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

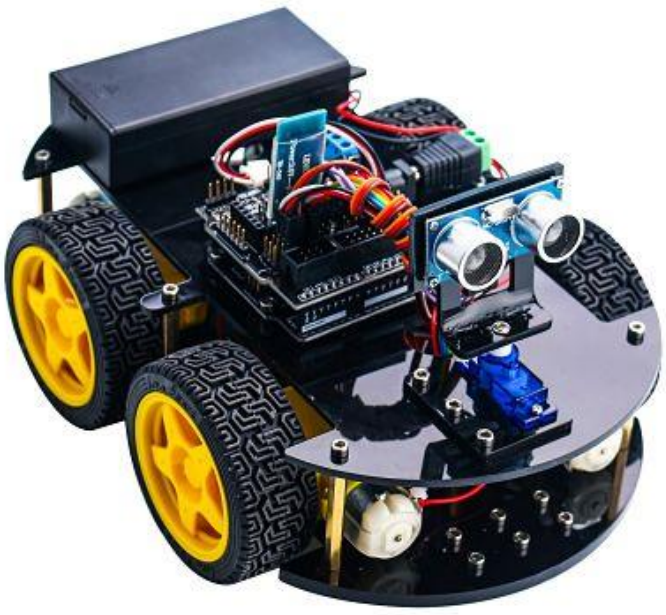
Introduction: 1
Components: 1
Working Principle: 3
System Hardware Design: 5
1. project components layout 5
2. System Schematic: 6
System Software Design 6
1. Static Design 6
1.1. The Layered Architecture 7
1.2. Description For Each Module APIs and Types: 7
2. Dynamic Design 41
2.1. System State Machine 41
Conclusion: 41

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.

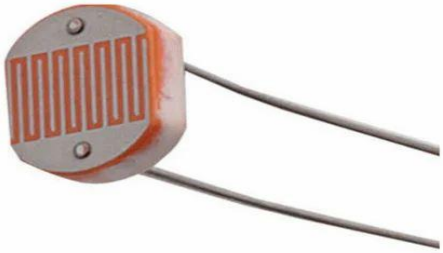
Components:

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.



2. 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.

3. 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.



4. 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.

5. 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.

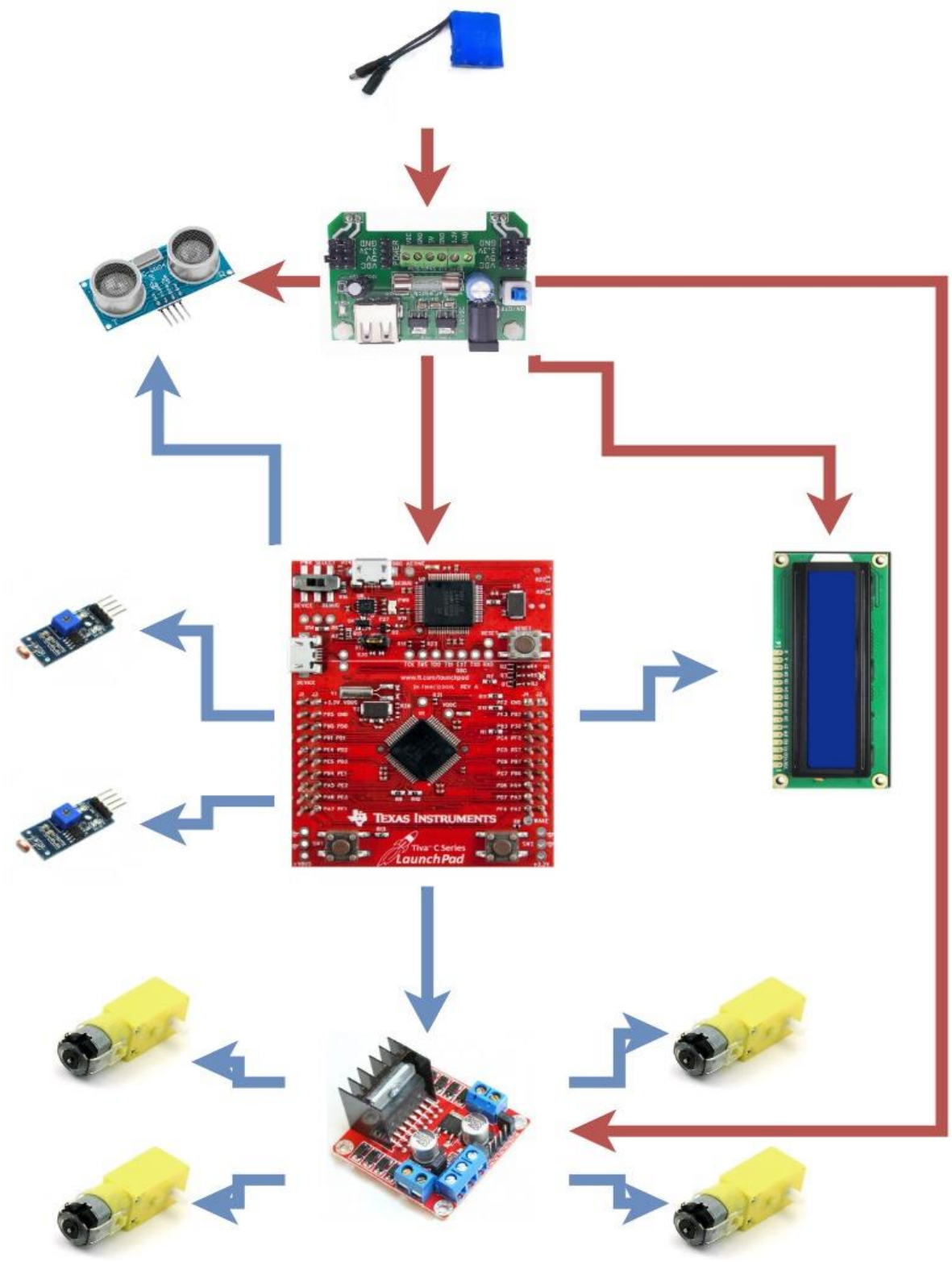
Working Principle:

- 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.

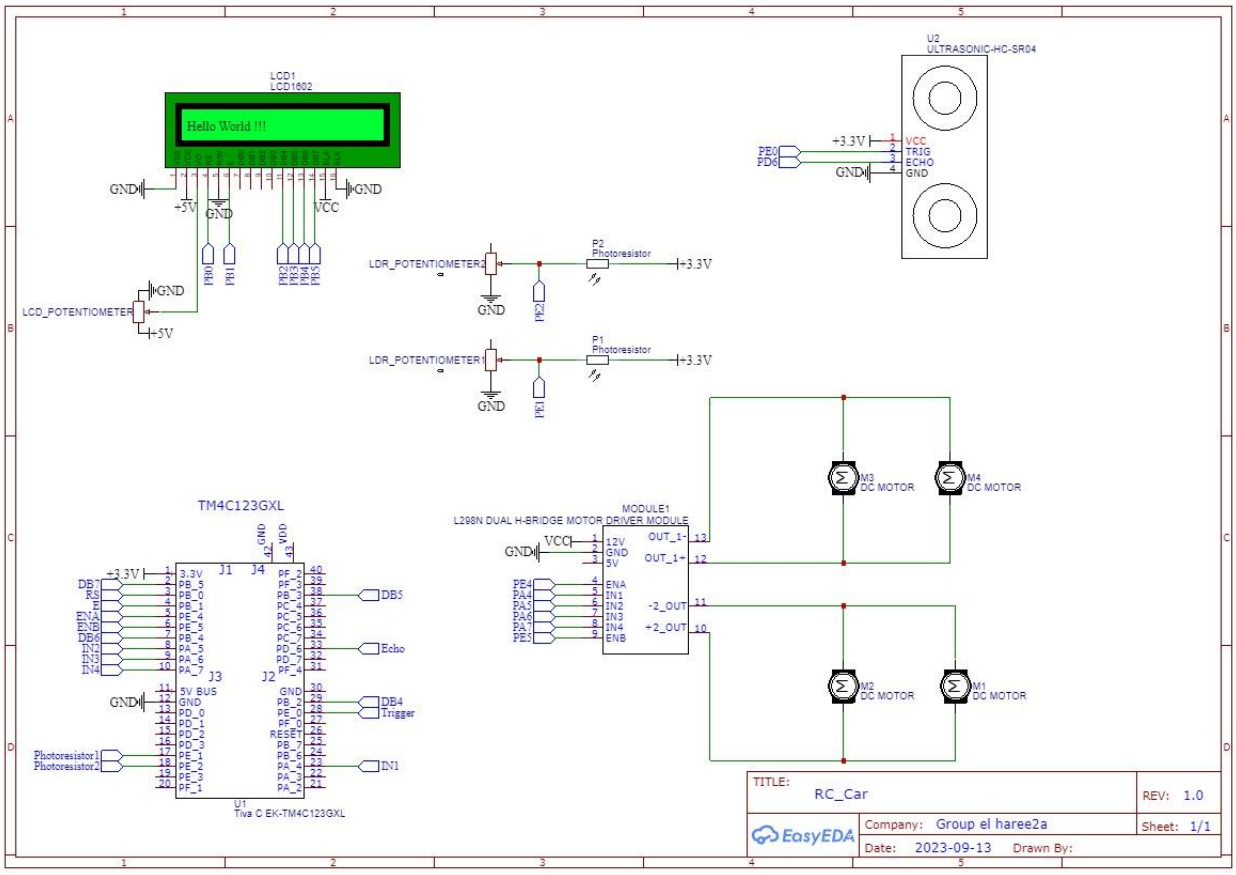
- Non-Preemptive Priority-Based Scheduler:
  - The scheduler assigns priorities to various tasks based on their importance and system requirements, following the Gomaa criteria.
  - Higher priority tasks, such as obstacle detection and switch handling, are executed before lower priority tasks.
  - The 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:

1. project components layout



2. 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:

1. 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.

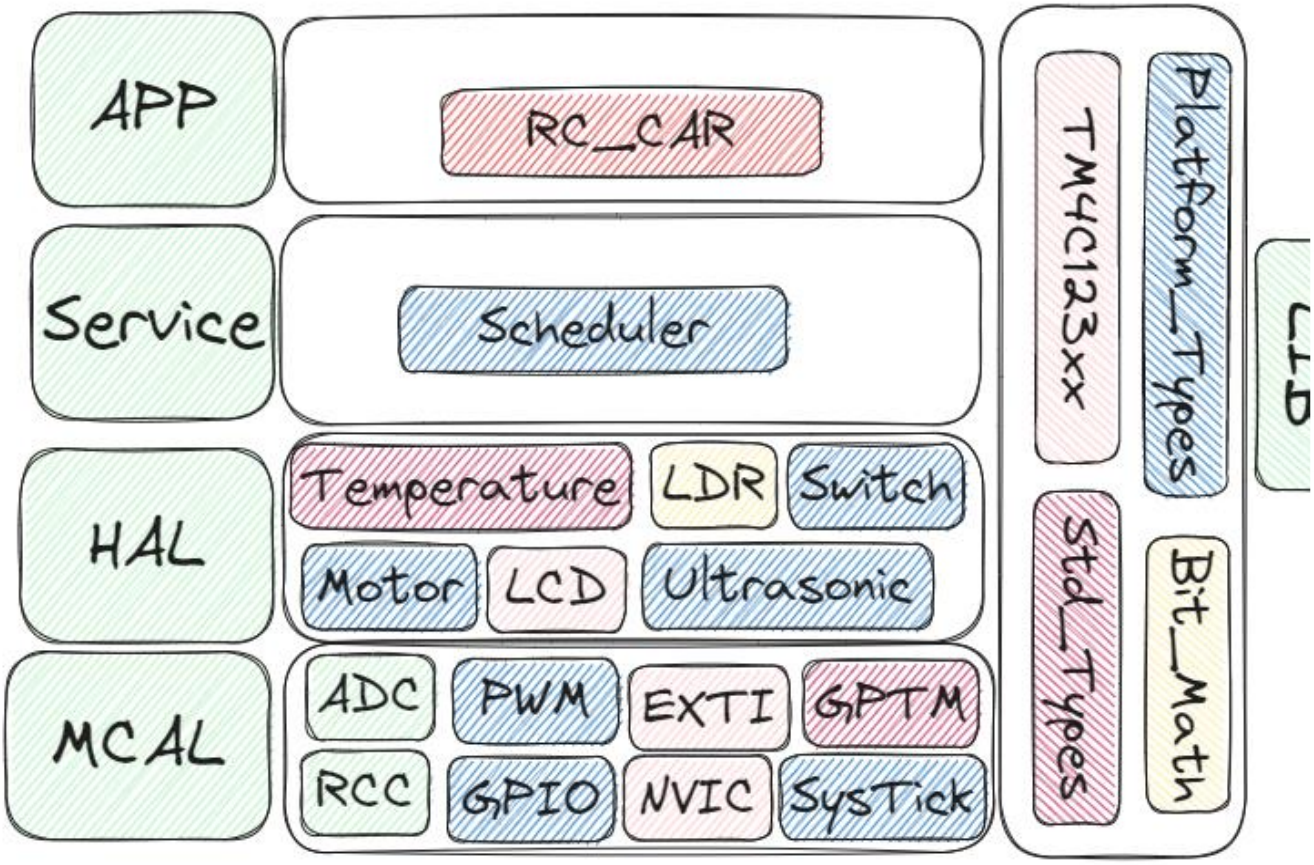
Static design is typically done during the early stages of software development, before any code is written. It is often represented

graphically using diagrams such as class diagrams, package diagrams, and component diagrams.

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

1.1. 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.



1.2. 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.

i. MCAL:

a. 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.

b. 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.

c. 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.

d. 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.

e. 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.

f. 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

g. 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

ii. HAL

a. 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.

b. 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.

c. 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

d. 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

e. 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.

f. 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.

iii. Service

a. 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.

b. 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.

iv. 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.

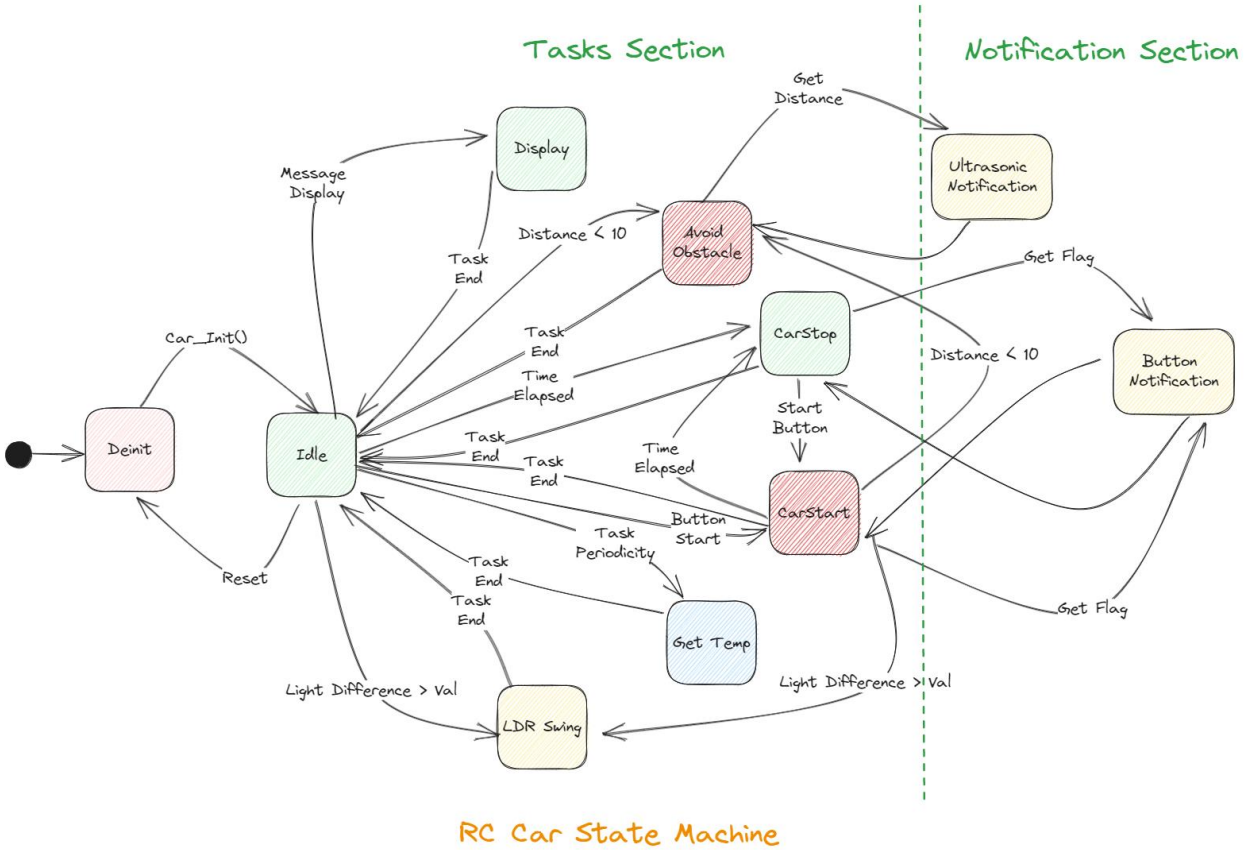
Name	LCD_LDRDisplay
Syntax	void LCD_LDRDisplay(void)
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (in)	None
Parameters (out)	None
Return Value	Void
Description	Display the Difference of the 2 LDRs Reading on LCD.

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.

2. Dynamic Design

2.1. System State Machine



RC Car 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.