[ ] ~~bc cocksure pumpscrew~~
- Rupan try ur best

[ ] ~~change the electrical archi~~
- Ming yuan ask eugene

[ ] ~~pull down resistor for microswitch~~
- Kj ask

[ ] ~~nfc docking - my~~
- My edit

[ ] ~~esp32 model ask prince~~
- Kj ask

[] ~~buy stuff~~
- Aditi to send link for confirmation and buy by tmr

[] ~~discuss cupholder~~
- 3d print base, nail aluminium sheet on the cup holder, rupan

[] ~~mdf alternative~~
- Kj dp sets wood

To Buy (aditi)
- PN532 (slim version) + tags 20th mar
- Number pad 20th mar
- Find adapter and cable with correct power rating 5V 3A

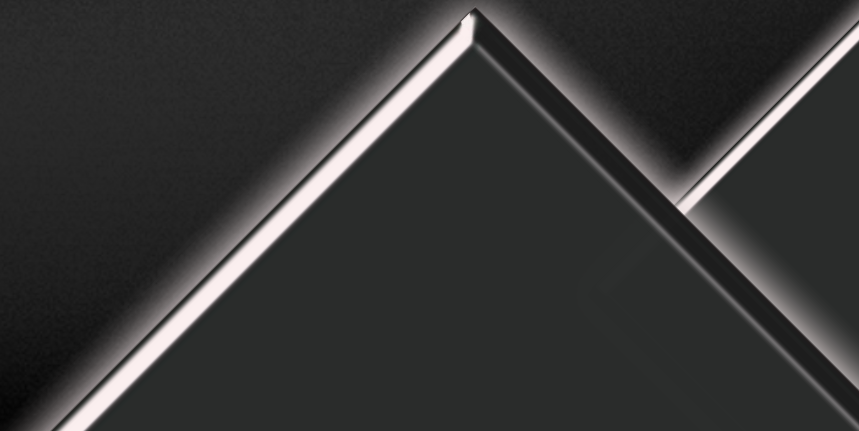
EG2310: Design Review (Group 9)

Rupan
KJ
Ming Yuan
Aditi

TABLE OF CONTENTS

01

INTRODUCTION



OVERVIEW

Problem Definition

No.	Purpose	Description
1	Autonomous Navigation of the turtlebot	- Determines the location of the robot - Able to navigate autonomously to the designated table number
2	Physical Interface of the entire system	- Deliver the payload from the dispenser to the robot - Detection of load by the bot after can is dispensed.
3	Digital Interface between the turtlebot and dispenser	- Takes the table number input from the user via dispenser and transmit it to the turtlebot. - Implementing a precise docking system
4	Mechanical Designs	- Modifying the bot with a feasibly designed cup holder - Making an efficient dispensing structure

OVERVIEW

Components - Dispenser

Functionality - To receive the can from the TA and smoothly dispense it into the Turtlebot after taking in the input table number from the TA.

- Mechanical design - We plan to use a mechanism similar to vending machines. The design involves three basic components - the frame, coil and rails made up of wood, steel and aluminium respectively.

- Electrical configuration - The dispenser will consist of three main electrical components - Servo motor (to rotate the coil), ESP32 (mode of communication between the Turtlebot and dispenser), and the numpad (to take in the input of table number).

- Software - Communication between ESP32 and the R-pi

Components - Turtlebot

Functionality - To receive the can from the dispenser, successfully navigate its way to its destination autonomously and dock back to the dispenser for the next order.

- Mechanical design - To carry the payload, we have modified the turtlebot by placing a fifth waffle on the top along with a one-side open cup holder.

- Electrical configuration - PN532 will be installed on the bot to detect the NFC tags near the dispenser so that there is proper docking. For can detection in the bot, we will place a microswitch that will activate when it detects load.

- Software - Autonomous navigation

OVERVIEW

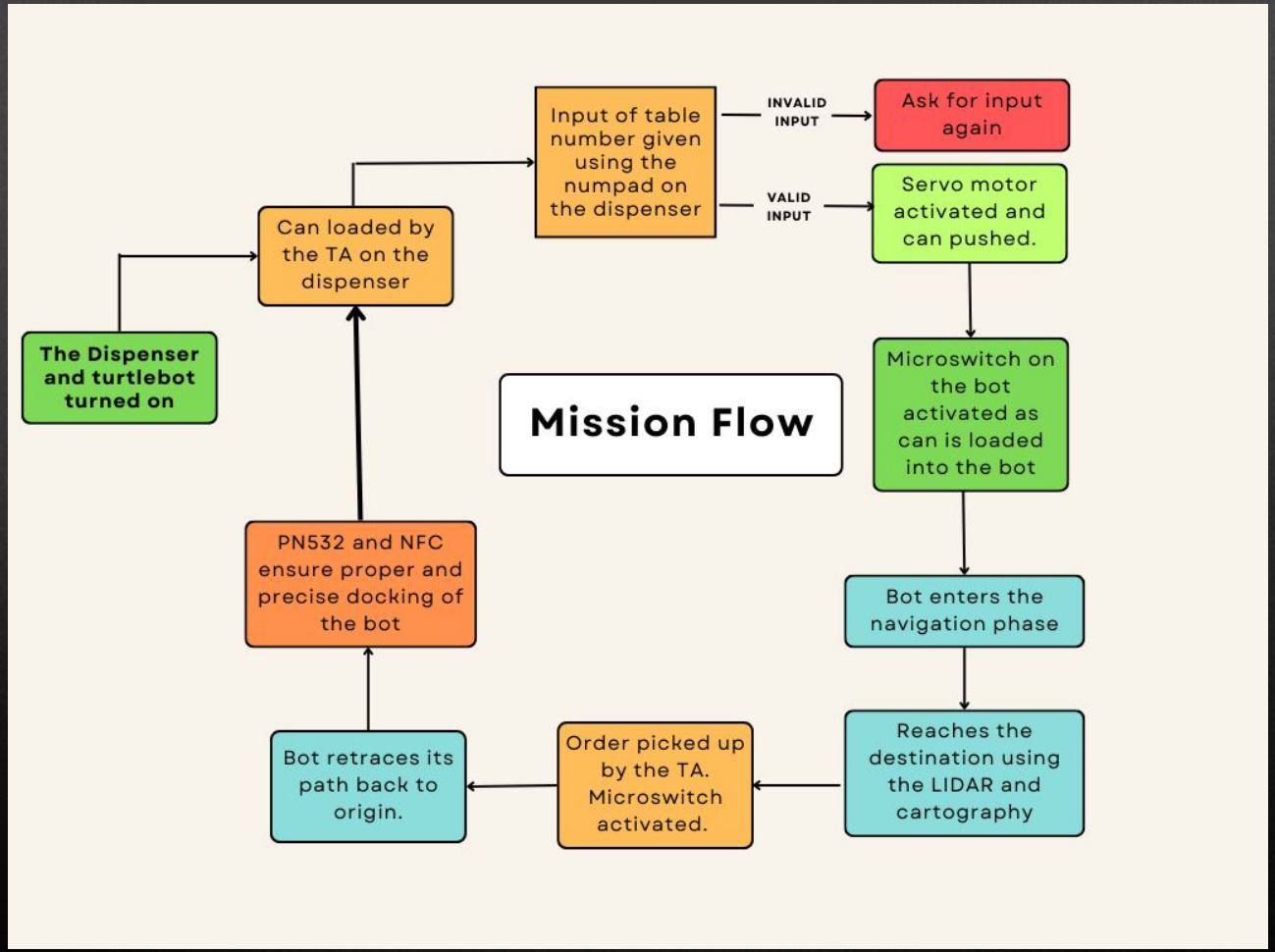
01 Problem Definition

02 Components

03 Flow

04 Mechanical Design

Flow


Mission Flow
The Dispenser and turtlebot turned on
Can loaded by the TA on the dispenser
Input of table number given using the numpad on the dispenser
INVALID INPUT
Ask for input again
VALID INPUT
Servo motor activated and can pushed.
Microswitch on the bot activated as can is loaded into the bot
Bot enters the navigation phase
Reaches the destination using the LIDAR and cartography
Order picked up by the TA. Microswitch activated.
Bot retraces its path back to origin.
PN532 and NFC ensure proper and precise docking of the bot

OVERVIEW

01 Problem Definition

02 Components

03 Flow

04 Mechanical Design

Mechanical Design

Dispenser

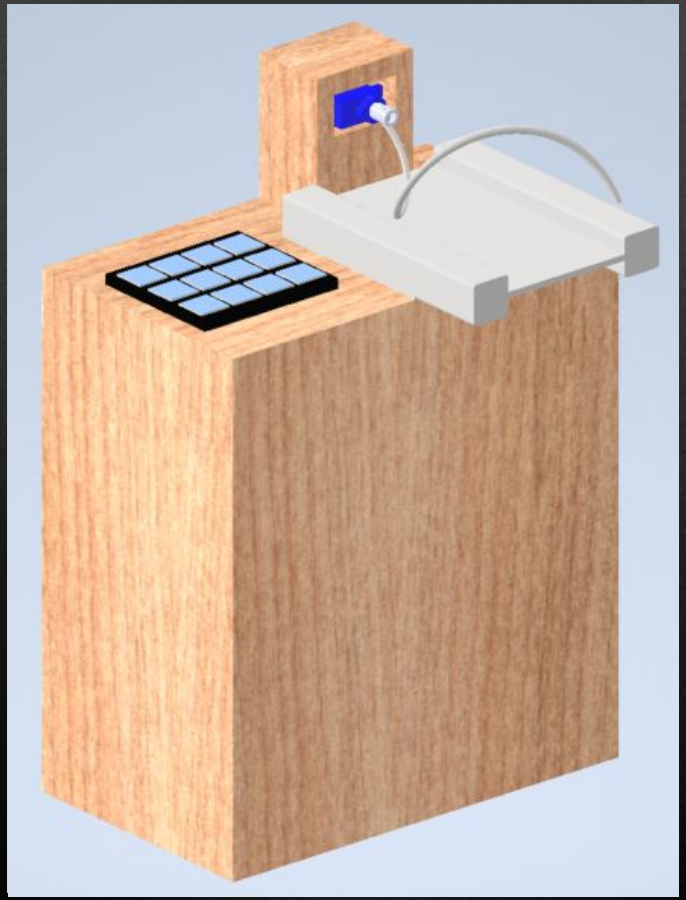
- Receive the can from the TA
- Facilitate the smooth transfer of the can onto the turtlebot
- Take input of table no.

Turtlebot

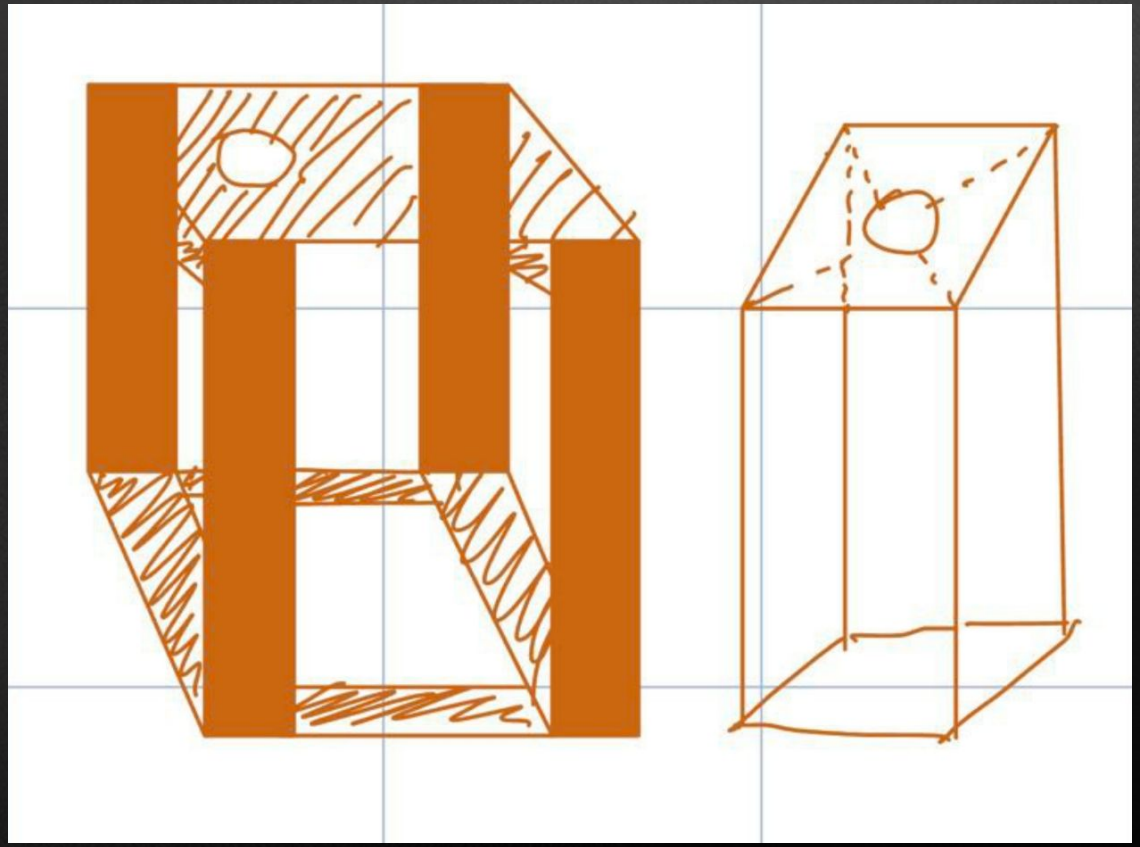
- Receive the can from the dispenser
- Ensure that the can is not dropped, shaken or damaged
- Loading of the can does not affect operation, i.e. can move smoothly, is mechanically balanced

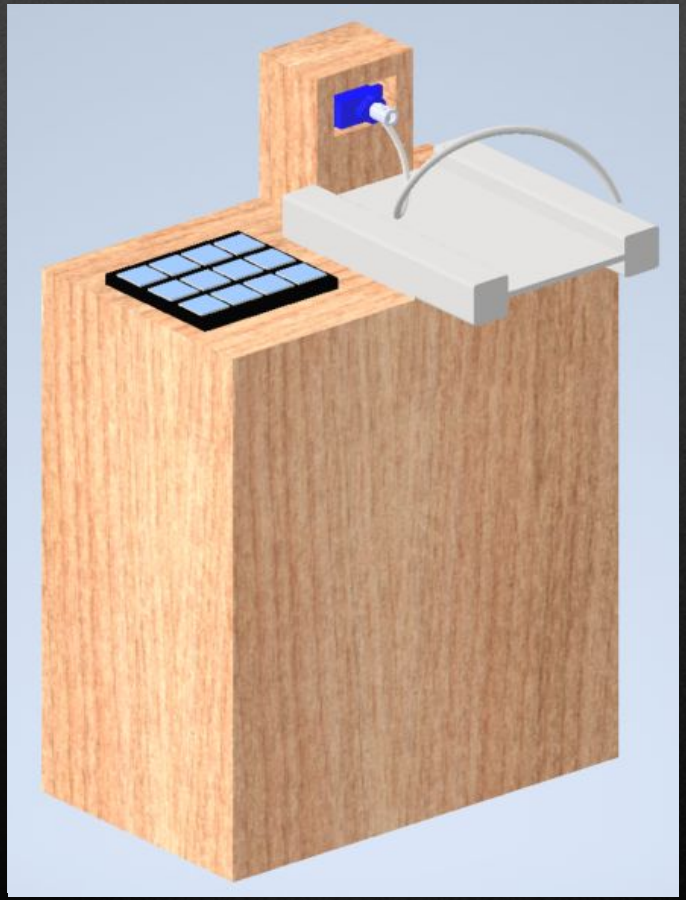
Dispenser

- Cuboid shaped frame, 220x132x250 (LxWxH)
  - Made using cut MDF (Medium Density Fiber Wood) panels, 5 panels attached using PVA glue and fastened with wood screws
    - Cheaper than plywood
    - Stronger than acrylic
    - Easier to cut and work with than metal
- Aluminium “Rail”
  - Made using two aluminium extrusion profile bars (132mm length) and three aluminium sheets (one 132x66 and two 132x40)
    - To reduce coefficient of friction with can
    - To make sure can stays upright
    - Protect MDF from (water) damage
- Keypad on top surface
  - Electronics such as ESP32 to be inside the frame, and connections from keypad and servo motor to be routed via drilled holes
- Servo + Coil Dispenser (why reinvent the wheel)
  - Metal coil: 66mm pitch and external diameter, 2 revolutions, 4mm wire diameter
  - Continuously rotating 360 deg servo (SG90-HV)
    - For precise control of rotation
    - Cheaper than stepper motors
    - Easier than using dc motor drivers and gears
    - Familiarity with servo library



Cuboid Shaped Frame





Specifications:

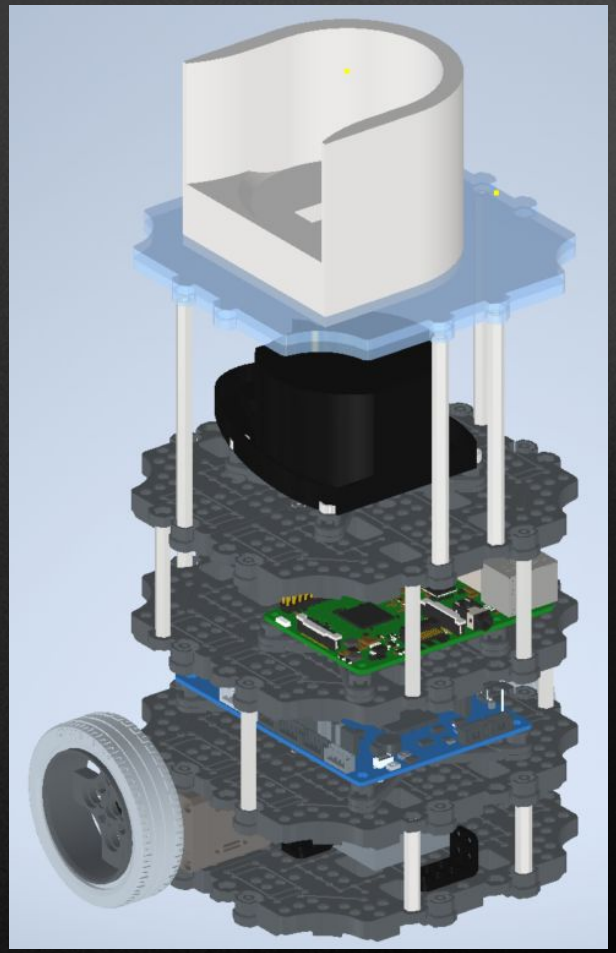
Weight: 9g

Operating Angle: 360 degree
Dimension: 23×12.5x22mm
Stall torque: 1.3kg/cm(4.8v), 1.5kg/cm(6V)
Gear type: POM gear set
Operating speed: 110RPM(4.8V);130RPM(6V)
Operating voltage: 4.8v-6V
Temperature range: 0°C_ 55°C
Control system: Digital

Calculations

Coefficient of friction of aluminium on aluminium, $\mu_s = 0.3$
Mass of aluminium can = 0.35kg
Force required to horizontally push the can = μ_s x mg
= 0.3 x 0.35 x 9.81
= 1.03005N
Assuming we want the can to be pushed at 30mm/s,
Power required = F x V
= 1.03005 x 0.03
= 0. 0309015W
From SG90 servo data sheet, operating speed = 110rpm
Torque from servo required = 30P/(π x rpm)
= (30 x 0.0309015) / (π x 110)
= 0.00268Nm (3sf)
From SG90 servo data sheet, stall torque = 1.3kgcm = 0.127Nm (3sf)

Therefore, maximum torque produced by servo is well above required torque (~50 times).



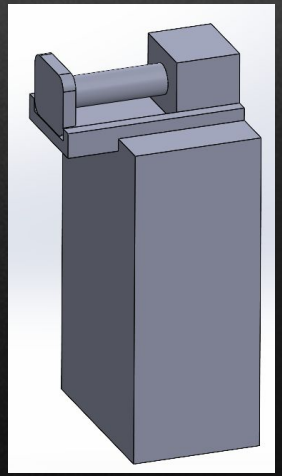
Turtlebot modifications

- Fifth waffle - building upon the design features of the turtlebot 3 burger
  - Laser cut using acrylic
  - 90mm supports (join two 45mm supports with a set screw) x4
- Cupholder
  - 3D-printed using PLA
    - Less chance of warping
    - Strong enough for use case
  - Designed to facilitate receiving the can
    - Funnel-like Tapered opening
    - Gentle slope
  - Walls on the other 3 sides to keep can from tilting or falling during motion

Rejected Concepts

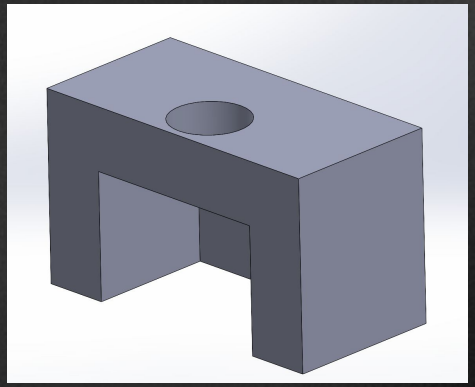
Linear Actuator

- More expensive
- More complicated
  - Motor + gears + lead screw
  - Motor driver
- Larger and longer; would require size of dispenser to be larger
- Heavier

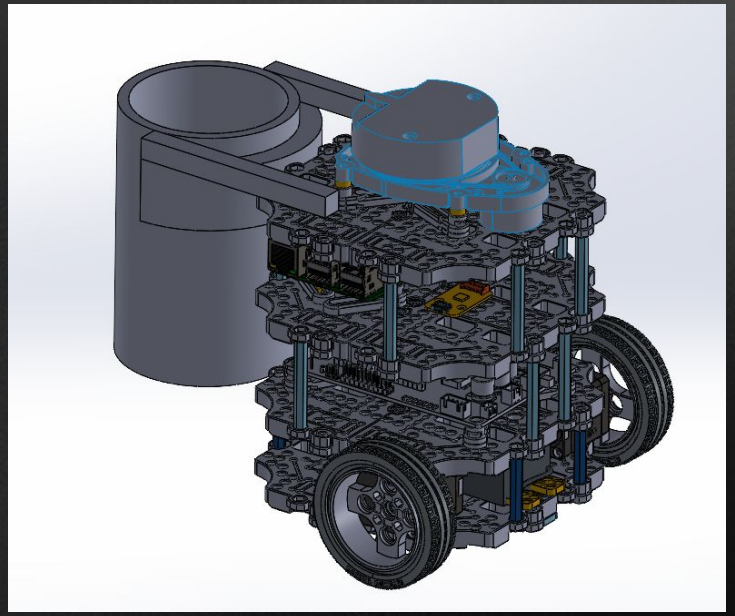


Trapdoor

- Involves dropping the soda can down and onto the bot
  - May damage or shake the can
  - May damage components on the bot, or shake the bot and affect navigational capabilities



Rejected Concept

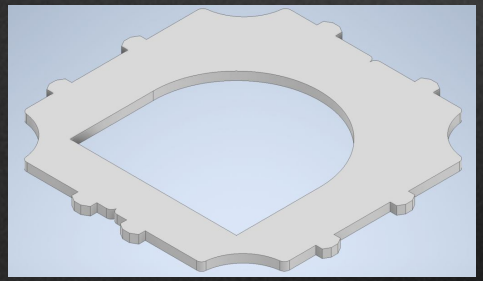


Piggyback concept

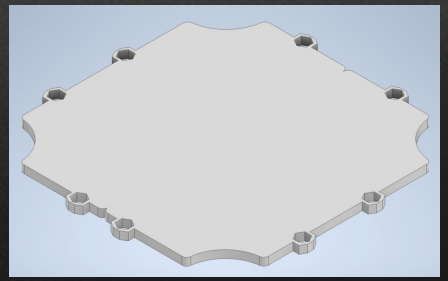
- Designed to work with trap door dispenser; would not work with chosen design
  - Hence can needs to be dropped into cup holder
    - May cause bot to tip which may then result in impairing navigation
    - May cause damage to the can, or the cupholder, or the bot, especially over prolonged use
- May cause problems with navigation using lidar as effective length of the bot has been increased - cup holder may collide with objects when the bot is turning
- Weight distribution is greatly affected which may result in instability or even tripping over
  - Cup Holder + Can (~500g) would weigh around half of the entire turtlebot
  - In comparison, the consequence of the higher CG in the fifth waffle concept is not as great/is negligible

Group Fabrication Proposal

Fifth Waffle - Laser Cut Acrylic (4mm)



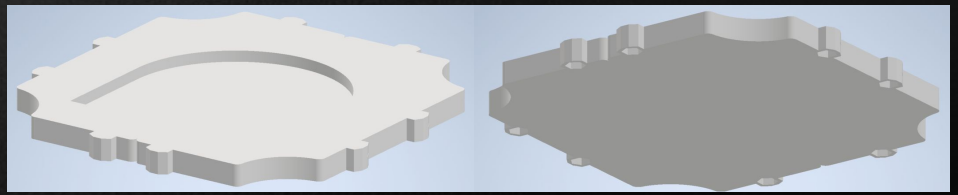
Top plate



Bottom plate

Combined using acrylic glue

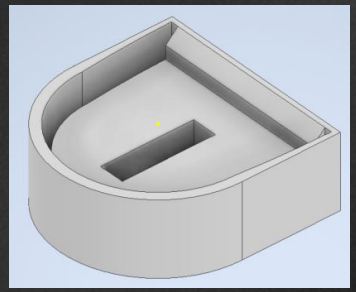
Top view



Bottom View

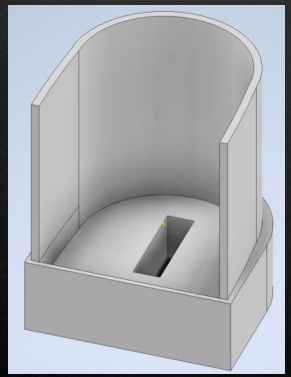
Cup Holder - 3D Printed PLA

Top view

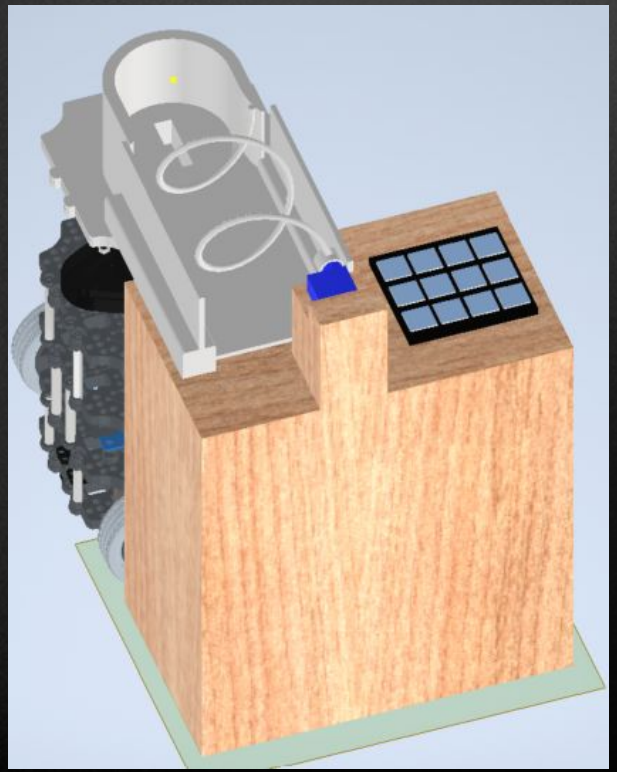


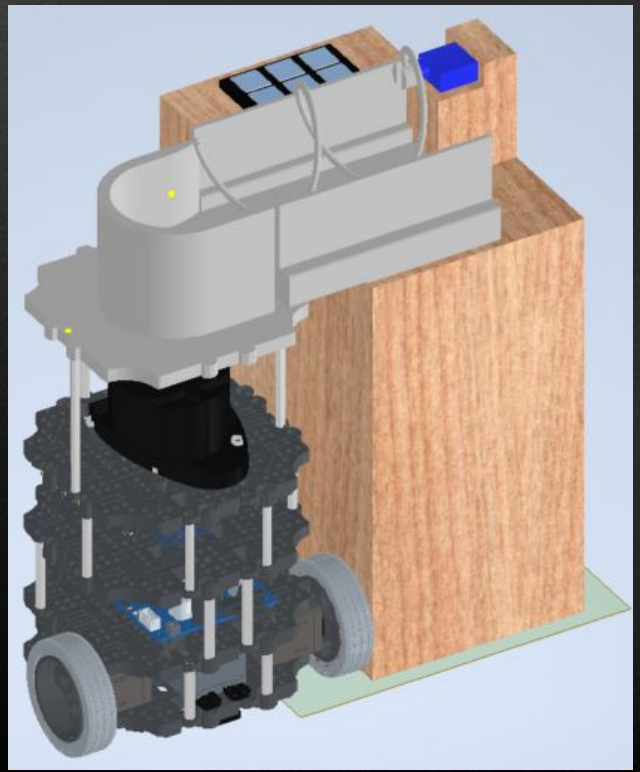
Side view

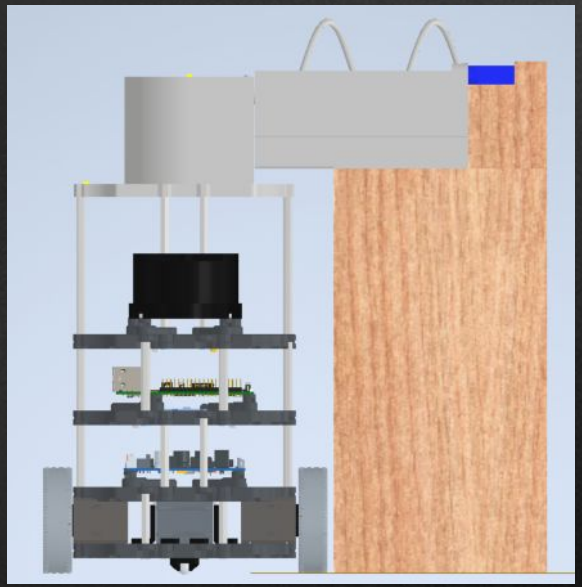
Insert metal sheet (141.74x70x3)



Full Design CAD Mockup

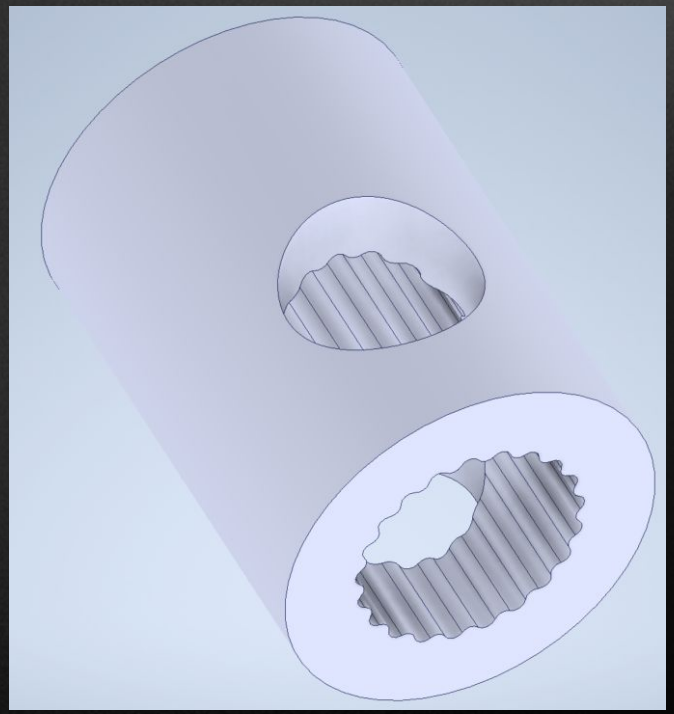






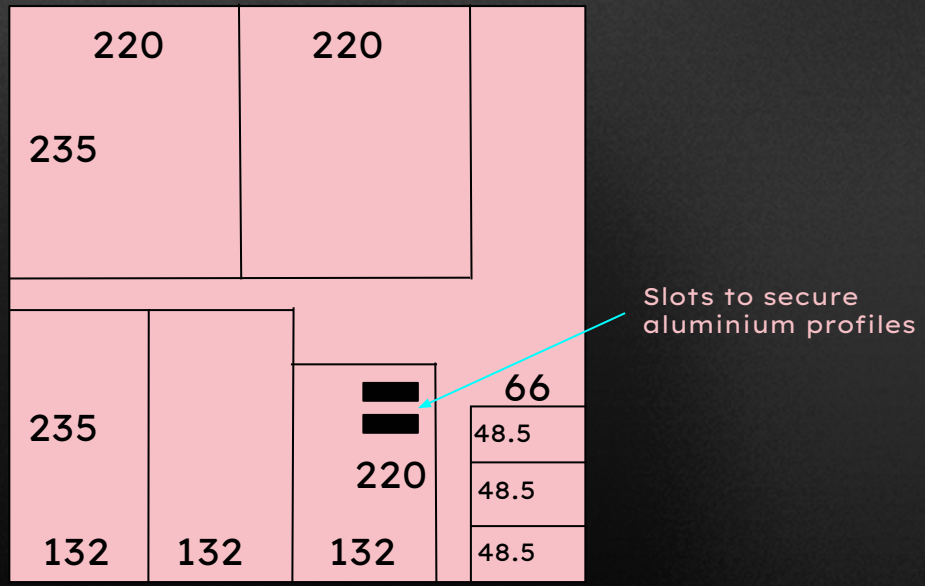
Group Fabrication Proposal

Servo Arm - 3D Printed



Group Fabrication Proposal

Dispenser Body - Buy one 500mm x 500mm x 15mm panel, cut using band saw, (PVA) glue edges and secure with wood screws



02

OPERATION

OVERVIEW

01 Dispensing

02 Navigation

03 Docking

OVERVIEW

01 Dispensing

02 Navigation

03 Docking

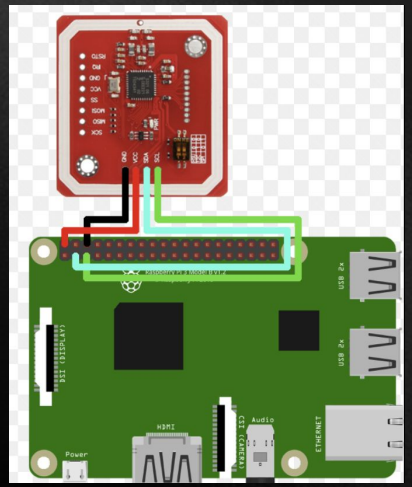
01 Dispensing

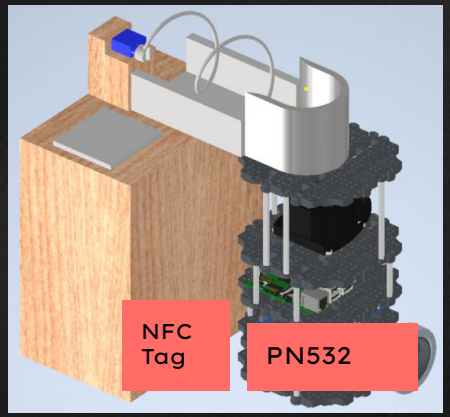
- Loading Can

01 Dispensing

- Loading Can
- NFC Interfacing

PN532 module which will interface via I2C with the raspberry pi on the bot will detect NFC tags on the dispenser, signalling that the bot is docked properly. This will be explained further in the docking phase



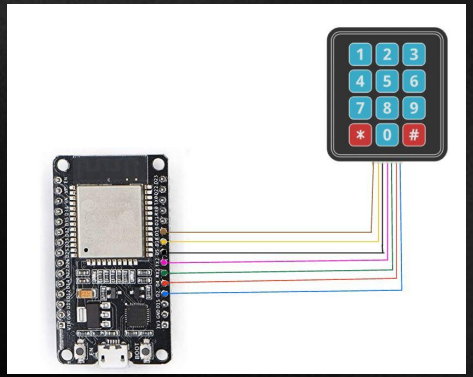


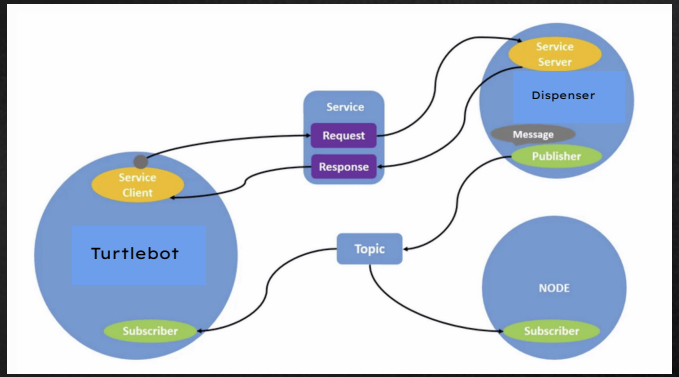
01 Dispensing

- Loading Can
- NFC Interfacing

  PN532 module on the bot will detect a NFC tag that is pasted on the dispenser, signalling that the bot is docked properly
- Transmit table number

  Table number is entered into the keypad which reaches the via the 6 GPIO pins of ESP32. ESP32 will publish the information to a topic on the ROS2 graph where the raspberry pi will be listening





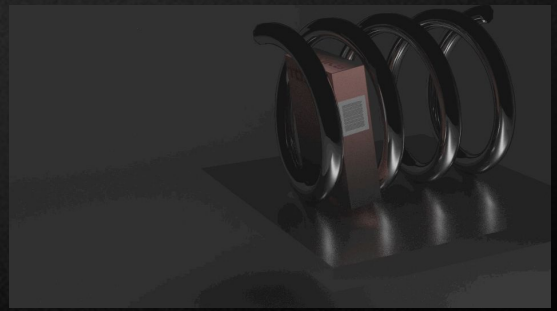
01 Dispensing

- Loading Can
- NFC Interfacing

  PN532 module on the bot will detect a NFC tag that is pasted on the dispenser, signalling that the bot is docked properly
- Transmit table number

  Table number is entered into the keypad which reaches the ESP32. ESP32 will publish the information to a topic on the ROS2 graph where the raspberry pi will be listening
- Activate spiral

  When all the conditions above are met, the servo motor will be activated, moving the can from the dispenser into the turtlebot's cup-holder



01 Dispensing

Loading Can

NFC Interfacing

PN532 module on the bot will detect a NFC tag that is pasted on the dispenser, signalling that the bot is docked properly

Transmit table number

Table number is entered into the keypad which reaches the ESP32. ESP32 will publish the information to a topic on the ROS2 graph where the raspberry pi will be listening

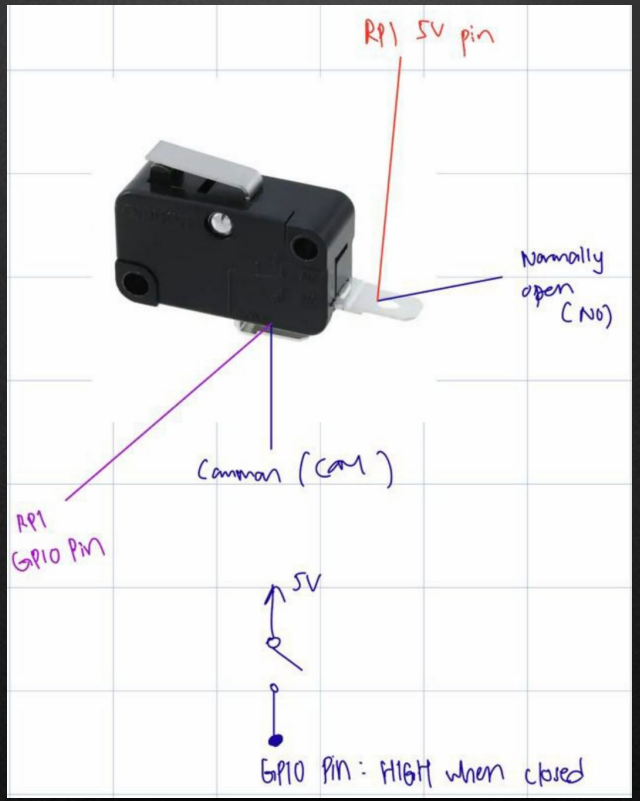
Activate spiral

When all the conditions above are met, the servo motor will be activated, moving the can from the dispenser into the turtlebot's cup-holder

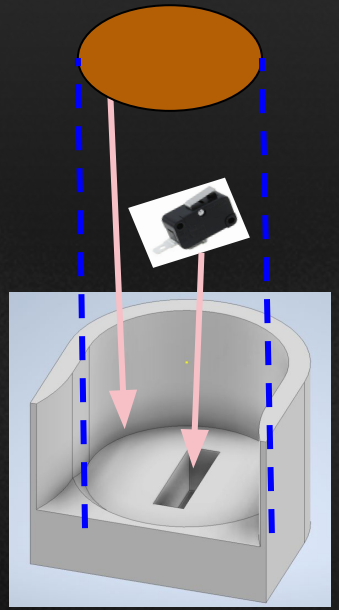
Can detection

There will be a microswitch mounted on the cup-holder, which will close the switch when there is a load above, setting the GPIO pin it is connected to, to HIGH signalling that the can is transferred successfully

Can detection



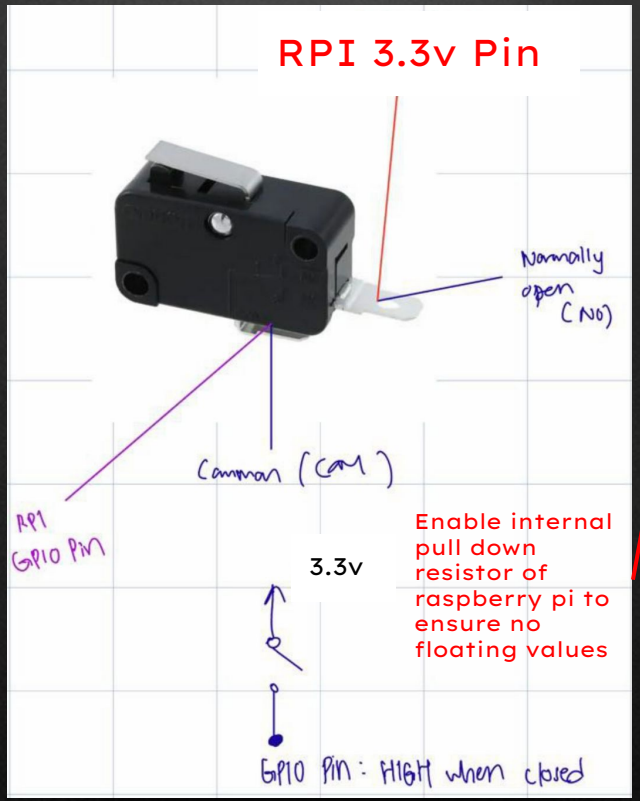
The microswitch that we have chose have a return force of 0.49N and max operating force of 3.92N. Taking F=ma, where a is the acceleration due to earth's gravity, we have derived that about 50g of mass will trigger the microswitch, setting the GPIO pin to high. There will be a flat platform above the microswitch to ensure that the switch is triggered reliably regardless of where it is on the cup holder. The mass of a flat platform such as cupboard only weighs a couple grams (<5g). The maximum mass it can take is about 400g, which is higher than the 350g can and the <5g of flat platform combined



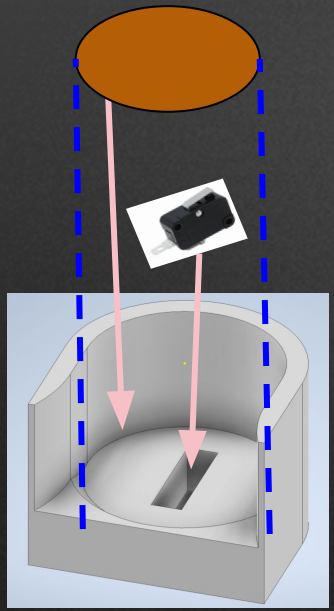
Operating characteristics / Model	V-151-1□6	V-151-1□5 V-101-1□5	V-101-1□4
OF max. RF min.	3.92N 0.49N	1.96N 0.49N	0.98N 0.15N
PT max. OT min. MD max.	1.6mm 0.8mm 0.6mm		
OP	15.2±0.5mm		

50g - 400g

Can detection




```python
# coding=utf-8

import RPi.GPIO as GPIO
import datetime

def my_callback(channel):
    if GPIO.input(channel) == GPIO.HIGH:
        print('\n▼  at ' + str(datetime.datetime.now()))
    else:
        print('\n ▲ at ' + str(datetime.datetime.now()))

try:
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(6, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
    GPIO.add_event_detect(6, GPIO.BOTH, callback=my_callback)

    message = raw_input('\nPress any key to exit.\n')

finally:
    GPIO.cleanup()

print("Goodbye!")
```



Operating characteristics / Model	V-151-1□6	V-151-1□5 V-101-1□5	V-101-1□4
OF max. RF min.	3.92N 0.49N	1.96N 0.49N	0.98N 0.15N
PT max. OT min. MD max.	1.6mm 0.8mm 0.6mm		
OP	15.2±0.5mm		

50g - 400g

Pull Down

When you have a circuit that connects 3.3v to a GPIO pin, it'll read HIGH when the circuit is closed. When it's open, it could read anything. You need a "pull down" resistor connecting your circuit to ground, so that it reads LOW when the circuit is open. *(I'll show this in effect later.)*

Pull Up

Similarly, if you have a circuit connecting your GPIO pin to ground when it's closed, it'll read `LOW`. You need a "pull up" resistor so that, when it's open, it defaults to the `HIGH` state.

In both cases, the button has no resistance (or at least, less resistance), and so when the circuit is closed it short-circuits around the pull up or pull down resistor and reads the other value. Hopefully this will make more sense with a couple demonstrations.

https://grantwinney.com/using-pullup-and-pulldown-resistors-on-the-raspberry-pi/

Component Rationale

Component	Rationale
ESP32	Wide documentation, cheap, serve our purpose and powerful
PN532	One of the most popular module around. After looking through our seniors' repo, we realise that many groups used it, therefore we could ask them about the technical implementation
V151-3A6	Simple, elegant and effective solution to detect can. Using HX711 load sensor would have been messier due to more wire connections and there would have been many unnecessary features
4x3 numpad	To have access to more keys such as # to act as confirmation of cancellation keys

OVERVIEW

01 Dispensing

02 Navigation

03 Docking

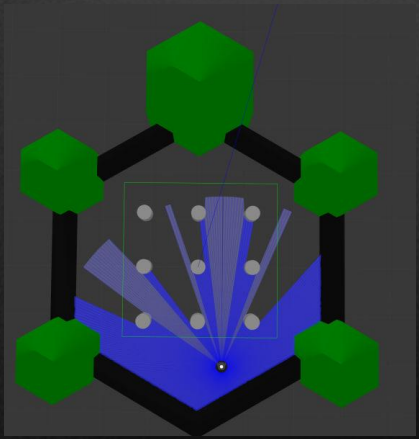
02 Navigation Overview

- Have 2 maps running simultaneously
  - preloaded map for movement and orientation
  - active map for confirmation and reliability
- Localisation capabilities
- Use coordinates for calculations

02 Navigation

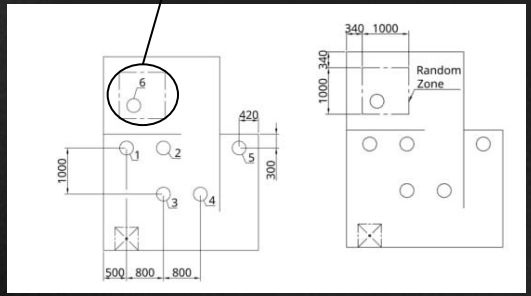
- Replicate test environment & save map

Recreate the test environment in simulation and in the real world and save the map. Create a launch file to load the saved map.

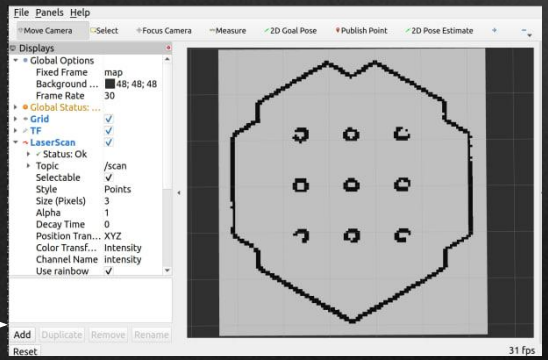


Simulation

Table 6 will be left out of the mapping due to its unknown position



Actual map



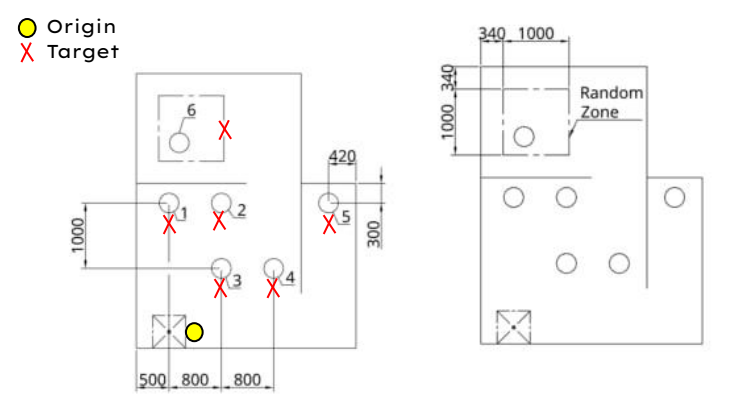
Navigate the robot around the map to fully map out map, save and load when needed

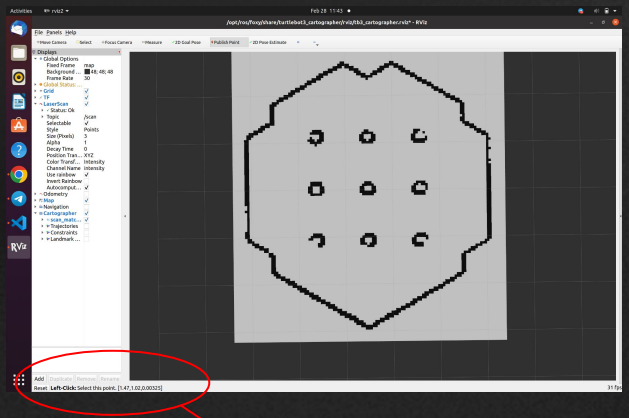
* fix the origin of where the bot starts

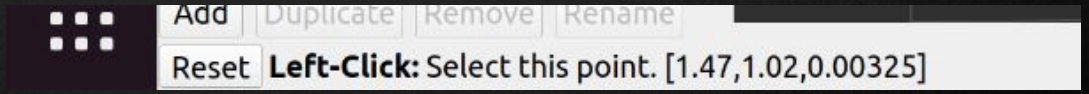
02 Navigation

- Saving coordinates of 6 tables

  Plot and save coordinates of the 6 possible destination





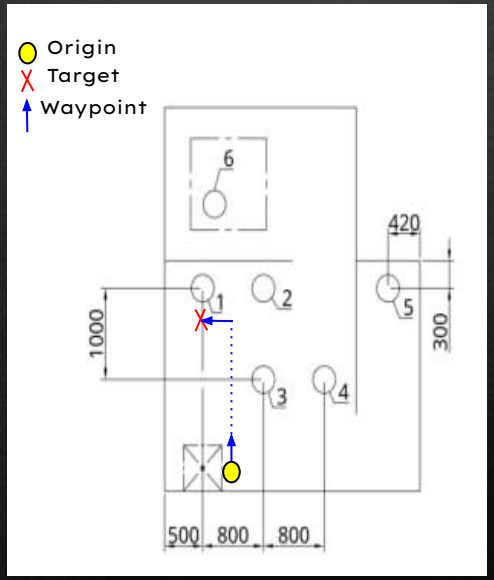


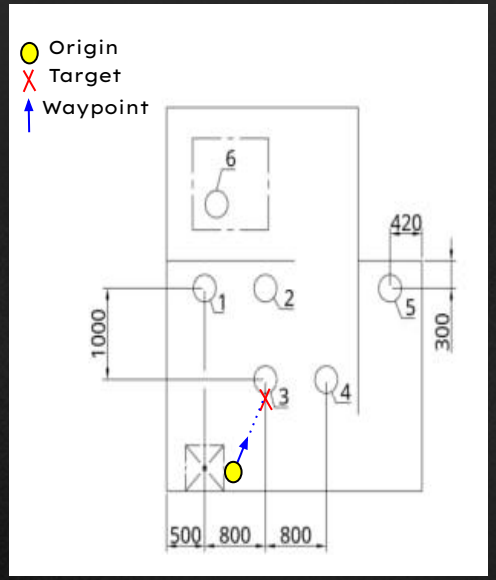
* fix the origin of where the bot starts

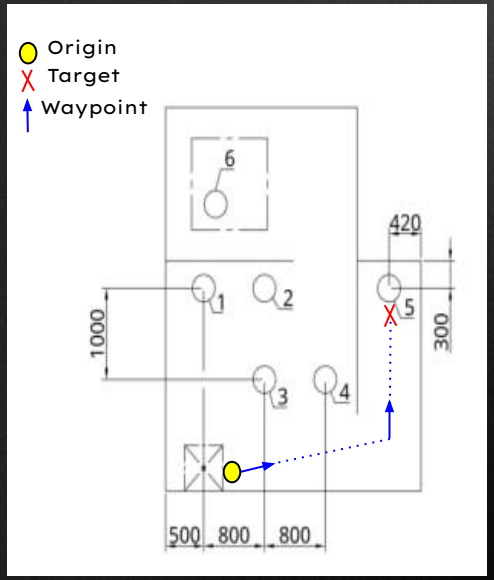
02 Navigation

- Creating waypoints for tables 1-5

  Plot and save coordinates of waypoints to each table.



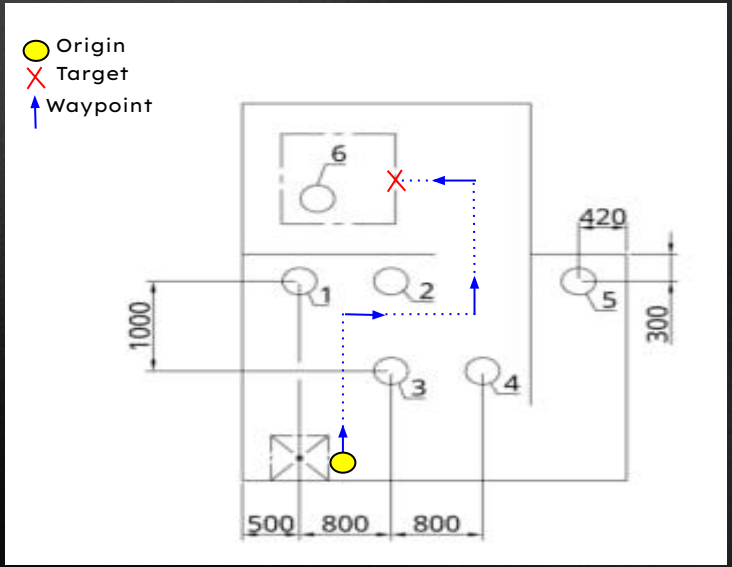


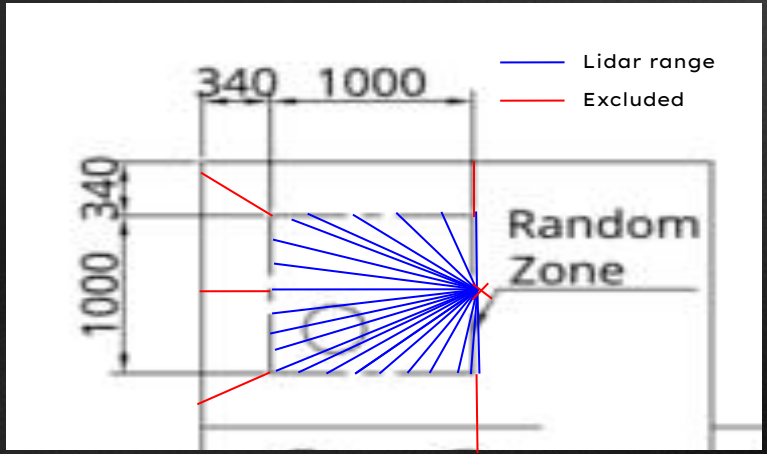


02 Navigation

- Creating waypoints for tables 6

  Special case as location of table is unknown

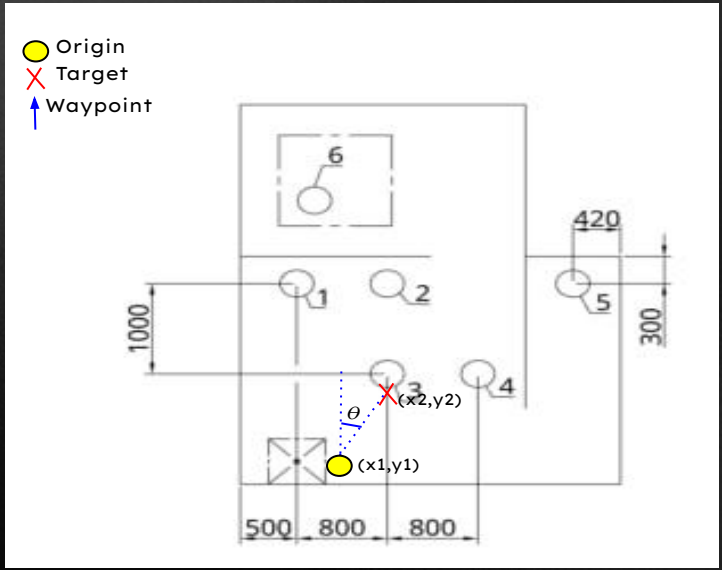




02 Navigation

- Calculation of waypoints

The angle and distance between 2 points will be calculated and saved



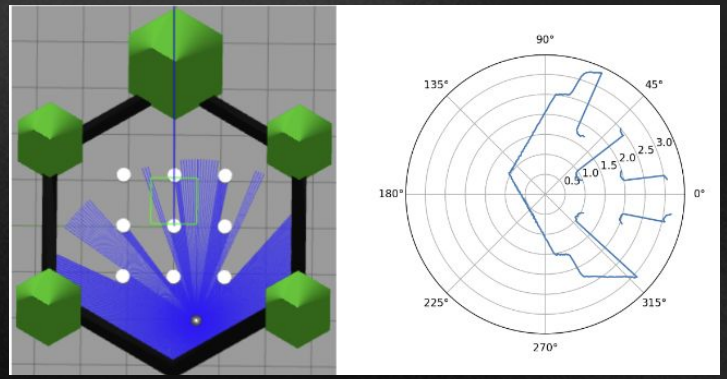
$$\text{Distance} = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

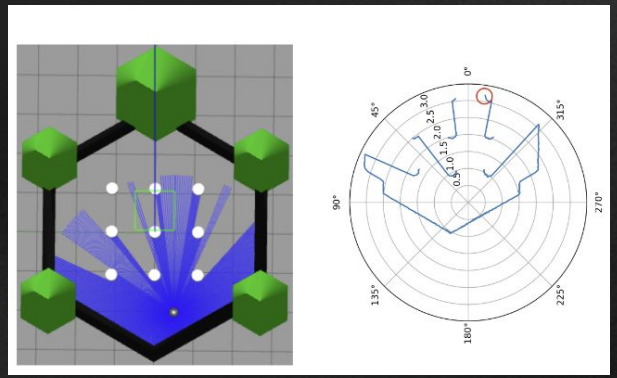
$$\text{Angle} = \tan^{-1}((X_2 - X_1)/(Y_2 - Y_1))$$

02 Navigation

- Localisation of robot

  Subscribe to the topic that ros2 **map2base** repo publishes to, which provides the coordinates of the robot in reference to the map frame. Ensures that loaded map and active scanning are on the same frame.





02 Navigation

Autonomous Navigation Flow

1) Load saved map into RVIZ
2) Boot up turtlebot to start dynamic mapping, ensuring that the turtlebot is at the origin that is defined when mapping the saved map
3) Table number will be sent to turtlebot
4) Turtlebot will map table number to the set of waypoints required to navigate to that table
5) Each waypoint will be an object containing distance to move and angle to turn
6) Twist object will be published to turn the bot and move the bot
7) Pre loaded map helps navigate towards the table, active scanning ensures no collision and to find the table reliably in case of map drifting
8) Detect table, wait for can to be lifted off the bot (poll the pin connected to microswitch)

OVERVIEW

01 Dispensing

02 Navigation

03 Docking

03 Docking

Plan the return journey

After the can is removed from the container, the microswitch is off. The robot will find the path back to the dispenser.

Docking process

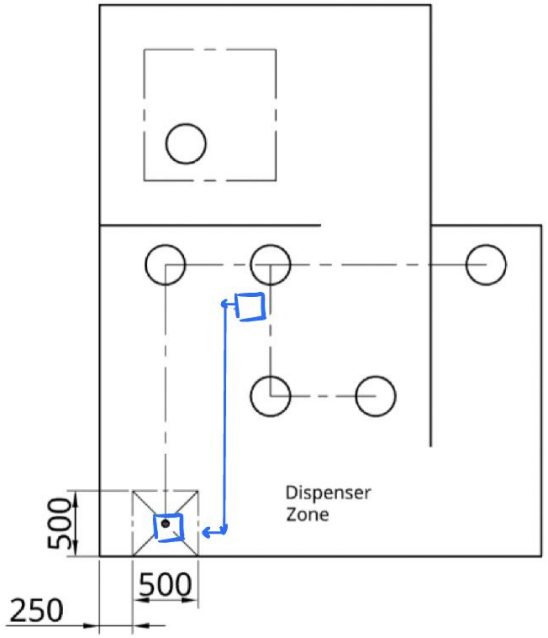
Align to the dispenser and ready for reloading

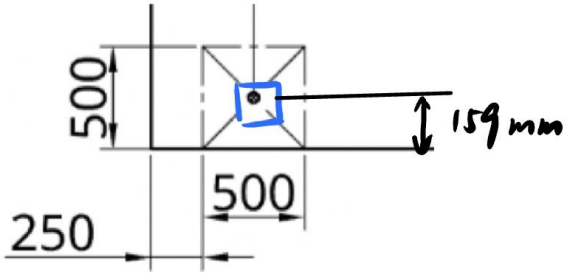
03 Docking

- Plan the return journey
- Docking process
  1. Find the origin coordinate, calculate the path
  2. Move the robot along the wall into the dispenser area
  3. Adjust its orientation
- Align to the dispenser and ready for reloading

Docking logic

1. **Find the origin coordinate, calculate the path:**
   The robot will obtain its present location from the loaded map and determine the various directional distances it needs to travel in order to return to its starting point, which is the dispenser.
2. **Move the robot into the dispenser area:**
   During the navigation process, the robot will continuously monitor its current coordinates and verify that it is following the correct path.
3. **Stop beside the dispenser and adjust its position:**
   After the robot arrives at the origin, it can utilize an NFC reader on board to verify its position by detecting an NFC tag on the floor. Additionally, the robot will use Lidar to gauge the distance to the wall(159mm), enabling it to modify its position.





Docking logic

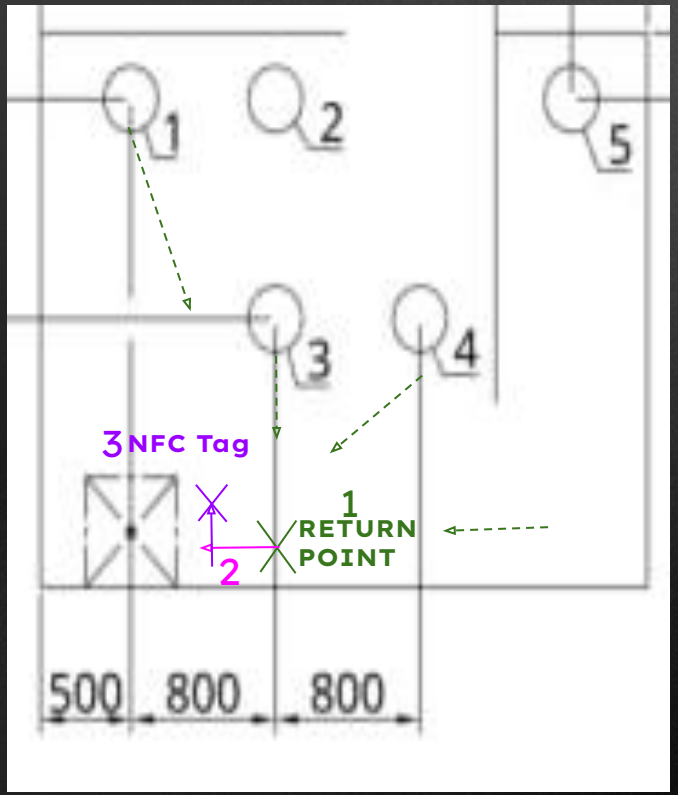
1. **Calculate and move to RETURN POINT**

From wherever it was at, move to the return point, marked with the green cross

2. **Move towards dispenser with wall followers algorithm**

Constantly check it's distance to the wall on the left and adjust such that it is walking straight

3. **Rotate right on the spot slowly, when it detects the NFC tag that is placed at the purple cross, it will stop, signalling that it is in the right position**

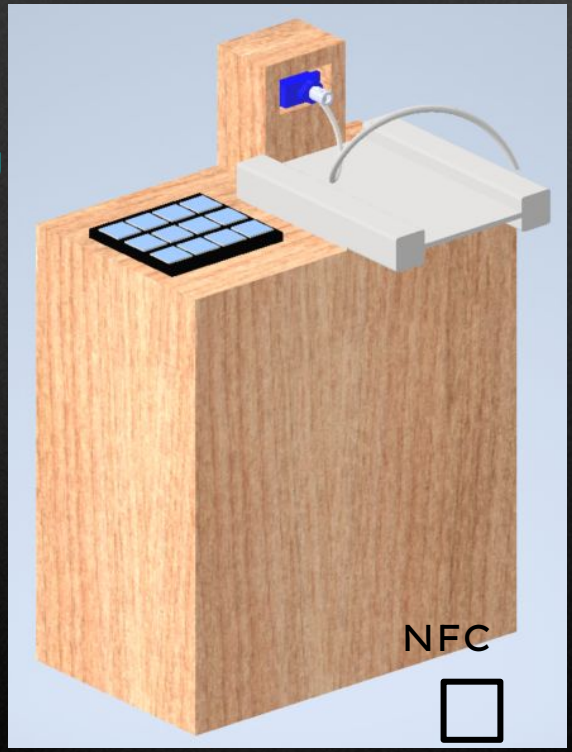


03 Docking

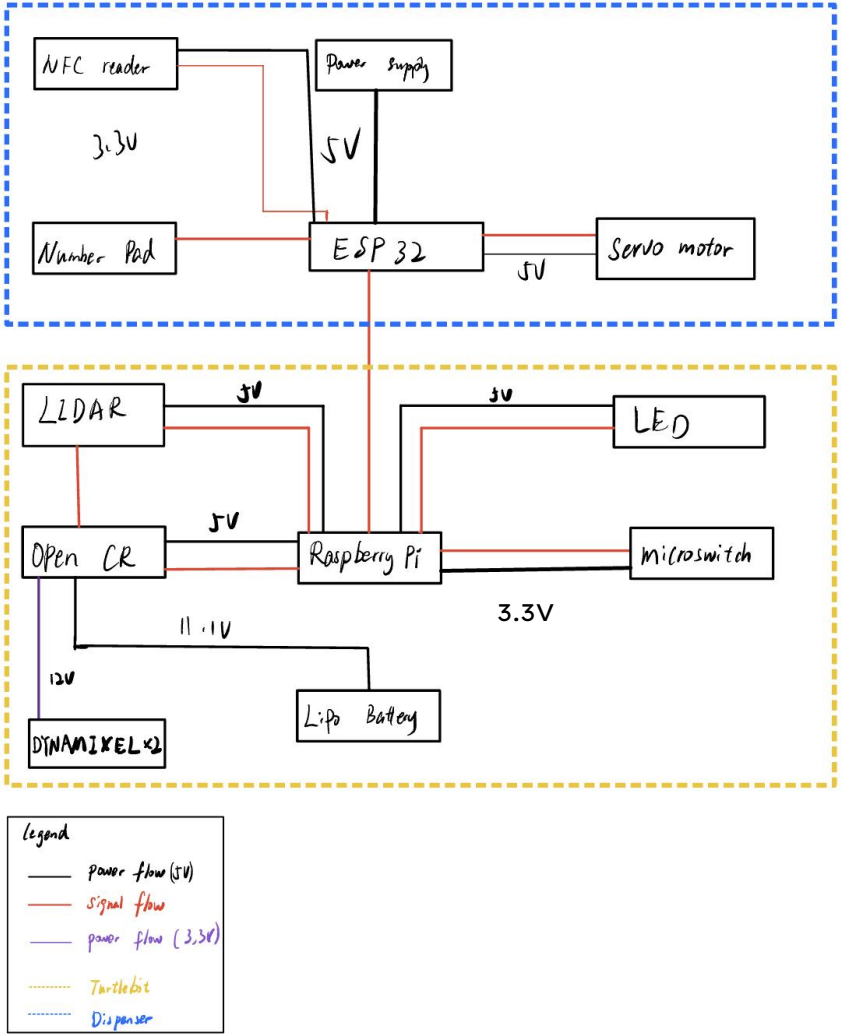
- Plan the return journey
- Docking process
- Align to the dispenser and ready for reloading

NFC tag:

To ensure the robot is oriented correctly, a second NFC tag will be positioned on the floor to provide guidance. As the robot **slowly** rotates, it will immediately stop movement once it detects the second NFC tag, ensuring proper alignment with the dispenser and readiness for reloading.



Electrical Architecture



Power Budget Table

Power Budget Table

For the dispenser:

Components	Voltage	Current	Quantity	Power
ESP32	Typical Voltage: 3.3V	0.5A	1	1.65W
Membrane 3*4 Matrix Keypad	3.3V	less than 10mA	1	0.033W
SG90 servo Motor	5.0V	270mA	1	1.35W
Power plug	5.0V	2.5A	1	12.5W

Power Budget Table

For the turtlebot:

Components		Voltage	Current	Quantity	Power
Turtlebot	Raspberry Pi	Standby: 12.105V Boot up: 12.105V Operation: 12.106V	Standby: 0.520A Boot up: 0.450A Operation: 0.637A	1	Standby: 6.52W Boot up: 5.44W Operation: 7.70W
	Lidar				
	OpenCR				
	Dyamixel				
NFC reader(PN532)		3.3V	100-200mA (in active state)	1	0.33-0.66W
microswitch		3.3V	Negligible	1	A few milliwatts

Testing

Testing

	Test Case	Description	Purpose
Mechanical	Push can with spiral	Put the can into the spiral and trigger the servo motor	Ensure that the can can fit into the spiral and there is enough power to move it forward
	Docking the bot on the dispenser	Use NFC tags to guide the bot into the correct orientation	Ensure precise docking
Electrical	Reading NFC tag	Place an NFC tag near the PN532 reader and check if there is data transmitted	Ensure the module is connected properly and working
	Key into number pad	Key in number into number pad that is connected to esp32 on the dispenser	Make sure the number pad is connected properly and proper conversion is done when the button is pressed
	Check microswitch minimum and maximum load	Set up the microswitch with the platform. Put something about 50g on the switch to see if it triggers and 400g to see if it still works.	Find out what is the minimum weight to trigger the microswitch and whether exceeding the theoretical max weight will have any adverse effects

Testing

	Test Case	Description	Purpose
Software	Load saved map and boot turtlebot at the same time	Load the map, boot up turtlebot to get lidar to map its surrounding	Ensure that 2 maps are being used together at the same time
	Publish twist object	Publish twist object to the pi in the right format and values	Make sure that the values we publish corresponds to the correct angular/linear velocity
	ESP32 interfacing with raspberry pi	Send information from ESP32 to raspberry pi	Ensure that table information can be transmitted properly

Bill of Materials

Bill Of Materials

No.	Item	Purpose	Cost + Link
1	2x Perfboard (5x7 cm)	Solder electronic components	$2.14 (link)
2	Vending machine spiral	To dispense can	$3 (link)
3	Continuously rotating 360 degree servo (SG-90HV variants)	To operate the Vending Machine Spiral	$6.60 (link)

Bill Of Materials

No.	Item	Purpose	Cost + Link
4	Numpad	To take input of desired table number from the user.	$9.45 (link)
5	LED	For feedback	Self-source
6	V-151-3A6 Microswitch - Solder terminal - 0.49N (50g) - 3.92N (400g) - datasheet:https://omronfs.omron.com/en_US/ecb/products/pdf/en-v.pdf	To detect presence of can in the bot's cup holder	$2.08 (link)
7	ESP32	For digital communication between the Dispenser and the Bot	Self-source link

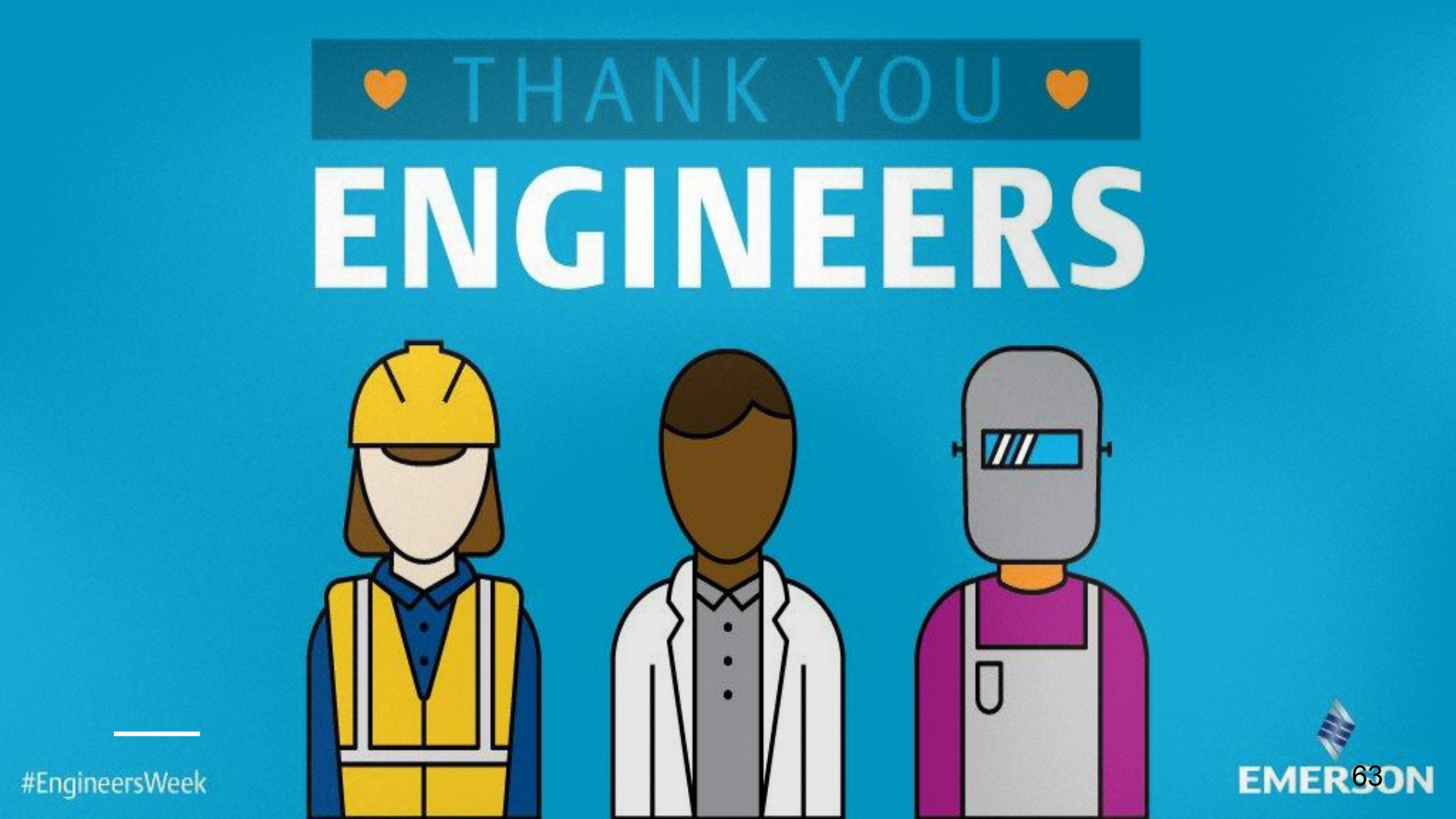
Bill Of Materials

No.	Item	Purpose	Cost + Link
8	Fifth waffle - laser cut acrylic Total area needed: 276mm x 276mm 4mm thickness	For placing the cup holder on the bot	Self-source/ Ask Miss Annie
9	Cup holder - 3D printed with PLA	Storing the can in the bot	$5/h, 4h, $20
10	PN532 + Tags	For precise docking of the bot near the dispenser.	Lab / Self source
11	GP Alkaline 9V battery	To power servo motor on the dispenser and esp32	$3.50 link

Bill Of Materials

No.	Item	Purpose	Cost + Link
12	MDF panel, 500mm x 500mm x 15mm	For the dispenser body	$18 link
13	PVA Glue/Korean Glue	To stick the panels together	Self-source
14	Aluminium sheet x3 One 132mmx66mm Two 132mmx40mm	For the rails on the dispenser	Self-source / Ask Miss Annie
15	Aluminium profiles x2 132mm length	For the rail on the dispenser	Self-source

Grand Total = $70.44


THANK YOU
ENGINEERS
#EngineersWeek

Q & A