1313131313131313131313131313
1313131313131313131313131313

We are Group 13.

Concept of Operations

Concept Design

Ball Launching Mechanism - Extendable Arm

- Allows payload to roll slowly into the bucket
- Arm is foldable
  - Increases reach
  - Minimal effects to robot's centre of gravity
- Simple ball carrier design
  - Minimal moving parts to reduce chances of failure

Electrical System - LiDAR, 6-DOF IMU, Motor Encoders

- Provides precise location data
- Generates accurate SLAM map
- Ease of implementation

Electrical System - Camera

- Accurate colour detection

Electrical System - Servo

- Cheap and easy to implement

Process of setup

- Turn on phone hotspot
- Load ping pong balls
- Turn on OpenCR switch, position robot
- SSH in from laptop
- Run rosbu
- Run rslam
- Run launch file, let it rip

Operational scenario

- Maze that yall set up

User interactions

- Turning robot on
- Loading balls
- Initial positioning of robot
- Taking back robot after we finish

System Requirements

System Specifications

- Raspi 4 - Quad Core BCM2711, 1.5GHz
- OpenCR 1.0 - ARM Cortex-M7 STM32F7
- LDS-02 LiDAR - 2.3kHz sampling rate, 160 ~ 8,000 mm detection distance, 5Hz scan frequency
- Pi Camera V1 - 5MP, OmniVision OV5647
- 2x Dynamixel - 57 rpm no-load speed, 1.4Nm stall torque
- 2x SG90 Servo - 0.12s / 60°, 1.7kg-cm stall torque

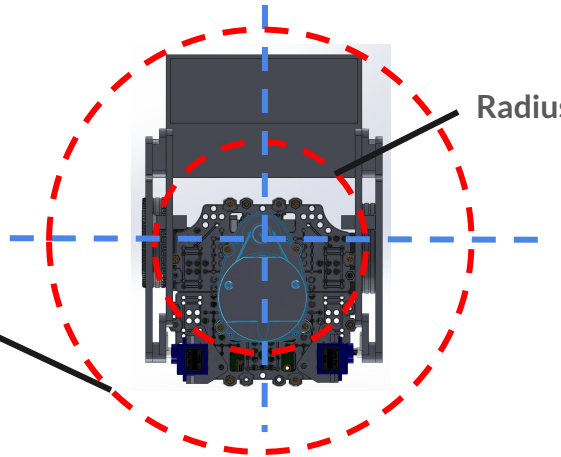
Turtlebot Specifications

Items	Burger
Maximum translational velocity	0.22 m/s
Maximum rotational velocity	2.84 rad/s (162.72 deg/s)
Maximum payload	15kg
Size (L x W x H)	138mm x 178mm x 192mm
Weight (+ SBC + Battery + Sensors)	1kg
Threshold of climbing	10 mm or lower
Expected operating time	2h 30m
Expected charging time	2h 30m
SBC (Single Board Computers)	Raspberry Pi
MCU	32-bit ARM Cortex®-M7 with FPU (216 MHz, 462 DMIPS)
Remote Controller	-
Actuator	XL430-W250
LDS(Laser Distance Sensor)	360 Laser Distance Sensor LDS-01 or LDS-02
Camera	-
IMU	Gyroscope 3 Axis Accelerometer 3 Axis
Power connectors	3.3V / 800mA 5V / 4A 12V / 1A
Expansion pins	GPIO 18 pins Arduino 32 pin
Peripheral	UART x3, CAN x1, SPI x1, I2C x1, ADC x5, 5pin OLLO x4
DYNAMIXEL ports	RS485 x 3, TTL x 3
Audio	Several programmable beep sequences
Programmable LEDs	User LED x 4
Status LEDs	Board status LED x 1 Arduino LED x 1 Power LED x 1
Buttons and Switches	Push buttons x 2, Reset button x 1, Dip switch x 2
Battery	Lithium polymer 11.1V 1800mAh / 19.98Wh 5C
PC connection	USB
Firmware upgrade	via USB / via JTAG
Power adapter (SMPS)	Input : 100-240V, AC 50/60Hz, 1.5A @max Output : 12V DC, 5A

Functional Requirements

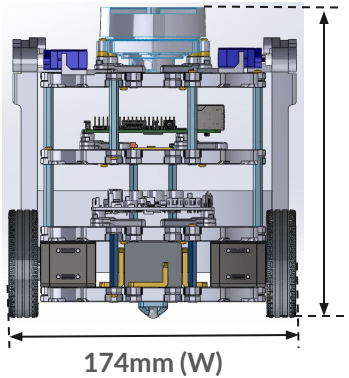
Mission Requirements
Mapping: The robot must map out the maze elements and develop a full SLAM map of the area.
Navigation: The robot must navigate through the maze in order to reach the “elevator”.
Localisation (Coordinates): The robot must locate the door based on coordinates given.
Communication: The robot must unlock a door by making a HTTP call to a web server.
Localisation (Object): The robot must locate the red bucket in the room.
Launcher: The robot must launch five ping pong balls (payload) into the red bucket

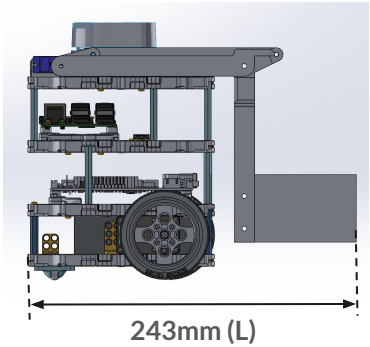
Robot dimension (Turtlebot3 Burger + attachments)



= 243 x 174 x 190
(L x W x H , mm)

= 1.2kg





Software Algorithm Design

```
if (!is_maze_solved) {
    solve_maze();
}
```

Where ROS runs

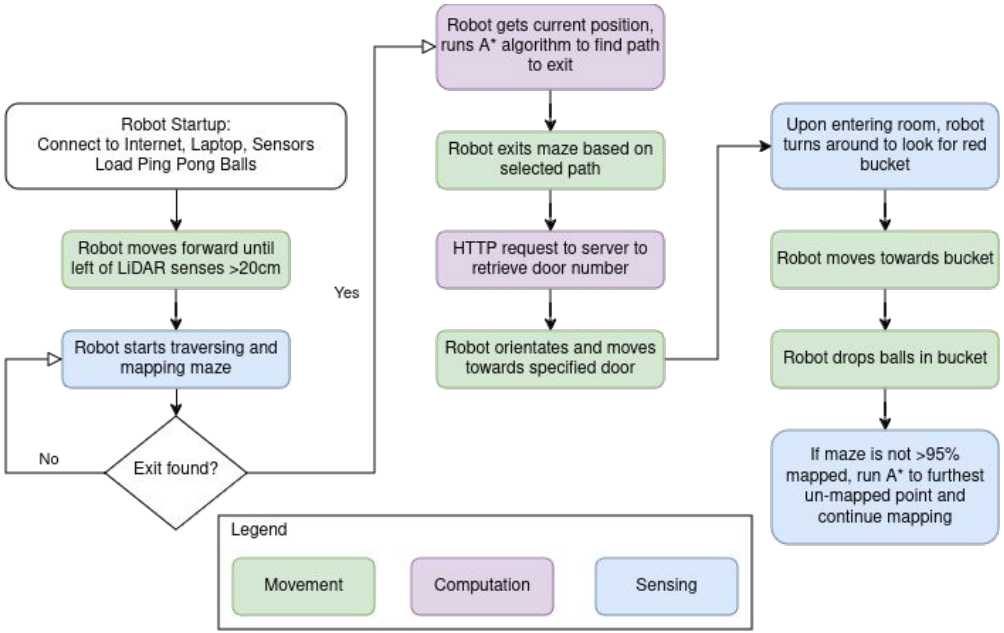
On the RPi:

- rosbu
- ROS node subscribing to servo topic, to actuate servos (will edit rosbu launch file to include this)

On the laptop:

- rslam
- Maze manipulation & path finding
- Colour detection
- Mission planning, wayfinding

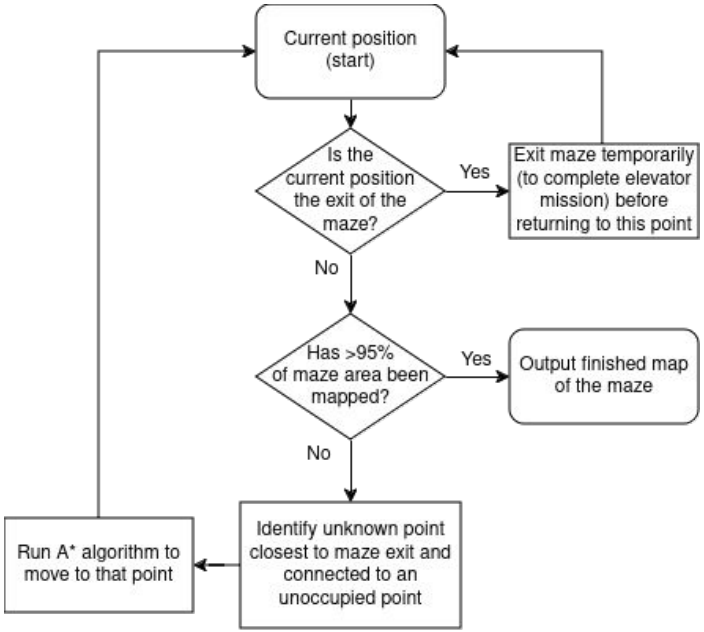
Overall Mission Plan



Priority List for Maze-Solving

- In total, we have 3 approaches to solve the maze:
    1. LiDAR SLAM map based approach
    2. Camera coloured marker based approach
    3. Birds-eye image based approach
- We will develop the camera and LiDAR programs simultaneously (between 3 people), but place more focus on the LiDAR approach.
- If we are very stuck with the LiDAR approach even after ~wk 10, we will fully commit to camera based approach.
- If we magically finish the LiDAR approach early (~wk 11), we will attempt the image based approach.

Maze Loop Flowchart



SLAM Algorithm Choice

\+ Already implemented, don't need to waste time testing new methods

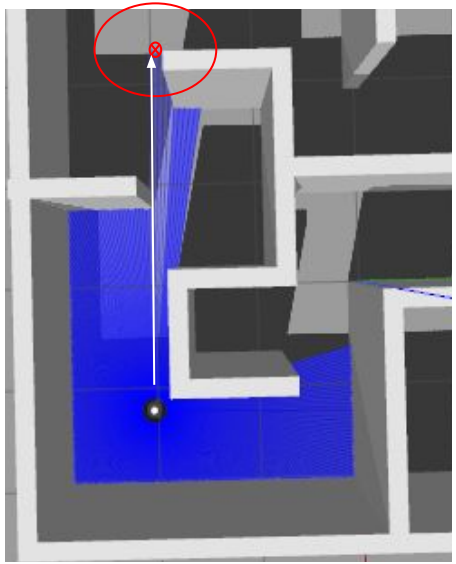
Gmapping is one of the most robust algorithms. It succeeded in building a representable map using the Hokuyo data set. From this we can conclude that Gmapping is better in handling noisy data and short laser measurements in difficult environments.

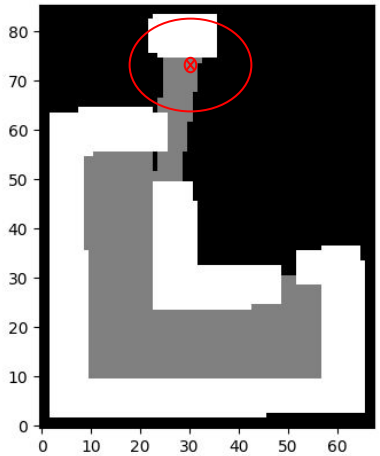
Cartographer is the only graph-based approach used in this evaluation. From the visual inspection, it can be concluded that its a robust algorithm. Similar to Gmapping, Cartographer succeeded in generating a representable map using the Hokuyo dataset. This algorithm is the most certain of the position of objects and walls. It generates the least possible corners which means it can deal with noisy data better than other algorithms. Cartographer is one of the most promising algorithms considered for this project.

The overall conclusion is that Cartographer is the most promising. Gmapping and HectorSLAM are also considered for further use in this project. TinySLAM generates the least promising maps. Therefore this algorithm will still be used and further evaluated but will not be prioritized. We omit DP-SLAM for the moment due to its segmentation fault.

Mapping & Maze-Solving Algorithm

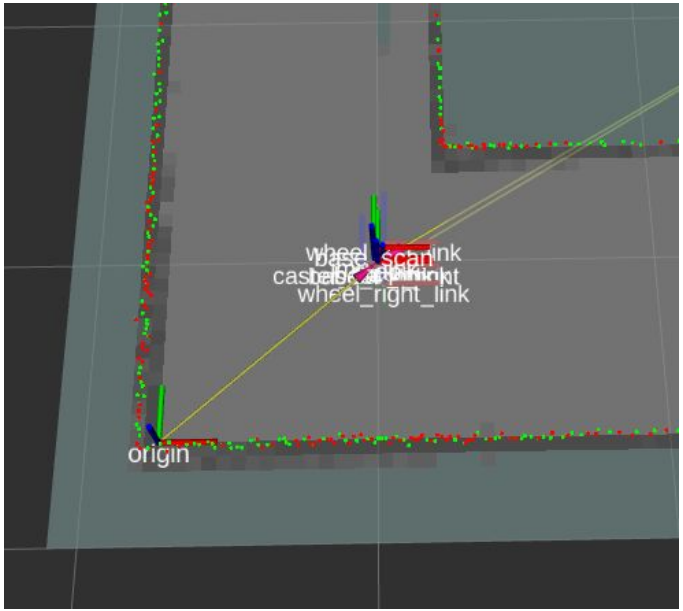
1. Define Origin
2. Identify closest ‘unknown’ region to maze exit (y = 2.1m) that is adjacent to an ‘unoccupied’ region
3. Path finding, move the robot to that point
4. Repeat steps 2 to 4
5. If there is an unoccupied point with y > 2.1m, an exit is detected





Origin Definition

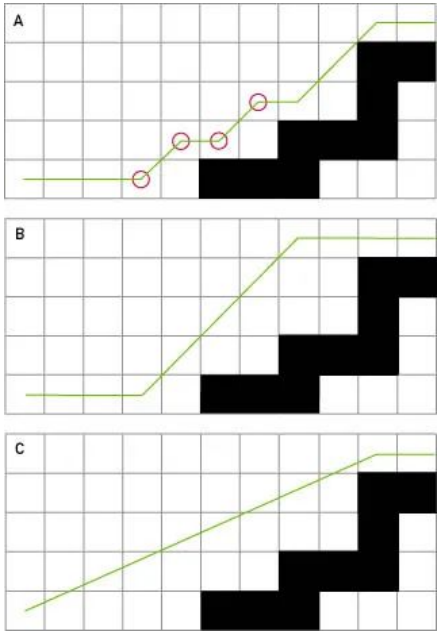
- The lowest x and y coordinate value (from the perspective of the robot) that corresponds to an unoccupied space will be identified as the origin
- This will enable us to track the position of the robot relative to the origin throughout the entire maze
- Also allows us to determine when the exit of the maze is reached (when its position relative to origin is +2.1m)



Path Optimisation

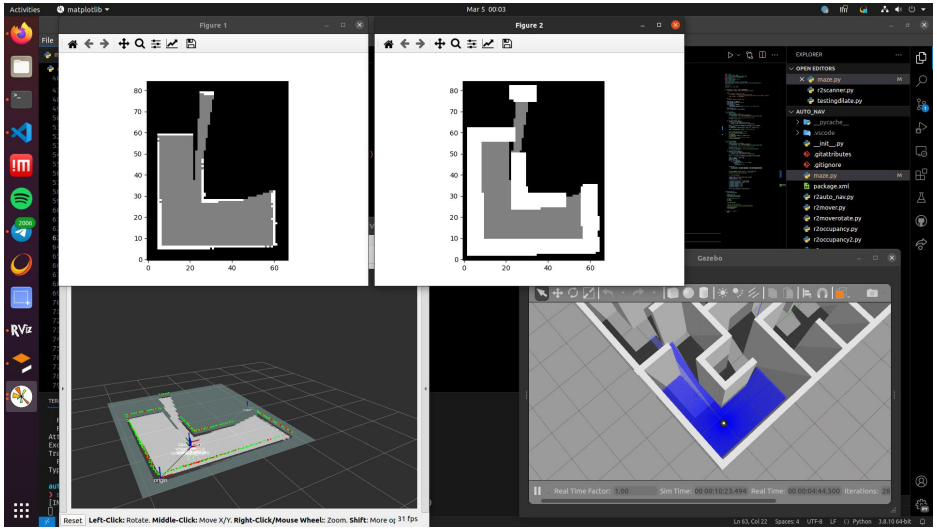
One simple method of reducing the number of turns is to make the following modification to the A* algorithm: Add a cost penalty each time a turn is taken. This will favor paths which are the same distance, but take fewer turns, as shown in Figure 2b. Unfortunately, this simplistic solution is not very effective, because all turns are still at 45-degree angles, which causes the movement to continue to look rather unrealistic. In addition, the 45-degree-angle turns often cause paths to be much longer than they have to be. Finally, this solution may add significantly to the time required to perform the A* algorithm.

The actual desired path is that shown in Figure 2c, which takes the most direct route, regardless of the angle. In order to achieve this effect, we introduce a simple smoothing algorithm which takes place after the standard A* algorithm has completed its path. The algorithm makes use of a function Walkable(pointA, pointB), which samples points along a line from point *A* to point *B* at a certain granularity (typically we use one-fifth of a tile width), checking at each point whether the unit overlaps any neighboring blocked tile. (Using the width of the unit, it checks the four points in a diamond pattern around the unit's center.) The function returns true if it encounters no blocked tiles and false otherwise. See Figure 3 for an illustration, and Listing 1 for pseudocode.

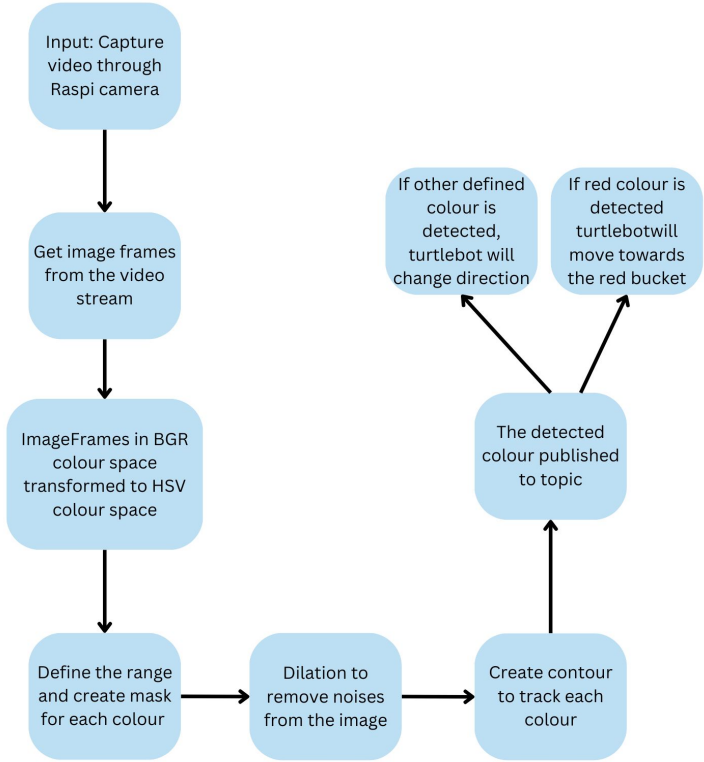


Wall Avoidance Algorithm

- To ensure that A* does not return a path that is too close to the wall (since the robot has a width), we will dilate the walls by the circumscribed radius of the robot.
- These will thus be “no-go zones” for the robot, and when robot follows the path planned it would not go into these zones as well
- Also eliminates problem of holes between walls, since they will be dilated and filled in



Colour Detection via Camera



Colour Marker Detection Flowchart

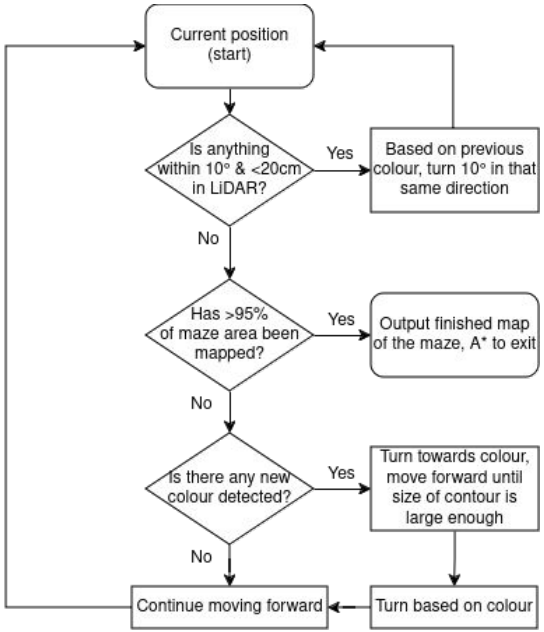
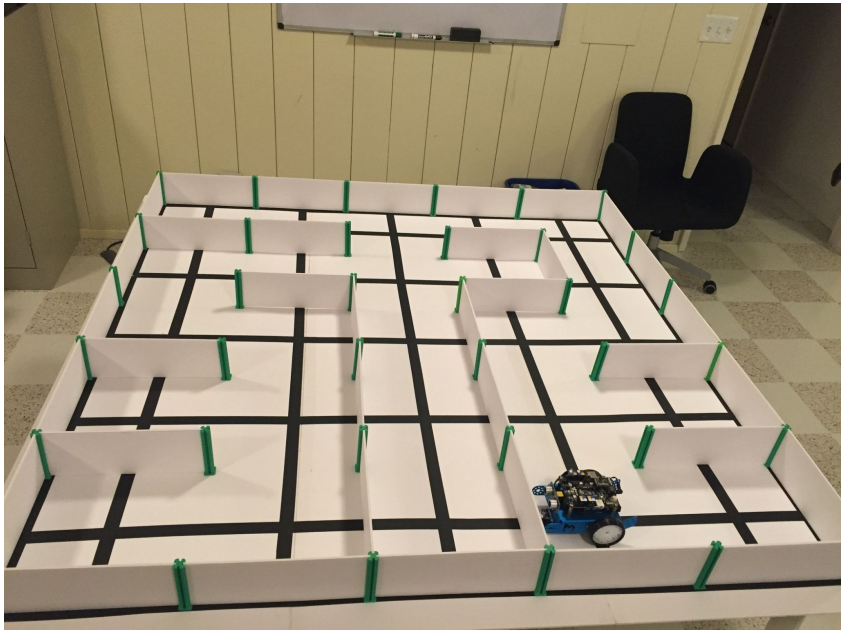
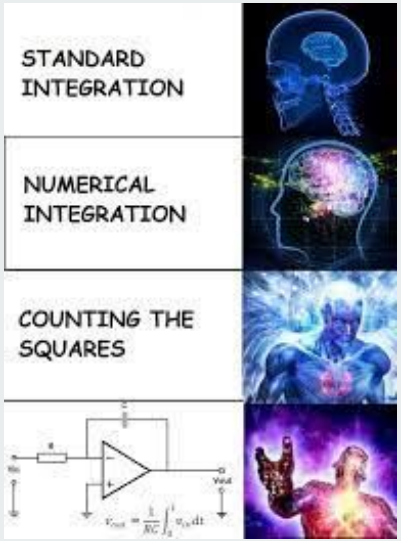


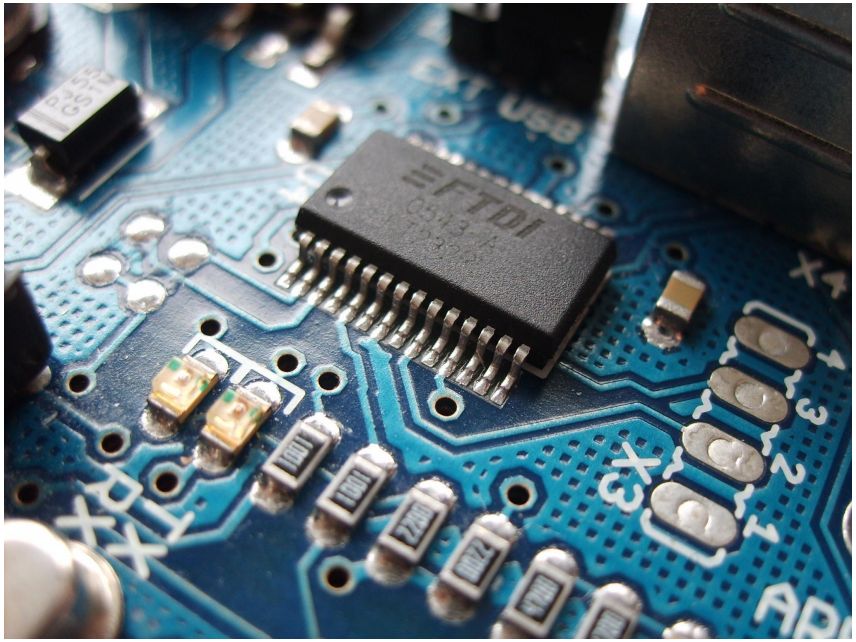
Image-based Mapping

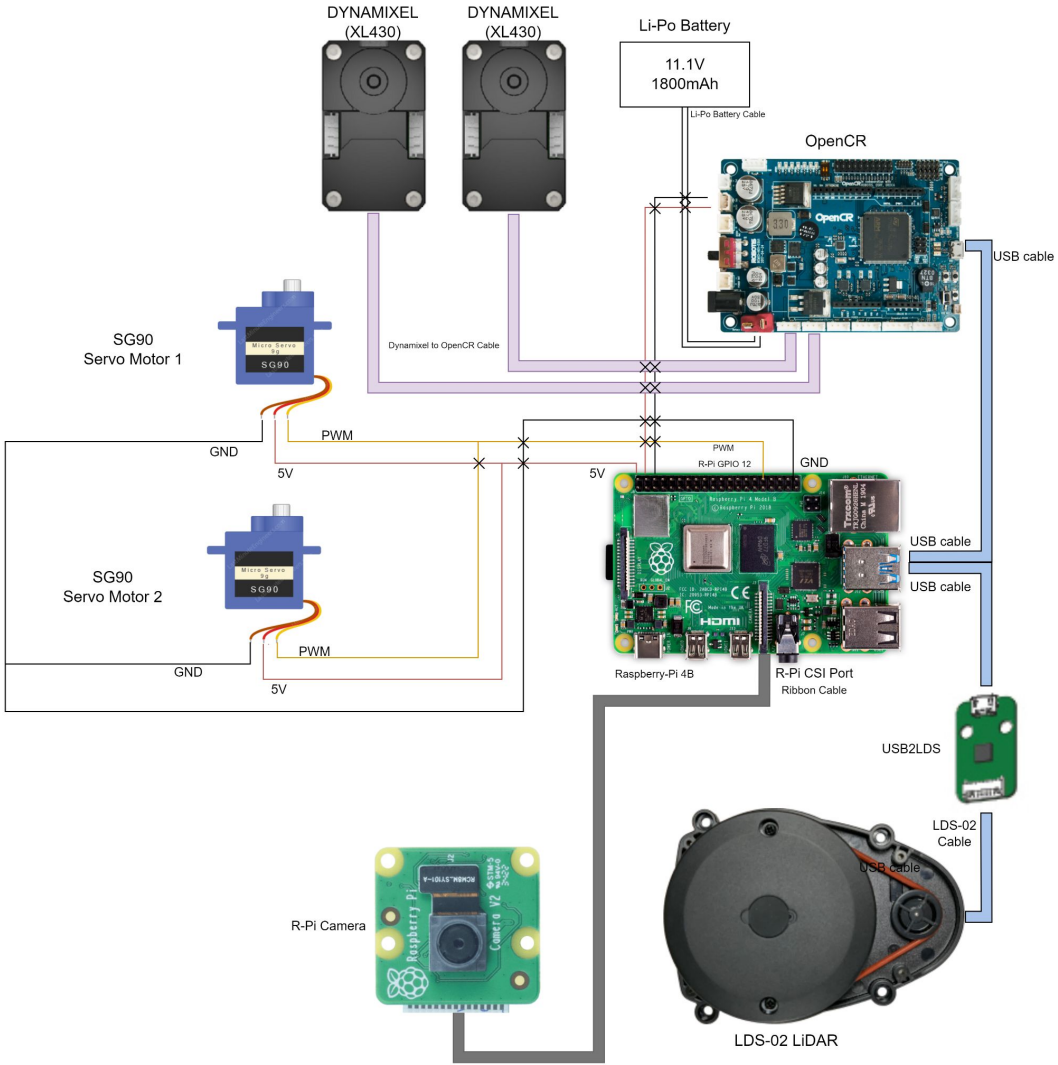
- We will tape the top of the walls with coloured tape (for e.g. red tape)
- When we take an image from above, we can threshold that image for red tape to obtain the wall lines
- Image will first be rectified
- Robot will use this to path plan straight to the exit
- Occupancy grid map will be published, then the SLAM will build on top of the captured map to get a more accurate resultant map



Electrical and Electronics Design

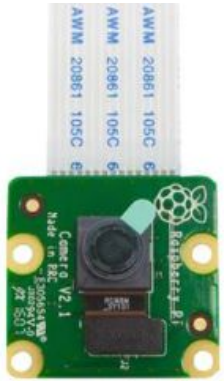





DYNAMIXEL
(XL430)
DYNAMIXEL
(XL430)
Li-Po Battery
11.1V
1800mAh
Li-Po Battery Cable
OpenCR
USB cable
SG90
Servo Motor 1
Dynamixel to OpenCR Cable
GND
PWM
5V
PWM
R-Pi GPIO 12
GND
5V
SG90
Servo Motor 2
USB cable
USB cable
GND
PWM
5V
Raspberry-Pi 4B
R-Pi CSI Port
Ribbon Cable
USB2LDS
LDS-02
Cable
USB cable
R-Pi Camera
LDS-02 LiDAR

Powered Devices

- Actuators
  - 2 x SG90 Servos
    - Voltage: 5V
    - Peak Current: 180mA
    - Power: 0.9W
  - 2 x Dynamixel (XL430)
- LiDAR (LDS-02)
- R-Pi Camera



Power Consumption

	Voltage / V	Current / A	Quantity	Power / W
SG90 Servo (Peak)	5	0.180	2	1.80
Turtlebot3 Burger	11.1	0.771	1	8.56
Total				10.36

*The values shown are measurements taken during standard operation (LiDAR, rosbu, motors running at full speed)

*Power consumption of LiDAR and Dynamixel motors are taken into account in the standard operating power consumption of the Turtlebot

*Power consumption of the R-Pi camera is considered to be negligible

Battery Life

Li-Po Battery

- 11.1 V
- 1800mAh
- 19.98Wh

Battery Life (excluding activation of Servos)

- 19.98 / 8.56 = 2.33 hours

Taking into account the activation of the actuators (servos), as well as the aging and deteriorating of the Li-Po battery,

Estimated battery life ~ 1 hour 45 minutes

Peak Current Output

Output Power Sources	*12V max 4.5A(SMW250-02) *5V max 4A(5267-02A), 3.3V@800mA(20010WS-02)

Product	Recommended PSU current capacity	Maximum total USB peripheral current draw	Typical bare-board active current consumption
Raspberry Pi 4 Model B	3.0A	1.2A	600mA

Servo Performance

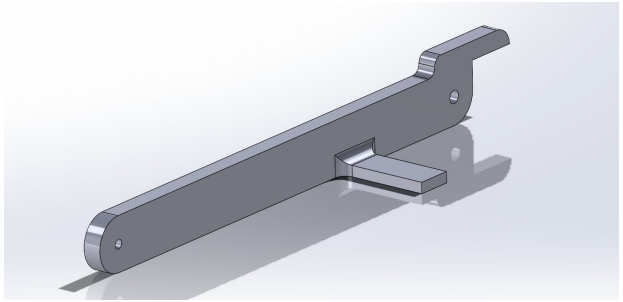
Current Draw (stationary):	3.57 milliamps [mA] @ 5.00 **VDC**
Current Draw (moving):	up to 200.0 milliamps [mA] @ 5.00 **VDC**

0.3A (Servo) + 0.3A (Servo) + 3A (RPi) = 3.6A < 4A

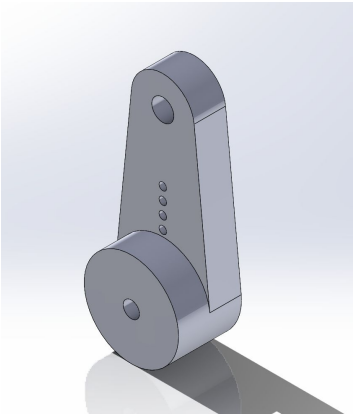
Mechanical CAD

SOLIDWORKS

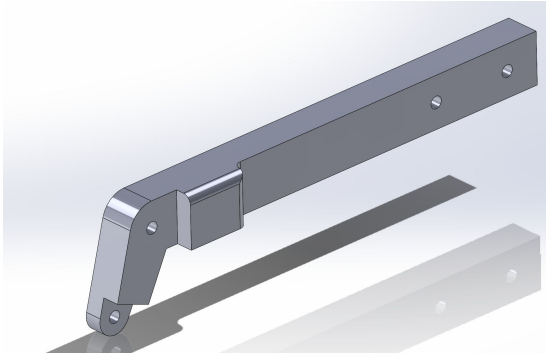
Arm design (4 parts, 2 sets)



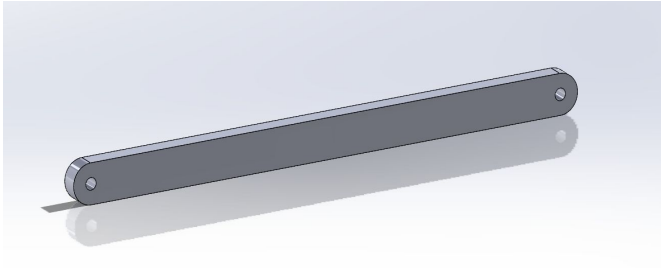
Support arm



Servo adaptor

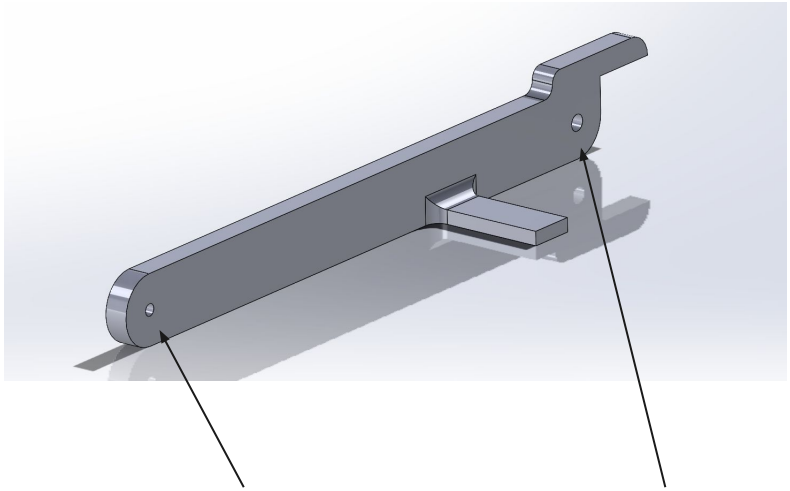


Carrier arm



Linking arm

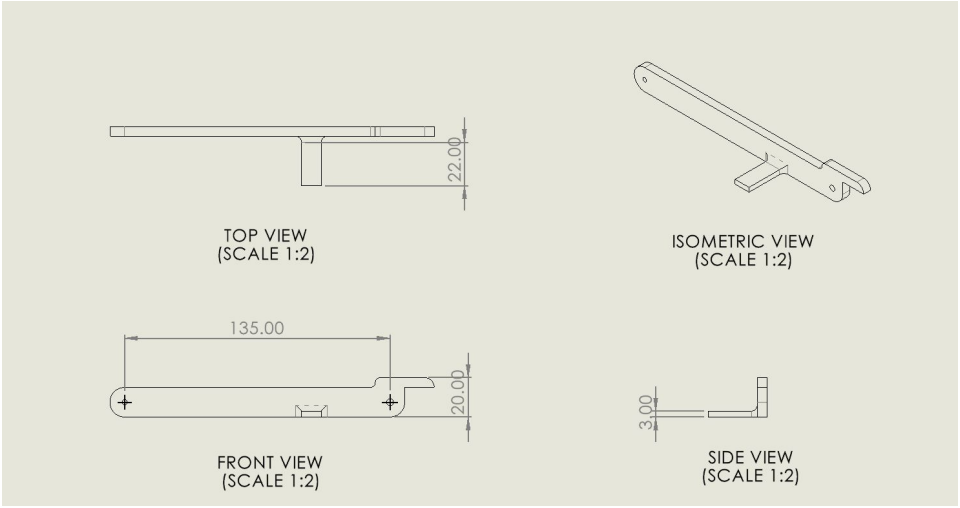
Support arm



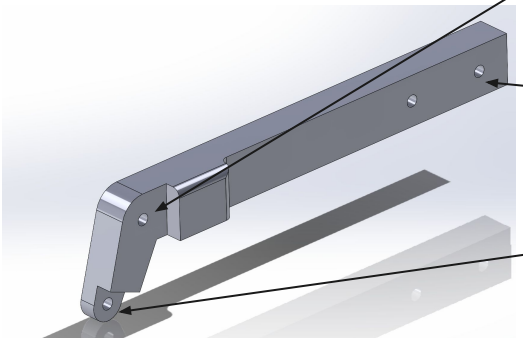
2.6mm hole for
(M2 Crosshead screw)
Clearance fit

3.5mm hole for
(M3 Crosshead screw)
Clearance fit

Engineering Drawing



Carrier arm

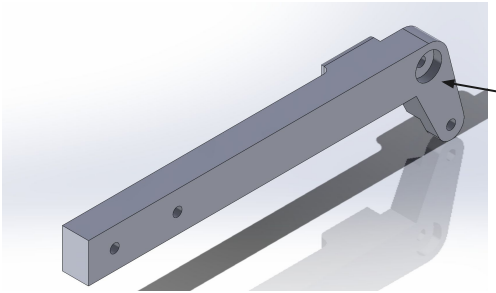


3.5mm hole for
(M3 Crosshead screw)
Clearance fit

3.5mm hole for
(M3x20 Crosshead screw)
Clearance fit

3mm hole for
(M2.5 Crosshead screw)
Clearance fit

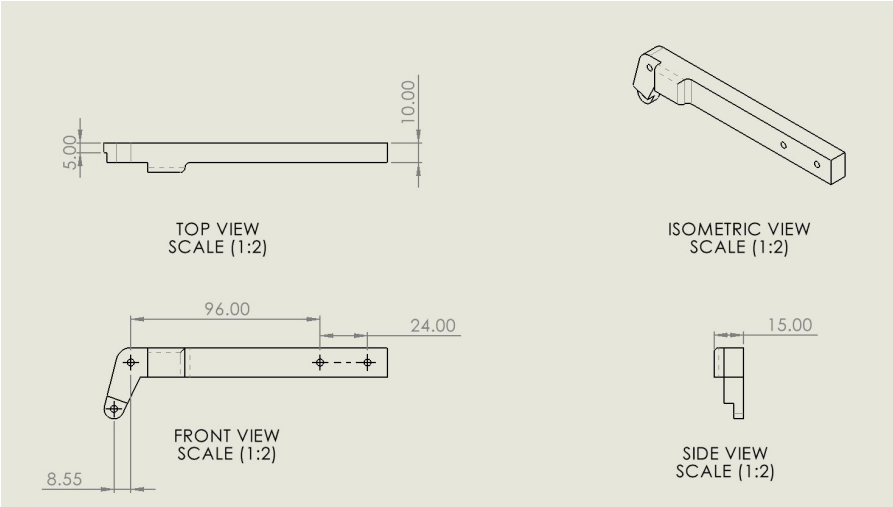
FRONT VIEW



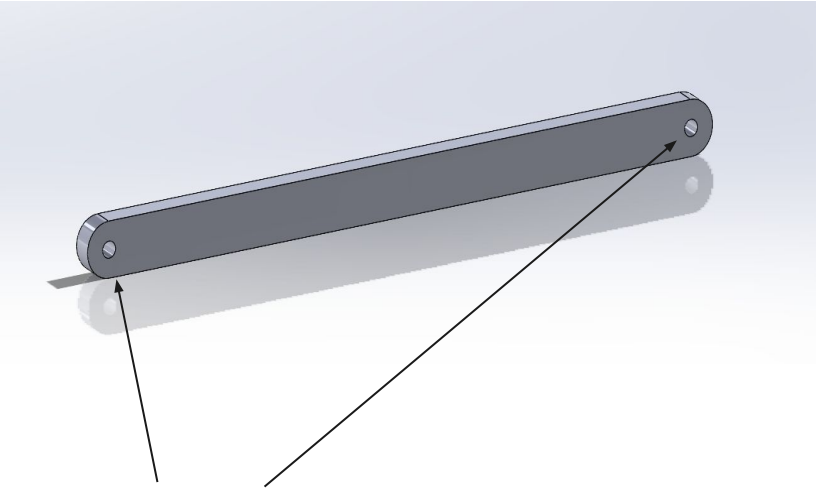
10mm Diameter opening
(Allow screw head to be flushed to the side of the part)

BACK VIEW

Engineering Drawing

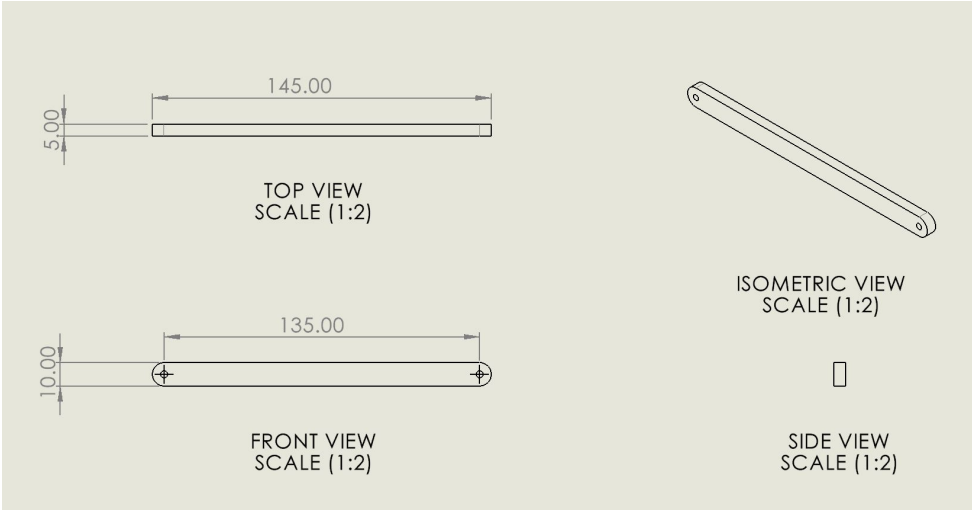


Linking arm

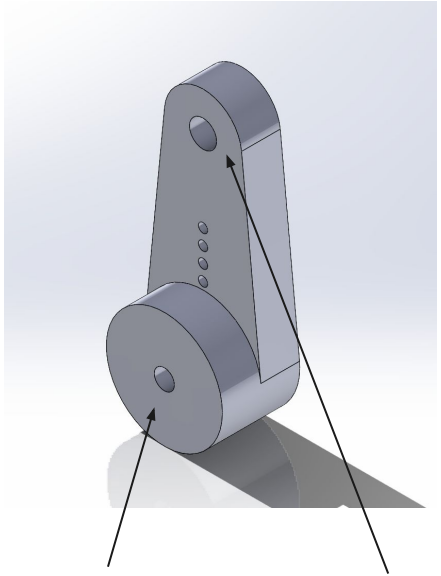


3mm hole for
(M2.5 Crosshead
screw)
Clearance fit

Engineering Drawing



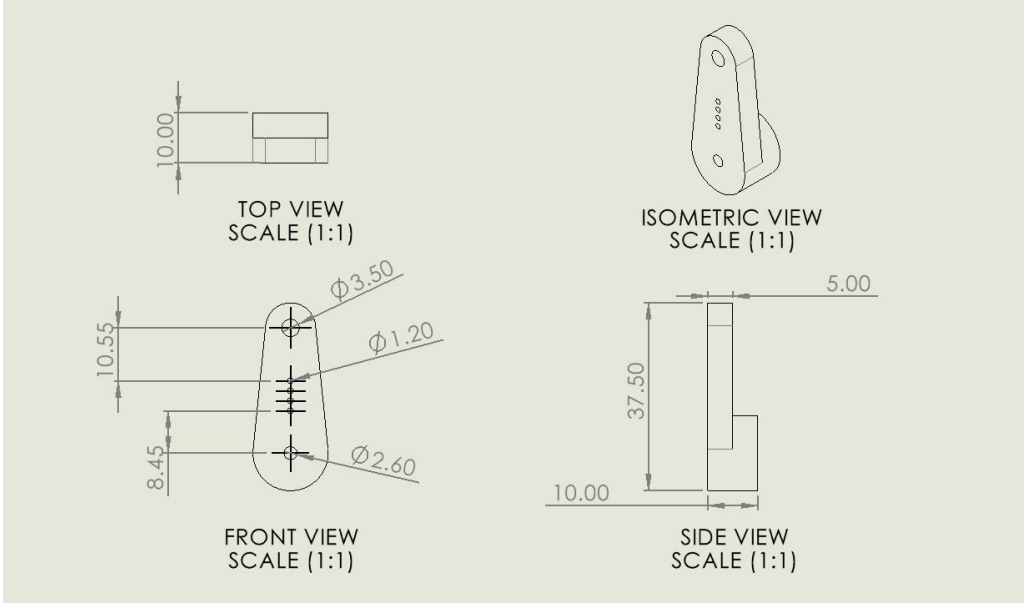
Servo adaptor



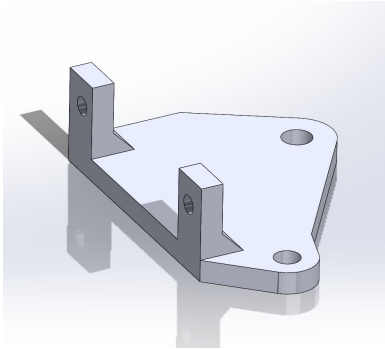
2.6mm hole for
(M2 Crosshead screw)
Clearance fit

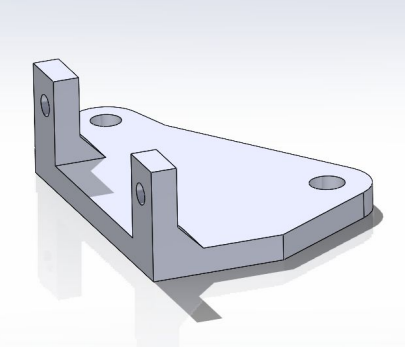
3mm hole for
(M2.5 Crosshead screw)
Clearance fit

Engineering Drawing

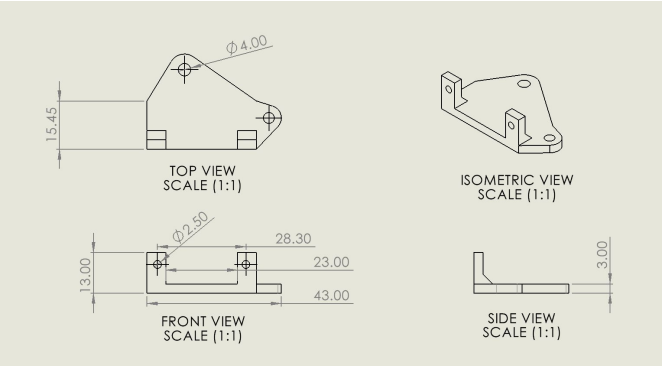


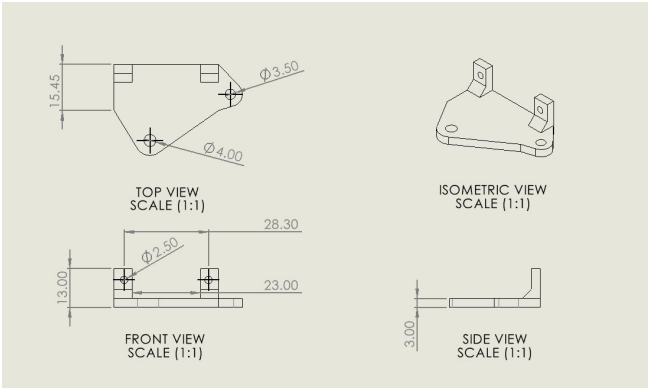
Servo mount x2



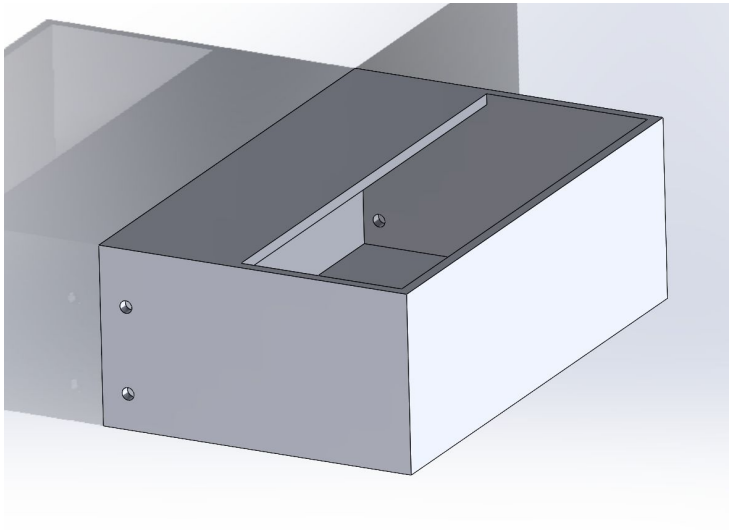


Engineering Drawing

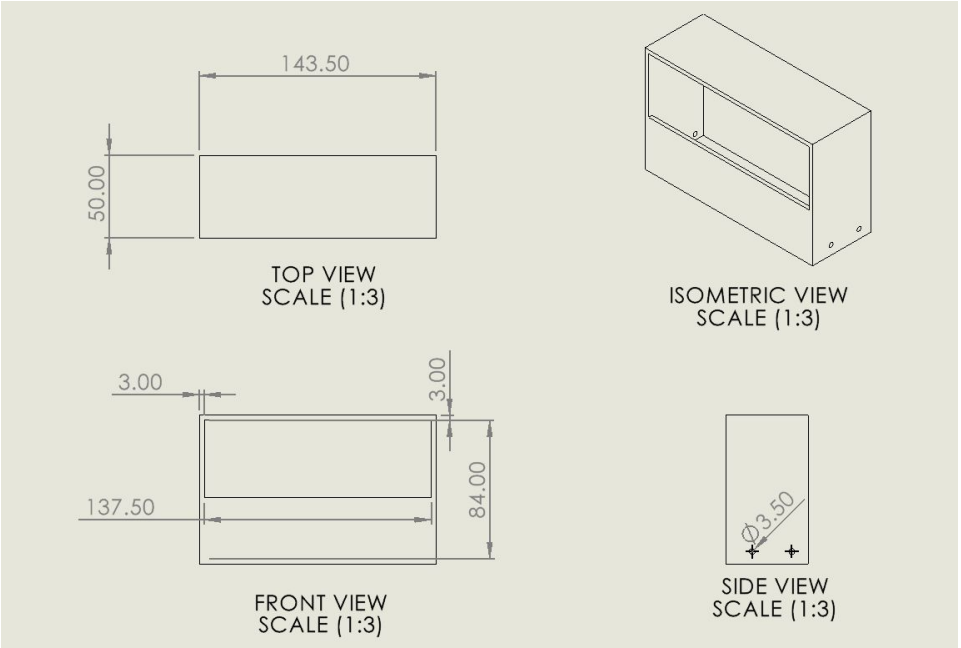




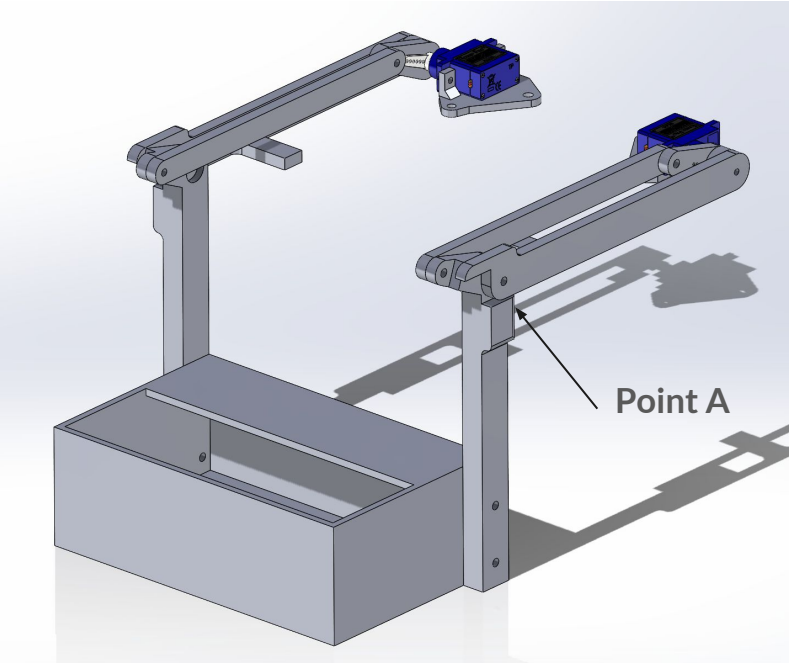
Carrier basket



Engineering Drawing



Arm assembly plan

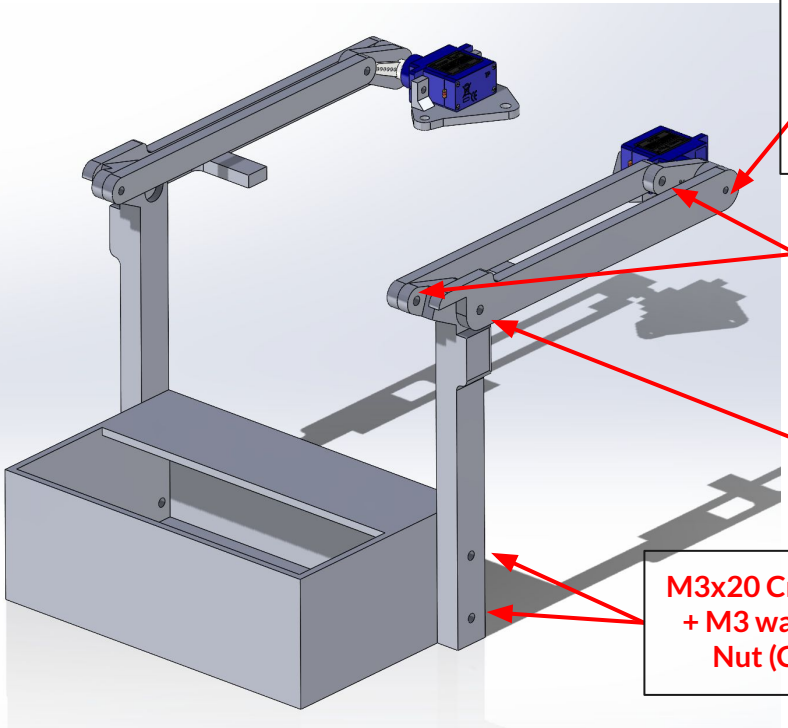


1. 2 Pivot points
   - Servo motor
   - Point A

2. Foldable arm design.
   - Enable turtlebot to mount a longer arm without affecting the robots CG

3. 1 Motor per lifting arm
   - Ensure that the servo have sufficient torque for lifting.

Arm assembly plan



M2x25 Cross-head screw (Clearance fit) + M2 spring washer + M2.5 Hex Nut (Spacer) - Screw into servo

M2.5x16 Cross-head screw + M2.5 Lock Nut (Clearance fit)

M3x20 Cross-head screw + M3 Lock Nut (Clearance fit)

M3x20 Cross-head screw + M3 washer + M3 Hex Nut (Clearance fit)

Parts list

Nuts and Bolts

arm:
2 m3x20
2 m3 locknut
4 m2.5x16
4 m2.5 locknut
2 m2x25
2 m2.5 nut (spacer)
2 m2 spring washer

servo + holder:
4 m3 washer
4 m2.5 washer
2 m2.5 nut
2 m3 nut
2 m2.5x20
2 m3x20
2 m2x8
2 m2 nut

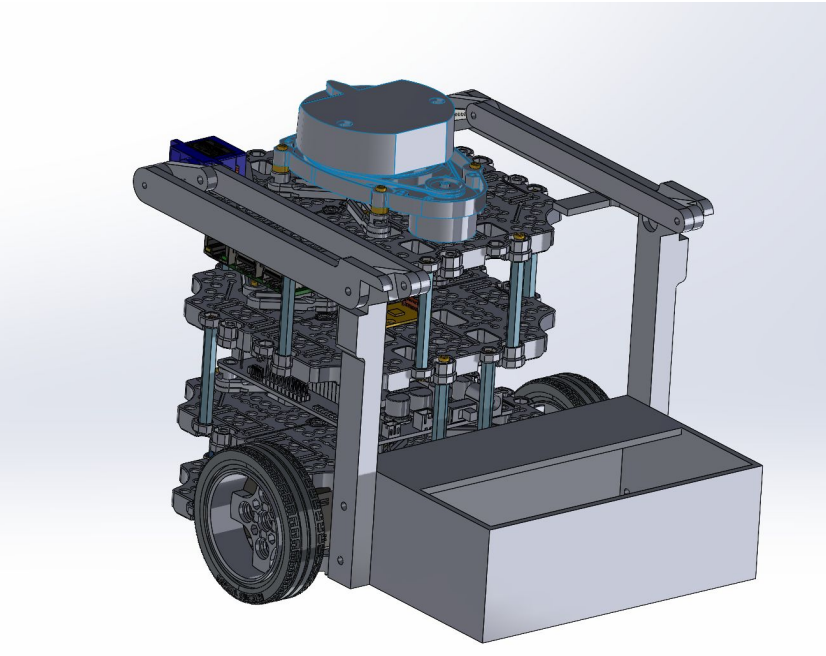
carrier:
4 m3x20
4 m3 nut

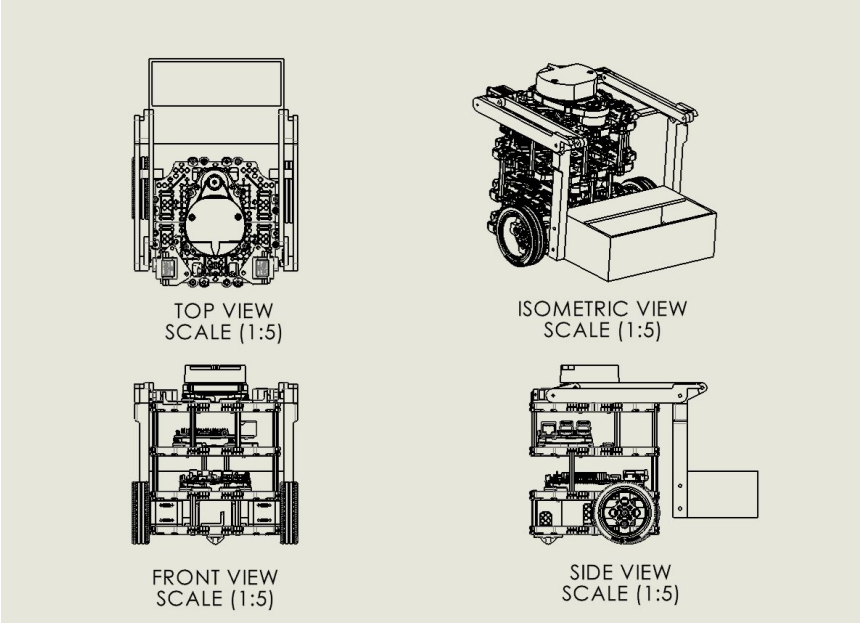
Arm parts

- 2x Carrier arms
- 2x Support arms
- 2x Linking arms
- 2x Servo adaptor
- 2x Servo mount
- 1x Carrier basket

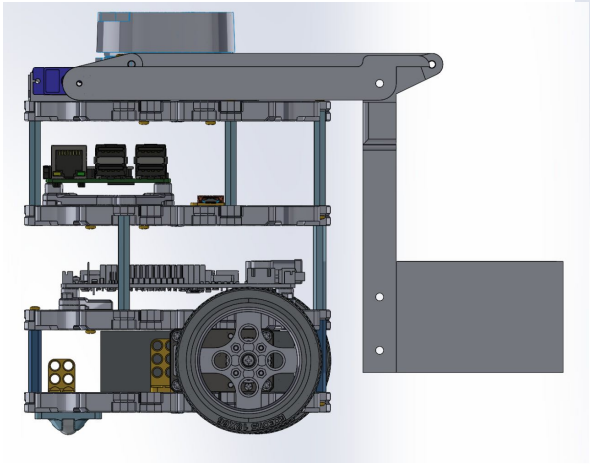
edited 8:24 PM

Overall design

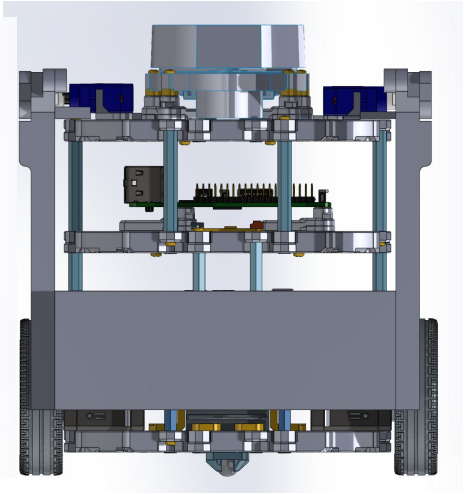




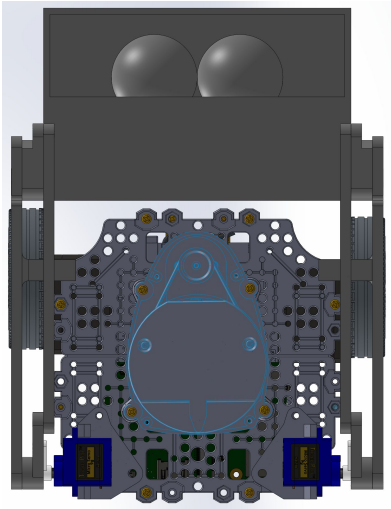
Overall design (Perspective views)



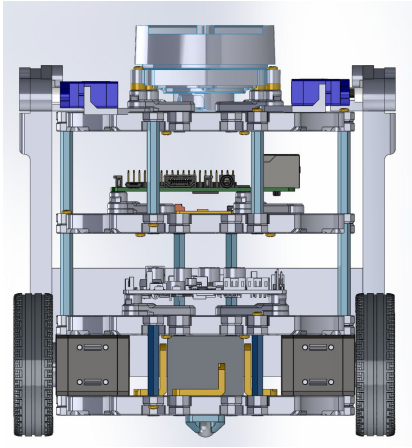
Left side view



Back view



Top view



Front view

Torque Calculation

SolidWorks software to obtain:

- Center of Gravity (CG) position of rotation assembly = 170mm

Physical measurements:

- Weight of Rotating assembly + 5 balls = 110g

Calculations:

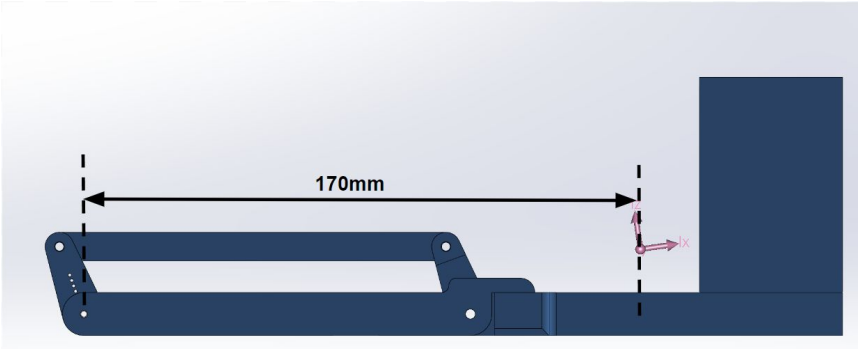
$$Distance\ of\ CG \times Weight\ of\ Rotating\ Assembly = Torque$$

$$Torque = 0.17m \times 9.81\ m/s^2 \times 0.11kg = 0.183Nm$$

Torque required at full extension: **0.183 Nm**

Since max torque of servo = 2 x 1.3 kg cm = **0.255 Nm**

- 0.255Nm > 0.183 Nm **(Got sufficient torque!)**

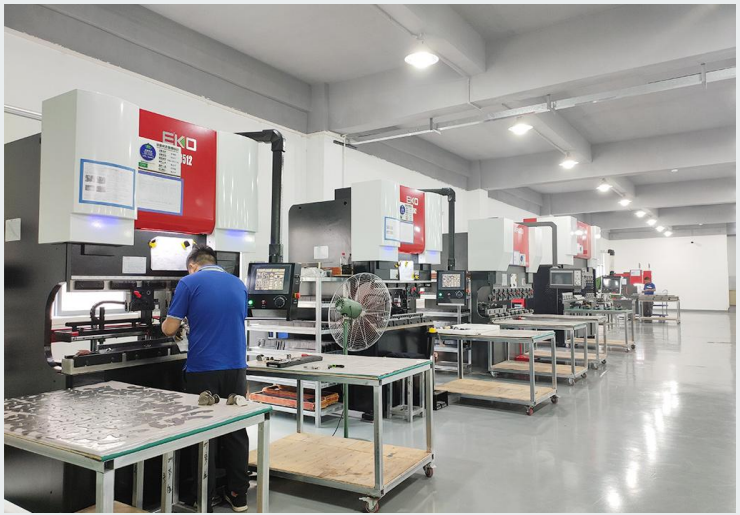


Rubber Band (Counterweight)

Since torque required > 0.255 Nm (2 x 1.3 kg cm)

- Reduce component weight
  - Reprint (added costs)
- Rubber bands (counterweight)
  - Trial & error, just add

Fabrication

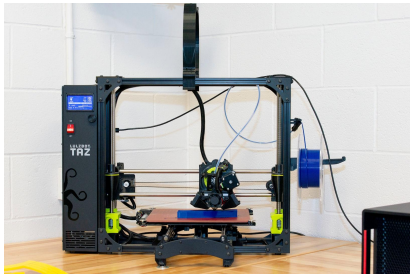


3D Printing (Our decision)

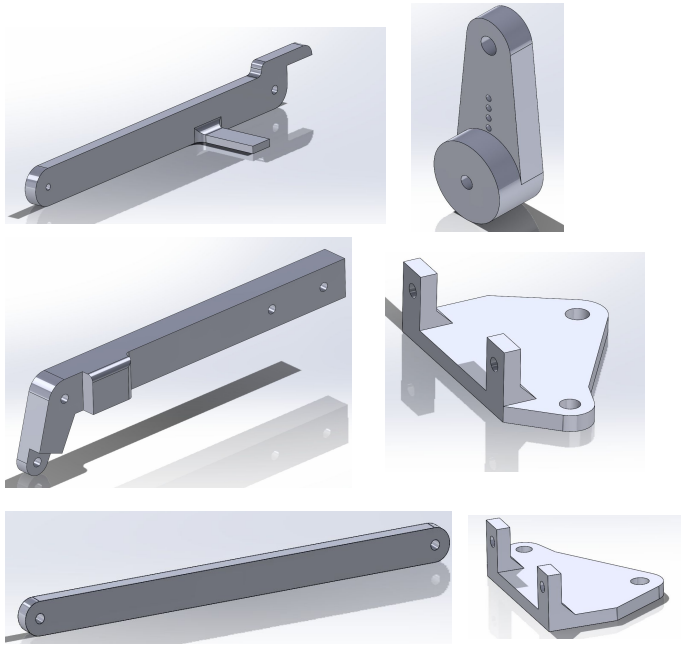
- Versatile (Able to create custom intricate parts)
- Easily accessible
- Low cost of parts

Laser Cutting (Possibility)

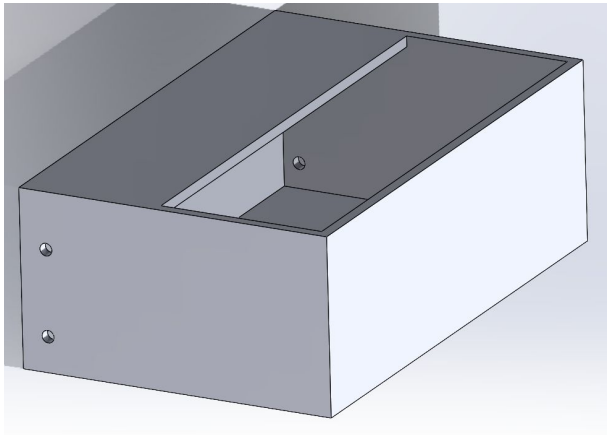
- Faster compared to 3D printing
- Less versatile (Need to cut each component out and assemble)



Fabrication materials

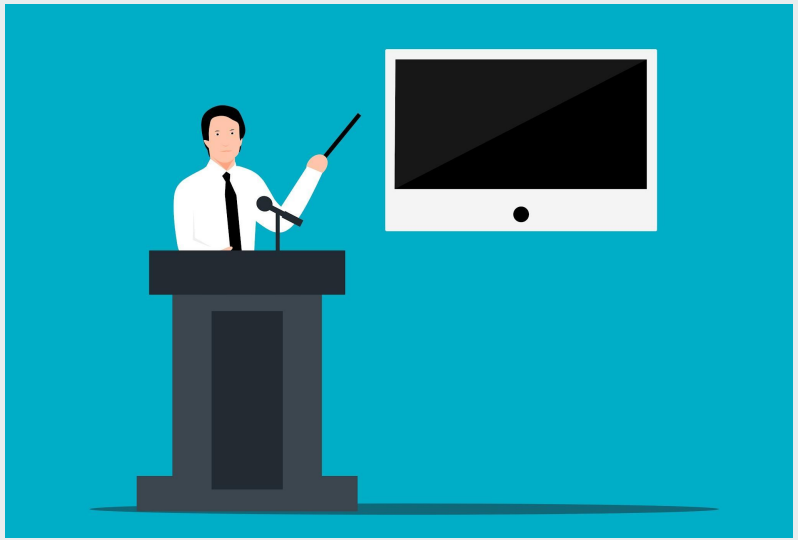


3D Printed parts (PLA)
Reason: Custom parts (Easier to make)



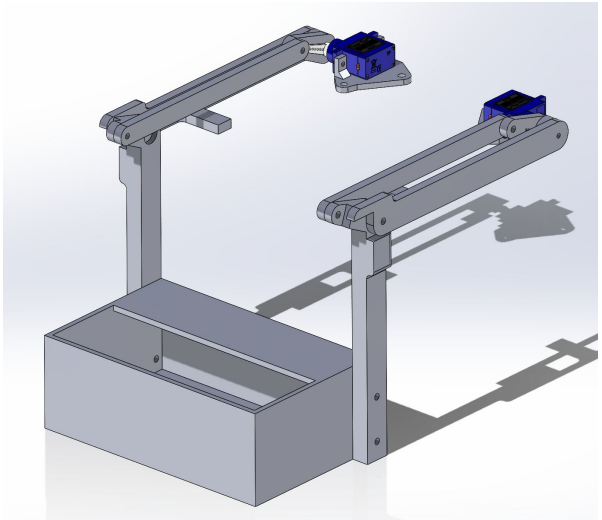
Arts and Craft (Cardboard)
Reason: Save Filament & light material

Demonstration



Choice of components and justifications

Foldable Arm



The foldable arm able to dispense the ball into the bucket in a:

- Dropping mechanism
- More precise (Not heavily influenced by air resistance)
- More controlled manner.

Choice of components and justifications

Sensors and Actuators

Sensors chosen:

- LDS-02 LiDAR
  - Accurate, already implemented
- R-Pi Camera V1
  - Accurate and affordable
  - No assembly required

Previously considered sensors:

- Line Follower
  - Requires assembly
  - Elaborate setup and teardown
- Colour Sensor
  - Requires assembly
  - Difficult to obtain consistent results from far distance
- Other Cameras
  - More expensive
  - Don’t have it on hand

Choice of components and justifications

Sensors and Actuators

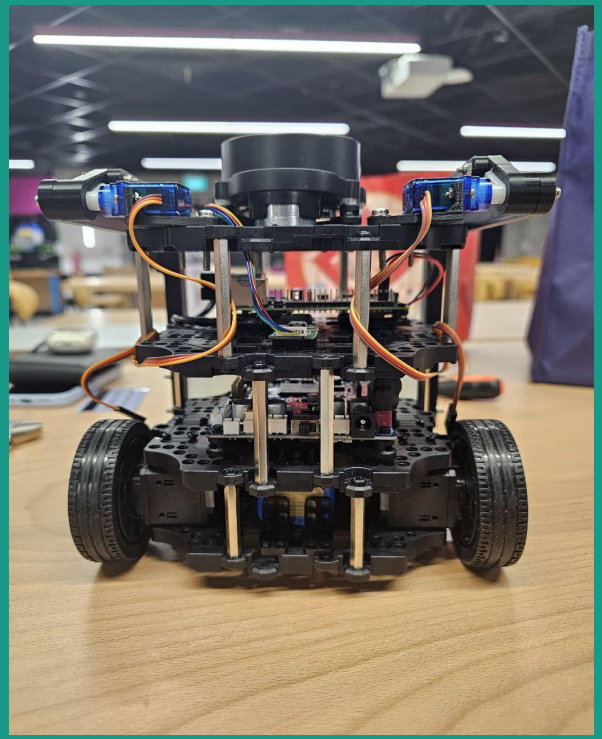
Actuators chosen:

- SG90 Servo motor
  - Easy to mount and implement, with included rotation sensor (POT)
  - Provides rotational movement for our payload mechanism
  - Provided, FOC, can immediately prototype with it

Previously considered actuators:

- Solenoid plunger
  - Requires assembly of circuit, more difficult to implement
  - Does not provide rotational movement required for our payload mechanism
- DC motor
  - Large, requires separate encoder

Current Status and Next Steps



Bill of materials

TurtleBot Set			
Raspberry Pi	2	$804.49	Robotis Shop
Open Cr	1		
Lidar	1		
Dynamixel Motor	2		
Battery and Charger	2		
Other Mechanical Components, Cables and Wires (Waffle plates, Wheels, screws, spacers, etc)	204		
Total	212		-

Mechanical components

Component	Quantity	Price
m2	41	$10
m2 nut	2	
m2 spring washer	2	
m2.5	104	
m2.5 nut	2	
m2.5 nut (spacer)	2	
m2.5 washer	4	
m2.5 locknut	4	
m3	160	
m3 nut	6	
m3 washer	4	
m3 locknut	2	
3D printed pieces	9	$20.13

Budget plan

Component Name	Quantity	Price	Link	Total
SG90 Servo	2	FOC ($4.2)	https://www.makersupplies.sg/products/sg90-9g-servo-motor	$8.4
Raspi Camera	1	$6.99	https://shopee.sg/LOCAL-STOCK-1-month-warranty-Element14-Raspberry-Pi-NoIR-Camera-1080P-IR-with-LED-i.52750998.11543905857?sp_atk=6dc514e3-063d-43d0-b9ba-6110ad85ae2d&xptdk=6dc514e3-063d-43d0-b9ba-6110ad85ae2d	$6.99
3D print charges	3 hr 44 mins 141.86g	$2/hr $40/1000g filament	-	$13.14
Nuts and bolts	333	$10	-	$10
Total (if we buy everything from expensive SG sources)				$38.53

Potential Challenges and Improvements

Potential challenges:

-Implementation of A* algorithm

–arm and carrier might collapse after certain degree

Improvements needed:

-Orientation of OpenCR Board

-refine the carrier

-mount for camera


THANKS, TEAM
YOU ARE FABULOUS!
memegenerator.ne