Power window control system using Tiva C running FreeRTOS

Team :

- **Eslam Sayed Rady Mtrawy**(2301468)

- **Doha Gamal Amin** (2301579)

- **Anas** (2201822)

- **Joseph Hanna** (2201657)

Under supervision of

–Dr: Sherif Hammad

–Eng: Hesham Salah

Contents

1 Introduction

In this project, our team has developed a Power Window Control System utilizing the Tiva C Series TM4C123GH6PM microcontroller and FreeRTOS.
The project is aimed at demonstrating the capabilities of a real-time operating system (RTOS) in handling multiple tasks such as sensor input processing, motor control, and user interface management within an embedded system.
This project not only serves as an excellent practical application of embedded system programming but also underscores the importance of real-time task management, synchronization, and effective use of system resources in automotive safety applications. By integrating components such as DC motors, push buttons, and limit switches, the system mimics the operation of a full-scale power window, providing insights into the complexities of automotive control systems.

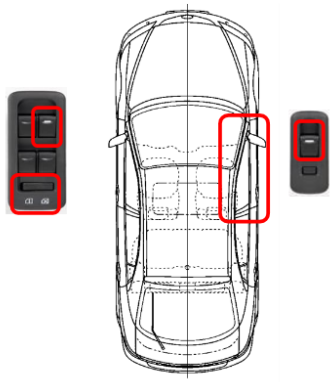


Figure 1 Power Window Control System

2 System Requirements and Features

The design and implementation of the Power Window Control System were guided by the following detailed requirements and features:

1. Dual Control Panels:

   Functionality: Both the driver and front passenger have control panels capable of independently controlling the window.Interactivity: Controls include up, down, and stop functions for manual operation, with override functions for safety.

2. FreeRTOS Implementation:

   Task Management: Multiple tasks run concurrently, including reading sensor data, controlling motor operations, and handling user interface inputs.Synchronization: Utilizes semaphores and mutexes to manage resource sharing and ensure the system operates without deadlock or resource starvation.

3. Safety Mechanisms:

   Limit Switches: Positioned at both the top and bottom of the window trajectory to automatically stop the motor when the window reaches its full open or closed position. Obstacle Detection: Simulated using a push button that, when pressed,

triggers the system to stop the window movement, mimicking the detection of an object obstructing the window path.

4. Functional Modes:

Manual Mode: Users can control the window manually by continuously pressing the control buttons Automatic Mode: Features a one-touch operation where a single press fully opens or closes the window, demonstrating the ease of use and advanced control Lock Function:
Allows the driver to lock the passenger window controls, preventing operation from the passenger side, enhancing the control and security features.

3 System Architecture

3.1.1 Hardware Components

The Power Window Control System is built around the Tiva C Series TM4C123GH6PM microcontroller, chosen for its robust performance and
suitability for real-time applications.
The hardware setup includes:

- **Tiva C Microcontroller**:
  Acts as the central control unit, managing inputs from sensors and user commands, and controlling the window motor.DC Motor: Drives the window mechanism, simulating the opening and closing of a car window.
- **Limit Switches**:
  Two limit switches are installed at the top and bottom of the window frame to detect the window's maximum open and closed positions, ensuring the motor stops at correct points.
- **Push Buttons**:
  Used for manual control of the window and simulating obstacle detection. Separate buttons are provided on both the driver and passenger control panels.

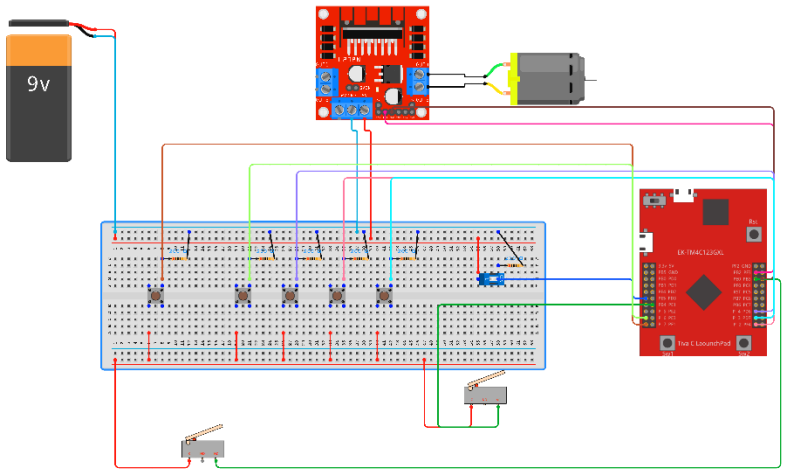


Figure 2circuit_wiring_diagram

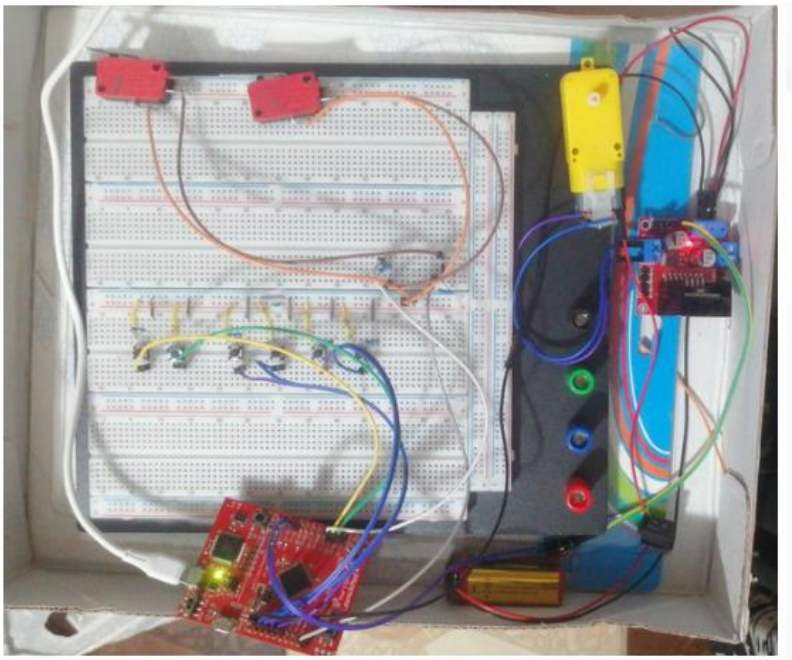


Figure 3 Hardware Components

3.1.2 Software Components

The software of the Power Window Control System is structured around FreeRTOS, which provides the framework for multitasking and task management. Key software features include:

- **Task Scheduling**:
  FreeRTOS schedules and manages all tasks, ensuring that higher-priority tasks (like obstacle detection) preempt lower-priority tasks when necessary.
- **Semaphore and Mutex Handling**:
  Used to handle resource sharing between tasks, preventing race conditions and ensuring system reliability.
- **Interrupt Service Routines (ISRs)**:
  Configured to handle inputs from the limit switches and obstacle detection button, providing immediate response to critical events.

3.1.3 Integration and Communication

The system architecture is designed for seamless integration of hardware and software components. Tasks communicate via queues, sending messages between the control interface and motor control tasks to coordinate the window's movements. Mutexes and semaphores ensure that access to shared resources like the motor control interface is properly synchronized among tasks.

4 Implementation Details

4.1.1 Code Structure and Task Implementation

The software for the Power Window Control System is structured around several key tasks, each designed to handle specific aspects of the system's functionality:

- Driver and Passenger Control Tasks: These tasks handle inputs from the driver and passenger control panels. They read the state of push buttons to either initiate or stop the movement of the window.
- Window Motor Control Task: This task controls the DC motor based on commands received from the control tasks. It integrates feedback from the limit switches to stop the motor at the correct positions and handles the automatic movement logic.
- Obstacle Detection Task: Activated by a push button, this task takes the highest priority when triggered, immediately stopping the motor and initiating a reverse movement to free any detected obstacles.

4.1.2 Resource Management

- Semaphores and Mutexes: Critical resources, such as the motor control interface, are protected using mutexes to avoid simultaneous access conflicts. Semaphores are used to signal between tasks, particularly from ISR routines to tasks that manage motor control and obstacle handling.
- Queues: Messages, such as commands and status updates, are passed between tasks using queues, ensuring that communication is organized and non-blocking. Interrupt Handling

- **Limit Switches and Obstacle Detection:**

Interrupts are configured to respond to inputs from the limit switches and the obstacle detection button. The ISRs are designed to quickly handle these inputs, either by stopping the motor or by notifying the relevant tasks to take immediate action.

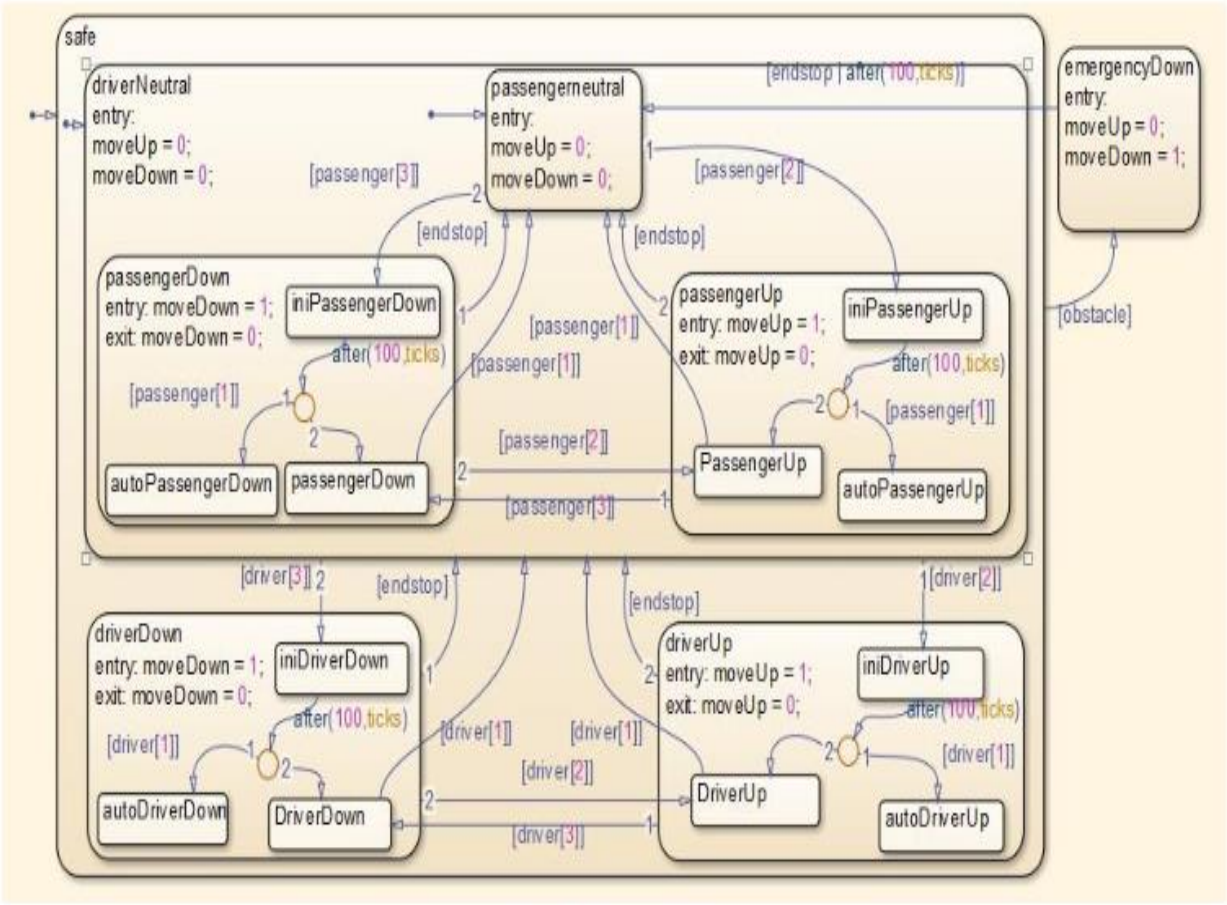
- Finite State Machine (FSM)

  Implementation: The system logic is governed by a Finite State Machine, which is crucial for managing the states of the window (e.g.,opening, closing, stopped). This FSM ensures that transitions between states are smooth and logical, based on user inputs and sensor outputs.

  - Tools: The FSM is visualized and tested using tools like MATLAB Simulink before implementation to ensure accuracy and reliability.
- Debugging and Validation

The system underwent rigorous testing, using both simulated inputs and real-world trials to validate each component's functionality and the system's overall reliability. Corner cases, such as simultaneous button presses and rapid command changes, were particularly focused to ensure the system's robustness.

5 Finite State Machine and Flowcharts



Simulink State Machine

Figure 4 Simulink state machine.

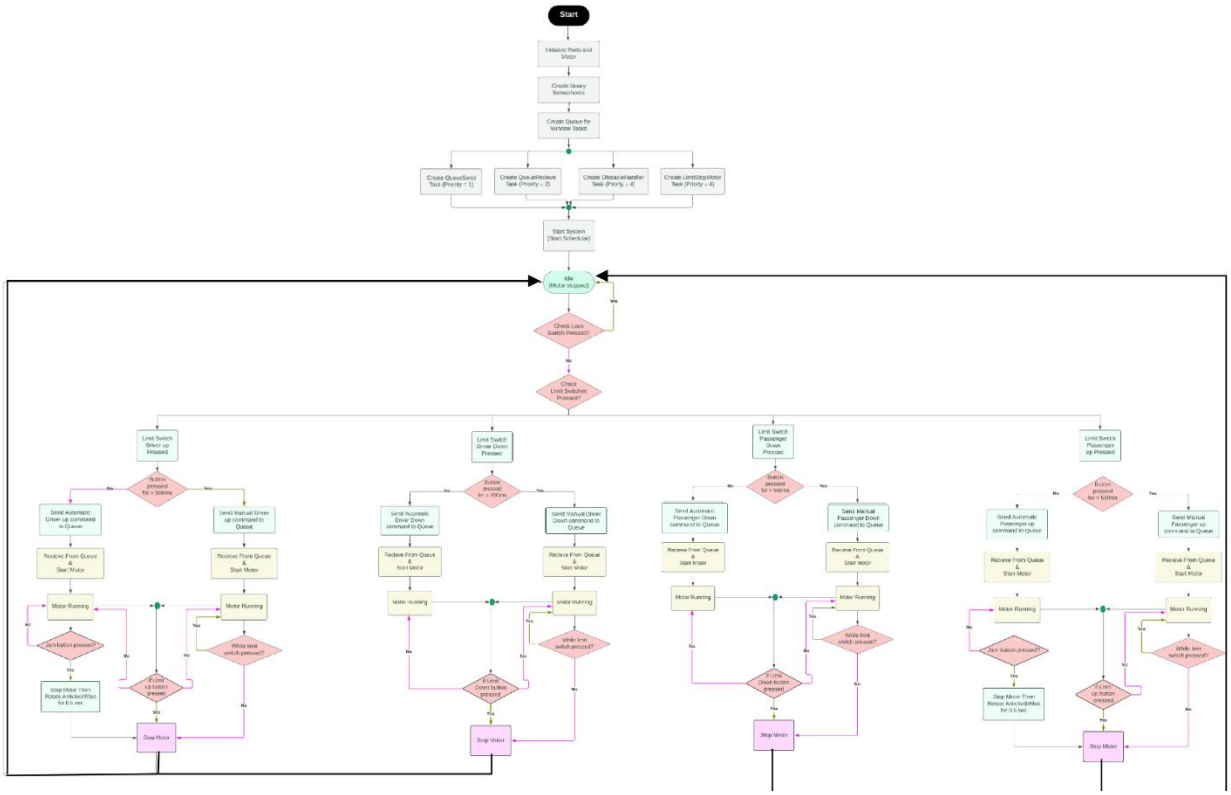


Figure 5flowchart

- **References**

□□ 1- Tiva™ C Series TM4C123GH6PM Microcontroller Data Sheet. Texas Instruments, 2021.

□2- FreeRTOS User Guide.