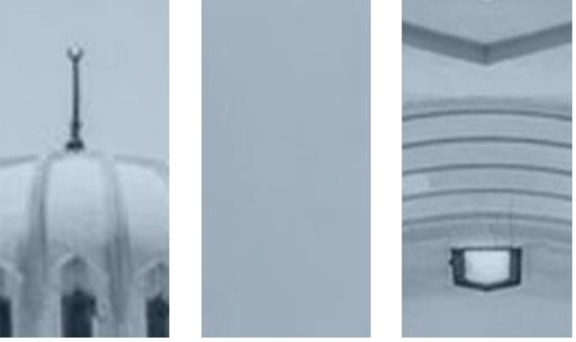

Team 17

Power window control system Project Report

Program: Computer Engineering and Software Systems

Course Code: CSE411

Course Name: Real Time and Embedded Systems Design

Submitted to

Dr. Sherif Hammad

Eng. Hesham Salah

Eng. Ayman Baharia

Ain Shams University

Faculty of Engineering

Spring Semester – 2024

Students Personal Information

NAME	ID
Nadine Hisham	20P9880
Youssef Emad	20P3844
Mohamed Amr	20P3485
Heba Hesham	20P8736
Mohamed Ibrahim	20P8449

Project Link

Table of Contents

Table of Figures

1. Introduction

The power window control system is a crucial component of modern automotive design, enhancing convenience and safety for vehicle occupants. This project focuses on developing a power window control system for the front passenger door of a vehicle using the Tiva C microcontroller running FreeRTOS. The system will include both passenger and driver control panels, ensuring ease of use and safety through features like manual and automatic window operation, window lock functionality, and jam protection.

1.1. Technology Overview

The power window control system will leverage the Tiva C microcontroller and the FreeRTOS real-time operating system to manage the various functions of the window. This includes manual and one-touch automatic operations, as well as safety features such as window lock and jam protection. The system will use limit switches to prevent the window motor from overdriving, and a push button will simulate obstacle detection for jam protection.

1.2. Purpose of this Project

The purpose of this project is to design and implement a reliable and user-friendly power window control system that integrates seamlessly with the existing vehicle infrastructure. By using FreeRTOS, the system will demonstrate the capabilities of real-time operating systems in automotive applications, providing deterministic control over the window functions. This project aims to enhance the understanding of RTOS-based embedded systems and their practical applications in automotive electronics.

1.3. Project Scope

1. **Front Passenger Door Window Control**

   Implement control mechanisms for the front passenger door window from both the passenger and driver panels, allowing both occupants to operate the window.

2. **FreeRTOS Integration**

   Ensure the entire control system is built upon the FreeRTOS platform, leveraging its real-time capabilities for task management and system responsiveness.

3. **Limit Switches**

   Incorporate two limit switches to define the upper and lower bounds of the window's movement, preventing motor overdrive and potential mechanical damage.

4. **Obstacle Detection**

   Implement a basic obstacle detection mechanism using a push button to simulate window jamming. This feature will enhance safety by stopping and reversing the window if an obstruction is detected.

2. System Overview

2.1. Board

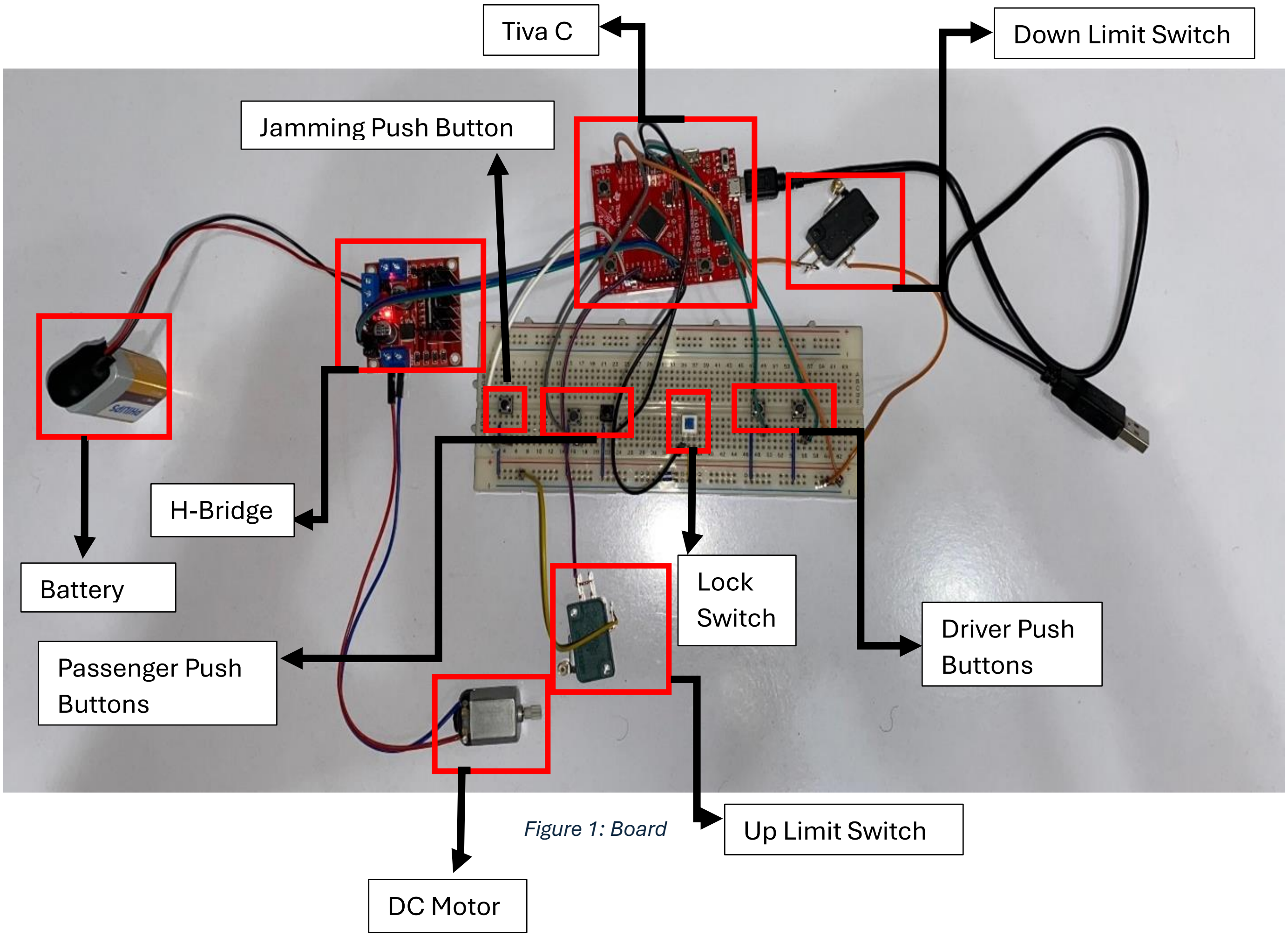

Figure 1: Board

3. List of components

3.1. Components

- Tiva C (TM4C123GH6PM)

Figure 2: Tiva C (TM4C123GH6PM)

- 5 Push Buttons

Figure 3: Push Button

- 2 Limit Switches

Figure 4 : Limit Switch

- DC Motor

Figure 5: DC Motor

- On/Off Switch (Lock Button)

Figure 6: ON/OFF Switch

- H-Bridge

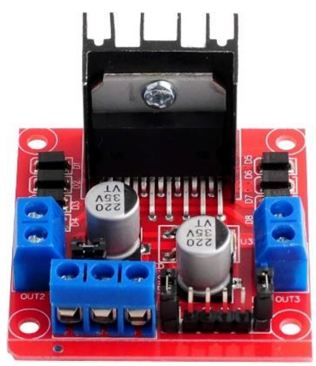

Figure 7: H-Bridge

3.2. Operation

- Implementation of front passenger door window with both passenger and driver control panels (2 pushbuttons for each panel).
- Implementation of 2 limit switches to limit the window motor from top and bottom limits of the window.
- Obstacle detection implementation is done using a push button to indicate jamming.
- Lock Mode is implemented using switch button.

4. Contribution of Group Members

" Nadine Hisham"

- **Role**: Driver & Passenger Tasks
- **Contributions:** Nadine was responsible for developing the tasks that handle the driver and passenger inputs. She worked on ensuring that the system correctly interprets button presses to move the motor in the appropriate direction, handling both automatic and manual modes based on the duration of the button press.

"Heba Hesham"

- **Role:** Interrupts + Tasks
- **Contributions:** Heba implemented the interrupt service routines that handle edge-triggered events from the GPIO ports. She also contributed to task synchronization, ensuring that tasks are properly managed and triggered by these interrupts without conflicts.

"Youssef Emad"

- **Role:** Driver & Passenger Tasks
- **Contributions:** Youssef collaborated closely with Nadine in refining the logic of the driver and passenger tasks. He focused on the manual control logic, adding conditions to manage state transitions when buttons are held down for longer durations.

"Mohamed Amr"

- **Role:** Circuit Topology + Lock Task
- **Contributions:** Mohamed designed the circuit topology to ensure that all components are correctly interfaced with the MCU. He also developed the lock task, which manages system security by allowing tasks to change priority based on certain inputs, thereby enhancing the control logic.

"Mohamed Ibrahim"

- **Role:** Circuit Topology + Lock Task
- **Contributions:** Mohamed designed the circuit topology to ensure that all components are correctly interfaced with the MCU. He also developed the lock task, which manages system security by allowing tasks to change priority based on certain inputs, thereby enhancing the control logic.

5. Circuits Wiring

5.1. Simulation

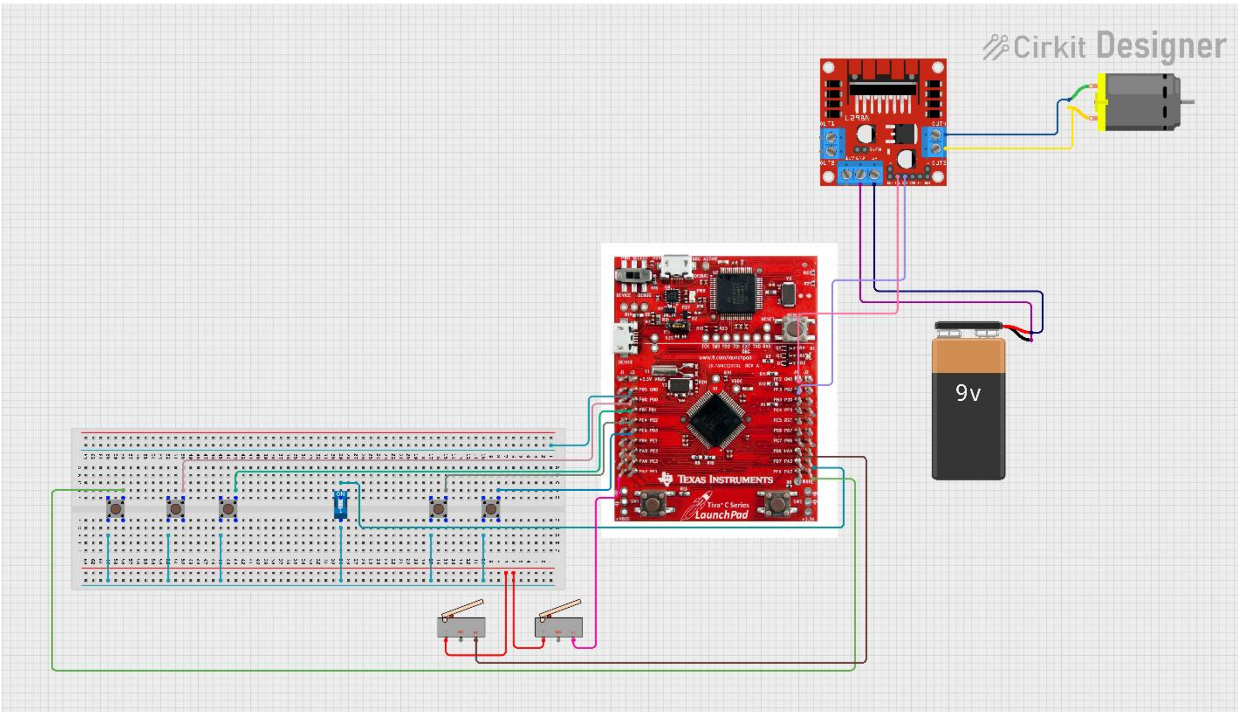

Figure 8: Circuit Simulation

6. Flowchart of the main flow of the program

6.1. Passenger Flow Chart

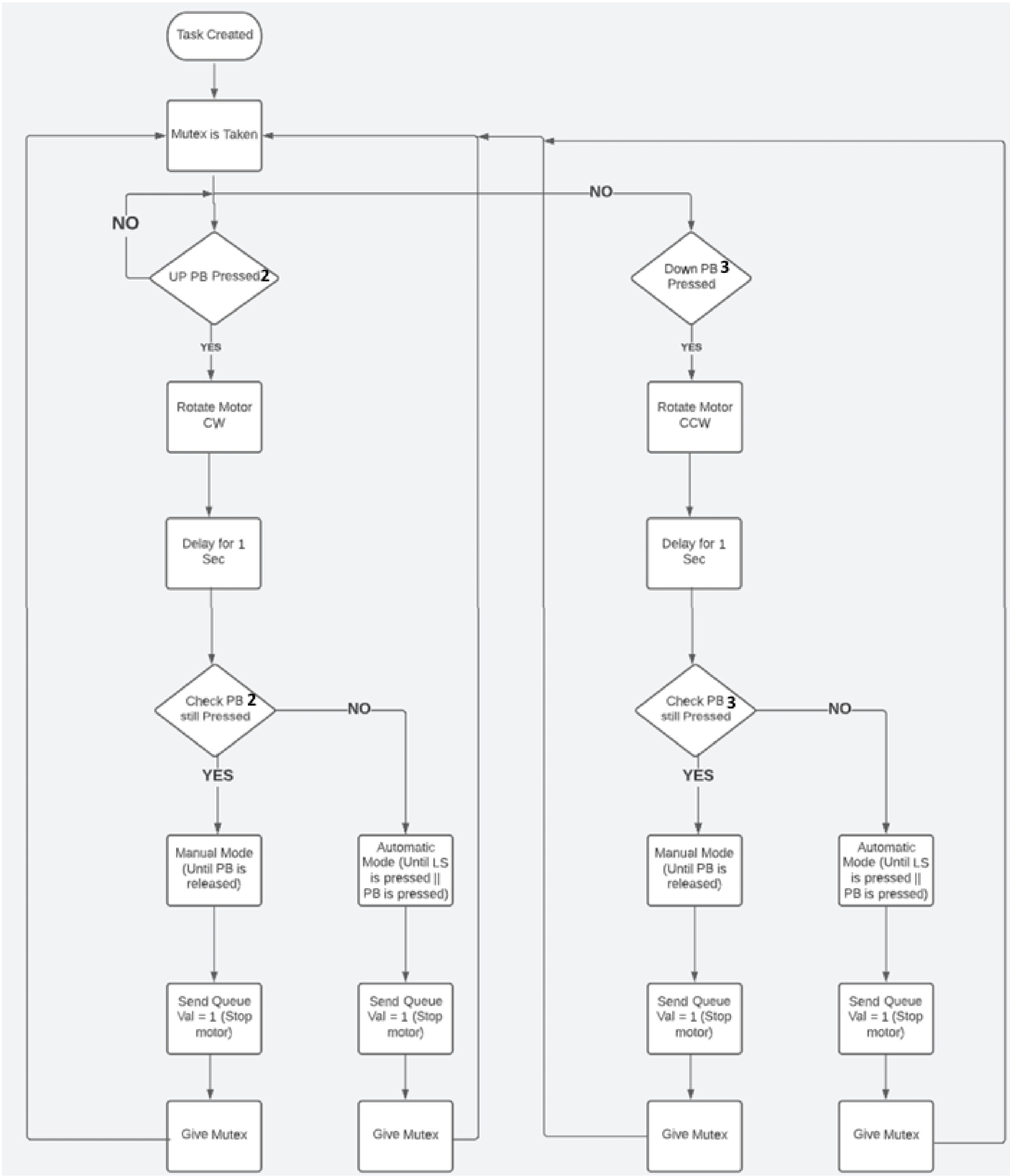

Figure 9: Passenger Flow Chart

6.2. Driver Flow Chart

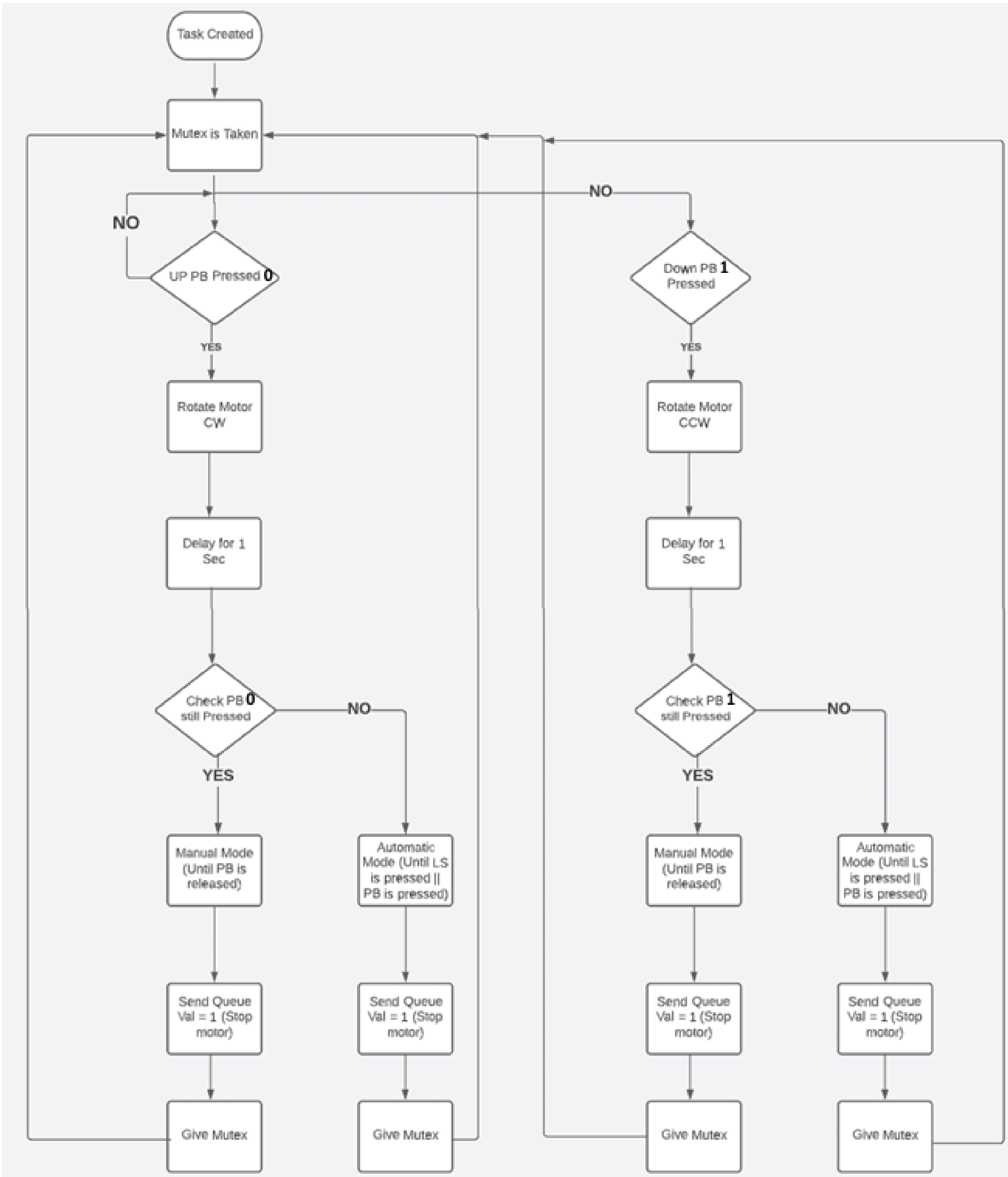

Figure 10: Driver Flow Chart

6.3. Queue

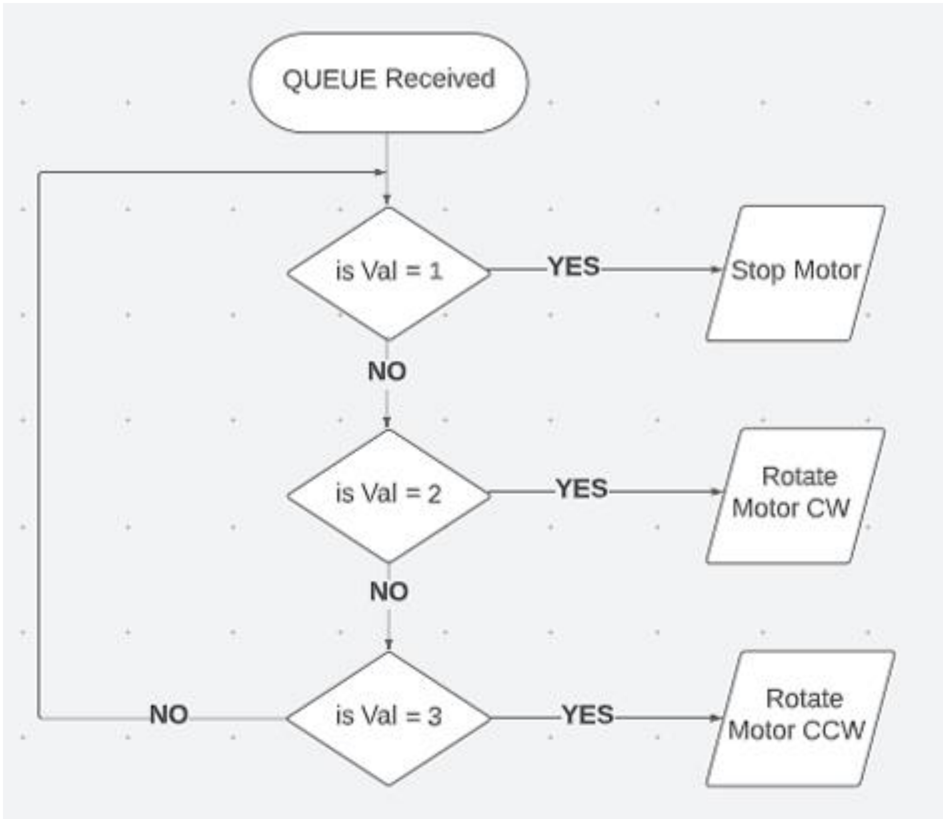

Figure 11: Queue Flow Chart

6.4. Jamming

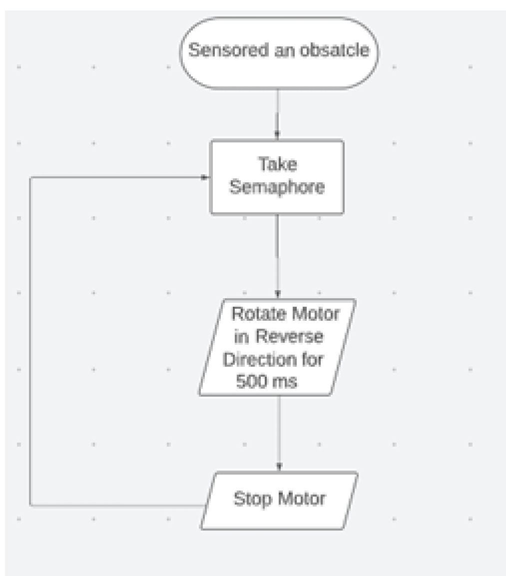

Figure 12: Jamming Flow Chart

6.5. ISR

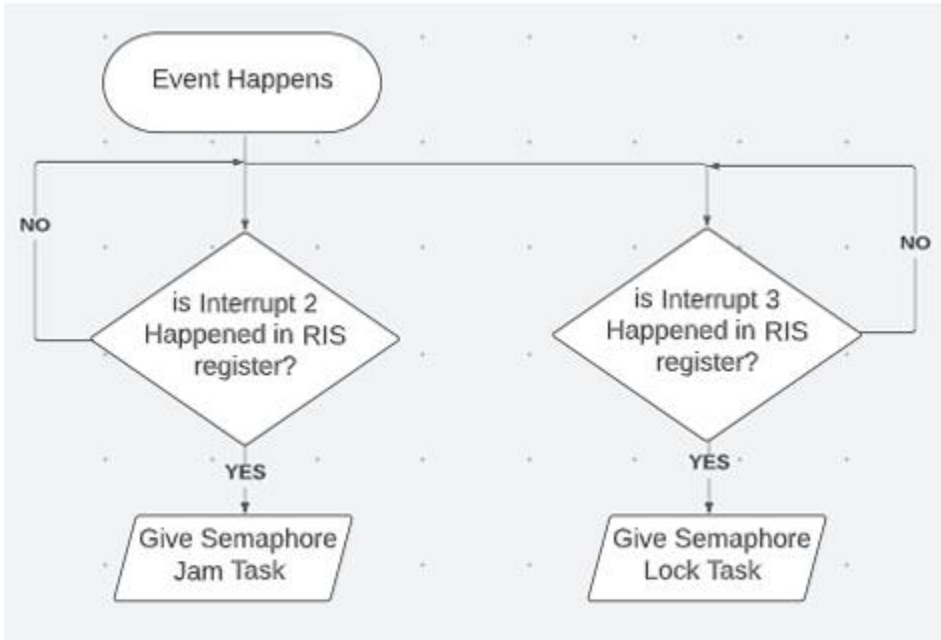

Figure 13: ISR Flow Chart

6.6. Locking

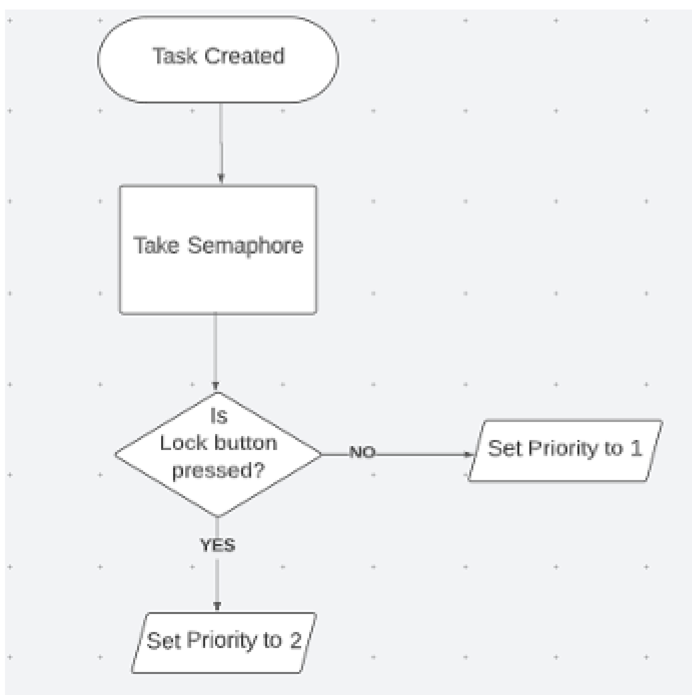

Figure 14: Locking Flow Chart

7. Problems Faced and Solutions:

1. Synchronization Issues

- **Problem:** In multitask environments, tasks often need to share resources like GPIOs or timers, leading to race conditions or deadlocks if not managed properly.
- **Solution:** Use semaphores and mutexes effectively to manage resource access. In your project, the implementation of mutexes in tasks that modify GPIO states for motor control ensures that only one task at a time can control the motor, thus preventing race conditions.

2. Interrupt Management

- **Problem:** Incorrect handling of interrupts can lead to missed signals or erratic behavior if the interrupt priorities are not managed correctly.
- **Solution:** Proper configuration of interrupt priorities and ensuring that interrupt service routines (ISRs) are kept short and efficient. Your project uses edge-triggered interrupts for handling specific events, which should be precisely configured to capture all relevant state changes without missing any.

3. Timing and Delays

- **Problem:** Ensuring tasks run at the correct times and managing delays accurately can be challenging, especially with multiple tasks influencing the same hardware components.
- **Solution:** Use of the FreeRTOS delay functions (like **vTaskDelay**) and timer configurations to manage task timing. In your project, the timer setup for delays in motor control helps ensure that operations like opening and closing windows occur for the correct durations.

4. Debugging and Testing

- **Problem:** Finding and fixing bugs in a real-time multitasking environment can be difficult, especially when issues only appear under specific timing or load conditions.
- **Solution:** Implement logging and use debugging tools that can trace task execution and resource usage. Additionally, systematic testing of each task individually before integration can help isolate and correct issues.

8. Corner Cases

1. **CASE 1:** (Driver & Passenger Press Simultaneously)
   - In this case, our system responds by giving control to which ever button was pressed first. For example, if driver pressed UP first and the passenger pressed DOWN after, the motor will rotate UPWARDS.
2. **CASE 2:** (Activating Lock while Passenger PB is pressed)
   - Since the lock button uses interrupt when triggered, the lock will disable the passenger control panel immediately and will stop the motor.
3. **CASE 3: (** Pressing PB again while automatic is activated)
   - If the motor is in automatic mode, then the driver or passenger pressed the button again, the motor will exit automatic mode and execute whichever task is sent to it. It can be automatic again in the other direction, or manual mode in either direction.
4. **CASE 4:** (Using PB during Jam Protection Process)
   - The jam protection task has the highest priority in our system. Any other activity is blocked until the task is done executing.