ADVANCED POWER WINDOW CONTROL SYSTEM

Course: CSE411 – Real-Time and Embedded Systems Design

Semester: Spring 2025

Submitted to:

Emeritus Professor

Sherif Ali Mohamed Hammad

Name	ID
Mohamed Maged Elsayed Ahmed El-Berry	**2401845**

Contents

1-Introduction:

The automotive industry continues to evolve with an increasing focus on automation, safety, and user convenience. One such feature that enhances both comfort and safety in modern vehicles is the power window system. This project presents the design and implementation of an Advanced Power Window Control System using the TM4C123GH6PM microcontroller and FreeRTOS. The system controls the front passenger-side window of a car, offering both manual and automatic operation, obstacle detection, and position tracking, all while ensuring safety and energy efficiency.

The project emphasizes real-time responsiveness and modular task management using FreeRTOS, allowing each subsystem—such as motor control, user input handling, emergency response, and display updates—to operate concurrently and efficiently. Safety features like IR-based obstacle detection, emergency motor reversal, and limit switch integration are implemented to prevent damage and injury during window operation.

Additionally, a lock mechanism allows the driver to disable the passenger-side control, and the system enters a low-power sleep mode when idle to conserve energy. The position of the window is tracked precisely using a quadrature encoder, and real-time status feedback is provided via an LCD display and a buzzer.

This report documents the system's architecture, real-time task scheduling, implementation challenges, and future upgrade potential, demonstrating a complete embedded control solution aligned with modern automotive design practices.

2-System Design and Architecture:

The system is designed to control a car's power window using a TM4C123GH6PM microcontroller, featuring manual and automatic operation from both the driver and passenger sides. Safety is prioritized through obstacle detection, limit switches, and an emergency reverse mechanism.

Key Hardware Components

- Microcontroller: TM4C123GH6PM
- Motor & H-Bridge: Controls window movement up/down.
  DC Motor Control: Motor direction control via H-Bridge → PF2 & PF3
  PF2: Motor Up / PF3: Motor Down
- Incremental Encoder (QEI): Tracks position in cm (0–40 cm)
  Quadrature Encoder: Position tracking → PD6 & PD7
  PD6: PHA0 / PD7: PHB0 (QEI0)
- Limit Switches: Detect upper and lower window limits.

  Upper Limit Switch Detect Fully Closed Window → PB0 (Input & Pullup)

  Down Limit Switch Detect Fully Opened Window → PB1 (Input & Pullup)

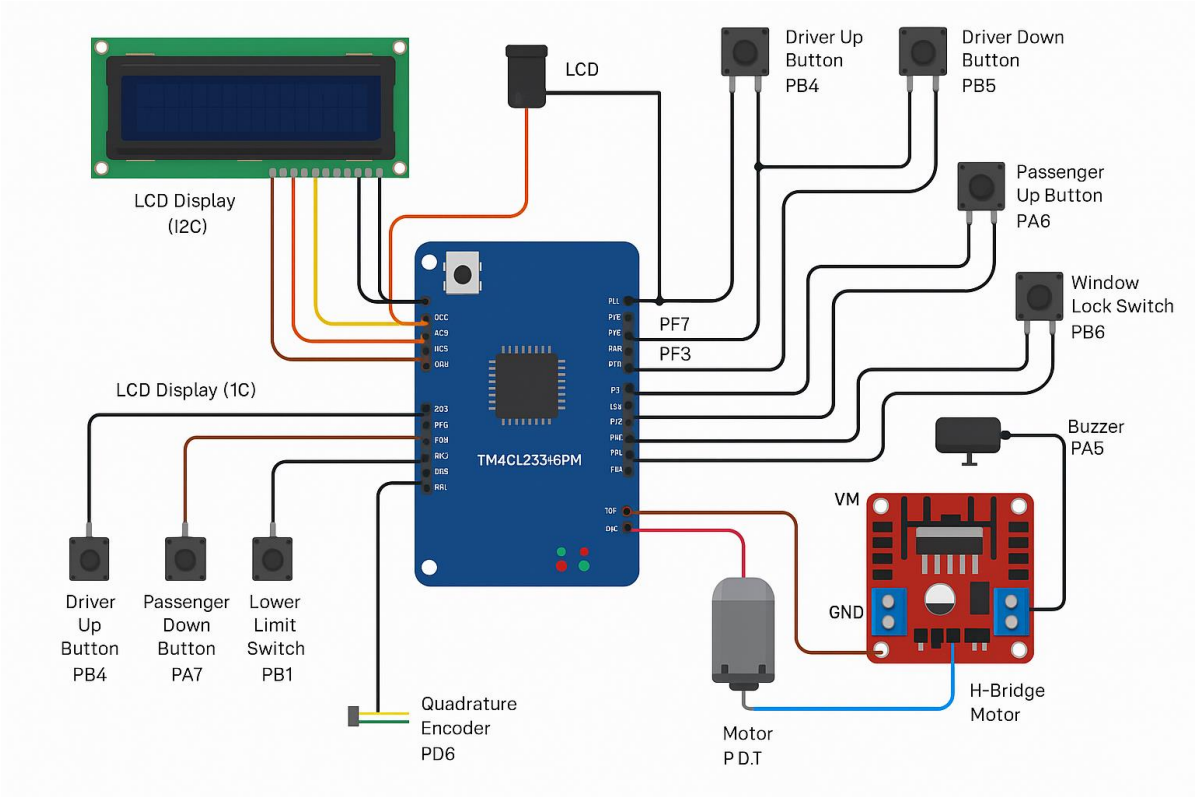
- IR Sensor: Detects obstacles during closing
  Detect Obstacles during Closing → PB7 (Input & Pullup)
- LCD (I2C): Displays position and status messages
  Displays status and motor positioning → PB2(SCL), PB3(SDA) I2C (Peripherals)
- Buzzer: Alerts during emergency stops
  Audio Alert on Emergency Stop → PA5
- Push Buttons: Control from both driver and passenger
  Driver's Up →(PB4) (Input & Pullup)
  Driver's Down →(PB5) (Input & Pullup)

Passenger Up →(PA6) (Input & Pullup)

Passenger Down →(PA7) (Input & Pullup)

- Lock Switch: Driver disables/enables passenger controls

  Disable Passenger's Control→(PB6) (Input & Pullup)



Functional Overview

- Manual Mode: Window moves while the button is held.
- Auto Mode: One short press moves window completely up/down.
- Emergency Stop: If an obstacle is detected while closing, the system stops and reverses the motor briefly while buzzer ringing.
- Lock Feature: Prevents passenger-side control if enabled.
- Idle Sleep Mode: Microcontroller enters low-power state when no tasks are active.

3-FreeRTOS Implementation Details

Used **FreeRTOS** to divide system functionality into modular and prioritized tasks, ensuring real-time behavior, concurrency, and responsiveness.

Task Summary:

Task Name	Function	Priority
EmergencyTask	Monitors IR sensor for obstacles and triggers emergency stop	3 (highest)
DriverControlTask	Handles driver's window control (manual and auto modes)	2
PassengerControlTask	Handles passenger control if not locked	1

Synchronization Tools:

Semaphores (Mutual Exclusion): We use a binary semaphore (xMotorSemaphore) to protect access to the shared motor control lines. Since both driver and passenger tasks can issue motor commands (e.g., open or close the window), a race condition could occur if these operations are not mutually exclusive. The semaphore ensures that only one task can access the motor at a time, preventing hardware conflicts or erratic behavior.

```
void Motor_Up(void) {
    if (xSemaphoreTake(xMotorSemaphore, portMAX_DELAY) == pdTRUE) {
        LCD_ClearLine(0);
        LCD_Print("Closing..   ");
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2 | GPIO_PIN_3, GPIO_PIN_2);
        xSemaphoreGive(xMotorSemaphore);
    }
}
```

This ensures that the motor driver signals are written atomically, and no other task can interrupt or interfere during this operation.

Idle Hook: Used for entering low-power mode via WFI instruction

Delays: vTaskDelay() and vTaskDelayUntil() manage timing and polling efficiently

Queues (Inter-Task Communication): We implement a message queue (xEmergencyQueue) to notify the EmergencyTask of critical events, such as an obstacle detected by the IR sensor. This decouples the detection logic from the response logic, improving modularity and reliability.

When an obstacle is detected in the driver or passenger task:

```
if (IsObstacleDetected()) {
    uint8_t event = 1;
    xQueueSend(xEmergencyQueue, &event, 0);
    break;
```

The Emergency Task waits for messages from the queue and reacts accordingly:

```
for (;;) {
    if (xQueueReceive(xEmergencyQueue, &emergencyEvent, 0) == pdPASS) {
        if (emergencyEvent == 1) { // Obstacle detected
            emergencyStop = true;

            Motor_Stop();
            Buzzer On();
```

This design ensures that emergency handling is performed asynchronously, without blocking the control tasks. It also improves scalability, allowing future extensions (e.g., different event types) with minimal modification.

Scheduler:

- Round-robin style scheduling handled by FreeRTOS, respecting priority levels.
- Preemptive task switching ensures emergency response takes precedence.

4-Challenges and Solutions:

Challenge	Solution
Ensuring accurate position updates	Integrated QEI module with proper initialization and scaled to cm units.
Differentiating Auto vs Manual	Used press duration with xTaskGetTickCount() to detect short/long press
Avoiding race conditions on motor	Used binary semaphore (xMotorSemaphore) to guard motor access

Emergency reverse timing	Used vTaskDelay() inside a timed loop with condition checks
Power saving in idle	Enabled FreeRTOS Idle Hook and implemented __asm("WFI") sleep logic
Debouncing and responsiveness	Delays were tuned to 50–100 ms for smooth and responsive controls

5-Future Improvements

Though the system is fully functional and meets all the project requirements, several enhancements could be considered:

- **Deep Sleep Mode**: Reduce power further by disabling peripheral clocks in idle state.
- **Bluetooth/Mobile Control**: Add UART/BLE interface to control window remotely.
- **Partial Memory Save**: Save last known window position to EEPROM or flash.
- **GUI Dashboard**: Build a PC or mobile GUI to monitor real-time position and logs.
- **Motor Current Monitoring**: Use ADC to monitor motor current for jam detection.
- **Unit Testing**: Integrate a framework for embedded unit tests for better maintainability.

6-Google Drive Link:

https://drive.google.com/drive/folders/1w9UgQPuosjZ-6z1q6mkd8jeKcKtCS7uZ?usp=sharing