PURDUE
UNIVERSITY®

Department of Computer and Electrical Engineering

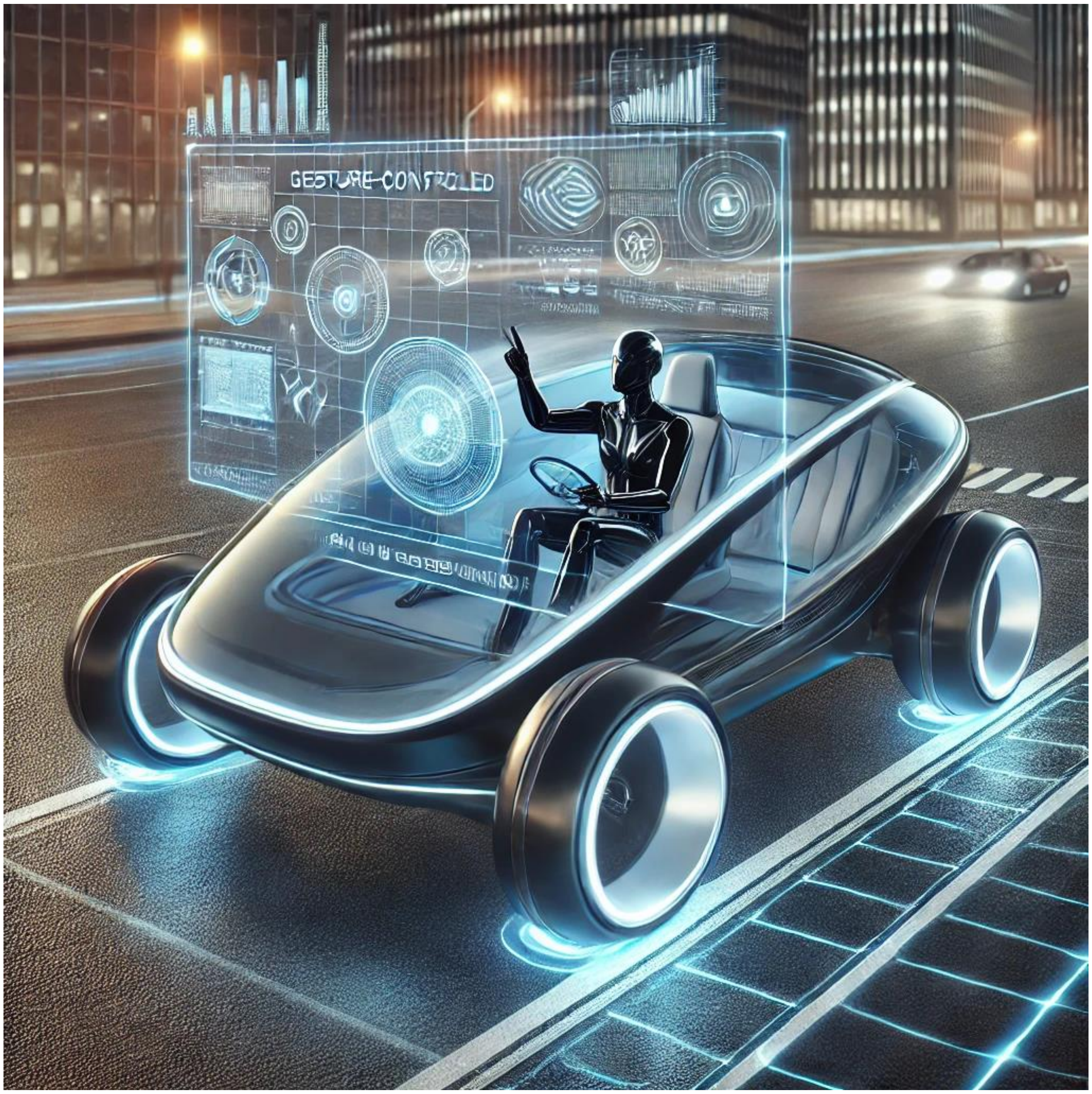
Indianapolis, Indiana

May 2025

Professor: Greg Sturm

HandiCar Team

Gesture Operated Vehicle System (GOVS)



Gesture Operated Vehicle System (GOVS)

HandiCar Team Members

Name	Email
Amit Suryadevara	avsuryad@purdu.edu
Biak Par	bpar@purdue.edu
Divya Patel	pate2130@purdue.edu
Maha Ali	ali208@purdue.edu
Martine Cardichon	mcardich@purdue.edu
Yamini Sri Krubha	ykrubha@purdue.edu

Final Submission Check List

Project Title: Gesture Operated Vehicle System (GOVS) aka The HandiCar

Project Sponsor Name: Purdue University **email:** gsturm@purdue.edu

Each group member's credit for this project:

Name: Maha Ali	credit: 16.67%
Name: Biak Par	credit: 16.67%
Name: Martine Cardichon	credit: 16.67%
Name: Yamini Sri Krubha	credit: 16.67%
Name: Divya Patel	credit: 16.67%
Name: Amit Suryadevara	credit: 16.67%

Notes: Everyone gives their 100%.

1. **Before the written deliverables deadline specified by the instructor (5/1), we have submitted, via Brightspace:**

   Yes ___ The Final Project Report

   Yes ___ The Demonstration Video

   Yes ___ The Team Project Notebook

   Yes ___ The Poster in PDF format

   Yes ___ All other items, in PDF format, generated in the course or referenced in the written deliverables (all software source code, copies of references, manuals, schematics, PCB design files, etc.)

Submission Date: ______05/01/2025__________________________

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

EXECUTIVE SUMMARY

Project Overview

The HandiCar project is a gesture-operated vehicle system designed for hands-free control using an MPU6050 accelerometer and gyroscope sensor. The system translates hand movements into wireless commands via Bluetooth, enabling precise robotic vehicle control. It is particularly beneficial for individuals with physical disabilities and has applications in assistive mobility, industrial automation, and smart transportation. The system consists of gesture input, processing, communication, control, motion, and power management. Real-time obstacle detection ensures smooth navigation, while LED indicators provide real-time feedback on motion and system status. The car is designed to recognize commands for forward, backward, left, right, and stop, with an emergency stop function activated when obstacles are detected within a defined range.

The project utilizes a microcontroller-based architecture integrated with open-source software, ensuring cost-effective implementation and scalability. The software optimizes motor control algorithms, distinguishing between valid gestures and unintended movements. The system also features voice command integration as a backup control mechanism, enhancing accessibility and usability. Compliant with ISO standards for gesture recognition, obstacle detection, and software quality, the HandiCar project prioritizes safety and efficiency. Future improvements include AI-driven gesture refinement, increased autonomy, and advanced obstacle avoidance techniques. With potential applications in autonomous vehicles and assistive robotics, HandiCar represents a step forward in intuitive human-machine interaction.

Stakeholders

Divya (Communication Leader, Design Lead)

Divya serves as the Communication Leader and one of the Design Leads. She manages internal team updates and ensures clear communication between team members. Divya also leads system-level design discussions, helping define how subsystems interact and ensuring the project's architectural integrity is maintained across stages.

Biak Par (Hardware Designer, Project Manager)

As Project Manager, Biak is responsible for coordinating team tasks, maintaining project documentation, scheduling activities, and overseeing the procurement process to ensure steady project progress. In addition to management responsibilities, Biak supports hardware integration by assisting with system testing to help keep the technical development on track.

Amit (Hardware Designer, Project Manager, Budget Manager)

Amit takes on multiple responsibilities as a Hardware Designer, Project Manager, and Budget Manager. He contributes to physical assembly, motor integration, and chassis setup. Additionally, he manages the project by ensuring components are both cost-effective and meet technical requirements.

Maha (Hardware Designer, Design Lead, Budget Manager)

Maha contributes to the project by handling hardware integration tasks such as wiring, component setup, and chassis assembly. She selects and configures essential modules to ensure reliable performance and manages power distribution across the system. Maha also oversees the project budget, ensuring components are both cost-effective and technically suitable. She works closely with the team during testing and troubleshooting to meet project objectives.

Yamini (Software Developer, Project Manager)

Yamini serves as a Software Developer and Project Manager. She develops core software logic for gesture interpretation and vehicle control. In her project management role, she coordinates task distribution and ensures that software milestones align with the overall timeline and requirements.

Martine (Software Developer)

Martine contributes as a Software Developer, focusing on the integration of gesture-based inputs with vehicle actuation. She is responsible for debugging and testing embedded code that enables real-time responsiveness. She also oversees 3D modeling aspects of the project.

FINAL PROJECT REPORT

Project Motivation

Traditional vehicle control methods—such as joysticks, buttons, and smartphone apps—have several limitations when it comes to usability, accessibility, and adaptability. These systems often require fine motor skills and good hand-eye coordination, which can make them difficult or impossible to use for individuals with physical impairments. In high-pressure or fast-changing environments like emergency response or assistive mobility, these controls may be too slow or unintuitive, reducing user safety and overall effectiveness. Additionally, people who are unfamiliar with technology may struggle to operate these systems, leading to frustration and reduced adoption.

Project Background Information

Several control systems already exist to operate vehicles remotely, including joystick-based remotes, smartphone applications, and even voice-controlled systems. While these approaches have been widely used, they each come with significant limitations. Joysticks and buttons require fine motor skills, making them inaccessible to individuals with physical impairments. Smartphone apps, although common, often rely on touchscreens that are not always responsive or intuitive during dynamic use. Voice-controlled systems offer hands-free control but tend to perform poorly in noisy environments and may not support complex directional input. Vision-based gesture recognition systems using cameras and computer vision libraries such as OpenCV provide more natural interaction, but they typically require high processing power, controlled lighting conditions, and are expensive to implement, especially for real-time mobile applications.

While recent advancements in microcontrollers, motion sensors (e.g., accelerometers and gyroscopes), and wireless communication modules (like the HC-05 Bluetooth or NRF24L01) have made gesture-based systems more feasible and affordable, these technologies have not been fully leveraged to create low-cost, reliable, and accessible vehicle control solutions. Existing systems often lack adaptability for diverse users or require significant technical expertise to implement. This project addresses that gap by developing a gesture-operated vehicle system that combines these modern, affordable technologies into a practical solution aimed at improving accessibility, enhancing usability, and supporting applications in assistive technology, automation, and education.

System Block Diagram

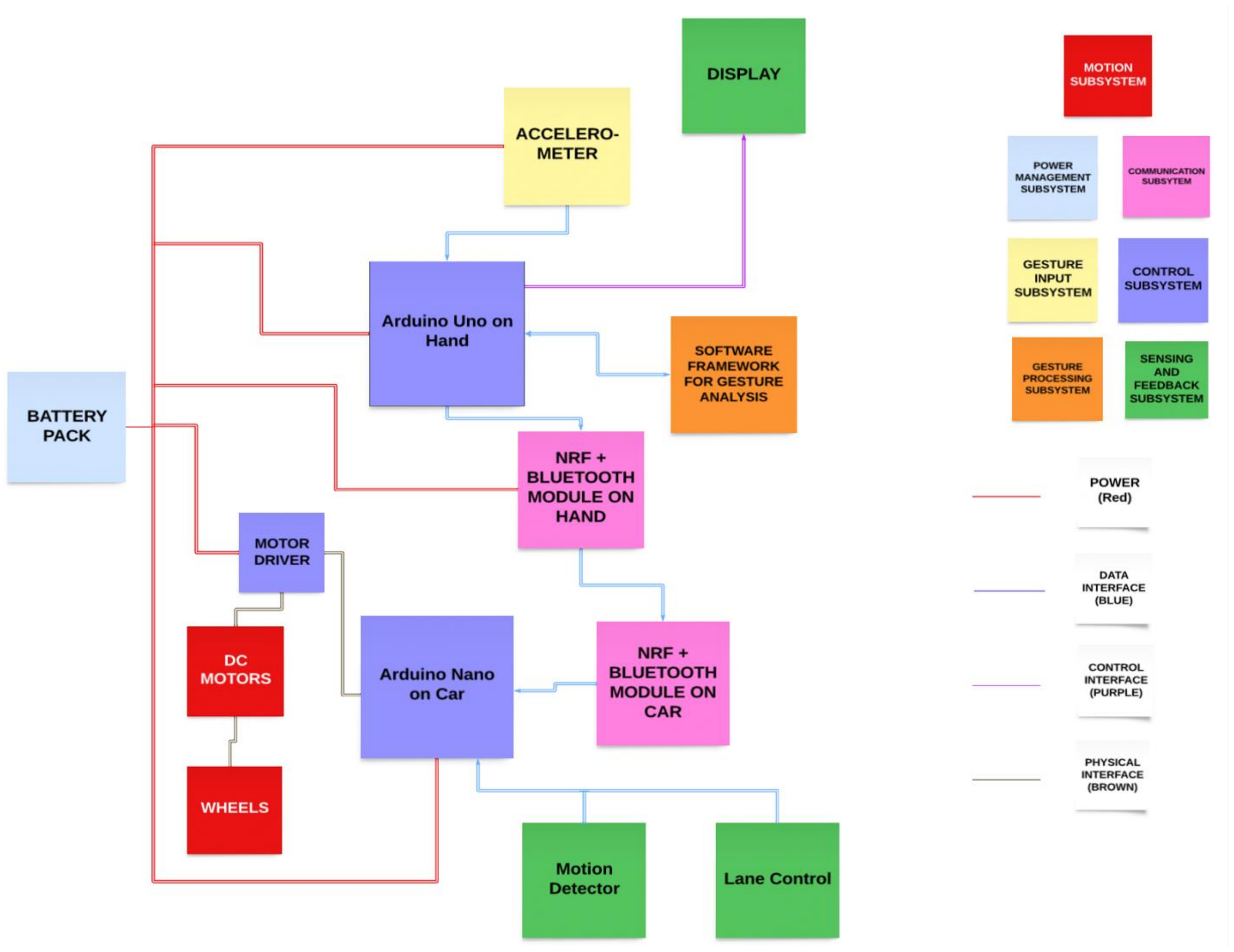


Figure 1: GOVS System Architecture

Project Requirements

Technical Requirements

The system is designed to respond precisely and consistently to hand gestures, ensuring the vehicle moves intuitively based on user input. The gesture controller will communicate wirelessly with the car via Bluetooth, enabling real-time bidirectional communication. To enhance interactivity, the controller will provide visual feedback, displaying motion-related indicators such as speed, stop, brake status, and nearby object detection. The vehicle is expected to meet performance standards, including a minimum speed of 8 mph, and feature individually motorized wheels for improved control. Additional functionalities include gesture-based activation of special features like Dance Mode, with synchronized LED blinking and audio output via an onboard speaker. The software component must be robust, capable of distinguishing valid gestures from accidental movements, while logging system uptime and resource usage. It will also alert users to critical errors or nearby obstacles and support voice commands as an alternative control method for accessibility and redundancy.

Standards Requirements

To ensure the system meets international expectations for usability, accessibility, and consistency in human-machine interaction, our project aligns with **ISO/IEC 30113-1:2025**, a standard specifically focused on gesture-based interfaces. This standard defines how gesture inputs should be structured, processed, and interpreted to ensure reliable and user-friendly control in both general and assistive environments.

In our system, the **MPU6050 sensor** is the critical component that enables compliance with this standard. The MPU6050 includes both a 3-axis accelerometer and 3-axis gyroscope, allowing it to capture dynamic hand gestures—such as tilts, rotations, and directional motions—in real time. By applying structured preprocessing and calibration techniques, we ensure that the data it generates aligns with the standard's expectations for gesture consistency and repeatability.

For example, ISO/IEC 30113-1:2025 emphasizes that gestures must be distinct and reliable across users and environments. To meet this, we implemented dead zones and noise filtering to eliminate unintended gestures and used threshold detection to map specific motion patterns (like a forward tilt) to defined actions (e.g., move forward).

Aligning with this standard ensures that our gesture recognition subsystem, powered by the MPU6050, offers an intuitive and accessible interface, particularly for users who may have physical limitations or require assistive technologies. It also ensures system interoperability and supports future scalability in smart environments where gesture control must be reliable and universally understood.

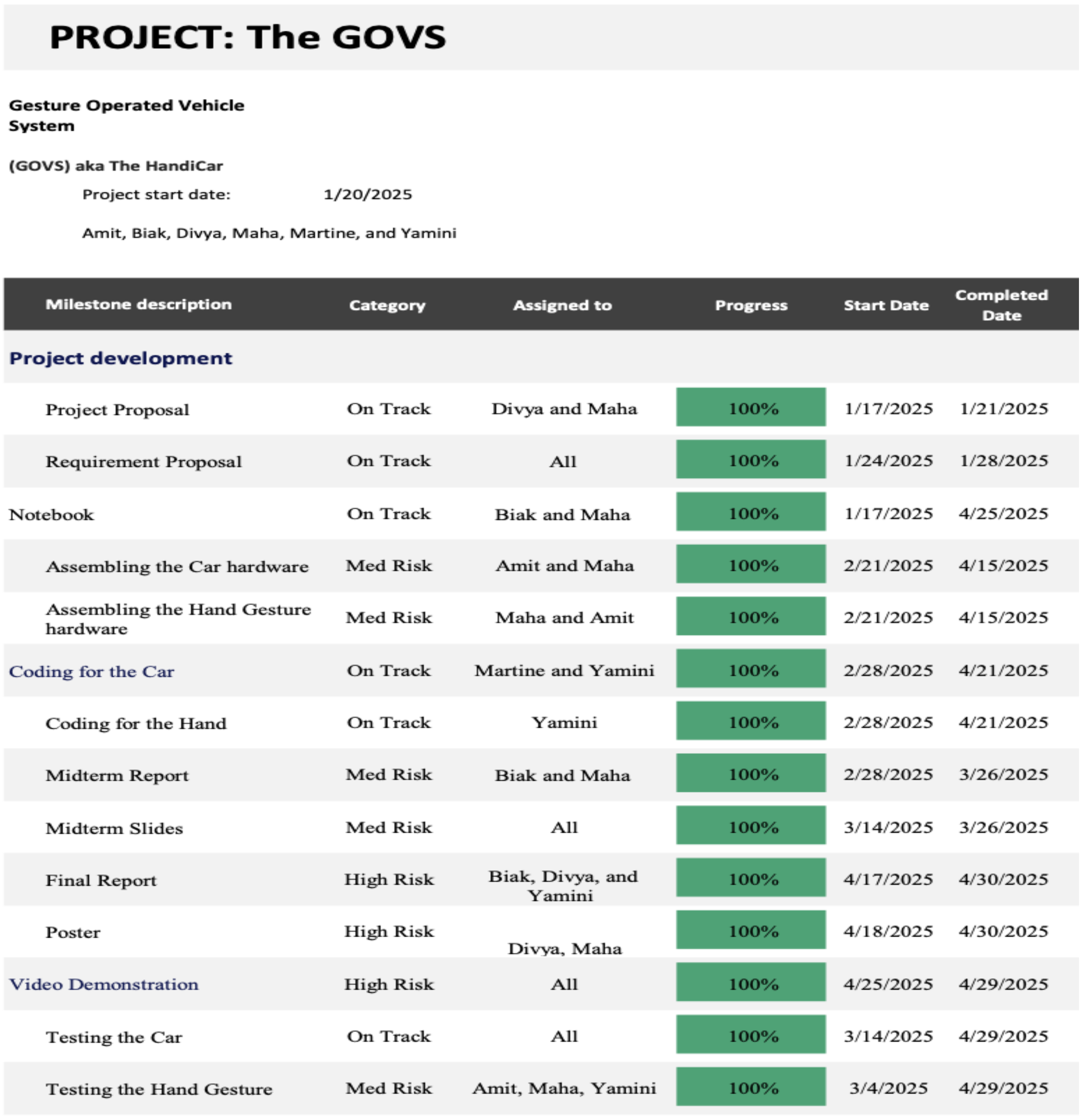
Safety Requirements

Safety is a central design consideration in both hardware and software components. The system will include a built-in emergency stop function, activated automatically if the vehicle fails to reroute successfully after multiple attempts. Furthermore, a low-battery warning will trigger when the power level drops below 10%, allowing users to take preventive action before total shutdown. These safety mechanisms ensure the system operates reliably and avoids harm to users or the environment.

Fault Tolerance

To improve robustness and reliability, the system will incorporate fault-tolerant features. These include filtering techniques to ignore unintended or noisy gesture inputs and redundant sensor setups that ensure reliable operation even if one sensor fails. Automatic recalibration mechanisms will correct sensor inconsistencies without requiring user intervention. In situations where gesture control becomes unreliable, the system will offer fallback options, including voice and touch-based control, ensuring continuous functionality under varying conditions.

Schedule Requirements



PROJECT: The GOVS

Gesture Operated Vehicle System

(GOVS) aka The HandiCar

Project start date:	1/20/2025

Amit, Biak, Divya, Maha, Martine, and Yamini

Milestone description	Category	Assigned to	Progress	Start Date	Completed Date
Project development					
Project Proposal	On Track	Divya and Maha	100%	1/17/2025	1/21/2025
Requirement Proposal	On Track	All	100%	1/24/2025	1/28/2025
Notebook	On Track	Biak and Maha	100%	1/17/2025	4/25/2025
Assembling the Car hardware	Med Risk	Amit and Maha	100%	2/21/2025	4/15/2025
Assembling the Hand Gesture hardware	Med Risk	Maha and Amit	100%	2/21/2025	4/15/2025
Coding for the Car	On Track	Martine and Yamini	100%	2/28/2025	4/21/2025
Coding for the Hand	On Track	Yamini	100%	2/28/2025	4/21/2025
Midterm Report	Med Risk	Biak and Maha	100%	2/28/2025	3/26/2025
Midterm Slides	Med Risk	All	100%	3/14/2025	3/26/2025
Final Report	High Risk	Biak, Divya, and Yamini	100%	4/17/2025	4/30/2025
Poster	High Risk	Divya, Maha	100%	4/18/2025	4/30/2025
Video Demonstration	High Risk	All	100%	4/25/2025	4/29/2025
Testing the Car	On Track	All	100%	3/14/2025	4/29/2025
Testing the Hand Gesture	Med Risk	Amit, Maha, Yamini	100%	3/4/2025	4/29/2025

Other Requirements

To optimize energy efficiency, the system will include a low-power mode that activates automatically when the vehicle or controller remains idle for more than five minutes. This feature ensures better battery management and longer operational time, particularly important in field applications or extended usage scenarios where charging opportunities may be limited.

Constraints

The development of the Gesture Operated Vehicle System (GOVS) was subject to strict academic timelines and weekly deadlines. Each stage of implementation—sensor integration, wireless communication, motor control, and system testing—required careful time management. Unforeseen issues such as debugging communication delays or hardware incompatibility occasionally extended expected durations. Additionally, team availability, especially during exam weeks or holidays, impacted progress and required rescheduling of tasks. These time constraints highlighted the need for early planning and efficient task delegation within the team.

Budget limitations played a major role in component selection. While advanced sensors and high-performance microcontrollers were considered, affordability became a key decision-making factor. Components like the MPU6050 and NRF24L01 were chosen for their balance between cost and functionality. All hardware had to be sourced within a constrained budget, which limited our ability to explore more advanced or redundant systems. Financial constraints also meant relying on available lab tools and open-source libraries to reduce software development costs.

The scope of the project was initially defined to include gesture-based vehicle control with basic feedback and motion. However, as the team progressed and discovered additional possibilities—like obstacle avoidance, emergency stop features, and dance mode—the scope gradually expanded. While these additions enhanced the system's capabilities, they also introduced greater complexity and increased development time. This dynamic scope required continuous reassessment to ensure the project remained achievable within academic and resource limits.

Design Options

Design Option Analysis

Gesture Sensor Selection – MPU6050 vs. APDS-9960

We selected the **MPU6050** for our system due to its superior responsiveness, orientation flexibility, and real-time tracking capabilities. Unlike the APDS-9960, which detects only discrete gestures and requires a clear line of sight, the MPU6050 provides continuous motion data and functions reliably under various lighting conditions. Its ability to track rotation and acceleration on all axes enables precise control that's essential for intuitive vehicle navigation. Although it consumes slightly more power, the enhanced performance in responsiveness and environmental flexibility makes it the most suitable choice for a gesture-operated system.

Programming Language – C++ vs. Python

C++ was chosen for system development due to its fast execution speed, low latency, and direct hardware access—critical features for real-time embedded systems like ours. While Python offers faster development time and is ideal for high-level tasks or AI integration, it introduces latency and requires additional libraries for hardware interaction. C++'s compatibility with platforms like Arduino and STM32, along with its efficiency in memory and power management, makes it better suited for our low-power, high-performance needs.

Chassis Type – Mecanum vs. Standard 4WD

We opted for the **Mecanum omnidirectional chassis** due to its advanced maneuverability and full 360° motion capabilities. This allows the vehicle to move sideways, rotate in place, and make precise adjustments—key for effective gesture-based control. Although it requires more complex motor programming and offers lower torque compared to standard 4WD, the improved navigation control and suitability for indoor assistive environments make it the preferred choice for our application.

Communication Module – NRF24L01 vs. Bluetooth

For wireless communication, the **NRF24L01** was selected over the Bluetooth module due to its long range, high data rate, and low latency. Bluetooth suffers from higher power consumption and interference issues in busy environments. In contrast, NRF24L01 provides reliable, multi-channel communication and supports mesh networking if needed. While its setup is slightly more complex, its superior performance in real-time responsiveness and robustness makes it the ideal choice for ensuring seamless communication between the controller and vehicle.

Design Details

Hardware Design Details

Hardware Design Option – Chassis Selection and Motor Control System

A key design decision in the Gesture Operated Vehicle System (GOVS) was the selection of a **Mecanum wheel omnidirectional chassis** (see *Figure 1*) to achieve smooth, responsive, and multidirectional movement. Unlike a standard four-wheel drive (4WD) chassis, the Mecanum chassis, with its 45-degree angled rollers, allows the vehicle to move forward, backward, sideways, diagonally, and rotate in place, making it highly adaptable for tight or indoor spaces. This flexibility aligns perfectly with the project's goal of enabling intuitive, real-time gesture-based control.

To drive this chassis, the **L298N Motor Driver Module** (*Figure 2*) was used to manage the power and control requirements of the dual DC motors. The L298N is a dual H-bridge motor driver that enables bidirectional motor control and speed regulation through pulse width modulation (PWM). It acts as an interface between the Arduino Nano and the motors, safely handling high current demands that the Arduino alone could not manage. Powered by an external 12V battery and sharing common ground with the Arduino, the L298N's outputs (OUT1–OUT4) connect to the motors, with control pins wired as follows: ENA to D5 (PWM), IN1 to D2, IN2 to D3, IN3 to D4, IN4 to D7, and ENB to D6 (PWM), as shown in *Figure 3*. This setup allows precise speed and direction control, enabling the vehicle to perform omnidirectional movements smoothly in response to hand gestures. Together, the chassis selection and motor control system form a critical foundation for achieving the project's objectives of responsive, accessible, and flexible gesture-operated mobility

.

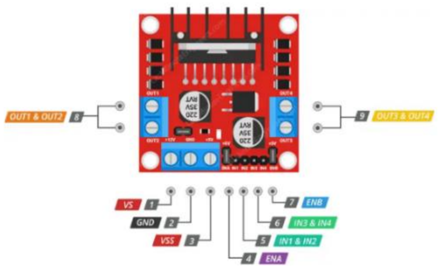


Figure 1: Omnidirectional Chassis Figure2: Motor Module L298N

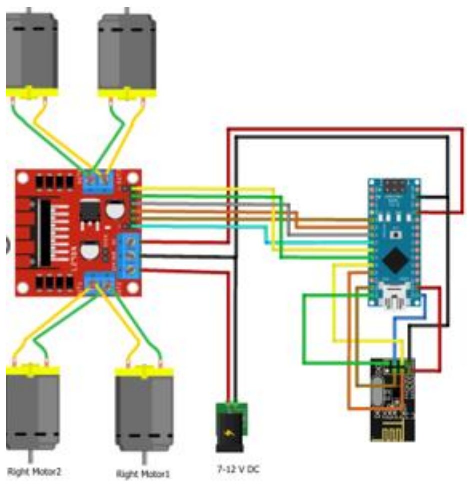


Figure 3: Chassis, Motor Control System Connections

Hardware Design Option – MPU6050 Motion Sensor

As part of the gesture recognition system in GOVS, the **MPU6050 motion sensor** (*Figure 4*) was selected for its ability to detect both linear and rotational movement with high precision. The MPU6050 integrates a 3-axis accelerometer and a 3-axis gyroscope, allowing it to capture full-motion orientation data that is critical for interpreting dynamic hand gestures. The sensor communicates with the Arduino Nano through the I2C interface, which enables efficient two-wire data transmission. The connections are made as follows: the VCC pin of the MPU6050 is connected to the 5V pin of the Arduino Nano, GND is connected to GND, SCL is connected to analog pin A5, and SDA is connected to analog pin A4, as shown in *Figure 5*. This setup ensures continuous, real-time motion data acquisition, allowing the system to accurately translate hand gestures into vehicle control commands.

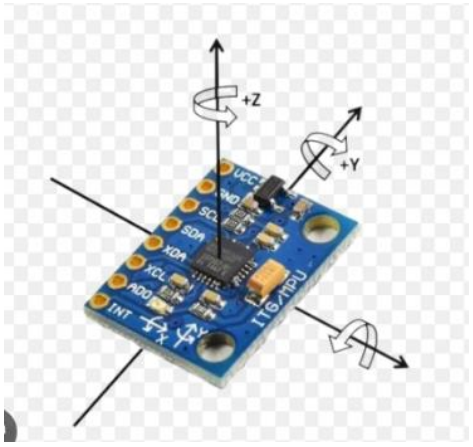


Figure 4: MPU6050

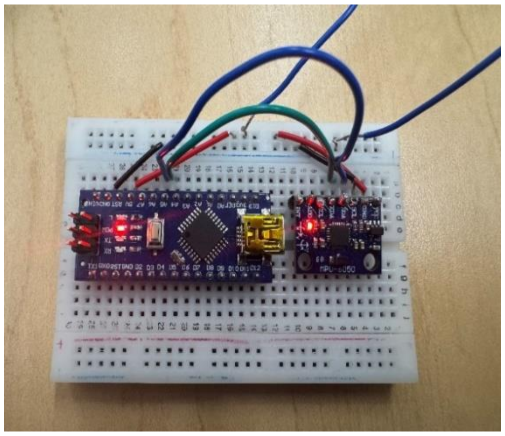
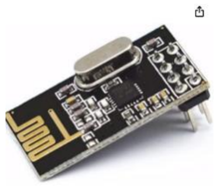


Figure 5: MPU6050 and Arduino Connection

Hardware Design Option- Wireless Communication

For wireless communication between the gesture control unit and the vehicle, the **NRF24L01 RF transceiver module** (*Figure 6*) was selected over traditional Bluetooth modules such as the HC-05. The NRF24L01 enables reliable, low-latency, and long-range data transmission, which is critical for real-time control in gesture-based vehicle systems. In this system, a gesture sensor (such as an accelerometer or RGB gesture sensor) captures hand motion data and sends it to a microcontroller, which then processes the data and transmits it using the NRF24L01 module. On the vehicle side, another NRF24L01 module receives the data and forwards it to a second microcontroller that controls the vehicle's movement and response.
The NRF24L01 uses the SPI (Serial Peripheral Interface) protocol to communicate with the Arduino Nano, connecting to digital pins D8 through D13—D11 (MOSI), D12 (MISO), and D13 (SCK) handle SPI communication, while D9 (CE) and D10 (CSN) manage module control. The module is powered using the Arduino's 3.3V output (not 5V), and a 10µF decoupling capacitor is recommended across VCC and GND to stabilize the power supply. The pin mapping to the Arduino Nano is shown in *Figure 7*.
Compared to Bluetooth modules like the HC-05, the NRF24L01 offers longer range, higher data rate (up to 2 Mbps), lower latency, and improved reliability. These features make the NRF24L01 the ideal choice for GOVS. The overall **data transmission process** from gesture detection to vehicle control is illustrated in *Figure 11* and *Figure 12*.



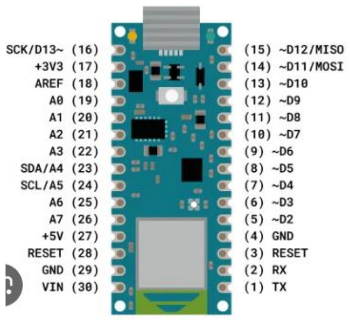
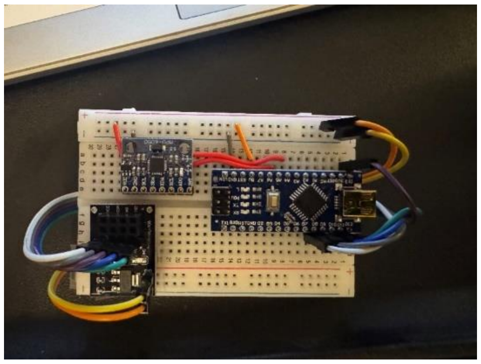


Figure 6: NRF24L01 Figure 7: Arduino Nano



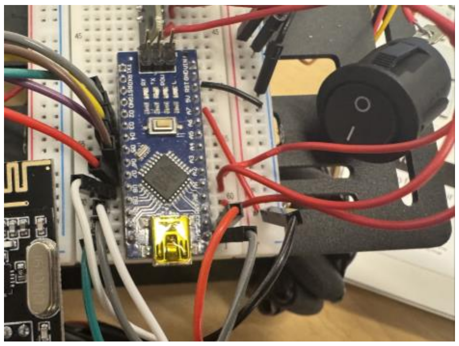


Figure 11: Hand Gesture Data Transmission Process

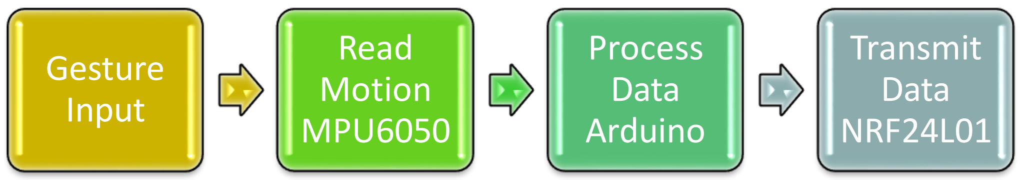


Figure 11: Hand Gesture Data Transmission Process

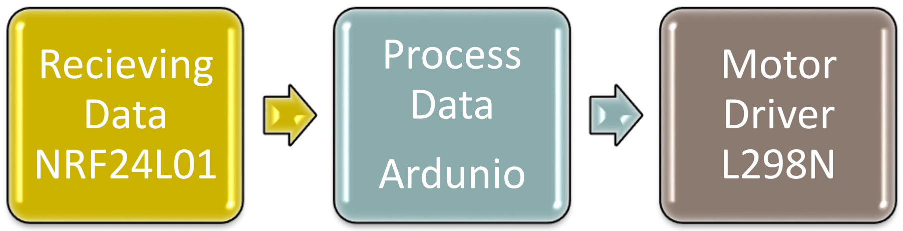


Figure 12: Car Gesture Transmission Process

Hardware Design Option Obstacle Detection and Scanning System

To enhance obstacle detection and environmental awareness in GOVS, an ultrasonic sensor (HC-SR04) paired with a servo motor was used. The ultrasonic sensor measures the distance to obstacles by emitting pulses via the Trig pin and detecting the echo via the Echo pin, enabling real-time object avoidance. The sensor is connected to the Arduino Nano: Trig to A2, Echo to A3, VCC to an external battery, and GND shared with the Arduino. Mounted on a servo motor, the sensor rotates to scan different directions. The servo motor receives its control signal from A0, powered by the same external battery, and shares GND with the rest of the system. This connection ensures power stability and prevents voltage conflicts with the NRF24L01 module. Together, this setup enhances real-time obstacle detection, improving safe navigation through gesture control.Software Design Details

Gesture Input Subsystem

The gesture input subsystem, corresponding to the "Read Gesture Input" box in the flowchart, utilizes the MPU6050 sensor with its built-in Digital Motion Processor (DMP) to capture the user's hand movements. In the code, this is achieved through the ‘mpu.dmpGetCurrentFIFOPacket()’ function, which reads quaternion data and then converts it into yaw, pitch, and roll angles (‘ypr’). These angles are then mapped to control values, such as tilting forward or backward for speed control and tilting left or right for steering direction. The "No" path in the flowchart accounts for invalid or absent gesture input, while the "Yes" path leads into the command processing sequence for further action.

Gesture Processing Subsystem

The gesture processing subsystem, shown as "Process Gesture?" in the flowchart, interprets the raw sensor data into actionable driving commands. In the code, specific gestures correspond to defined actions: a stop command is recognized when the pitch and roll angles are near zero (‘abs(ypr[1]) < 0.2’), while a brake is triggered when a backward tilt is detected (‘ypr[1] < -0.3’). For speed control, a forward tilt (measured when ‘yValue > 10’) is mapped to a speed range between ‘MIN_SPEED’ and ‘MAX_SPEED’. These processed values are organized and stored inside a ‘PacketData’ structure, which holds fields like ‘speedCommand’ and ‘commandFlags’ for communication to the car.

Communication Subsystem

The communication subsystem, illustrated by the "Transmit via nRF24L01" and "Receive Feedback from Car" boxes in the flowchart, enables wireless data transfer between the controller and the car. Using the ‘RF24’ library, the controller sends the structured ‘PacketData’ to the car using the ‘radio.write()’ function. After executing the received command, the car responds with a ‘FeedbackData’ packet containing information like obstacle detection status, battery level, and system health, which the controller reads via ‘radio.read()’. This feedback mechanism ensures that real-time system information is displayed on the OLED and also triggers specific actions, like exiting low-power mode when activity is detected.

Control Subsystem

The control subsystem is directly linked to the "Control Motors" and "Avoid Obstacle" steps in the flowchart, orchestrating the actual movement and safety behavior of the car. In the car’s code, motor actions are directed by the ‘rotateMotor()’ function, which manipulates speed and direction through PWM signals sent to the motor driver pins. Obstacle avoidance is managed through the ‘handleObstacle()’ function, which commands the servo and alters motor behavior when the ‘detectObstacle()’ function identifies nearby obstacles. Additionally, safety routines such as the ‘emergencyStop()’ function are triggered in response to conditions like low battery levels or multiple obstacle encounters.

Motion Subsystem

The car’s motion control, forming part of the flowchart’s execution logic, is handled primarily through the ‘rotateMotor()’ function. This function sets motor directions using ‘rightMotorPin1/Pin2’ and ‘leftMotorPin1/Pin2’, and controls the motors’ speed via PWM outputs to the ‘enableRightMotor’ and ‘enableLeftMotor’ pins. The car also continuously monitors its current speed for reporting purposes by calibrating the ‘currentSpeed’ variable. For situations where a brake gesture is recognized, the ‘applyBrakes()’ function provides a gradual deceleration to ensure smooth stops instead of sudden halts, enhancing user control and system stability.

Power Management Subsystem

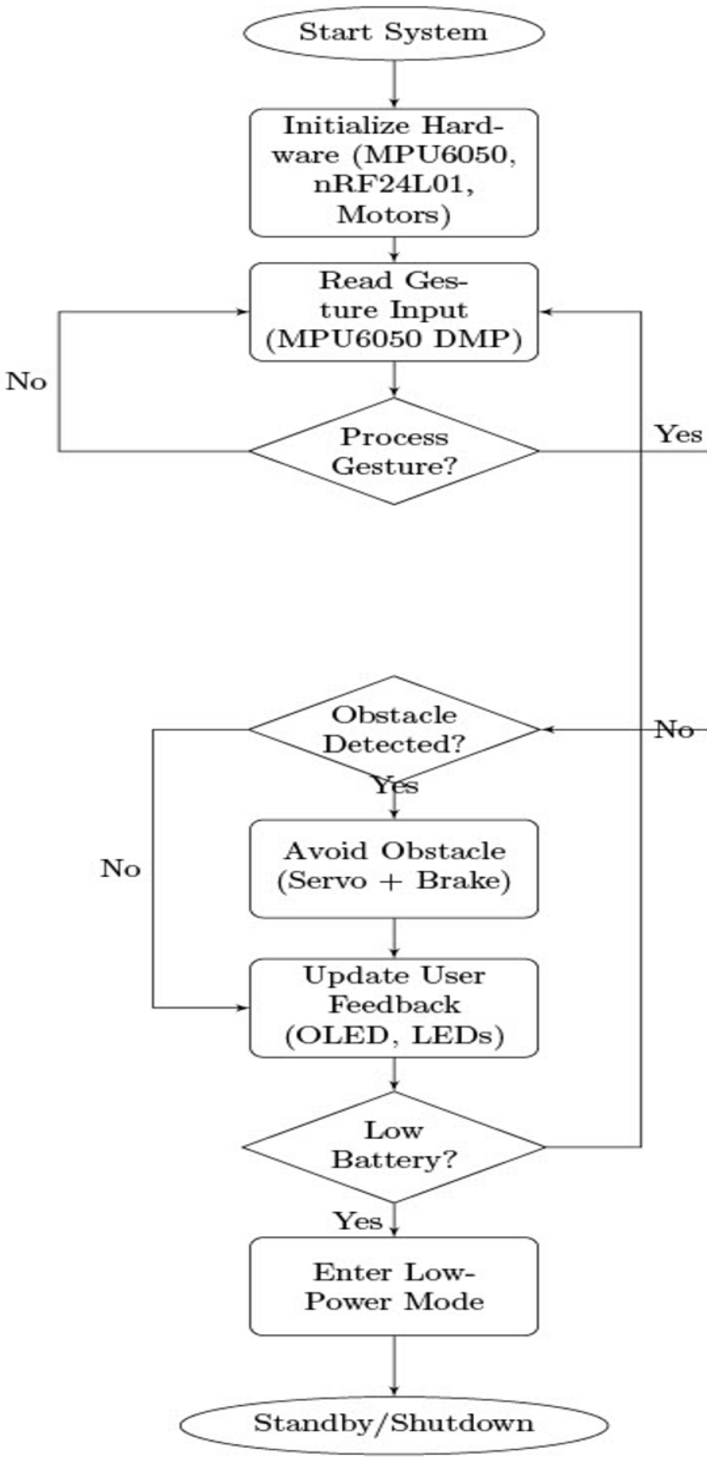
The power management subsystem, corresponding to the "Low Battery?" and "Enter Low-Power Mode" decisions in the flowchart, ensures the car and controller manage their energy efficiently. Within the car’s code, a simulated battery level (‘simulatedBatteryLevel’) is decremented periodically to mimic real-world battery drain. When the battery drops below a critical threshold, the car automatically triggers low-power behaviors. On the controller side, inactivity for a prolonged period (specifically, five minutes) results in entering low-power mode using the ‘enterLowPowerMode()’ function. This disables the radio and OLED to conserve energy until a significant gesture or movement reactivates the system.

Sensing and Feedback Subsystem

The sensing and feedback subsystem covers obstacle detection and system monitoring, aligning with the "Obstacle Detected?" and feedback update sections in the flowchart. Using an ultrasonic sensor (HC-SR04), the ‘detectObstacle()’ function measures distances to objects in front of the car. If an obstacle is within a critical range, the system can trigger an avoidance maneuver or an emergency stop. Additionally, the car monitors overall system health, including uptime, battery voltage, and motor operation errors, packing this data into ‘FeedbackData’ structures that are transmitted back to the controller for user display and internal checks.

User Feedback Subsystem

The user feedback subsystem, shown as "Update User Feedback" in the flowchart, focuses on keeping the user informed of the system’s status. The controller uses the ‘updateDisplay()’ function to refresh the OLED screen, providing real-time updates on speed, battery status, and any system warnings. Simultaneously, the car employs visual indicators via RGB LEDs, handled by the ‘updateLED()’ function. For instance, a red LED signals that the car has stopped, while a blinking yellow LED indicates the detection of an obstacle. These intuitive feedback mechanisms ensure that the user can easily understand the car’s state without needing to interpret raw sensor or debug data.


Start System
Initialize Hardware (MPU6050, nRF24L01, Motors)
Read Gesture Input (MPU6050 DMP)
No
Process Gesture?
Yes
Transmit Command via nRF24L01
Receive Feedback from Car
Control Motors (PWM, Direction)
No
Obstacle Detected?
Yes
No
Avoid Obstacle (Servo + Brake)
Update User Feedback (OLED, LEDs)
Low Battery?
Yes
Enter Low-Power Mode
Standby/Shutdown

Integration Workflow

All subsystems are interconnected to form a cohesive gesture-controlled system. The **Gesture Input Subsystem** collects hand motion, which is interpreted by the **Gesture Processing Subsystem**. Commands are transmitted via the **Communication Subsystem**, received and decoded by the **Control Subsystem**, and translated into physical motion by the **Motion Subsystem**. The **Sensing and Feedback Subsystem** works alongside the control unit to monitor surroundings and provide real-time alerts, while the **Power Management Subsystem** ensures energy efficiency. Together, these components enable smooth, reliable operation of the vehicle based on natural human input.

Implementation Details

The Gesture Operated Vehicle System is implemented through a continuous real-time chain starting from hand motion input and ending with intelligent vehicle movement and feedback display. When the user moves their hand, the **MPU6050 accelerometer and gyroscope sensor** mounted on the hand controller measures changes in orientation along three axes. The **pitch** value (y-axis) captures forward and backward tilting, while the **roll** value (x-axis) captures left and right tilting. These axis readings are processed onboard the **Arduino Uno** connected to the sensor, using the MPU's built-in **Digital Motion Processor (DMP)** to produce stable yaw, pitch, and roll data, which is read via the I2C protocol. As soon as a motion is detected, the Arduino software evaluates the axis values: a significant positive pitch (forward tilt) triggers a forward movement command, a significant negative pitch (backward tilt) triggers a reverse command, a positive roll (right tilt) commands a right turn, and a negative roll (left tilt) commands a left turn. If both pitch and roll values are close to zero, indicating a flat hand, the Arduino immediately triggers a **brake command** to stop the vehicle. Alongside command generation, the Arduino updates the vehicle's status lighting: when moving (any tilt direction), the **green LED** is turned ON, during braking triggered by a flat hand, the **yellow LED** activates, and when the vehicle is stopped or encounters an emergency, the **red LED** is illuminated.

Once the command packet, containing the mapped axis values, speed commands, and control flags, is generated, it is sent wirelessly through an **NRF24L01 transmitter** mounted on the remote. This packet travels over a dedicated RF communication channel and is received by the **NRF24L01 receiver** module mounted on the car chassis. An **Arduino Nano** connected to the receiver immediately reads the incoming data and processes it through onboard software. The Nano deciphers the x- and y-axis values to determine the intended motion and then drives the four **individually controlled DC motors** through a **dual H-bridge L298N motor driver**. Depending on the interpreted command, the Nano activates the motors accordingly: if forward motion is intended (positive y-axis), both sets of wheels are driven forward; if backward motion is indicated (negative y-axis), both motors are reversed; if a right tilt is commanded (positive x-axis), the left wheels move forward and the right wheels move backward to pivot the car right; similarly, a left tilt (negative x-axis) causes a pivot left. In the case of a flat hand detected, all motors are immediately stopped to apply braking, and the **yellow LED** is triggered.

Simultaneously, the vehicle's Arduino continuously operates an **ultrasonic distance sensor** mounted on a **servo motor** that sweeps side to side. The ultrasonic sensor precisely measures the distance to nearby objects by sending out sound waves and calculating the time taken for echoes to return. The software specifically checks if an object is detected within **60 centimeters** — a threshold set carefully to ensure accurate, reliable obstacle detection without false positives. If an object is detected within this 60 cm range, the car's Arduino immediately pauses the motors, and the **yellow LED** switches on to indicate caution. The vehicle then attempts a rerouting sequence: it stops briefly, steers to right using the motors, and tries to proceed forward again. If obstacles persist after two rerouting attempts, the vehicle triggers an **emergency stop**, fully shutting down all motor movement, and switches the **red LED** ON or blinking to visually alert the user.

In parallel, the Arduino Nano continuously collects live feedback, including the vehicle's **current speed** (estimated based on motor PWM output and a calibration factor), **battery level** (simulated and decremented over time), **obstacle detection status**, **system uptime**, and **emergency stop conditions**. This feedback data is packaged into a response packet and sent back wirelessly through the **NRF24L01** to the remote Arduino Uno. Upon receiving the feedback, the Uno processes the information and updates the **OLED display** mounted on the hand controller. The display shows the latest speed in mph, battery percentage, braking status, obstacle detection alerts, system uptime in hours/minutes/seconds, and critical errors if present. The software ensures that the display refreshes in real-time without flickering, using timers to update only every 100 milliseconds, maintaining a smooth and responsive user experience.

Altogether, the system creates a tightly integrated real-time loop: hand motion is instantly captured, processed, transmitted, acted upon by motors and LEDs, obstacles are detected and handled automatically, and vehicle feedback is continually returned to the user's hand, ensuring full situational awareness and control. The design achieves low-latency response, safety through emergency handling, and a professional user interface, adhering to engineering standards for gesture recognition (ISO/IEC 30113) and user interface reliability (ISO/IEC 25010).

Testing Procedures, Results, and Analysis

Hand Gesture Controller (Transmitter Side)

For the Hand Gesture Controller subsystem, testing was conducted to verify the ability of the MPU6050 sensor, Arduino Nano, and nRF24L01+ transmitter to correctly detect gestures and transmit wireless control signals to the vehicle side. Testing began with verifying that the MPU6050 sensor was correctly connected over I2C to the Arduino Uno and that the yaw, pitch, and roll data could be read accurately without error. This was validated through demonstration, using a simple Arduino sketch that printed the real-time orientation values to the Serial Monitor. Upon successful initialization of the sensor, gesture thresholds were mapped: forward tilts (positive pitch), backward tilts (negative pitch), left tilts (negative roll), right tilts (positive roll), and a flat hand (near-zero pitch and roll) for braking.

Once gesture detection was stable, the transmission path was tested. Using the nRF24L01+ module, a control packet containing x-axis, y-axis, speed, and command flags was transmitted wirelessly. Transmission success was verified by printing acknowledgment feedback on the Arduino Serial Monitor after each radio write operation.

The demonstration confirmed that the controller successfully recognized gestures and transmitted the corresponding data packets reliably. The OLED display on the controller was also tested during this phase, where real-time status updates including speed, obstacle detection flags, and system uptime were shown correctly, refreshed every 100 milliseconds to maintain smooth visual output.

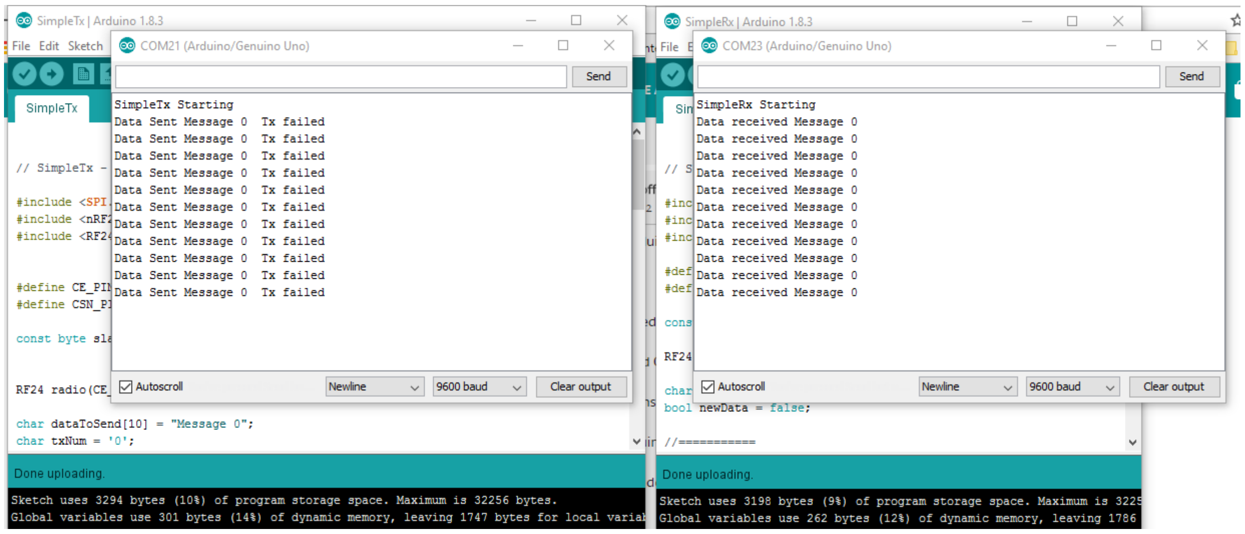
Each test aligns with the Verification Cross-Reference Matrix (VCRM) requirements by confirming sensor accuracy, gesture mapping correctness, wireless transmission reliability, and display responsiveness.

Car System (Receiver Side)

For the Car System, testing focused on ensuring that the nRF24L01+ receiver, Arduino Nano, L298N motor driver, and DC motors could properly receive transmitted commands, process them, and generate accurate movement. Initial testing began by verifying the physical connections of the receiver module to the Arduino Nano. Using a basic test sketch, it was confirmed that the nRF24L01+ receiver was initialized correctly and entered listening mode. However, during early tests, no control packets were received from the transmitter. After hardware connections were inspected and ruled out as a cause, the issue was attributed to possible Bluetooth interference on the 2.4 GHz band, as multiple wireless modules were operating nearby. To mitigate this, future steps include reassigning NRF channels and optimizing antenna separation.

Despite the temporary receiver signal issue, independent motor control testing was performed. Using manual Arduino sketches to drive the L298N motor driver, each motor was commanded forward, backward, left-turn, and right-turn. Speed control was tested by varying PWM signals sent to the enable pins, and braking was tested by setting motor inputs to LOW. Motors responded correctly to all commands, and LED indicators correctly reflected the vehicle's state: green during movement, yellow during braking, and red during full stops.

These results confirm that the motor control subsystem meets the functional requirements, and that the wireless reception path only requires minor adjustments before full integration.



```
SimpleTx Starting
Data Sent Message 0  Tx failed
Data Sent Message 0  Tx failed
Data Sent Message 0  Tx failed
Data Sent Message 0  Tx failed
Data Sent Message 0  Tx failed
Data Sent Message 0  Tx failed
Data Sent Message 0  Tx failed
Data Sent Message 0  Tx failed
Data Sent Message 0  Tx failed
Data Sent Message 0  Tx failed
Data Sent Message 0  Tx failed
```

```
char dataToSend[10] = "Message 0";
char txNum = '0';
```

```
Done uploading.
Sketch uses 3294 bytes (10%) of program storage space. Maximum is 32256 bytes.
Global variables use 301 bytes (14%) of dynamic memory, leaving 1747 bytes for local variab
```

```
SimpleRx Starting
Data received Message 0
Data received Message 0
Data received Message 0
Data received Message 0
Data received Message 0
Data received Message 0
Data received Message 0
Data received Message 0
Data received Message 0
Data received Message 0
Data received Message 0
```

```
bool newData = false;

//===========
```

```
Done uploading.
Sketch uses 3198 bytes (9%) of program storage space. Maximum is 3225
Global variables use 262 bytes (12%) of dynamic memory, leaving 1786
```

[i]Figure 13 – NRF Reciver and transmitter test in NRF

Obstacle Detection and Rerouting Subsystem

For the Obstacle Detection Subsystem, testing focused on validating the ultrasonic sensor's ability to reliably detect objects at a fixed distance threshold and trigger appropriate vehicle behavior changes.

The ultrasonic sensor, mounted on a servo motor for sweeping, was connected to the Arduino Nano, and a basic distance measurement sketch was run to verify sensor response. Objects were placed at various distances, and the system consistently detected obstacles within the set threshold of 60 centimeters, ensuring precise and early detection without false positives or negatives.

Once an object was detected, the system was tested to ensure the vehicle would immediately stop, update LED status to yellow, and attempt a reroute by performing a turning maneuver. If obstacles persisted after two rerouting attempts, the vehicle triggered a full emergency stop, switched to a blinking red LED, and halted all motion for safety. This full rerouting logic was validated by demonstration with controlled obstacles placed ahead of the vehicle.

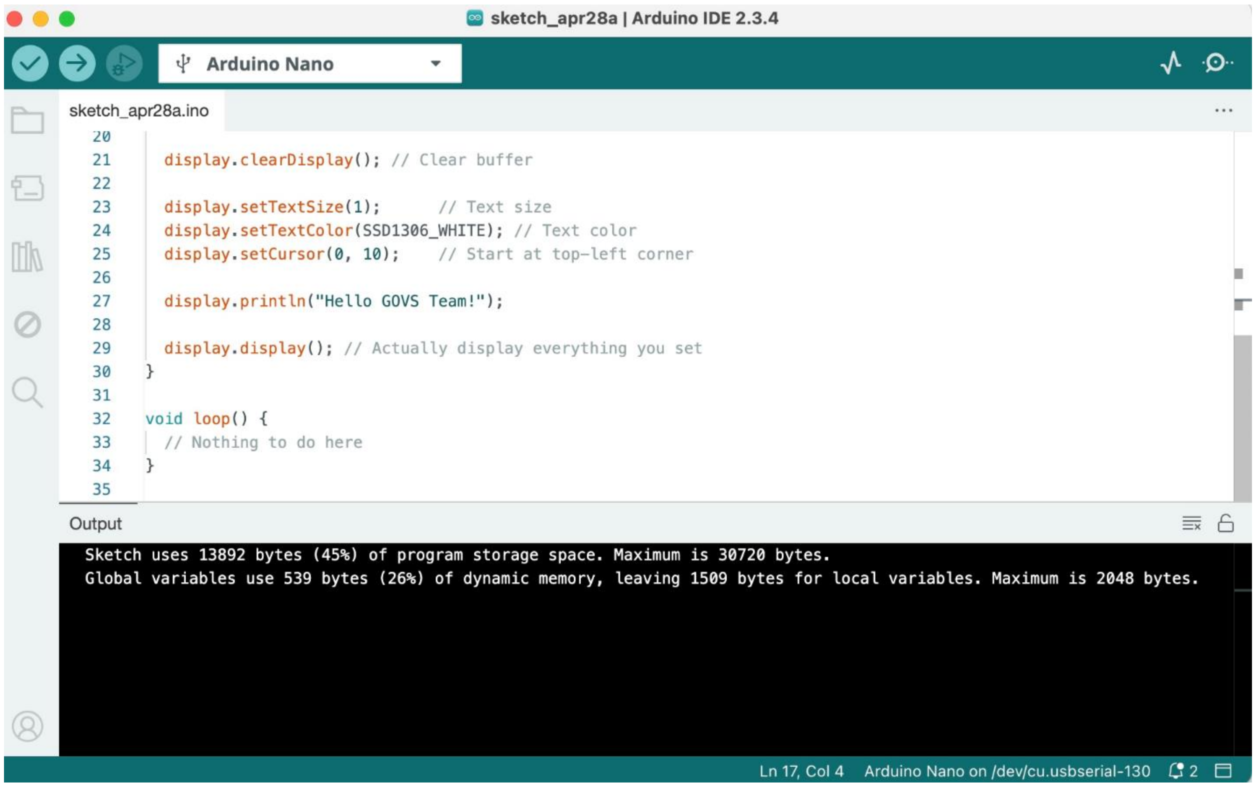
Each test step confirmed that the Obstacle Detection Subsystem successfully prevents collisions and complies with the project’s environmental awareness goals.

System Feedback Display Subsystem

For the System Feedback Display Subsystem, testing verified that real-time data about the vehicle's speed, obstacle status, battery health, and uptime could be collected, transmitted back to the hand controller, and displayed accurately on the OLED screen.

Testing began by simulating motor speeds, battery levels, and obstacle detections on the vehicle Arduino, and verifying that feedback packets were properly sent over the NRF module. On the hand controller side, the Arduino Nano received these packets and updated the OLED display accordingly.

Display refresh intervals were optimized to 100 milliseconds to prevent screen flickering while maintaining live updates. Testing showed that when an obstacle was detected, "OB" appeared immediately on the OLED. When battery levels dropped below 10%, a "LOW!" warning was displayed. System uptime was correctly tracked and formatted as hours, minutes, and seconds.


sketch_apr28a | Arduino IDE 2.3.4
Arduino Nano
sketch_apr28a.ino
Output
Ln 17, Col 4 Arduino Nano on /dev/cu.usbserial-130 2

```
  display.clearDisplay(); // Clear buffer

  display.setTextSize(1);      // Text size
  display.setTextColor(SSD1306_WHITE); // Text color
  display.setCursor(0, 10);    // Start at top-left corner

  display.println("Hello GOVS Team!");

  display.display(); // Actually display everything you set
}

void loop() {
  // Nothing to do here
}
```

```
Sketch uses 13892 bytes (45%) of program storage space. Maximum is 30720 bytes.
Global variables use 539 bytes (26%) of dynamic memory, leaving 1509 bytes for local variables. Maximum is 2048 bytes.
```

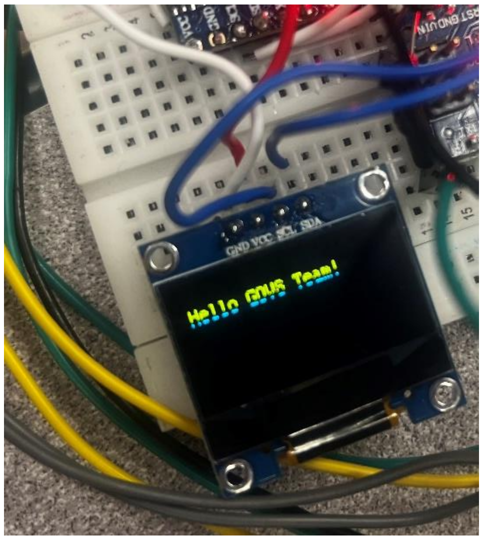


Figure 14: TESTING FOR DISPLAY: “Hello Govs Team”

These results confirm that the System Feedback Display Subsystem meets functional requirements for providing real-time, user-friendly information to the operator.

Design Challenges

Challenge 1: Power Instability Affecting NRF24L01 Communication

Risk: If The NRF24L01 wireless module shares unstable or overload power with other peripherals (40%) then signal transmission may become unreliable, causing command loss and system delay.
Mitigation Plan: Isolate NRF24L01 by providing it with a separate regulated 3.3V power source and add 100µF capacitor close to the module to stabilize voltage under bursts.

Challenge 2: Signal and Slowness Due to NRF24L01 Full-Duplex Conflict

Risk: If the NRF24L01 module simultaneously tries to handle both transmitting and receiving without proper handshake protocols (40%) THEN it may cause data collisions, slow gesture response, or communication failure.
Mitigation Plan: Implement a stopListening()/startListening() mechanism in the code, use ACK payloads to synchronize transmissions, and balance the communication timing carefully to avoid wireless congestion.

Challenge 3: MPU6050 Detects Unintended Vibrations

Risk: If the MPU6050 sensor picks up unintended hand vibrations or ambient motion (30% likelihood), **then** the car may execute false movements, resulting in unpredictable behavior and potential safety issues, delaying final calibration and testing by 2–3 days.

Mitigation: Add digital low pass filtering in software, apply gesture thresholds to distinguish between intentional and unintentional movements, and validate gestures over multiple consistent readings before acting.

Challenge 4: SPI Conflict Between NRF24L01 and OLED LCD Display

Risk: If the OLED LCD display and NRF24L01 Both access the shared SPI bus at the same time without proper chip select (CS) Handling (25%) then communicating errors could corrupt gesture data or display updates.
Mitigation Plan: Separate chip-select (CS)control pins for each device manage the bus carefully by asserting only one device's CS low at a time, and coordinate communication timing by switching SPI focus between modules during operation.

Challenge 5: Motor Driver (L298N) Unexpected Stops Due to Servo and Ultrasonic Operation

Risk: IF the servo motor and HC-SR04 ultrasonic sensors draw power from the same supply line as the L298N motor driver (30%) THEN the voltage may drop during servo movement or sensor pulses, causing the motors to reset or stop.
Mitigation Plan: Power the servo and ultrasonic sensor separately using a 5V regulated output (via LM1117T-ADJ circuit) and keep the motor driver on its own battery, ensuring independent, stable current flow for each critical subsystem.

Challenge 6: LCD Display Failure to update Due to Communication Overload

Risk: If the OLED display update frequency is too high, combined with NRF24L01 communication tasks (20%) THEN the processor may miss gesture inputs or fail to transmit movement commands.
Mitigation Plan: Limit display refresh rate to no faster than every 100 ms, buffer status data updates, and prioritize gesture communication over visual updates to preserve critical control timing.

Challenge 7: Emergency Stop System Trigger by Noise on Ultrasonic Reading

Risk: IF ultrasonic sensor readings generate false positives from noise, reflections, or missed pings (20%) THEN the car may incorrectly perform an emergency stop even without a real obstacle.
Mitigation Plan: Require multiple consecutive detections before triggering obstacle avoidance or emergency routines and apply basic averaging to smooth out spurious sensor data.

Challenge 8: Unreliable Obstacle Detection

Risk: IF the car fails to detect nearby obstacles consistently (30%) THEN it may collide with objects, risking physical damage to the chassis or motor components.
Root Cause (Software): The detectObstacle() function lacked a timeout and did not validate distance values, allowing occasional false readings from the HC-SR04 sensor.

Mitigation Plan: A timeout parameter (pulseIn(echoPin, HIGH, 30000)) was added, and software logic was improved to discard zero-distance readings. This improved detection reliability and prevented hardware failures during autonomous rerouting.

Challenge 9: NRF24L01 Delayed Signal Reception

Risk: IF the NRF24L01 transceiver sends and receives data too slowly (25%) THEN the car may miss gesture input, causing delayed or inaccurate motor responses.
Root Cause (Software): The same radio.write() function was redundantly called twice per loop, and startListening() was not reinitialized optimally, affecting timing.
Mitigation Plan: Optimized the control loop to send gesture data only once per cycle and correctly handle acknowledgment payloads. This minimized transmission delay and restored responsive movement on the car hardware.

Challenge 10: OLED Display Slows Gesture Processing

Risk: IF OLED display updates are called too frequently (20%) THEN gesture input from the MPU6050 can be delayed, leading to jerky or missed motion commands on the vehicle.
Root Cause (Software): The updateDisplay() function was executed every 100ms, even when no gesture or system status had changed.
Mitigation Plan: The display update was limited using a timed condition (millis() - lastDisplayUpdate > 100) and could be further improved with a flag that checks for actual data change. This improved MPU response time, reducing gesture lag and avoiding sudden braking or misdirection.

Conclusions and Future Work

During the development of the Gesture Operated Vehicle System (GOVS), our team gained valuable experience working with hardware and software systems. We successfully created a working prototype that can be controlled using hand gestures, and we learned how to combine different technologies like sensors, wireless communication, motor drivers, and microcontrollers.

One of the biggest lessons we learned was the importance of early testing and calibration, especially for gesture recognition. Small issues in sensor data or communication timing can have a big effect on how the system behaves. We also realized how important it is to keep clear documentation and have good teamwork when working on a complex project like this.

For future work, we plan to improve the gesture accuracy by using machine learning to better recognize different hand motions. We would also like to add a more advanced obstacle detection system and increase the system's range and speed. Lastly, we hope to make the system more user-friendly by adding a mobile app and improving voice control.

This project showed us how real-world problems can be solved through teamwork, engineering skills, and creative thinking.

ABET Outcomes

Individual Reflections Report Content (to be completed by each senior design student in light of ABET student outcomes 2 and 7)

ABET Student Outcomes: https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-engineering-programs-2025-2026/

REFLECTIONS

Name: Amit Vardhan Suryadevara

Major: Electrical Engineering

Course: Senior design (ECE 49200)

Design Team: GOVS

(a) Describe your personal contributions to the project.

My primary contributions to the project were focused on the hardware implementation. I was responsible for assembling the chassis of the car, integrating the necessary electronic components, and ensuring all connections were correctly made. Specifically, I mounted and connected the Arduino Uno as the main microcontroller for the car, installed and wired the MPU-6050 for motion sensing, and connected and configured the nRF24L01+ wireless adapter for communication with the hand gesture controller. I also wired the L298N motor driver and ensured proper voltage supply for the DC motors. Additionally, I debugged and verified the nRF24L01+ signal reception on the car side to ensure reliable wireless communication.

(b) Describe how your contributions to this project built on the knowledge and skills you acquired in earlier course work.

This project built upon the knowledge I gained from my coursework in embedded systems, control systems, and circuit design. My previous experience with Arduino-based projects helped me understand how to interface sensors and modules, which was essential for integrating the different hardware components. My coursework in electronics and circuit design provided a foundation for selecting appropriate voltage levels and ensuring that each component received the correct power supply. Knowledge gained from control systems courses helped me configure the L298N motor driver and manage motor speed and direction effectively. Additionally, my familiarity with communication protocols allowed me to troubleshoot and resolve issues related to the nRF24L01+ wireless module.

(c) Describe how you acquired and applied new knowledge as needed to contribute to this project. What learning strategies (see definition on first page) did you employ to do so?

To successfully complete my role in the project, I had to learn and apply new knowledge using several strategies. I engaged in self-learning and research by referring to datasheets for the MPU-6050, nRF24L01+, and L298N motor driver to understand their functionalities and wiring requirements. I also used trial-and-error testing, particularly when debugging the nRF24L01+ connectivity issues, by trying different wiring setups and power configurations. Additionally, I explored online resources such as Arduino forums, GitHub repositories, and YouTube tutorials to troubleshoot hardware problems. Team collaboration also played an important role, as I regularly discussed challenges with my teammates, especially while working through the wireless communication issues on the car side.

(d) Discuss your ethical and professional responsibilities as they relate to this engineering design experience.

As an engineer, I upheld several ethical and professional responsibilities throughout this project. I prioritized accuracy and safety by ensuring that all components received the correct voltage supply, preventing overheating or damage. I practiced responsible material use by avoiding unnecessary component waste and selecting cost-effective yet high-quality parts. When debugging issues, such as the nRF24L01+ reception problem, I maintained honesty by transparently reporting all encountered problems to the team. Additionally, I worked collaboratively and with integrity, ensuring clear and professional communication regarding hardware challenges and proposed solutions. These practices helped ensure that the project was completed responsibly, efficiently, and with a strong commitment to professional standards.

(e) Consider what impact the product of this engineering design experience could have in economic, environmental, societal, and global contexts. Discuss how you would make (or did make) an informed judgment (see definition on first page) as to your product's impact in each of these four contexts

The product of this engineering design experience has meaningful impacts in economic, environmental, societal, and global contexts. From an economic perspective, the design emphasizes cost-efficiency by utilizing readily available, low-power components, with future improvements aimed at exploring alternative motor drivers to further reduce costs without compromising performance. Environmentally, the use of rechargeable batteries helps minimize electronic waste, and future iterations could integrate solar charging capabilities to enhance long-term sustainability. Societally, the project advances assistive technologies, as gesture-controlled vehicles offer more intuitive control options for individuals with disabilities. Globally, the wireless control framework developed through this project could be expanded to applications such as remote robotics, industrial automation, and military reconnaissance. In making these informed judgments, I carefully evaluated factors including component sustainability, user accessibility, cost-effectiveness, and the potential for broader technological adoption across diverse industries and regions.

Name: Divya Patel
Major: Computer Engineering
Course: Senior Design (ECE 49200)
Design Team: GOVS (HandiCar)

(a) Personal Contributions to the Project:

My personal contributions to the project focused on system-level planning, design, and coordination. I ensured system-level clarity by developing detailed planning documents and organizing the overall workflow. I designed the system architecture, including creating a block diagram to outline component interactions, and compared multiple hardware and software options to guide efficient design decisions. I assisted in hardware selection to ensure compatibility with project requirements and supported the assembly and validation of circuits. Throughout the project, I maintained task flow and deadlines to ensure steady progress and bridged the communication gap between the software and hardware teams. Additionally, I learned new software tools and libraries to support system development and contributed to optimizing system performance by selecting the most effective implementation approaches.

(b) Application of Prior Knowledge and Skills:

My contributions to this project built directly upon the knowledge and skills I developed through earlier coursework. I applied concepts from Embedded Systems to interface microcontrollers with hardware components effectively. Signals and Systems coursework helped me understand sensor outputs, particularly interpreting pitch and roll data from motion sensors. My experience with C/C++ programming supported firmware-level development and performance tuning for the system. Additionally, lab-based courses provided practical experience in planning, managing tasks within a team, and meeting project deadlines. I also leveraged circuit and system design concepts gained from previous classes to guide hardware selection and system optimization.

(c) Acquisition and Application of New Knowledge:

To contribute effectively to the project, I actively acquired and applied new knowledge using several learning strategies. I conducted self-directed research on hardware components such as the MPU6050, nRF24L01, and motor drivers to better understand their functions and integration requirements. I used visual learning techniques by creating and analyzing block diagrams and circuit schematics to clarify system interactions. Comparative analysis was applied to evaluate design options, such as choosing between C and Python for different parts of the system. I also engaged in collaborative learning by discussing findings and troubleshooting strategies with teammates. Studying reference code and datasheets helped me understand communication protocols and sensor integration, and I regularly shared useful resources with the team while seeking help when needed to promote collective understanding.

(d) Ethical and Professional Responsibilities:

Throughout this engineering design experience, I consistently upheld ethical and professional responsibilities essential to the integrity and success of the project. I maintained honesty and transparency by accurately reporting system limitations, software bugs, and hardware challenges as they arose. I promoted responsible and cost-effective hardware usage by minimizing waste and making informed procurement decisions based on technical and budgetary constraints. Ensuring the safety of both users and components remained a priority, particularly in the handling of power supplies and motor drivers. I fostered a respectful and inclusive team environment by encouraging open communication and valuing the contributions of all team members. Furthermore, I made all design decisions thoughtfully and in alignment with established professional engineering standards and best practices.

(e) Impact on Economic, Environmental, Societal, and Global Contexts:

The product of this engineering design experience has meaningful impacts across economic, environmental, societal, and global contexts. Economically, the system utilized low-cost, widely available components, ensuring that the design remains affordable and easily replicable for future development and broader adoption. Environmentally, the modular design approach promoted reusability and repairability, helping to minimize electronic waste and extend the life cycle of system components. From a societal perspective, the project developed an alternative control method that can significantly benefit individuals with physical disabilities by providing a more accessible and intuitive interface. Globally, the technology offers broad applications in fields such as robotics, education, and automation, with the potential to enhance technological accessibility and innovation across diverse regions.

Name: Yamini Sri Krubha
Major: Computer Engineering
Course: Senior Design (ECE 49200)
Design Team: GOVS

(a) Personal Contributions to the Project:

My personal contributions to the project were primarily focused on software development for the hand gesture recognition system that controls the car. I developed the software responsible for interpreting gesture inputs and translating them into car movement commands. Throughout the project, I conducted multiple rounds of testing to ensure accurate gesture detection and smooth car operation, and I continuously debugged and optimized the code for better performance and reliability. I collaborated with team members to integrate the software with the hardware system and participated in key software design decisions, including selecting the programming language and completing the full codebase. Additionally, I contributed to drafting the project proposal, ensuring that all software-related requirements complied with project policies and technical specifications. I also documented the software development process, recording testing results, identified issues, and improvements for future reference.

(b) Application of Prior Knowledge and Skills:

My contributions to this project built directly on the knowledge and skills I acquired through earlier coursework. I applied coding experience gained from Advanced C/C++ Programming to write efficient and structured code for the gesture recognition system. Concepts from Signals and Systems for Sensors helped me process and interpret gesture inputs accurately. I utilized my background from Introduction to Electronics to interface the Arduino with sensors and microcontrollers. Additionally, I implemented techniques learned in Embedded Systems and Machine Learning courses to develop the gesture recognition algorithms. Knowledge from Microcontroller Programming also played a key role in integrating the car’s hardware with the software, ensuring seamless communication between components.

(c) Acquisition and Application of New Knowledge:

To contribute effectively to this project, I acquired and applied new knowledge using several learning strategies. I researched various gesture recognition algorithms to enhance the accuracy of hand gesture detection and explored different sensor calibration techniques to improve system responsiveness. I experimented with multiple signal processing methods to filter noise and optimize sensor data. Through hands-on testing and debugging, I refined the software’s performance with iterative improvements. I also collaborated with teammates to better understand hardware constraints and adjusted the software design accordingly. To deepen my understanding of sensor integration and embedded systems, I referred to technical documentation and research papers. Additionally, I utilized online resources and forums like GitHub to troubleshoot issues and implement best practices in coding and system design.

(d) Ethical and Professional Responsibilities:

During this engineering design experience, I consistently upheld ethical and professional responsibilities essential to the success and integrity of the project. I prioritized safety and reliability by rigorously testing the software to prevent malfunctions that could result in unintended car movements. Transparency and accuracy were maintained in both coding and documentation to facilitate future enhancements and effective troubleshooting. I adhered to ethical coding practices by avoiding plagiarism and properly citing all external resources. Furthermore, I emphasized usability and accessibility to ensure the system could be operated effectively by a diverse range of users, including individuals with disabilities. Professional integrity was demonstrated through consistent adherence to project deadlines, effective collaboration with team members, and transparent communication regarding technical challenges. Finally, I considered data privacy and security measures when handling sensor inputs, ensuring that user data remained protected from unauthorized access or misuse.

(e) Impact on Economic, Environmental, Societal, and Global Contexts:

The product of this engineering design experience offers impactful contributions across economic, environmental, societal, and global contexts. Economically, it enables cost-effective automation in assistive technologies and robotics, reduces manufacturing costs through the use of widely available components, and presents commercialization opportunities in sectors such as smart vehicles and human-machine interfaces. Environmentally, the system helps minimize e-waste by reducing reliance on physical controllers, promotes energy-efficient embedded system designs, and encourages the adoption of sustainable materials. Societally, it enhances accessibility for individuals with disabilities, improves user safety through hands-free operation, and fosters innovation by encouraging interest in STEM fields. Globally, the technology is adaptable across healthcare, automation, and defense industries, advances AI-driven human-computer interaction, and addresses mobility challenges in developing regions. These informed judgments were made by evaluating sustainability, economic feasibility, societal benefit, and the scalability of the technology in diverse global markets.

Name: Martine Cardichon

Major: Electrical Engineering

Course: Senior design (ECE 49200)

Design Team: GOVS (the HandiCar)

(a) Personal Contributions to the Project:

My contributions to the project pertain to the software. Specifically, I wrote the software for the sensors and brake/accelerate lights. I also helped troubleshoot the possible issues with the nrf24l01 chip when the receiver was not receiving messages from the remote. Initially, the software team (Yamini and I) were going to program the software in Python, but late switched to C++ due to its compatibility with the Arduino. Regarding the car, python has a myriad of libraries (especially related to robotics) that would especially for the nrf24l01 chip (the Bluetooth chip that communicates between the remote and the car) that make the function of car possible. Specifically, I wrote the base code for the object detection sensor and the car lights (the brake and forward light). When we switched the project to C++, I researched the language and libraries utilized in C++ regarding the specific functions in the code. I also 3D modeled the chassis cover and remote cover. I took thorough measurements of both the car and remote, then I made several protoypes to ensure the best design. *We ended up not using the chassis cover due to potential wiring unitentionally touching each other and short circuiting.

(b) Application of Prior Knowledge and Skills:

As part of the software team, we had to have extensive knowledge in several programming languages (such as C, C++, and Python). Yamini and I were the most comfortable in these skills. Regarding the chips that we purchased, they are Arduino compatible meaning they can be coded in C++ and given that that is the first coding language I learned and the one I code the most in (I also have my own Arduino kit). Though I didn't work explicitly on the hardware team, I helped with soldering the wires to the chip (we were practicing soldering as a viable solution for wire connections). I mainly assisted in troubleshooting hardware and software I've also had to rely on my soft skills such as interpersonal communication in order to help our vision of the HandiCar come to fruition.

(c) Acquisition and Application of New Knowledge:

For the project, I continued to develop my python skills. Most of my coding knowledge lied within the scope of C and C++ so I had to further develop my Python skills.When we switched to C++ (that is the language compatible with Arduino), I got to further develop my C++ with some more advanced coding. I also did extensive research on different libraries available, specifically in robotics, in order to implement it in the software. I also researched the hardware in order to find out what the hardware could do. I utilized my CAD coursework experience to create the 3D modeling needed for the project.

(d) Ethical and Professional Responsibilities:

As an engineer, in any design experience safety is of the utmost importance. Within the scope of the project I helped ensured the safety my making sure the battery connection did not short circuit or overheat. As a dual degree major, my schedule was incompatible most of colleauges so I needed to maintain a level of transparency and professional integrity through constant communication. I constant called, text, or emailed them if I was running late and to communicate all internal deadlines. All sources I researched and used in the project were properely sourced and cited, avoiding any instances of plagiarism.

(e) Impact on Economic, Environmental, Societal, and Global Contexts:

The HandiCar can serve as a prototype for the semi-autonomous vehicle economy. If scaled up to the size need for human transportation, the production of the HandiCar will boost the semi-autonomous vehicle sector. The cost of production, with the commercialization of the HandiCar, can be cost effective, especially with the extensive use of microcontrollers. Societally, the HandiCar can increase the accessibility of cars, allownig those with mobility impairment to continue drive. This will give people with physical disabilities more freedom. The HandiCar will make for more effcient driving therefore reducing some negative environmental impacts. The there will still be some negative impacts regarding raw material depletion due to the dependency on microcontrollers. Globally, it will have net positive economic and societal impacts, further expanding upon the semi-autonomus/electric vehicle industry, further stimulating the global economy. The HandiCar will also make transportation more accesible, especially for communities lacking public transportation.

Name: Maha Ali
Major: Electrical Engineering
Course: Senior Design (ECE 49200)
Design Team: GOVS (HandiCar)

(a) Personal Contributions to the Project:

Throughout the development of the Gesture Operated Vehicle System (GOVS), I contributed extensively to the hardware subsystem design and implementation. I was responsible for selecting and integrating core components, including the MPU6050 motion sensor, NRF24L01 wireless transceiver, L298N motor driver, ultrasonic sensor, and servo motor. My work involved circuit design, wiring, power distribution, and resolving component conflicts to ensure system stability. I also contributed to chassis assembly and supported collaborative debugging during testing phases to align hardware functionality with the gesture-based control software.

(b) Application of Prior Knowledge and Skills:

This project allowed me to apply foundational knowledge gained in prior coursework such as Embedded Systems, Digital Logic Design, and Microcontrollers. Specifically, I utilized skills in circuit prototyping, sensor interfacing, and serial communication protocols (e.g., I2C and SPI). My experience with Arduino programming and hardware troubleshooting directly supported my ability to integrate the various modules into a cohesive system. These earlier experiences gave me the confidence and technical base to approach this complex design problem systematically.

(c) Acquisition and Application of New Knowledge:

To complete this project successfully, I needed to learn how to interface the NRF24L01 wireless module and address power management issues when multiple components shared limited voltage sources. I studied technical datasheets, explored online forums, and consulted open-source community examples to understand each component's electrical behavior and constraints. I used active learning strategies—such as experimentation, troubleshooting through trial and error, and engaging in peer discussions—to deepen my understanding and apply new concepts in real time. This hands-on approach allowed me to adapt quickly and resolve issues efficiently as new challenges emerged..

(d) Ethical and Professional Responsibilities:

As an engineering student engaged in a real-world design project, I took my ethical and professional responsibilities seriously. I ensured that all components were safely connected and operated within their electrical limits to prevent damage or unsafe behavior. I practiced academic integrity by acknowledging online resources used during the research phase, and I promoted inclusive teamwork by encouraging all members to contribute ideas and participate in testing. Additionally, I prioritized clear communication with teammates and maintained professionalism in both documentation and group presentations.

(e) Impact on Economic, Environmental, Societal, and Global Contexts:

The GOVS project has potential applications in assistive mobility and human-machine interaction, particularly for users with physical disabilities. From an **economic** standpoint, our use of affordable components like the Arduino Nano and NRF24L01 makes the system scalable and cost-effective for broader accessibility. In terms of **environmental impact**, the project emphasizes energy efficiency by carefully selecting low-power modules and separating power sources to prevent waste. On a **societal level**, the project introduces a gesture-controlled platform that can reduce barriers to mobility, particularly for individuals with limited dexterity. From a **global perspective**, gesture-operated systems can support inclusive technology adoption in underserved communities or areas with infrastructure limitations. Informed by these contexts, we made conscious decisions to prioritize affordability, safety, accessibility, and modular design throughout the development process.

Name: Biak Par
Major: Electrical Engineering
Course: Senior Design (ECE 49200)
Design Team: GOVS (HandiCar)

(a) Personal Contributions to the Project:
As one of the projects managers of the GOVS design team, I took the lead in organizing all project-related reports and documentation, ensuring that our materials were well structured and submitted on time. I was responsible for submitting assignments on behalf of the team, maintaining clear communication with faculty and adhering to all submission guidelines. Additionally, I actively participated in testing the car, providing support during critical evaluation phases to validate system functionality and performance. I also contributed to the development of system flowcharts, collaborating with team members to visually represent the project's architecture and processes. These efforts helped maintain clear communication across the team and ensured that both administrative and technical aspects of the project progressed smoothly.
(b) Application of Prior Knowledge and Skills:
This senior design project allowed me to integrate and apply knowledge from my previous coursework. ECE 27000 (Digital System Design) provided a foundation in digital logic and hardware description languages, which helped in designing the digital components of our system. ECE 36200 (Microprocessor Systems and Interfacing) deepened my understanding of microcontrollers, assembly programming, and interfacing; although I initially had limited knowledge about devices like the Arduino Nano or Uno, this project helped me gain a much deeper understanding. Technical Communication (TCM) courses improved my skills in documenting technical processes and collaborating with the team effectively. Additionally, experience from Creo-based design courses strengthened my understanding of mechanical integration through CAD modeling. Together, these courses provided the essential technical and communication skills needed for the success of our senior design project.
(c) Acquisition and Application of New Knowledge:

Throughout this project, I gained new technical and communication skills by engaging in both individual learning and team collaboration. I learned how to write professional technical reports by studying example documents, applying feedback from instructors, and incorporating technical details with clarity and precision. I also familiarized myself with the structure of system flowcharts and learned to create diagrams that clearly convey the flow of data and control in the system. During car testing and troubleshooting sessions, I expanded my knowledge of embedded systems by observing the interaction between hardware and software components. I also improved my organizational and project management skills by managing deadlines, aligning team tasks, and ensuring that all required documentation met academic and professional standards.

(d) Ethical and Professional Responsibilities:

As a project manager and contributor, I upheld ethical and professional responsibilities by ensuring all submissions were accurate, timely, and reflected the true work of the team. I practiced academic honesty by properly attributing contributions and ideas and made sure that our documentation met the required standards for clarity and completeness. I communicated professionally with teammates and faculty, maintained transparency in reporting issues, and emphasized accountability in both technical and administrative tasks. I also made it a priority to represent the work of the team fairly, promote inclusivity during collaboration, and support an environment of mutual respect and shared responsibility.

(e) Impact on Economic, Environmental, Societal, and Global Contexts:

The Gesture Operated Vehicle System (HandiCar) has meaningful potential across multiple contexts. Economically, it demonstrates that assistive technology can be developed using cost-effective components such as Arduino boards and wireless modules, making it accessible for individuals and organizations with limited budgets. Environmentally, the system was designed with modular components that can be reused or upgraded, reducing electronic waste and extending the device's lifespan. From a societal perspective, the project supports accessibility and inclusion by offering a hands-free control method tailored to individuals with mobility challenges. Globally, the design can be adapted to enhance mobility in regions lacking infrastructure or traditional vehicle accessibility, offering a versatile platform for educational, assistive, or exploratory applications. These considerations informed our decisions throughout the design process, guiding our focus on affordability, usability, and responsible innovation.

REFERENCES

APPENDIX A. SPECIFICATION OF ALL HARDWARE PARTS

Fig 2: L298N Motor Driver Board Module Stepper Motor DC Dual H- Bridge for Arduino

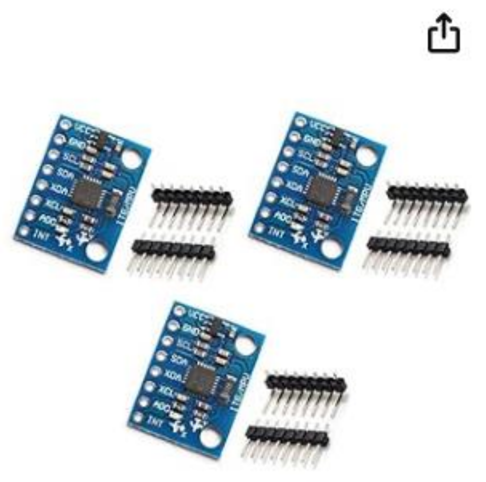


Fig 4: HiLetgo 3pcs GY-521 MPU-6050 MPU6050 3 Axis Accelerometer Gyroscope Module 6 DOF 6-axis Accelerometer Gyroscope Sensor Module 16 Bit AD Converter Data Output IIC I2C for Arduino

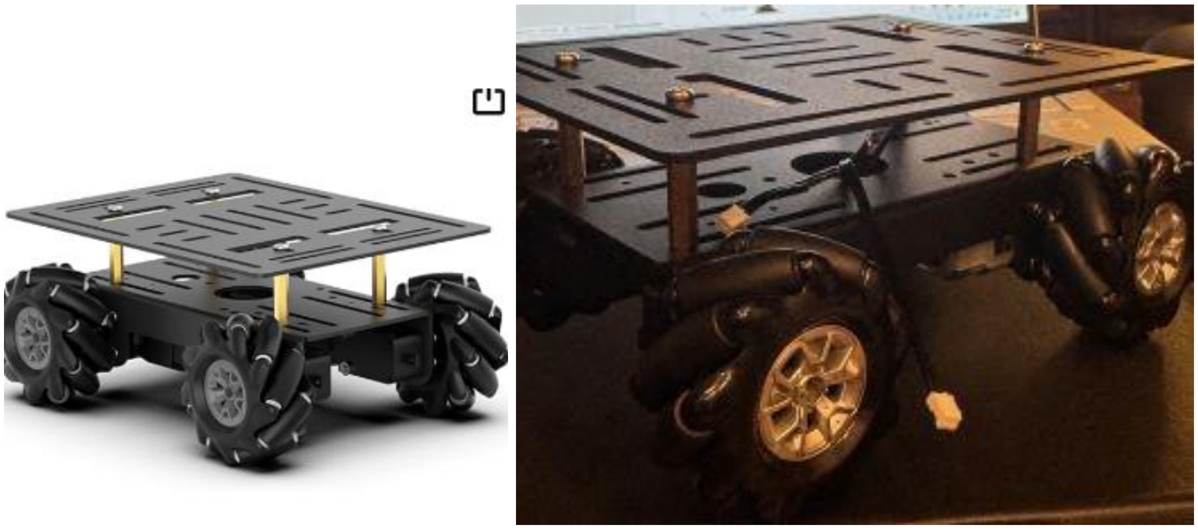


Fig 6: Omnidirectional Chassis

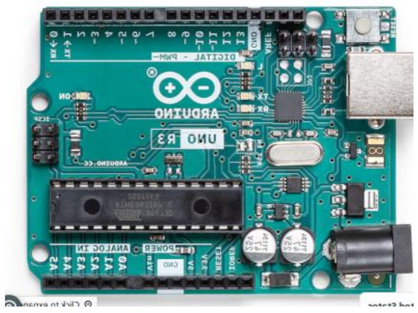


Fig 8: Arduino Nano(Car) Figure 9: Arduino Uno(Hand Gesture)

APPENDIX B. SPECIFICATION OF ALL SOFTWARE AND TOOLS USED

Hardware Components

- **Arduino Board** (Uno/Nano or compatible)
- **MPU6050** (6-axis IMU with accelerometer + gyroscope)
- **NRF24L01** (2.4GHz RF transceiver module)
- **Motor Driver** (L298N or similar)
- **DC Motors** (x2 for wheel movement)

Software & Libraries

- **Arduino IDE** (Primary development environment)
- **I2Cdev Library** (I²C communication for MPU6050)

- **MPU6050_6Axis_MotionApps20** (DMP sensor fusion)
- **RF24 Library** (NRF24L01 wireless communication)
- **SPI Library** (Built-in, for NRF24L01 communication)

Key Protocols

- **I2C** (For MPU6050 communication)
- **SPI** (For NRF24L01 communication)
- **DMP** (Digital Motion Processing for sensor fusion)

Debugging Tools

- **Serial Monitor** (For debug output when PRINT_DEBUG is enabled)
- **Logic Analyzer** (for signal verification)

Additional Tools

- **Wire Library** (Built-in Arduino I²C communication)
- **PWM** (For motor speed control)

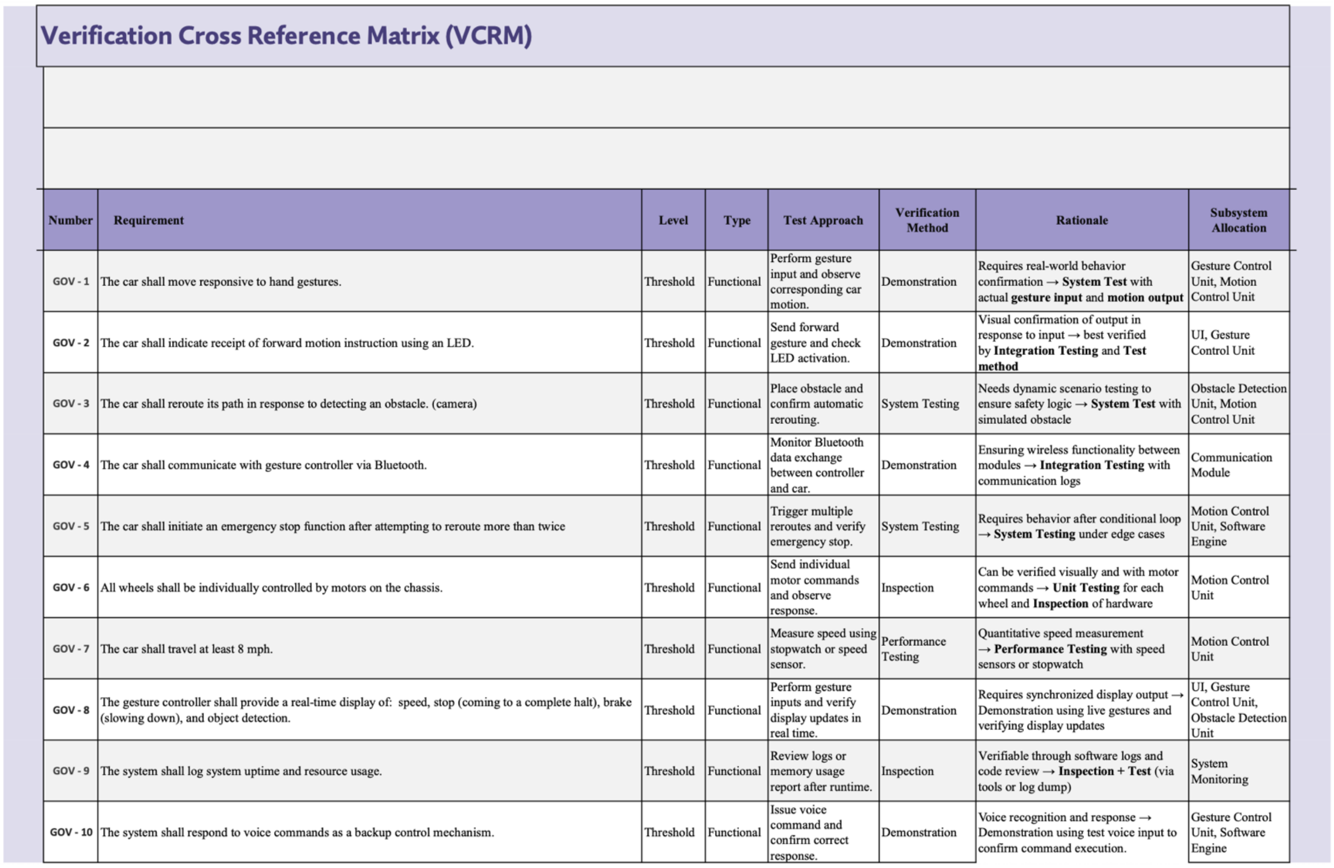
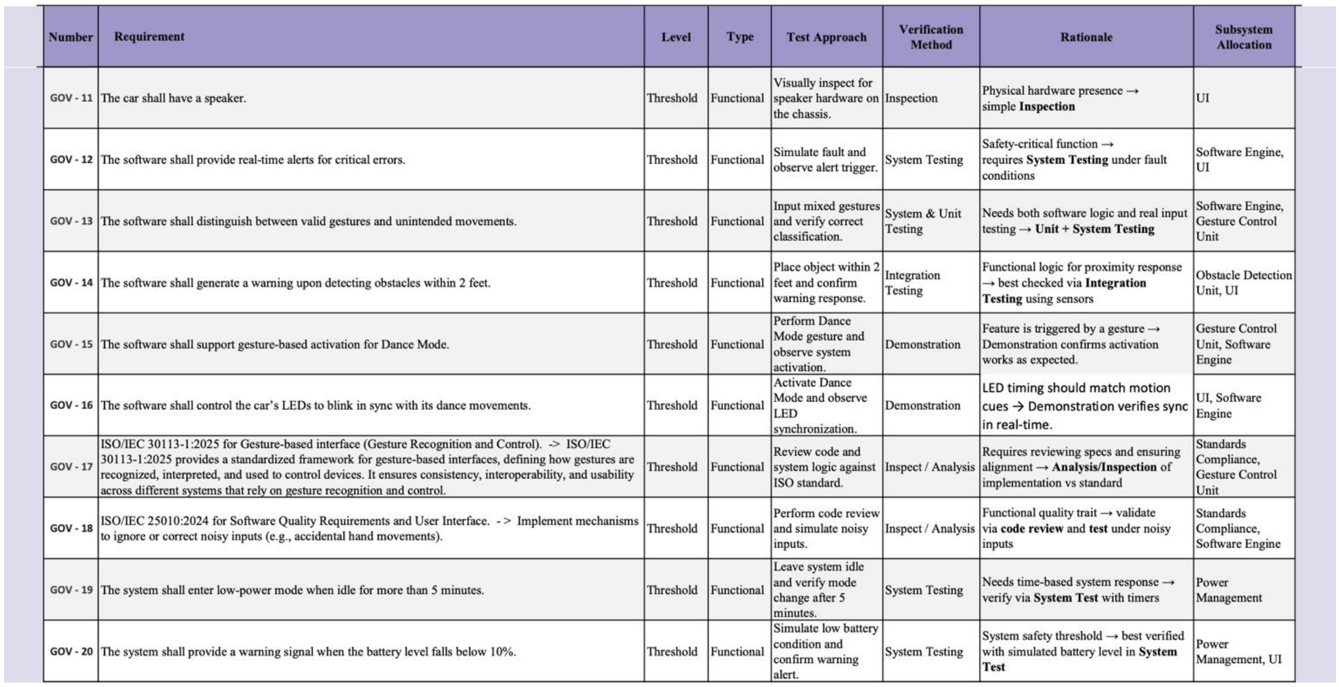
APPENDIX D. BILL OF MATERIALS

Bill of Materials Table

ITEM DESCRIPTION	QUANTITY	UNIT COST
Nano V3.0, Nano Board ATmega328P 5V 16M Micro-Controller Board Compatible with Arduino IDE (Nano x 3 with USB Cable)	1	$**15.99**
MakerHawk 3.7V 1100mAh LiPo Battery 102540 Rechargeable 1S 3C Lithium Polymer Battery with Protection Board Insulated Rubber Tape, Micro JST 1.25 Plug for Arduino ESP32 Development Board (4 Pack)	1	$**16.49**
Voltage Step Up Converter, 15PCS 2V-24V to 5V-28V 2A DC DC Boost MT3608 DC-DC Adjustable Boost Power 100W High Power Output 3.5-35V Power Regulator Board with LED Digital Voltage Meter	1	$**16.99**
HiLetgo 3pcs GY-521 MPU-6050 MPU6050 3 Axis Accelerometer Gyroscope Module 6 DOF 6-axis Accelerometer Gyroscope Sensor Module 16 Bit AD Converter Data Output IIC I2C for Arduino	1	$**10.99**
COCOBOO 4 Inch x 9 Feet Hook and Loop Tape for Couch Cushions, Strips with Adhesive, Nylon Self-Adhesive Heavy-Duty Strips for Home Office School Car and Crafting Organization	1	$**9.99**
LAFVIN 4WD Multi Robot Car Kit Upgraded V2.0 Robot STEM/Graphical Programming Robot Car Compatible with Arduino IDE with Tutorial	1	$36.**99**

HiLetgo 2PCS APDS-9960 APDS9960 RGB Gesture Sensor Module Hand Gesture Recognition Moving Direction Ambient Light RGB Proximity Sensor Module Infrared Move Sensor	2	$**17.98**
Mecanum Robot Chassis Car Kit with 360° Omnidirectional Wheel, Aluminum Alloy Frame, TT Motor, 66mm Omnidirectional Wheels, Robotic Moving Platform Robot Car DIY Kit (Unassembled)	1	$41**.99**
1"x16ft Hook and Loop Straps with 25 Metal Buckles, Adjustable Fastening Cable Straps, Cut-to-Length Cinch Strap, Reusable Nylon Securing Cord Ties for Organizer or Storage, 1 inch Wide, Black	1	$9.99
Breadboard-Friendly PCB Mount Mini Speaker - 8 Ohm 0.2W	2	$3.70
HiLetgo PAJ7620U2 3.3V Gesture Recognition Sensor Module Gesture Detection Gesture Monitor Motion Sensor Recognize Multiple Gestures for Arduino Raspberry Pi	2	$20.**98**
4pcs 1□8□6□5□0 Rechargeable Batter□y W□i□th 18650 Battery Charger,Universal Smart Charger for 3□.7V L□ithium ion Batteries LSXdetoro (4pcs Button Top Batter□□y+Charger)	1	$19.98
HiLetgo HC-05 Wireless Bluetooth RF Transceiver Master Slave Integrated Bluetooth Module 6 Pin Wireless Serial Port Communication BT Module for Arduino	1	$10.39
MakerFocus 3pcs NRF24L01+PA+LNA Wireless Transceiver RF Transceiver Module 2.4G 1100m with Antenna and 3pcs NRF24L01+ Breakout Adapter with 3.3V Regulator on-Board for Ar Duino	1	$14
HiLetgo 4pcs L298N Motor Driver Controller Board Module Stepper Motor DC Dual H-Bridge for Arduino Smart Car Power UNO MEGA R3 Mega2560	1	$11
Total Cost		$300.11

APPENDIX E. VERIFICATION CROSS REFERENCE MATRIX

Verification Cross Reference Matrix (VCRM)

Number	Requirement	Level	Type	Test Approach	Verification Method	Rationale	Subsystem Allocation
GOV - 1	The car shall move responsive to hand gestures.	Threshold	Functional	Perform gesture input and observe corresponding car motion.	Demonstration	Requires real-world behavior confirmation → **System Test** with actual **gesture input** and **motion output**	Gesture Control Unit, Motion Control Unit
GOV - 2	The car shall indicate receipt of forward motion instruction using an LED.	Threshold	Functional	Send forward gesture and check LED activation.	Demonstration	Visual confirmation of output in response to input → best verified by **Integration Testing** and **Test method**	UI, Gesture Control Unit
GOV - 3	The car shall reroute its path in response to detecting an obstacle. (camera)	Threshold	Functional	Place obstacle and confirm automatic rerouting.	System Testing	Needs dynamic scenario testing to ensure safety logic → **System Test** with simulated obstacle	Obstacle Detection Unit, Motion Control Unit
GOV - 4	The car shall communicate with gesture controller via Bluetooth.	Threshold	Functional	Monitor Bluetooth data exchange between controller and car.	Demonstration	Ensuring wireless functionality between modules → **Integration Testing** with communication logs	Communication Module
GOV - 5	The car shall initiate an emergency stop function after attempting to reroute more than twice	Threshold	Functional	Trigger multiple reroutes and verify emergency stop.	System Testing	Requires behavior after conditional loop → **System Testing** under edge cases	Motion Control Unit, Software Engine
GOV - 6	All wheels shall be individually controlled by motors on the chassis.	Threshold	Functional	Send individual motor commands and observe response.	Inspection	Can be verified visually and with motor commands → **Unit Testing** for each wheel and **Inspection** of hardware	Motion Control Unit
GOV - 7	The car shall travel at least 8 mph.	Threshold	Functional	Measure speed using stopwatch or speed sensor.	Performance Testing	Quantitative speed measurement → **Performance Testing** with speed sensors or stopwatch	Motion Control Unit
GOV - 8	The gesture controller shall provide a real-time display of: speed, stop (coming to a complete halt), brake (slowing down), and object detection.	Threshold	Functional	Perform gesture inputs and verify display updates in real time.	Demonstration	Requires synchronized display output → Demonstration using live gestures and verifying display updates	UI, Gesture Control Unit, Obstacle Detection Unit
GOV - 9	The system shall log system uptime and resource usage.	Threshold	Functional	Review logs or memory usage report after runtime.	Inspection	Verifiable through software logs and code review → **Inspection** + **Test** (via tools or log dump)	System Monitoring
GOV - 10	The system shall respond to voice commands as a backup control mechanism.	Threshold	Functional	Issue voice command and confirm correct response.	Demonstration	Voice recognition and response → Demonstration using test voice input to confirm command execution.	Gesture Control Unit, Software Engine
GOV - 11	The car shall have a speaker.	Threshold	Functional	Visually inspect for speaker hardware on the chassis.	Inspection	Physical hardware presence → simple **Inspection**	UI
GOV - 12	The software shall provide real-time alerts for critical errors.	Threshold	Functional	Simulate fault and observe alert trigger.	System Testing	Safety-critical function → requires **System Testing** under fault conditions	Software Engine, UI
GOV - 13	The software shall distinguish between valid gestures and unintended movements.	Threshold	Functional	Input mixed gestures and verify correct classification.	System & Unit Testing	Needs both software logic and real input testing → **Unit + System Testing**	Software Engine, Gesture Control Unit
GOV - 14	The software shall generate a warning upon detecting obstacles within 2 feet.	Threshold	Functional	Place object within 2 feet and confirm warning response.	Integration Testing	Functional logic for proximity response → best checked via **Integration Testing** using sensors	Obstacle Detection Unit, UI
GOV - 15	The software shall support gesture-based activation for Dance Mode.	Threshold	Functional	Perform Dance Mode gesture and observe system activation.	Demonstration	Feature is triggered by a gesture → Demonstration confirms activation works as expected.	Gesture Control Unit, Software Engine
GOV - 16	The software shall control the car's LEDs to blink in sync with its dance movements.	Threshold	Functional	Activate Dance Mode and observe LED synchronization.	Demonstration	LED timing should match motion cues → Demonstration verifies sync in real-time.	UI, Software Engine
GOV - 17	ISO/IEC 30113-1:2025 for Gesture-based interface (Gesture Recognition and Control). -> ISO/IEC 30113-1:2025 provides a standardized framework for gesture-based interfaces, defining how gestures are recognized, interpreted, and used to control devices. It ensures consistency, interoperability, and usability across different systems that rely on gesture recognition and control.	Threshold	Functional	Review code and system logic against ISO standard.	Inspect / Analysis	Requires reviewing specs and ensuring alignment → **Analysis/Inspection** of implementation vs standard	Standards Compliance, Gesture Control Unit
GOV - 18	ISO/IEC 25010:2024 for Software Quality Requirements and User Interface. - > Implement mechanisms to ignore or correct noisy inputs (e.g., accidental hand movements).	Threshold	Functional	Perform code review and simulate noisy inputs.	Inspect / Analysis	Functional quality trait → validate via **code review** and **test** under noisy inputs	Standards Compliance, Software Engine
GOV - 19	The system shall enter low-power mode when idle for more than 5 minutes.	Threshold	Functional	Leave system idle and verify mode change after 5 minutes.	System Testing	Needs time-based system response → verify via **System Test** with timers	Power Management
GOV - 20	The system shall provide a warning signal when the battery level falls below 10%.	Threshold	Functional	Simulate low battery condition and confirm warning alert.	System Testing	System safety threshold → best verified with simulated battery level in **System Test**	Power Management, UI

i