Autonomous Car

Documentation Name	Autonomous Car Design	
Owner	Name	Code
	Ahmed Abdulazeem	500
	Ahmed Ragab	390
	Mahmoud Emara	1010
	Mazen Tamer	549
Team	Class 1 EME-EUI	

Document Change History

Date	**Release**	**Changed by**	**Change Description**
15-09-2023	1.0	Ahmed Abdulazeem Ahmed Ragab Mahmoud Emara Mazen Tamer	Initial release

Table of Contents

1 Introduction and Functional Overview

Autonomous car robots have rapidly advanced in recent years, showcasing their potential in various applications. In this article, we introduce a cutting-edge autonomous car robot that combines light-dependent resistor (LDR) sensors for light following, an ultrasonic sensor for obstacle avoidance, a temperature sensor to measure the surrounding temperature, and an LCD display for real-time data visualization. Powered by the Tiva C Launchpad microcontroller, this robot employs a powerful scheduler software to efficiently manage its tasks, ensuring seamless operation and enhanced functionality.

1.1 Light Following with LDR Sensors

The autonomous car robot presented here utilizes LDR sensors to follow light sources in its environment. LDR sensors are passive components that change their resistance based on the intensity of light falling on them. By strategically placing these sensors around the car's chassis, it can perceive the direction of the light source and autonomously maneuver towards it. This capability makes the robot ideal for applications such as light-guiding systems, solar panel tracking, or even seeking out light sources in dimly lit environments.

1.2 Obstacle Avoidance with Ultrasonic Sensor

Safety is a paramount concern for autonomous car robots. To ensure safe navigation, this robot incorporates an ultrasonic sensor for obstacle avoidance. The ultrasonic sensor emits high-frequency sound waves and measures the time it takes for the waves to bounce back after hitting an object. By analyzing the time delay, the robot can estimate the distance between itself and the obstacle. If an obstacle is detected within a predefined range, the car's control system triggers an appropriate response, such as stopping or altering its path to avoid a collision. This capability enables the robot to navigate crowded environments with ease and reliability.

1.3 Temperature Sensing

In addition to its light-following and obstacle avoidance capabilities, this autonomous car robot is equipped with a temperature sensor. The temperature sensor measures the ambient temperature of the robot's surroundings, providing valuable environmental information. This data can be used in a variety of applications, such as monitoring temperature-sensitive environments, optimizing heating or cooling systems, or gathering information for scientific research. By incorporating temperature sensing into the robot's repertoire, it gains a broader understanding of its environment and can adapt its behavior accordingly.

1.4 LCD Display for Real-Time Data Visualization

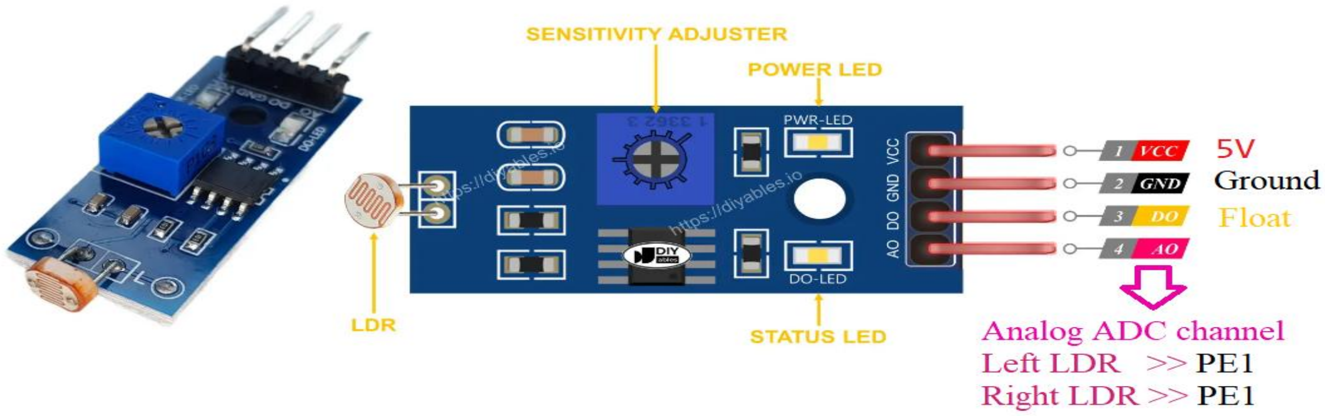
To provide real-time feedback and enhance user interaction, the autonomous car robot is equipped with an LCD display. The LCD display serves as a user interface, presenting critical information to users or observers. In this case, the display shows the temperature of the surrounding environment, the difference in LDR readings, and the time passed since the car was powered on. This visual feedback allows users to monitor the robot's performance, assess environmental conditions, and make informed decisions based on the displayed data. It also enhances the overall user experience and increases the robot's versatility.

1.5 Task Scheduling with Scheduler Software

Efficient task management is crucial for the smooth operation of an autonomous car robot. In this system, a scheduler software operates on the Tiva C Launchpad microcontroller, effectively managing the car's tasks. The scheduler software allocates processing time to each task, ensuring that the LDR system captures light data, obstacle avoidance is performed, and temperature sensing is carried out accurately and without conflicts. Additionally, it coordinates the display of data on the LCD screen, providing a seamless and synchronized user experience. The scheduler software optimizes the robot's performance and enables efficient multitasking, making it an indispensable component of the autonomous car robot's operation.

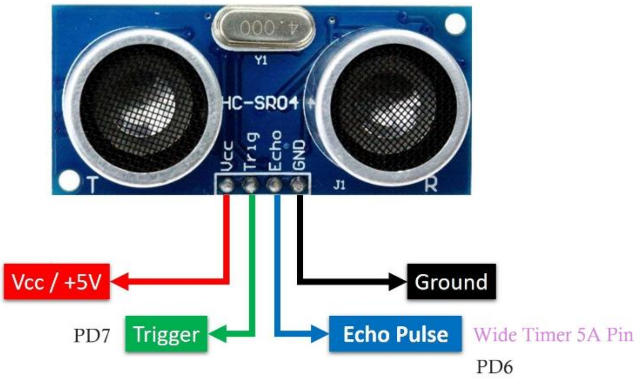
2 Sensors

2.1 Light Dependent Resistor (LDR)



The Light Dependent Resistor (LDR) is a sensor that detects light intensity and converts it into voltage. The LDR works by changing its resistance based on the amount of light it receives. When the LDR is exposed to light, its resistance decreases, which leads to a decrease in voltage. Conversely, when it is in darkness, its resistance increases, leading to an increase in voltage. This property makes it useful for detecting light in various applications. In this particular case, we have two LDR sensors - one on the left and one on the right - that are used to track light. When either of these sensors detects light, we drive the car towards the light source.

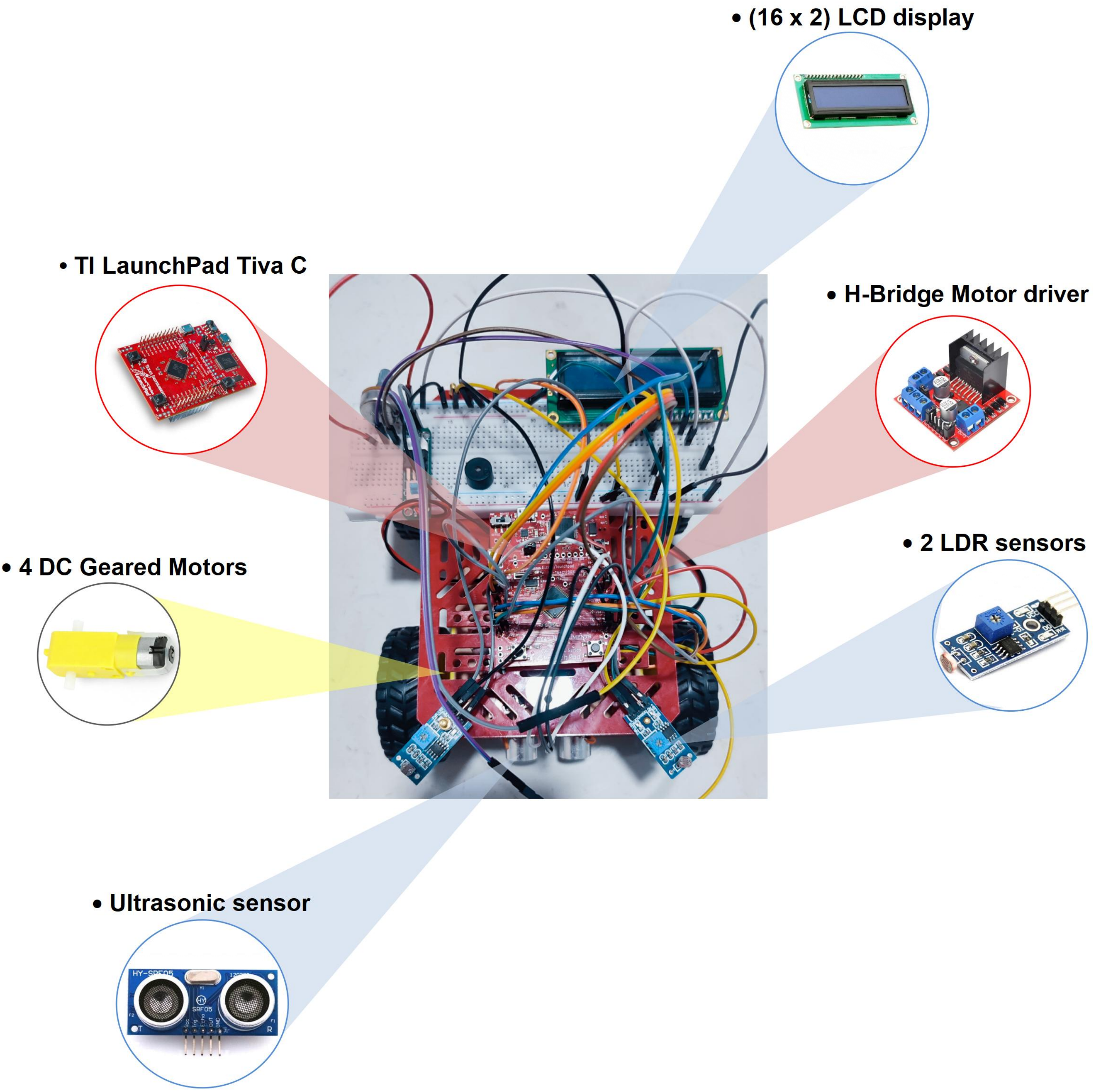
2.2 Ultrasonic Sensor



The HC-SR04 is an ultrasonic sensor module that is commonly used in robotics, automation, and other projects that involve measuring distances or detecting objects. It can measure the distance between the sensor and an object in a room or space, and it uses ultrasonic waves to do so. It is a small, low-cost sensor that is easy to use and integrate into projects. It is typically used in combination with other sensors and microcontrollers to build more complex systems. For example, it can be used to detect obstacles in a room and trigger an alarm or shutdown a robot's movement when it gets too close to a wall or other object.

The HC-SR04 has two pins for power and signal, and it outputs a voltage that ranges from 0 to 5 volts, which can be read by a microcontroller or other device. To use the HC-SR04, it simply needs to be connected to a microcontroller or other device that can interpret the voltage signal output by the sensor. The microcontroller can then use the signal to determine the distance to the object and take appropriate action based on that information.

3 Project Component layout



4 Features and delimitation

4.1 Features of the Autonomous Car Robot

1. Light Following: The robot utilizes LDR sensors to detect and follow light sources in its environment, allowing it to navigate towards light.

2. Obstacle Avoidance: An ultrasonic sensor enables the robot to detect obstacles in its path and take appropriate actions to avoid collisions, ensuring safe navigation.

3. Temperature Sensing: The robot incorporates a temperature sensor to measure the ambient temperature of its surroundings, providing valuable environmental data.

4. LCD Display: The robot is equipped with an LCD display that shows real-time information, including the surrounding temperature, the difference in LDR readings, and the time passed since the robot was powered on.

5. Tiva C Launchpad: The robot is powered by the Tiva C Launchpad microcontroller, which provides computational power, control interfaces, and memory resources necessary for efficient operation.

6. Scheduler Software: A scheduler software efficiently manages the tasks of the robot, ensuring coordinated execution of the LDR system, obstacle avoidance, temperature sensing, and LCD display functions.

4.2 Delimitations of the Autonomous Car Robot

1. Light Sensitivity: The LDR sensors are sensitive to light intensity variations but may have limitations in accurately perceiving light sources under certain conditions, such as extreme brightness or darkness.

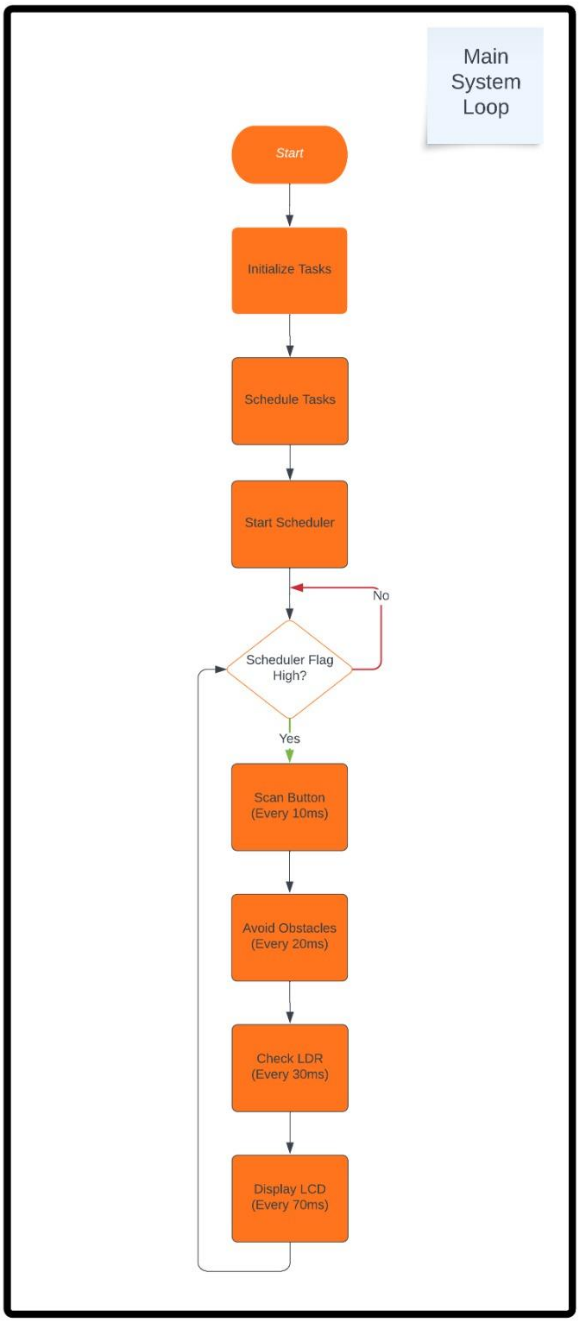
2. Obstacle Detection Range: The effectiveness of the ultrasonic sensor in detecting obstacles depends on the range and reflectivity of the objects. Very small or transparent obstacles may not be detected reliably.

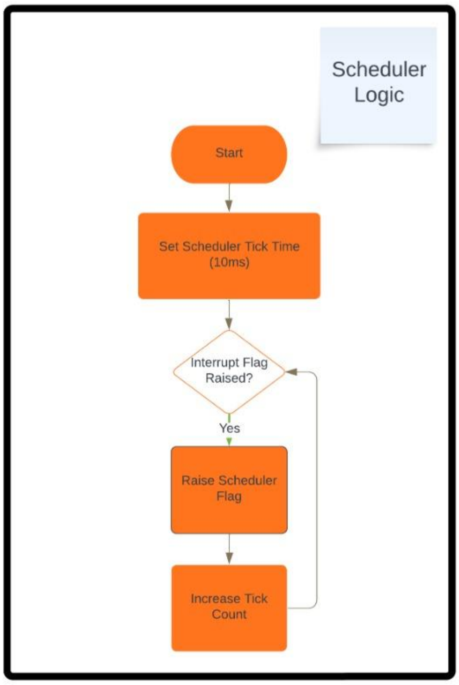
3. Temperature Measurement Accuracy: The accuracy of the temperature sensor may be influenced by factors such as sensor calibration, environmental conditions, and proximity to heat sources or airflow.

4. LCD Display Size and Complexity: The size and complexity of the LCD display may limit the amount of information that can be displayed simultaneously and the level of detail that can be presented.

5. Task Scheduling Optimization: While the scheduler software efficiently manages the tasks of the robot, there may be limitations in terms of task prioritization as it is a non preemtive scheduler

5 Project flowcharts





6 API Specification

6.1 GPIO Module

6.1.1 Type Definitions

6.1.1.1

Name	GPIO_Port_t		
Kind	Enumeration		
Range	GPIO_PORTA	0x00	GPIO Port A
	GPIO_PORTB	0x01	GPIO Port B
	GPIO_PORTC	0x02	GPIO Port C
	GPIO_PORTD	0x03	GPIO Port D
	GPIO_PORTE	0x04	GPIO Port E
	GPIO_PORTF	0x05	GPIO Port F
Description	GPIO Port Number		
Available	GPIO.h		

6.1.1.2

Name	GPIO_Pin_t		
Kind	Enumeration		
Range	PIN0	0x00	GPIO Pin 0
	PIN1	0x01	GPIO Pin 1
	PIN2	0x02	GPIO Pin 2
	PIN3	0x03	GPIO Pin 3
	PIN4	0x04	GPIO Pin 4
	PIN5	0x05	GPIO Pin 5
	PIN6	0x06	GPIO Pin 6
	PIN7	0x07	GPIO Pin 7
Description	GPIO Pin Number		
Available	GPIO.h		

6.1.1.3

Name	GPIO_PinValue_t		
Kind	Enumeration		
Range	OUTPUT_LOW	0x00	Output Low
	OUTPUT_HIGH	0x01	Output High
Description	Set Output Pin Value		
Available	GPIO.h		

6.1.2 Function Definitions

6.1.2.1

Func Name	GPIO_vidInitPort	
Syntax	void GPIO_vidInitPort(GPIO_Port_t Copy_enuPortId)	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters	Copy_enuPortId	Port Id Number
Return	None	
Description	Initialize GPIO PORT	
Available Via	GPIO.h	

6.1.2.2

Func Name	GPIO_vidOutputPin	
Syntax	void GPIO_vidOutputPin(GPIO_Port_t Copy_enuPortId, GPIO_Pin_t Copy_enuPinNum)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters	Copy_enuPortId	Port Id Number
	Copy_ enuPinNum	Pin Number
Return	None	
Description	Set Specific Pin as an output	
Available Via	GPIO.h	

6.1.2.3

Func Name	GPIO_vidWritePin	
Syntax	void GPIO_vidWritePin (GPIO_Port_t Copy_enuPortId, GPIO_Pin_t Copy_enuPinNum , GPIO_PinValue_t Copy_u8PinVal)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters	Copy_enuPortId	Port Id Number
	Copy_ enuPinNum	Pin Number
	GPIO_PinValue_t	value to be set
Return	None	
Description	Set Specific Pin Value	
Available Via	GPIO.h	

6.1.2.4

Func Name	GPIO_u8ReadPin	
Syntax	Uint8_t GPIO_vidOutputPin(GPIO_Port_t Copy_enuPortId, GPIO_Pin_t Copy_enuPinNum)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters	Copy_enuPortId	Port Id Number
	Copy_ enuPinNum	Pin Number
Return	Pin value	
Description	Read Specific Pin value	
Available Via	GPIO.h	

6.1.2.5

Func Name	GPIO_vidInputPin	
Syntax	void GPIO_vidInputPin (GPIO_Port_t Copy_enuPortId, GPIO_Pin_t Copy_enuPinNum)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters	Copy_enuPortId	Port Id Number
	Copy_ enuPinNum	Pin Number
Return	None	
Description	Set Specific Pin as an Input	
Available Via	GPIO.h	

6.1.2.6

Func Name	GPIO_ vidInputPinPullUp	
Syntax	void GPIO_vidInputPinPullUp (GPIO_Port_t Copy_enuPortId, GPIO_Pin_t Copy_enuPinNum)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters	Copy_enuPortId	Port Id Number
	Copy_ enuPinNum	Pin Number
Return	None	
Description	Set Specific Pin as Pull up	
Available Via	GPIO.h	

6.2 GPTM Module

6.2.1 Type Definitions

6.2.1.1

Name	timermode		
Kind	Enumeration		
Range	onesohot	0x00	One shot timer mode
	periodic	0x01	Periodic timer mode
	edgetime	0x02	Edge time timer mode
Description	General purpose timer modes		
Available	GPTM.h		

6.2.1.2

Name	timerblock		
Kind	Enumeration		
Range	timer0	0x00	Timer 0 Prephiral
	timer1	0x01	Timer 1 Prephiral
	widetimer5	0x02	Wide Timer 5 Prephiral
Description	General purpose timer blocks		
Available	GPTM.h		

6.2.1.3

Name	countdirection		
Kind	Enumeration		
Range	up	0x00	
	down	0x01	
Description	General purpose timer count direction		
Available	GPTM.h		

6.2.1.4

Name	subtimer		
Kind	Enumeration		
Range	timerA	0x00	individual timer A
	timerB	0x01	individual timer B
	concatenated	0x02	Concatenated timer
Description	General purpose sub timer		
Available	GPTM.h		

6.2.1.5

Name	edge		
Kind	Enumeration		
Range	positive	0x00	Positive edge
	negative	0x01	Negative edge
	none	0x02	--
	both	0x03	Both edges
Description	General purpose timer edge mode event		
Available	GPTM.h		

6.2.2 Function Definitions

6.2.2.1

Func Name	Timer_Vid_Init	
Syntax	void Timer_Vid_Init(t timerBlock timer, timerMode mode , countDirection direction, subtimer block, uint32_t loadregister, edge edge_state)	
Sync/Async	Synchronous	
Reentrancy	Non Re-entrant	
Parameters	timer	The timer block Prephiral
	mode	Timer mode
	direction	Timer count direction
	block	The sum timer
	loadregister	The value of the interval load register
	edge_state	The edge mode event
Return	None	
Description	Initialize the timer with the specific configurations	
Available Via	GPTM.h	

6.2.2.2

Func Name	Timer_vidSetCallbackFunction	
Syntax	void Timer_vidSetCallbackFunction(void(*Copy_ptrFunction)(void))	
Sync/Async	Synchronous	
Reentrancy	Re-entrant	
Parameters	Copy_ptrFunction	Pointer to the function that will set in the interupt
Return	None	
Description	Set the function that will be handled in the timer interrupt	
Available Via	GPTM.h	

6.2.2.3

Func Name	Timer_u32GetCurrentValue	
Syntax	uint32_t Timer_u32GetCurrentValue(timerBlock Copy_enuTimer, subtimer Copy_enuBlock)	
Sync/Async	Asynchronous	
Reentrancy	Re-entrant	
Parameters	timerBlock Copy_enuTimer	Timer Prephiral
	Subtimer Copy_enuBlock	Sub timer
Return	uint32_t Local_u32TimerValue	The tar register value
Description	Set the function that will be handled in the timer interrupt	
Available Via	GPTM.h	

6.3 SysTick Module

6.3.1 Function Definitions

6.3.1.1

Func Name	STK_vidStartUS	
Syntax	void STK_vidStartUS(uint32_t Copy_u32Microseconds)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters	Copy_u32Microseconds	SysTick Time In Microseconds
Return	None	
Description	Set-up the SYSTICK Timer to Operate in Microsecond Unit Time	
Available Via	SysTick.h	

6.3.1.2

Func Name	STK_vidStartMS	
Syntax	void STK_vidStartMS(uint32_t Copy_u32Milliseconds)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters	Copy_u32Miilliseconds	SysTick Time In Miilliseconds
Return	None	
Description	Set-up the SYSTICK Timer to Operate in Millisecond Unit Time	
Available Via	SysTick.h	

6.3.1.3

Func Name	STK_vidStartSEC
Syntax	void STK_vidStartSEC(void)
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters	None

Return	None
Description	Set-up the SYSTICK Timer To Operate in a Second Interval
Available Via	SysTick.h

6.3.1.4

Func Name	STK_vidSetInterruptCallback	
Syntax	void STK_vidSetInterruptCallback(void(*Copy_ptrFunction)(void))	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters	Copy_ptrFunction	Pointer to the Callback Function
Return	None	
Description	Set-up the SYSTICK Timer To Operate in a Second Interval	
Available Via	SysTick.h	

6.3.1.5

Func Name	STK_vidStopCounter
Syntax	void STK_vidStopCounter(void)
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters	None
Return	None
Description	Stops the SYSTICK Timer
Available Via	SysTick.h

6.4 ADC Module

6.4.1 Type Definitions

6.4.1.1

Name	Sequencers		
Kind	Enumeration		
Range	SEQ_0	0x00	GPIO Port A
	SEQ_1	0x01	GPIO Port B
	SEQ_2	0x02	GPIO Port C
	SEQ_3	0x03	GPIO Port D
Description	ADC Sequencer Number		
Available	ADC.h		

6.4.2 Function Definitions

6.4.2.1

Func Name	ADC_vidInit	
Syntax	void ADC_vidInit(Sequencers Copy_enuSeq)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters	Copy_enuSeq	Sequencer Number
Return	None	
Description	Initialize ADC Sequencer	
Available Via	ADC.h	

6.4.2.2

Func Name	ADC_u32ReadChannel	
Syntax	uint16_t ADC_u32ReadChannel (Sequencers Copy_enuSeq)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters	Copy_enuSeq	Sequencer Number
Return	uint16_t	Channel 0 read from sequencer
Description	Read channel 0 value in a specific sequencer	
Available Via	ADC.h	

6.5 PWM Module

6.5.1 Type Definitions

6.5.1.1

Name	PWM_ID		
Kind	Enumeration		
Range	PWM0	0x00	PWM Prephiral 0
	PWM1	0x01	PWM Prephiral 1
Description	PWM Prephiral numbers		
Available	PWM.h		

6.5.1.2

Name	Channel_ID		
Kind	Enumeration		
Range	Channel0	0x00	PWM channel 0
	Channel1	0x01	PWM channel 1
	Channel2	0x02	PWM channel 2
Description	PWM channel numbers		
Available	PWM.h		

6.5.2 Function Definitions

6.5.2.1

Func Name	PWM_vidInit
Syntax	void PWM_vidInit(Void)
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters	None
Return	None
Description	Initialize the PWM Prephiral and Alternative function pins used
Available Via	PWM.h

6.5.2.2

Func Name	PWM_vidSetDutyCycle	
Syntax	void PWM_vidSetDutyCycle(Channel_ID Copy_Channel_ID, uint8_t Copy_u8DutyValue)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters	Copy_Channel_ID	channel Number of the PWM
	Copy_u8DutyValue	Duty Cycle Value
Return	None	
Description	Set the duty cycle of a given channel	
Available Via	PWM.h	

6.6 Temperature Module

6.6.1 Function Definitions

6.6.1.1

Func Name	Temperature_vidInit
Syntax	void Temperature_vidInit(void)
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters	None
Return	None
Description	Initialize the internal temperature sensor
Available Via	Temperature.h

6.6.1.2

Func Name	Temperature_vidGetTemperature	
Syntax	void Temperature_vidGetTemperature (uint16_t *Loc_u16Read)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters	Loc_u16Read	Pointer to a variable that holds sensor reading
Return	None	
Description	Retrieves temperature from internal sensor	
Available Via	Temperature.h	

6.7 LCD Module

6.7.1 functions Definitions

6.7.1.1

Func Name	LCD_vidInit
Syntax	void LCD_vidInit(void)
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters	None
Return	None
Description	Initialize the lcd
Available Via	LCD.h

6.7.1.2

Func Name	LCD_vidSendNibbleData	
Syntax	void LCD_vidSendNibbleData(uint8_t Local_u8_Nibble_Copy)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters	Local_u8_Nibble_Copy	A nibble of the transmitted data
Return	None	
Description	Send a nibble of data	
Available Via	LCD.h	

6.7.1.3

Func Name	LCD_vidSendNibbleCMD	
Syntax	void LCD_vidSendNibbleCMD(uint8_t Local_u8_Nibble_Copy)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters	Local_u8_Nibble_Copy	A nibble of the transmitted command

Return	None	
Description	Send a nibble of data	
Available Via	LCD.h	

6.7.1.4

Func Name	LCD_vidWriteChar	
Syntax	void LCD_vidWriteChar (uint8_t Copy_u8DataCopy)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters	Copy_u8DataCopy	The printed character
Return	None	
Description	Print a character on the lcd	
Available Via	LCD.h	

6.7.1.5

Func Name	LCD_vidWriteString	
Syntax	void LCD_vidWriteString (uint8_t* Copy_ptr_u8StringCopy)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters	Copy_ptr_u8StringCopy	The printed string
Return	None	
Description	Print a full string on the lcd	
Available Via	LCD.h	

6.7.1.6

Func Name	LCD_movecusor
Syntax	void LCD_movecusor(uint8_t Copy_u8column, uint8_t Copy_u8row)

Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters	Copy_u8column	The column bumber
	Copy_u8row	The row number
Return	None	
Description	Change the position of the cursor on the lcd	
Available Via	LCD.h	

6.7.1.7

Func Name	LCD_clear
Syntax	void LCD_clear(void)
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters	None
Return	None
Description	clear the lcd
Available Via	LCD.h

6.7.1.8

Func Name	LCD_vidWriteNumber	
Syntax	void LCD_vidWriteNumber (uint16_t Copy_u16num)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters	Copy_u16num	The printed number
Return	None	
Description	Print a number on the lcd	
Available Via	LCD.h	

6.8 Motors Module

6.8.1 Type Definitions

6.8.1.1

Name	Motors		
Kind	Enumeration		
Range	MOTOR_1	0x00	Motor Number 1
	MOTOR_2	0x01	Motor Number 2
Description	Type to select the motor number		
Available	Motor.h		

6.8.2 Function Definitions

6.8.2.1

Func Name	MOTOR_vidInit
Syntax	void MOTOR_vidInit(void)
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters	None
Return	None
Description	Initialize the Motors pin used and PWM initialization
Available Via	Motor.h

6.8.2.2

Func Name	MOTOR_vidForward
Syntax	void MOTOR_vidForward(void)
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters	None
Return	None
Description	set the pins to move the 2 motors forward and the duty cycle for each motor
Available Via	Motor.h

6.8.2.3

Func Name	MOTOR_vidBackward
Syntax	void MOTOR_vidBackward(void)
Sync/Async	Synchronous
Reentrancy	Non Reentrant

Parameters	None
Return	None
Description	set the pins to move the 2 motors backward and the duty cycle for each motor
Available Via	Motor.h

6.8.2.4

Func Name	MOTOR_vidTurnRight
Syntax	void MOTOR_vidTurnRight(void)
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters	None
Return	None
Description	set the pins to move one motor forward and the other motor backward to Turn right and set the duty cycle for each motor
Available Via	Motor.h

6.8.2.5

Func Name	MOTOR_vidTurnLeft
Syntax	void MOTOR_vidTurnLeft(void)
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters	None
Return	None
Description	set the pins to move one motor forward and the other motor backward to Turn left and set the duty cycle for each motor
Available Via	Motor.h

6.8.2.6

Func Name	MOTOR_vidStop
Syntax	void MOTOR_vidStop(void)
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters	None
Return	None
Description	set the pins to stop the 2 motors and the duty cycle for each motor to stop
Available Via	Motor.h

6.8.2.7

Func Name	MOTOR_vidSetSpeed	
Syntax	void MOTOR_vidSetSpeed(Motors motor, uint16_t value)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters	motor	Motor number (Motor1 - Motor2)
	value	Duty Cycle Value (0 - 100)
Return	None	
Description	set the motors speed using PWM	
Available Via	Motor.h	

6.9 LDR Module

6.9.1 Function Definitions

6.9.1.1

Func Name	LDR_vidInit
Syntax	void LDR_vidInit (void)
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters	None
Return	None
Description	Initialize Both LDRs Sensors
Available Via	LDR.h

6.9.1.2

Func Name	LDR_vidGetLeftBright	
Syntax	void LDR_vidGetLeftBright (uint16_t *Loc_u16Read)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters	Loc_u16Read	To carry Left LDR Value
Return	None	
Description	Get Left LDR sensor Value	
Available Via	LDR.h	

6.9.1.3

Func Name	LDR_vidGetRightBright	
Syntax	void LDR_vidGetRightBright (uint16_t *Loc_u16Read)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters	Loc_u16Read	To carry Right LDR Value
Return	None	
Description	Get Right LDR sensor Value	
Available Via	LDR.h	

6.10 Ultrasonic Module

6.10.1 Function Definitions

6.10.1.1

Func Name	Ultrasonic_vidInit
Syntax	void Ultrasonic_vidInit(void)
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters	None
Return	None
Description	Initialize The Ultrasonic Module
Available Via	Ultrasonic.h

6.10.1.2

Func Name	Ultrasonic_vidGetDistance
Syntax	void Ultrasonic_vidGetDistance(void)
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters	None
Return	None
Description	Initiate Ultra-sonic Distance Reading
Available Via	Ultrasonic.h

6.11 Scheduler Module

6.11.1 Type Definitions

6.11.1.1

Name	Task	
Kind	structure	
Elements	void (*TaskHandler)(void)	
	Type	Pointer to function
	Comment	Points to a function of task to be handled
	Period	
	Type	uint16_t
	Comment	Set the periodicity of the task
Description	acts as a task control block	
Available	scheduler.h	

6.11.2 Function Definitions

6.11.2.1

Func Name	OS_vidInit
Syntax	void OS_vidInit(Void)
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters	None
Return	None
Description	set the tick time of the scheduler using the systick timer and set the callback shall be handled every tick
Available Via	scheduler.h

6.11.2.2

Func Name	create_task	
Syntax	void create_task(void (*Task)(void), uint16_t ms_periodicity)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters	Task	Function to be handled by the task
	ms_periodicity	Task periodicity
Return	none	
Description	Set Function to be handled by the task and the task periodicity	
Available Via	scheduler.h	

6.11.2.3

Func Name	tasks_schedular
Syntax	void tasks_schedular(void)
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters	None
Return	None
Description	Function that handles the tasks of system
Available Via	scheduler.h

6.12 Button Module

6.12.1 Type Definitions

6.12.1.1

Name	Button_State		
Kind	Enumeration		
Range	Pullup	0x00	Input Pin Mode Pull-Up
	Pulldown	0x01	Input Pin Mode Pull-Down
	Float	0x02	Floating Pin Mode
Description	Different PIN Operation Modes		
Available	Button.h		

6.12.1.2

Name	Button_Val		
Kind	Enumeration		
Range	Button_Low	0x00	Button Pin State LOW
	Button_High	0x01	Button Pin State HIGH
Description	Different Button PIN States		
Available	Button.h		

6.12.2 Function Definitions

6.12.2.1

Func Name	Button_vidInit
Syntax	void Button_vidInit(void)
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters	None
Return	None
Description	Initializes The PIN Connected To A Button
Available Via	Button.h

6.12.2.2

Func Name	Button_vidGetButtonValue	
Syntax	Button_Val Button_vidGetButtonValue(uint8_t * Copy_u8PortId, uint8_t Copy_u8PinNumber)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters	Copy_u8PortId	Port that button is associated with
	Copy_u8PinNumber	PIN that button is connected to
Return	Button_Val	
Description	Gets current PIN state	
Available Via	Button.h	