EUI University

EME initiative

Report of:

RC Car Project

Prepared By:

Mahmoud Badr Rabea 336

Adham EL-Serougy 1323

Omar Ayman 1045

Hazem Magdy 1004

Contents

[Introduction: 1](#_Toc145681230)

[Components: 1](#_Toc145681231)

[Working Principle: 3](#_Toc145681232)

[System Hardware Design: 5](#_Toc145681233)

[1. project components layout 5](#_Toc145681234)

[2. System Schematic: 6](#_Toc145681235)

[System Software Design 6](#_Toc145681236)

[1. Static Design 6](#_Toc145681237)

[1.1. The Layered Architecture 7](#_Toc145681238)

[1.2. Description For Each Module APIs and Types: 7](#_Toc145681239)

[2. Dynamic Design 41](#_Toc145681240)

[2.1. System State Machine 41](#_Toc145681241)

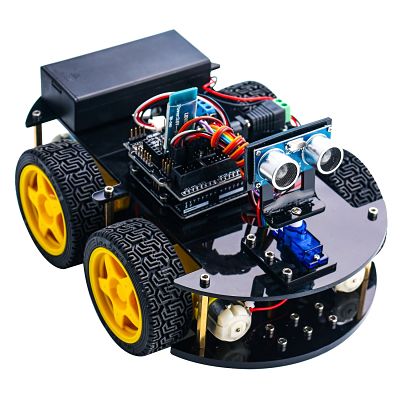
[Conclusion: 41](#_Toc145681242)

# Introduction:

The RC Car with Ultrasonic and LDR Sensors project combines obstacle detection, brightness-based navigation, and a time-controlled start/stop feature. By integrating an ultrasonic sensor, two LDR sensors, interrupt-based switches, a non-preemptive priority-based scheduler with a first delay feature, and a system priority design based on the Gomaa criteria, this project creates an autonomous remote-controlled car. The car can be initiated using one of two switches and will continue moving until either the other switch is pressed or a predefined time of 60 seconds elapses. These additional features enhance the car's functionality, optimize resource utilization, and improve responsiveness. The HCSR04 model is selected as the ultrasonic sensor for accurate obstacle detection.

# Components:

1. RC Car: The RC car serves as the mobile platform for this project, allowing for remote control and autonomous movement based on sensor inputs.



1. HCSR04 Ultrasonic Sensor: The ultrasonic sensor is positioned on the front of the car to detect obstacles within its path. It emits ultrasonic waves and measures the time it takes for the waves to bounce back from obstacles, calculating the distance to the nearest obstacle.



1. LDR Sensors: Two LDR sensors are placed on either side of the car to detect the brightness levels in the surroundings. These sensors change their resistance based on the amount of light falling on them, providing information about the relative brightness on each side.



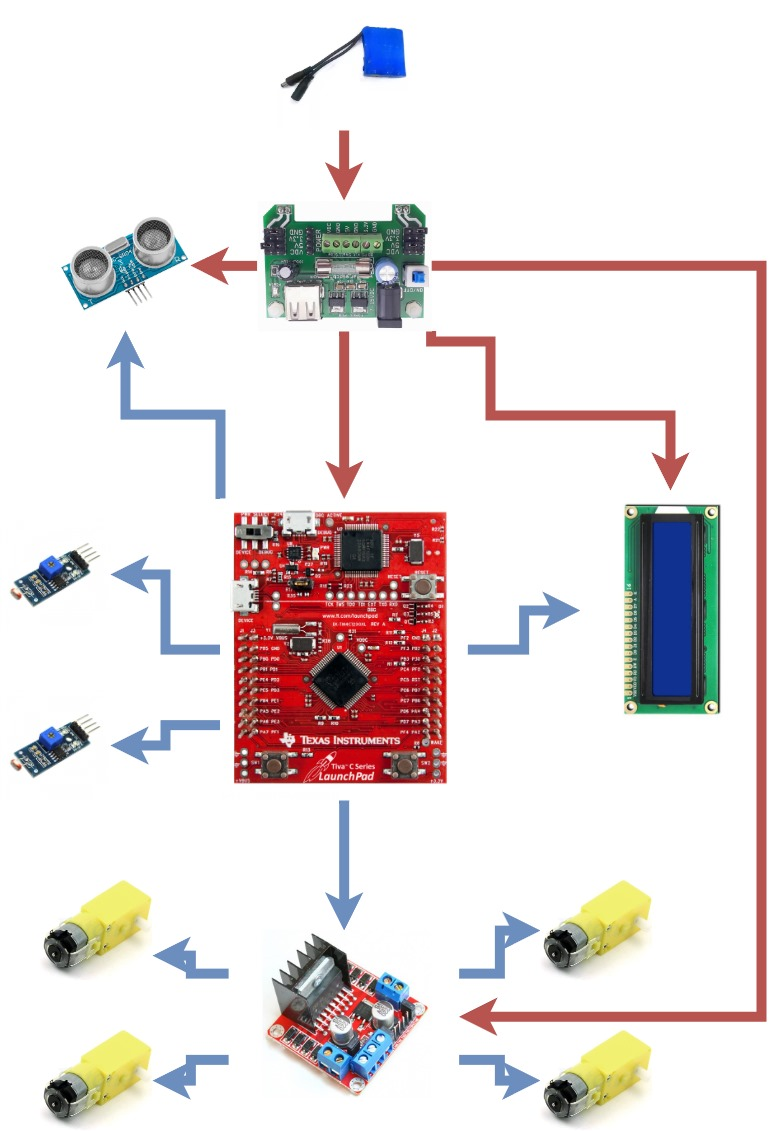
1. Switches: Two switches are incorporated into the system. One switch initiates the car's movement, while the other switch stops it. The movement will also automatically cease after 60 seconds. The switches incorporated into the system are equipped with interrupt functionality. When activated, they generate interrupts that trigger immediate responses, eliminating the need for continuous polling and ensuring quick and reliable interactions.
2. Scheduler: A scheduler is implemented to control the timing of the car's actions. It enables the car to start moving upon the activation of the first switch and to stop when the second switch is pressed or when the predetermined time of 60 seconds has passed. The scheduler implemented in the system utilizes a non-preemptive priority-based approach. It assigns priorities to different tasks within the RC car based on the Gomaa criteria, considering factors such as task importance, dependencies, and resource utilization. The scheduler includes a first delay mechanism, introducing a delay before some tasks at the start of the system that makes the tasks being executed at different time so we can reduce CPU load and optimize task execution.

# Working Principle:

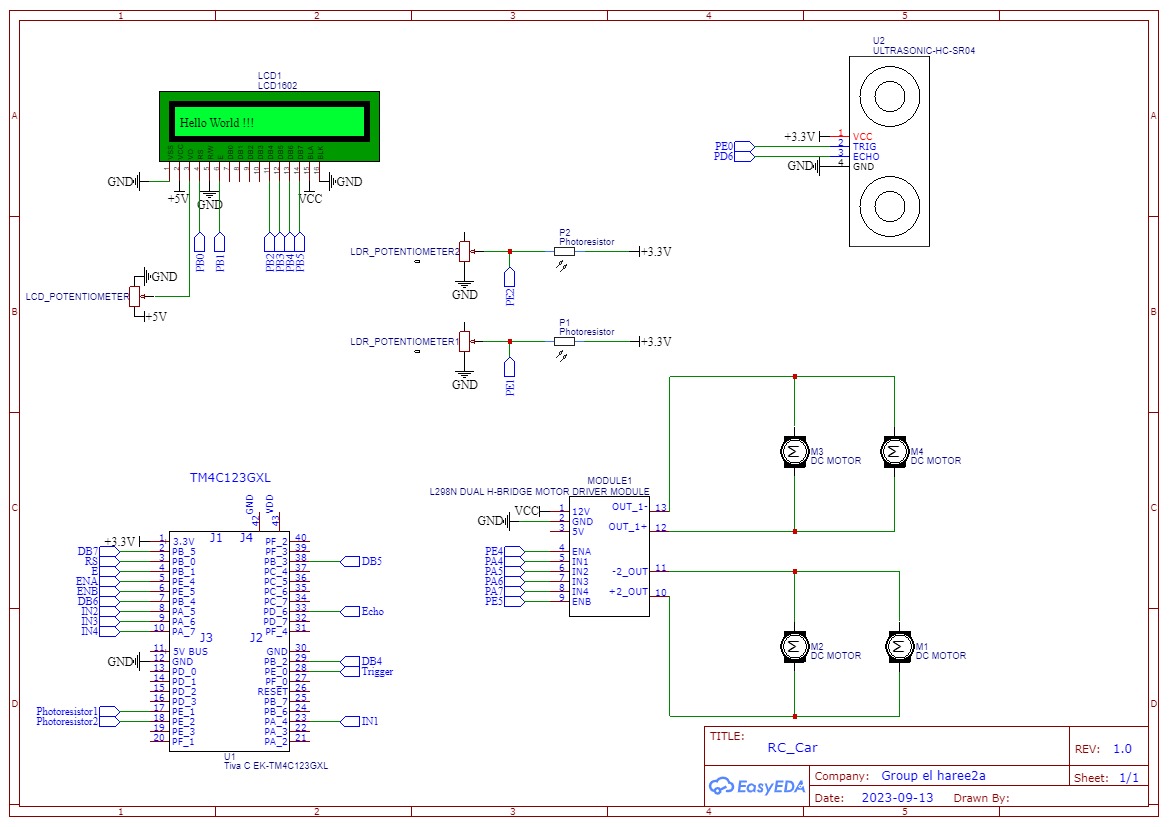
* Start and Stop Mechanism:
* The system is initialized with the car in a stationary state.
* When the first switch is activated, the scheduler triggers the car to start moving.
* The car continues its movement until the second switch is pressed or the 60-second time limit is reached.
* If the second switch is pressed, the scheduler instructs the car to stop immediately.
* If the 60-second time limit is reached before the second switch is pressed, the scheduler automatically stops the car.
* Obstacle Detection and Avoidance:
* While the car is in motion, the ultrasonic sensor continuously emits ultrasonic waves and measures the time taken for them to bounce back.
* If an obstacle is detected within a predetermined distance that is 10 CM, the scheduler communicates with the car to halt its movement and move back until the distance is 30 CM then change its direction with 90 degrees to avoid a collision.
* Non-Preemptive Priority-Based Scheduler:
* The scheduler assigns priorities to various tasks based on their importance and system requirements, following the Gomaa criteria.
* Higher priority tasks, such as obstacle detection and switch handling, are executed before lower priority tasks.
* The non-preemptive nature of the scheduler ensures that a task is completed before a lower priority task interrupts its execution.
* Brightness-Based Navigation:
* The LDR sensors measure the brightness levels on each side of the car.
* The scheduler analyzes the readings from the LDR sensors and determines the side with the higher brightness level.
* The car adjusts its direction towards the side with greater brightness, enabling it to navigate toward well-lit areas autonomously.

# System Hardware Design:

## project components layout



## System Schematic:



# System Software Design

The main goal of software design is to create a high-quality software system that is reliable, efficient, maintainable, and scalable. We can divide this process into two subprocesses:

## Static Design

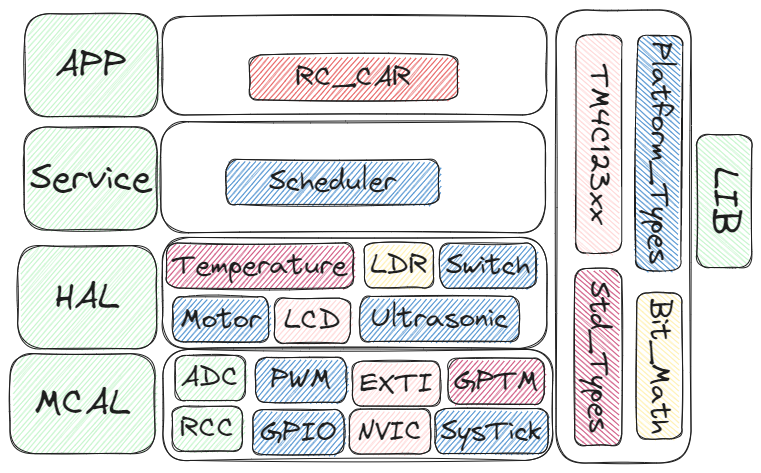
The primary goal of static design is to create a well-organized and modular software system that is easy to understand, maintain, and extend. This is achieved by defining a clear hierarchy of components and their relationships, as well as using standard design patterns and principles to ensure consistency and reusability.

Static design is typically done during the early stages of software development, before any code is written. It is often represented graphically using diagrams such as class diagrams, package diagrams, and component diagrams.

Some of the key principles and techniques used in static design include:

### The Layered Architecture

The layered architecture typically consists of three or more layers, with each layer having a clear separation of concerns so that every layer has a specific responsibility.



### Description For Each Module APIs and Types:

In this step we define each module APIs that will be used to communicate with the module by the others.

So, we will start with our components starting with MCAL layer to APP layer.

1. MCAL:
   1. RCC

|  |  |
| --- | --- |
| Name | RCC\_Prph\_t |
| Type | Enum |
| Description | The ID of peripheral |

|  |  |
| --- | --- |
| Name | RCC\_PrphClkState\_t |
| Type | Enum |
| Description | The peripheral clock state |

|  |  |
| --- | --- |
| Name | RCC\_SetPrephralClockState |
| Syntax | ErrorState\_t RCC\_SetPrephralClockState(RCC\_Prph\_t Copy\_Peripheral, RCC\_PrphClkState\_t Copy\_ClockState) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Peripheral: The peripheral to control its clock  Copy\_ClockState: The state to be set |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Enable The clock to a peripheral. |

|  |  |
| --- | --- |
| Name | ErrorState\_t RCC\_PeripheralSWReset |
| Syntax | ErrorState\_t RCC\_PeripheralSWReset(RCC\_Prph\_t Copy\_Peripheral) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Peripheral: The peripheral to control its clock |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Do software reset on a peripheral. |

* 1. GPIO

|  |  |
| --- | --- |
| Name | GPIO\_Port\_t |
| Type | Enum |
| Description | The GPIO Port |

|  |  |
| --- | --- |
| Name | GPIO\_Pin\_t |
| Type | Enum |
| Description | The GPIO Pin |

|  |  |
| --- | --- |
| Name | GPIO\_PinDirection\_t |
| Type | Enum |
| Description | The Direction of GPIO Pin |

|  |  |
| --- | --- |
| Name | GPIO\_PINMode\_t |
| Type | Enum |
| Description | The Mode of GPIO Pin |

|  |  |
| --- | --- |
| Name | GPIO\_PinPullUpDown\_t |
| Type | Enum |
| Description | The Pull up down state of GPIO Pin |

|  |  |
| --- | --- |
| Name | GPIO\_PinState\_t |
| Type | Enum |
| Description | The push pull / open drain state of GPIO Pin |

|  |  |
| --- | --- |
| Name | GPIO\_PinValue\_t |
| Type | Enum |
| Description | The Value of GPIO Pin |

|  |  |
| --- | --- |
| Name | GPIO\_PortValue\_t |
| Type | Enum |
| Description | The Value of GPIO Port |

|  |  |
| --- | --- |
| Name | GPIO\_AltFunc\_t |
| Type | Enum |
| Description | The Alternate function of GPIO Pin |

|  |  |
| --- | --- |
| Name | GPIO\_OutputCurrent\_t |
| Type | Enum |
| Description | The Output drive current of GPIO Pin |

|  |  |
| --- | --- |
| Name | GPIO\_Config\_t |
| Type | Struct |
| Description | The Configuration structure of GPIO Pin |

|  |  |
| --- | --- |
| Name | GPIO\_Init |
| Syntax | ErrorState\_t GPIO\_Init(const GPIO\_Config\_t\* Copy\_Config,u8 Copy\_PinNum) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | GPIO\_Config\_t Array of configuration structure for GPIO  Copy\_PinNum Number of Pins to be initialized |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Initialize the GPIO Pins. |

|  |  |
| --- | --- |
| Name | GPIO\_SetPinValue |
| Syntax | ErrorState\_t GPIO\_SetPinValue(GPIO\_Port\_t Copy\_Port,GPIO\_Pin\_t Copy\_Pin,GPIO\_PinValue\_t Copy\_PinValue) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Port The port of pin to set its value  Copy\_Pin The pin to set its Value  Copy\_PinValue The Pin value |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Set Value to Digital Pin. |

|  |  |
| --- | --- |
| Name | GPIO\_GetPinValue |
| Syntax | ErrorState\_t GPIO\_GetPinValue(GPIO\_Port\_t Copy\_Port,GPIO\_Pin\_t Copy\_Pin,GPIO\_PinValue\_t\* Copy\_PinValue) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Port The port of pin to set its value  Copy\_Pin The pin to set its Value |
| Parameters (out) | Copy\_PinValue The Pin value |
| Return Value | ErrorState\_t |
| Description | Get Value of Digital Pin. |

* 1. EXTI

|  |  |
| --- | --- |
| Name | EXTI\_Port\_t |
| Type | Enum |
| Description | The EXTI Port |

|  |  |
| --- | --- |
| Name | EXTI\_Pin\_t |
| Type | Enum |
| Description | The EXTI Pin |

|  |  |
| --- | --- |
| Name | EXTI\_Detect\_t |
| Type | Enum |
| Description | The EXTI Pin Detect type edge or level |

|  |  |
| --- | --- |
| Name | EXTI\_TriggerDetect\_t |
| Type | Enum |
| Description | The EXTI Pin Trigger Type |

|  |  |
| --- | --- |
| Name | EXTI\_Config\_t |
| Type | Struct |
| Description | The EXTI Pin Configuration Structure |

|  |  |
| --- | --- |
| Name | EXTI\_Init |
| Syntax | ErrorState\_t EXTI\_Init(EXTI\_Config\_t\* Copy\_config,u8 Copy\_PinsNum) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_config Configuration structure  Copy\_PinsNum Number of Pins to be configured |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | set the required configuration to number of pins. |

|  |  |
| --- | --- |
| Name | EXTI\_SetCallBack |
| Syntax | ErrorState\_t EXTI\_SetCallBack(EXTI\_Port\_t Copy\_Port, EXTI\_Pin\_t Copy\_Pin,void (\*Copy\_CallBackFunc)(void)) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Port Port of pin to set the callback to  Copy\_Pin Pin to set the callback to  Copy\_CallBackFunc pointer to the callback function to be set |
| Return Value | ErrorState\_t |
| Description | Set the callback function to specific pin. |

* 1. GPTM

|  |  |
| --- | --- |
| Name | GPTM\_BlockNum\_t |
| Type | Enum |
| Description | GPTM Block number from TIMER0 to WTIMER5 |

|  |  |
| --- | --- |
| Name | GPTM\_Conc\_t |
| Type | Enum |
| Description | GPTM concatenation type |

|  |  |
| --- | --- |
| Name | GPTM\_Channel\_t |
| Type | Enum |
| Description | GPTM channel TIMERA or TIMERB |

|  |  |
| --- | --- |
| Name | GPTM\_ConState\_t |
| Type | Enum |
| Description | GPTM Continuity state Periodic or one shot |

|  |  |
| --- | --- |
| Name | GPTM\_Mode\_t |
| Type | Enum |
| Description | GPTM operation mode |

|  |  |
| --- | --- |
| Name | GPTM\_Count\_t |
| Type | Enum |
| Description | GPTM count type Count up or down. |

|  |  |
| --- | --- |
| Name | GPTM\_State |
| Type | Enum |
| Description | GPTM state on or off. |

|  |  |
| --- | --- |
| Name | GPTM\_Int\_t |
| Type | Enum |
| Description | GPTM Interrupt type |

|  |  |
| --- | --- |
| Name | GPTM\_EventTrigger\_t |
| Type | Enum |
| Description | GPTM Event Trigger error |

|  |  |
| --- | --- |
| Name | GPTM\_PWMOutLevel\_t |
| Type | Enum |
| Description | GPTM PWM output level |

|  |  |
| --- | --- |
| Name | GPTM\_IntState\_t |
| Type | Enum |
| Description | GPTM Interrupt State. |

|  |  |
| --- | --- |
| Name | GPTM\_Config\_t |
| Type | Struct |
| Description | GPTM Configuration structure |

|  |  |
| --- | --- |
| Name | GPTM\_Init |
| Syntax | ErrorState\_t GPTM\_Init(GPTM\_Config\_t\* Copy\_Config, TimersNum\_t Copy\_Number);Copy\_Peripheral) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Config GPTM configuration structure  Copy\_Number Channels Number |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Initialize the required Timers Channels. |

|  |  |
| --- | --- |
| Name | GPTM\_SetState |
| Syntax | ErrorState\_t GPTM\_SetState(GPTM\_BlockNum\_t Copy\_TimerNum, GPTM\_Channel\_t Copy\_TimerChannel,GPTM\_State Copy\_State) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_TimerNum Timer Block Number  Copy\_TimerChannel Timer Channel Number  Copy\_State State of timer |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Set the timer state start or stop. |

|  |  |
| --- | --- |
| Name | GPTM\_SetPWMOutputLevel |
| Syntax | ErrorState\_t GPTM\_SetPWMOutputLevel(GPTM\_BlockNum\_t Copy\_TimerNum, GPTM\_Channel\_t Copy\_TimerChannel,GPTM\_PWMOutLevel\_t Copy\_PWMOutput) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_TimerNum Timer Block Number  Copy\_TimerChannel Timer Channel Number  Copy\_PWMOutput PWM output type |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Set the timer PWM mode output type. |

|  |  |
| --- | --- |
| Name | GPTM\_SetEventTrigger |
| Syntax | ErrorState\_t GPTM\_SetEventTrigger(GPTM\_BlockNum\_t Copy\_TimerNum, GPTM\_Channel\_t Copy\_TimerChannel,GPTM\_EventTrigger\_t Copy\_Trigger) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Peripheral: The peripheral to control its clock |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Set the timer Event trigger. |

|  |  |
| --- | --- |
| Name | GPTM\_SetTimerLoadValue |
| Syntax | ErrorState\_t GPTM\_SetTimerLoadValue(GPTM\_Config\_t\* Copy\_Config, TimerValue\_t Copy\_Value) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Config GPTM configuration structure  Copy\_Value The value to be set |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Set the load value of the required timer channel. |

|  |  |
| --- | --- |
| Name | GPTM\_SetTimerValue |
| Syntax | ErrorState\_t GPTM\_SetTimerValue(GPTM\_Config\_t\* Copy\_Config, TimerValue\_t Copy\_Value); |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Config GPTM configuration structure  Copy\_Value The value to be set |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Set the value of the required timer channel. |

|  |  |
| --- | --- |
| Name | GPTM\_SetMatchValue |
| Syntax | ErrorState\_t GPTM\_SetMatchValue(GPTM\_Config\_t\* Copy\_Config, TimerValue\_t Copy\_Value,u16 Copy\_MatchPrescaler) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Config GPTM configuration structure  Copy\_Value Value to be loaded in Timer Register  Copy\_MatchPrescaler Prescaler of match counter |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Load the Match value of timer. |

|  |  |
| --- | --- |
| Name | GPTM\_GetTimerValue |
| Syntax | ErrorState\_t GPTM\_GetTimerValue(GPTM\_Config\_t\* Copy\_Config, TimerValue\_t\* Copy\_Value); |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Config GPTM configuration structure |
| Parameters (out) | Copy\_Value Current Value of Timer |
| Return Value | ErrorState\_t |
| Description | Get the current value of Timer. |

|  |  |
| --- | --- |
| Name | GPTM\_SetInterruptState |
| Syntax | ErrorState\_t GPTM\_SetInterruptState(GPTM\_BlockNum\_t Copy\_TimerNum, GPTM\_Channel\_t Copy\_TimerChannel, GPTM\_Int\_t Copy\_IntType,GPTM\_IntState\_t Copy\_IntState) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_TimerNum Timer Block Number  Copy\_TimerChannel Timer Channel Number  Copy\_IntType Interrupt to set the callback function  Copy\_IntState Interrupt state to be set |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Do software reset on a peripheral. |

|  |  |
| --- | --- |
| Name | GPTM\_SetCallBack |
| Syntax | ErrorState\_t GPTM\_SetCallBack(GPTM\_BlockNum\_t Copy\_TimerNum, GPTM\_Channel\_t Copy\_TimerChannel,GPTM\_Int\_t Copy\_IntType,void(\*Copy\_CallBackFunc)(void)); |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_TimerNum Timer Block Number  Copy\_TimerChannel Timer Channel Number  Copy\_IntType Interrupt to set the callback function  Copy\_CallBackFunc Pointer to called-back function |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Set the call back to a specific mode in a specific channel in a specific Timer block. |

* 1. NVIC

|  |  |
| --- | --- |
| Name | NVIC\_IRQ\_t |
| Type | Enum |
| Description | The Interrupt Request Number |

|  |  |
| --- | --- |
| Name | NVIC\_IntState\_t |
| Type | Enum |
| Description | The Interrupt Request State |

|  |  |
| --- | --- |
| Name | NVIC\_SetInterruptState |
| Syntax | ErrorState\_t NVIC\_SetInterruptState(NVIC\_IRQ\_t Copy\_IRQ\_n,NVIC\_IntState\_t Copy\_State) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_IRQ\_n The Interrupt Request Numbe  Copy\_State The Interrupt Request State |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Set the Interrupt Request state. |

|  |  |
| --- | --- |
| Name | NVIC\_SetPendingFlag |
| Syntax | ErrorState\_t NVIC\_SetPendingFlag(NVIC\_IRQ\_t Copy\_IRQ\_n,NVIC\_IntState\_t Copy\_State) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_IRQ\_n The Interrupt Request Numbe  Copy\_State The Pending Flag State |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Set the Interrupt Pending Flag by Software. |

|  |  |
| --- | --- |
| Name | NVIC\_GetActiveFlag |
| Syntax | ErrorState\_t NVIC\_GetActiveFlag(NVIC\_IRQ\_t Copy\_IRQ\_n,NVIC\_IntState\_t\* Copy\_State) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Config GPTM configuration structure |
| Parameters (out) | Copy\_Value Current Value of Timer |
| Return Value | ErrorState\_t |
| Description | Get the Active Flag State. |

* 1. PWM

|  |  |
| --- | --- |
| Name | PWM\_Init |
| Syntax | void PWM\_Init(PWM\_RegDef\_t \*PWMx, PWM\_InitTypeDef \*PWM\_Config) |
| Sync/Async | Synchronous |
| Parameters (in) | \*PWM\_Config: PWM peripheral configuration  \*PWMx: PWM module to initialize |
| Parameters (out) | NONE |
| Return Value | void |
| Description | Initialize PWM peripheral |

|  |  |
| --- | --- |
| Name | PWM\_Set\_Load |
| Syntax | void PWM\_Set\_Load(PWM\_RegDef\_t \*PWMx, uint8\_t Generator, uint16\_t Load\_Value) |
| Sync/Async | Synchronous |
| Parameters (in) | Load\_Value: Value to place in load register  Generator: which generator to modify  \*PWMx: PWM module to initialize |
| Parameters (out) | None |
| Return Value | void |
| Description | Set PWM Load Register value |

|  |  |
| --- | --- |
| Name | PWM\_Set\_Comp |
| Syntax | void PWM\_Set\_Comp(PWM\_RegDef\_t \*PWMx, uint8\_t Channel, uint16\_t Comp\_Value) |
| Sync/Async | Synchronous |
| Parameters (in) | Comp\_Value: Value to place in Comp register  Channel: which channel to modify  \*PWMx: PWM module to initialize |
| Parameters (out) | NONE |
| Return Value | void |
| Description | Set PWM Comp Register value |

|  |  |
| --- | --- |
| Name | PWM\_Start |
| Syntax | void PWM\_Start(PWM\_RegDef\_t \*PWMx, uint8\_t Generator) |
| Sync/Async | Synchronous |
| Parameters (in) | Generator: which Generator to modify  \*PWMx: PWM module to initialize |
| Parameters (out) | NONE |
| Return Value | void |
| Description | Start PWM Generator |

|  |  |
| --- | --- |
| Name | PWM\_Output\_Enable |
| Syntax | void PWM\_Output\_Enable(PWM\_RegDef\_t \*PWMx, uint8\_t Channel) |
| Sync/Async | Synchronous |
| Parameters (in) | Channel: which channel to modify  \*PWMx: PWM module to initialize |
| Parameters (out) | NONE |
| Return Value | void |
| Description | Enable PWM output for channel |

* 1. ADC

|  |  |
| --- | --- |
| Name | ADC\_SequencerInit |
| Syntax | void ADC\_SequencerInit(ADC\_RegDef\_t \*ADCx,  ADC\_Sequencer\_InitTypeDef \*Sequencer\_Config) |
| Sync/Async | Synchronous |
| Parameters (in) | \*ADCx: ADC module to initialize  \*Sequencer\_Config: Sequencer Configuration |
| Parameters (out) | NONE |
| Return Value | void |
| Description | Initialize ADC Sequencer |

|  |  |
| --- | --- |
| Name | ADC\_SequencerConfig |
| Syntax | void ADC\_SequencerConfig(ADC\_RegDef\_t \*ADCx,  ADC\_Sequencer\_ConfigTypeDef \*Sequence) |
| Sync/Async | Synchronous |
| Parameters (in) | \*ADCx: ADC module to initialize  \*Sequence: Sequence for sequencers |
| Parameters (out) | None |
| Return Value | void |
| Description | Set PWM Load Register value |

|  |  |
| --- | --- |
| Name | ADC\_SequenceIntEnable |
| Syntax | void ADC\_SequenceIntEnable(ADC\_RegDef\_t \*ADCx, uint8\_t Sequencer) |
| Sync/Async | Synchronous |
| Parameters (in) | \*ADCx: ADC module to initialize  Sequencer: Sequencer to modify |
| Parameters (out) | NONE |
| Return Value | void |
| Description | Enable Sequencer Interrupt |

|  |  |
| --- | --- |
| Name | ADC\_SequencerEnable |
| Syntax | void ADC\_SequencerEnable(ADC\_RegDef\_t \*ADCx, uint8\_t Sequencer) |
| Sync/Async | Synchronous |
| Parameters (in) | \*ADCx: ADC module to initialize  Sequencer: Sequencer to modify |
| Parameters (out) | NONE |
| Return Value | void |
| Description | Enable Sequencer |

|  |  |
| --- | --- |
| Name | ADC\_SequencerDisable |
| Syntax | void ADC\_SequencerDisable(ADC\_RegDef\_t \*ADCx, uint8\_t Sequencer) |
| Sync/Async | Synchronous |
| Parameters (in) | \*ADCx: ADC module to initialize  Sequencer: Sequencer to modify |
| Parameters (out) | NONE |
| Return Value | void |
| Description | Disable Sequencer |

|  |  |
| --- | --- |
| Name | ADC\_StartConversion |
| Syntax | void ADC\_StartConversion(ADC\_RegDef\_t \*ADCx, uint8\_t Sequencer) |
| Sync/Async | Synchronous |
| Parameters (in) | \*ADCx: ADC module to initialize  Sequencer: Sequencer to modify |
| Parameters (out) | NONE |
| Return Value | void |
| Description | Start conversion for processor trigger |

|  |  |
| --- | --- |
| Name | ADC\_GetData |
| Syntax | uint16\_t ADC\_GetData(ADC\_RegDef\_t \*ADCx, uint8\_t Sequencer) |
| Sync/Async | Synchronous |
| Parameters (in) | \*ADCx: ADC module to initialize  Sequencer: Sequencer to modify |
| Parameters (out) | NONE |
| Return Value | uint16\_t |
| Description | Get data from ADC FIFO |

|  |  |
| --- | --- |
| Name | ADC\_Clear\_Interrupt |
| Syntax | void ADC\_Clear\_Interrupt(ADC\_RegDef\_t \*ADCx, uint8\_t Sequencer) |
| Sync/Async | Synchronous |
| Parameters (in) | \*ADCx: ADC module to initialize  Sequencer: Sequencer to modify |
| Parameters (out) | NONE |
| Return Value | void |
| Description | Clear Interrupt Flags of Sequencer |

|  |  |
| --- | --- |
| Name | ADC\_IntRegister |
| Syntax | void ADC\_IntRegister(ADC\_RegDef\_t \*ADCx, uint8\_t Sequencer, void (\*pfnIntHandler)(void)) |
| Sync/Async | Synchronous |
| Parameters (in) | \*ADCx: ADC module to initialize  Sequencer: Sequencer to modify |
| Parameters (out) | NONE |
| Return Value | void |
| Description | Set Callback function to interrupt |

1. HAL
   1. LCD

|  |  |
| --- | --- |
| Name | LCD\_Mode\_t |
| Type | Enum |
| Description | LCD Connection Mode 4 or 8 bit |

|  |  |
| --- | --- |
| Name | LCD\_Cursor\_t |
| Type | Enum |
| Description | LCD Cursor State |

|  |  |
| --- | --- |
| Name | LCD\_Font\_t |
| Type | Enum |
| Description | LCD Font Size |

|  |  |
| --- | --- |
| Name | LCD\_LineNum\_t |
| Type | Enum |
| Description | LCD Lines Number |

|  |  |
| --- | --- |
| Name | LCD\_Config\_t |
| Type | Enum |
| Description | LCD Configuration Structure |

|  |  |
| --- | --- |
| Name | LCD\_Init |
| Syntax | ErrorState\_t LCD\_Init(LCD\_Config\_t\* Copy\_Config) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Config The LCD Configuration structure |
| Parameters (out) | Copy\_Value Current Value of Timer |
| Return Value | ErrorState\_t |
| Description | Initialize the LCD module. |

|  |  |
| --- | --- |
| Name | LCD\_SendChar |
| Syntax | ErrorState\_t LCD\_SendChar(LCD\_Config\_t\* Copy\_Config,char Copy\_Char) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Config The LCD Configuration structure  Copy\_Char character to be displayed on LCD |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Display a character on LCD. |

|  |  |
| --- | --- |
| Name | LCD\_SendString |
| Syntax | ErrorState\_t LCD\_SendString(LCD\_Config\_t\* Copy\_Config,char \*Copy\_String) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Config The LCD Configuration structure  Copy\_String String to be displayed on LCD |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Display string on LCD. |

|  |  |
| --- | --- |
| Name | LCD\_Clear |
| Syntax | ErrorState\_t LCD\_Clear(LCD\_Config\_t\* Copy\_Config) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Config The LCD Configuration structure |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Clear Display of LCD. |

|  |  |
| --- | --- |
| Name | LCD\_WriteNumber |
| Syntax | ErrorState\_t LCD\_WriteNumber(LCD\_Config\_t\* Copy\_Config,s64 Copy\_Number) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Config The LCD Configuration structure  Copy\_Number Number to be displayed on LCD |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Display Number on LCD. |

|  |  |
| --- | --- |
| Name | LCD\_GoToXY |
| Syntax | ErrorState\_t LCD\_GoToXY(LCD\_Config\_t\* Copy\_Config,u8 Copy\_XPosition,u8 Copy\_YPosition) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Config The LCD Configuration structure  Copy\_XPosition Location on the row  Copy\_YPosition Location on the Column |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Go to specific position on LCD. |

* 1. Switch

|  |  |
| --- | --- |
| Name | Switch\_State\_t |
| Type | Enum |
| Description | Switch State Pressed or Not |

|  |  |
| --- | --- |
| Name | Switch\_Init |
| Syntax | ErrorState\_t Switch\_Init(GPIO\_Port\_t Copy\_SwPort,GPIO\_Pin\_t Copy\_SwPin,GPIO\_PinPullUpDown\_t Copy\_SwState) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_SwPort The port of pin switch connected to  Copy\_SwPin The pin switch connected to  Copy\_SwState The wanted Pull state of Switch |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Clear Display of LCD. |

|  |  |
| --- | --- |
| Name | Switch\_IntConfig |
| Syntax | ErrorState\_t Switch\_IntConfig(EXTI\_Config\_t \*Copy\_Button,void(\*Copy\_NotificationFunc)(void)) |
| Sync/Async | Asynchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Button The Interrupt Configuration structure of the Switch  Copy\_NotificationFunc The Notification function of the switch Pressing |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Initialize The interrupt configuration of the Switch. |

|  |  |
| --- | --- |
| Name | Switch\_GetStatus |
| Syntax | ErrorState\_t Switch\_GetStatus(GPIO\_Port\_t Copy\_SwPort,GPIO\_Pin\_t Copy\_SwPin,GPIO\_PinPullUpDown\_t Copy\_SwPullType,Switch\_State\_t\* Copy\_SwState) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_SwPort The port of pin switch connected to  Copy\_SwPin The pin switch connected to  Copy\_Sw PullType The wanted Pull state of Switch |
| Parameters (out) | Copy\_SwState The Switch state Pressed or Not |
| Return Value | ErrorState\_t |
| Description | Get the Switch state Pressed or not. |

* 1. Motor

|  |  |
| --- | --- |
| Name | Motor\_Init |
| Syntax | void Motor\_Init(void) |
| Sync/Async | Synchronous |
| Parameters (in) | NONE |
| Parameters (out) | NONE |
| Return Value | void |
| Description | Initialize Motors |

|  |  |
| --- | --- |
| Name | Motor\_Start |
| Syntax | void Motor\_Start(void) |
| Sync/Async | Synchronous |
| Parameters (in) | NONE |
| Parameters (out) | NONE |
| Return Value | void |
| Description | Start Motors |

|  |  |
| --- | --- |
| Name | Motor\_Set\_Speed |
| Syntax | void Motor\_Set\_Speed(uint16\_t Speed, uint8\_t Motors) |
| Sync/Async | Synchronous |
| Parameters (in) | Speed: Speed in percentage  Motors: which motor to change speed |
| Parameters (out) | NONE |
| Return Value | void |
| Description | Set Motor Speed |

|  |  |
| --- | --- |
| Name | void Motor\_Set\_Direction(uint8\_t Direction) |
| Syntax | Motor\_Set\_Direction |
| Sync/Async | Synchronous |
| Parameters (in) | Direction: Forward or Backward |
| Parameters (out) | NONE |
| Return Value | void |
| Description | Set Motor Direction |

|  |  |
| --- | --- |
| Name | Motor\_Stop |
| Syntax | void Motor\_Stop(void) |
| Sync/Async | Synchronous |
| Parameters (in) | NONE |
| Parameters (out) | NONE |
| Return Value | void |
| Description | Stop Motors |

* 1. Ultrasonic

|  |  |
| --- | --- |
| Name | Ultra\_Sonic\_init |
| Syntax | void Ultra\_Sonic\_init(void); |
| Sync/Async | Asynchronous |
| Parameters (in) | NONE |
| Parameters (out) | NONE |
| Return Value | void |
| Description | Function to initialize the GPIO pins required (PE0)  for the trigger (ultra sonic input)  and (PD6 widetimer5 channel A) for echo (ultrasonic output) |

|  |  |
| --- | --- |
| Name | Get\_Distance |
| Syntax | void Get\_Distance(void); |
| Sync/Async | Synchronous |
| Parameters (in) | u8 captureFlag: flag to indicate that interrupt happened at both edges  u64 risingEdgeTime: to get the echo rising time  u64 fallingEdgeTime: to get the echo falling time  u64 difference: to get the difference between two times  f64 time: to get the whole time according to frequency of CPU |
| Parameters (out) | u64\* distance\_global: this the required distance from the ultrasonic sensor |
| Return Value | void |
| Description | Function to get the required distance by the ultrasonic sensor by modifying the global pointer and after get the distance from the function then it calls the global pointer to function to execute the required function which address is put in this pointer |

|  |  |
| --- | --- |
| Name | ultrasonic\_distance |
| Syntax | void ultrasonic\_distance (u64\*distance\_local,void (\*ptr\_func\_local)(void)); |
| Sync/Async | Asynchronous |
| Parameters (in) | u64\*distance\_local: to assign the distance saved from global pointer to this local pointer  ptr\_func\_local: to put required function to be executed after get the distance from ultrasonic |
| Parameters (out) | NONE |
| Return Value | void |
| Description | Function gets the distance which calculated and execute the required function which its name is put in this pointer to function |

* 1. Temperature

|  |  |
| --- | --- |
| Name | Temperature\_Init |
| Syntax | void Temperature\_Init(void); |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | void |
| Description | Function to initialize the built in Temperature Sensor and  enable ADC0 module & Sequencer3. |

|  |  |
| --- | --- |
| Name | Temp\_Send\_Read |
| Syntax | u32 Temp\_Send\_Read(void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | U32 |
| Description | This function returns the Temperature value calculated. |

* 1. LDR

|  |  |
| --- | --- |
| Name | LDR\_Init |
| Syntax | void LDR\_Init(void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | Void |
| Description | This function stores the values of the two LDR Sensors in an  array. |

|  |  |
| --- | --- |
| Name | LDR\_Read |
| Syntax | void LDR\_Read(u32\* Arr) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | None |
| Parameters (out) | Arr : The Readings of LDRs |
| Return Value | void |
| Description | This function stores the values of the two LDR Sensors in an  array. |

1. Service
   1. SysTick

|  |  |
| --- | --- |
| Name | STK\_ClcSource\_t |
| Type | Enum |
| Description | SysTick Clock Source |

|  |  |
| --- | --- |
| Name | STK\_IntState\_t |
| Type | Enum |
| Description | SysTick Int State. |

|  |  |
| --- | --- |
| Name | STK\_Init |
| Syntax | ErrorState\_t STK\_Init(STK\_ClcSource\_t Copy\_ClockSource, STK\_Value\_t Copy\_Value, STK\_IntState\_t Copy\_IntState) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_ClockSource SysTick Clock Source  Copy\_Value The Top Value of SysTick  Copy\_IntState The Initial interrupt state |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | initialize the SysTick with required configuration. |

|  |  |
| --- | --- |
| Name | STK\_SetIntState |
| Syntax | ErrorState\_t STK\_SetIntState(STK\_IntState\_t Copy\_IntState) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_IntState The interrupt state |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | set the SysTick interrupt state. |

|  |  |
| --- | --- |
| Name | STK\_SetCallBack |
| Syntax | ErrorState\_t STK\_SetCallBack(void (\*Copy\_CallBackFunc)(void)) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_CallBackFunc Pointer to SysTick Callback Function |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Set the SysTick callback function. |

|  |  |
| --- | --- |
| Name | STK\_SetValue |
| Syntax | ErrorState\_t STK\_SetValue(STK\_Value\_t Copy\_Value) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Value The Top Value of SysTick |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Set the SysTick Top Value. |

|  |  |
| --- | --- |
| Name | STK\_Delyms |
| Syntax | void STK\_Delyms(u32 Copy\_Delayms) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Delayms Delay in ms |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | used for delay of max 4192 ms. |

* 1. Scheduler

|  |  |
| --- | --- |
| Name | Task\_State\_t |
| Type | Enum |
| Description | Task State Suspended or Resumed |

|  |  |
| --- | --- |
| Name | Create\_Task |
| Syntax | ErrorState\_t Create\_Task(void(\*Copy\_TaskFunc)(void),u32 Copy\_Periodicity,u32 Copy\_FirstDelay,u8 Copy\_Priority) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_TaskFunc Pointer to Task function  Copy\_Periodicity Task Periodicity  Copy\_FirstDelay Task First delay used to get offset of starting task  Copy\_Priority Task Priority with Max 0 and Minimum NUM\_OF\_TASKS-1 |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Function used to create Task for user. |

|  |  |
| --- | --- |
| Name | Resunme\_Task |
| Syntax | ErrorState\_t Resunme\_Task(u8 Copy\_Priority) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Priority Task Priority with Max 0 and Minimum NUM\_OF\_TASKS-1 |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Resume task. |

|  |  |
| --- | --- |
| Name | Suspend\_Task |
| Syntax | ErrorState\_t Suspend\_Task(u8 Copy\_Priority) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Priority Task Priority with Max 0 and Minimum NUM\_OF\_TASKS-1 |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Suspend task. |

|  |  |
| --- | --- |
| Name | Delete\_Task |
| Syntax | ErrorState\_t Delete\_Task(u8 Copy\_Priority) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Copy\_Priority Task Priority with Max 0 and Minimum NUM\_OF\_TASKS-1 |
| Parameters (out) | None |
| Return Value | ErrorState\_t |
| Description | Delete task. |

|  |  |
| --- | --- |
| Name | Tasks\_Sceduler |
| Syntax | void Tasks\_Sceduler(void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | Void |
| Description | Start scheduling tasks. |

1. App

|  |  |
| --- | --- |
| Name | CAR\_Init |
| Syntax | void CAR\_Init(void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | Void |
| Description | Initialization of hardware and creation of tasks. |

|  |  |
| --- | --- |
| Name | UltraSonic\_Task |
| Syntax | void UltraSonic\_Task(void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | Void |
| Description | Get the Ultrasonic Reading. |

|  |  |
| --- | --- |
| Name | avoid\_obstacles |
| Syntax | void avoid\_obstacles(void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | Void |
| Description | A task Implements the logic of obstacles avoid. |

|  |  |
| --- | --- |
| Name | CarStart\_Task |
| Syntax | void CarStart\_Task(void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | Void |
| Description | Start Moving the Car. |

|  |  |
| --- | --- |
| Name | CarStop\_Task |
| Syntax | void CarStop\_Task(void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | Void |
| Description | Stop Moving the Car. |

|  |  |
| --- | --- |
| Name | Watch\_Task |
| Syntax | void Watch\_Task(void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | Void |
| Description | Calculate the elapsed time of the car moving. |

|  |  |
| --- | --- |
| Name | ldr\_swing\_car |
| Syntax | void ldr\_swing\_car(void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | Void |
| Description | A task that Implements the logic of Swing of car. |

|  |  |
| --- | --- |
| Name | LCD\_Distancedisplay |
| Syntax | void LCD\_Distancedisplay(void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | Void |
| Description | Display the Distance on LCD. |

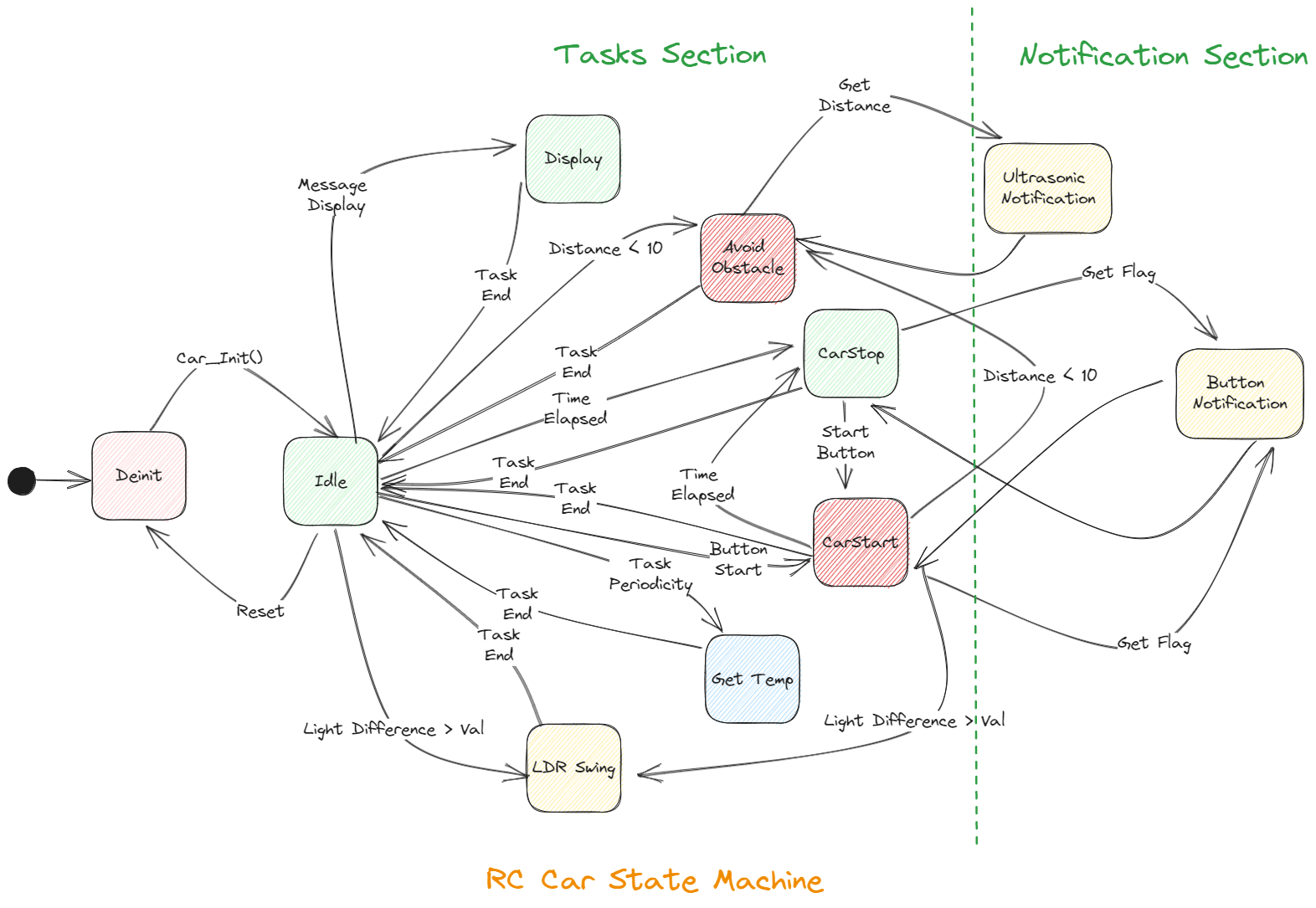
|  |  |
| --- | --- |
| Name | LCD\_LDRDisplay |
| Syntax | void LCD\_LDRDisplay(void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | Void |
| Description | Display the Difference of the 2 LDRs Reading on LCD. |

|  |  |
| --- | --- |
| Name | LCD\_TimeDisplay |
| Syntax | void LCD\_TimeDisplay(void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | Void |
| Description | Display the elapsed Time on LCD |

|  |  |
| --- | --- |
| Name | Temperature\_Task |
| Syntax | void Temperature\_Task(void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | Void |
| Description | Get the temperature every 4 Seconds. |

## Dynamic Design

### System State Machine



# Conclusion:

The RC Car with Ultrasonic and LDR Sensors project has been enhanced with a time-controlled start/stop feature and a scheduler. This addition allows the car to commence its movement upon the activation of one switch and continue until the other switch is pressed or a predetermined time of 60 seconds elapses. With the integration of the ultrasonic sensor, LDR sensors, and scheduler, the car can autonomously detect obstacles, navigate towards brighter areas, and operate within predefined time constraints. This project demonstrates the application of sensor integration and scheduling techniques in creating an intelligent and time-controlled RC car.