AVOID OBSTACLE CAR PROJECT

Team Member:

1. Ahmed Ayman.
2. Aly Hasan.
3. Ahmed Hassan.
4. Micheal Onsy.

Supervisor :

Eng \ Mohamed Abdelhai.

|  |  |
| --- | --- |
| Contents | page |
| 1- Project Description | 1 |
| 2- Requirements Hardware | 1 |
| 3- Requirements Software | 1 |
| 4- Static Design | 2 |
| * Layered Architecture | 2 |
| * Layer Modules | 2 |
| * Modules APIs | 2 |
| 5- Flow Charts | 3 |
| * System Overview | 3 |
| * System in Detailed | 4 |
| 6- Project Demo Videos | 4 |

# Project Description

This project is to design and implement an autonomous car to avoid collision less than 5 cm. The autonomous car consists of 4 DC Motors, Ultrasonic Sensor, LD239, Battery based on ATMEGA32 Miro-controller. The car can change its speed depends on the obstacle distance.

# Requirements Hardware

1. Atmega32 ECU.
2. One 16x2 LCD.
3. DC Motors (4 Motors).
4. One L298N Motor Driver.
5. One On/Off Switch.
6. One Ultrasonic Sensor.

# Requirements Software

1. Ultrasonic sensor added to the Robot to detect the obstacles.
2. If there were no obstacles detected the Robot moves forward with 80% of its max speed.
3. If there is an object detected 50 cm distant from the Robot then the Robot should slow down to 30% of its maximum speed.
4. If there is an object detected 30 cm distant from the Robot then the Robot will stop then turn right and continue moving under the same distance and speed criteria.
5. If there is an object detected less than 30 cm distant from the Robot then the Robot will stop then moves backward until the distance is 30 cm then stop and turn right and continue moving under the same distance and speed criteria.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| APP Control | | Steering | |  |
| MOTOR | SW.PWM | Ultra Sonic | | Library  And  Utils |
| SW.ICU | |
| DIO | TIMER | | External  Interrupt |  |
| Micro-Controller | | | | |

## Layer Modules

* MCAL Layer Modules:

1. DIO Module.
2. External Interrupt.
3. Timer Module.

* HAL Layer Module:

1. MOTOR Module.
2. UltraSonic Module.
3. SW.PWM
4. SW.ICU

* APP Layer Module:

1. Steering Module.
2. APP Control Module.

## Modules APIs

1. MCAL:
2. DIO Modules APIs.

void HAL\_GPIO\_Pin\_Init(StrGPIO\_t PORT, GPIO\_InitTypeDef \* PIN\_CONFIG);

GPIO\_PinState HAL\_GPIO\_READPIN(StrGPIO\_t PORT,uint8\_t PIN\_NUM);

void HAL\_GPIO\_WRITEPIN(StrGPIO\_t PORT,uint8\_t PIN\_NUM,GPIO\_PinState PIN\_STATE);

void HAL\_GPIO\_TOGGLE\_PIN(StrGPIO\_t PORT,uint8\_t PIN\_NUM);

void HAL\_GPIO\_WRITEPORT(StrGPIO\_t PORT,uint8\_t PINS,GPIO\_PinState PINS\_STATE);

typedef struct { uint8\_t PIN\_REG;

uint8\_t DDR\_REG;

uint8\_t PORT\_REG;

}GPIO\_TypeDef;

1. External Interrupt.

void EXTI\_Init(EXIT\_Handler\_t \* Handler);

void EXIT\_INT0\_CallBack(PtrFun PtrToFun);

void EXIT\_INT1\_CallBack(PtrFun PtrToFun);

typedef enum { EXTI\_NUM\_0 , EXTI\_NUM\_1 , EXTI\_NUM\_2 } EXIT\_Select\_t ;

typedef enum { EXTI\_EDGE\_LOW\_LENEL , EXTI\_EDGE\_ANY\_LENEL, EXTI\_EDGE\_FAILING\_EDGE , EXTI\_EDGE\_RISING\_EDGE } LevelSelect\_t;

typedef struct{ EXIT\_Select\_t EXTI\_NUM ;

LevelSelect\_t EXTI\_EDGE\_DETECTION ;

}EXIT\_Handler\_t;

3-Timer:

TIM\_Status\_t TIM\_NormalModeInit(TIMInit\_t \* TIMConfig );

TIM\_Status\_t TIM\_OutCompareModeInit(TIMInit\_t \* TIMConfig );

TIM\_Status\_t TIM\_DeInit ( TIM\_Instance\_t TIM\_Instance );

TIM\_Status\_t TIM\_PWMMode\_SetDuty(TIMInit\_t \* TIMConfig, uint8\_t DutyCycle);

TIM\_Status\_t TIM\_InputCaptureModeInit( uint8\_t Edge );

TIM\_Status\_t TIM\_SetValue( TIM\_Instance\_t Instance ,uint8\_t CountVal);

TIM\_Status\_t TIM\_GetValue( TIM\_Instance\_t Instance ,uint8\_t \* CountVal);

TIM\_Status\_t TIM\_Start(TIMInit\_t \* TIMConfig );

TIM\_Status\_t TIM\_Stop(TIM\_Instance\_t TIM\_Instance);

TIM\_Status\_t TIM\_PWMModeInit(TIMInit\_t \* TIMConfig );

TIM\_Status\_t TIM\_CallBack\_FuctionSet(IT\_SelBIT\_t Interrupt\_Num , TIMCaLL\_BackFun callbackfunction);

Utilies\_Status\_t Utilites\_DelayMs\_IT(TIMInit\_t \* Tim\_Handler ,uint16\_t MsDelay , DalayType\_t Dalay\_type , TIMCaLL\_BackFun callbackfunction);

Utilies\_Status\_t Utilites\_DelayUs(uint8\_t TimerInstance , uint16\_t UsDelay);

Utilies\_Status\_t Software\_PWM\_Init( StrGPIO\_t PORT , uint8\_t PIN\_Num ,TIMInit\_t \* Tim\_PWM\_Handler );

Utilies\_Status\_t Software\_PWM\_Start (TIMInit\_t \* TIMConfig );

Utilies\_Status\_t Software\_PWM\_UpdateDuty( uint8\_t SetDuty ,TIMInit\_t \* Tim\_PWM\_Handler );

Utilies\_Status\_t Software\_PWM\_Stop( TIMInit\_t \* Tim\_PWM\_Handler );

typedef enum{ TIM\_IT\_DIS , TIM\_0\_IT\_OVER =0x01, TIM\_0\_IT\_COMP =0x02

,TIM\_1\_IT\_OVER =0x04,TIM\_1\_IT\_COMPB=0x08, TIM\_1\_IT\_COMPA =0x10 ,TIM\_1\_IT\_CAPT =0x20,TIM\_2\_IT\_OVER =0x40 ,TIM\_2\_IT\_COMP =0x80 }IT\_SelBIT\_t ;

typedef union {struct {

uint8\_t CompAction ;

uint8\_t CompValue ;

uint8\_t CompNum ;

}TIM16Bit;

struct {

uint8\_t CompAction ;

uint8\_t CompValue ;

}TIM8Bit;

}TIM\_COMPConfig\_t;

typedef struct{

uint32\_t TIMMode ; TIM\_COMPConfig\_t COMPConfig ; TIM1\_Prescaller\_t TimPreScaler ; TIM\_Instance\_t Instance ; uint8\_t TIM\_Interrupt ; }TIMInit\_t;

typedef enum{Delay\_Periodic , Delay\_Once }DalayType\_t;

typedef enum { UTIL\_OK =0 , UTIL\_PARAM\_ERROR , UTIL\_TIM\_ERROR } Utilies\_Status\_t ;

typedef struct {

TIMCaLL\_BackFun Delaycallbackfunction ; uint16\_t MsDelay ; DalayType\_t DelayAttr ; }DelayConfig\_t;

1. Hal:
2. UltraSonic:

Ultrs\_Status\_t Ultrasonic\_Init( Ultrasonic\_GPIOPINS\_t \* Ultrasonic\_PINS )Ultrs\_Status\_t Ultrasonic\_GetDistance(float \* Distance , Ultrasonic\_GPIOPINS\_t \* Ultrasonic\_PINS );

typedef uint8\_t PIN\_TypeDef ;

typedef enum {Ultrasonic\_OK , Ultrasonic\_PARAM\_ERROR , Ultrasonic\_CONFIG\_ERROR }Ultrs\_Status\_t;

typedef struct { GPIO\_TypeDef \* PORT ;PIN\_TypeDef PINNum ;}Ultrasonic\_pinConfig\_t;

typedef struct { Ultrasonic\_pinConfig\_t ECO\_PIN ;Ultrasonic\_pinConfig\_t TRIGGER\_PIN ;

}Ultrasonic\_GPIOPINS\_t;

1. Motor:

MOTOR\_STATUS\_t Motor\_Init(void);

MOTOR\_STATUS\_t Motor\_Dir(Motor\_DIR\_t DIR , uint8\_t Speed );

typedef struct { StrGPIO\_t Port[2] ;

uint8\_t Pin[2] ;

}Motor\_Pins;

typedef struct { Motor\_Pins Motor[Total\_MOTORS]; }MotorSelect\_t ;

typedef enum { MOTOR\_OK , MOTOR\_PARAM\_ERROR , MOTOR\_CONFIG\_ERROR , MOTOR\_PWM\_ERROR }MOTOR\_STATUS\_t;

typedef enum {DIR\_LEFT , DIR\_RIGHT , DIR\_FORWARD , DIR\_BACKWARD , DID\_STOP }Motor\_DIR\_t;

1. Application:
2. App :

void APP\_Init(void);

void APP\_UPdate(void);

## Flow Chart:

