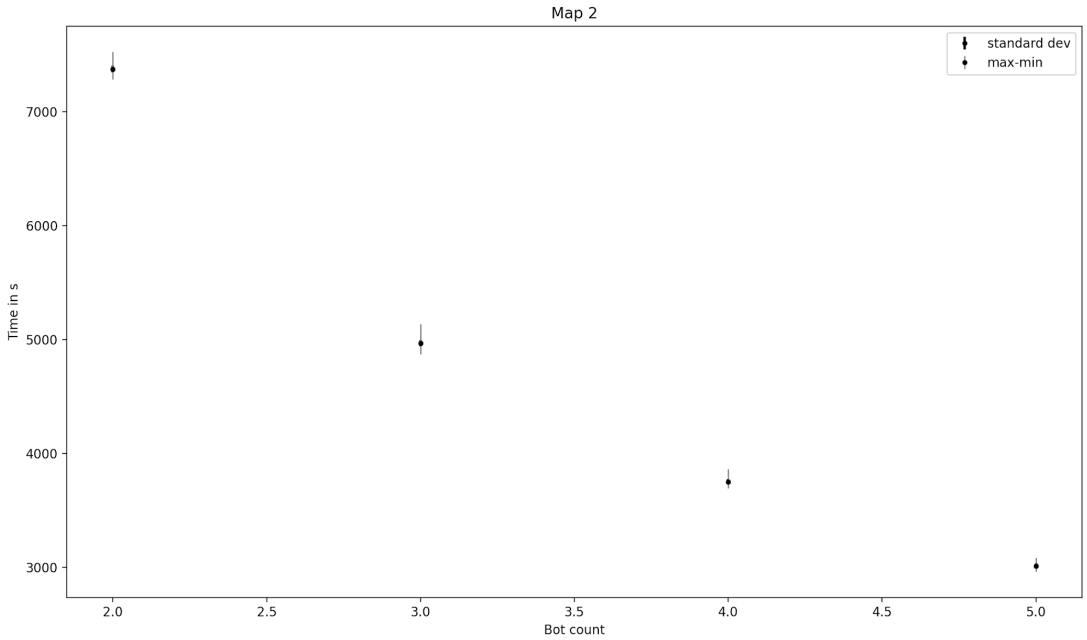
Following our natural intuition, the time taken for cleaning the map keeps on reducing with increasing bot count.

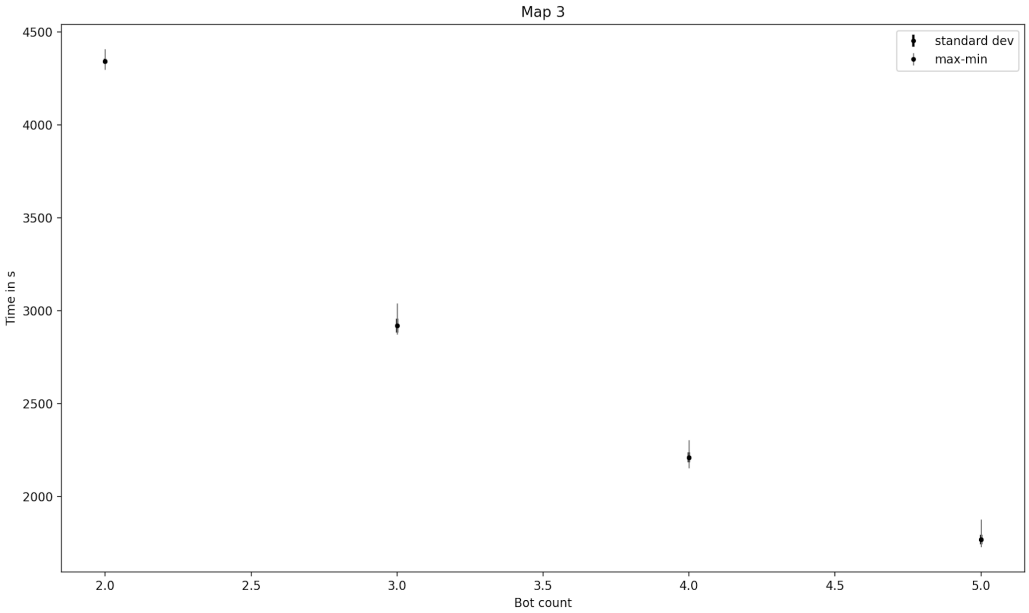
The mathematical model used to calculate the times is,

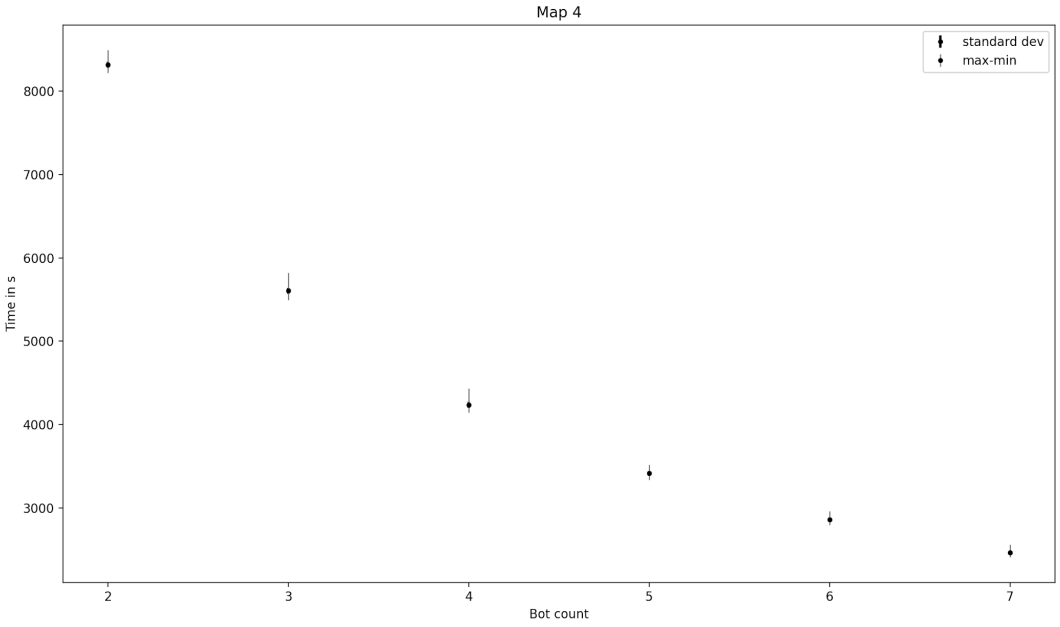
```
V_straight : 0.305m/s

T_straight : 0.5m/V_straight nearly= 1.65s

T_turns : 2.5s

T_total : T_turns*num_turns + T_straight*num_straights
```

Where `num_turns` and `num_straights` are the number of turns and straight moves for a bot in a given configuration.


Map 1
standard dev
max-min
Time in s
8000
7000
6000
5000
4000
3000
2
3
4
5
6
7
Bot count

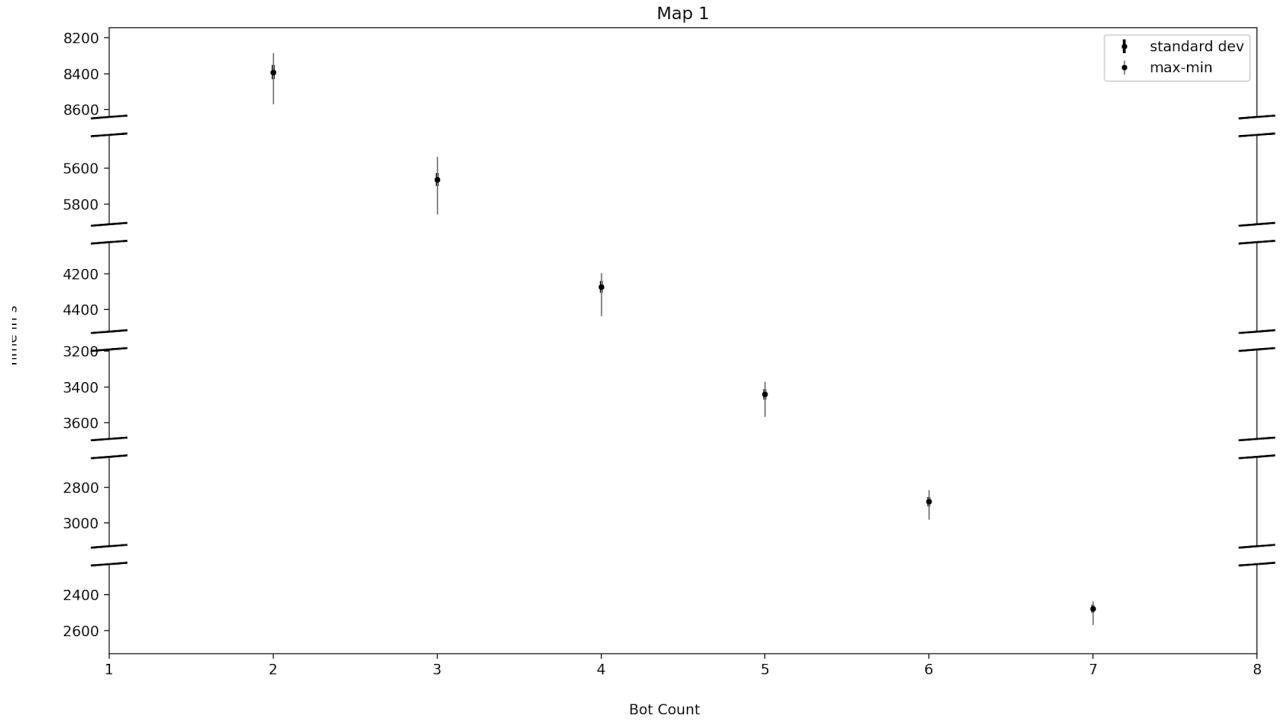

Map 2
standard dev
max-min
Time in s
7000
6000
5000
4000
3000
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Bot count

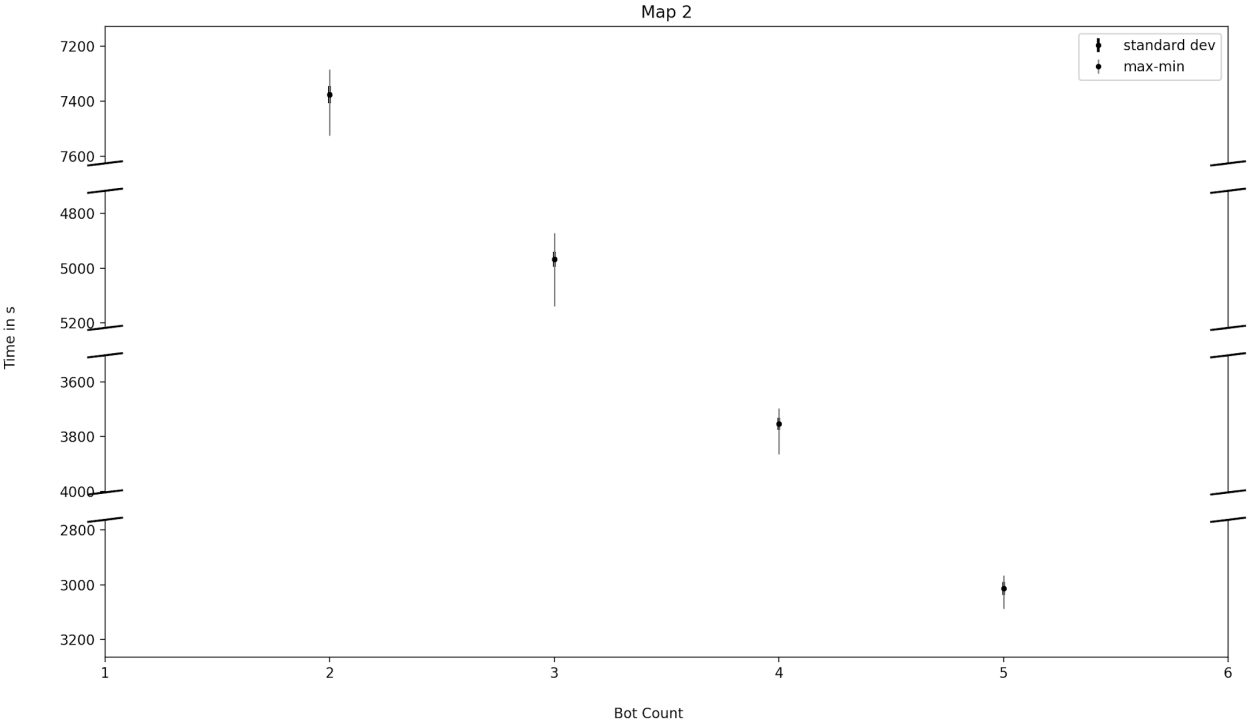


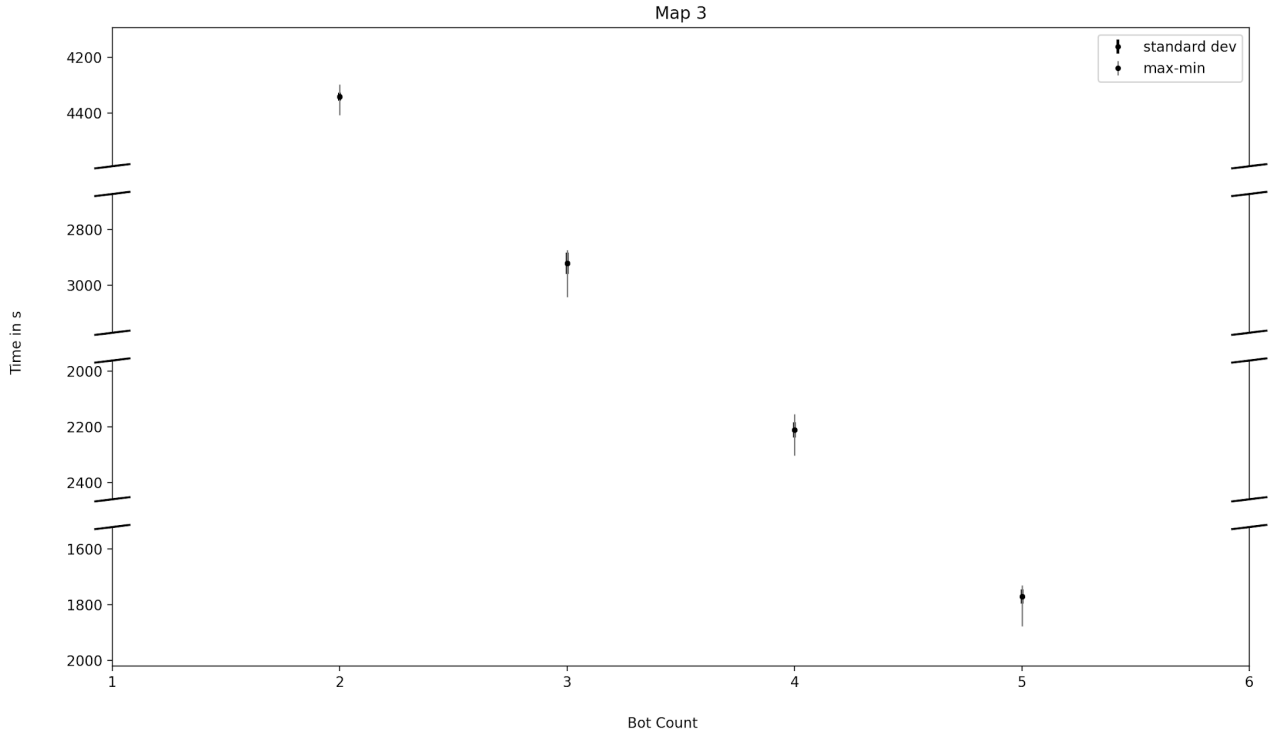


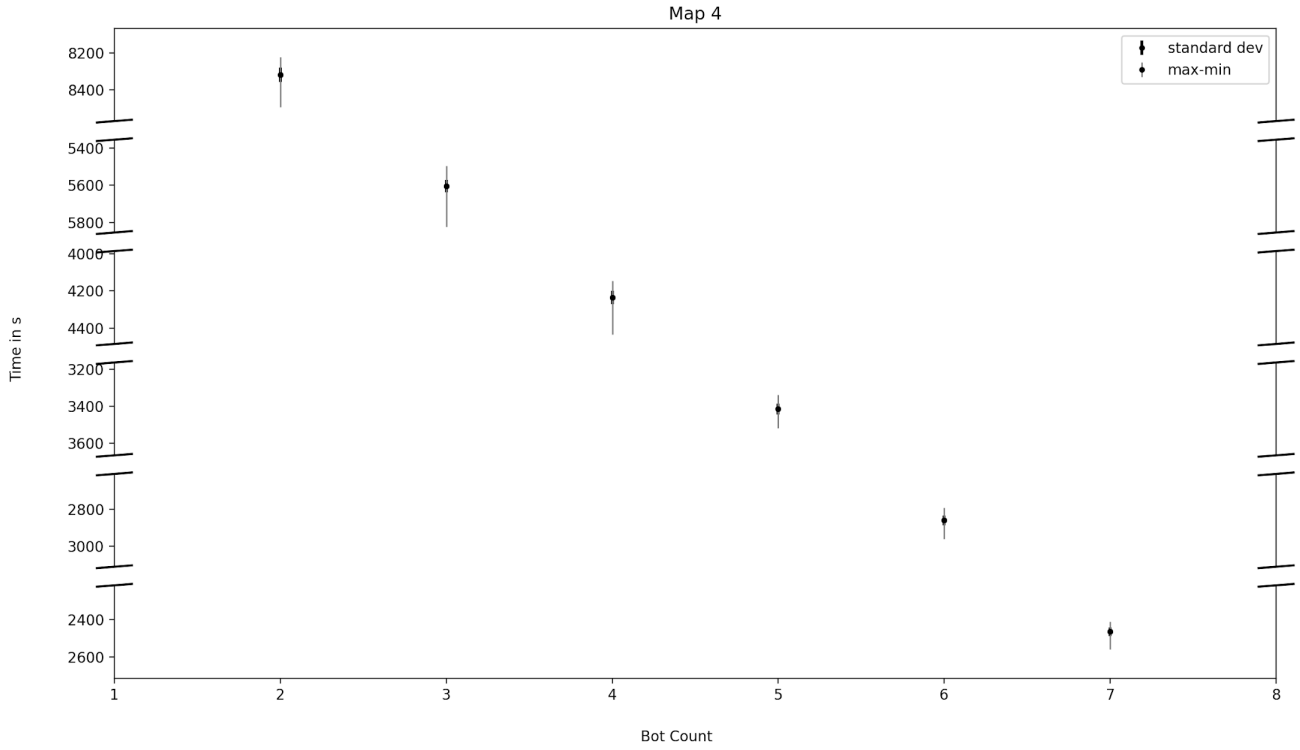
Now to enhance the properties of the plot, changing the y-axis demarcations.

By doing so, you can see the difference between the extremities and the standard deviation of the time taken for the group of simulations to run. We have simulated the case with the minimum runtime and hence we have stored the position which gives this minimum runtime.


Map 1
standard dev
max-min
8200
8400
8600
5600
5800
4200
4400
3200
3400
3600
2800
3000
2400
2600
1
2
3
4
5
6
7
8
Bot Count


Map 2
standard dev
max-min
7200
7400
7600
4800
5000
5200
3600
3800
4000
2800
3000
3200
Time in s
1
2
3
4
5
6
Bot Count


Map 3
standard dev
max-min
Time in s
Bot Count
4200
4400
2800
3000
2000
2200
2400
1600
1800
2000
1
2
3
4
5
6


Map 4
standard dev
max-min
Time in s
Bot Count
8200
8400
5400
5600
5800
4000
4200
4400
3200
3400
3600
2800
3000
2400
2600
1
2
3
4
5
6
7
8

Map	Num_Bots	T_min	T_mean	T_stdDev
Map1	2bots	8282.78	8392.33	40.80
	3bots	5535.98	5663.91	35.22
	4bots	4194.75	4273.44	33.38
	5bots	3369.91	3440.96	29.81
	6bots	2815.65	2880.30	26.68
	7bots	2435.65	2478.66	22.66
Map2	2bots	7285.04	7376.95	31.27
	3bots	4871.14	4967.64	28.08
	4bots	3696.31	3753.46	22.90
	5bots	2966.06	3013.20	23.48
	6bots	--	--	--
	7bots	--	--	--
Map3	2bots	4298.19	4342.44	14.65
	3bots	2872.74	2921.02	38.29
	4bots	2154.42	2210.91	26.88
	5bots	1729.79	1770.15	25.96
	6bots	--	--	--
	7bots	--	--	--
Map4	2bots	8223.15	8318.07	38.82
	3bots	5494.83	5604.42	34.37
	4bots	4145.73	4235.53	37.32

	5bots	3337.45	3414.24	29.35
	6bots	2792.86	2860.30	26.72
	7bots	2409.42	2464.13	23.58

3.2. Cleaning Efficiency

As mentioned in the Time Analysis section, after running the simulation, we have in our hands the number of straight line moves and turns for each bot for each configuration for all maps.

And also from the Mechanical Design section, we know the efficiency of our robot in straight lines and while turning. So knowing the total combined straight moves and total combined turns for all the bots, we calculated the average efficiency and its standard deviation of all the configurations for a specific bot count, for a specific map for all bot counts and for all maps.

The mathematical model used to calculate the efficiency is,

```
eff_straights : 90.66%

eff_turns : 90.78%

Map_eff : eff_straights*num_straights + eff_turns*num_turns
```

Where `num_straights` and `num_turns` are the total straight moves and turns for all bots combined for a given configuration.

Map	Num_Bots	Eff_mean	Eff_stdDev
Map1	2bots	90.66481	0.00084
	3bots	90.66653	0.00101
	4bots	90.66736	0.00100
	5bots	90.66820	0.00094
	6bots	90.66890	0.00087

	7bots	90.66951	0.00077
Map2	2bots	90.66715	0.00076
	3bots	90.66857	0.00098
	4bots	90.66945	0.00092
	5bots	90.67003	0.00087
	6bots	--	--
	7bots	--	--
Map3	2bots	90.66709	0.00099
	3bots	90.66901	0.00195
	4bots	90.66993	0.00139
	5bots	90.67041	0.00135
	6bots	--	--
	7bots	--	--
Map4	2bots	90.66548	0.00087
	3bots	90.66704	0.00103
	4bots	90.66810	0.00121
	5bots	90.66893	0.00101
	6bots	90.66957	0.00097
	7bots	90.67031	0.00089

3.3. Optimal Number of Bots

Since we know the range of time taken for a given number of bots (2-7) to clean the area, and the cost of each bot, we can evaluate the cost vs time saved and **it is upto the company adopting this solution to decide how much their time is worth**

4. Side Notes

- **Size of bot vs cell resolution :** cell resolution should be equal to the size of robot for the optimal cleaning calculation mentioned in the proposal.
  For a 50m*50m map, taking the resolution to be 0.5*0.5 gives us a 100*100 grid. Running the algorithm multiple times for multiple configurations on a 100*100 grid is a task in itself.

  Taking a 0.35*0.35 resolution, we would have obtained a 145*145 grid.which would not be possible on the limited computational power that we have. Hence we have used a resolution of 0.5*0.5m for the path planning purposes while limiting the robot size to 0.35*0.35m as per the mentioned specifications

- **Initial robot configuration setup :** For robots placed close by, it takes a much larger number of iterations to get to the solution. Due to limited computational power, we have only considered cases where the robots are ‘atleast’ 10-15 meters apart.

  The Data for the number of combinations taken into considerations and the number of left out due to
  this constraint has also been recorded to give a clear idea of the amount of data being used to conclude our result.

  With higher computational power, we can find optimal paths for all possible combinations of initial positions of the robots.

- **Minimum Resolution of Obstacles :**
  To find a Minimum Spanning tree, the obstacles need to occupy an even number of cells in length and width.

  Thus, the resolution upto which an obstacle can be stated as in input depends on the resolution of a unit cell. As we increase the resolution, the paths get more freedom and hence the runtime to get the solution also increases.

  With the limited computation power at hand, we have considered 0.5mX0.5m a decent resolution to serve the purpose of demonstrating our project idea.

6. References

Mechanical Design:

Robovac
Reference report
turn Calculations
Slip calculations
Geometry
Dynamics

Electronics and Sensors:

Robot Vaccum Cleaner - Report
Design of an Autonomous Cleaning Bot
MPU-6000 and MPU-6050 Product Specification Revision 3.4
Product Manual of TFmini-S
LG HG2 18650 - DATASHEET
RS-540SH-5045

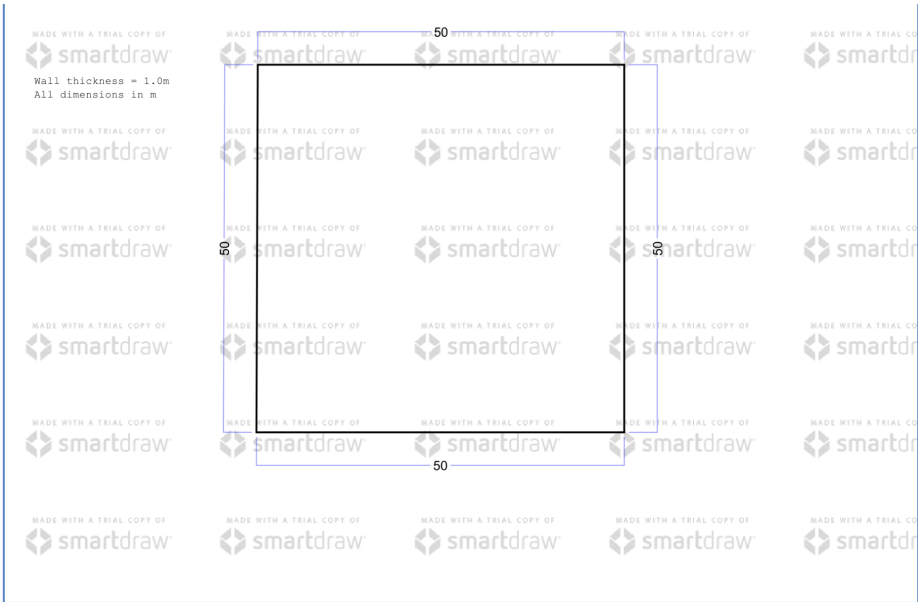
Multi-Agent Complete Coverage Path Planning:

DARP - Java GUI
DARP - Python
DARP: Divide Areas Algorithm for Optimal Multi-Robot Coverage Path Planning
DARP_Divide_Areas_Algorithm_for_Optimal_Multi-Robot_Coverage_Path_Planning

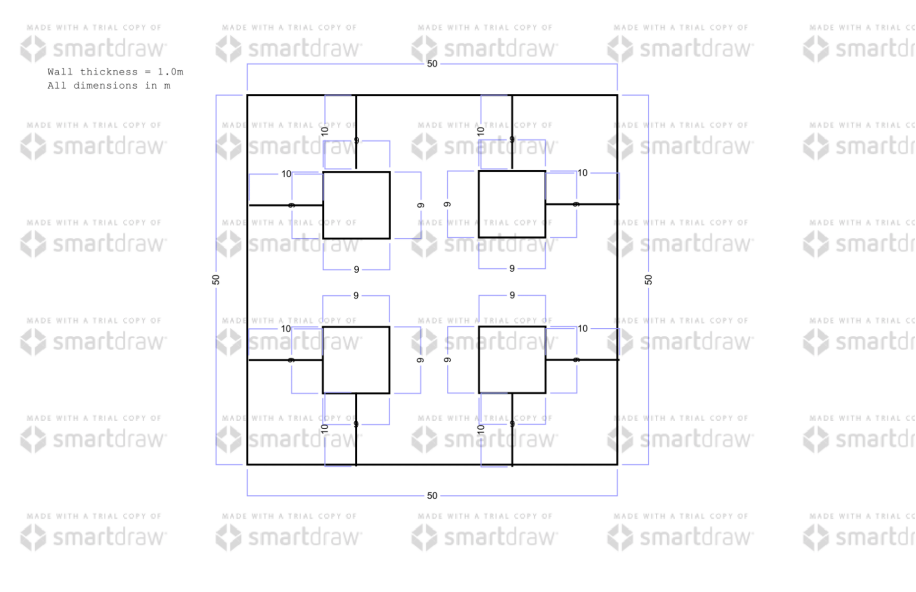
7. Appendix

7.1 Detailed Maps with Dimensions

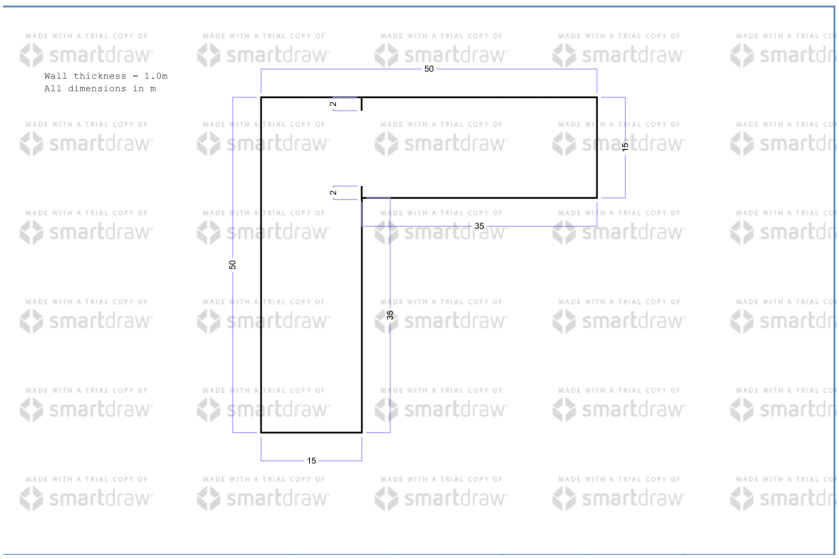
Map1



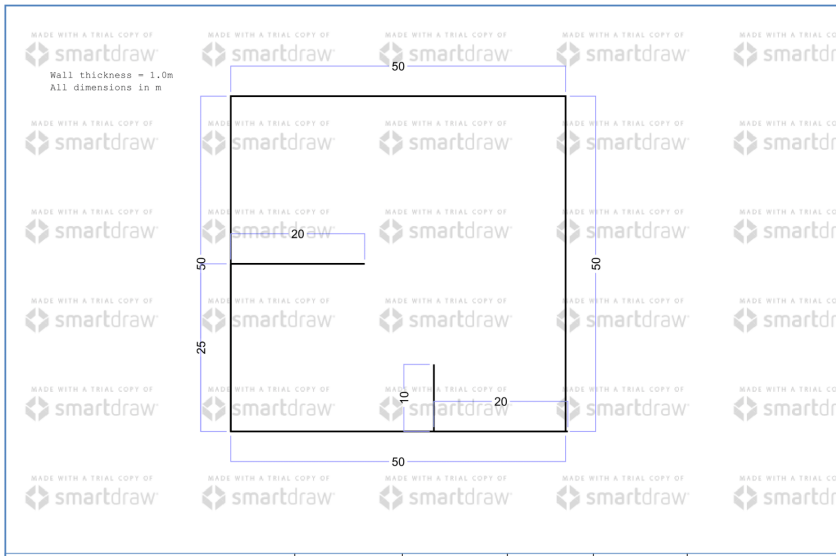
Map2



Map3



Map4



These are the maps that have been used in the path planning algorithm and while simulation of the robot on ROS.