Project Simple Robot - Final Report

Group Members:

Mani Raj Rejinthala

Koushik Goud Shaganti

Kenny Jia Hui Leong

Ethan Olderog

Client:

Elvis Kimara

Course:

COM S 4020

Instructor:

Simanta Mitra

TAs:

Shaiqur

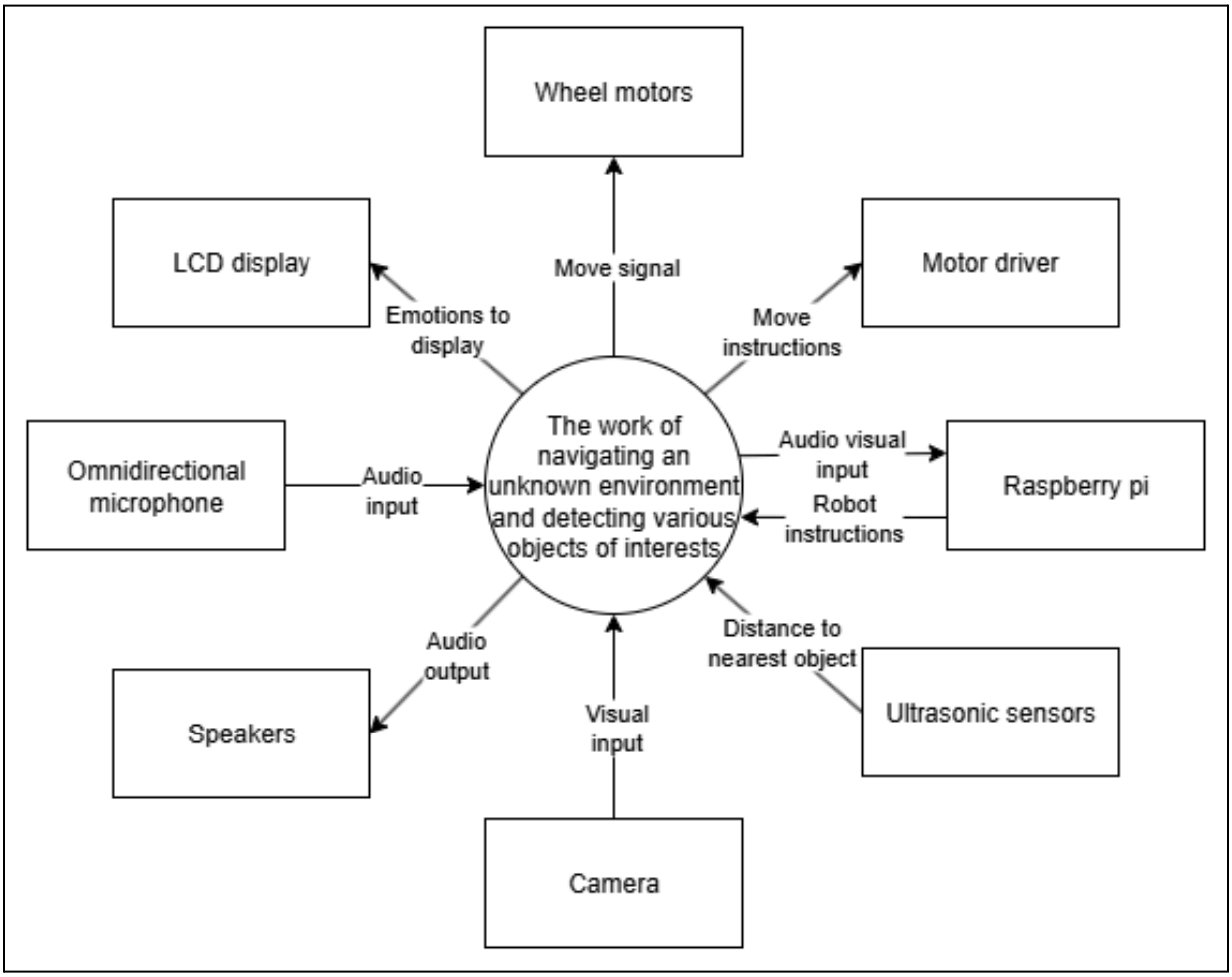
Scott Song

Table of Contents

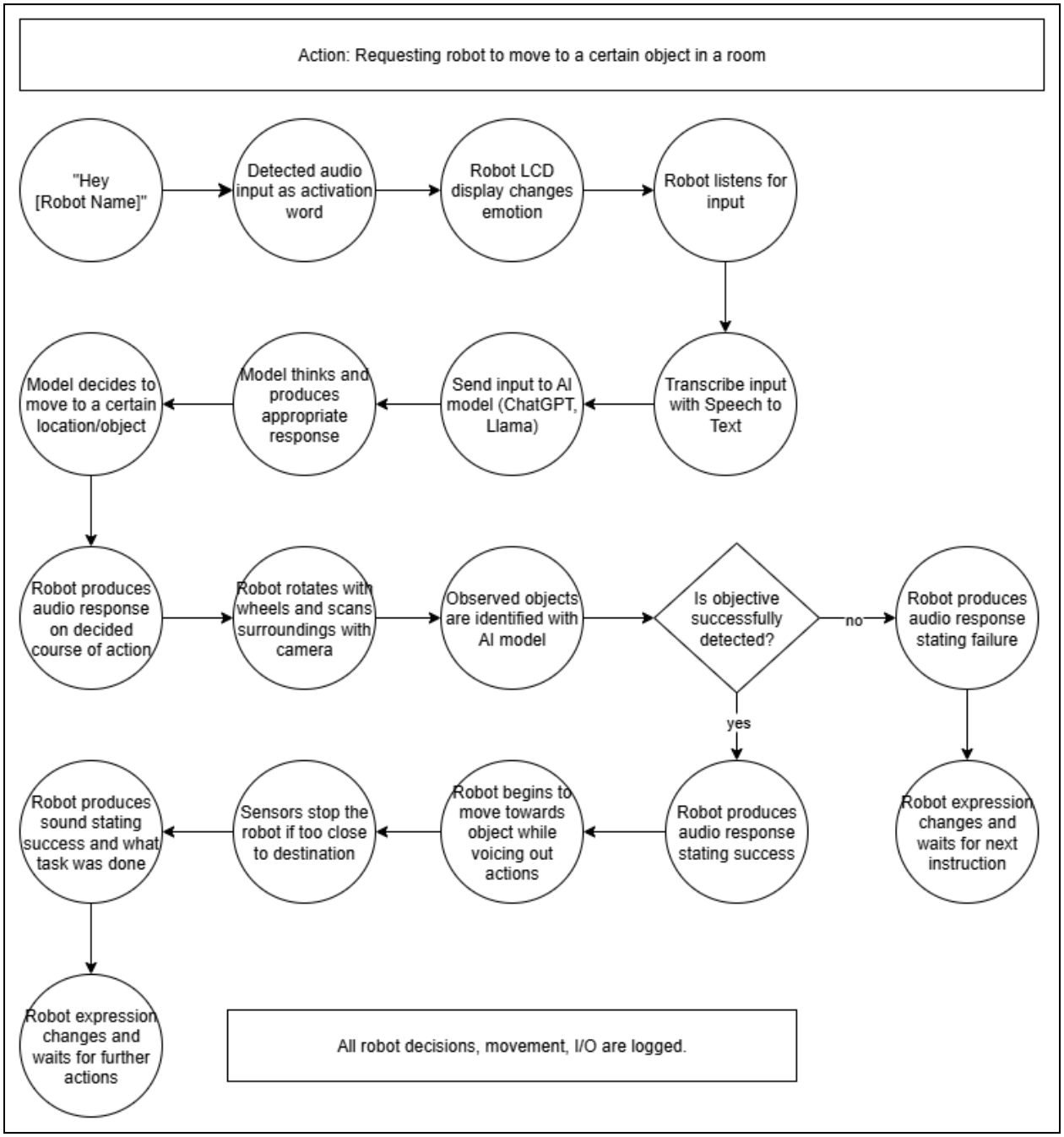
CHAPTER 1: Requirements..... 3
Simple Robot Context Diagram..... 3
Simple Robot Behavioral Specification Flow Chart..... 4
Functional Requirements..... 5
Non-Functional Requirements..... 6
CHAPTER 2: Design..... 7
a) Robot backend..... 7
b) Prototyping the chassis..... 8
c) Assembly of a remote-controlled, movable robot prototype using the cardboard chassis..... 14
d) 3D modelling the robot chassis..... 18
e) Improving the robot prototype through cost effective means..... 19
f) Connecting the ultrasonic sensors..... 20
g) The website for remote controlling the robot..... 23
h) Issues with robot turning, weight distribution, fair navigation..... 24
i) Robot Software Workflow..... 27
CHAPTER 3: Work Done..... 28
CHAPTER 4: Results..... 30
CHAPTER 5: Appendix..... 31
Final presentation slides..... 31
Final demo video..... 40

CHAPTER 1: Requirements

Simple Robot Context Diagram



Simple Robot Behavioral Specification Flow Chart



Functional Requirements

a) WHILE the robot is in sleep state, WHEN the operator says “Hey Robot”, the robot shall receive audio input through the omnidirectional microphone and enter a ready state for receiving future commands.

b) WHILE the robot is in a ready state, WHEN the operator voices out commands, the robot shall receive those commands through the omnidirectional microphone.

c) WHEN the robot receives audio commands, it shall process it into text using text-to-speech.

d) WHEN the robot has transcribed audio commands in text form, it shall send the commands to XX AI model to be processed.

e) WHILE the robot is in find object mode, the robot shall capture images through the pi camera and send it to the XX AI model.

f) WHILE the robot is in find object mode, IF the XX AI model does not find the desired object and returns false, THEN the robot shall rotate in the X direction to process a new view.

g) WHILE the robot is in find mode, WHEN the desired object has been detected by the XX AI model, the robot shall enter a move state and move towards the object.

h) WHILE the robot is in a move state, the robot shall receive distance to object using the ultrasonic sensors.

i) WHILE the robot is in a move state, WHEN the ultrasonic sensors indicate that the object is in X distance to the robot, the robot shall stop.

j) WHEN the robot successfully completes a command, the robot shall produce an audio output “XX” indicating success through the speakers and enter a sleep state.

k) WHEN the robot receives a command to follow the user, the robot shall first move and find the user through its camera.

l) WHILE the robot is carrying out the follow me command, WHEN the robot detects the first person in its camera, the robot shall continuously move and turn towards the moving target.

Non-Functional Requirements

a) Enable the robot to respond in an optimal time lap.

b) WHEN the robot receives an audio command through its omnidirectional microphone, the robot shall process and return an audio response within 5 seconds.

c) WHEN the robot receives an audio command, IF the command is to find an object, then the robot shall begin autonomous exploration to find said object.

d) WHILE the robot is in autonomous mode, WHEN the robot successfully finds the desired object, the robot shall stop and return an audio response signalling its success.

e) WHILE the website is in manual mode, WHEN the user presses or clicks one of the move commands, the website shall send a movement command to the robot.

f) WHILE the website is in autonomous mode, WHEN the user sends a command to find an object, the website shall send a find object command to the robot.

g) WHILE the robot is controlled by the website, WHEN the robot performs an action or processes a request/command, the robot shall send a log of said action/command to the website.

h) WHEN the robot is deciding on an action, the robot shall change its facial expressions accordingly.

i) WHEN the robot has determined a course of action, the robot shall express its thought processes through its speakers to the user.

CHAPTER 2: Design

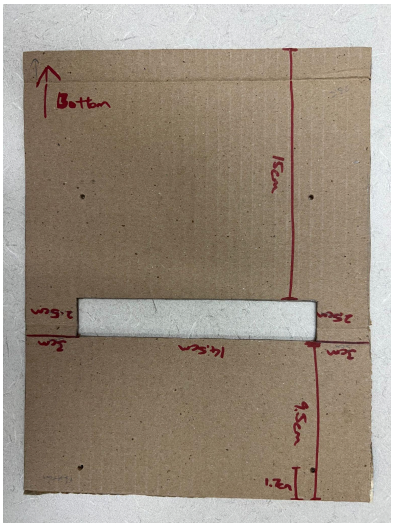
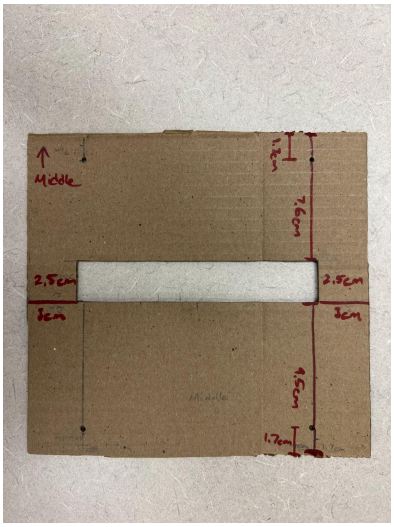
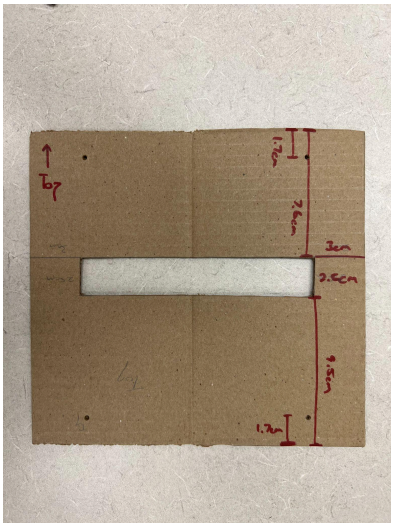
a) Robot backend

File	Purpose
main.py	Main entry point
CommandHandler.py	Routes commands to actions
IpcClient.py	WebRTC socket communication
voice_listener.py	Wake word + STT
robot_utils.py	TTS, LLM, object detection
groq_utils.py	LLM API with memory
MotorControl.py	GPIO motor control
ObstaclePrediction.py	Ultrasonic sensors
Autonomous.py	Self-navigation logic
Face.py	Animated face display
Controller.py	Mode(FollowMe/Find/Manual) and Medium (Voice/Web client) controllers for Mutual exclusive control over resources
Camera.py	Camera + virtual camera feed
config.py	TTS/STT configuration for the client.
find_target.py	Autonomous target Discovery
follow_me.py	Follow Person in the frame

b) Prototyping the chassis

There were a lot of ideas for partitioning various components of the Simple Robot and ultimately we decided to split it up into three distinct layers. Each layer is supported by standoff screws and is responsible for holding various components that would be needed for the autonomous AI robot to function.

Three main layers of the Simple Robot

		
Bottom	Middle	Top

To the left is the bottom layer, it will be responsible for wheel/motor attachment, for storing the battery holder (powers the motors) including batteries and the L298N motor driver. The speaker box/container will be installed at the front of this layer (represented by the location of the red arrow).

In the center we have the middle layer. This layer is responsible for storing the raspberry pi and powerbank.

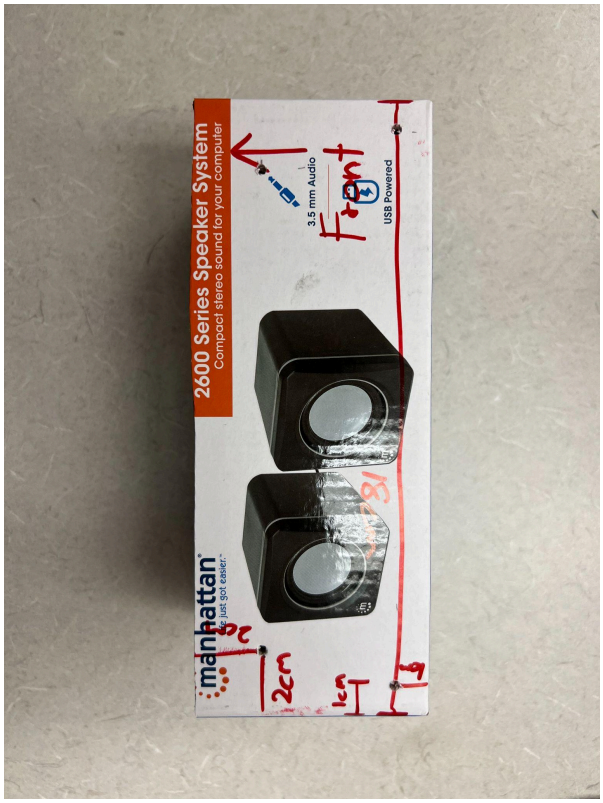
Finally, we have the topmost layer to the right. This top layer is responsible for storing the omni-directional microphone as well as providing some stability for the LCD screen holder.

Rectangles were carved out in the middle of all layers to allow for wires to interweave between them instead of dangling by the sides.

This is the speaker container which will be installed at the front of the bottom layer. It will also be the base for installing the LCD screen holder. Rectangles have also been cut out to let the speaker wires pass through through the back.



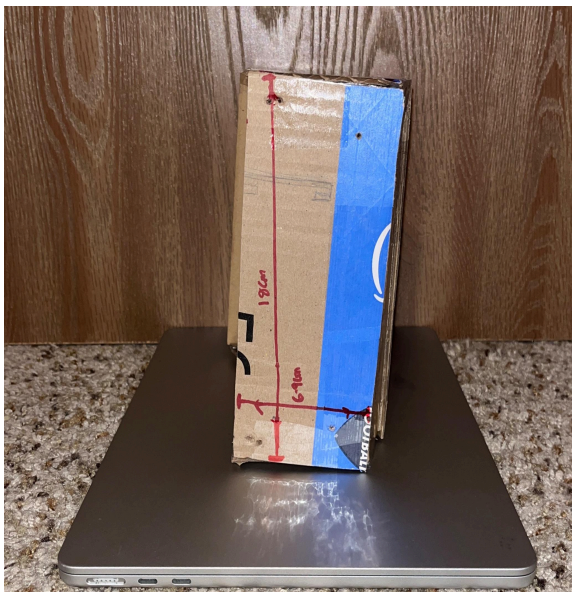
Inside



Topside



Front side



Bottom side



Right side



Left side

This is the LCD screen holder that is designed with 1 cm extensions at the top and bottom on the front side to accommodate external connection pins that extend outward like hdmi and other pins. This holder will be mounted onto the speaker box (shown above) and secured using screws for stability. Additionally, the camera will be mounted directly on the screen, ensuring proper alignment and integration with the overall setup.

Three ultrasonic sensors will be attached in the space between the screen and the speakers—one on the left, one on the right, and one in the center.

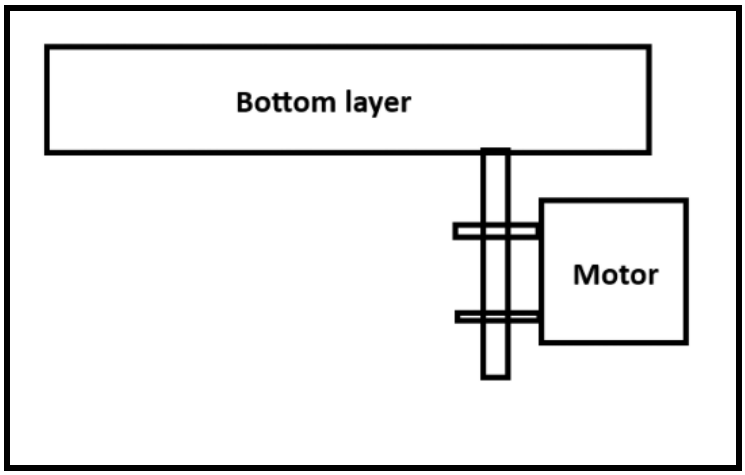
The following are some views for when the chassis is assembled (excluding wheel motors for now).

		
Front	Side	Top

As you can see, while somewhat structurally sound, not all components have been placed on the layers yet as cardboard is not sturdy enough to support all of the components' combined weight (notice the downward bending of the layers). Thus, it is at this point that we had decided to get into contact with the software engineering team to model and 3D print a chassis based on this working prototype.

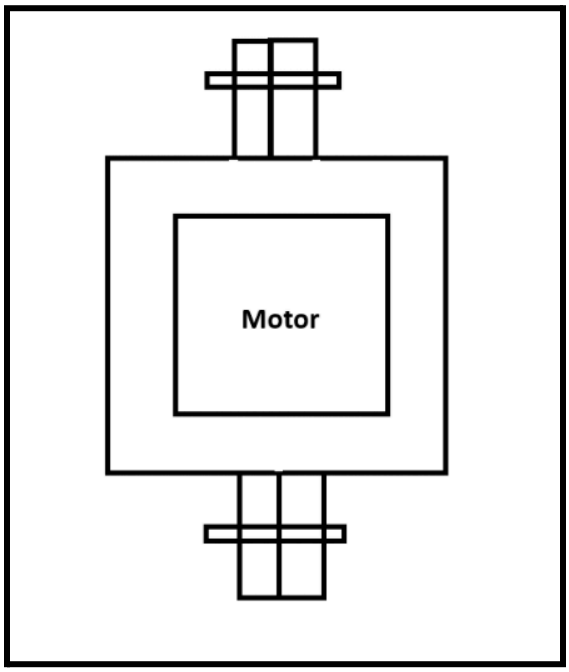
There were a number of suggestions on how the motors could be attached to the base layer of the chassis.

1) Have rectangle protrusions for the attachment of the motor. At first, this was thought of as a fairly good choice. It was easy to 3D print as it allows for motors to be attached to its sides. However, this design poses some stability issues and making it thicker would cause the standoff screws to not fit through the protrusion.



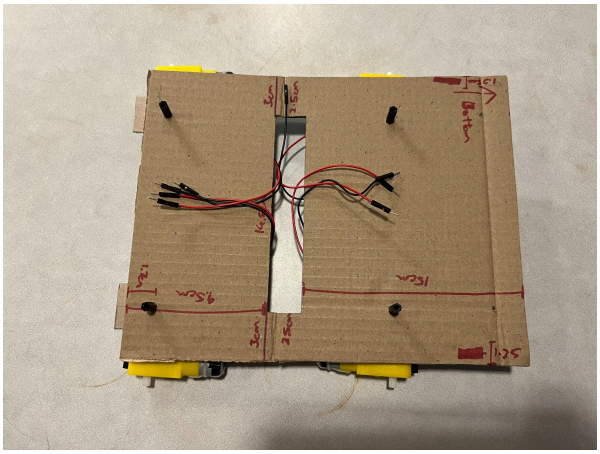
Suggestion 1

2) Have a clamp to secure the motor. However, attaching the clamp itself to the base would be challenging since it would be hard to be 3D printed together with the bottom layer. There were also concerns with the structural integrity of the attaching points.

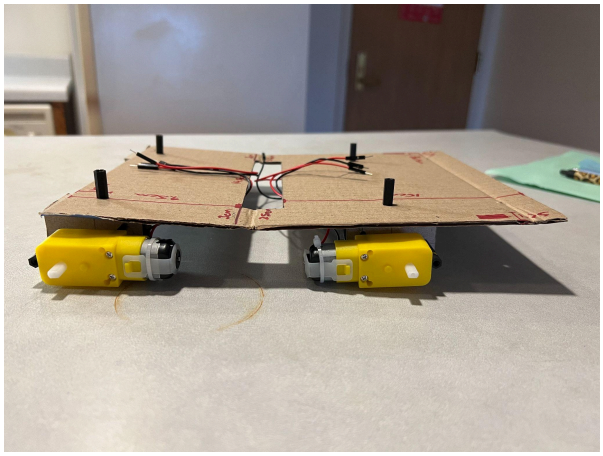


Suggestion 2

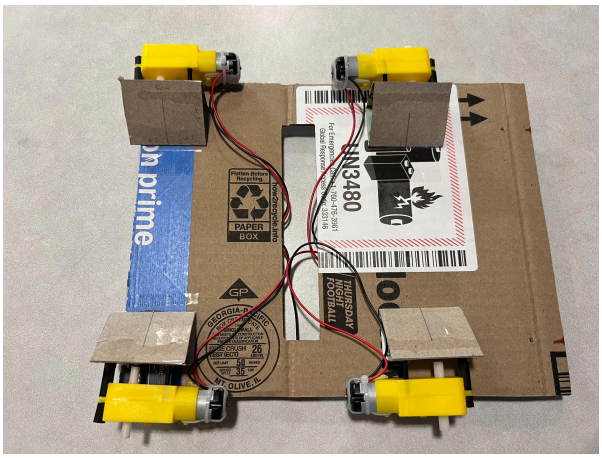
3) Use triangles to create extensions that can attach to both the bottom layer as well as the wheel motors. The usage of triangle extensions was that it would distribute the weight of the entire robot to the wheel motors effectively and maintain the stability of the overall structure. The following are pictures detailing the third suggestion.



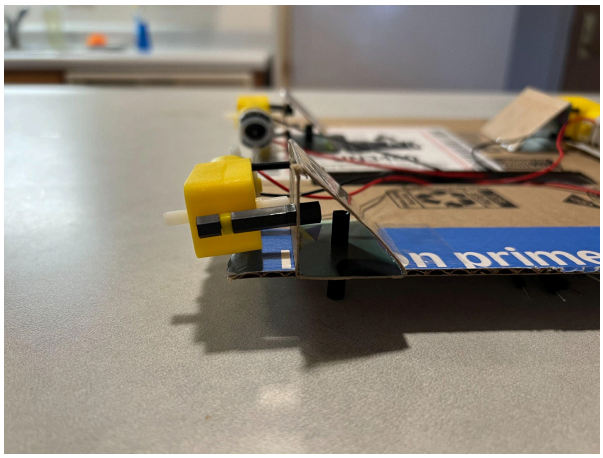
Top



Side



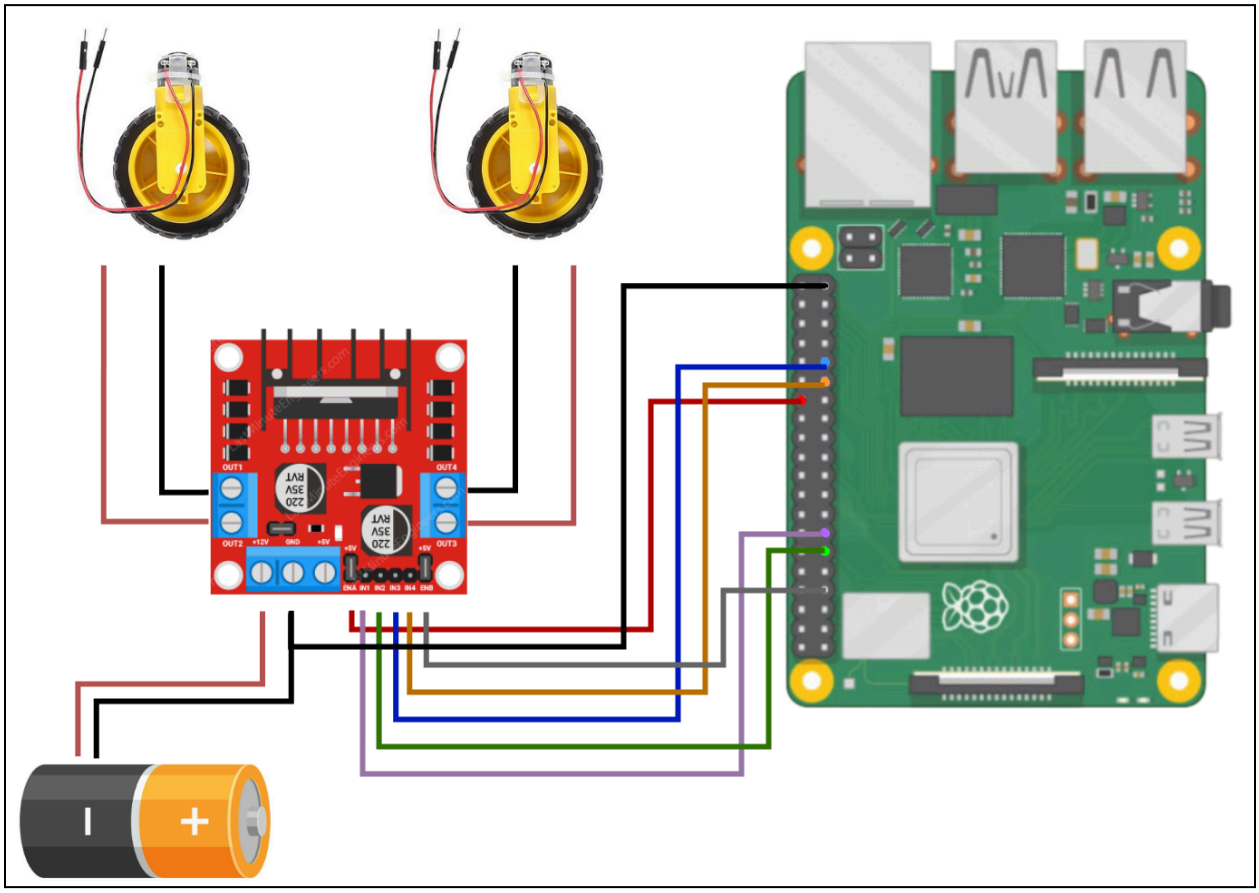
Bottom



Closeup

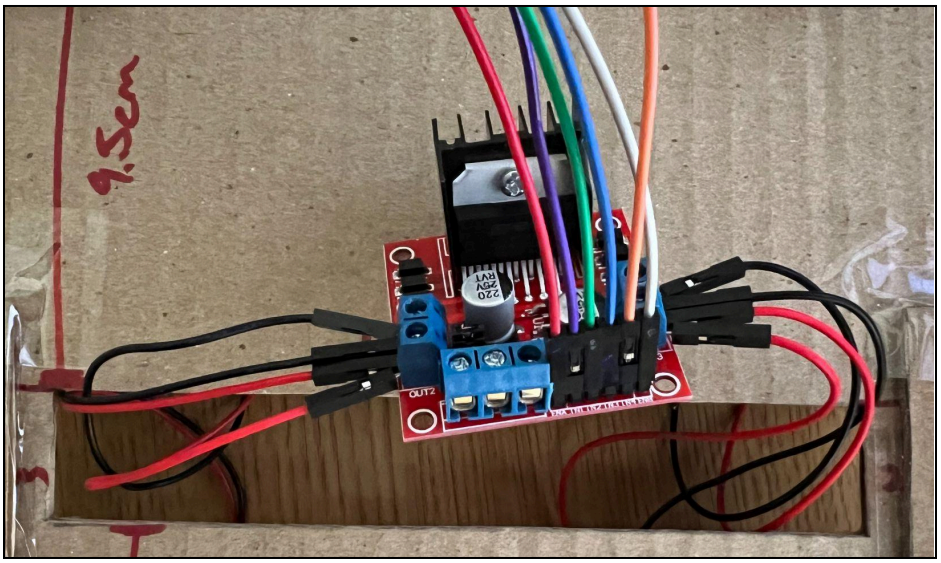
c) Assembly of a remote-controlled, movable robot prototype using the cardboard chassis

The following wiring diagram has been created to illustrate the overview of how the raspberry pi, DC wheel motors, L298N motor driver, and batteries are wired together to create a remote-controlled, movable robot.



Robot Wiring Diagram

i) Connection between the DC wheel motors and the L298N motor driver



Wiring the wheel motors to the L298N motor driver was divided into two sections: one for the **left-side wheels** and one for the **right-side wheels**.

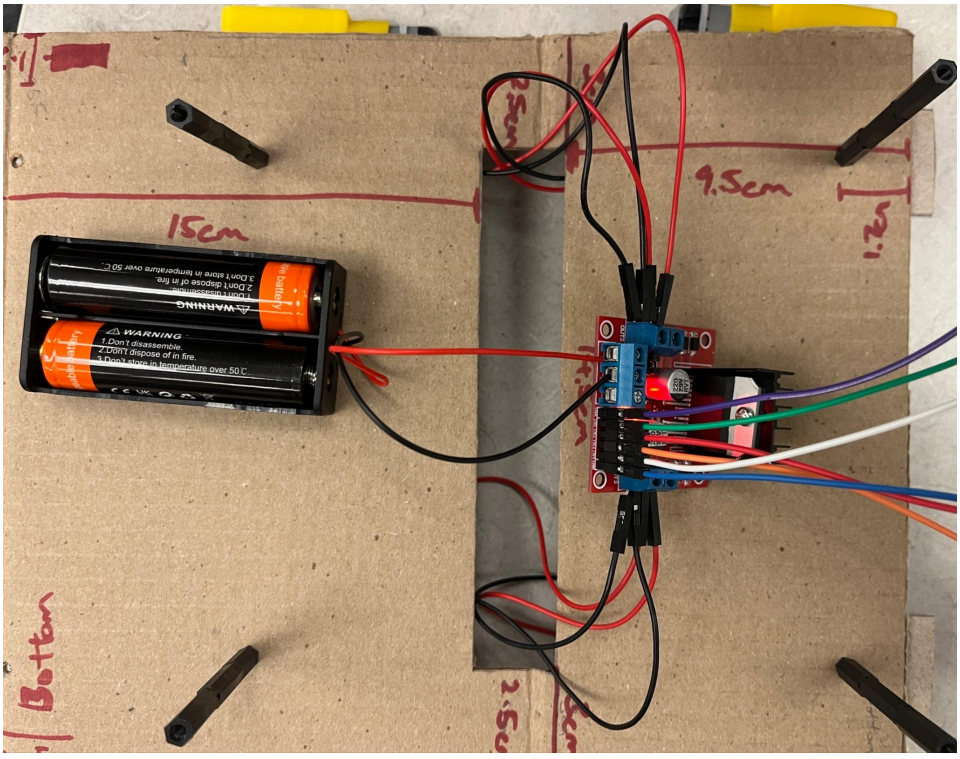
- For the **left-side motors** (front and back):
    - The **negative wires** are connected to the **OUT2 terminal** of the L298N.
    - The **positive wires** are connected to the **OUT1 terminal**.
- For the **right-side motors** (front and back):
    - The **negative wires** are connected to the **OUT4 terminal**.
    - The **positive wires** are connected to the **OUT3 terminal**.

As you can see, both the front and rear motors on each side are paired together and connected to either side of the motor driver. This is because from the raspberry pi's perspective, the robot only has two wheels, left and right, when in reality there are actually four. This is to simplify the process of moving and turning the robot and is shown in the table below.

Movement	Left Side (Left front and rear wheels)	Right Side (Right front and rear wheels)
Forward	Forward	Forward

Reverse	Reverse	Reverse
Left	Reverse	Forward
Right	Forward	Reverse

ii) Connection with the Raspberry Pi to the L298N motor driver.



L298N Motor Driver Pins (6 control pins)

- ENA - Controls power output (speed) and duration of rotation for the wheel left wheel motors.
- IN1 & IN2 - Controls clockwise or counterclockwise rotation of the wheel motor.
- IN3 & IN4 - Controls clockwise or counterclockwise rotation of the wheel motor.
- ENB - Controls power output (speed) and duration of rotation for the wheel left wheel motors.

Raspberry Pi GPIO Connections

Female-to-female jumper cables (with pin headers) were used to connect:

- ENA → GPIO 1 (pin 28)
- IN1 → GPIO 17 (pin 11)
- IN2 → GPIO 27 (pin 13)
- IN3 → GPIO 6 (pin 31)
- IN4 → GPIO 5 (pin 29)
- ENB → GPIO 4 (pin 7)

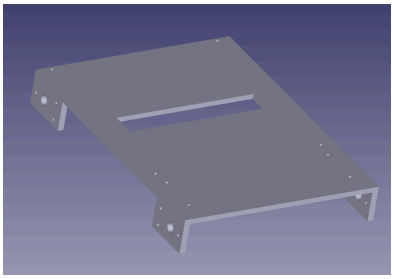
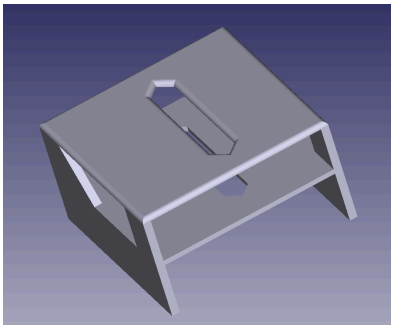
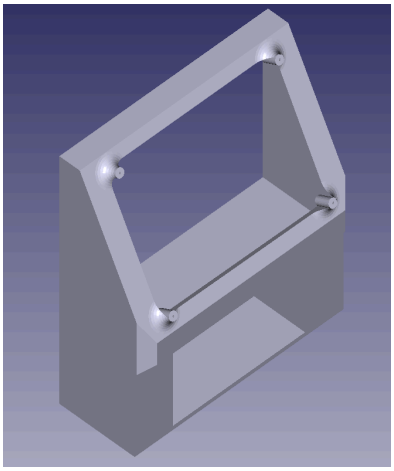
The raspberry pi GPIO pinout was referenced to ensure all connections were correct. Since the raspberry pi has no hardware protections in place, an incorrect pin connection may result in permanent damage to the entire board. Caution is advised.

Battery to L298N motor driver Connections (including raspberry pi)

- 12V+ → negative wires of battery
- GND → positive wires of battery
- GND → pin 36 (to ground the connected components)

d) 3D modelling the robot chassis

To create a 3D model for the chassis of the robot, we acquired the help of a mechanical engineering student introduced to us by the client. Using our cardboard prototype chassis as reference for the exact measurements we want for our 3d model, we ended up with three separate parts.

Base layer	Middle section	Front section
		

Throughout the modelling process, it was decided that some changes were needed to accommodate the limitations of 3D printing (printing of a given component can't exceed the size of the machine) as well as the structural integrity of the resulting product (connections that may break off due to over exertion of force). Since disassembly of the robot was not planned in the near future, the base layer will be connected to the middle and front section by screwing a M.3 screw through a perfect (or exact sized) hole. As for the wheel motors, they will be connected through screws and nuts.

The middle section is actually composed of the middle and topmost layers of the cardboard prototype. They were combined to make assembly easier as there was a challenge in connecting all separate layers through standoff screws alone since they were relatively short on the connection joints.

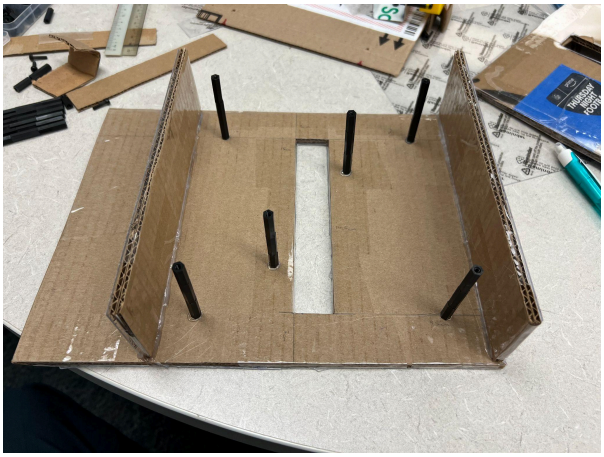
The front section was modelled to combine the speaker box and LCD screen holder, again for ease of printing and assembly. There are four protrusions for securing the LCD screen using M.2 screws. A gradual slope around the circumference of the protrusions were added to add structural integrity and to prevent them from breaking off.

Once the modelling was completed, the design was handed off to the Additive Manufacturing Lab of Black Engineering for a quote on the 3D printing job. However, the quote or cost for printing that we had received was unexpectedly high and exceeded our budget (up to ⅔ of the robot's cost on components so far). So, we had to come up with a more cost effective means of acquiring or improving our preexisting robot chassis.

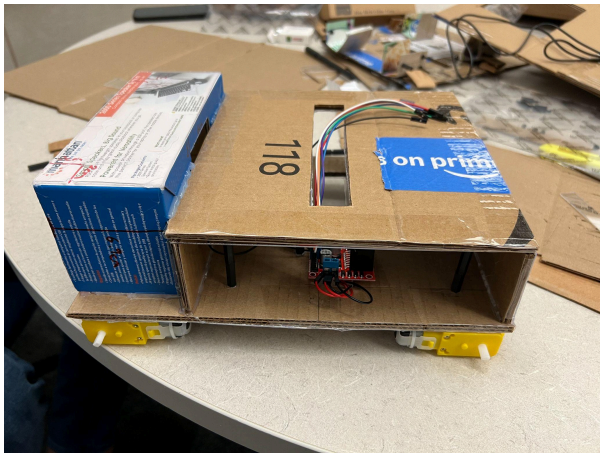
e) Improving the robot prototype through cost effective means

The problem so far with cardboard is that it is simply too soft, and is prone to deforming when too much force is applied to it. Thus, we decided that acrylic sheets could be used to reinforce the cardboard layers. But, acrylic sheets are not cheap either, and the cheaper options tend to only have a thickness of 0.1-0.2 mm, which also changes shape under pressure. What about combining both to create the layers of the robot? Acrylic sheets provide the rigidness necessary for the layers to not bend under pressure, and cardboard gives thickness to the layers so the acrylic sheets do not easily change shape. As for the medium used to connect these two parts together, glue gun seems to be the most cost effective option as well.

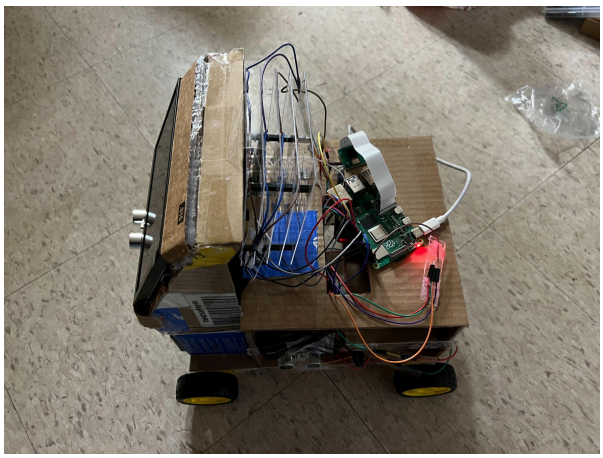
Thus, it was decided that the team would proceed with creating a acrylic sheet reinforced cardboard frame, connected using a glue gun.



Base layer



Base layer + middle layer + speaker box

	
All three layers + speaker box	All layers + speaker box + LCD holder

All layers/walls were using double-layered cardboard reinforced with acrylic sheets and secured with glue. The new design is mostly faithful to the original cardboard prototype, with the exception of walls in the front and back sections of the layers instead of standoff screws for connecting them. Wheel motors are also now directly glued to the base layer instead of using any external joints for connection.

Some standoff screws were erected on the base layer to prevent the middle layer from deforming too much from the weight of the components.

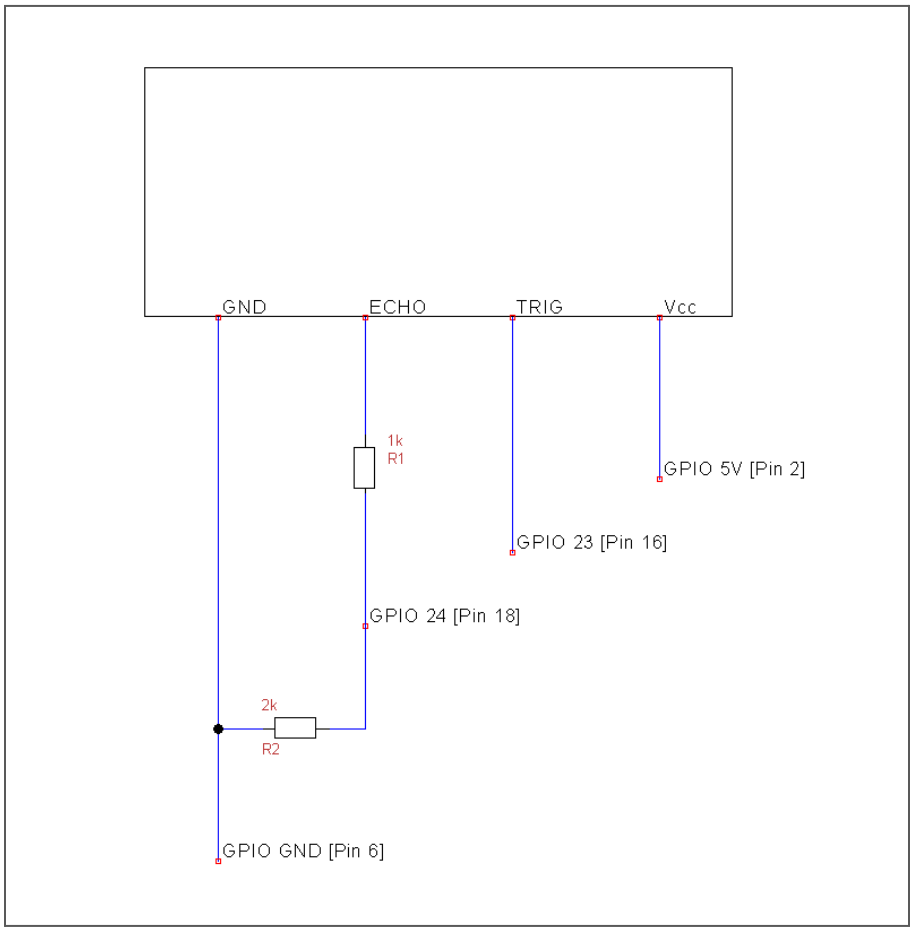
The newer design is overall a lot more sturdy compared to the cardboard prototype, but has the downside of being significantly heavier. This had unexpectedly affected the turning radius of the robot, and was compensated by increasing the turning time.

f) Connecting the ultrasonic sensors

Connections of the ultrasonic sensors were not as straightforward as we had initially thought when we attempted to get them working. To be put it simply, the sensors have four connections, a ground connection, a power connection, an ECHO connection for receiving signal data, and a trigger connection for signaling the sensor to start.

The focus is on the ECHO connection, where its output is received by the raspberry pi and is used to determine/calculate the distance between the sensor and its nearest detected obstacle. However, this connection output signals at 5.0 volts, whereas the raspberry pi accepts 3.3 volts. This causes two problems: 1. We would, overtime,

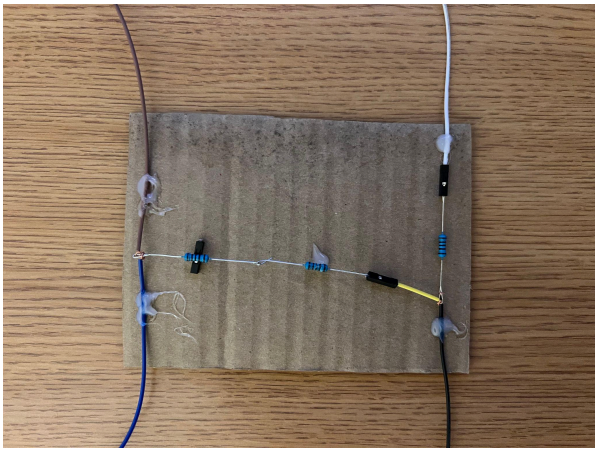
damage the raspberry pi through this difference of voltages. 2. The signals we did manage to receive were simply not consistent or accurate, and the distance calculations were not usable whatsoever. So we would need some way to reduce the voltage of the ECHO output in order to receive correct signal readings.



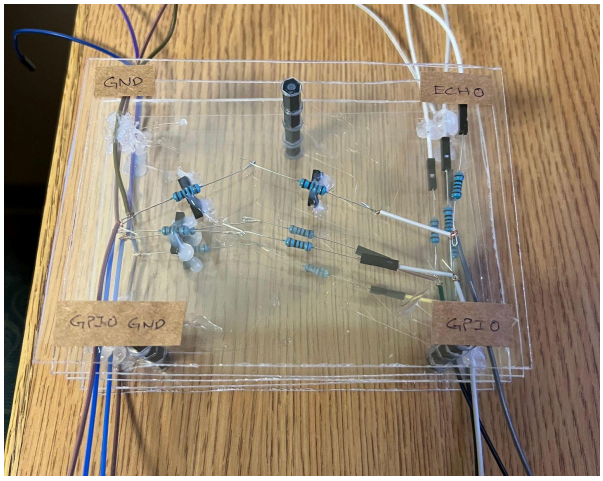
Schematic for connecting the ultrasonic sensor from PiHut

Ideally, we would use a breadboard, resistors, and jumper cables to create this voltage divider. But since we were trying to spend less so as to not exceed the budget, we decided to create this through a cardboard prototype.

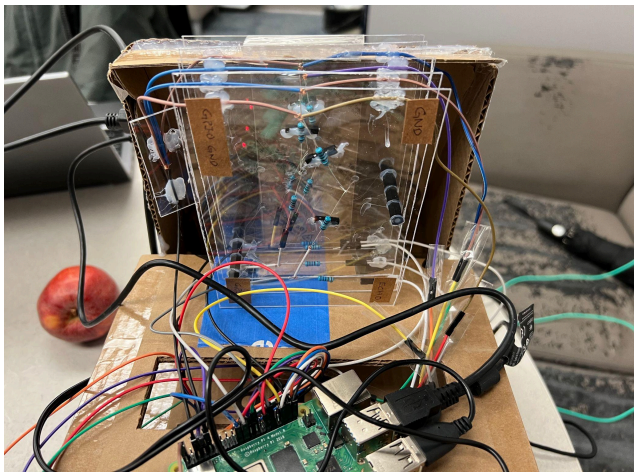
To create the voltage divider, we made use of three 1k resistors, some modified jumper cables, and a piece of cardboard and glue to hold the connections in place. This ended up working, as we began to receive accurate distance readings from one sensor. And so with this as a base for our voltage divider, we needed to build four of them. This is because the output signal of each ECHO output needs to be independent from other circuits, as mixing their signals would result in us not knowing which readings belong to which direction. The trigger and power connections were allowed to be shared as they were not connected to the voltage divider.



Voltage divider prototype

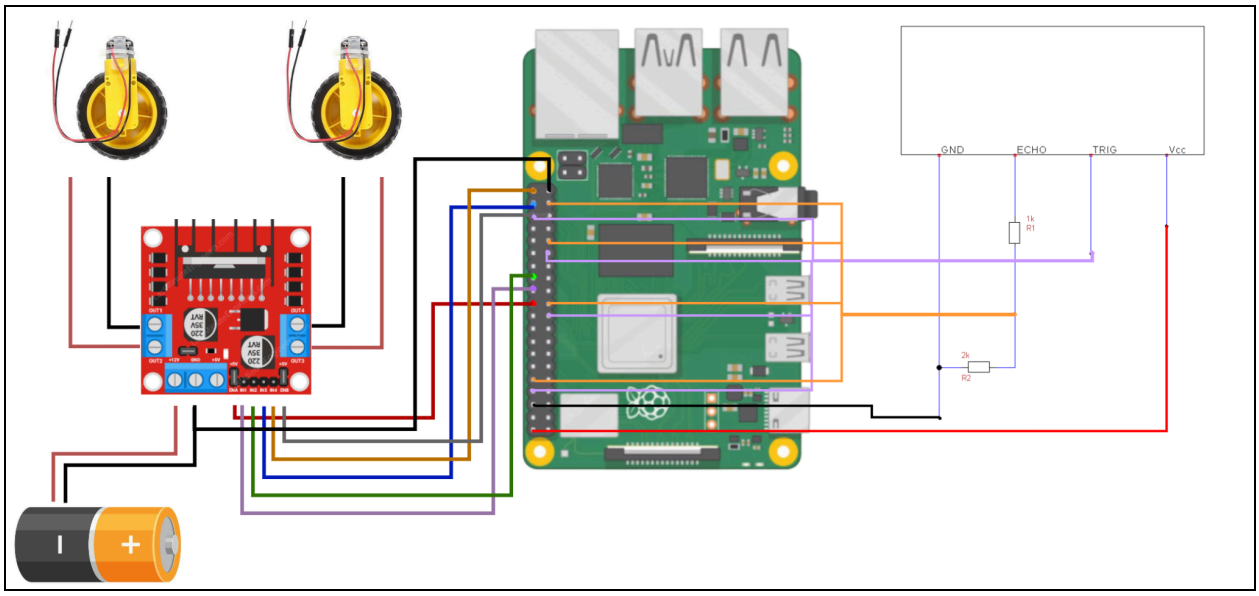


4-layered voltage divider



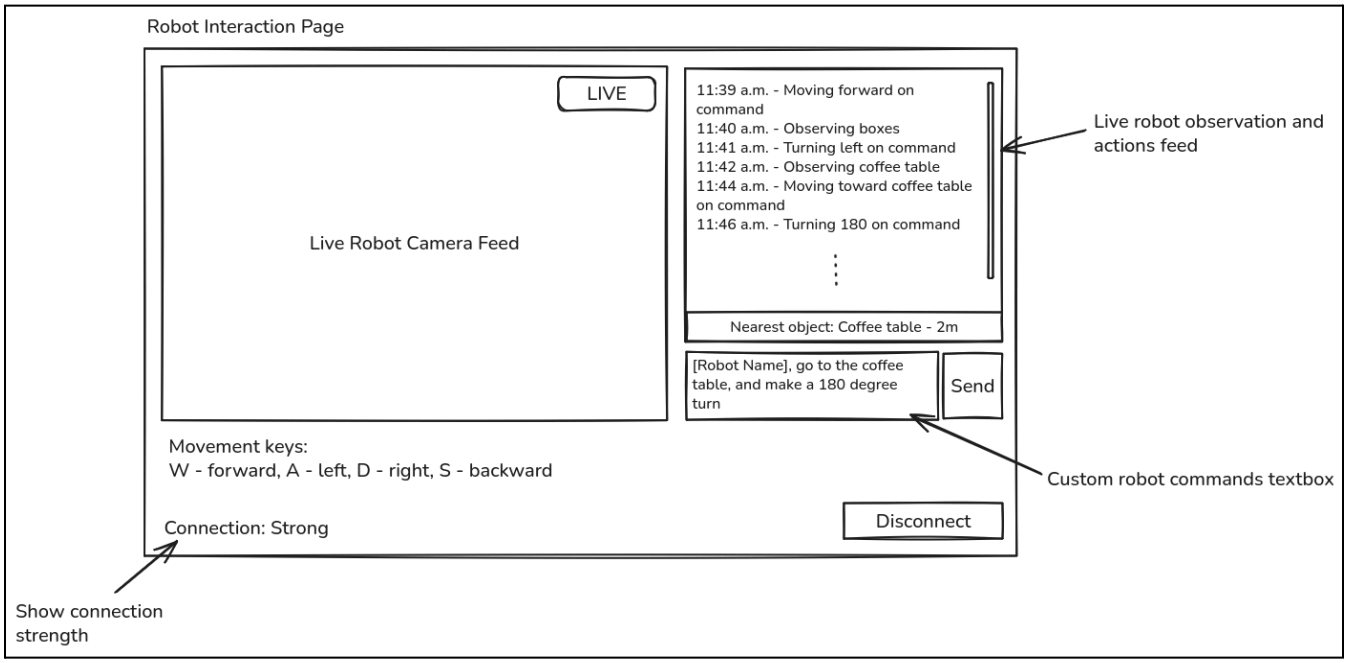
Voltage divider attached to the robot chassis

Since the addition of ultrasonic sensors added new connections to the raspberry pi, the wiring diagram was updated to reflect this change.



g) The website for remote controlling the robot

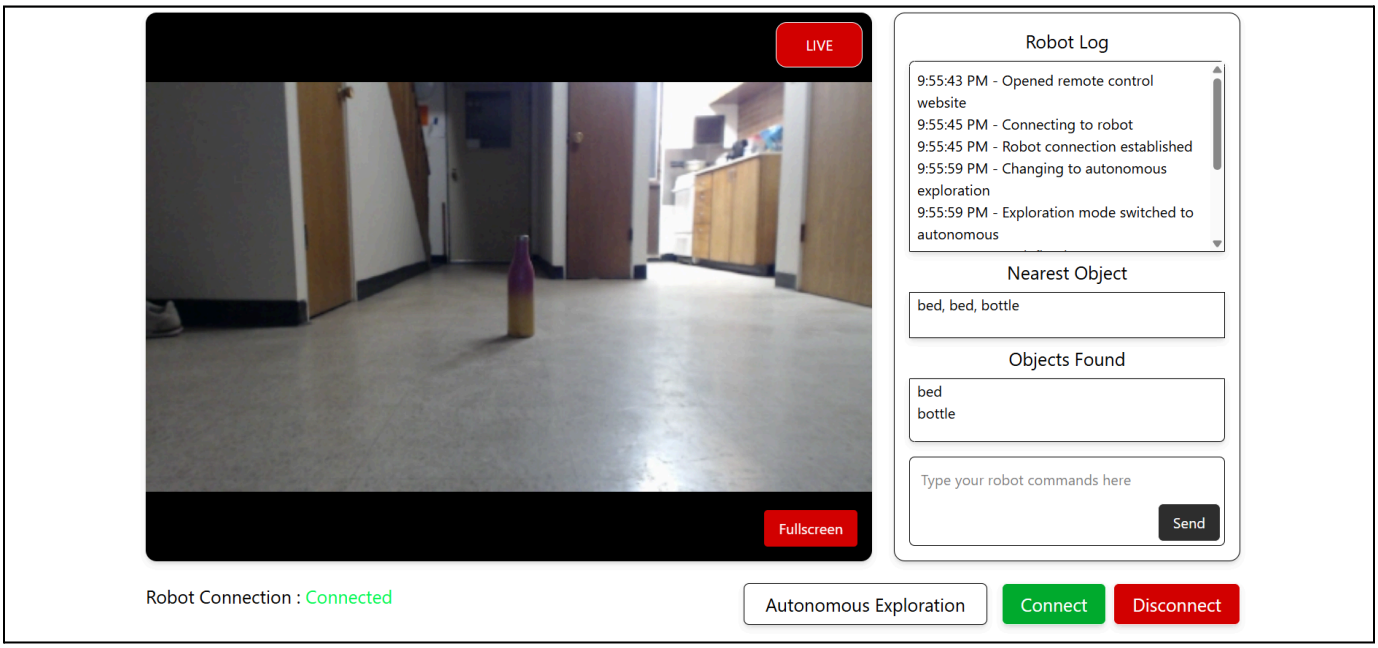
The team had an idea to build a website which gives the user more control over the robot. Currently, the robot can receive audio commands from the user and carry said commands, while also verbally communicating its thought processes to the user. However, they cannot manually control the robot, nor view what the robot is “seeing” either. Below is an initial screen sketch of how the website will look like.



The website will have a section for live camera view on the left, showing the user the perspective of the robot at all times. To the right of it, there will be a section for logging all information sent by the robot, a section for objects detected by the robot at that moment in time, and a section for the user to type and send commands to the robot. Currently there are a few commands that the user can send to the robot:

- Follow me - instructs the robot to follow a person once detected
- Find object - instructs the robot to explore autonomously around its environment and find the aforementioned object

At the bottom section, there will be an indication of connection strength or status of the website to the robot. There will also be a disconnect, and potentially connect button to the robot as well. Here is a picture of the development remote control website.



h) Issues with robot turning, weight distribution, fair navigation

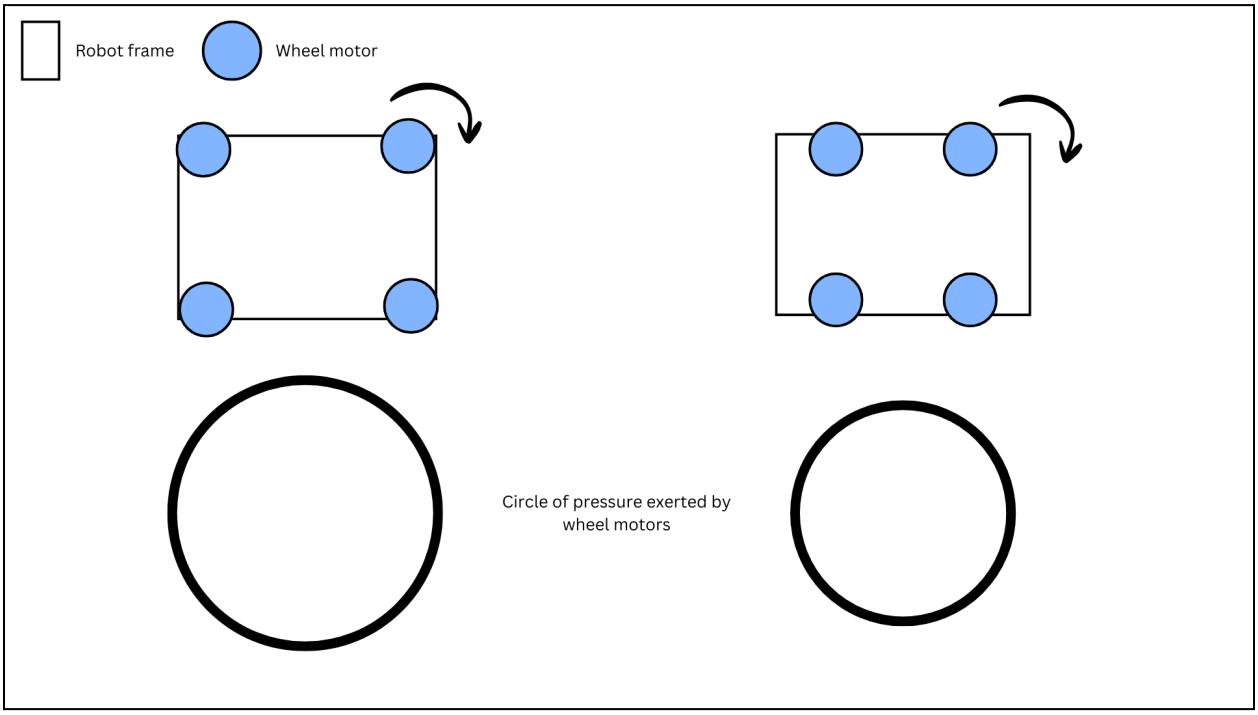
In terms of the robot's ability to navigate an unfamiliar environment, it had a number of issues:

- Should the robot explore its environment fairly, resulting in a gradual and radial exploration of its environment at the cost of appearing "stuck" due to excessive back tracking? Or should the robot explore its environment with bias towards one direction, resulting in a more "responsive" way of exploration at the cost of unfairness in exploration?

- Due to the weight of the robot and the strength of the wheel motors, turning left and right faces issues where the turning radius is not consistent and weak. As for moving forwards and backwards, these movement directions seem to not be affected.
- The weight of the robot is front heavy, and it would be hard to redesign the robot at this point of the developmental process.

As for the question of exploration, we decided to go with iterative deepening, but with a bias towards the front direction, then right, left, and backwards. Although this method of exploration is unfair, it was a better choice. When fair exploration was tested on the robot, it gave off the illusion that it is not working due to its constant backtracking.

To solve the issue of inconsistent and weak turning radiuses in conjunction with the weight distribution problem, we decided to figure out why the robot is struggling in terms of left/right movement, instead of front/reverse movement.

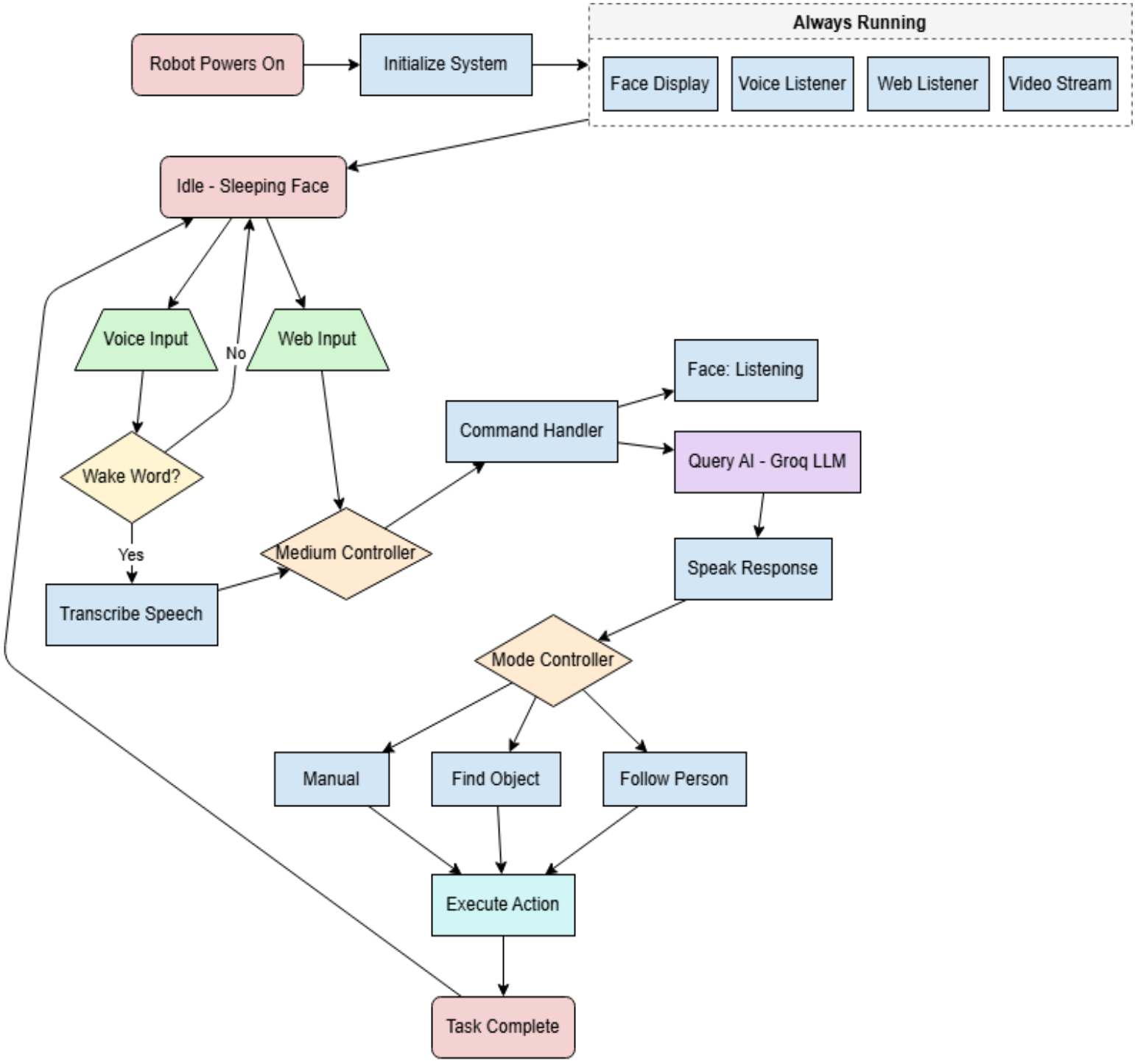


We theorized that the reason why the robot was struggling so much in terms of turning is due to the placement of the wheels. On the left is the wheel placement of the robot used in the prototype. Due to the rectangular shape of the robot frame, placing wheel motors at the edges of the frame results in a larger “circle of pressure” when we trace the wheel motors as we rotate the robot. We believe that the larger circle of pressure is too much for the wheel motors to handle, and that placing the wheel motors in a square shape would result in a smaller circle of pressure, relieving the burden on the motor wheels.

The new placement of motor wheels in a square manner did indeed help the robot to turn more effectively, but it was still a bit weak. And so, the next issue could potentially be weight distribution. As the robot is front heavy, we decided to move components situated at the back and middle to be together in the middle. We decided to move the powerbank and omnidirectional microphone more towards the middle of the robot frame, as they weigh significantly as compared to components. The end result is that the robot turns much more effectively now despite the weight of the components.

There was also the issue with the wheel motors experiencing a failure of sorts in terms of movement. There were on occasions (two so far) where one of the motor wheels would emit a large grinding noise and fail to turn properly. When opened for investigation, we discovered that a component responsible for turning the gears had been misaligned and needed to be adjusted back in place. We theorized that this may have happened because the gears are made of plastic, and may have misaligned from the vibration of movement. However, apart from applying some lubricant to the gears, there seems to be no solution to fixing the problem due to the materials used to design the motors.

i) Robot Software Workflow



CHAPTER 3: Work Done

Mani Raj Rejinthala

- Set-Up the working environment on raspberryPi by installing the Operating System and other dependencies.
- Developed the website backend using the WebRTC framework for remote monitoring
- Created the Socket based interface to facilitate the web clients and robot interactions for robot remote control and action log.
- Set up a local Mosquitto server on the Raspberry Pi to exchange initial signaling messages between the web client and the robot in order to establish a peer-to-peer connection.
- Implemented the Motor control, Obstacle prediction logic using real time sensor readings, and Random Autonomous Exploration.
- Implemented the camera interface to allow multiple programs simultaneously access the visual cues in real time.
- Orchestrated the full robot workflow using Asynchronous, Threads and Sub-Processes to make seamless robot actions and responses by making the programs concurrent.
- Implemented Mode controller to ensure only one major command run at a time and to allow interruptions or cancellations of requests that are ongoing.
- Implemented Medium Controller to ensure only one user user either web client or voice commander get control over robot resources to prevent the interleaving or uninterrupted actions.
- Developed find target until utilising the visual cues from yolov8, sensor reading, and motor control.
- Deployed the frontend and backend services on a Raspberry Pi, served through Nginx

Koushik Goud Shaganti

- Prototyped the robot chassis using cardboard, constructing the initial physical structure of the robot.
- Worked with the Mechanical Engineering student to refine and create the 3D CAD model of the robot chassis.
- Built the second chassis using cardboard reinforced with acrylic sheets, including components-holding parts for electronics.
- Designed and integrated robot facial expressions using OpenCV, enabling real-time emotional responses such as neutral, happy, sad, angry, and surprised.

- Implemented the Follow-Me feature using YOLOv8n, allowing the robot to detect and track a person in real time.
- Implemented object detection logic as part of the Follow-Me system to improve tracking accuracy and responsiveness.

Kenny Jia Hui Leong

- Researched and created required robot components lists (pricing, purchase site, specs)
- Ordered and gathered all components.
- Prototyped robot chassis with cardboard.
- Contacted ME team to 3D model robot chassis and Additive Manufacturing Lab of Black Engineering for quote on 3D printing.
- Developed a website frontend with a page for remote controlling the robot (manual movement, send command, live feed, manual/autonomous mode toggle).
- Built the second version of robot chassis (double-layered cardboard reinforced with acrylic sheets, held together with glue).
- Built voltage divider for ultrasonic sensors to function properly.
- Fine-tuned robot autonomous movement, sensor placement and code logic, general maintenance, cable management, RPi rewiring for newly added components.

Ethan Olderog

- Assisted Mani with setting up the Raspberry Pi working environment, including OS installation and Wi-Fi configuration.
- Implemented a fully local LLM using Tinyllama.
- Implemented initial object detection using YOLOv8n.
- Implemented TTS (text-to-speech) locally using faster-whisper
- Implemented STT (speech-to-text) locally using the piper model lessac-medium.
- Added logic rules to the system prompt to improve response consistency.
- Implemented humorous response behavior to make interactions more engaging with the LLM.
- Added optional offloaded compute to external APIs for LLM, TTS, STT for higher quality outputs and faster response times.
- Implemented listening and speaking robot emotion states.
- Implemented conversational logging so that the robot can remember past conversations to enable context-aware interactions.

CHAPTER 4: Results

All in all, we have successfully created an AI robot that is capable of autonomously exploring an unfamiliar environment, navigating accordingly while avoiding obstacles to reach the desired object. The robot is capable of receiving a verbal command, translating it from speech to text, determining the desired object to be found by the user, exploring its environment in search of said object, and eventually reaching and stopping in front of it.

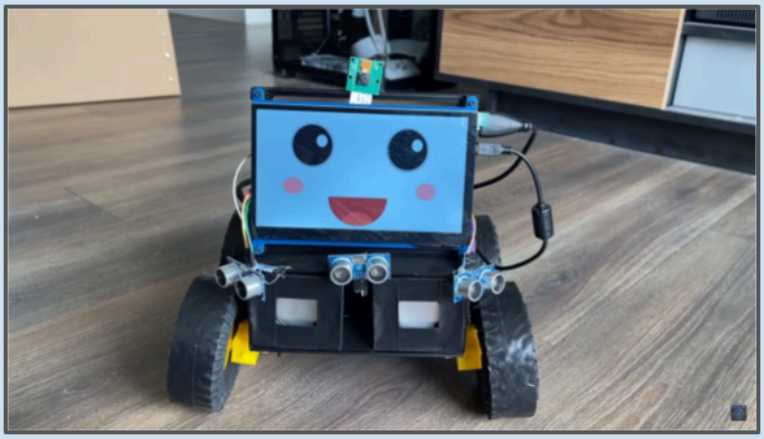
In addition to the main object finding feature of the robot, we have also added a website for remote controlling the robot, as well as a follow me feature for the robot to follow the first discovered user. The website gives the user manual control over the robot when needed, with an interface that provides a live view of what the robot is observing, what objects are detected, as well as a command bar for entering textual commands to the robot.

Of course, that is not to say that the robot is entirely perfect. The robot may still run into turning issues due to the nature of the wheel motors (made of mostly plastic) which may need occasional maintenance, the AI model may fail to detect objects if not under ideal lighting conditions, and the general flow of operations does take some amount of time which could definitely be improved. However, despite all of that, the team does agree that this project has given us the opportunity to learn a lot of new tools that we would have otherwise not even considered. The requirement to work with hardware, an unfamiliar topic, as well as the AI tools needed to process all the required information and respond accordingly, and the trials and errors necessary to integrate everything together has provided the team with an invaluable amount of experience.

CHAPTER 5: Appendix

Final presentation slides

Simple Robot



Mani Raj Rejinthala, Ethan Olderog, Koushik Shaganti, Kenny Jia Hui Leong

1

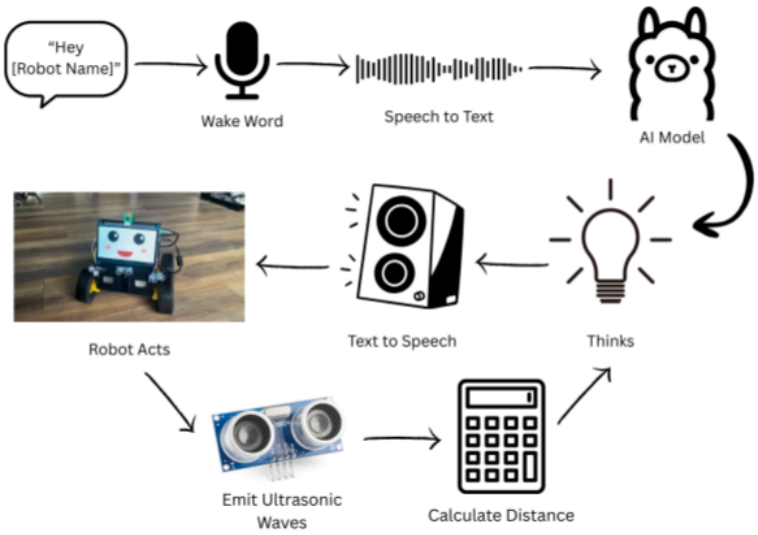
Recap

- Build an AI robot capable of autonomous exploration.

Software	Hardware
- RaspberryPi Linux OS - ML models: Tiny Llama, Coqui, Piper, Whisper, TFlite, Yolo - Python 3, Rpi.Gpio, OpenCV and Curses	- Raspberry pi 4 model B, L298N motor driver, HDMI display, omnidirectional microphone, speakers, pi camera, ultrasonic sensor, etc.

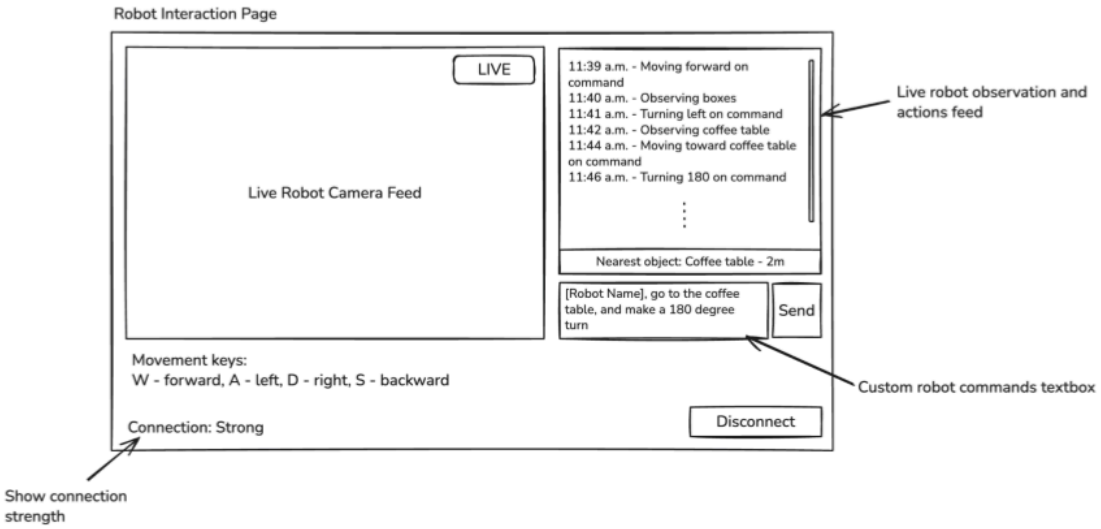
2

System Architecture



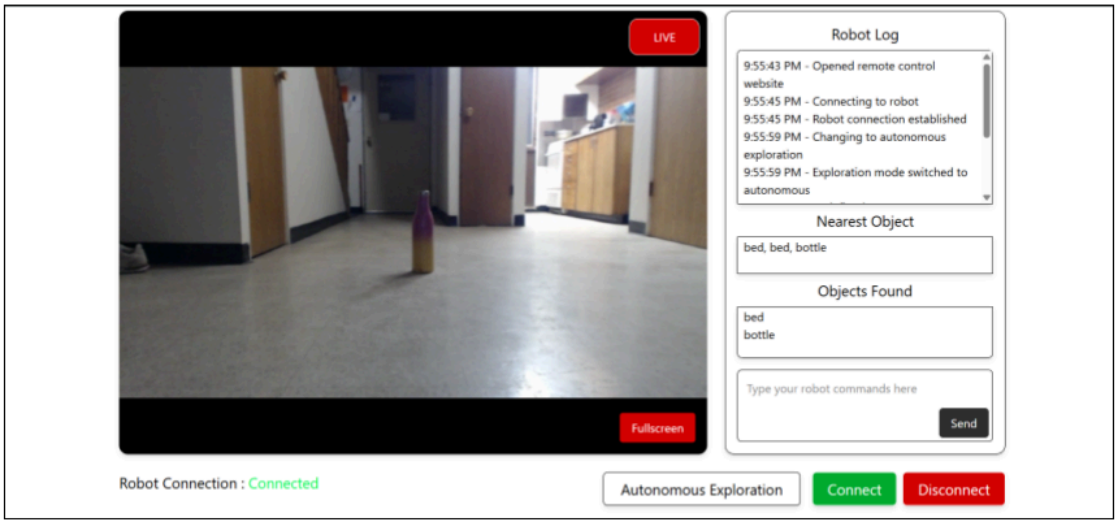
3

Website - Remote Controlling Robot



4

Website - Remote Controlling Robot



5

Website - Remote Controlling Robot

sendMessage():

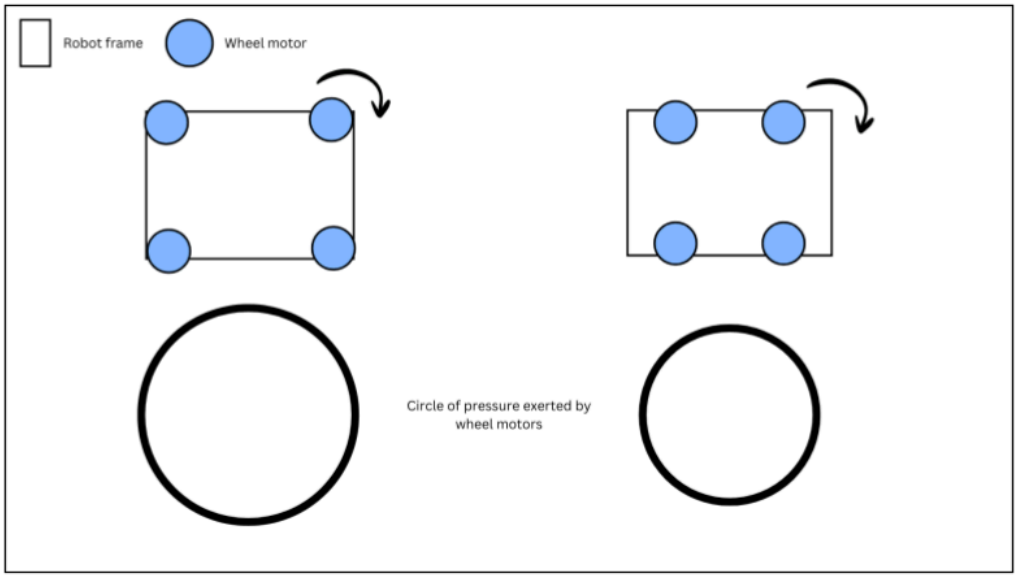
- {“type”: “motor”, “command”: “front”}
- {“type”: “motor”, “command”: “left”}
- {“type”: “motor”, “command”: “right”}
- {“type”: “motor”, “command”: “back”}
- {“type”: “find”, “command”: “<string>”}
- {“type”: “mode”, “command”: “autonomous”}
- {“type”: “mode”, “command”: “manual”}

onMessage():

- {“type”: “mode”, “command”: “manual”}
- {“type”: “mode”, “command”: “autonomous”}
- {“type”: “log”, “command”: “<string>”}

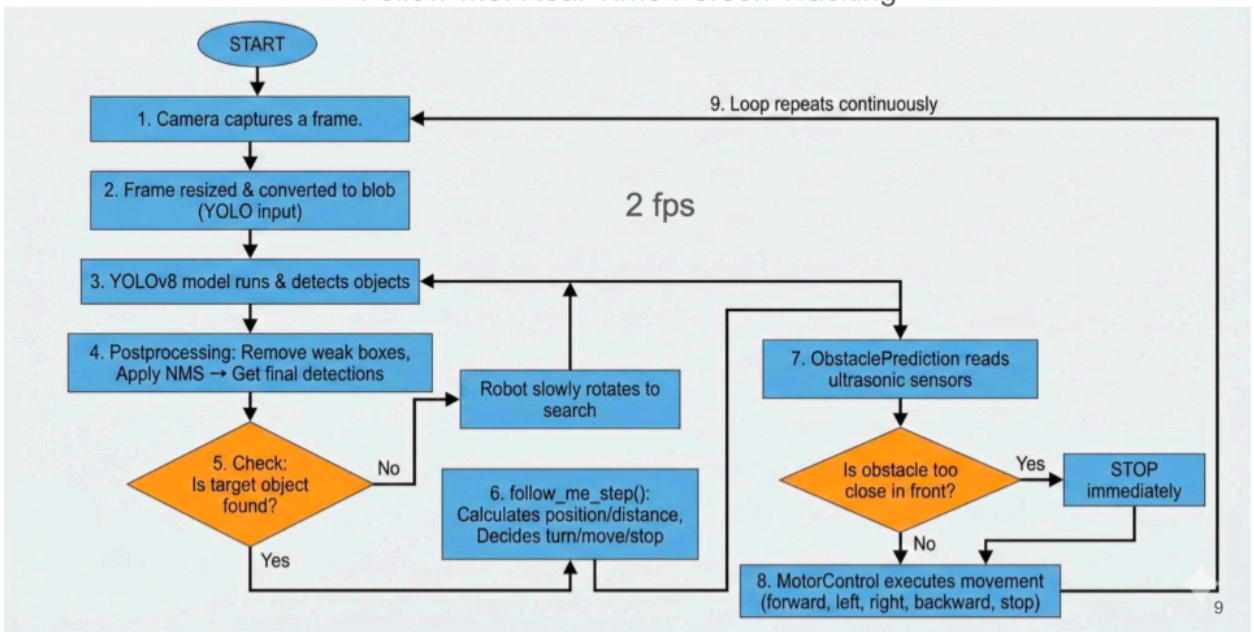
6

Robot Weight Distribution & Turning Effectiveness



7

Follow-Me: Real-Time Person Tracking



9

Output

```
Person CENTERED – adjusting distance
Moving FORWARD – distance: 61.45cm (target: 40cm)

0: 480x640 2 persons, 1 tie, 1050.3ms
Speed: 0.6ms preprocess, 1050.3ms inference, 7.4ms postprocess
Distances – Front: 56.51cm, Left: 49.72cm, Right: 60.65cm
Person detected: center | Front distance: 56.51cm
Person CENTERED – adjusting distance
Moving FORWARD – distance: 56.51cm (target: 40cm)

0: 480x640 2 persons, 1024.9ms
Speed: 0.6ms preprocess, 1024.9ms inference, 7.7ms postprocess
Distances – Front: 49.7cm, Left: 49.72cm, Right: 183.56cm
Person detected: center | Front distance: 49.7cm
Person CENTERED – adjusting distance
MAINTAINING POSITION – distance: 49.7cm (safe zone)
```

10

Model Selection

Speech-to-Text (STT)🎤 → 💬

Option	Model	Quality	Response Time	Cost
Cloud	OpenAI Whisper whisper-1	Very high, handles accents and background noise	1-3 seconds	Virtually free for our use.
Local	faster-whisper base.en	Fair, but slow	5-10 seconds	Free

Text-to-Speech (TTS) 💬→ 🎤

Option	Model	Quality	Response Time	Cost
Cloud	OpenAI TTS gpt-4o-mini-tts	Natural, human-like	1-2 seconds	Virtually free for our use.
Local	Piper lessac -medium	Clear but robotic, can't pronounce certain phrases	2-4 seconds	Free

Large-Language-Model (LLM) 🧠

Option	Model	Quality (params)	Response Time	Cost
Cloud	Groq llama-3.3-70b-versatile	70 Billion	< 1 second	Free
Local	Tinyllama	1.1 Billion	10 seconds	Free

10

System prompt - Features/Rules/Behavior

```
Response:
{
    "find": "<object>" or "",
    "follow": "true" or "",
    "emotion": "happy|sad|angry|neutral|surprised|confused",
    "response": "<spoken response>",
    "command": "right|left|back|front" or ""
}
```

FEATURES

- **Conversation Memory**: Remembers last 100 Q&A exchanges
- **Object Detection Integration**: Recognizes 80 YOLO/COCO objects
- **Message Identification**: If user says "I'm Ethan, what's my favorite color?", robot looks up Ethan's stored info.

BEHAVIOR

- **Personality**: Witty, playful, humorous. Clever and lighthearted.
- **Nonsense Input**: Set emotion to confused, funny clarification like "My robot brain did a somersault!"
- **Action Responses**: For find/follow/command requests, set the appropriate field and give a short, witty confirmation. Invalid objects get emotion sad with a humorous apology.

RULES

- **Mutually Exclusive Actions**: Only ONE per response, find OR follow OR command.
- **JSON-Only Output**: Must respond with ONLY a valid JSON object, no other text.
- **Short Responses**: 1-2 sentences max.
- **Required Fields**: Every response must include emotion and response.

11

Autonomous Target Tracking

- Robot turns in 4 directions at each step
- Moves towards detected target if found
- If not, takes random obstacle-free step
- Uses YOLO v8 visual cues + sensors + motor control
  - Why did not we use standard exploration algorithms?

Unified Camera Interface for Multi-Program Access

- WebRTC client requires exclusive camera control
- Camera interface captures physical frames continuously
- Frames fed to virtual camera (v4l2loopback + ffmpeg) and object detection
- Multiple programs can access camera simultaneously

Socket.io communication

- Interaction between web client and robot
- Low latency
- JSON formatted messages

Challenge:

- Concatenated two immediate consecutive messages
- Affected the structured requests

Seamless Robot Actions with Concurrency

- Programs run asynchronously: threads, async tasks, subprocesses
- voice listener, web listener, video streaming, logging, socket communication

Safe Resource Management

- Mode Controller: only one major command at a time; supports safe interruptions
- Medium Controller: only one user (web or voice) controls robot resources
- Prevents conflicts and unintended actions

Deployment on Raspberry Pi

- Frontend + backend served via Nginx
- Enables remote monitoring and control

Team Contributions

Mani Raj Rejinthala

- Set-Up the working environment on raspberryPi by installing the Operating System and other dependencies.
- Developed the website backend using the WebRTC framework for remote monitoring
- Created the Socket based interface to facilitate the web clients and robot interactions for robot remote control and action log.
- Set up a local Mosquitto server on the Raspberry Pi to exchange initial signaling messages between the web client and the robot in order to establish a peer-to-peer connection.
- Implemented the Motor control, Obstacle prediction logic using real time sensor readings, and Random Autonomous Exploration.
- Implemented the camera interface to allow multiple programs simultaneously access the visual cues in real time.
- Orchestrated the full robot workflow using Asynchronous, Threads and Sub-Processes to make seamless robot actions and responses by making the programs concurrent.
- Implemented Mode controller to ensure only one major command run at a time and to allow interruptions or cancellations of requests that are ongoing.
- Implemented Medium Controller to ensure only one user user either web client or voice commander get control over robot resources to prevent the interleaving or uninterrupted actions.
- Developed find target until utilising the visual cues from yolov8, sensor reading, and motor control.
- Deployed the frontend and backend services on a Raspberry Pi, served through Nginx

Kenny Jia Hui Leong

- Researched and created required robot components lists (pricing, purchase site, specs)
- Ordered and gathered all components.
- Prototyped robot chassis with cardboard.
- Contact ME team to 3D model robot chassis and Additive Manufacturing Lab of Black Engineering for quote on 3D printing.
- Developed a website frontend with a page for remote controlling the robot (manual movement, send command, live feed, manual/autonomous mode toggle).
- Built the second version of robot chassis (double-layered cardboard reinforced with acrylic sheets, held together with glue).
- Built voltage divider for ultrasonic sensors to function properly.
- Fine-tuned robot autonomous movement, sensor placement and code logic, general maintenance, cable management, RPi rewiring for newly added components.

18

Team Contributions (continued)

Koushik Goud Shaganti
- Prototyped the robot chassis using cardboard, constructing the initial physical structure of the robot.
- Worked with the Mechanical Engineering student to refine and create the 3D CAD model of the robot chassis.
- Built the second chassis using cardboard reinforced with acrylic sheets, including components-holding parts for electronics.
- Designed and integrated robot facial expressions using OpenCV, enabling real-time emotional responses such as neutral, happy, sad, angry, and surprised.
- Implemented the Follow-Me feature using YOLOv8n, allowing the robot to detect and track a person in real time.
- Implemented object detection logic as part of the Follow-Me system to improve tracking accuracy and responsiveness.

Ethan Olderog
- Assisted Mani with setting up the Raspberry Pi working environment, including OS installation and Wi-Fi configuration.
- Implemented a fully local LLM using Tinyllama.
- Implemented initial object detection using YOLOv8n.
- Implemented TTS (text-to-speech) locally using faster-whisper
- Implemented STT (speech-to-text) locally using the piper model lessac-medium.
- Added logic rules to the system prompt to improve response consistency.
- Implemented humorous response behavior to make interactions more engaging with the LLM.
- Added optional offloaded compute to external APIs for LLM, TTS, STT for higher quality outputs and faster response times.
- Implemented listening and speaking robot emotion states.
- Implemented conversational logging so that the robot can remember past conversations to enable context-aware interactions.

19

Final demo video

