Power Window Control System with Safety Features

Real-Time Embedded System – CSE411
SPRING 2024

Presented to:

Dr. Sherif Ali Mohamed Hammad
Eng. Hesham Salah

Team 34

Mohamed Sohiel Sayed	**20P3576**
Ahmed Abu Bakr Hafez	**20P4588**
Omar Magdy Mostafa	**1900884**
Ahmed Mohamed Ahmed	**20P2629**

Table of Contents

LIST OF FIGURES

Drive Link:

Video + code files + Report

System Design

Introduction:

This project describes a car window control system designed using Texas Instruments (TI) Tiva microcontrollers and programmed in C with FreeRTOS, a real-time operating system.

Key Features:

- Controls the front passenger window using switches on both the driver and passenger sides.
- Uses FreeRTOS for multitasking and efficient operation.
- Includes limit switches to prevent the window from going beyond its safe range.
- Provides obstacle detection using a push button to simulate jamming.

Basic Functions:

- Manual window open/close with switch press and hold.
- One-touch window operation for full open/close with a short switch press.
- Window lock to disable passenger window control from the driver's side.
- Jam protection that automatically reverses the window slightly if it encounters resistance during one-touch closing.
- This system offers convenient window control with safety measures to prevent damage from overtravel or obstructions.

.

System Layout:

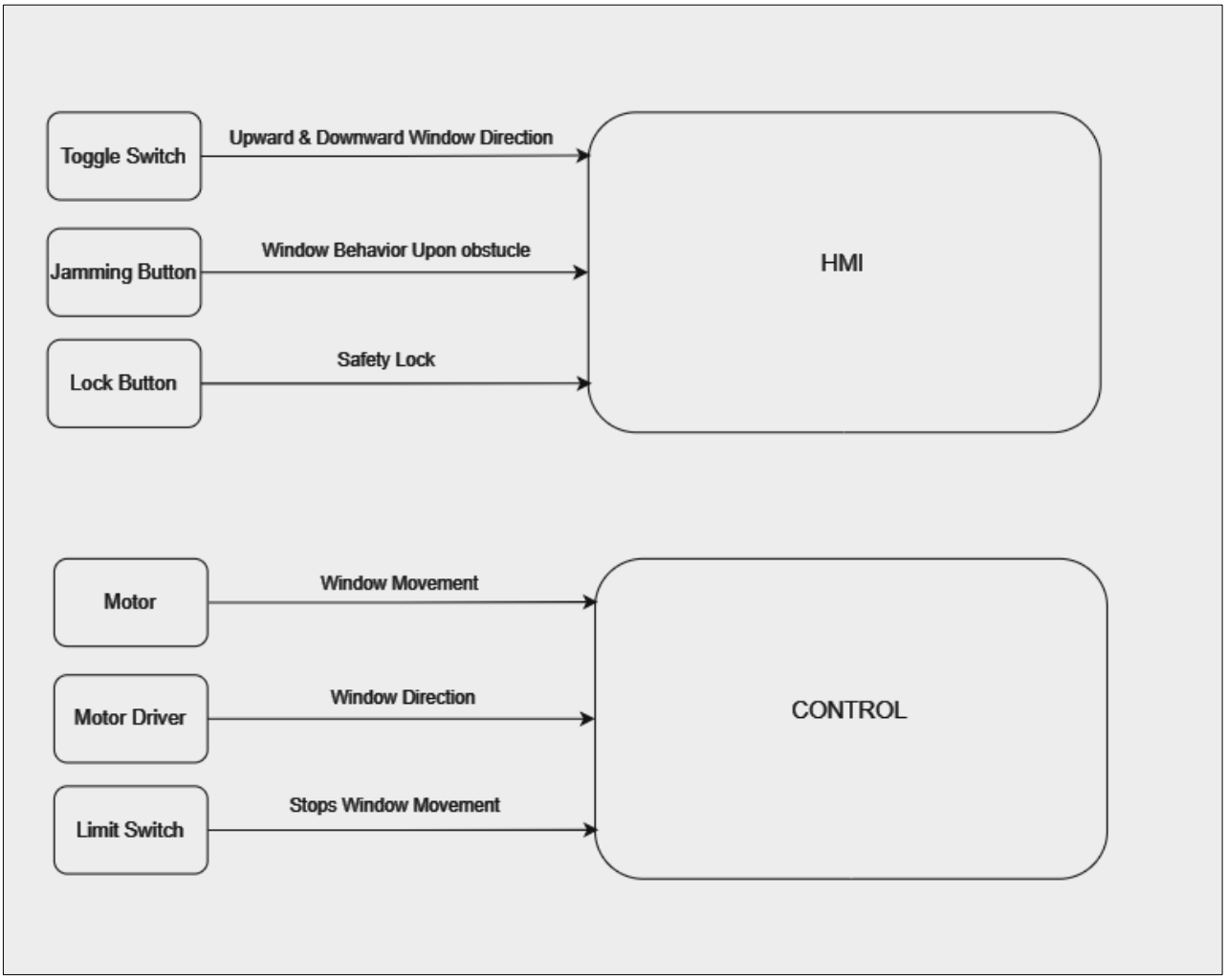


Figure 1. System Layout Diagram

The system comprises two main modules:

1. **Control Unit:** This unit is the brain of the power window system. It utilizes a Tiva microcontroller running FreeRTOS to manage window movement and safety features. This includes:
    - Processing user inputs from switches on both driver and passenger sides.
    - Controlling the window motor based on manual or one-touch open/close commands.
    - Monitoring limit switches to prevent the window from over-traveling.
    - Detecting obstacles using a push button for jam protection.
    - Implementing window lock functionality to disable passenger window control from the driver's side.

2. **Human Machine Interface (HMI):** This module acts as the user interface, providing physical switches for passengers and the driver to interact with the system. These switches allow users to:
    - Manually open or close the window.
    - Initiate one-touch open or close functionality.
    - Activate the window lock.
    - Simulate jamming by pressing a button (replacing a current stall sensor).

Component Diagram:

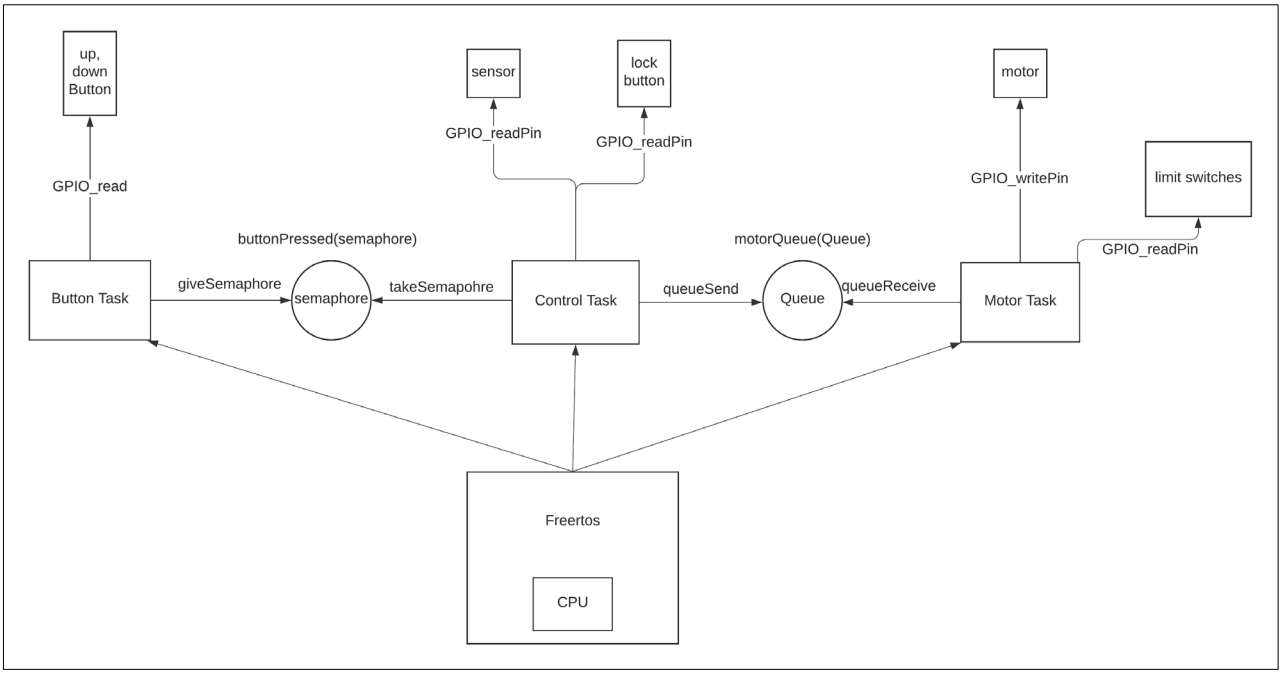


Figure 2. Component Diagram

State diagram:

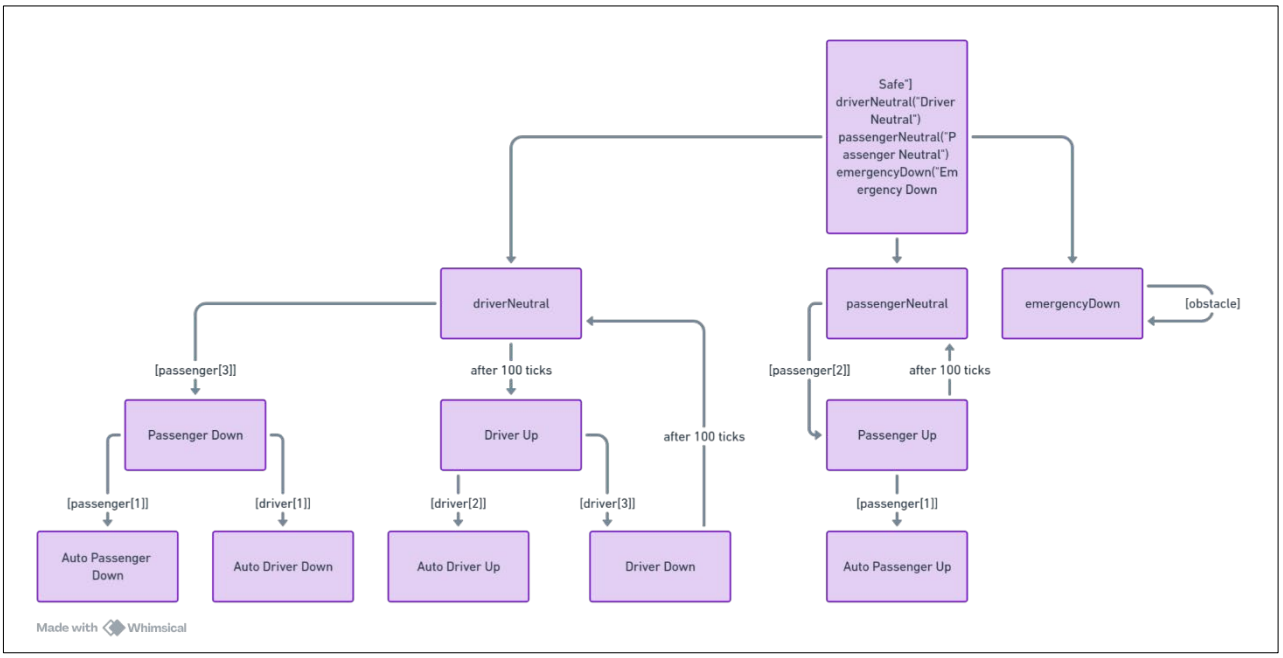


Figure 3. State Diagram

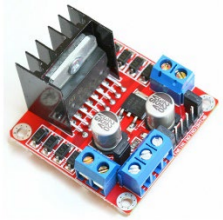
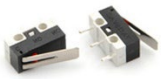
FreeRTOS APIs:

1. **xTaskCreate**: used to create tasks. create button task, control task and motor task for each seat.
2. **vTaskStartScheduler**: This function starts the FreeRTOS scheduler.
3. **vTaskSuspendAll**: suspends all tasks. It stops the scheduler from executing any further tasks. it's called after starting the scheduler, but since it's followed by an infinite loop (for (;;);), the code effectively halts execution at this point
4. **xSemaphoreCreateBinary**: create a binary semaphore. Semaphores are used for task synchronization and communication.
5. **xQueueCreate**: creates a queue. Queues are used for inter-task communication; queues are used in this system for the motor to receive commands through them.
6. **xSemaphoreGive**: This function is used to release (or 'give') a semaphore.
7. **xSemaphoreTake**: This function is used to take (or 'acquire') a semaphore. It blocks the execution of the task until the semaphore becomes available.
8. **xQueueSend**: This function is used to send an item to the end of a queue. it's used to send motor control states to motor control task.
9. **vTaskSuspend/vTaskResume**: These functions suspend and resume the execution of a task. They are used to suspend and resume the button task when a motor is jammed.
10. **xQueueReceive**: This function is used to receive an item from the front of a queue. It blocks the task until an item becomes available in the queue.
11. **vTaskDelay**: blocks the execution of the current task for a specified period of time.
12. **xTaskGetTickCount**: returns the current tick count of the FreeRTOS scheduler.
13. **vTaskDelete**: This function is used to delete the calling task. It’s called at the end of the task function, indicating that the task will delete itself after execution.

14.

Hardware Design

List of Components:

	Mini DC Gearbox Motors Pair (2 Motors) With black shaft	Responsible for window movement
	Toggle switch 3 pin	Responsible for window controls (up or down)
	L298 Module Red Board	Manges power and control of the motor
	Eco SMPS Output +36Vdc/10A Input 220Vac With Cooling Fan	Supplies 12V to the motor
	Micro Switch – MS1	prevent the window from going beyond its safe range
	Push Button	Push button for safety lock

Circuit wiring:

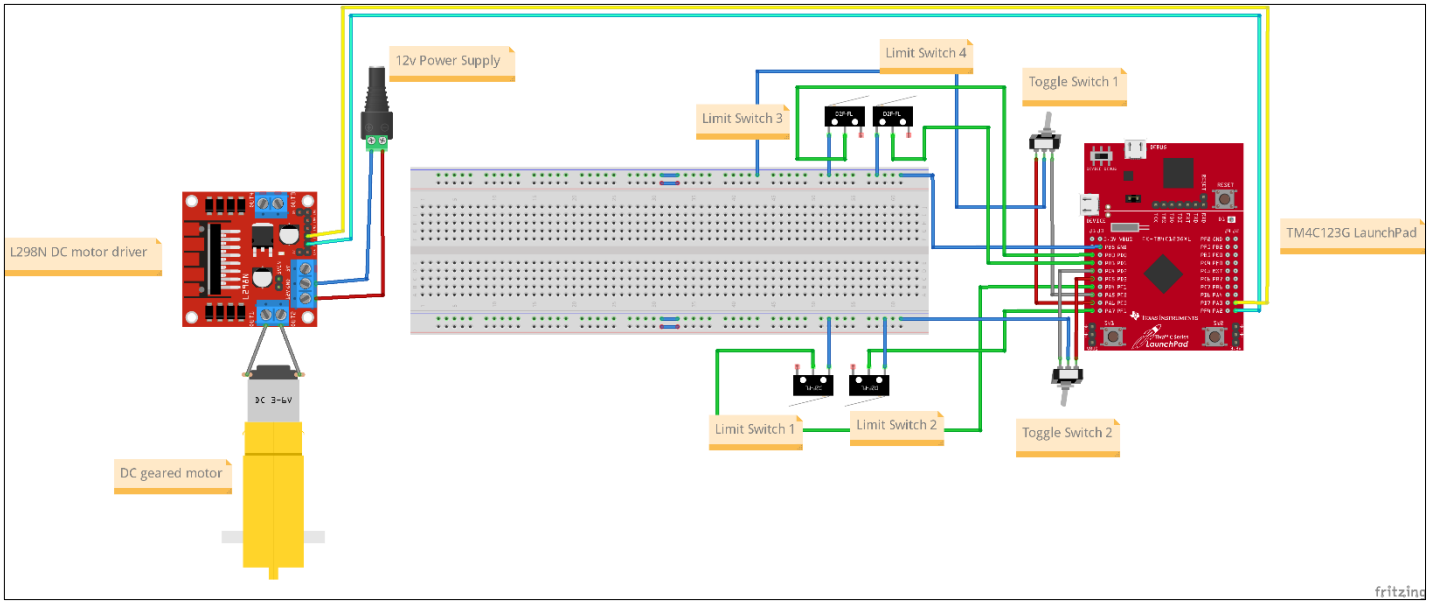


Figure 4. Circuit Wiring

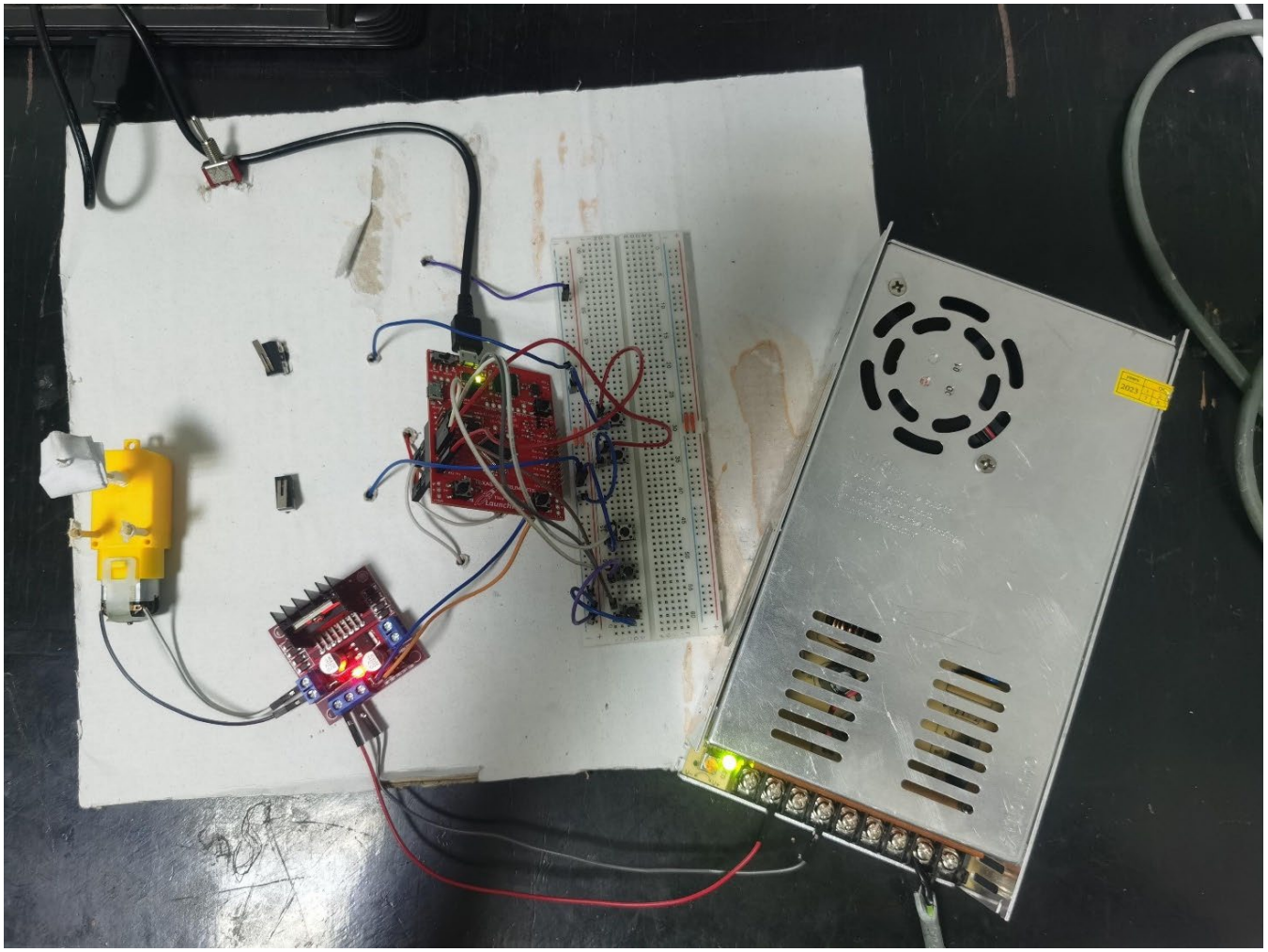


Figure 5. Hardware Layout