<u>Seneca College of Applied Arts and Technology</u>
<u>TPJ655</u>

<u>Technical Report</u>

AI Mobile Robot

Submitted by: Saket Chahal and Hemant Vohra

Professor: Karen Craigs

Course Section: NBBL/NAB

Term: Fall 2025

Date Submitted: November 21st, 2025

§02: Abstract

Our project was to design and build an AI-powered mobile robot that combines large language model (LLM) conversation, real-time sensing, and IoT monitoring on a single Raspberry Pi platform. The goal was to move beyond a stationary smart speaker and create a robot that can move around, “see” its environment, express emotions on an OLED display, and report its status through a browser-based dashboard. To do this, we designed a custom PCB for embedding multiple HC-SR04 ultrasonic sensors and used a motor driver expansion board for four DC motors and a pan/tilt servo mechanism. We also integrated a camera, microphone, and speaker for vision and voice interaction.

On the software side, we used ROS 2 nodes for movement, sonar, camera, and IMU filtering, and connected them to a web dashboard and for IoT monitoring so that all key values (distance, battery, logs, and commands) are visible in real time. The finished prototype can drive on a mecanum-wheel chassis, avoid obstacles, respond to voice prompts with human like answers, and display matching facial expressions on the OLED. Testing showed that the ultrasonic distance readings and movement commands are reliable enough for classroom demos and indoor kiosk scenarios. Overall, the project demonstrates that a low-cost, mobile, and conversational robot can be built with off-the-shelf parts and open-source tools while still leaving room for future improvements and commercialization.

§03: Table of Contents

Table of Contents

§03: Table of Figures

§04: Introduction

Advances in robotics and artificial intelligence have opened new possibilities for creating intelligent systems capable of meaningful human interaction. While traditional robots have excelled in industrial settings, the potential of socially interactive and service-oriented robots is still untapped, and their demand has only grown over the years across healthcare, education, and customer service sectors. Despite this growing interest, many existing solutions remain either prohibitively expensive for widespread adoption or limited in their functional capabilities, often lacking the integration of modern AI technologies. This integration can make can go a long way in making these programmable machines into useful products on a mass scale.

This project presents the development of an AI-powered mobile robot built on the Raspberry Pi5 platform (Raspberry Pi Foundation, n.d.), designed to bridge the gap between IOT devices and LLMs. The system integrates multiple technologies including real-time computer vision through a 12-megapixel camera module with autofocus capabilities (Raspberry Pi Foundation, 2024), GPT-based conversational AI for human-like speech interaction, facial expression on OLED display using the SSD1306 controller (Solomon Systech Limited, 2008), and comprehensive IoT monitoring through a Node-RED dashboard. Unlike conventional smart assistants that remain stationary, this design emphasizes mobility and environmental awareness, enabling the robot to navigate autonomously using HC-SR04 ultrasonic sensors (Morgan, 2014) for collision detection, and to provide transparent system monitoring.

The project draws inspiration from pioneering work in social robotics, including commercial platforms like the Emo Robot (Living.AI, n.d.) and research prototypes such as MIT's Kismet social robot (MIT Media Lab, n.d.), while distinguishing itself through the integration of affordable hardware components and open-source platforms. By combining DC gearbox motors for mobility (Adafruit Industries,018),ultrasonic sensors for collision avoidance, servo-controlled camera movement using the SG90 micro servo motor (PiShop, n.d.), and modern large language models for conversation, the system demonstrates how interactive robotics can be both cost-effective and educationally valuable. The dual motor control is managed through a motor driver expansion board, which efficiently controls four independent DC motors while drawing minimal power from the Raspberry Pi's GPIO pins.

The primary objective extends beyond simply constructing a functional robot. This project explores the meaningful integration of mobility, artificial intelligence, and IOTs (Internet of Things) devices to demonstrate the potential of next-generation service robots. Through this work, we aim to illustrate how readily available AI models can be embodied in physical systems to deliver engaging, human-like interactions that bridge the gap between software-based virtual assistants and hardware-enabled robotic platforms like kiosks. The robot's potential applications span HR assistance, reception services, and interactive learning environments.

§5: Background Research & Development

When developing our **AI-powered mobile robot**, we explored similar projects from both industry and DIY communities. These examples helped us identify best practices and differentiate our design.

Emo Robot by Living.AI

The **Emo Robot** is a commercially available desktop companion robot that integrates artificial intelligence for emotional interaction. It features **facial recognition, voice response, and OLED-based expressive eyes**.

Similarities:

- Like Emo, our robot uses an OLED display for expressions, voice interaction, and a camera for real-time engagement.

Differences:

- Emo is stationary, while our design includes mobility via DC motors and IoT dashboard monitoring. Emo relies heavily on cloud-based AI, while our build prioritizes local processing on Raspberry Pi 4.

Figure 1: EMO Robot

Note. From “Emo Robot” [Photograph], by Living.AI, n.d., Living.AI (https://living.ai). Copyright Living.AI.

MIT “Kismet” Social Robot

Kismet was an experimental research robot created at MIT to study **social and emotional interaction between humans and machines**. It featured **servo-driven facial expressions, speech synthesis, and recognition of vocal tone**.

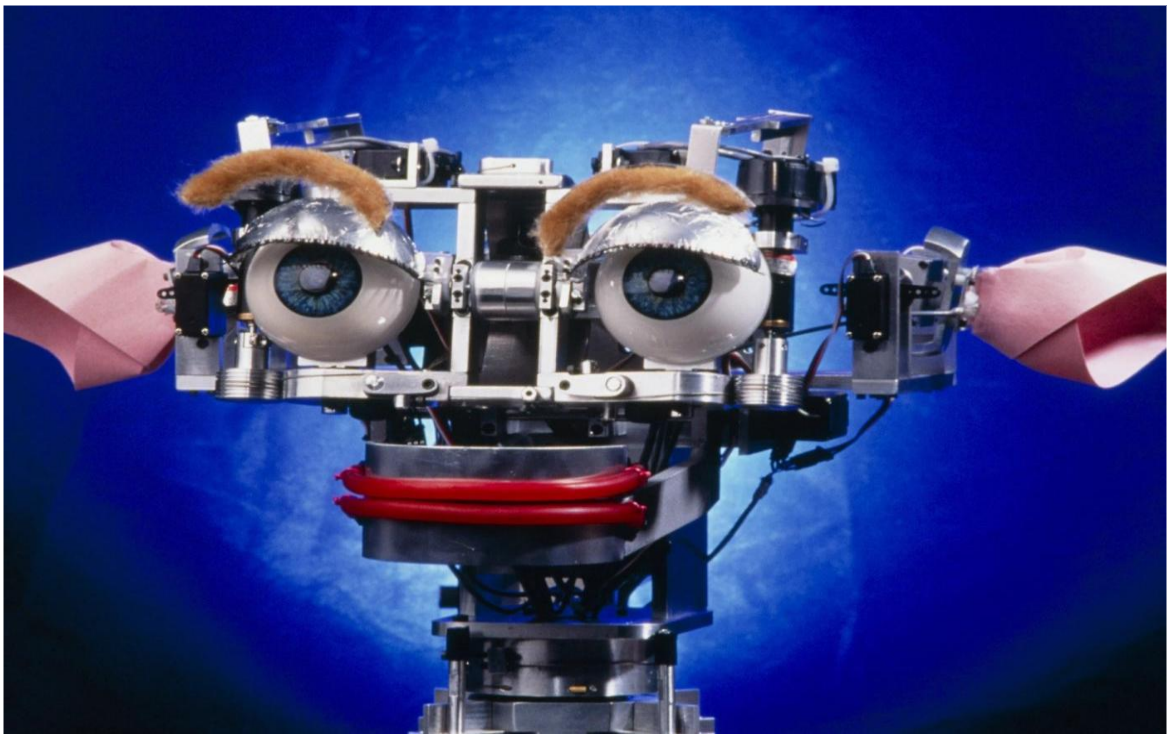
Similarities:

- Both Kismet and our design emphasize **human-robot interaction**
  Through expressions and speech.

Differences:

- Kismet was purely a research prototype, not mobile, and relied on older hardware. Our design integrates **modern AI (GPT-based responses)** and real-time mobility for

Figure 2: KISMET Robot



practical deployment.

Note. From “Kismet” [Photograph], by Robots Guide, n.d., Robots Guide (https://robotsguide.com/robots/kismet). Copyright Robots Guide

TARS Robot Replica

Finally, we reviewed the TARS Robot Replica developed by the Raspberry Pi community (see Figure 3). This DIY project recreates the famous Interstellar robot using a Raspberry Pi for control, sensors for navigation, and a mobile chassis (Raspberry Pi, 2023).

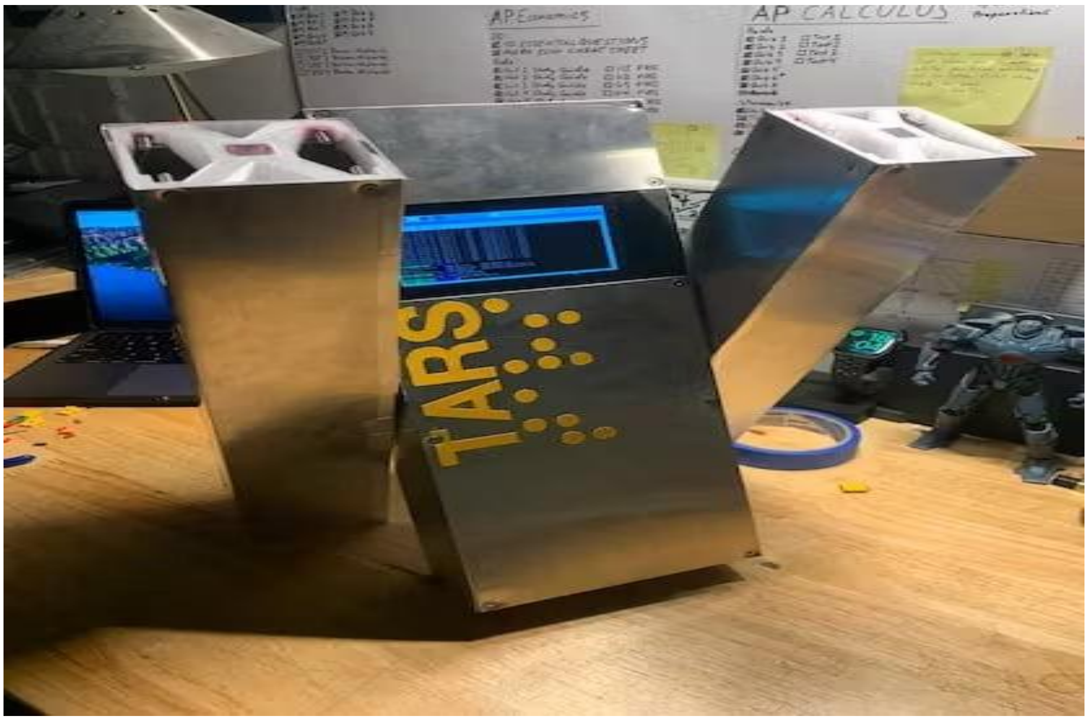
Similarities:

- **Raspberry Pi Based:** Both projects use the Raspberry Pi as the main controller for sensors, motors, and interaction.
- **Mobile Platform:** TARS and our robot both feature a **moving chassis** powered by DC motors and a motor driver.

Differences:

- **AI Features:** TARS mainly focuses on **movement and form factor**, while our robot integrates **GPT-based conversation, real-time vision, and expressive OLED feedback**.
- **Dashboard Integration:** TARS do not include IoT or dashboard logging, whereas our project adds a **Node-RED IoT dashboard** for monitoring robot activity and system health.
- **Interaction Style:** TARS relies on **basic movement and mechanical replication**, while our robot combines **voice interaction, OLED expressions, and camera-based awareness** to enhance engagemenT

Figure 3: TARS Robot



Note. From "Go interstellar and build your own replica TARS robot," by Raspberry Pi, 2023, Raspberry Pi (https://www.raspberrypi.com/news/go-interstellar-and-build-your-own-replica-tars-robot/). Copyright © Raspberry Pi.

Together, these projects influenced our development in three ways: **Emo Robot** guided our design of personality and expression, **Kismet** highlighted the importance of social interaction, and **TARS** validated the Raspberry Pi as a suitable platform for building a mobile, intelligent robot. By combining these insights, our project stands out as a unique integration of mobility, expressive feedback, AI-driven conversation, and IoT connectivity.

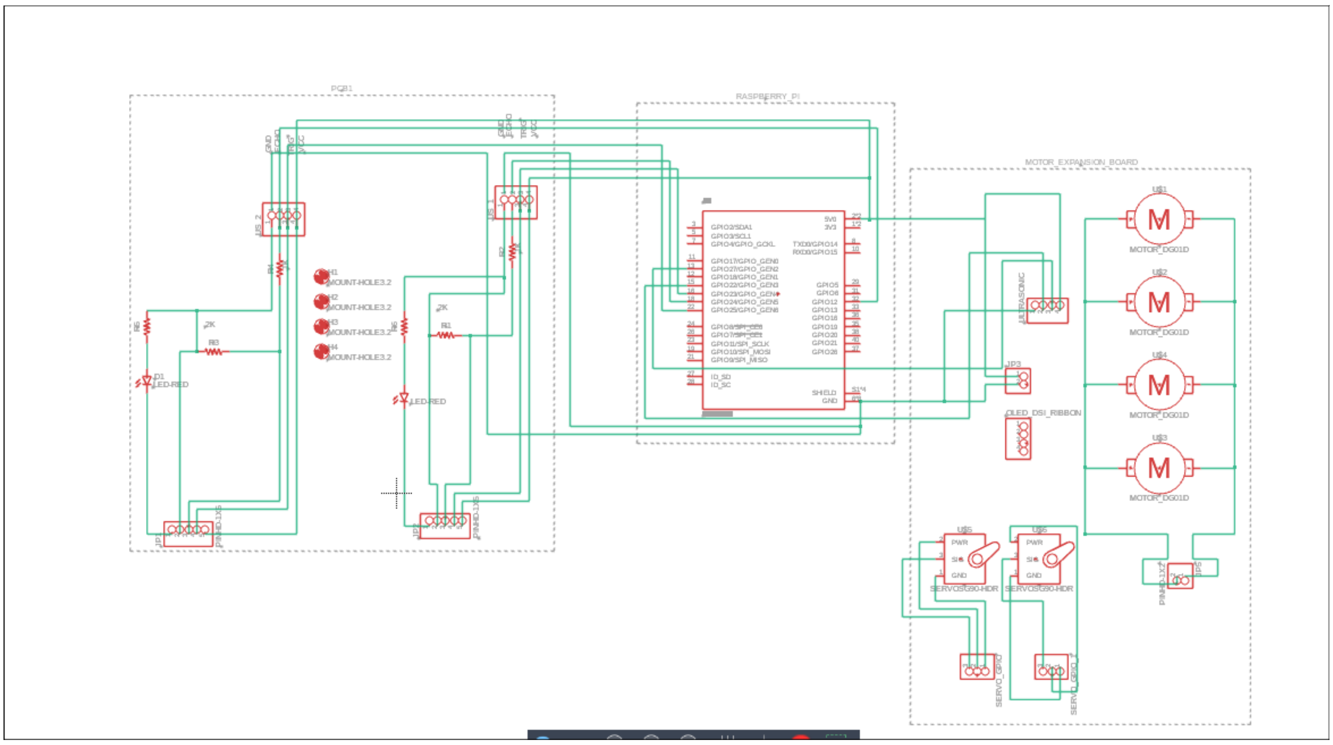
Development

Insights from Emu, Kismet and TARS guided the development of our project. From Emu, we have learnt the OLED display for face expressions. Kismet demonstrated the importance of social engagement and emotional response in a robot leading to Human-Robotic personality and TARS operates on Raspberry Pi as a low-cost mobile controlled device. Together all these projects have provided a foundation that influenced our project.

Our project includes an OLED display for facial expressions but unlike EMU, our project is movable because of a 4-wheel chassis design while emu is only a stationary device for desktop use only. Our project integrates real time live emotional interactions, and it uses AI for live interactions, camera-based vision and an IoT dashboard. Unlike TARS, which focused mainly on mechanical replication, our design integrates voice interaction, expressive feedback, and logged performance data to expand functionality.

§6: System Functional Features & Specifications

Figure 4: AI Robot circuit diagram



The AI Robot's entire circuit system is organized into three sections. These components work together to create a fully operational AI Robot.

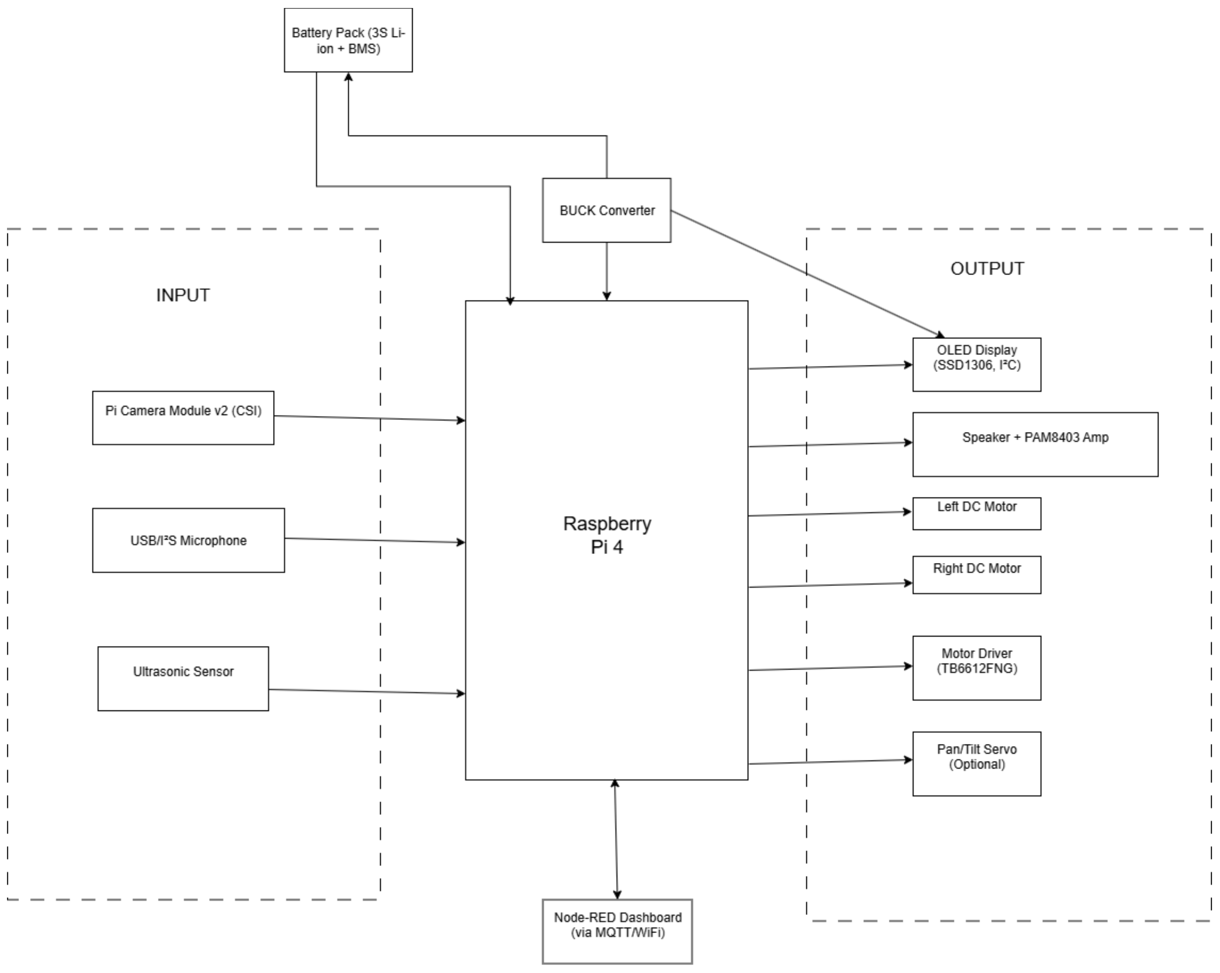
The three components are:

1. Raspberry Pi 5: All signal connections from Custom build PCB. Motor Expansion Board
2. Motor Expansion Board: Main Power Unit, Contains 4 Motors, 2servos and OLED Connections
3. Custom designed PCB: Integrated with Ultrasonic sensors along with indicator LEDs and voltage regulator resistors.

§07: System Design and Layout

This will be added in later phases. We are currently adding the SBD diagram for the project for reference.

Figure 5: System Design Layout



§08. Hardware Components & Specifications

Raspberry Pi 5:

Processor: Broadcom BCM2712, Quad-Core ARM Cortex-A76 @ 2.4 GHz

RAM Options: 4 GB / 8 GB / 16 GB LPDDR4X

USB Ports: 2 × USB 3.0 (higher bandwidth via RP1), 2 × USB 2.0

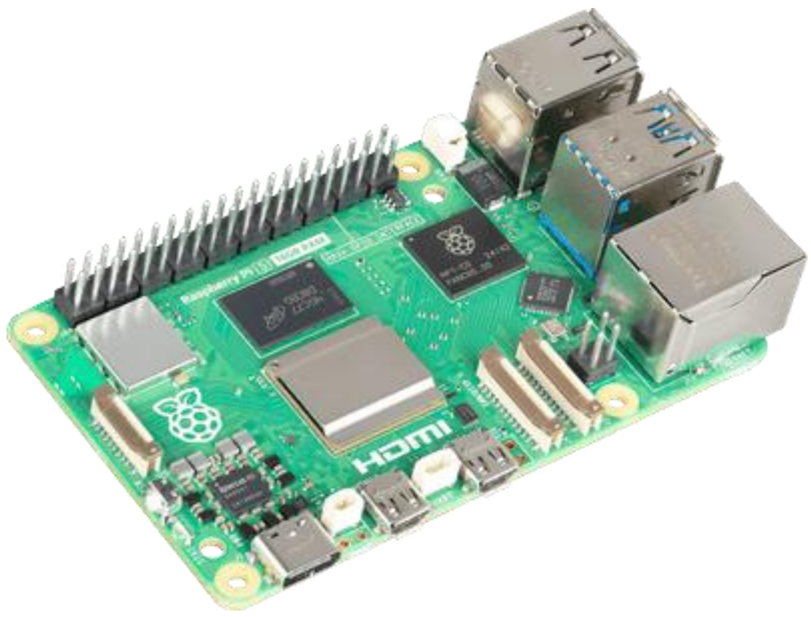
Display/Camera: 2 × interchangeable CSI/DSI ports (any combo: 1+1 or 2+0)

PCIe Support: 1 × PCIe 2.0 ×1 (supports M.2 SSD via adapter)

Networking: Gigabit Ethernet; Wi-Fi 2.4/5 GHz (802.11ac); Bluetooth 5.0/BLE

Power Supply: 5V DC via USB-C (recommended 5V 5A)

Figure 6: Raspberry Pi 5



Note. *From* Raspberry Pi 5*, by Raspberry Pi Foundation, (n.d.), Raspberry Pi (https://www.raspberrypi.com). Copyright Raspberry Pi Foundation.*

OLED DISPLAY

Display Size: 7 inches

Resolution: 800 × 480 pixels (IPS Panel)

Interface: MIPI DSI + USB (for touch input)

Operating Voltage: 5 V (via Raspberry Pi’s 5V rail)

Power Consumption: ≈ 3–5 W depending on brightness

Touch Type: 5-point capacitive touch

Compatibility: Raspberry Pi 5, 4B, 3B+, 3B, A+

Figure 7: OLED Display Freenova

Freenove. (n.d.). Freenove 7-inch Touch Display for Raspberry Pi 5/4B/3B+/A+: 800×480 IPS monitor with 5-point capacitive touchscreen (Model FNK0078H). Freenove Store. https://store.freenove.com/products/fnk0078h

Raspberry Pi Camera module 3

Sensor: Sony IMX708, 12 MP

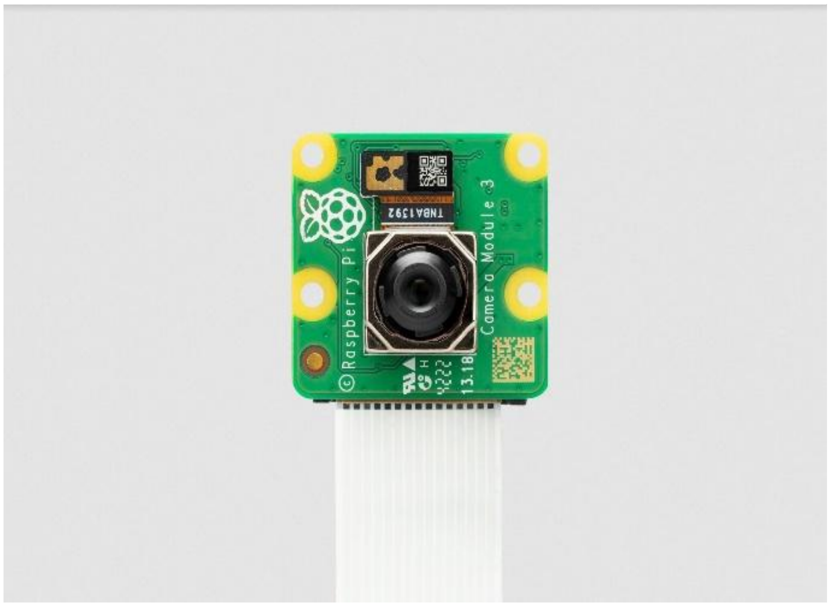
Resolution: 4608 × 2592 pixels

Lens: Standard and wide options

Autofocus: Supported

Interface: CSI ribbon cable

Figure 8: Pi Camera



Note. From Camera Module 3, by Raspberry Pi Foundation, (n.d.), Raspberry Pi (https://www.raspberrypi.com/products/camera-module-3/). Copyright Raspberry Pi Foundation.

Motor Driver (TB6612FNG)

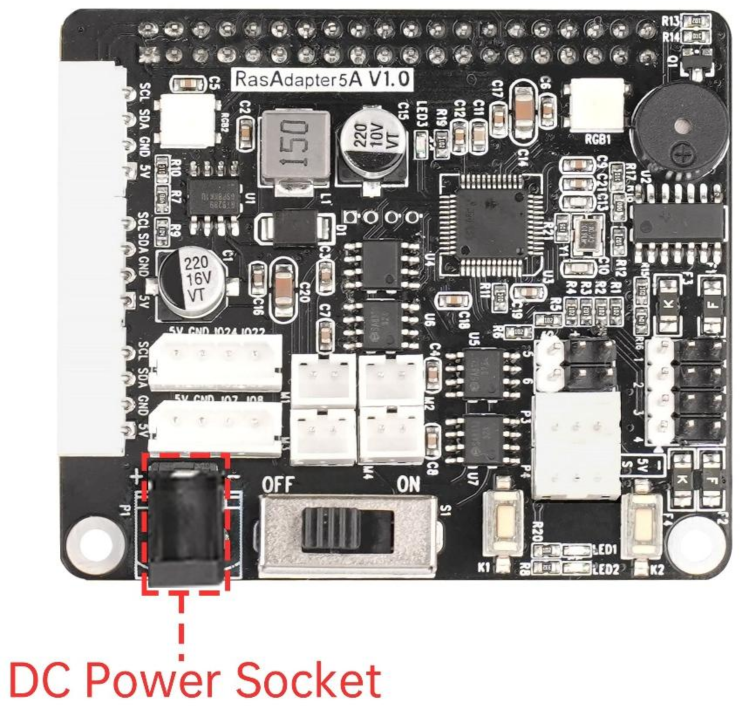
Raspberry Pi Compatibility: Works with both Raspberry Pi 5 models; no flat cables needed

MCU Processor: Cortex-M3 32-bit ARM chip

Integrated with built in female headers for Trig, Echo, VCC and Gnd signals.

Inbuild Power input(Labelled as DC POWER Socket in the figure below. Also, Powers raspberry pi and PCB)

Figure 9: Motor Expansion Board from HI wonder



Hiwonder. (n.d.). Expansion board for Raspberry Pi 5 [Product image]. Hiwonder. https://www.hiwonder.com/products/expansion-board-for-raspberry-pi-5

DC Gear Motors with Wheels (x4)

Operating Voltage: 3–6 V DC

No-Load Speed: ~200 RPM at 6 V

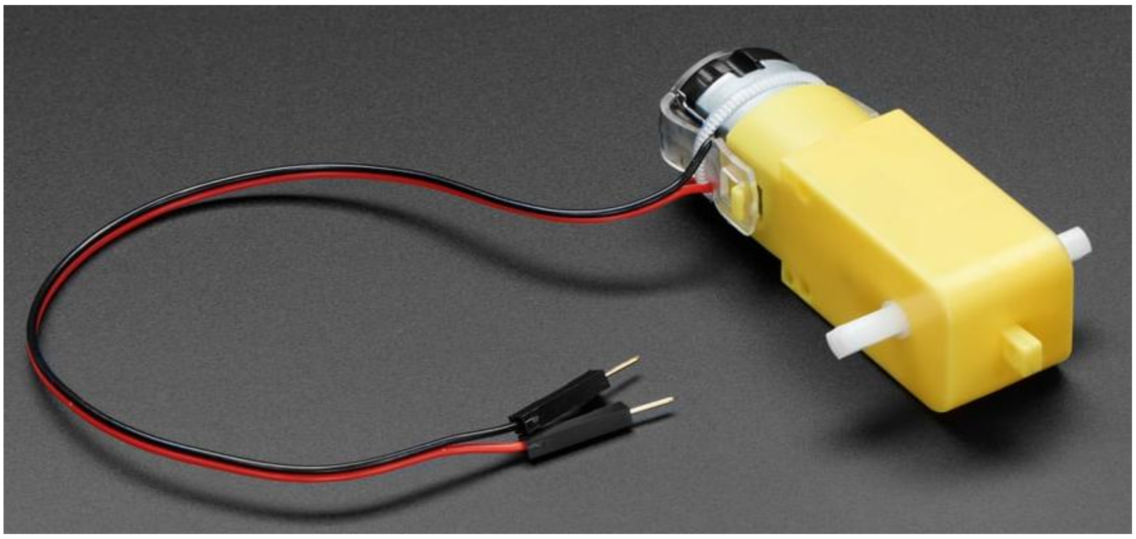
Stall Current: ~1 A

Torque: ~1.2 kg·cm

Shaft Type: 3 mm D-shaft

Dimensions: 70 × 22 × 18 mm (approx.)

Figure 10:TT DC Gearbox Motors



Note. From *DC Gearbox Motor – "TT Motor" – 200 RPM – 3 to 6 VDC*, by Adafruit Industries, (n.d.), Adafruit (https://www.adafruit.com/product/3777). Copyright Adafruit Industries.

Servo Motor (SG90 Pan/Tilt)

Operating Voltage: 4.8–6 V

Stall Torque: 1.8 kg·cm, 4.8 V

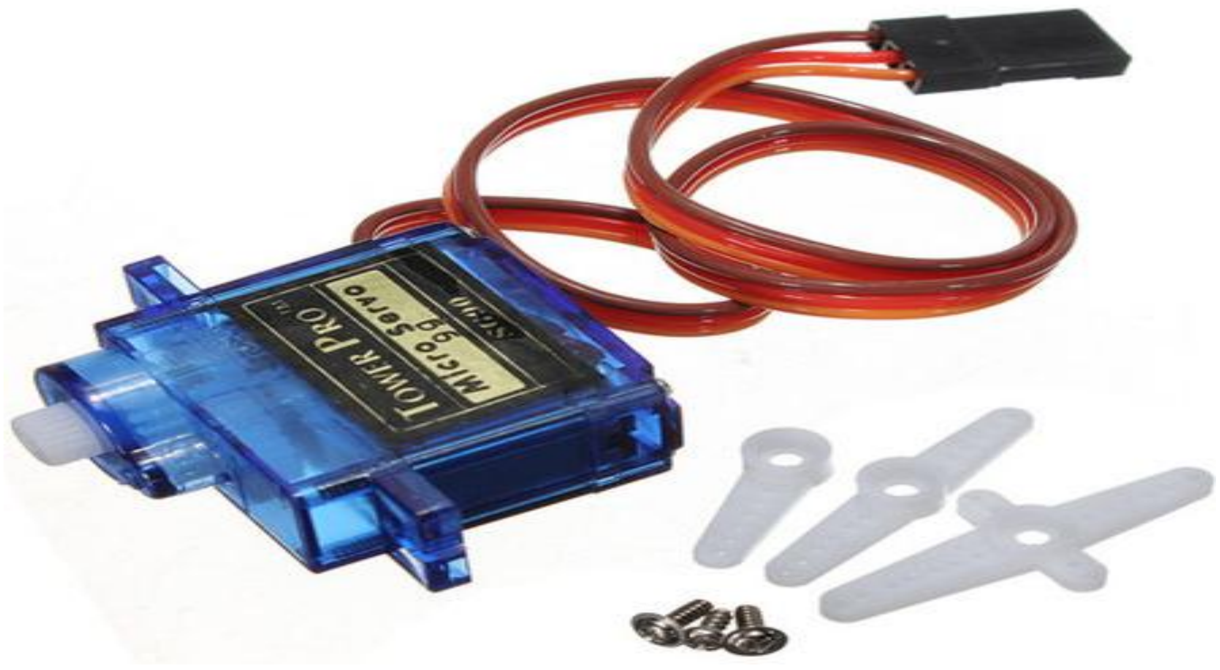
Speed: 0.1 s/60°, 4.8 V

Control Signal: PWM

Rotation Angle: ~180°

Weight: ~9 g

Figure 11: Servo Motor



Note. *From* SG90 180 Degrees 9g Micro Servo Motor Tower Pro*, by PiShop, (n.d.), PiShop (https://www.pishop.ca/product/sg90-180-degrees-9g-micro-servo-motor-tower-pro/). Copyright PiShop.*

Ultrasonic Sensor (HC-SR04)

Operating Voltage: 5 V DC

Operating Current: < 15 mA

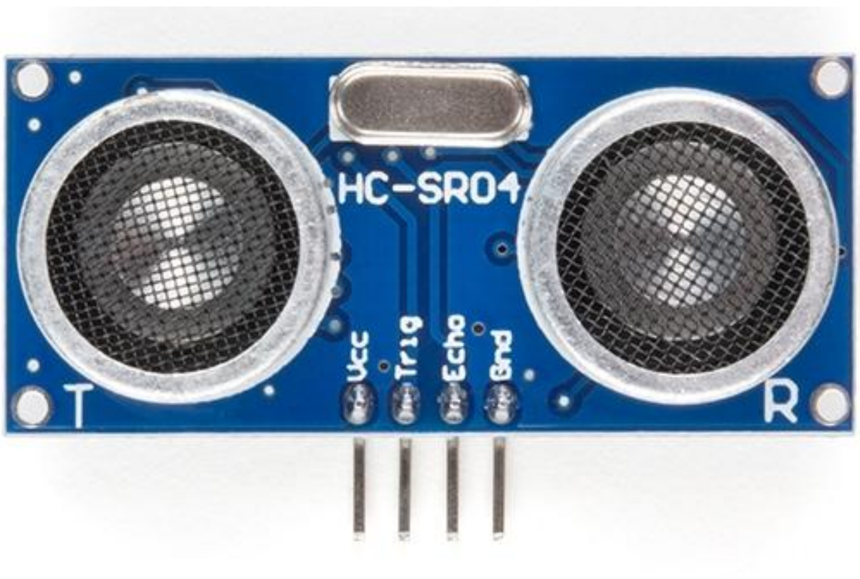
Range: 2 cm – 400 cm

Resolution: ~3 mm

Interface: Trigger (input), Echo (output)

Frequency: *40 kHz*

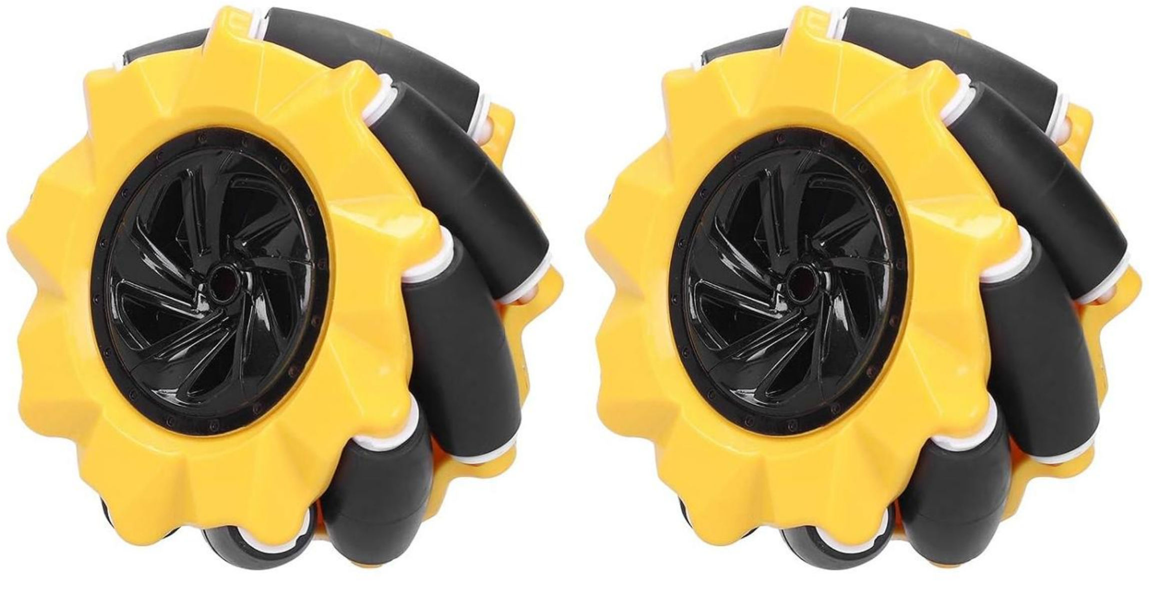
Figure 12: Ultra Sonic Sensor



Note. From *Ultrasonic Distance Sensor – 5V (HC-SR04)*, by SparkFun Electronics, (n.d.), SparkFun (https://www.sparkfun.com/ultrasonic-distance-sensor-hc-sr04.html). Copyright SparkFun Electronics

Mecanum Wheel Omni Directional

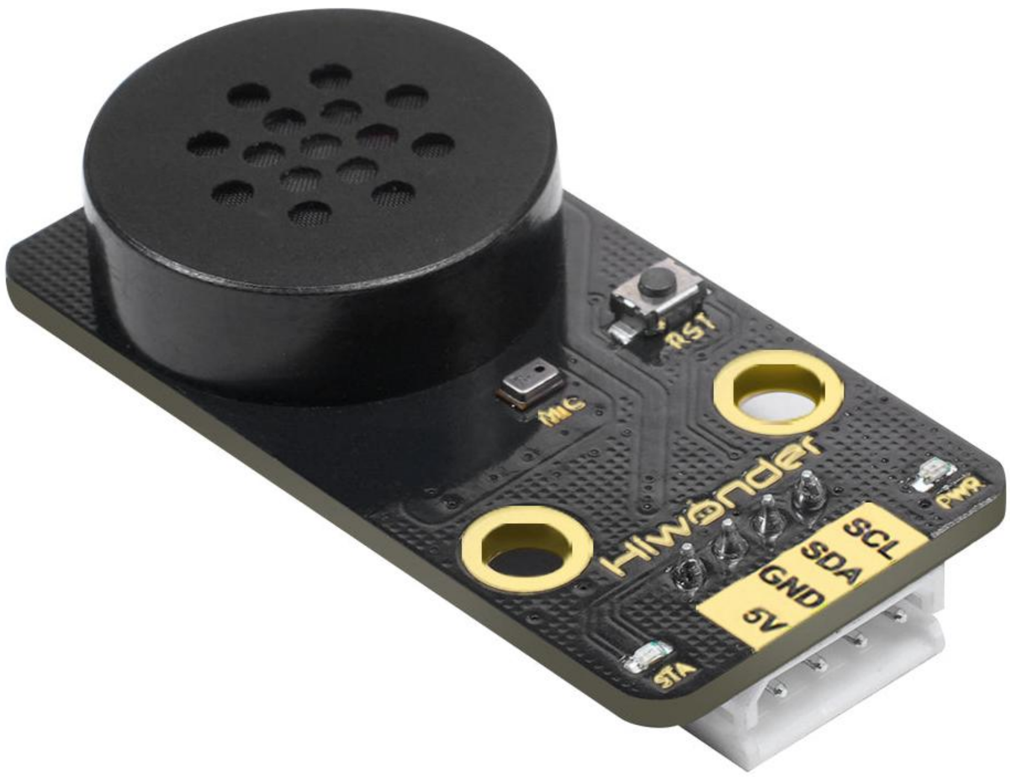
Figure 13: USB Microphone



Mecanum Wheel Omni Directional. (n.d.). Amazon product page. Amazon. https://www.amazon.ca/dp/B0DSLWMBKS

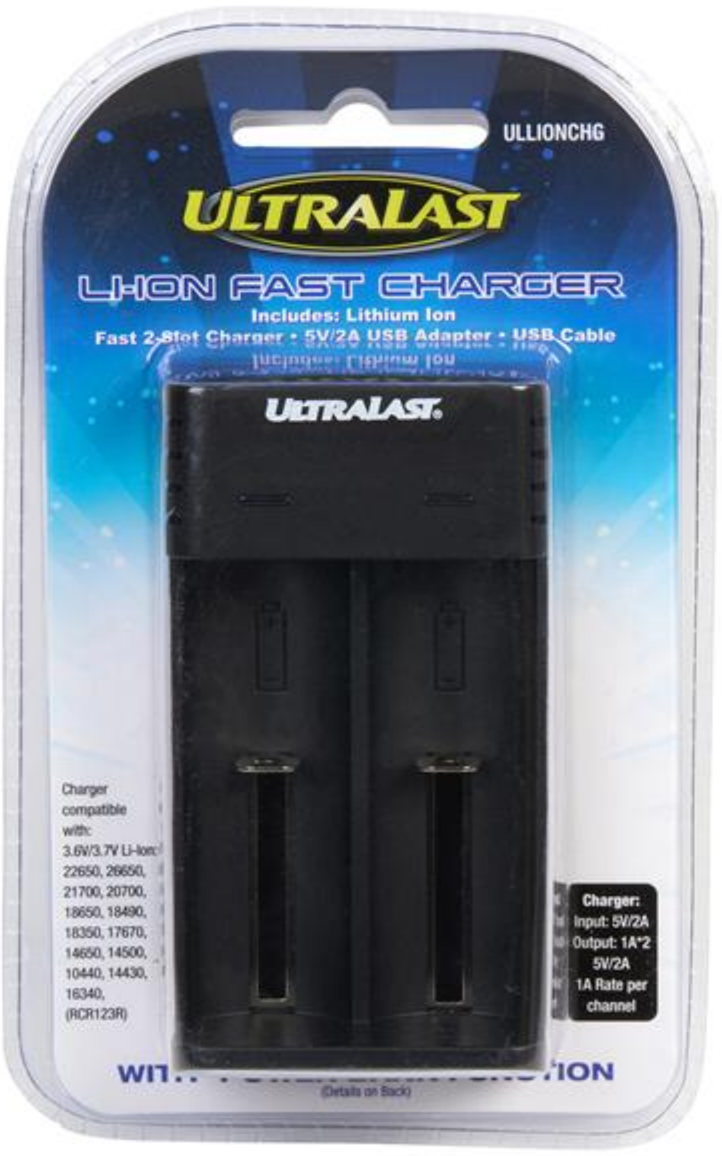
Wonder Echo Mic and Speaker

Figure 14: Microphone and speaker



Note: Hiwonder. (n.d.). *WonderEcho: AI voice recognition module* [Product image]. Hiwonder. https://www.hiwonder.com/products/wonderecho

Figure 15: Battery charger for Lithium-Ion batteries


ULLIONCHG
ULTRALAST
LI-ION FAST CHARGER
Includes: Lithium Ion
Fast 2-Slot Charger • 5V/2A USB Adapter • USB Cable
ULTRALAST
Charger compatible with: 3.6V/3.7V Li-Ion: 22650, 26650, 21700, 20700, 18650, 18490, 18350, 17670, 14650, 14500, 10440, 14430, 16340, (RCR123R)
Charger: Input: 5V/2A Output: 1A*2 5V/2A 1A Rate per channel
(Details on Back)

Dantona Industries. (n.d.). *ULLIONCHG – Battery charger* [Product image]. DigiKey. https://www.digikey.ca/en/products/detail/dantona-industries/ULLIONCHG

§09: Theory of Operation

The AI Robot is built around three main electronic control units that work together to provide movement, sensing, power management, and real-time intelligent responses. These three units are:

1. Raspberry Pi 5: the main computer responsible for running ROS 2, camera processing, LLM interaction, the dashboard server, and all the logic.
2. Custom PCB: a dedicated circuit board that carries two HC-SR04 ultrasonic sensors, the current limiting and level shifting resistors, and the LED indicators used for collision detection.
3. Motor Driver Expansion Board: responsible for motor control, servo control, and power distribution across the entire system through built in buck converters and motor driver chips.

Together, these three modules control every major function inside the robot.

1. Power Distribution and Control

The power management of the robot is controlled by the Motor Expansion Board, which includes its own buck converter. The main battery source consists of a 7.2V NiMH rechargeable pack that supplies power to the board.

- 5V is supplied directly to the Raspberry Pi 5 and the Free nova display.
- 3.3V is used for GPIO logic inputs and sensor signals.
- 3 to 6V is used to drive the four TT motors and the pan/tilt servo motors.

The custom PCB draws almost no power except for the ultrasonic sensors and two LEDs which tells users when ultrasonics detect anything in front of them. Since it only handles logic signals and uses resistors for level shifting, it does not interfere with the main power distribution.

2. Sensor Integration and Data Processing

All the sensing and intelligent based making is controlled by the Raspberry Pi 5. It communicates with the sensors and peripherals through GPIO, I²C, and USB connections depending on the respective devices.

Custom build PCB which contains Ultrasonic sensors and the LEDs

The custom build PCB carries two HC-SR04 ultrasonic sensors, connected with a dedicated LEDs.

- When any object is detected in front of Ultrasonic sensor, the corresponding LED lights up.
- The PCB includes resistors used to safely drop the 5V Echo signal down to 3.3V before reaching the Raspberry Pi.

Additional Sensors & Peripherals

- A third ultrasonic sensor is connected to the Motor Expansion board.
- The Raspberry Pi Camera Module streams video to the dashboard and supports face tracking.
- An external microphone and speaker allow voice interaction.

The Raspberry Pi uses ROS 2 to receive continuous sensor readings and publish movement commands. The sensor data is also streamed in real time to a web dashboard.

3. Motor Drivers and Motion Control

The movement of the robot is fully controlled by the Motor Driver Expansion Board and Raspberry Pi, which uses the TB6612FNG dual motor driver chip.

This board controls:

- Four TT motors with mecanum wheels, allowing omnidirectional movement
- The SG90 servo motor for moving the camera left and right (pan/tilt)

The Raspberry Pi does not directly drive any motor pins. Instead:

- The Pi sends Twist messages over ROS 2 on /cmd_vel
- The motor board interprets these commands and applies the correct PWM signals to each motor

This separation keeps the Raspberry Pi focused on processing while the motor board handles the electrical load and timing for motor control.

4. User Interface and Additional Components

A. Freenova 7″ Touch OLED Display (Model FNK0078)

The Freenova touchscreen acts as the “face” of the robot. It is connected through a DSI ribbon cable and USB for touch input. The screen displays:

- Animated facial expressions
- Battery status
- Collision alerts
- System information

Its size and brightness make the robot feel more interactive and human like.

B. Camera and Head Movements

The camera is mounted on a small servo controlled bracket that enables head movements. The servo responds to ROS messages, allowing the robot to “look” towards the user, detected object and the direction of the movement.

This makes the robot more responsive and human like.

§10: Product Operating Instructions

Power Setup

The robot is powered using a 7.2V NiMH battery pack, which plugs directly into the Motor Driver Expansion Board using the cylindrical DC jack.

The expansion board has a built-in buck converter that regulates the battery voltage and outputs:

- 5.0 V for the Raspberry Pi 5 and the Free nova display
- 3.3 V for GPIO logic
- 3 to 6 V for the four TT motors and the pan/tilt servo

The PCB also draws power from the expansion board. Each component of the Custom designed PCB draws its own power from the Motor Expansion board. Expansion board also consists of the female headers from Raspberry Pi but PCB is equipped with resistors for voltage drop for Ultrasonics and LEDs.

The PCB only draws small logic level current, so it does not overload the system.

The robot uses two organized 2*5 pin headers to keep wiring clean and errorfree.

These headers include

- All three ultrasonic sensors (VCC, Trig, Echo with level shifting, GND)
- Power lines (5V and 3.3V as required but voltage is regulated using resistors)
- GPIO signal connections to the servos and motor control logic
- The DSI ribbon pathway for the Free nova FNK0078 7" OLED touch display is directly connected from Motor Expansion board to the OLED Display

Both connectors ensure that all signals between the Raspberry Pi, the PCB, and the motor expansion board stay properly aligned and separated.

Sensor and Motor Connections

Each sensor and motor plugs into a designated connector:

Ultrasonic Sensors

- Each HC-SR04 plugs into a 4-pin header (VCC, Trig, Echo, GND)
- The PCB includes resistor dividers to safely drop the 5V Echo signal down to 3.3V
- LEDs on the PCB light up when an object is detected

Motors

- The four TT motors are connected to the motor terminals on the board
- The expansion board supplies 3 to 6V motor power from the main power input to components.

Servo

- The pan/tilt SG90 servo is connected to the dedicated servo header on the Motor Expansion board
- Used for camera movement (Head Movement)

Display Connection

The robot's face is shown on the Free nova FNK0078 7″ Touch OLED Display.
It connects using:

- DSI ribbon cable sends video signals to Raspberry Pi 5

This display shows facial expressions, sensor readings, and system information in real time.

Power ON

To power the robot:

1. Ensure the battery pack is connected to the motor driver's DC jack.
2. The expansion board automatically regulates power and supplies 5V to the Raspberry Pi.
3. Once the Pi boots, the ROS 2 nodes start handling motors, sensors, and the display.

The ultrasonic sensors begin measuring distances immediately, and the OLED display shows the robot's face, warnings, or status messages.

Testing

User Must follow these checks to confirm everything is working:

- Move an object in front of each ultrasonic sensor and verify distance readings appear on the display.
- Ensure the corresponding LED lights are up whenever an obstacle is detected.
- Run the motor test through the dashboard or joystick to confirm forward, reverse, turning, and mecanum sliding movements.
- Test the pan/tilt servo to confirm smooth head motion.
- Check that the display shows updated info without delays

Maintenance

To keep the robot running smoothly:

- Keep the PCB clean, dust-free, and dry.
- Avoid pulling or bending the ultrasonic sensor wires.
- Periodically tighten motor driver screw terminals.
- Recheck output if the battery starts losing capacity. Change the batteries if necessary.
- Inspect the DSI ribbon cable to ensure it stays firmly seated in both connectors.

Shutdown Procedure

To safely turn off the robot:

1. Stop all running programs and ROS 2 nodes on the Raspberry Pi.
2. Perform a proper shutdown through the Raspberry Pi OS (*DO NOT UNPLUG ANY CONNECTIONS WHILE ROBOT IS ON).
3. Once the Pi is fully off, disconnect the battery from the main power feed.
4. Ensure the motors stop spinning before storing the robot.

§11: Maintenance Requirements

The maintenance of AI Robot includes a lots of steps:

1. Regular/Daily maintenance:

A. All the components should be cleaned using Anti-Static cloth. Raspberry pi, Motor Expansion board along with PCB top surface should be cleaned using Anti-static dust remover cloth.

B. All the heated areas should have proper heat sinks along with proper ventilation on all sides.

C. All the jumper wires are properly attached and there should be no loose wire connections. All screws should be properly tightened and all the wires should be properly plugged at its place.

Figure 16: Anti- Static Cloth

2. Software Maintenance:

A. The Raspberry Pi should be fully updated according to latest updates.

B. All the errors and bugs in the code should be fixed during daily maintenance.

C. All the files should be backed up locally in an attachable SSD and in a cloud based server.

D. Li-Ion batteries lose life after every charge, so every 6 month the batteries should be replaced with new ones

3. Cleaning Procedure:

A. Robot must be shut down/ Power-off before doing any kind of maintenance.

Figure 17: DO-NOT use liquids

B. After shut down, the Raspberry Pi, Motor expansion board along with PCB board should be cleaned with an anti-static cloth.

C. Any kind of cleaning liquid, water, alcohol must not be used to clean the electrical components. This will damage the components.

4. User Interaction:

A. Anti-Static wrist band or Anti-static gloves should be used while handling the product during maintenance.

B All the connections should be mapped for easy cable routing in the robot during maintenance.

5. Summary:

Robot should be cleaned and maintained according to all the steps given. The overall life of an electrical is dependent on how well it is maintained, so maintenance of robot as described above is most important.

§12: Future Improvements

The current AI Mobile Robot prototype successfully demonstrates voice interaction, camera streaming, ultrasonic distance sensing, and IoT monitoring on a Raspberry Pi–based platform. It is already suitable for controlled indoor demonstrations (e.g., reception, classroom, or HR kiosk use). However, several realistic, technically improvements could make it more adaptable to the targeted industries and increase its long-term commercial value.

This section summarizes the main future upgrades, enhanced navigation and safety, human–robot interaction, and IoT / platform enhancements:

12.1 Summary of Proposed Improvements

Table 12.1 – Summary of Future Improvements

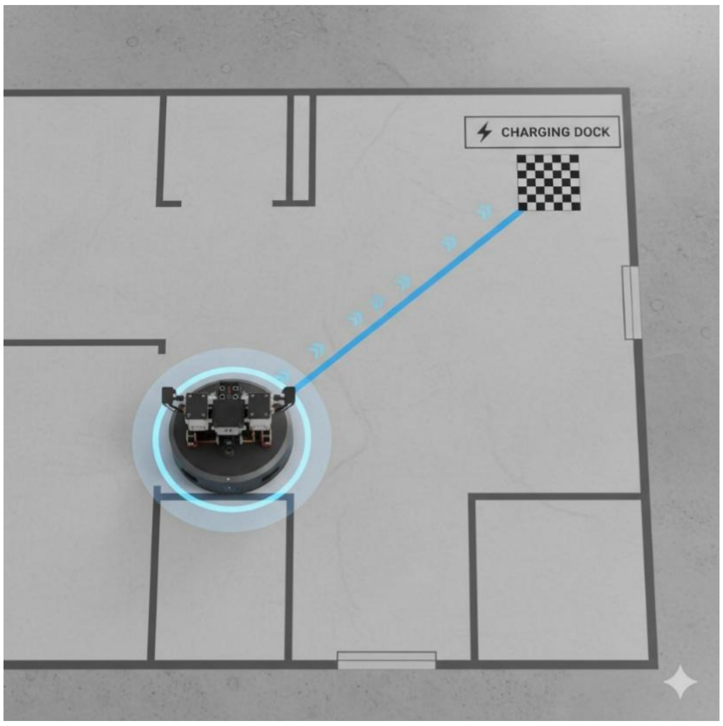
	Area	Planned Improvement (Short Description)	Main Benefit
1	Navigation & Docking	Apple and Google's Map-based navigation and automatic charging dock	True autonomy and longer unattended operation
2	Safety & Power	Bumper switches, E-stop, watchdogs, smarter battery management	Safer around users, protects motors and drivers
3	Perception & Interaction	Person-follow, richer face animations, gesture input	More natural, intuitive human–robot interaction
4	IoT & Analytics	Secure Android and IOS based app.	Easier monitoring, tuning, and troubleshooting, more secure connection.
5	Modular Hardware & Software	Standard mounts, swappable panels, modular code structure	Easier upgrades and reuse in other industries as required without the need of major investment in R&D.

12.2 Navigation, Docking, and Safety

Autonomous navigation and docking

Currently, the robot relies on ultrasonic distance readings and code scripts for movement. In future work, the same sensor data can be extended to simple 2D mapping and path planning, allowing the robot to store known corridors or rooms and move between “waypoints” such as a reception desk or entrance. A low-cost charging dock with visual markers (e.g., a QR code) would let the robot return to charge automatically once the battery level drops below a threshold, building on the existing Li-ion supply and power distribution already implemented.

Figure 18: Charging Dock guide



(Note: A robot going to a charging station by picking up instruction from a QR code represented by a chess board for demonstration (Gemini Ai, n.d))

Improved safety and protection

Because the robot operate near people, additional safety layers are important:

- Front and rear bumper switches to physically detect contact and immediately stop the motors.
- A clearly labelled emergency stop (E-stop) that cuts motor power but leaves the Raspberry Pi active so the event can be logged.
- Additional sensors solely for monitoring on the motor driver outputs (e.g., TB6612FNG) to detect stall conditions and shut down before motors or drivers

overheat (Toshiba Corporation, 2007).

- A watchdog timer that forces the robot into a safe “motors off” state if the main control loop stops sending periodic “heartbeat” signals.

These additions would make the platform safer for public demonstrations and reduce damage risk if a software crash occurs.

12.3 Summary

In summary, the current robot provides a strong foundation for demonstrating AI, IoT, and robotics in a compact mobile platform. The improvements proposed above: autonomous navigation and docking, layered safety and battery management, richer perception and expressions, and a more capable IOS/Android app, are all incremental upgrades to the existing design. Implementing them will allow the system to evolve into a reusable teaching and demonstration platform that future students as well as industry professionals can continue to build on.

§13: Conclusions

13. Conclusions: Challenges, Successes, Evidential Accuracy

13.1 Overall Summary

This project started with a simple goal: turn a stationary AI assistant into a mobile robot that can move, see, speak, and report its status like an IoT device. By the end of the term, we had a working prototype running on a Raspberry Pi with mecanum wheels for movement, ultrasonic sensors on a custom PCB, an OLED “face,” a camera, microphone, speaker, and a browser-based dashboard. The robot can drive, display distance readings, stream video, and interact using GPT-style responses. Thus, we were the main objectives, basic safe mobility, conversational interaction, real-time sensing, and status monitoring were met.

13.2 Key Challenges

We ran into three main types of challenges:

- **Hardware integration:** Designing a PCB that safely connects 5 V HC-SR04 sensors to 3.3 V GPIO pins forced us to think about resistor values, signal levels, and connector placement. Routing power and I^2C for three sensors plus the OLED on a small board took several iterations.
- **Power and mechanical design:** Getting the Li-ion battery, expansion board, motors, Raspberry Pi, and sensors to share power without brownouts was harder than expected. We had to redo wiring, check grounding points, and adjust how parts were mounted so that nothing hit the mecanum wheels.
- **Software and communication:** Making ROS 2 nodes, topic names, and the web dashboard talk to each other cleanly took time. Camera, sonar, D-pad input, and dashboard commands all needed to use consistent topics and not conflict with the already running PID processes, which required a lot of testing and fixes.

13.3 Project Successes

Even with those issues, several parts of the project turned out well:

- The robot responds reliably to dashboard commands (forward, backward, left, right, stop), and the motion is smooth enough for indoor demos.
- The ultrasonic sensors give stable distance readings that are good enough for obstacle avoidance.
- The OLED face and GPT based conversation give the robot a personality, which

makes it more engaging than a basic remote-controlled car.

- The dashboard is a major success: seeing distances, logs, and status in real time made debugging easier and makes the robot feel like a proper IoT device instead of a plug and play device.

13.4 Evidence and Accuracy of Results

We checked evidential accuracy using simple but meaningful tests:

- For the ultrasonic sensors, we placed targets at known distances measured with a tape measure. We then set the robot to stop and avoid if these is any object within 5 cm of range, it worked perfectly. We then tested the code again with ranges from 5 to 110 cm and a red led for status indicator of collision. All the tests were passed successfully.
- For movement, we verified that each dashboard button triggered the correct direction and that the stop command reliably set the values of x,y and z to zero.
- We tested the camera stream over the local network to make sure latency was low enough to drive while watching the video feed. Moreover, the robot replied to all the instructions in a humorous way, while following the instructions. Thus, the robot was able to retain the driving instructions while speaking to the person in real- time without any issue,
- During typical lab-length sessions, the robot did not reboot or crash from power issues, which suggests the power design is stable enough for this stage. The total runtime is 1.6 to 1.8 hrs for the robot.

13.5 Final Reflection

Overall, the project shows that a low-cost, raspberry pi-controlled robot can connect large language models to provide real-time monitoring and hold conversations. It combines electronics, PCB design, embedded programming, networking, ROS 2, and web development into one system that can be reused or extended by future groups. There is still clear room for improvement, especially in autonomous navigation, stronger safety layers, and deeper app integration. But the current prototype already meets its core goals and provides a solid base for future work.

§14: References

1. Adafruit Industries. (2018). *DC gearbox motor – "TT motor" – 200RPM – 3 to 6 VDC (Product ID: 3777)* [Data sheet]. Adafruit. https://media.digikey.com/pdf/Data%20Sheets/Adafruit%20PDFs/3777_Web.pdf
2. Adafruit Industries. (n.d.). *DC gearbox motor – "TT motor" – 200 RPM – 3 to 6 VDC*. Adafruit. https://www.adafruit.com/product/3777
3. Adafruit Industries. (n.d.). *Monochrome 0.96" 128x64 OLED graphic display*. Adafruit. https://www.adafruit.com/product/326
4. Amazon. (n.d.). *C G CHANGEEK mini USB microphone for laptop and desktop computer, with gooseneck & universal USB sound card, compatible with PC and Mac, plug & play, ideal condenser mic for remote work, online class*. Amazon. https://a.co/d/735ZZqu
5. Amazon. (n.d.). *Gikfun 1.5" 4Ohm 3W full range audio speaker stereo woofer loudspeaker for Arduino (Pack of 2pcs) EK1794*. Amazon. https://www.amazon.ca/Gikfun-Speaker-Stereo-Loudspeaker-Arduino/dp/B01LN8ONG4
6. Google. (2025). *Gemini* (November 14 version) [Large language model]. https://gemini.google.com/
7. Living.AI. (n.d.). *Emo robot*. Living.AI. https://living.ai
8. Mabuchi Motor. (n.d.). *31002 DC gear motor datasheet* [Data sheet]. Mabuchi Motor. https://www.mpja.com/download/31002md%20data.pdf
9. MIT Media Lab. (n.d.). *Kismet social robot project*. MIT Media Lab. https://www.media.mit.edu/projects/kismet/overview/
10. Morgan, E. J. (2014, November 16). *HC-SR04 ultrasonic sensor* [Technical document]. https://www.alldatasheet.com/datasheet-pdf/pdf/1132204/ETC2/HCSR04.html
11. PiShop. (n.d.). *SG90 180 degrees 9g micro servo motor Tower Pro*. PiShop. https://www.pishop.ca/product/sg90-180-degrees-9g-micro-servo-motor-tower-pro/
12. Raspberry Pi Foundation. (2024). *Raspberry Pi 4 Model B datasheet* [Data sheet]. Raspberry Pi. https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf
13. Raspberry Pi Foundation. (2024). Raspberry Pi *Camera Module 3* [Data sheet]. Raspberry Pi. https://datasheets.raspberrypi.com/camera/camera-module-3-product-

brief.pdf

14. Raspberry Pi Foundation. (n.d.). *Camera Module 3*. Raspberry Pi. https://www.raspberrypi.com/products/camera-module-3/

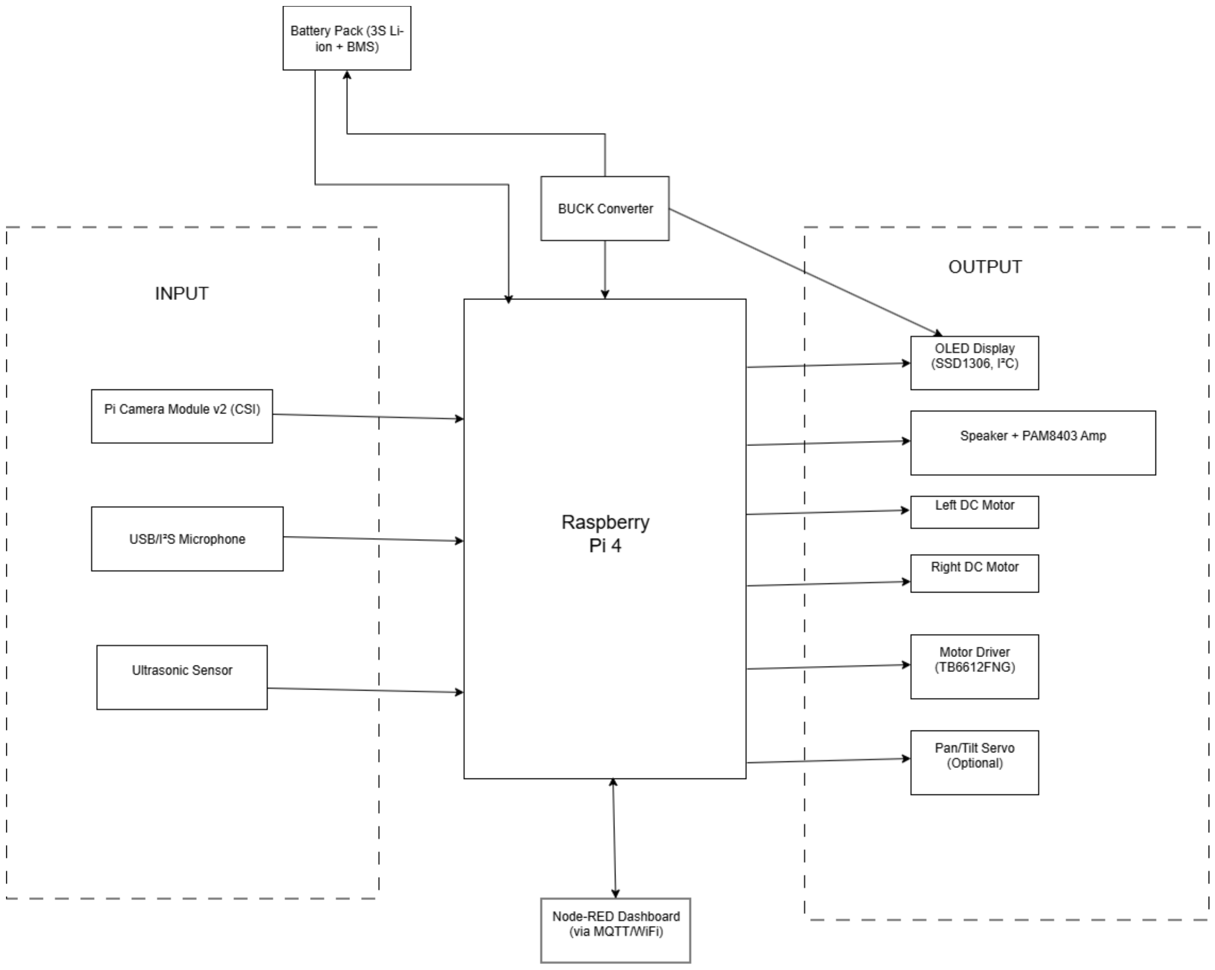
15. Raspberry Pi Foundation. (n.d.). *Raspberry Pi 4 Model B*. Raspberry Pi. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

16. Shutterstock. (n.d.). *Shutterstock – stock photos, royalty-free images, videos, and music* [Website]. Shutterstock. https://www.shutterstock.com/
17. Solomon Systech Limited. (2008). *SSD1306: 128 × 64 dot matrix OLED/PLED segment/common driver with controller (Rev. 1.1)* [Data sheet]. Solomon Systech. https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf
18. SparkFun Electronics. (n.d.). *SparkFun motor driver – dual TB6612FNG (1A)*. SparkFun. https://www.sparkfun.com/products/14451
19. SparkFun Electronics. (n.d.). *Ultrasonic distance sensor – 5V (HC-SR04)*. SparkFun. https://www.sparkfun.com/ultrasonic-distance-sensor-hc-sr04.html
20. Toshiba Corporation. (2007). *TB6612FNG motor driver datasheet* [Data sheet]. Toshiba. https://cdn.sparkfun.com/assets/0/1/b/b/3/TB6612FNG.pdf
21. Willoughby's. (n.d.). Microfiber anti-static cloth for your SLR lens lint free [Product page]. Willoughby's.https://www.willoughbys.com/Microfiber_Anti_Static_Cloth_For_Your_SLR_Lens_Lint_Free_251506.html

§15: Appendices

§15A: System Block Diagram

Figure 19: System Block Diagram



§15B: Electrical Diagrams

Figure 20Wiring Diagram designed in Fusion 360: Group 06

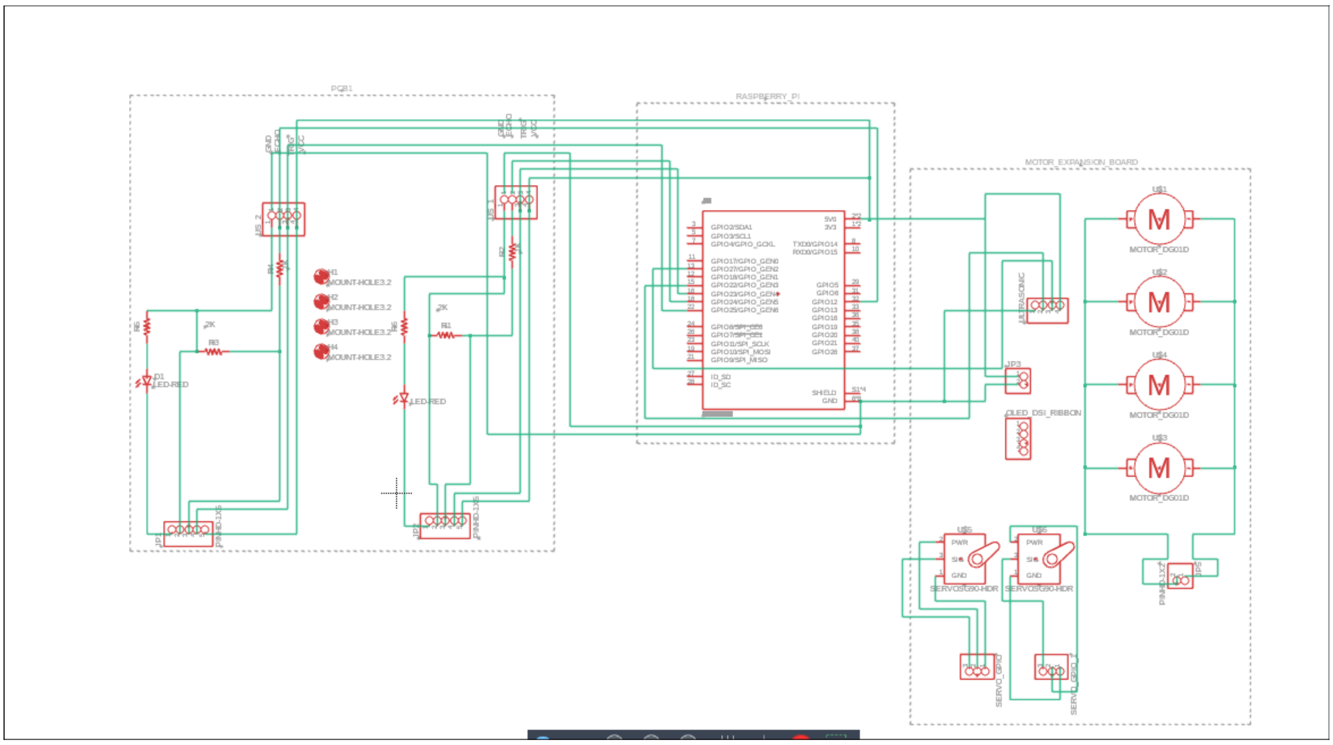


Figure 21: Schematic Diag

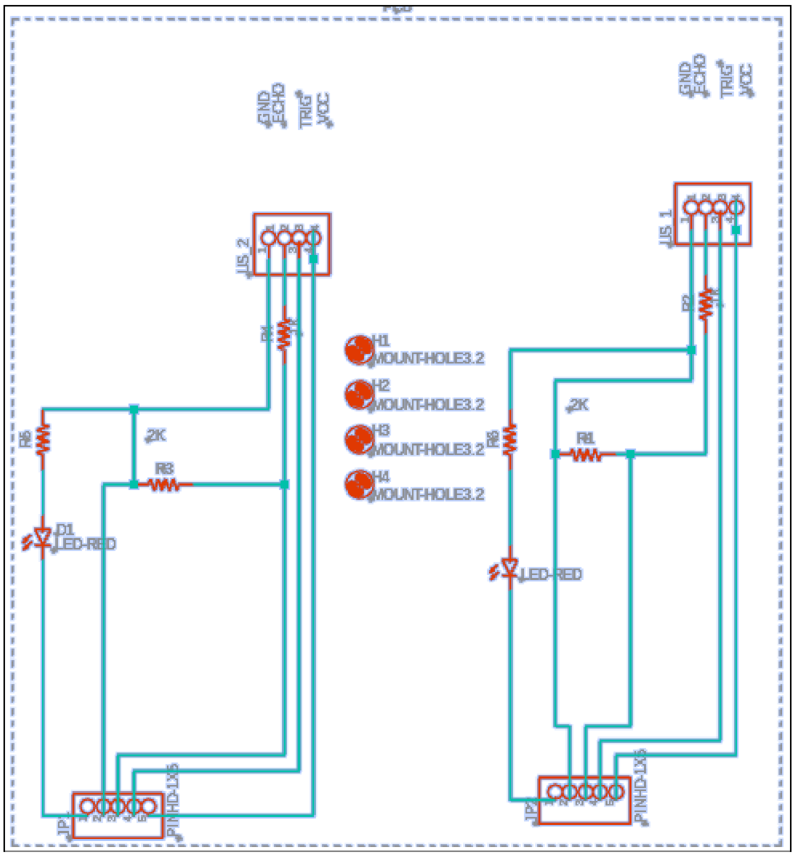


Figure 18: Schematic Fusion (for PCB)

§15C: PCB Design Diagrams

Figure 22: PCB

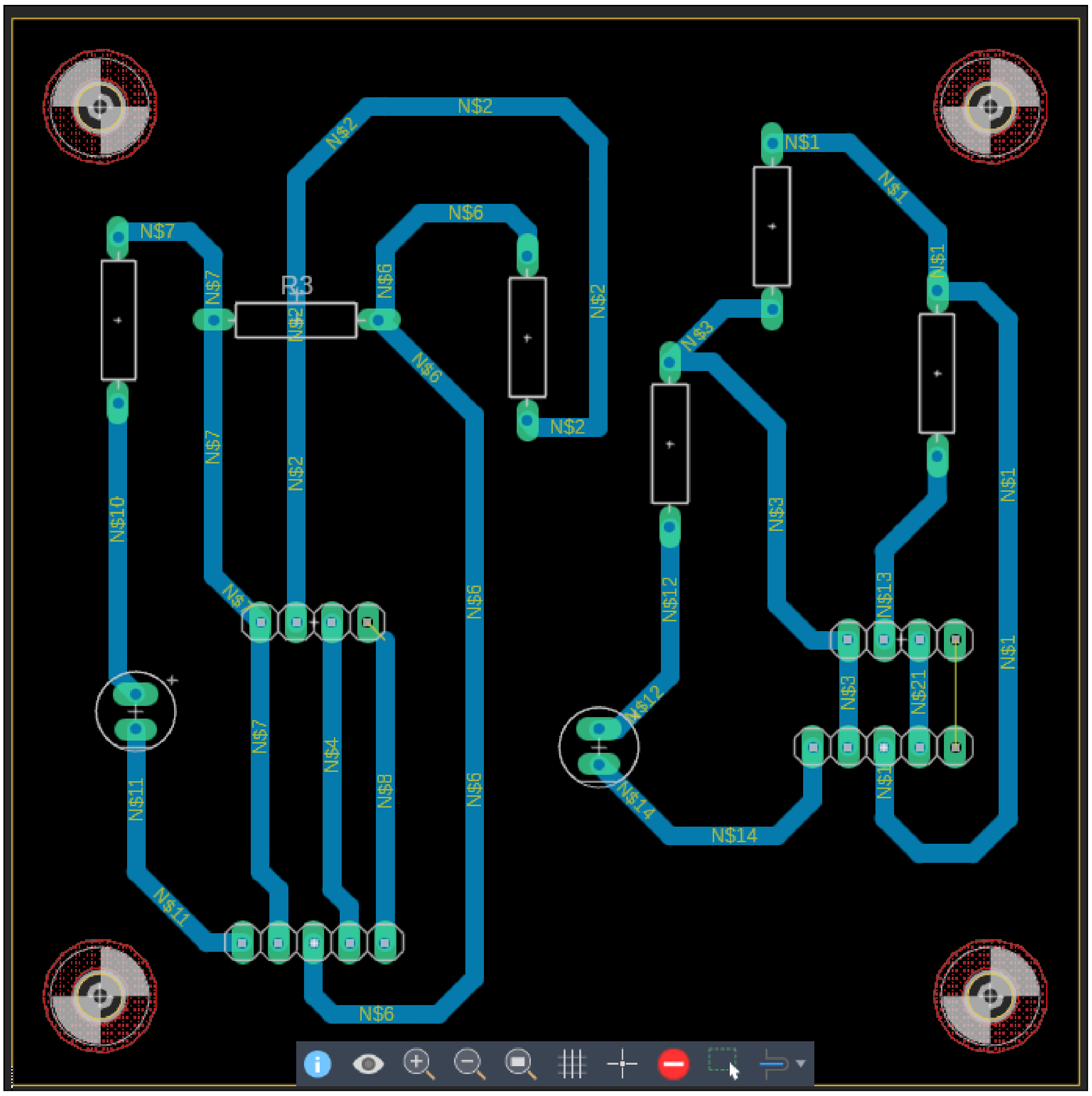


Figure 19: PCB Design

PCB 3D Model

Top Corner View

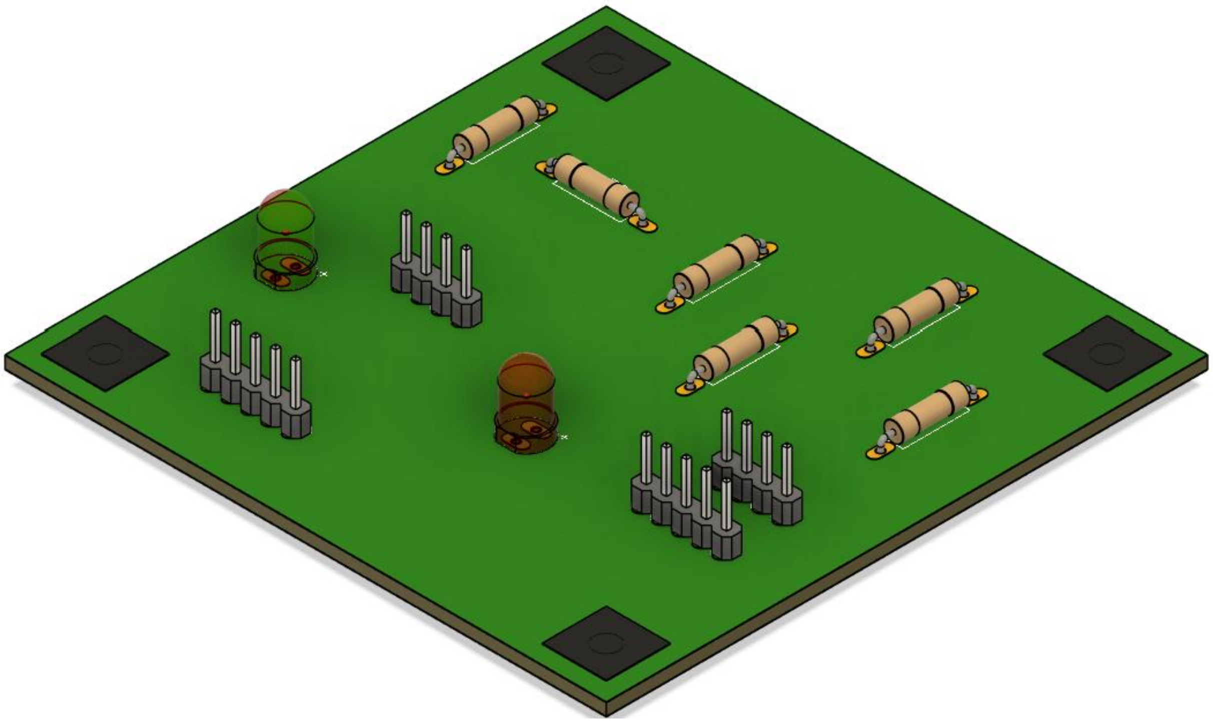


Figure 20: PCB Top Corner View

Top View

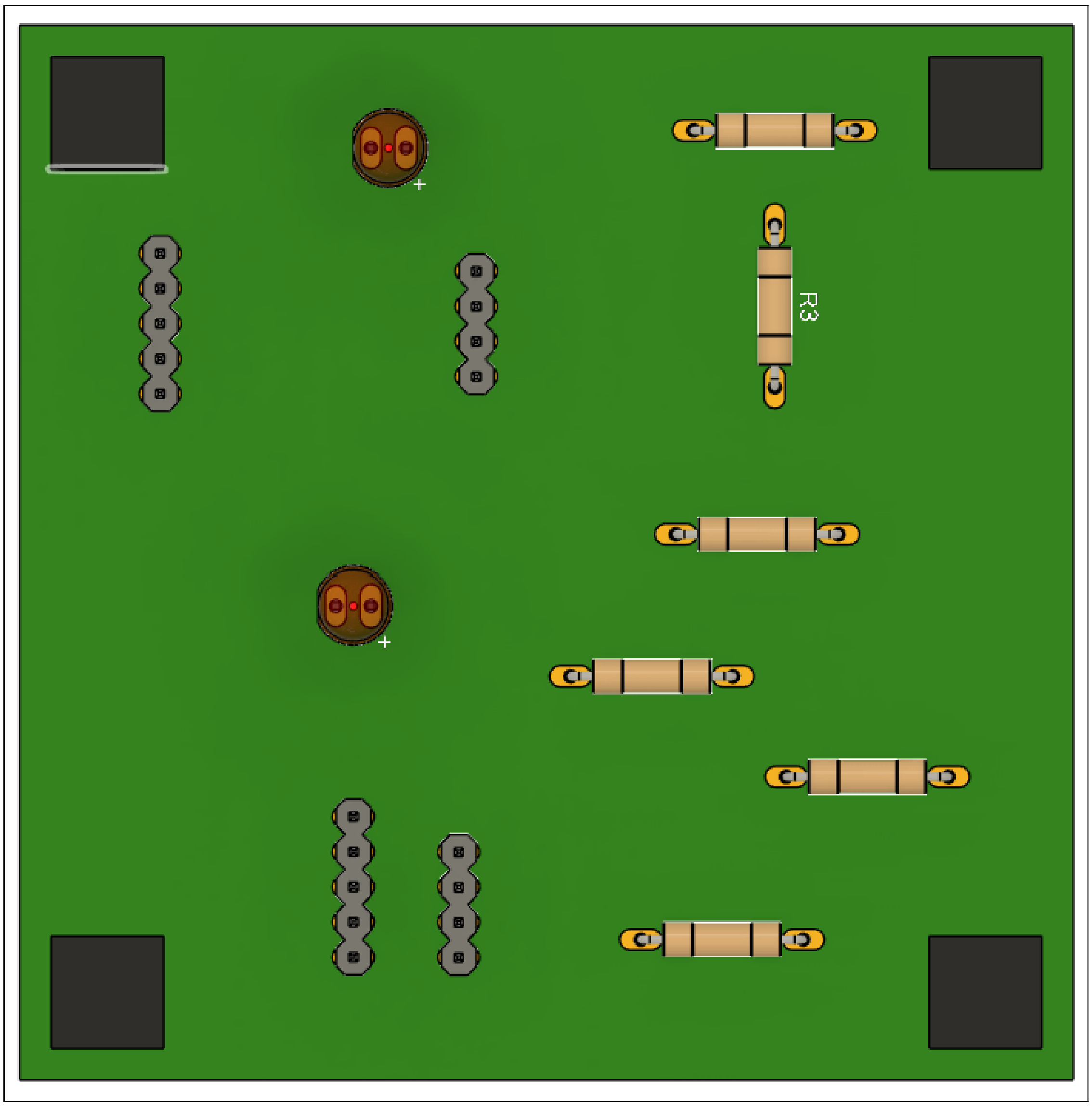


Figure 21: PCB Top View

Bottom View

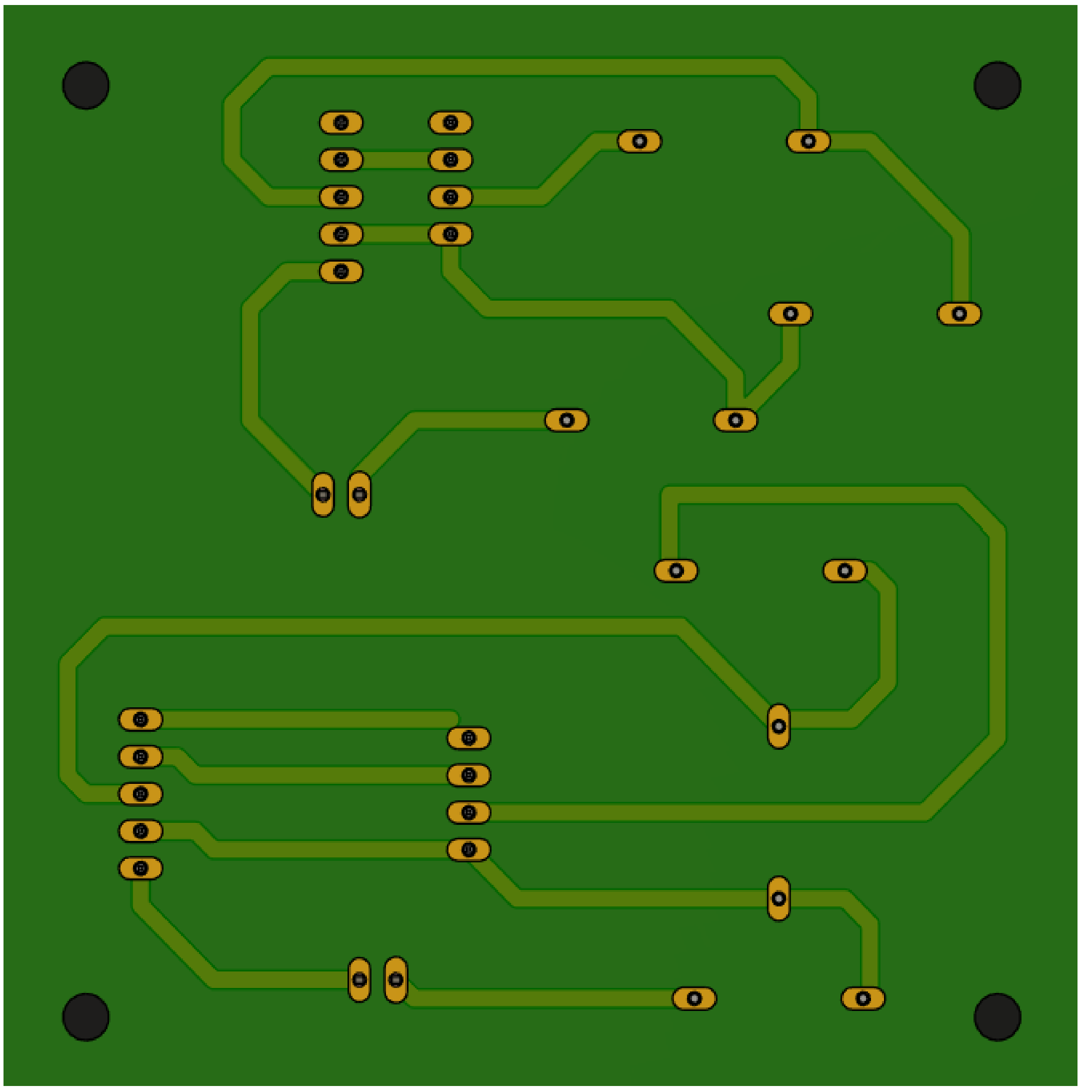


Figure 22: PCB Bottom View

<u>§15D: Mechanical Diagram</u>

Figure 23:mechanical Diagram designed in fusion (Top Axis View)

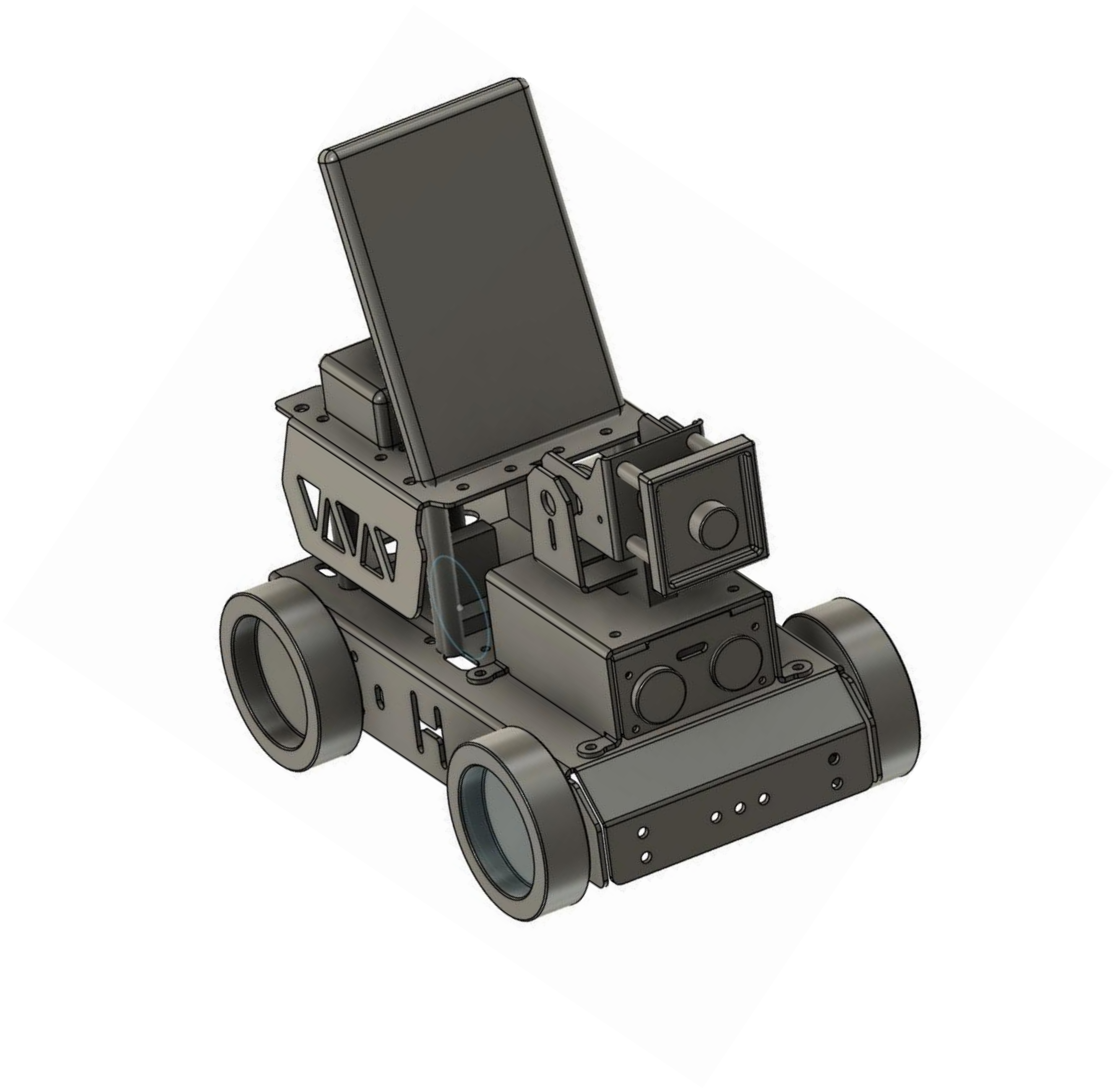


Figure 24: Mechanical Diagram (Front View)

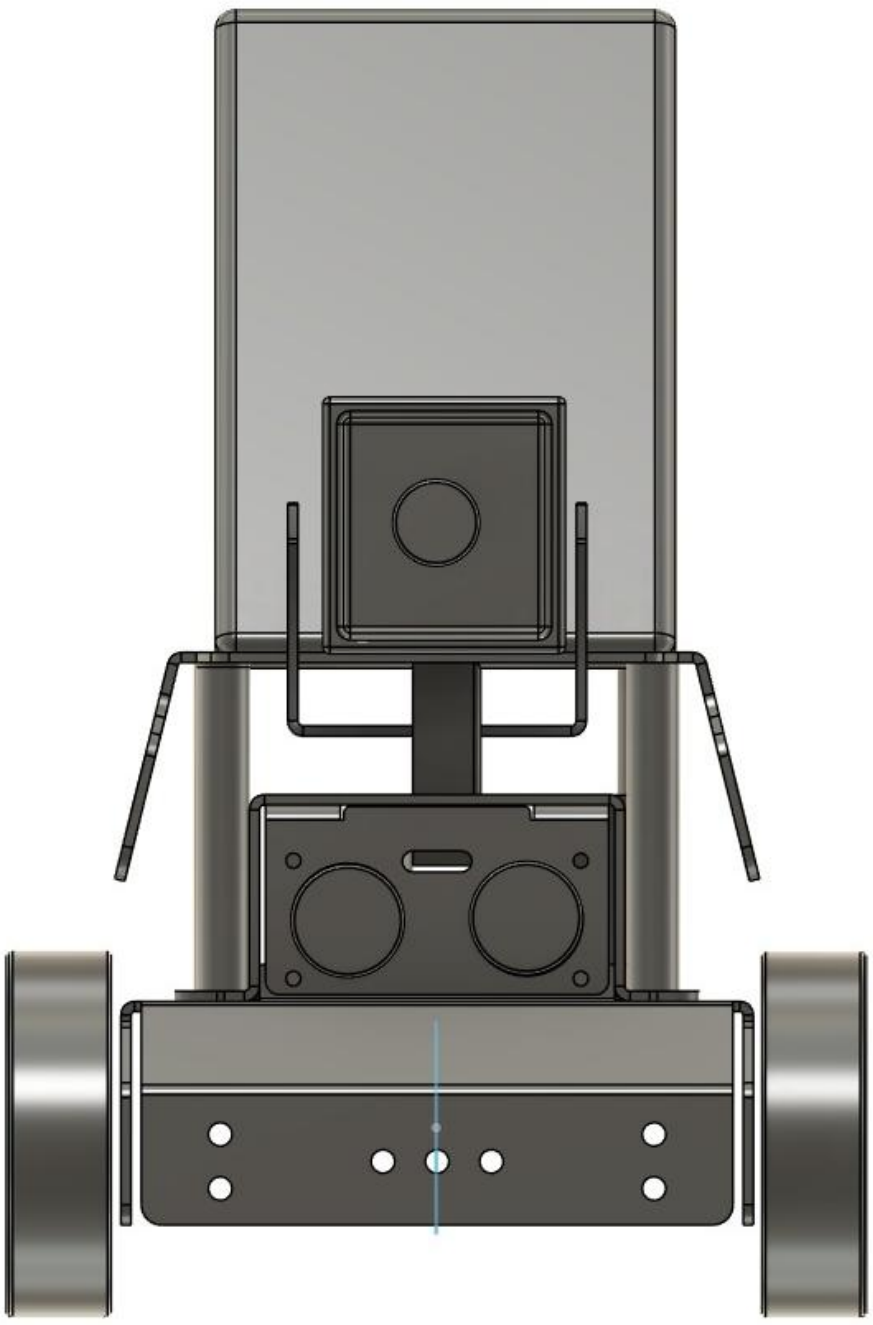


Figure 25:Mechanical Diagram (Top View)

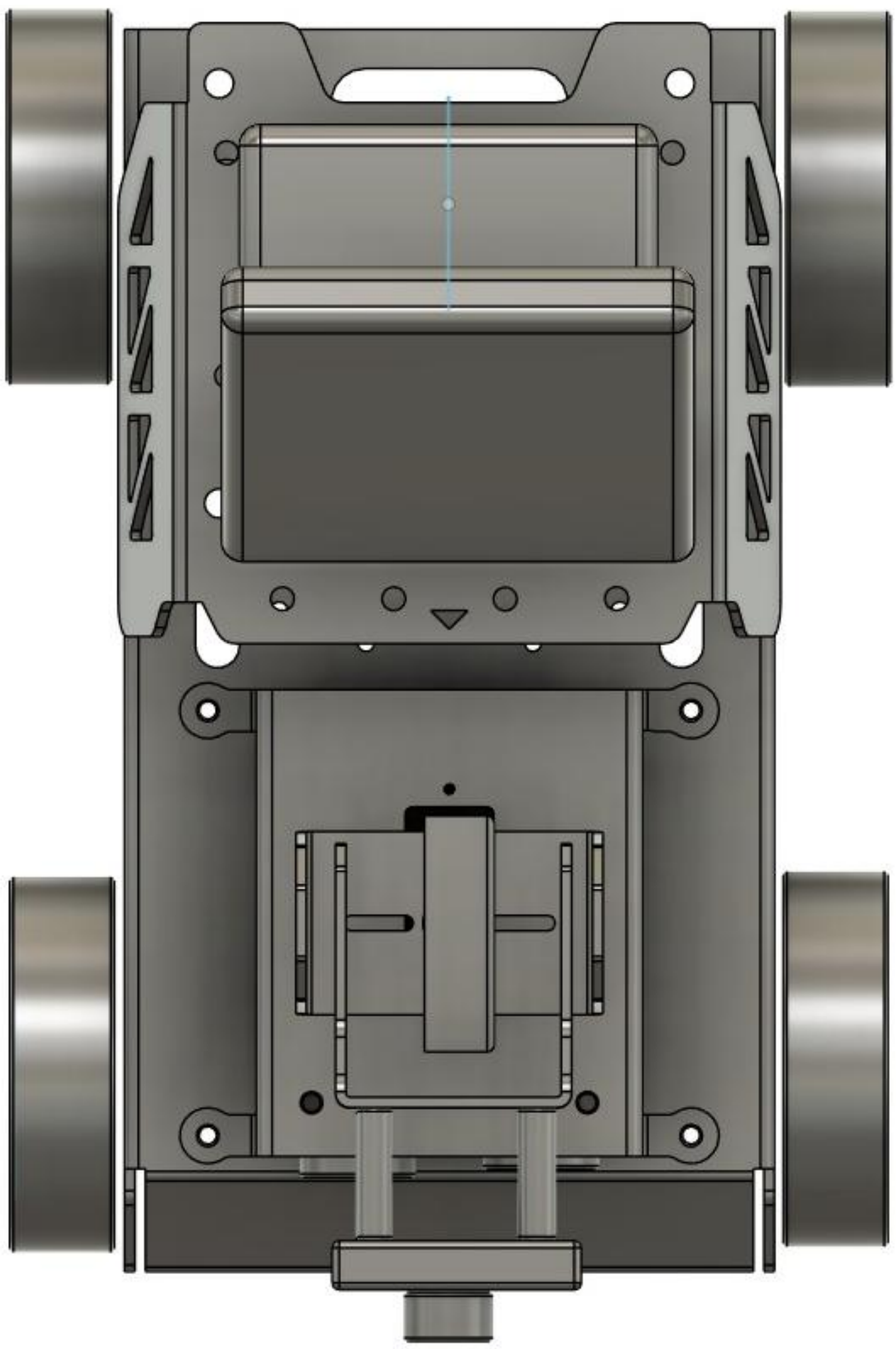
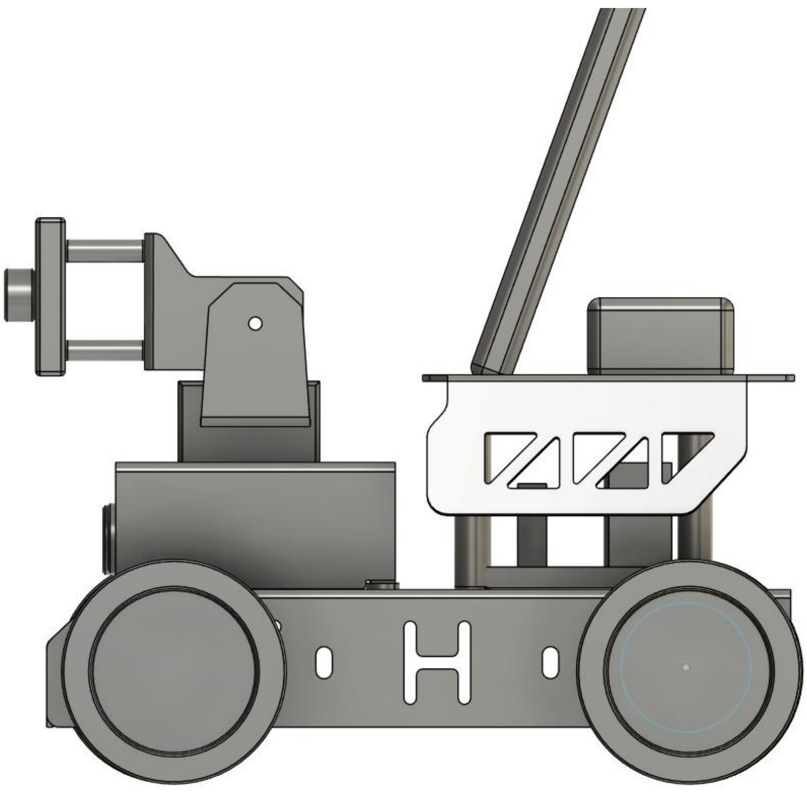
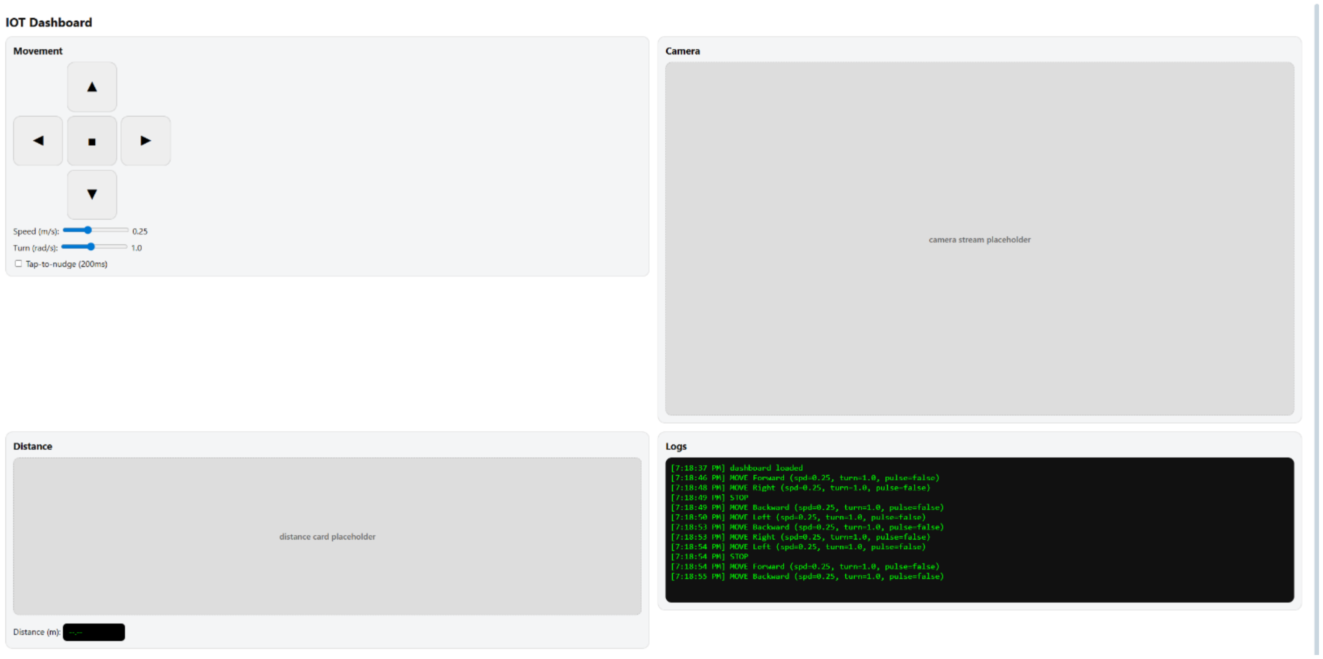


Figure 26: Mechanical Diagram (Side View)



§15E: IoT Dashboard

Figure 27: IoT Dashboard



Speed (m/s): 0.25
Turn (rad/s): 1.0
☐ Tap-to-nudge (200ms)

Distance (m): --.--

```
[7:18:37 PM] dashboard loaded
[7:18:46 PM] MOVE Forward (spd=0.25, turn=1.0, pulse=false)
[7:18:48 PM] MOVE Right (spd=0.25, turn=1.0, pulse=false)
[7:18:49 PM] STOP
[7:18:49 PM] MOVE Backward (spd=0.25, turn=1.0, pulse=false)
[7:18:50 PM] MOVE Left (spd=0.25, turn=1.0, pulse=false)
[7:18:53 PM] MOVE Backward (spd=0.25, turn=1.0, pulse=false)
[7:18:53 PM] MOVE Right (spd=0.25, turn=1.0, pulse=false)
[7:18:54 PM] MOVE Left (spd=0.25, turn=1.0, pulse=false)
[7:18:54 PM] STOP
[7:18:54 PM] MOVE Forward (spd=0.25, turn=1.0, pulse=false)
[7:18:55 PM] MOVE Backward (spd=0.25, turn=1.0, pulse=false)
```

§15F: Coding Flowchart

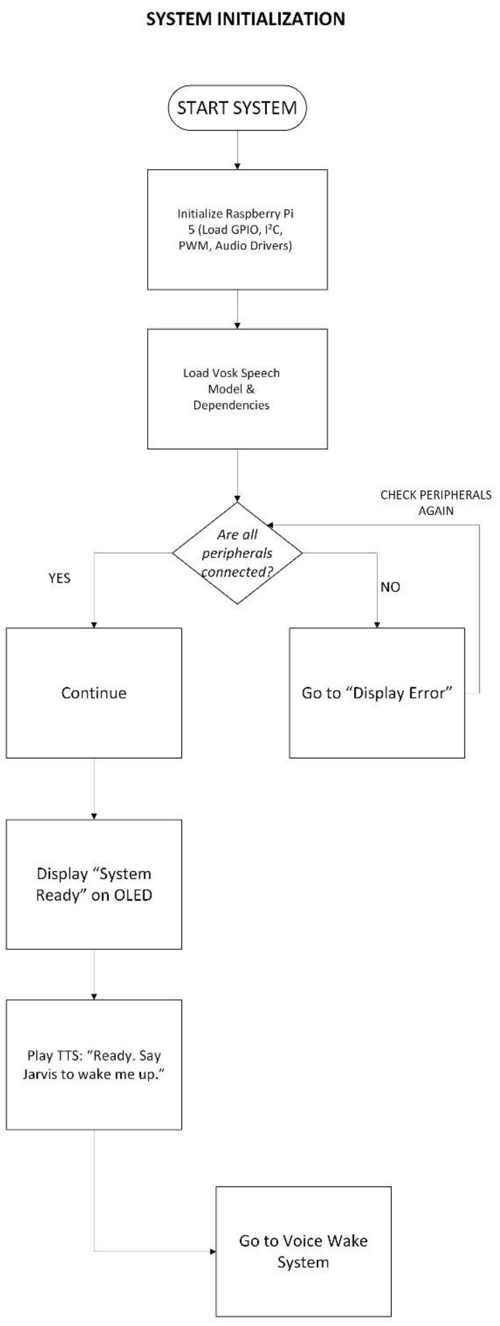

SYSTEM INITIALIZATION
START SYSTEM
Initialize Raspberry Pi 5 (Load GPIO, I²C, PWM, Audio Drivers)
Load Vosk Speech Model & Dependencies
Are all peripherals connected?
CHECK PERIPHERALS AGAIN
YES
NO
Continue
Go to "Display Error"
Display "System Ready" on OLED
Play TTS: "Ready. Say Jarvis to wake me up."
Go to Voice Wake System

Figure 25: Coding Flowchart Page 1

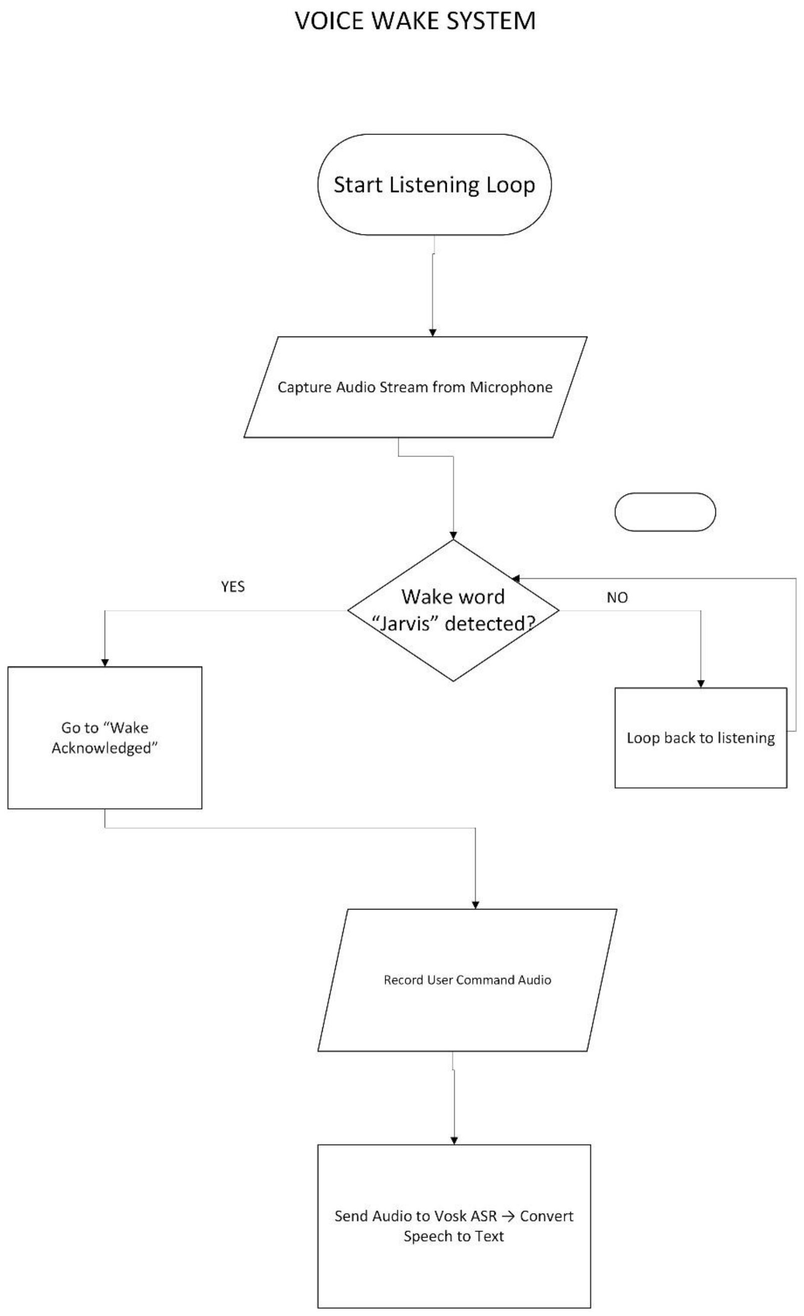

VOICE WAKE SYSTEM
Start Listening Loop
Capture Audio Stream from Microphone
YES
Wake word "Jarvis" detected?
NO
Go to "Wake Acknowledged"
Loop back to listening
Record User Command Audio
Send Audio to Vosk ASR → Convert Speech to Text

Figure 26: Coding Flowchart Page 2

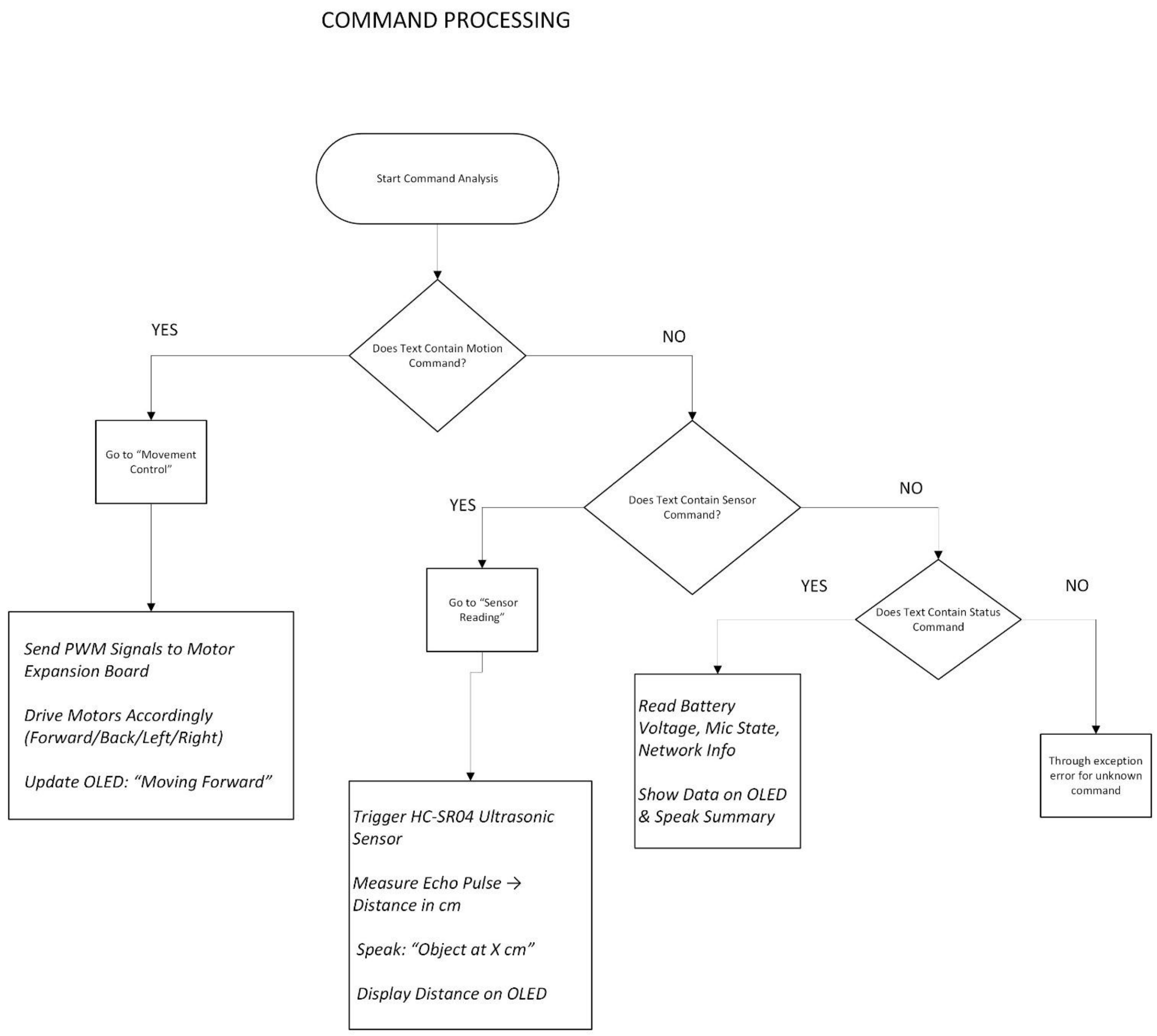

COMMAND PROCESSING
Start Command Analysis
Does Text Contain Motion Command?
YES
NO
Go to "Movement Control"
Send PWM Signals to Motor Expansion Board
Drive Motors Accordingly (Forward/Back/Left/Right)
Update OLED: "Moving Forward"
Does Text Contain Sensor Command?
YES
NO
Go to "Sensor Reading"
Trigger HC-SR04 Ultrasonic Sensor
Measure Echo Pulse → Distance in cm
Speak: "Object at X cm"
Display Distance on OLED
Does Text Contain Status Command
YES
NO
Read Battery Voltage, Mic State, Network Info
Show Data on OLED & Speak Summary
Through exception error for unknown command

Figure 27: Coding Flowchart Page 3

SAFETY & SHUTDOWN

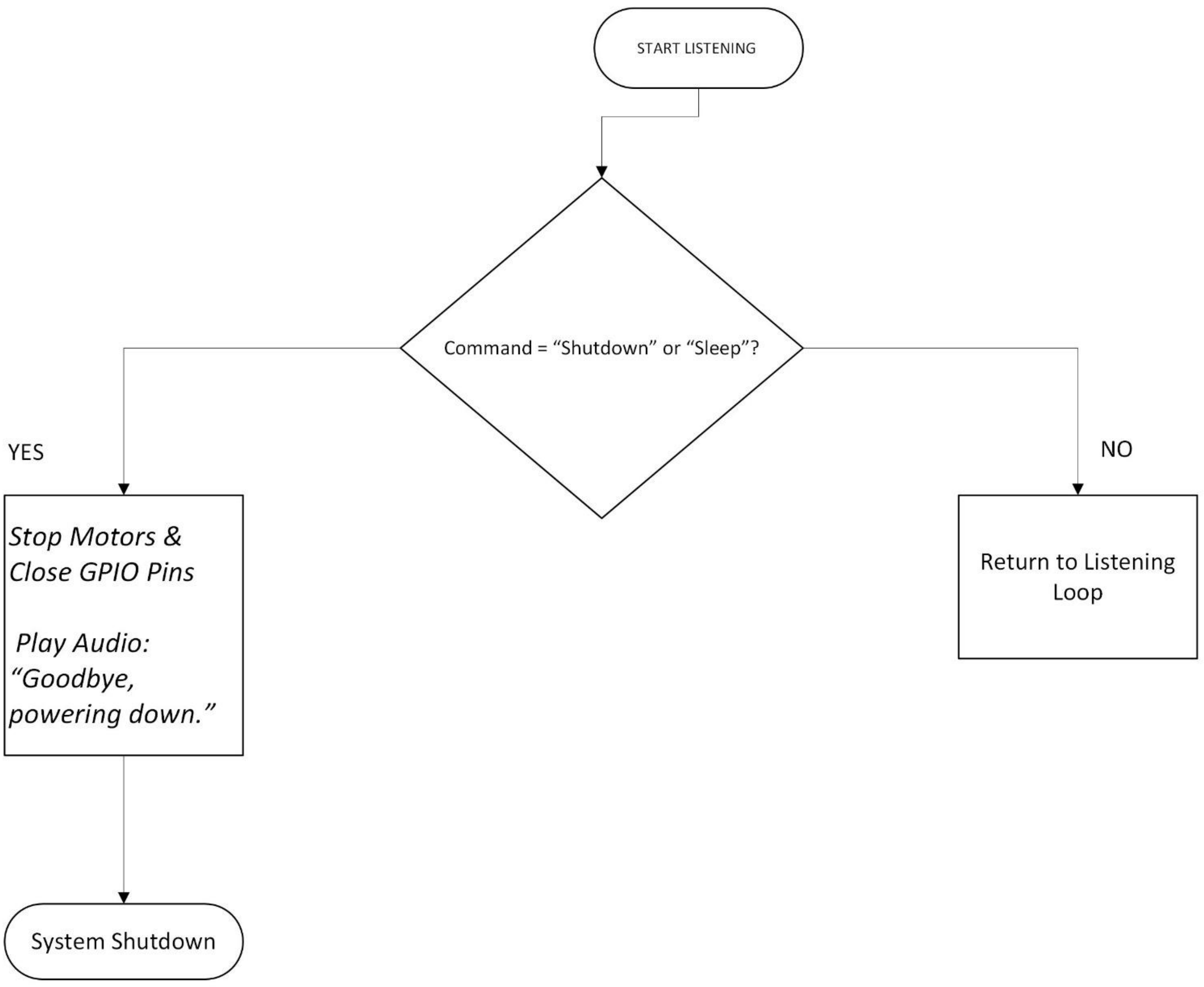

START LISTENING
Command = "Shutdown" or "Sleep"?
YES
NO
Stop Motors & Close GPIO Pins
Play Audio: "Goodbye, powering down."
Return to Listening Loop
System Shutdown

Figure 28: Coding Flowchart Page 4

JarvisDashboard.html

```
<!doctype html>
<meta charset="utf-8">
<title>Jarvis Dashboard (v2)</title>
<style>
:root{--card:#f4f5f6;--ink:#111}
*{box-sizing:border-box}
body{font-family:system-ui,Segoe UI,Arial;margin:24px;color:var(--ink)}
h2{margin:0 0 12px} h3{margin:0 0 10px}
.grid{display:grid;gap:16px}
.card{background:var(--card);border:1px solid #ddd;border-radius:14px;padding:14px}
.pad{display:grid;grid-template-columns:repeat(3,96px);gap:8px;align-items:center;justify-content:start}
button{height:96px;font-size:28px;border-radius:14px;border:1px solid #ccc;background:#eee; touch-
action:none;}
button:active{background:#d0d0d0; border-color:#aaa;}
#btnStop{background:#eaeaea}
label{display:block;margin-top:10px}
.rect{width:100%; aspect-ratio:16/9; background:#ddd; border:1px dashed #bbb; border-radius:12px;
display:flex; align-items:center; justify-content:center; color:#666; font-weight:600}
img.rect { object-fit: cover; }
#distRect{aspect-ratio:4/1}
#logsBox{height:280px; background:#111; color:#0f0; font-family:ui-
monospace,Menlo,Consolas,monospace; border-radius:12px; padding:10px; overflow:auto; white-
space:pre-wrap}
#distValue, #batteryValue{display:inline-block; min-width:120px; padding:8px 12px; background:#000;
color:#0f0; border-radius:8px; font-family:ui-monospace}
@media (min-width: 960px){ #layout{grid-template-columns: 1fr 1fr; align-items:start} }
</style>

<h2>Jarvis Dashboard</h2>

<div id="layout" class="grid">
<section class="card">
<h3>Movement</h3>
<div class="pad">
<div></div><button id="btnF">▲</button><div></div>
<button id="btnL">◀</button><button id="btnStop">■</button><button id="btnR">▶</button>
<div></div><button id="btnB">▼</button><div></div>
</div>
<label>Speed (m/s): <input id="spd" type="range" min="0.05" max="0.6" step="0.05" value="0.25">
<span id="spdVal">0.25</span></label>
<label>Turn (rad/s): <input id="turn" type="range" min="0.2" max="2.0" step="0.1" value="1.0">
<span id="turnVal">1.0</span></label>
<label><input id="pulseMode" type="checkbox"> Tap-to-nudge (200ms)</label>
</section>

<section class="card">
<h3>Camera</h3>
<img class="rect" id="cameraStream" alt="camera stream placeholder">
</section>
```

```
<section class="card">
<h3>Sensors</h3>
<div class="grid">
<div>Distance (m): <span id="distValue">--.--</span></div>
<div>Battery (mV): <span id="batteryValue">----</span></div>
</div>
</section>

<section class="card">
<h3>Logs</h3>
<div id="logsBox"></div>
</section>
</div>

<script>
// --- Globals & Helpers ---
const G=id=>document.getElementById(id);
const spd=G("spd"), spdVal=G("spdVal"), turn=G("turn"), turnVal=G("turnVal"), pulse=G("pulseMode");
const btnF=G("btnF"), btnB=G("btnB"), btnL=G("btnL"), btnR=G("btnR"), btnStop=G("btnStop");
let ws; // Our WebSocket connection
let pulseTimer = null; // For tap-to-nudge

// --- ROS 2 Topics (from the app code) ---
const MOVEMENT_TOPIC = "/cmd_vel";
const MOVEMENT_MSG_TYPE = "geometry_msgs/msg/Twist";
const SONAR_TOPIC = "/sonar_controller/get_distance";
const SONAR_MSG_TYPE = "std_msgs/msg/Int32";
const BATTERY_TOPIC = "/ros_robot_controller/battery";
const BATTERY_MSG_TYPE = "std_msgs/msg/UInt16";

// --- Logging & UI Updates ---
spd.oninput=()=>spdVal.textContent=(+spd.value).toFixed(2);
turn.oninput=()=>turnVal.textContent=(+turn.value).toFixed(1);

function addLog(line, type = 'info'){
const el=G("logsBox");
const t=new Date().toLocaleTimeString();
const color = type === 'error' ? 'color:#f88;' : (type === 'warn' ? 'color:#ff8;' : 'color:#0f0;');
el.innerHTML += `<span style="${color}">[${t}] ${line}\n</span>`;
el.scrollTop=el.scrollHeight;
}
function setDistanceMeters(v){ G("distValue").textContent=(typeof v==="number"?v.toFixed(2):v); }
function setBatteryVolts(v){ G("batteryValue").textContent=(typeof v==="number"?v:v); }

// --- WebSocket Connection ---
function connect() {
const wsHost = `ws://${window.location.hostname}:9090`;
addLog(`Connecting to ${wsHost}...`);
ws = new WebSocket(wsHost);
ws.onopen = () => {
addLog("Connected to robot server (ROS 2).", 'info');
// Advertise that we will be publishing movement
advertiseTopic(MOVEMENT_TOPIC, MOVEMENT_MSG_TYPE);

// Subscribe to sensor data
subscribeToTopic(SONAR_TOPIC, SONAR_MSG_TYPE);
subscribeToTopic(BATTERY_TOPIC, BATTERY_MSG_TYPE);
};

ws.onmessage = (event) => {
try {
const data = JSON.parse(event.data);
if (data.op === 'publish') {

// Handle Sonar Data
if (data.topic === SONAR_TOPIC) {
let meters = data.msg.data / 1000.0; // Convert mm to meters
setDistanceMeters(meters);
}

// Handle Battery Data
if (data.topic === BATTERY_TOPIC) {
setBatteryVolts(data.msg.data); // Show raw millivolts
}
}
} catch (e) {
addLog(`Error parsing RX: ${e}`, 'error');
}
};

ws.onerror = (event) => {
addLog("WebSocket error. Check console.", 'error');
console.error("WebSocket Error: ", event);
};

ws.onclose = () => {
addLog("Disconnected. Retrying in 3s...", 'error');
ws = null;
setTimeout(connect, 3000); // Auto-reconnect
};
}

// --- ROS 2 Communication ---
function advertiseTopic(topic, type) {
const advMsg = { op: "advertise", topic: topic, type: type };
ws.send(JSON.stringify(advMsg));
addLog(`Advertised: ${topic}`);
}
```

```
function subscribeToTopic(topic, type) {
const subMsg = { op: "subscribe", topic: topic, type: type };
ws.send(JSON.stringify(subMsg));
addLog(`Subscribed to: ${topic}`);
}

// Send a ROS 2 command
function sendRosCommand(linear, angular) {
if (!ws || ws.readyState !== WebSocket.OPEN) {
addLog("Not connected. Command not sent.", 'error');
return;
}

// This is the JSON structure for a geometry_msgs/msg/Twist
const twistMsg = {
linear: { x: linear.x, y: linear.y, z: 0.0 },
angular: { x: 0.0, y: 0.0, z: angular.z }
};

// This is the rosbridge wrapper command
const pubMsg = {
op: "publish",
topic: MOVEMENT_TOPIC,
msg: twistMsg
};

const cmdString = JSON.stringify(pubMsg);
ws.send(cmdString);

if (linear.x !== 0 || linear.y !== 0 || angular.z !== 0) {
addLog(`TX: ${JSON.stringify(twistMsg)}`);
}
}

// --- Control Logic ---
const zeroLinear = {x: 0.0, y: 0.0};
const zeroAngular = {z: 0.0};

function handleMove(direction) {
if (pulseTimer) clearTimeout(pulseTimer);

const linear = {...zeroLinear};
const angular = {...zeroAngular};
const speed = +spd.value;
const turnSpeed = +turn.value;

switch (direction) {
case 'forward':  linear.x = speed; break;
case 'backward': linear.x = -speed; break;
case 'left':     angular.z = turnSpeed; break;
case 'right':    angular.z = -turnSpeed; break;
}

sendRosCommand(linear, angular);

if (pulse.checked) {
pulseTimer = setTimeout(sendStop, 200);
}
}

const sendStop = () => {
if (pulseTimer) clearTimeout(pulseTimer);
sendRosCommand(zeroLinear, zeroAngular);
};

// --- Event Listeners ---
function setupButtonListeners(button, direction) {
button.addEventListener("pointerdown", (e) => {
e.preventDefault();
handleMove(direction);
});

button.addEventListener("pointerup", () => !pulse.checked && sendStop());
button.addEventListener("pointerout", () => !pulse.checked && sendStop());
}

setupButtonListeners(btnF, 'forward');
setupButtonListeners(btnB, 'backward');
setupButtonListeners(btnL, 'left');
setupButtonListeners(btnR, 'right');
btnStop.addEventListener("click", sendStop);

// --- Camera Stream ---
function setCameraStream() {
const streamUrl = `http://${window.location.hostname}:8080/stream?topic=/image_raw`;
addLog(`Setting camera stream to ${streamUrl}`);

const imgEl = G("cameraStream");
imgEl.src = streamUrl;
imgEl.onerror = () => {
addLog("Camera stream failed. Is web_video_server running?", 'error');
};
imgEl.onload = () => {
addLog("Camera stream loaded.", 'info');
};
}

// --- Initialize ---
addLog("Dashboard loaded. Initializing...");
```

```
setDistanceMeters("--.--");
setBatteryVolts("----");
connect(); // Start WebSocket
setCameraStream(); // Start camera

</script>
HTML
```

Imu_Filter,launch.py

```
#!/usr/bin/env python3
# This file is a ROS 2 launch file, used to start and configure multiple ROS 2 nodes at once.

# --- Imports ---
# Import necessary modules from the ROS 2 launch system
from launch import LaunchDescription
from launch_ros.actions import Node
from launch.actions import TimerAction
import os
from ament_index_python.packages import get_package_share_directory

# --- Main Launch Function ---
# This is the main function that ROS 2 executes when you run this launch file.
def generate_launch_description():

# --- 1. Find Configuration File ---
# This section finds the location of the 'imu_calib.yaml' file, which contains
# pre-calculated calibration data (like biases) for the IMU sensor.

# Check an environment variable to see if we're running from a compiled workspace or a
development workspace.
compiled = os.environ.get('need_compile', 'False')

if compiled == 'True':
# If compiled, find the package's "share" directory (the standard ROS 2 way)
calibration_package_path = get_package_share_directory('calibration')
else:
# If not (e.g., in development), use a hardcoded path to the 'src' directory.
calibration_package_path = '/home/ubuntu/ros2_ws/src/calibration'

# Create the full path to the calibration file.
calib_file_path = os.path.join(calibration_package_path, 'config/imu_calib.yaml')

# Safety check: If the file doesn't exist, stop and raise an error.
if not os.path.exists(calib_file_path):
raise FileNotFoundError(f"Calibration file not found: {calib_file_path}")

# --- 2. Define the Calibration Node ---
# This node reads the RAW data from the IMU and applies the calibration file to it.

imu_calib_node = Node(
package='imu_calib',           # The name of the ROS 2 package to use
executable='apply_calib',       # The specific program (node) to run from that package
```

```
name='imu_calib',               # A custom name for this running node
output='screen',               # Print this node's log messages directly to the terminal
parameters=[{"calib_file": calib_file_path}], # Pass the path to our .yaml file as a parameter to the
node
remappings=[
# This is the "wiring" for the node:
# It tells the node that its logical INPUT topic 'raw'
# should be connected to the REAL hardware topic '/ros_robot_controller/imu_raw'.
('raw', '/ros_robot_controller/imu_raw'),

# It tells the node that its logical OUTPUT topic 'corrected'
# should be broadcast on a new topic named 'imu_corrected'.
('corrected', 'imu_corrected')
]
)

# --- 3. Define the Filter Node ---
# This node takes the 'corrected' data and applies a complementary filter
# to fuse the accelerometer and gyroscope data, resulting in a stable orientation.

imu_filter_node = Node(
package='imu_complementary_filter', # The name of the filter package
executable='complementary_filter_node', # The specific program to run
name='imu_filter',               # A custom name for this node
output='screen',               # Print its logs to the terminal
parameters=[
{
# Configuration for the filter algorithm:
'use_mag': False,            # Don't use the magnetometer (which is often noisy)
'do_bias_estimation': True,   # Actively try to estimate and remove gyro drift
'do_adaptive_gain': True,     # Automatically adjust the filter's gain
'publish_debug_topics': True  # Broadcast extra topics for debugging
}
],
remappings=[
# "Wiring" for the filter node:
('/tf', 'tf'), # Standard remapping for transform data

# This is the key: It tells the filter that its INPUT topic '/imu/data_raw'
# should be connected to the 'imu_corrected' topic from our first node.
('/imu/data_raw', 'imu_corrected'),

# It tells the filter that its final, clean OUTPUT topic 'imu/data'
# should be broadcast on a new, simple topic named 'imu'.
('imu/data', 'imu')
]
)

# --- 4. Return the Launch Description ---
# This is what actually gets run.

return LaunchDescription([
# Use a TimerAction as a safety precaution.
TimerAction(
period=5.0, # Wait for 5.0 seconds after the launch file is started...

# ...before starting these nodes.
# This gives the main robot controller time to boot up first,
# preventing errors where these nodes can't find their input topics.
actions=[imu_calib_node, imu_filter_node]
)
])

# --- Main Entry Point ---
# This "if" block runs if you execute this file directly (e.g., `python3 imu_launch.py`).
# It's used for testing the launch file by itself.
if __name__ == '__main__':
from launch import LaunchService
ld = generate_launch_description()

ls = LaunchService()
ls.include_launch_description(ld)
ls.run()
```

usb_cam.launch.py

```
#!/usr/bin/env python3
# This is a ROS 2 launch file. Its purpose is to start the USB camera node
# and load its specific configuration file.

# --- Imports ---
import os  # Used to check environment variables and build file paths
from ament_index_python.packages import get_package_share_directory  # A ROS 2 utility to find the path to installed packages
from launch_ros.actions import Node  # The class used to define a ROS 2 node in a launch file
from launch import LaunchDescription, LaunchService  # Core launch system classes

# --- Main Launch Function ---
# This is the primary function that ROS 2 executes when the launch file is run.
# It defines all the nodes and actions that should be started.
def generate_launch_description():

# --- 1. Find Configuration File Path ---
# This logic checks if the code is being run from a "compiled" workspace
# or a "development" (source) workspace, and sets the path accordingly.

# Check for an environment variable named 'need_compile'. Default to 'False' if not found.
compiled = os.environ.get('need_compile', 'False')

if compiled == 'True':
# If 'need_compile' is True, find the package's "share" directory (standard ROS 2 install location)
peripherals_package_path = get_package_share_directory('peripherals')
else:
# If not, use the hardcoded path to the 'src' directory (for development)
peripherals_package_path = '/home/ubuntu/ros2_ws/src/peripherals'

# --- 2. Define the Camera Node ---
# This creates the main node that will connect to the physical USB camera
# and start broadcasting its images as a ROS 2 topic.

camera_nodes = Node(
package='usb_cam',            # Use the 'usb_cam' package
executable='usb_cam_node_exe', # Run the 'usb_cam_node_exe' program from that package
output='screen',              # Print any logs from this node directly to the terminal
name='usb_cam',               # Name this running node 'usb_cam'

# This is critical: It tells the node to load all its settings
# (like which /dev/video device to use, resolution, pixel format, etc.)
# from our 'usb_cam_param.yaml' file.
parameters=[os.path.join(peripherals_package_path, 'config', 'usb_cam_param.yaml')],

# 'Remappings' are used to change topic names.
# Here, they are all commented out, so the node will use its default topic names.
# If uncommented, it would change the default 'image_raw' topic to something else.
remappings = [
#('image_raw', '/ascamera/camera_publisher/rgb0/image'),
#('image_raw/compressed', '/ascamera/camera_publisher/rgb0/image_compressed'),
#('image_raw/compressedDepth', '/ascamera/camera_publisher/rgb0/compressedDepth'),
#('image_raw/theora', '/ascamera/camera_publisher/rgb0/image_raw/theora'),
#('camera_info', '/ascamera/camera_publisher/rgb0/camera_info'),
]
)

# --- 3. Return the Launch Description ---
# This returns a list of all nodes to be launched. In this case, just the camera node.
return LaunchDescription([camera_nodes])

# --- Main Entry Point ---
# This "if" block only runs if you execute this file directly (e.g., `python3 usb_cam.launch.py`).
# It's used for testing the launch file by itself.
if __name__ == '__main__':
# Create a LaunchDescription object
ld = generate_launch_description()

# Create a LaunchService to run the launch description
ls = LaunchService()
ls.include_launch_description(ld)
ls.run()
```

Sonar_controller_node.launch.py

```
#!/usr/bin/env python3
# This is a ROS 2 launch file. Its purpose is to start the node that
# controls and reads from the sonar (ultrasonic) sensor.

# --- Imports ---
from launch_ros.actions import Node  # The class used to define a ROS 2 node in a launch file
from launch import LaunchDescription, LaunchService  # Core launch system classes
from launch.actions import DeclareLaunchArgument
from launch.substitutions import LaunchConfiguration
```

```
# --- Main Launch Function ---
# This is the primary function that ROS 2 executes when the launch file is run.
def generate_launch_description():

# --- 1. Define the Sonar Node ---
# This creates the node that will communicate with the sonar hardware.
# Based on the code you sent earlier (sonar_controller_node.py), this node
# will read the distance and publish it on the '/sonar_controller/get_distance' topic.

sonar_controller_node = Node(
package='peripherals',         # The name of the ROS 2 package to use
executable='sonar_controller', # The specific program (node) to run from that package
output='screen',               # Print any logs from this node directly to the terminal
)

# --- 2. Return the Launch Description ---
# This returns a list of all nodes to be launched. In this case, just the sonar node.
return LaunchDescription([
sonar_controller_node
])

# --- Main Entry Point ---
# This "if" block only runs if you execute this file directly (e.g., `python3
sonar_controller_node.launch.py`).
# It's used for testing the launch file by itself.
if __name__ == '__main__':
# Create a LaunchDescription object
ld = generate_launch_description()

# Create a LaunchService to run the launch description
ls = LaunchService()
ls.include_launch_description(ld)
ls.run()
```

Joystick_control.launch.py

```
#!/usr/bin/env python3
# This is a ROS 2 launch file. Its purpose is to start the nodes
# required to control the robot using a physical USB joystick/gamepad.

# --- Imports ---
from launch_ros.actions import Node  # The class used to define a ROS 2 node
from launch.actions import DeclareLaunchArgument, IncludeLaunchDescription  # Classes for
handling launch arguments and including other launch files
from launch import LaunchDescription, LaunchService  # Core launch system classes
from launch.substitutions import LaunchConfiguration  # Used to get the value of a launch argument
from launch.launch_description_sources import PythonLaunchDescriptionSource  # Used to include
other Python launch files

# --- Main Launch Function ---
# This is the primary function that ROS 2 executes when the launch file is run.
def generate_launch_description():

# --- 1. Define Launch Arguments ---
# These are variables that can be set from the command line when you run the launch file.
# This makes the launch file flexible without needing to edit the code.

# 'max_linear': Sets the maximum forward/backward speed (default: 0.5 m/s)
max_linear = LaunchConfiguration('max_linear', default='0.5')

# 'max_angular': Sets the maximum turning speed (default: 2.0 rad/s)
max_angular = LaunchConfiguration('max_angular', default='2.0')

# 'remap_cmd_vel': Allows changing the name of the final movement topic
# This is useful if the robot's motor controller listens to a different topic name.
remap_cmd_vel = LaunchConfiguration('remap_cmd_vel', default='controller/cmd_vel')

# --- 2. Declare the Arguments for the Launch System ---
# This section formally tells the launch system that the arguments from Step 1 exist.

max_linear_arg = DeclareLaunchArgument(
'max_linear', default_value=max_linear)

max_angular_arg = DeclareLaunchArgument(
'max_angular', default_value=max_angular)

remap_cmd_vel_arg = DeclareLaunchArgument(
'remap_cmd_vel', default_value=remap_cmd_vel)

# --- 3. Define the Joystick Hardware Node ---
# This node is from the standard 'joy' package. Its only job is to find the
# physical joystick hardware and publish its raw button and axis data.

joy_node = Node(
package='joy',                  # Use the 'joy' package
executable='joy_node',            # Run the 'joy_node' program
name='joy_node',                  # Name this running node 'joy_node'
output='screen',                 # Print logs to the terminal
parameters=[{
'dev': '/dev/input/js0',       # Tell the node which device file to read (the first joystick)
'autorepeat_rate': 20.0        # How often to publish data (20 Hz)
}]
)

# --- 4. Define the Joystick Control Node ---
# This is our custom node from the 'peripherals' package. It subscribes to the
# raw data from 'joy_node' and translates it into ROS 2 'Twist' messages
# (movement commands) that the robot understands.

joystick_control_node = Node(
package='peripherals',            # Use our 'peripherals' package
executable='joystick_control',     # Run our 'joystick_control' program
name='joystick_control',          # Name this running node 'joystick_control'
output='screen',                 # Print logs to the terminal
parameters=[
{
# Pass in the launch arguments as parameters for this node to use
'max_linear': max_linear,
'max_angular': max_angular,
'disable_servo_control': True  # Tell the node to *not* try to control servos
}
],
remappings=[
# This "wires" the output of this node.
# It changes its default output topic 'controller/cmd_vel'
# to whatever 'remap_cmd_vel' is set to.
('controller/cmd_vel', remap_cmd_vel)
]
)

# --- 5. Return the Launch Description ---
# This returns the final "list of things to do" for ROS 2.
return LaunchDescription([
# First, add the argument declarations
max_linear_arg,
max_angular_arg,
remap_cmd_vel_arg,

# Then, add the nodes to be launched
joy_node,
joystick_control_node
])

# --- Main Entry Point ---
# This "if" block only runs if you execute this file directly (e.g., `python3 joystick_control.launch.py`).
# It's used for testing the launch file by itself.
if __name__ == '__main__':
# Create a LaunchDescription object
ld = generate_launch_description()

# Create a LaunchService to run the launch description
ls = LaunchService()
ls.include_launch_description(ld)
ls.run()
```

The ROS 2 launch files and dashboard code were developed and commented in consultation with Google's Gemini AI (2025).

§15G: Bill of Materials

Index	Item Name	Description	Mfr	Mfr P/N	Supp
1	Raspberry Pi 5	Main Controller, Handles AI, sensors etc	Raspberry Pi	SC1431	DigiKey
2	MicroSD card 32 GB	Storage for OS and project files	Olimex LTD	MICRO-SD-32GB-CLASS10	DigiKey
3	Micro HDMI to HDMI cable	Connects PI to display	Raspberry Pi	SC0546	DigiKey
4	GRAPHIC DISPLAY OLED WHITE 0.96"	Main face for the Robot	AdaFruit	326	DigiKey
5	Raspberry Pi Camera Module 3 (12 MP, Autofocus)	Camer for vision	AdaFruit	SC1223	DigiKey
6	HC-SR04 Ultrasonic Sensor	Distance sensing to avoid collision	SparkFun Electronics	15569	DigiKey
7	7.2 V NiMH Rechargeable Battery Pack (2 Cells)	Power supply for motors	Dantona Industries	L74A26-2-1-3WA3	DigiKey
8	BATT CHARGER	Charger for battery Pack	Ultralast	ULLIONCHG	DigiKey
9	Pan/Tilt Servo (SG90)	Moves camera for tracking	Adafruit Industries LLC	1142	DigiKey
10	20-PACK MULTI-COLOR JUMPER WIRES	Wiring and clean connections	DigiKey Standard	DKS-20MF-5	DigiKey
11	Microphone and Speaker Box	for speech input and output	HiWonder	CL1302	HiWonder
12	Gearbox TT Motors	for movement of tires	Adafruit Industries LLC	5857	DigiKey
13	BLACK MECANUM WHEEL (97MM) - RIG	Robot's Tires	DFRobot	FIT0768	DigiKey
14	HiWonder Motor Driver Expansion Board	Expansion board for motor driver	HiWonder	ModelA	HiWonder
15	114pcs Assorted M2.5 Standoff Kit	For connecting the chasis to the hardware parts	Micro Connectors, Inc.	SCW-114PC	DigiKey

Index	Supp P/N	Qty	$ / item	$ total	Link to Item
1	2648-SC1431-ND	1	94.91	94.91	https://www.digikey.ca/en/products/detail/raspberry-pi/SC1431/21658261
2	1188-MICRO-SD-32GB-CLASS10-ND	1	19.93	19.93	https://www.digikey.ca/en/products/detail/olimex-ltd/MICRO-SD-32GB-CLASS10/21662024
3	2648-SC0546-ND	1	7.87	7.87	https://www.digikey.ca/short/d2rzqbdf
4	2648-SC0546-ND	1	27.55	27.55	https://www.digikey.ca/short/5zr03779
5	2648-SC1223-ND	1	39.36	39.36	https://www.digikey.ca/short/r0vw1bq9
6	1568-15569-ND	1	8.27	8.27	https://www.digikey.ca/short/r52d47qm
7	3145-L74A26-2-1-3WA3-ND	1	25.11	25.11	https://www.digikey.ca/short/z21dmtz2
8	3145-ULLIONCHG-ND	1	12.99	12.99	https://www.digikey.ca/short/8hfv02ff
9	1528-1083-ND	2	19.95	39.9	https://www.digikey.ca/short/bh2wfz8q
10	DKS-20MF-5-ND	1	10.86	10.86	https://www.digikey.ca/short/z03h45dd
11	CL1302	1	23.99	23.99	https://www.hiwonder.com/products/wonderecho
12	1528-5857-ND	4	5.5	22	https://www.digikey.ca/short/phhqwtjz
13	1738-FIT0768-ND	4	8.9	35.6	*https://www.amazon.ca/dp/B0DSLWMBKS*
14	Model A	1	39.99	39.99	https://www.hiwonder.com/products/expansion-board-for-raspberry-pi-5
15	2768-SCW-114PC-ND	1	15.99	15.99	https://www.digikey.ca/short/vjp4tvrq

Index	Link to Datasheet
1	https://datasheets.raspberrypi.com/rpi5/raspberry-pi-5-product-brief.pdf
2	N/A
3	N/A
4	https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf
5	https://datasheets.raspberrypi.com/camera/camera-module-3-product-brief.pdf
6	https://www.alldatasheet.com/datasheet-pdf/pdf/1132204/ETC2/HCSR04.html
7	https://dantona.com/products/l74a26-2-1-3wa3/
8	https://dantona.com/products/ullionchg/
9	https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/2203/1142_Web.pdf
10	https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/6658/DKS_JumperWireKits_DS.pdf
11	https://docs.hiwonder.com/projects/WonderEcho/en/latest/docs/1_Module_Introduction.html#working-principle
12	https://media.digikey.com/pdf/Data%20Sheets/Adafruit%20PDFs/3777_Web.pdf
13	N/A
14	N/A
15	https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/5692/SCW-114PC.pdf
Toal (In Cad)	424.32

§15H: Contact Information

- **PM 1:** Hemant Vohra
  Seneca ID: 149116220
  Seneca Email: hvohra2@myseneca.ca
  Phone Number: +1 (647) 594-2422

- **PM 2:** Saket Chahal
  Seneca ID: 1550140221
  Seneca Email: schahal18@myseneca.ca
  Phone Number: +1 (437) 566-2106