Ain Shams University – Faculty of Engineering

Computer Engineering and Software Systems Program

CSE411 - Real-Time and Embedded Systems Design

Automotive Smart Safety System: Intelligent Door Locking & Rear Parking Assist

Team 24

Name	ID
Youssef Hany Emil	21P0082
Youssef Adel Albert	21P0258
Youssef Sameh Khairat	20P9605
Adham Osama	22P0071
Mark Saleh Sobhi	21P0206
Ahmed Tarek Mahmoud	2100561

Drive Link

Github Link

Contents

Abstract

The Automotive Smart Safety System is a real-time embedded project designed to simulate two essential vehicle safety features—intelligent door locking and rear parking assistance—within a modular, multitasking embedded framework. Developed using the TM4C123GH6PM microcontroller and FreeRTOS, the system models real-world automotive behavior through sensor integration, task scheduling, and event-driven programming. A potentiometer emulates vehicle speed, triggering automatic door locking when thresholds are exceeded, while manual switches allow user override. Rear obstacle detection is achieved via the HC-SR04 ultrasonic sensor, with alerts issued using a buzzer, LCD, and RGB LED. The system not only delivers a comprehensive learning experience in embedded systems design but also proposes scalable enhancements for future real-world implementations.

Introduction

As vehicles evolve toward smarter and safer designs, the integration of embedded systems into automotive safety features becomes increasingly essential. Modern cars depend on real-time data processing, sensor integration, and intelligent control logic to enhance driver and passenger safety. This project introduces a prototype Automotive Smart Safety System that simulates two key automotive safety mechanisms: intelligent car door locking based on vehicle speed and rear parking assistance for obstacle detection.

The project targets real-time responsiveness and task coordination using **FreeRTOS** on the **TM4C123GH6PM** microcontroller, a widely used platform in embedded system education and development. This system demonstrates how multiple safety-critical functions can coexist and cooperate in a resource-constrained environment. The smart door locking feature automates door control based on motion detection, while the rear parking assistant provides dynamic alerts based on obstacle proximity. Through this project, students learn essential concepts in sensor interfacing, multitasking, inter-task communication, and real-time embedded software design.

Project Overview

This embedded systems project aims to enhance vehicular safety through the development of an intelligent, real-time automotive safety system. The system incorporates two core functionalities:

1. **Intelligent Car Door Locking System**: Uses a potentiometer to simulate vehicle motion. The system automatically locks the doors if the vehicle exceeds a predefined speed threshold and unlocks them upon ignition shutdown.
2. **Rear Parking Assistance System**: Employs an ultrasonic sensor to detect rear obstacles when the car is in reverse gear. It provides real-time visual and auditory feedback to the driver to prevent collisions.

The microcontroller used is the **TM4C123GH6PM (Tiva C Series)**, and the firmware is built on **FreeRTOS**, enabling multitasking with real-time responsiveness. The project is designed to reflect real-world automotive systems while teaching embedded software design principles including task scheduling, sensor interfacing, inter-task communication, and priority handling.

System Design and Architecture

The system is modular and task-based, with different sensors and user interfaces triggering various events. Key modules include:

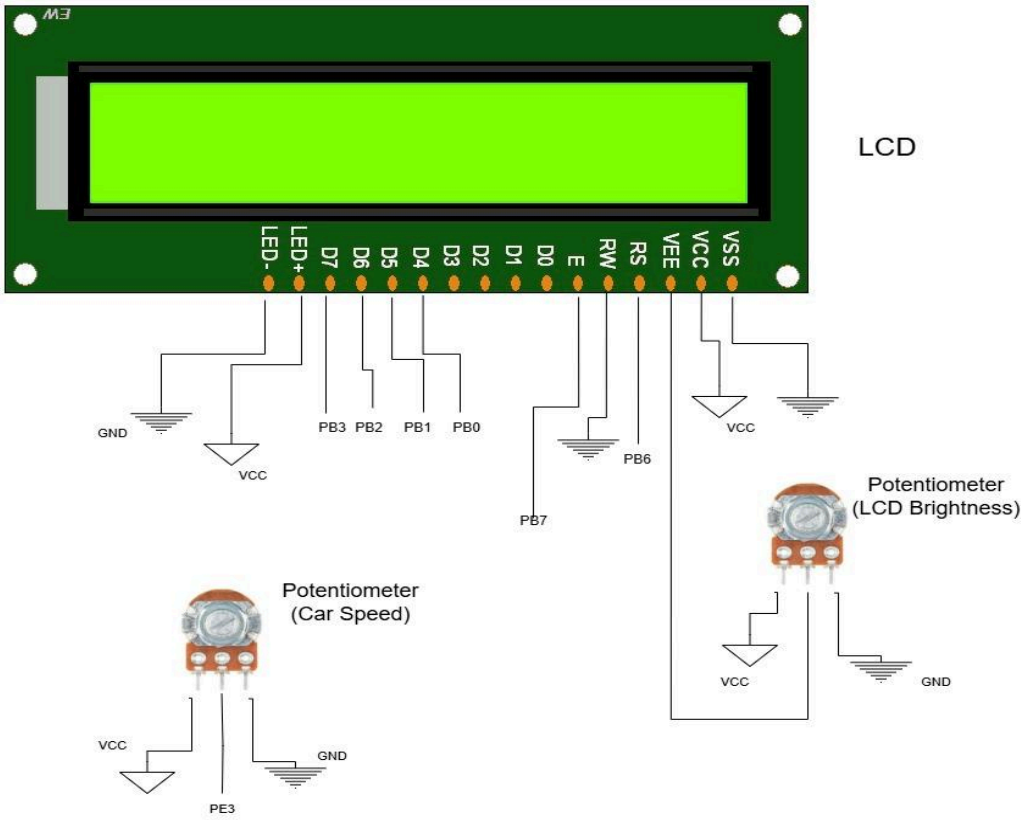
- **Speed Detection and Door Locking**: Reads analog data from a potentiometer, simulates vehicle speed, and triggers locking mechanisms.
- **Manual Door Controls**: Allows the driver to override automatic logic using pushbuttons.
- **Driver Door Monitoring**: Uses a switch to simulate the driver door status (open/closed). Triggers alerts if the door is open while the car is on.
- **Reverse Gear Detection and Proximity Sensing**: When the reverse switch is active, the HC-SR04 ultrasonic sensor begins scanning for nearby objects.
- **Display and Alerts**: Outputs real-time information on an LCD and uses a buzzer and RGB LED for immediate feedback.

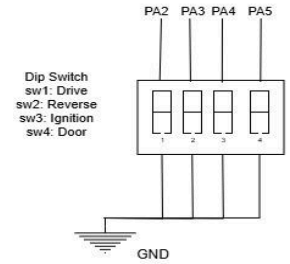
These components interact via shared queues and synchronization mechanisms (mutexes) to maintain system integrity and performance.

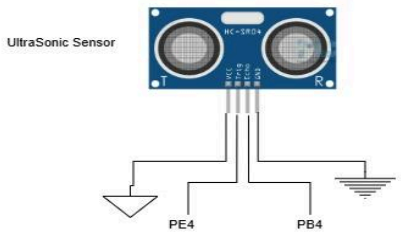
Hardware Components

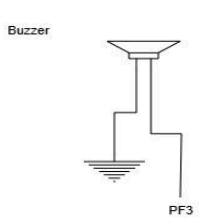
- **TM4C123GH6PM Microcontroller**: Central controller with GPIOs, timers and ADCs.
- **Potentiometer**: Simulates vehicle acceleration. Its analog value is read using the ADC and converted to a speed metric.
- **HC-SR04 Ultrasonic Sensor**: Measures distance to objects.
- **Switches**:
    - Gear simulation: Neutral, Drive, and Reverse
    - Ignition switch
    - Manual Lock and Unlock
    - Driver door state
- **RGB LED**: Indicates object proximity:
    - Green: Safe (>100 cm)
    - Yellow: Warning (30-100 cm)
    - Red: Danger (<30 cm)
- **Red LED**: Indicates doors locked or unlocked.
- **Buzzer**: Emits variable frequency alerts based on proximity or door safety status.
- **LCD Display**: Shows status messages and system readings like current speed and distance.

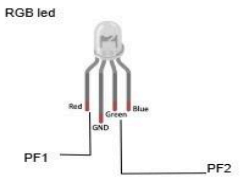
Wiring Diagrams

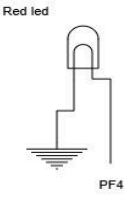


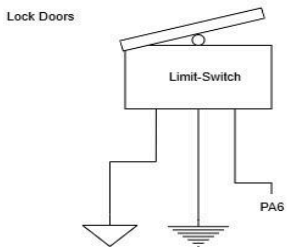


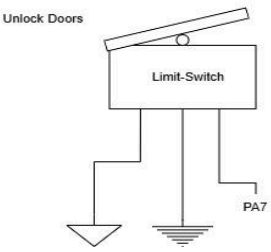












Features and Functionalities

- **Speed-Based Door Locking**: Vehicle speed is calculated using a potentiometer. When speed exceeds a predefined threshold (e.g., 10 km/h), doors are automatically locked.
- **Ignition-Linked Door Unlocking**: Turning off the ignition switch triggers automatic door unlocking, simulating safe vehicle shutdown.
- **Manual Lock/Unlock Override**: Two limit-switches allow the user to manually lock or unlock the doors at any time.
- **Gear Shift Simulation**: Switches simulate the gear shifter to switch between Park, Drive, and Reverse.
    - Switch 1: Drive
    - Switch 2: Reverse
    - Switch 3: Ignition
    - Switch 4: Door status (open/closed)
- **Driver Door Monitoring**: A switch monitors the state of the driver's door. If the door is open while the vehicle is on, a buzzer alert and LCD warning are triggered.
- **Real-Time Speed Display**: The LCD continuously updates the vehicle speed measured by the potentiometer.
- **Rear Obstacle Detection**: The HC-SR04 ultrasonic sensor activates in reverse gear and measures the distance to the nearest object.
- **Buzzer Alerts for Parking Assist**: The buzzer's beep frequency increases as objects get closer, giving the driver auditory proximity cues.
- **RGB LED Indication**:
    - Green: Object > 100 cm (Safe)
    - Yellow: Object between 30–100 cm (Caution)
    - Red: Object < 30 cm (Danger)

FreeRTOS Implementation Details

In this project, FreeRTOS is utilized to implement a multitasking real-time system on the TM4C123GH6PM microcontroller. The system comprises three main tasks, each responsible for a distinct component of the automotive smart safety system: intelligent door locking and rear parking assistance.

Task Structure

The function createSafetyTasks() initializes and creates all required system tasks along with the necessary synchronization primitives:

1. **doorTask**
    - **Function:**

      This task manages the intelligent door locking system. It monitors vehicle speed using the ADC, sends the speed data to the **speedQueue**, and automatically locks the doors based on a speed threshold (10 km/h), gear position, and door status. It also handles manual lock/unlock commands and manages buzzer activation and LCD alerts if the driver's door is opened while the car is on and unlocked.

      When the ignition is turned off, the task ensures the doors are unlocked and the buzzer is deactivated for safety.

    - **Key Operations:**
        i. Reads current speed via ADC and sends it to **speedQueue**.
        ii. Auto-locks doors if speed > 10 km/h and conditions are safe.
        iii. Responds to manual lock/unlock lever inputs.
        iv. Activates buzzer and displays "Door Opened" warning if needed.
        v. Resets system state when ignition is off.
    - **Periodicity:**

      Executes every 50 ms (20 Hz) for fast responsiveness.

    - **Priority:**

      Priority 3 — the highest in the system — ensuring rapid response to speed and safety conditions..

- **Resources:**
    i. Sends speed data to **speedQueue**.
    ii. Accesses **lcdMutex** to update the LCD safely.
    iii. Uses door lock actuator, buzzer, and related GPIO controls.

2. **rearAssistTask**

    - **Function:**

      This task implements the rear parking assist feature. When the ignition is on and the gear is in Reverse, it continuously reads distance data from the HC-SR04 ultrasonic sensor. Based on the measured distance, it updates the RGB LED color and controls the buzzer frequency to alert the driver. It also sends the distance value to **distanceQueue** for display.

    - **Key Operations:**
        i. Reads distance when car is in Reverse and ignition is ON.
        ii. Sends latest distance data to **distanceQueue**.
        iii. Controls buzzer frequency based on proximity:
            1. Green LED: Safe (>100 cm)
            2. Yellow LED: Caution (30–100 cm)
            3. Red LED: Danger (<30 cm)
        iv. Deactivates RGB LED and buzzer if distance is invalid or car exits Reverse.

    - **Periodicity:**

      Executes every 20 ms (50 Hz) for high responsiveness during parking maneuvers.

    - **Priority:**

      Priority 2 — shared with the **lcdUpdateTask**.

    - **Resources:**
        i. Sends distance data to **distanceQueue**.
        ii. Uses GPIOs to control RGB LED and buzzer.
        iii. Coordinates with **doorTask** to avoid buzzer conflicts.

3. **lcdUpdateTask**

    - **Function:**

      This task is responsible for managing the LCD output. It monitors ignition state changes to display "Car ON" or "Car OFF" messages, and periodically updates the display with gear status, door lock status, speed, and distance. It ensures safe access to the LCD via a mutex and avoids overwriting warning messages like "Door Opened" when active.

    - **Key Operations:**
        i. Displays ignition messages ("Car ON", "Car OFF").
        ii. Shows gear status (N, D, R) and door lock status (L/UL).
        iii. Reads from **speedQueue** and **distanceQueue** to display live data.
        iv. Handles conditional overwriting when door warnings are active.
        v. Resets display when the ignition state changes.

    - **Periodicity:**

      Executes every 100 ms (10 Hz) to balance performance and readability.

    - **Priority:**

      Priority 2 — same as **rearAssistTask** to enable cooperative execution.

    - **Synchronization:**

      Uses **lcdMutex** to ensure safe, mutually exclusive access to the LCD, preventing display corruption due to concurrent access.

Inter-Task Communication

- **Queues:**
    - **speedQueue:**
        - Used by **doorTask** to send the latest vehicle speed (as an integer) to **lcdUpdateTask**.
        - Queue length is **1**, which ensures only the most recent value is retained.
        - Old data is overwritten using **xQueueOverwrite()** if the queue is full — this prevents stale values from being displayed on the LCD.
    - **distanceQueue:**
        - Used by **rearAssistTask** to send the measured distance (as a uint32_t) to **lcdUpdateTask**.
        - Also configured with a length of **1** to preserve only the latest proximity data.
        - Ensures real-time reflection of distance changes in parking scenarios.

- **Mutex:**
    - **lcdMutex:**
        - A **binary mutex** created using **xSemaphoreCreateMutex()** to control access to the LCD.
        - Ensures **mutually exclusive access** so that only one task (either **doorTask** or **lcdUpdateTask**) can write to the display at a time.
        - Prevents display corruption due to concurrent writes and maintains **UI consistency**.
        - Access is protected using **xSemaphoreTake()** with timeout, followed by **xSemaphoreGive()** after display operations.

Task Scheduling and Prioritization

The system leverages **FreeRTOS's preemptive scheduling** model, where the highest-priority task that is ready to run is always selected for execution. This ensures that time-critical operations receive CPU attention as needed, enabling responsive and deterministic system behavior.

Priority Assignment Rationale

1. **doorTask — Priority 3 (Highest)**
   This task handles intelligent door locking, a safety-critical function that must react promptly to changes in vehicle speed and door status. Assigning it the highest priority ensures:
   - Immediate response when speed exceeds thresholds.
   - Fast actuation of lock/unlock commands.
   - Reliable enforcement of safety rules related to door and ignition state.

2. **rearAssistTask — Priority 2**
   This task provides real-time feedback during reversing, including obstacle detection and auditory/visual alerts. Although important, it is slightly less critical than door locking and thus given the next highest priority.

3. **lcdUpdateTask — Priority 2**
   Shares the same priority as **rearAssistTask** to allow balanced multitasking for real-time user feedback. Display updates are essential for usability but are not safety-critical, so they are designed to yield to **doorTask** when needed.

Scheduling Strategy

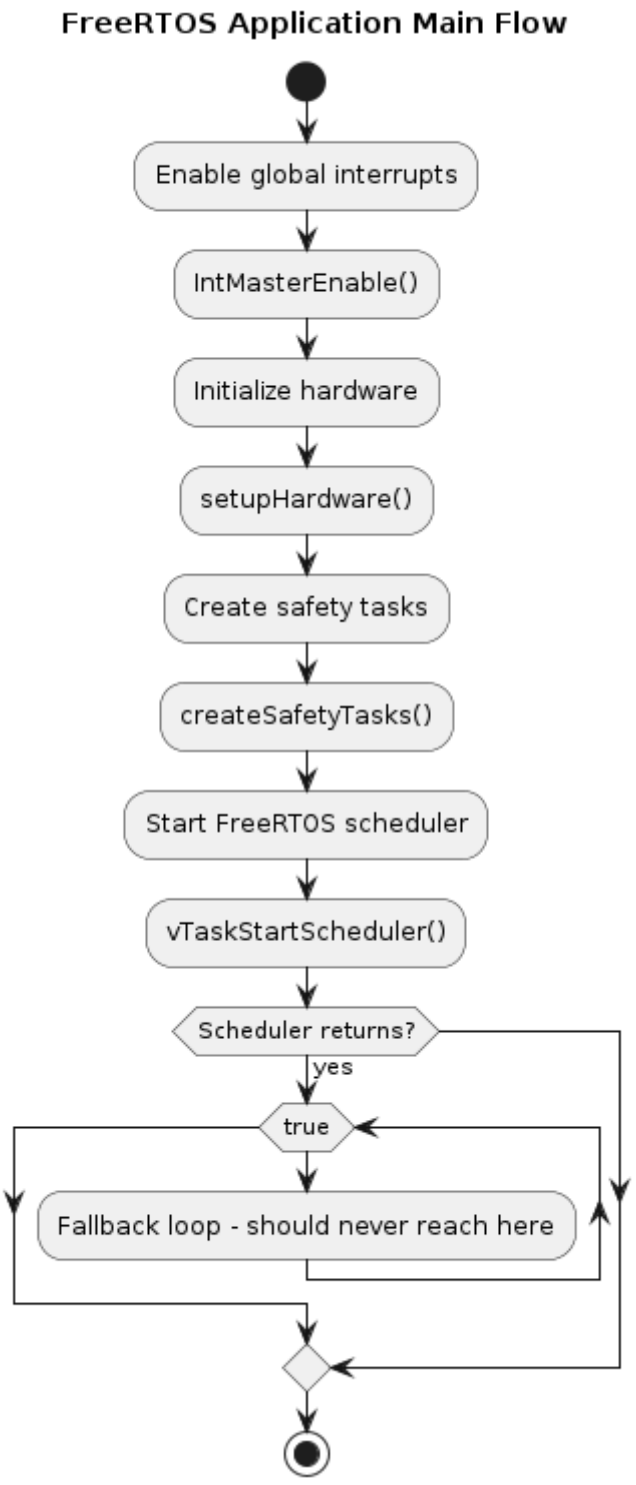
- **Preemption**: If **doorTask** becomes ready while other tasks are running, it preempts them to execute immediately due to its higher priority.
- **Cooperative Execution**: **rearAssistTask** and **lcdUpdateTask** are scheduled based on availability and run cooperatively unless preempted by **doorTask**.
- **Periodic Delays** (**vTaskDelay()**): Each task includes a delay period based on its expected response time, reducing CPU load and ensuring predictability:
  - **doorTask**: 50 ms (20 Hz)
  - **rearAssistTask**: 20 ms (50 Hz)
  - **lcdUpdateTask**: 100 ms (10 Hz)

Benefits of This Scheduling Approach

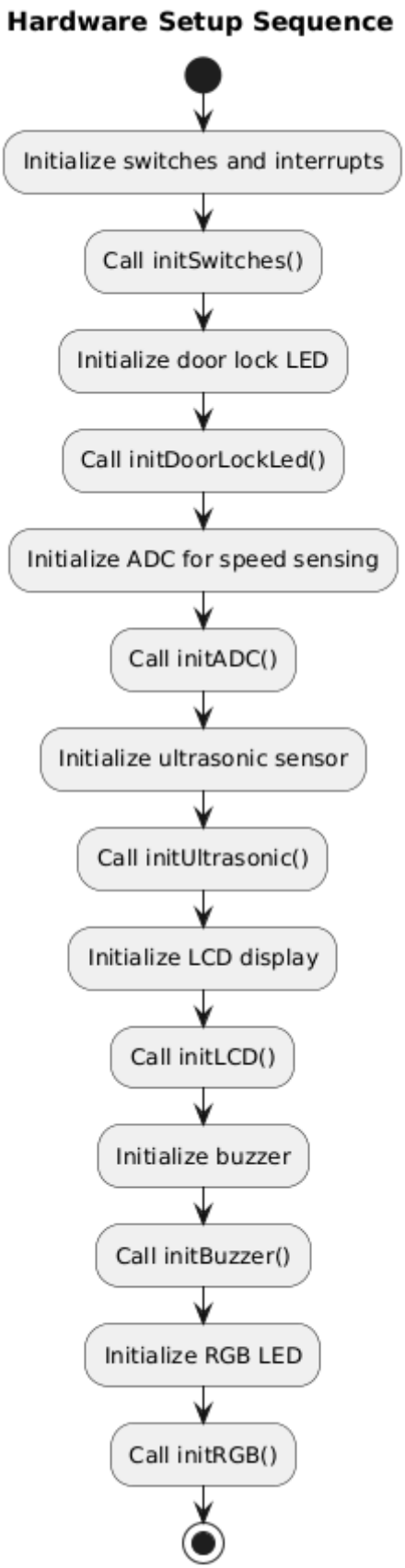
- **Responsiveness**: High-priority tasks like door safety logic respond quickly to real-world changes.
- **Real-Time Feedback**: Parking assist indicators and LCD updates are timely and synchronized.
- **Modularity and Maintainability**: Each task is logically separated, making the system easy to understand, test, and extend.
- **Resource Efficiency**: Proper prioritization and task delays help minimize CPU overhead and improve power efficiency.

Code Flow

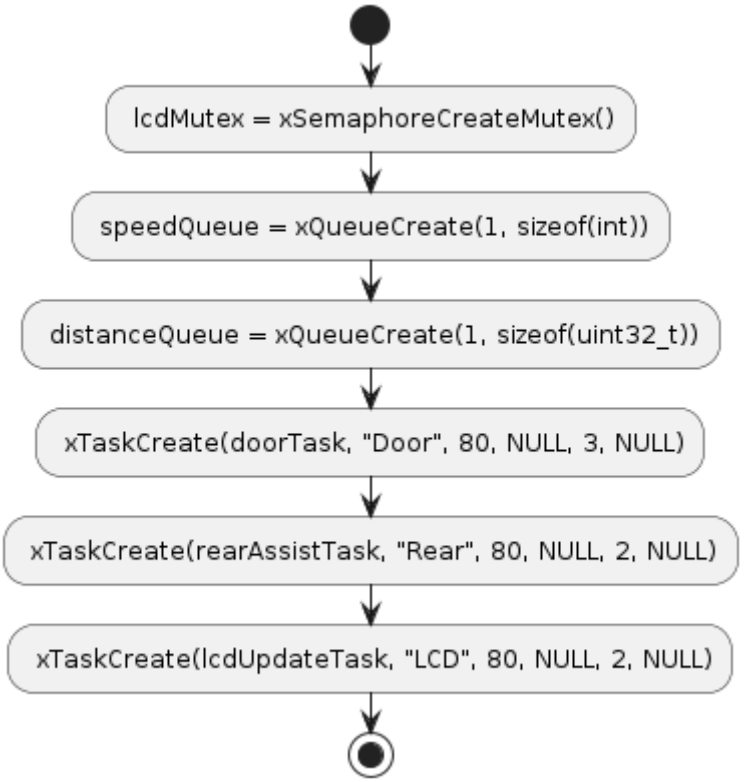
Main



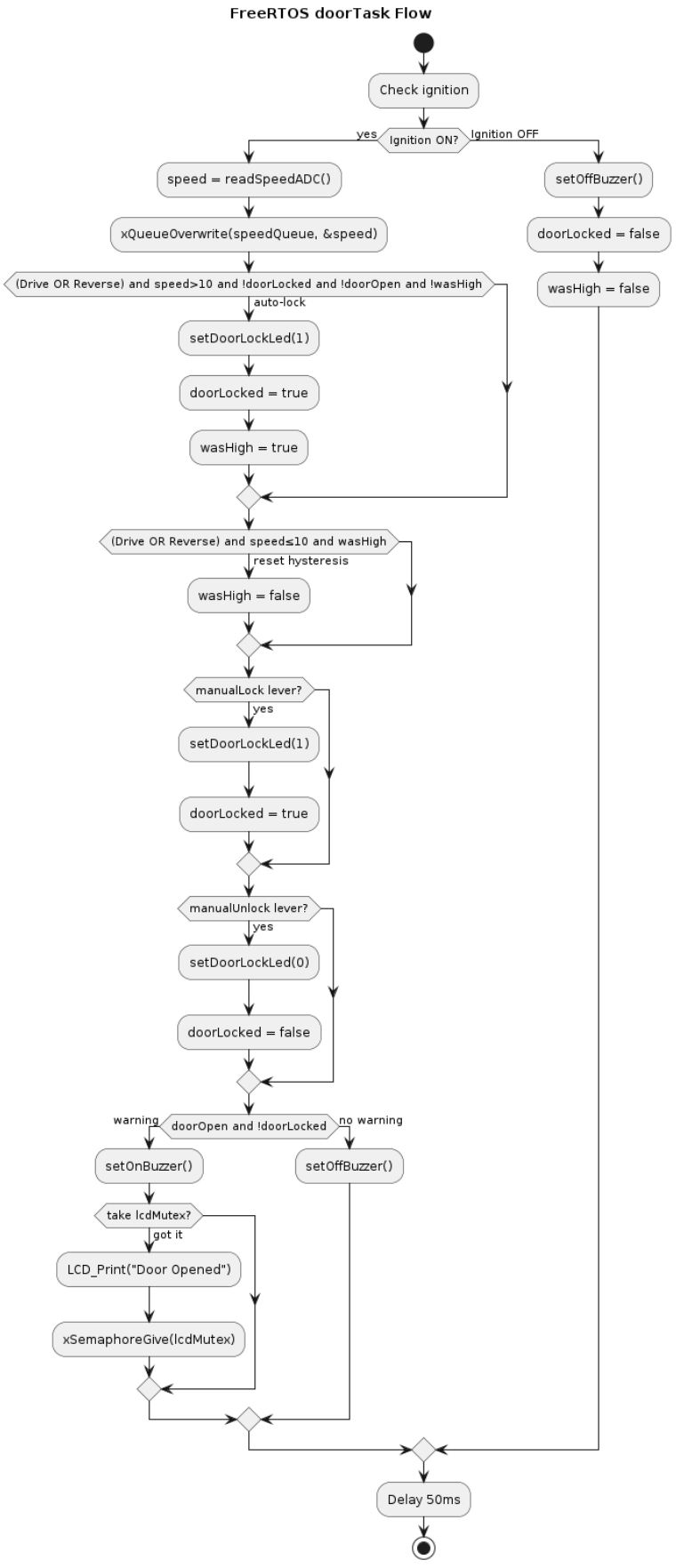
Setup Hardware Function



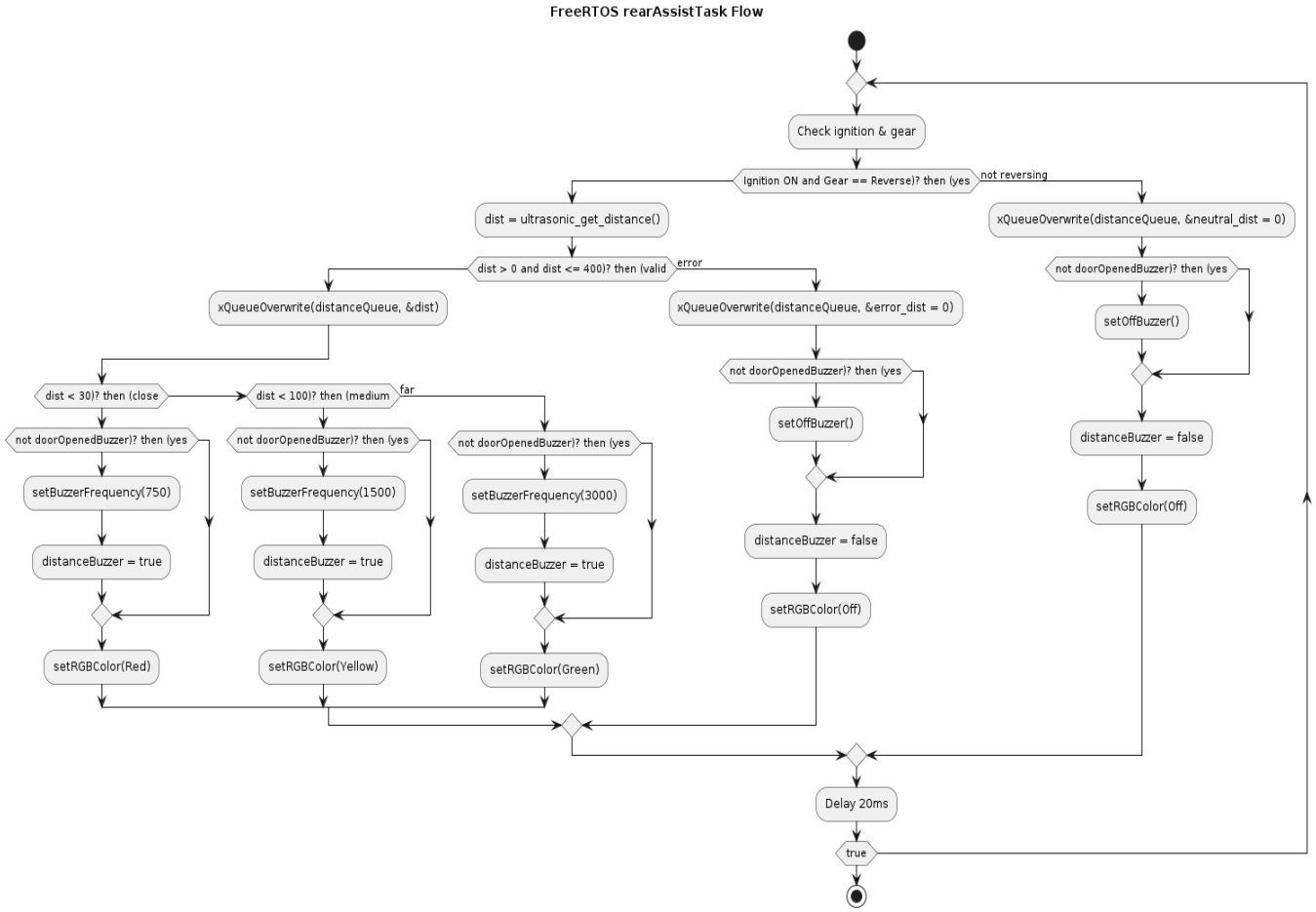
Create Safety Tasks Function



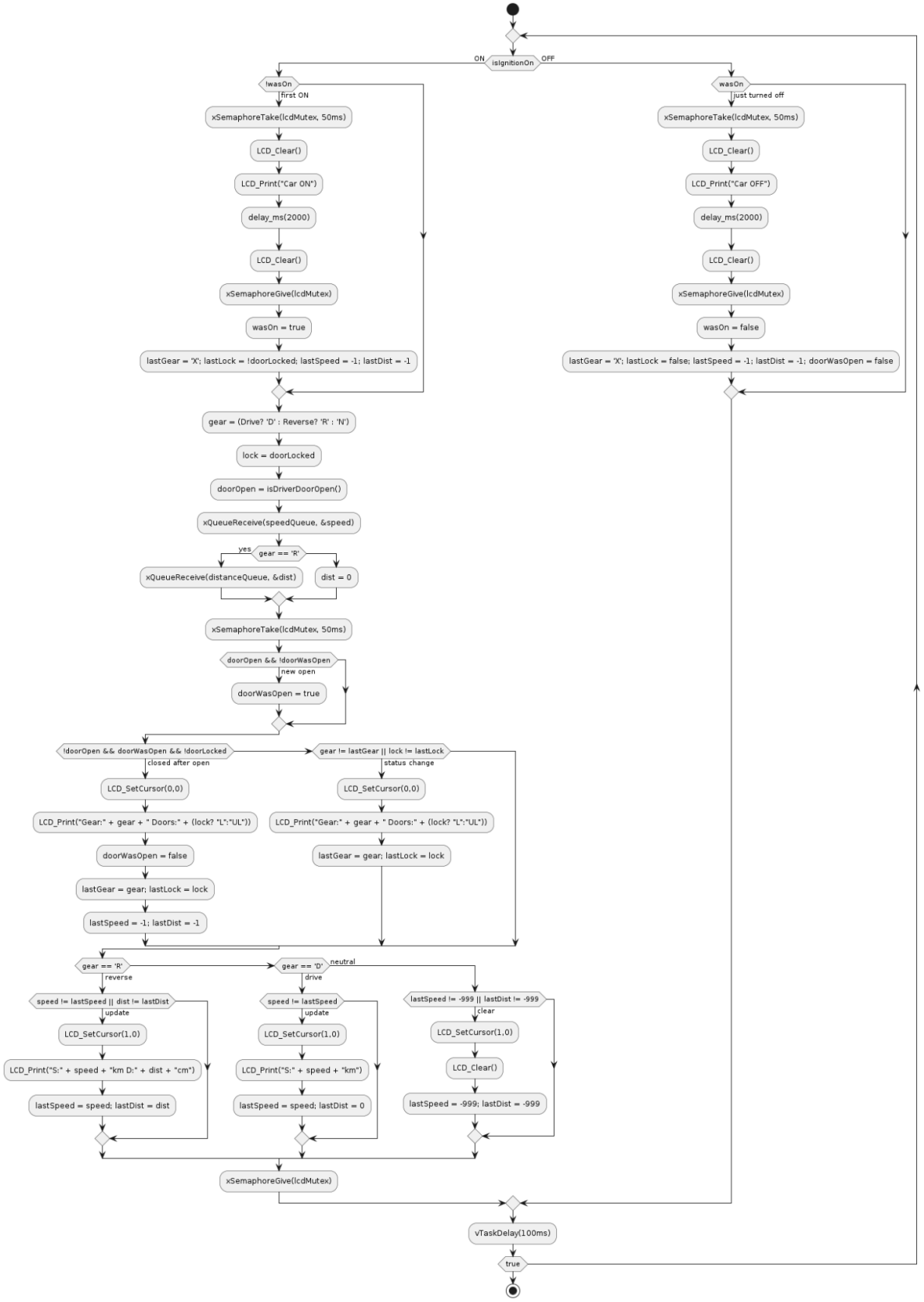
Door Task


FreeRTOS doorTask Flow
Check ignition
Ignition ON?
yes
Ignition OFF
speed = readSpeedADC()
xQueueOverwrite(speedQueue, &speed)
setOffBuzzer()
doorLocked = false
wasHigh = false
(Drive OR Reverse) and speed>10 and !doorLocked and !doorOpen and !wasHigh
auto-lock
setDoorLockLed(1)
doorLocked = true
wasHigh = true
(Drive OR Reverse) and speed≤10 and wasHigh
reset hysteresis
wasHigh = false
manualLock lever?
yes
setDoorLockLed(1)
doorLocked = true
manualUnlock lever?
yes
setDoorLockLed(0)
doorLocked = false
warning
doorOpen and !doorLocked
no warning
setOnBuzzer()
setOffBuzzer()
take lcdMutex?
got it
LCD_Print("Door Opened")
xSemaphoreGive(lcdMutex)
Delay 50ms

Rear Assist Task

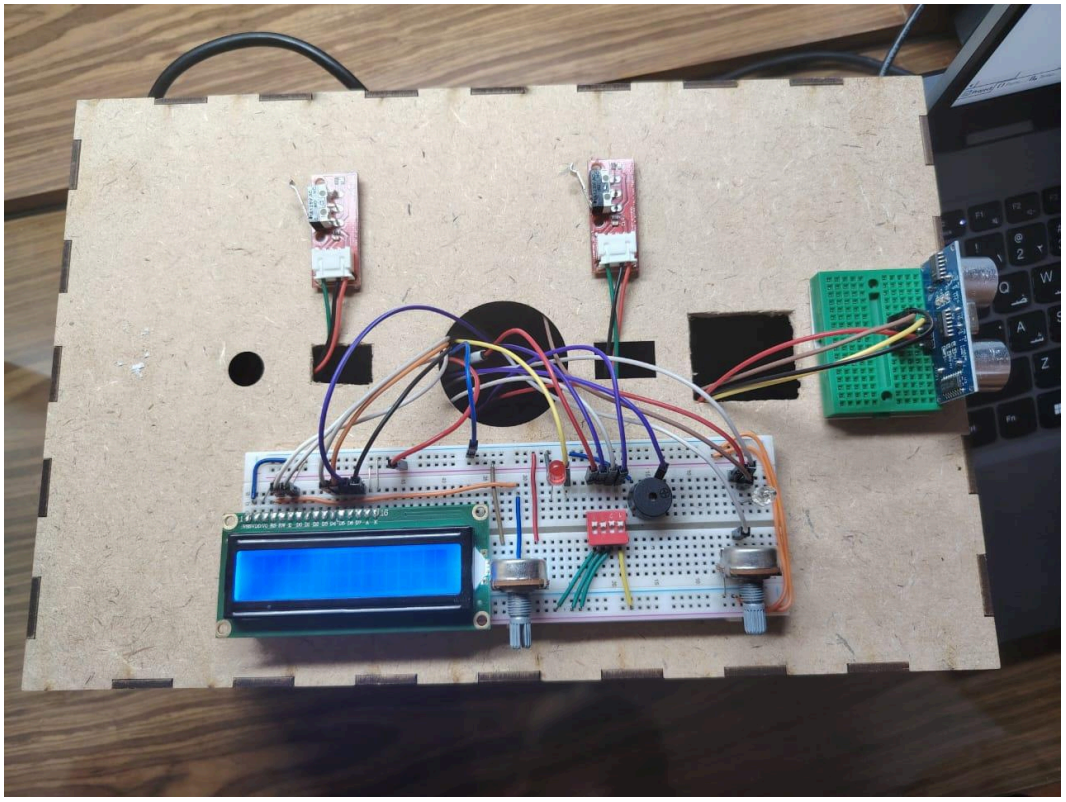


LCD Update Task

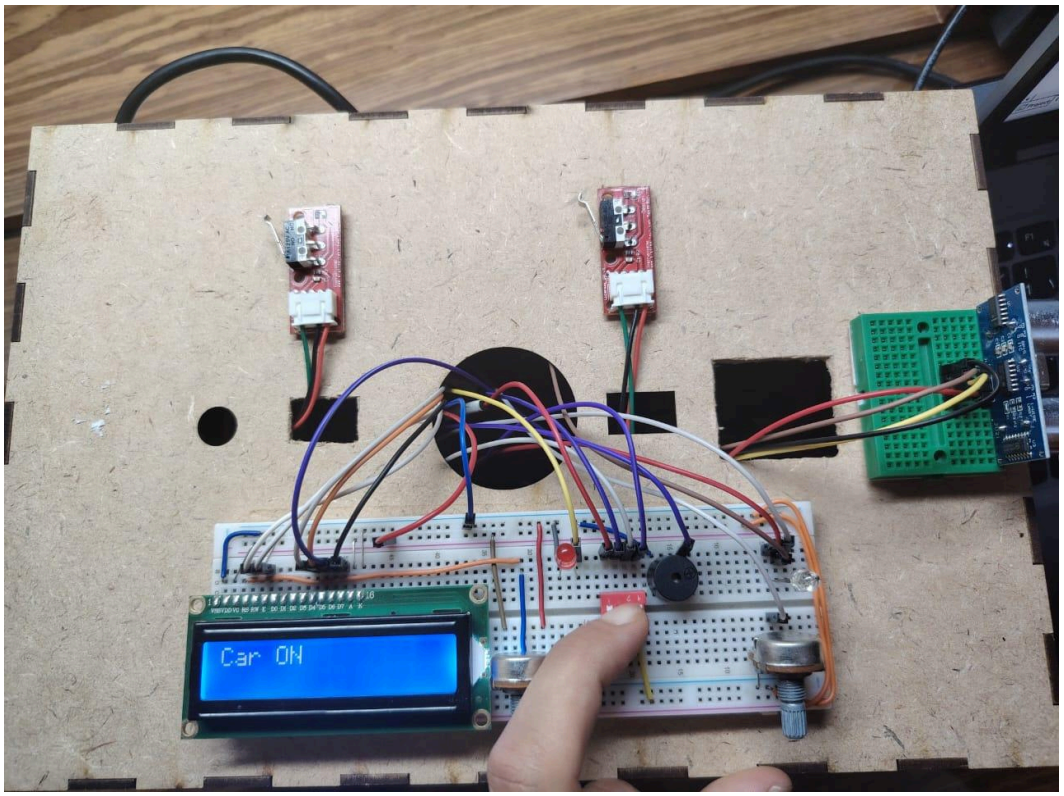


Prototype Photos

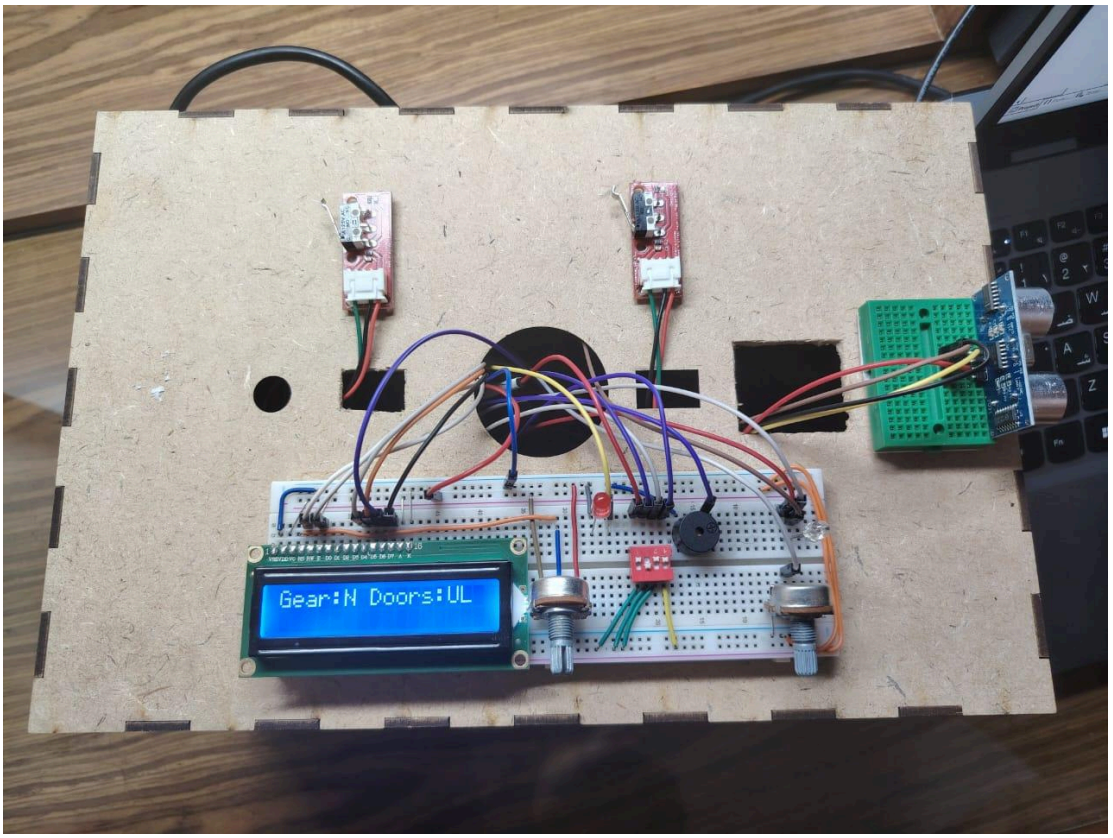
Ignition off



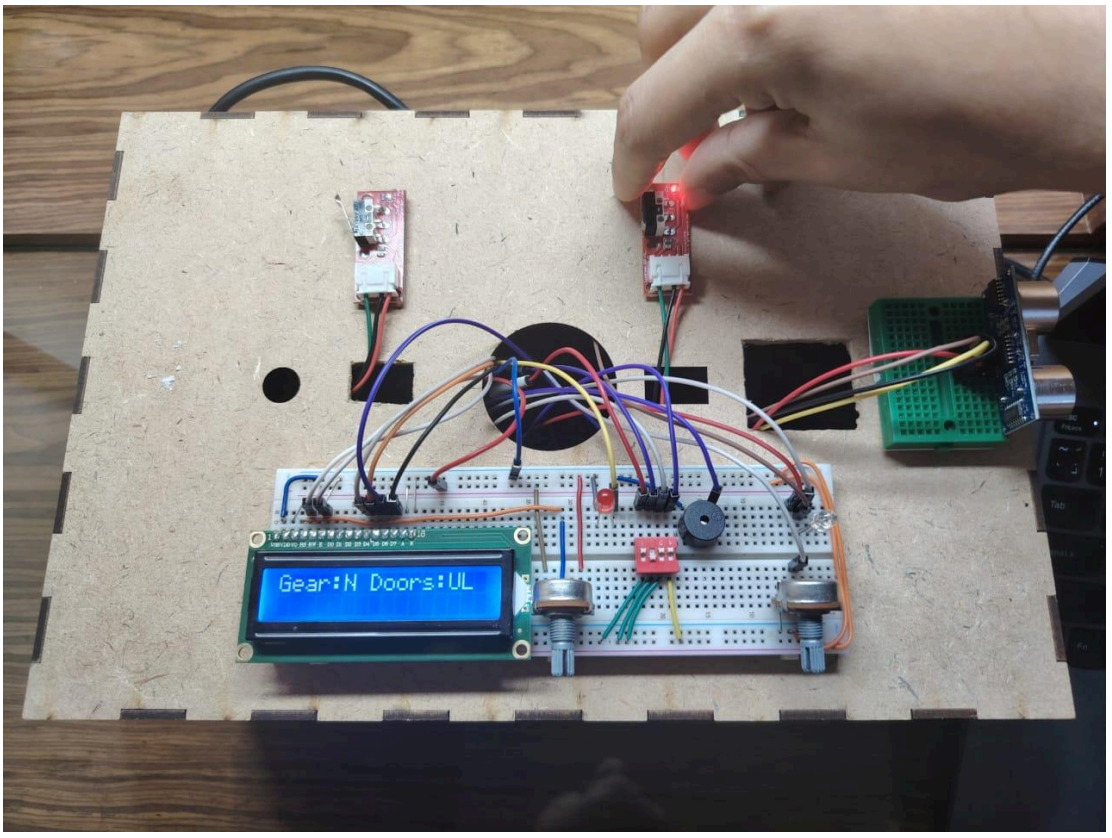
Turning Ignition on



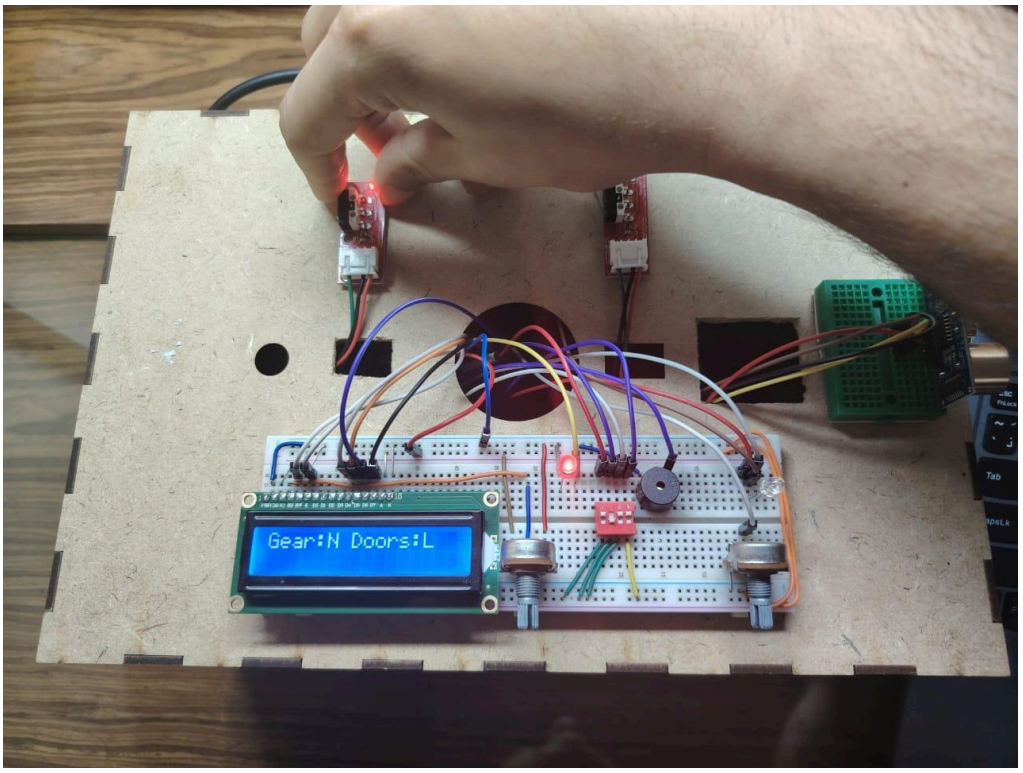
Gear: Neutral



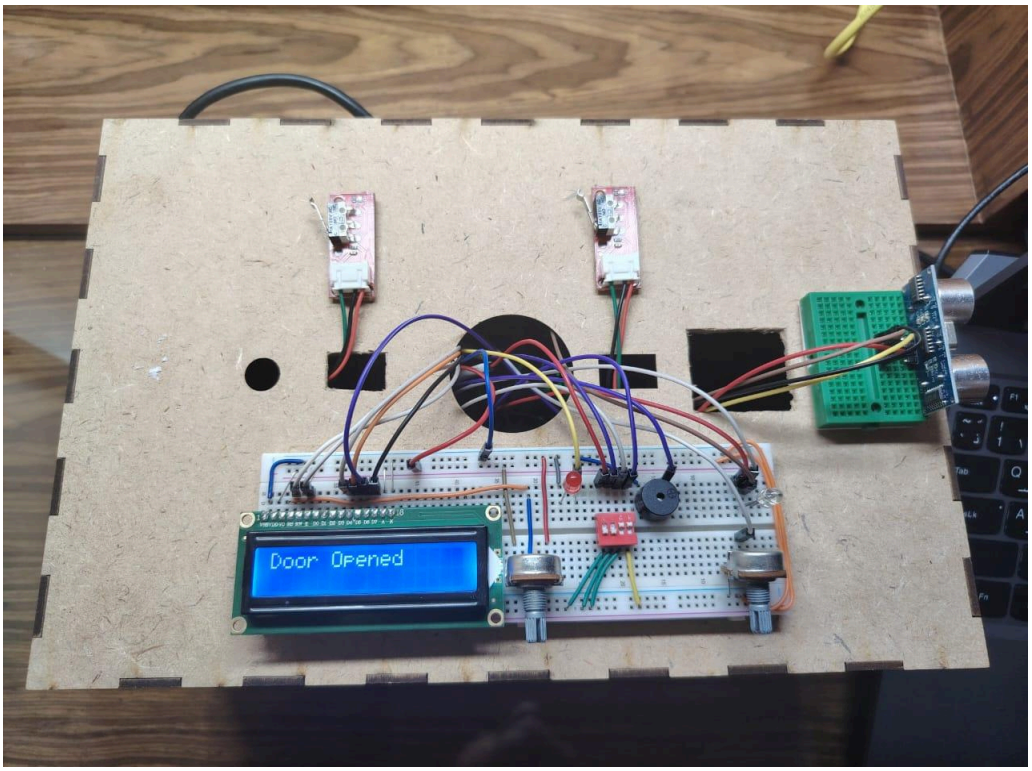
Manually Unlocking Doors



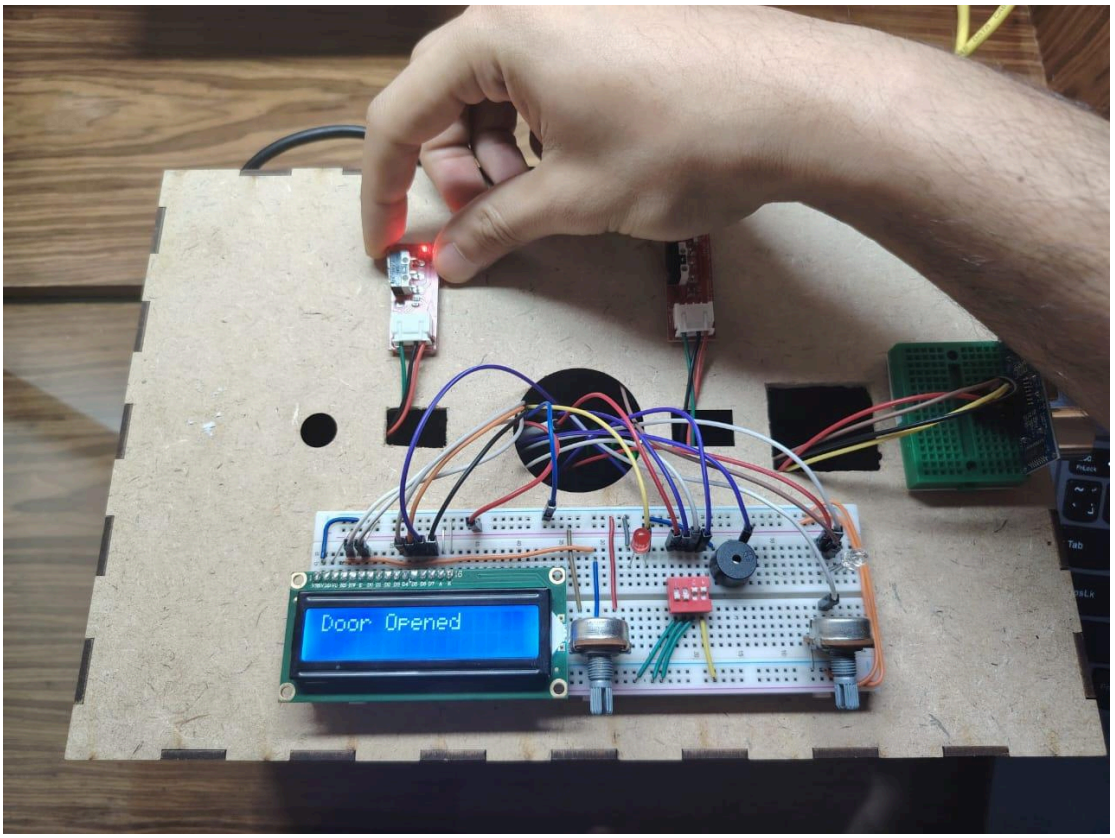
Manually Locking Doors



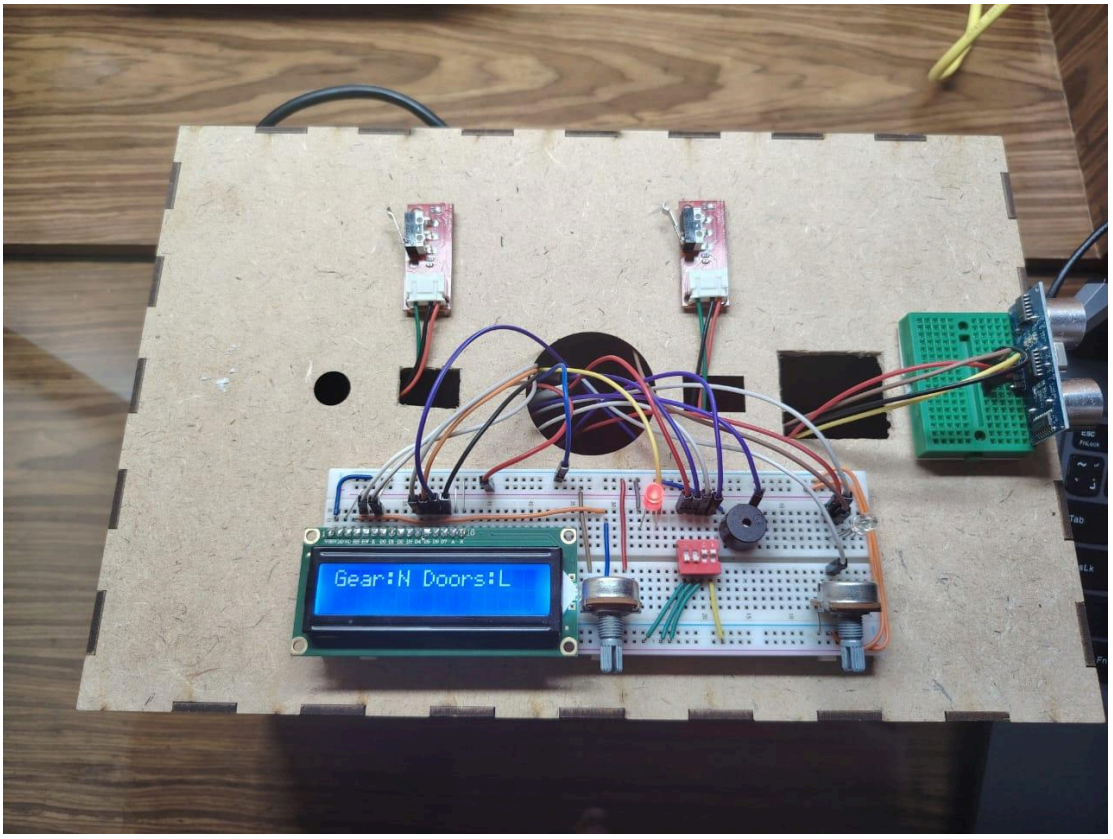
Opening Driver’s Door



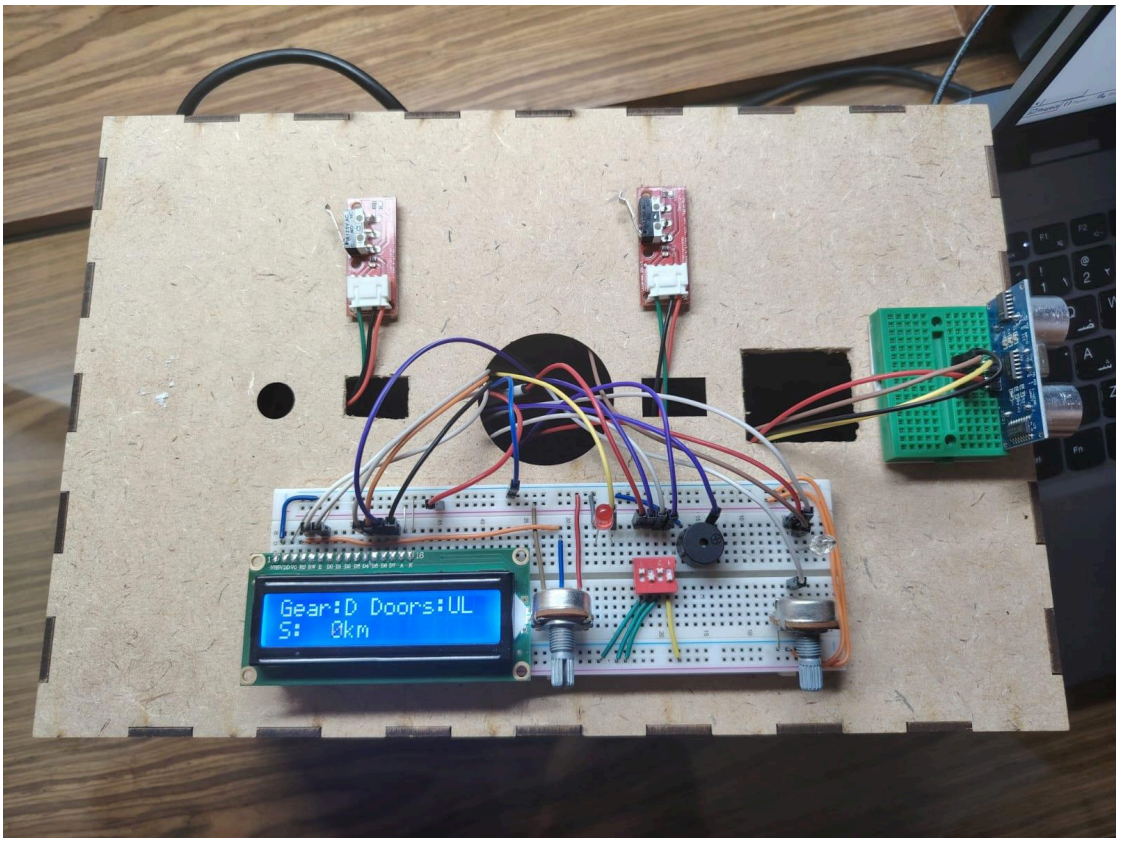
Locking Door while Driver's Door is opened



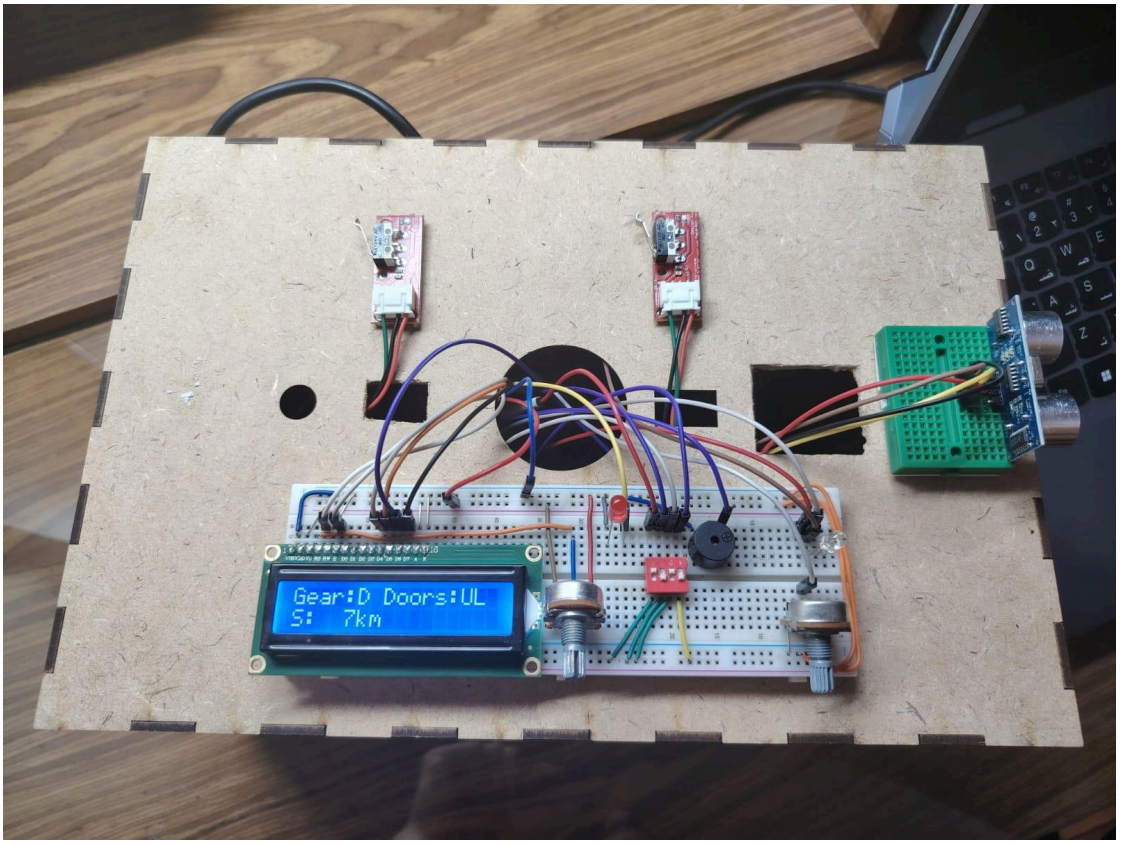
Opening Driver's Door while Door is locked



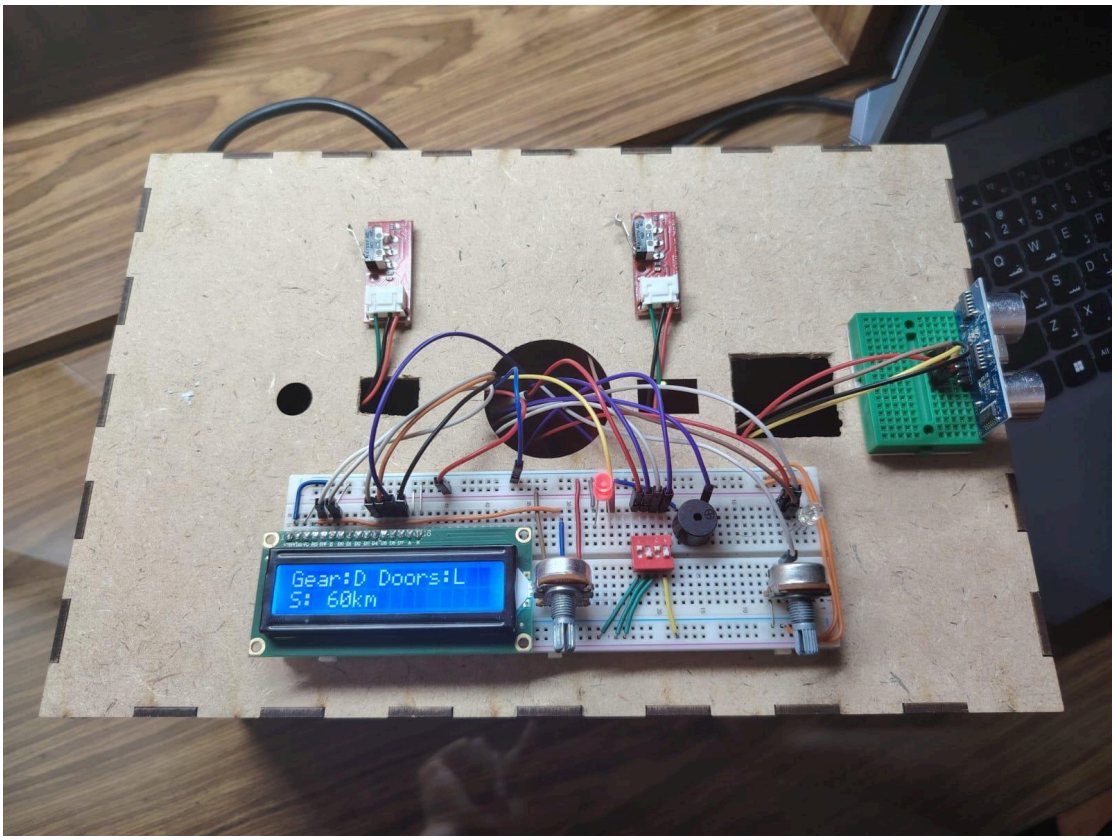
Gear: Drive



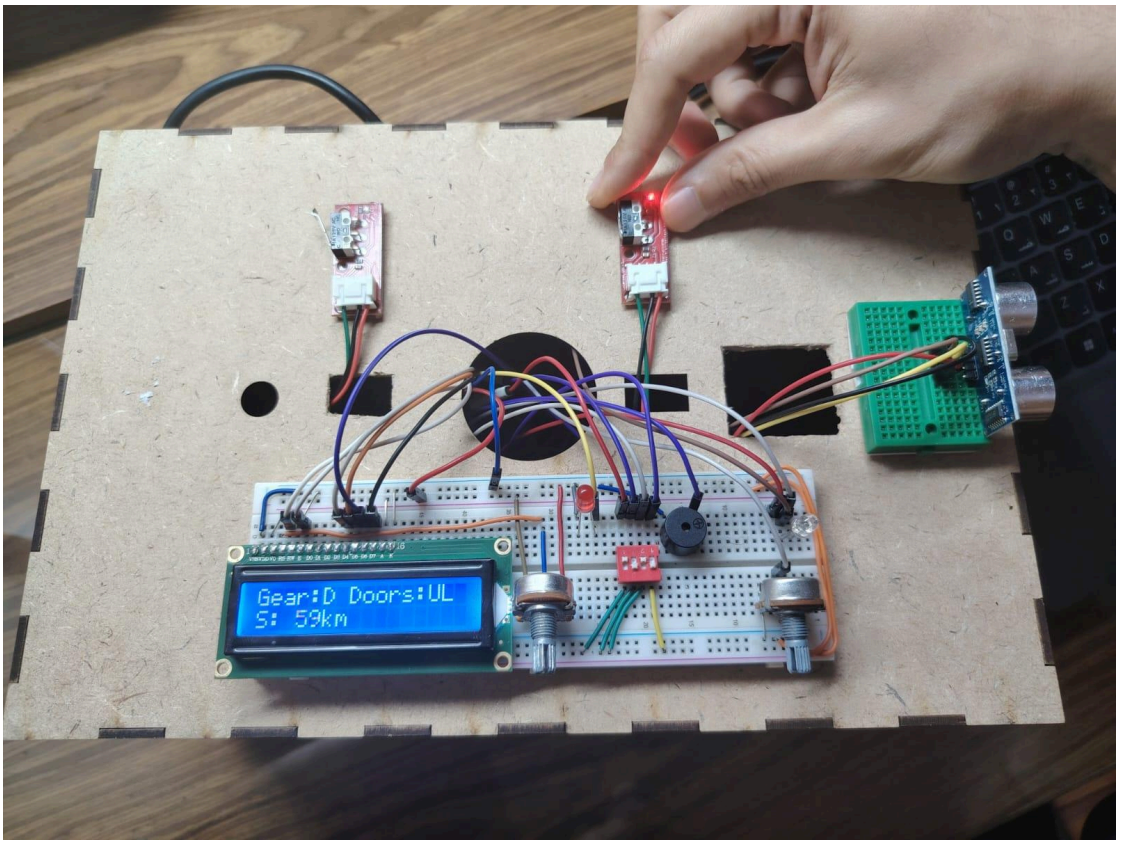
Speed below 10 km/h



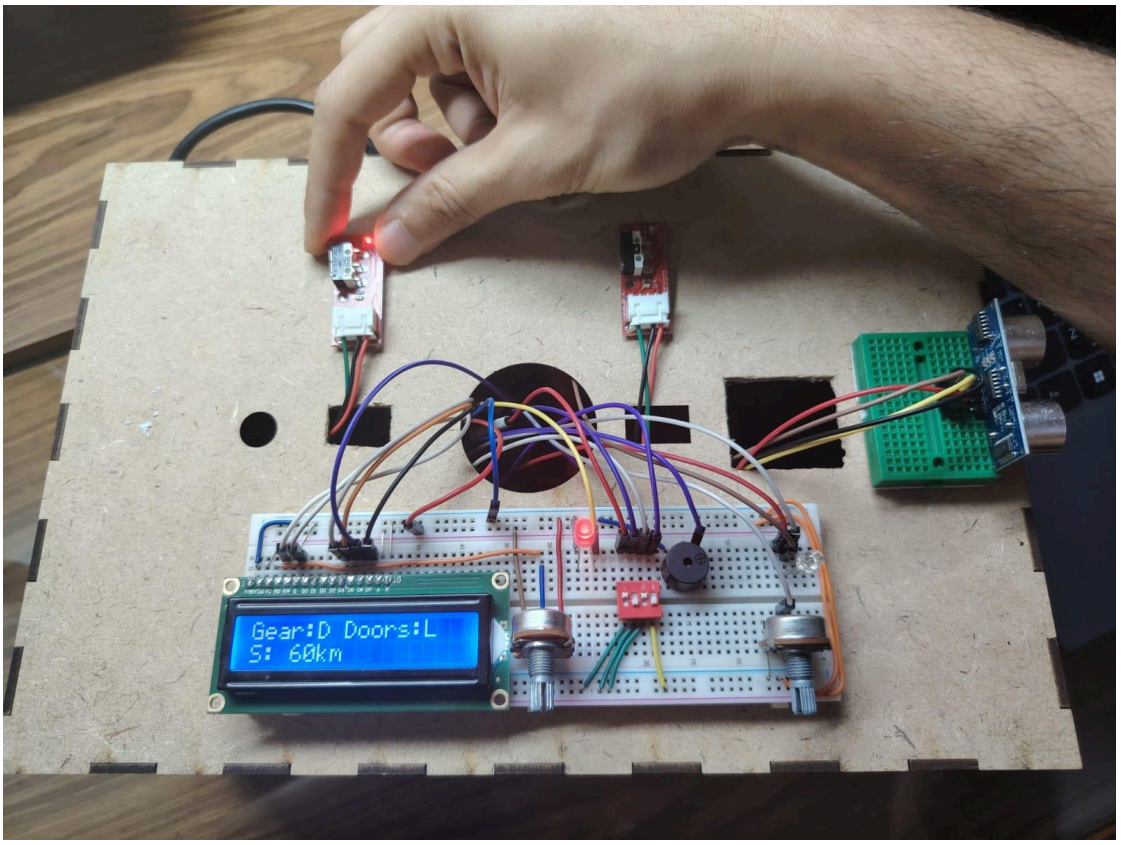
Speed above 10 km/h (doors automatically lock)



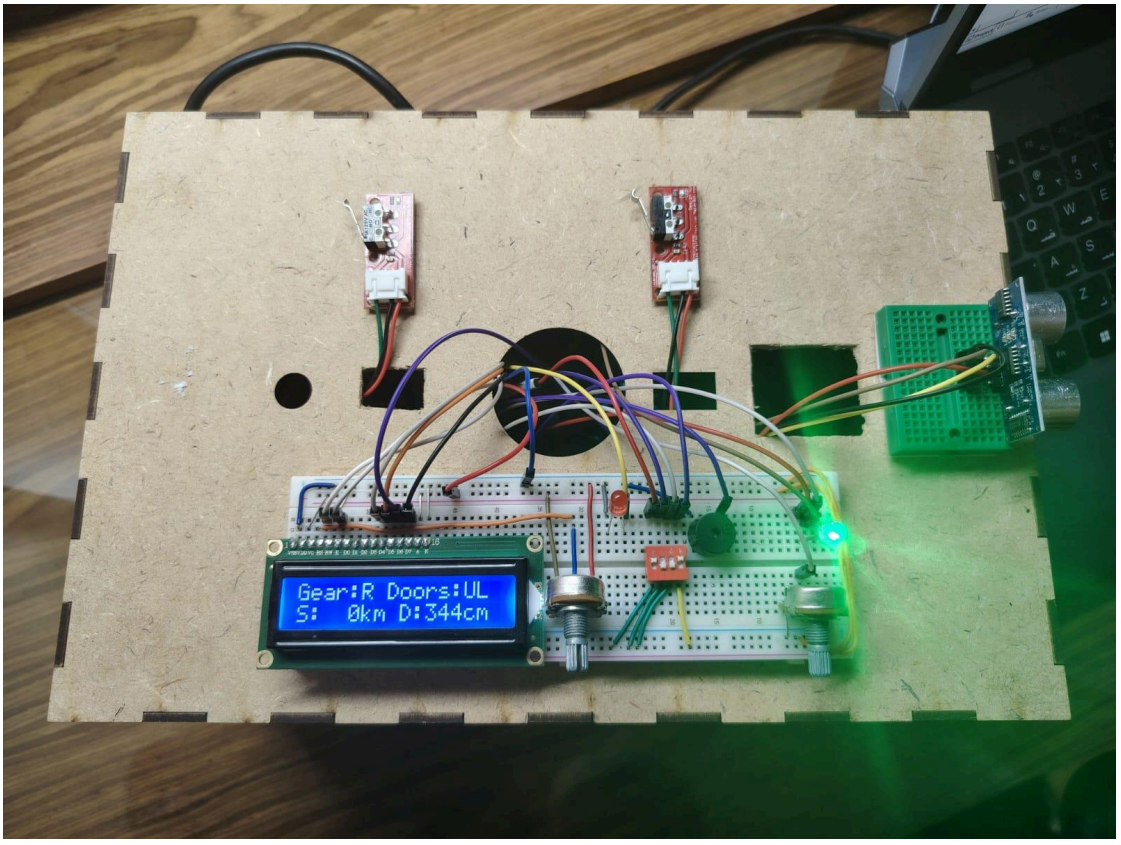
Manually Unlocking Doors while Speed above 10 km/h



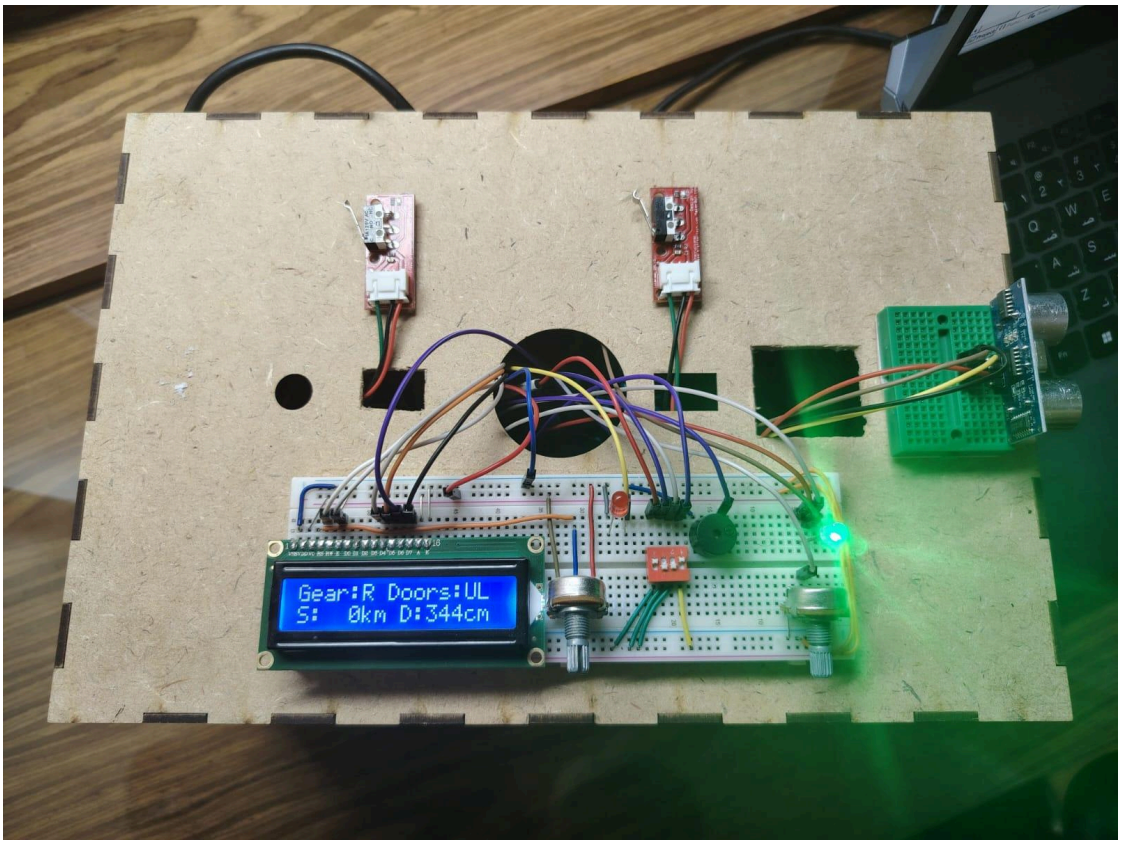
Manually Locking Doors while Speed above 10 km/h



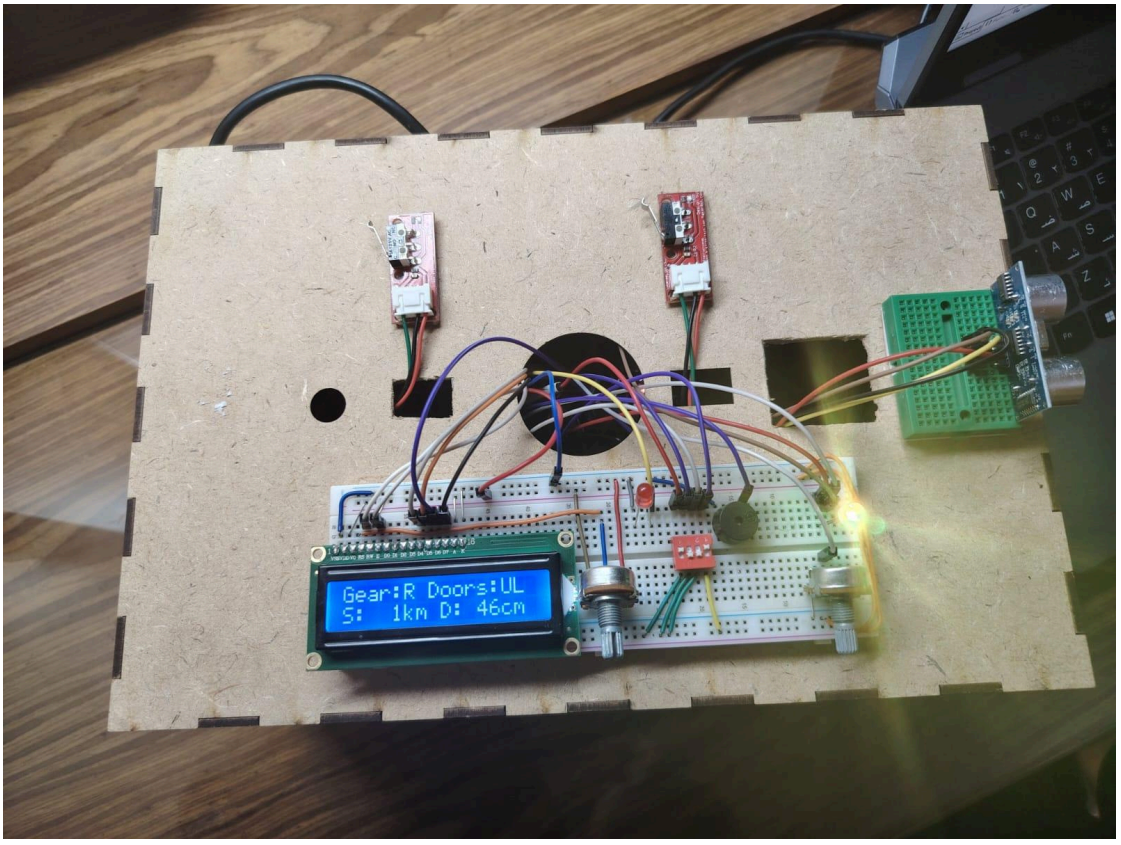
Gear: Reverse



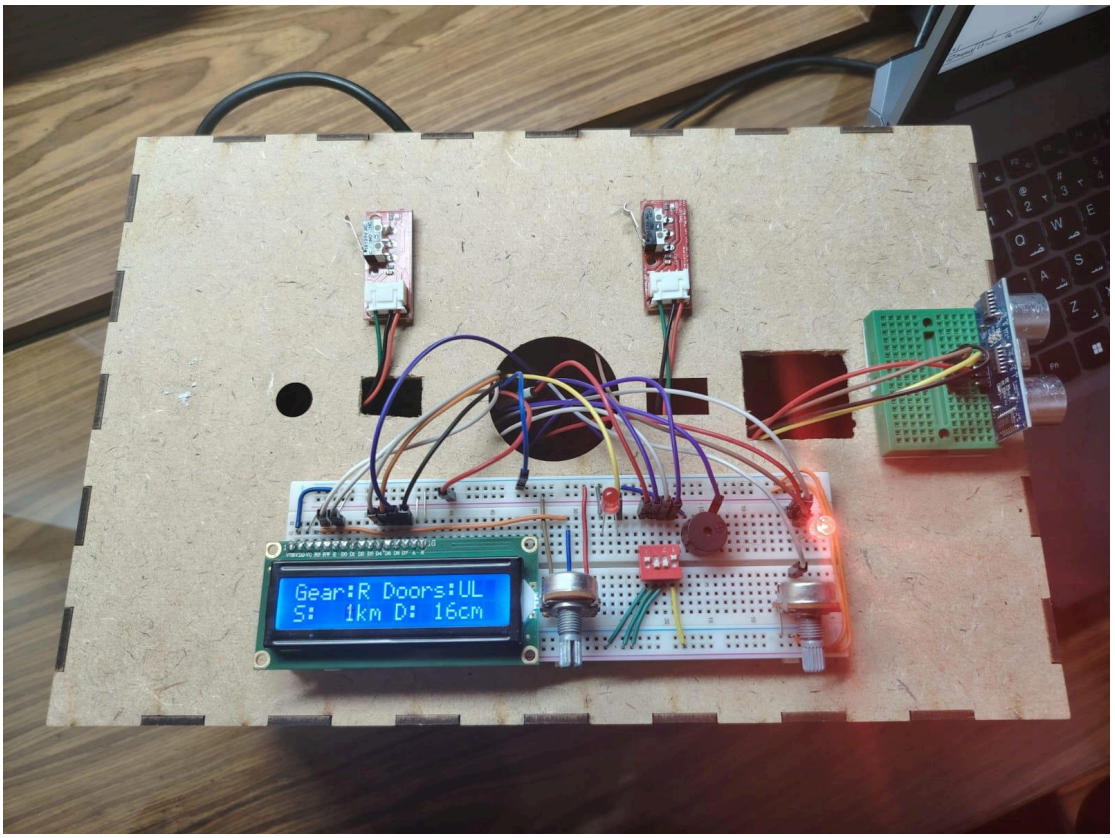
Reverse while Distance > 100 cm



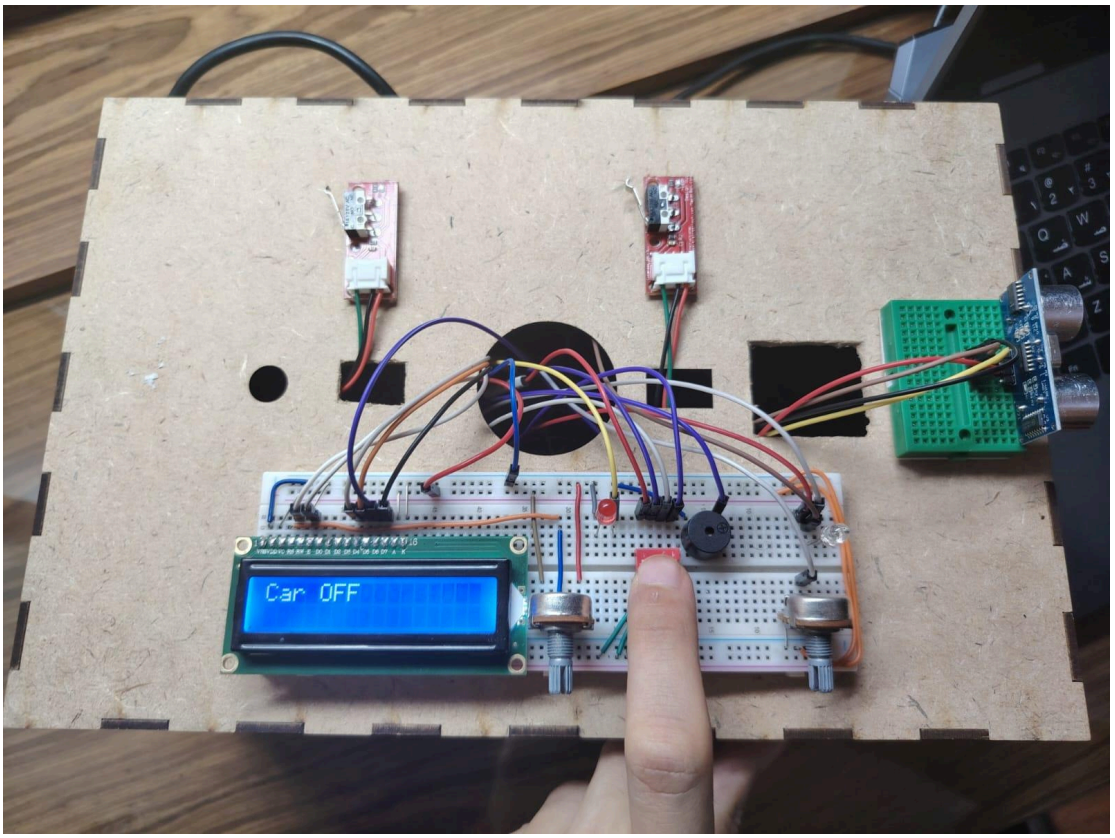
Reverse while Distance between (30 and 100 cm)



Reverse while Distance < 30 cm



Turning Ignition off



Challenges and Solutions

Challenge	Solution
1. Concurrent Access to Shared Resources Multiple tasks (doorTask, lcdUpdateTask) access the LCD, risking data corruption.	Used a **mutex (lcdMutex)** to ensure mutually exclusive access, preventing race conditions and maintaining display consistency.
2. Real-Time Responsiveness Tasks like door locking and rear obstacle detection require fast response to sensor changes.	Assigned higher priority to **doorTask** and used FreeRTOS scheduling to ensure time-sensitive tasks execute promptly.
3. Limited Queue Size and Data Freshness Speed and distance queues hold only one value, risking overwrites and lost data.	Purposefully limited queue size to **prioritize the most recent data**, ensuring the system always reacts to current conditions.

Results

The developed prototype system successfully demonstrates:

- Reliable auto-locking/unlocking behavior based on simulated vehicle speed and ignition.
- Functional override via manual lock/unlock buttons.
- Real-time obstacle detection and warning using the ultrasonic sensor.
- Multimodal driver alert system with LED, buzzer, and display.
- Efficient multitasking and resource management with FreeRTOS.

All features are modular, responsive, and maintain real-time performance.

Conclusion

The Automotive Smart Safety System project effectively showcases the power of real-time embedded systems in safety-critical applications. By integrating intelligent door locking and reverse obstacle detection into a unified system, it highlights key engineering practices including multitasking, hardware abstraction, real-time control, and inter-process communication. This project lays a strong foundation for further automotive innovations using embedded systems and RTOS-based development.