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

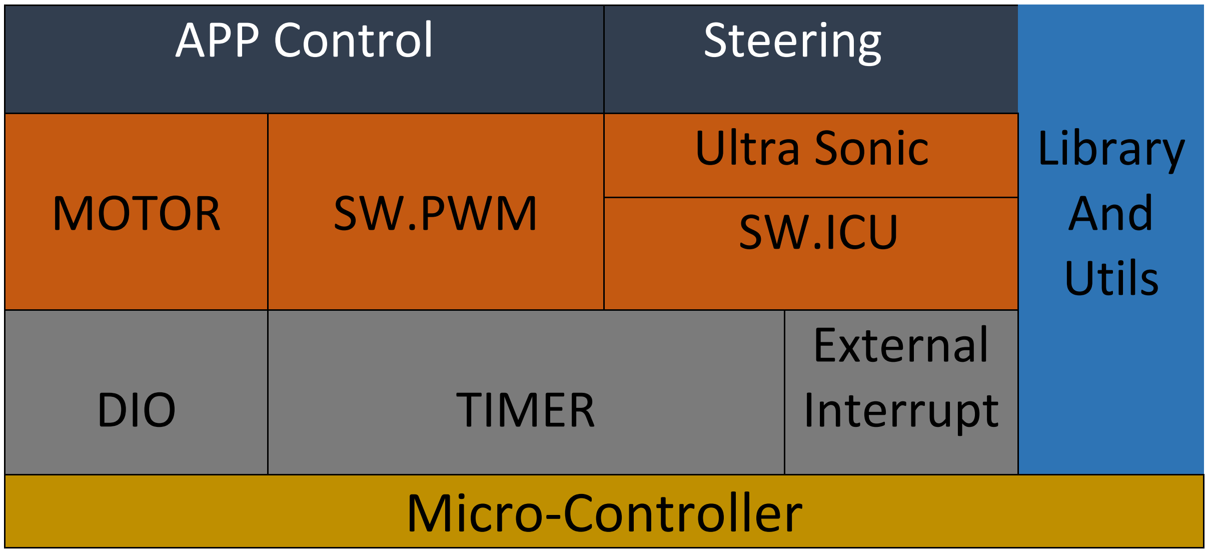
1- Ultrasonic sensor added to the Robot to detect the obstacles.

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.



- 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

A- MCAL:

1-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;
```

2-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;
```

B- Hal:

1- 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;

2- 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;

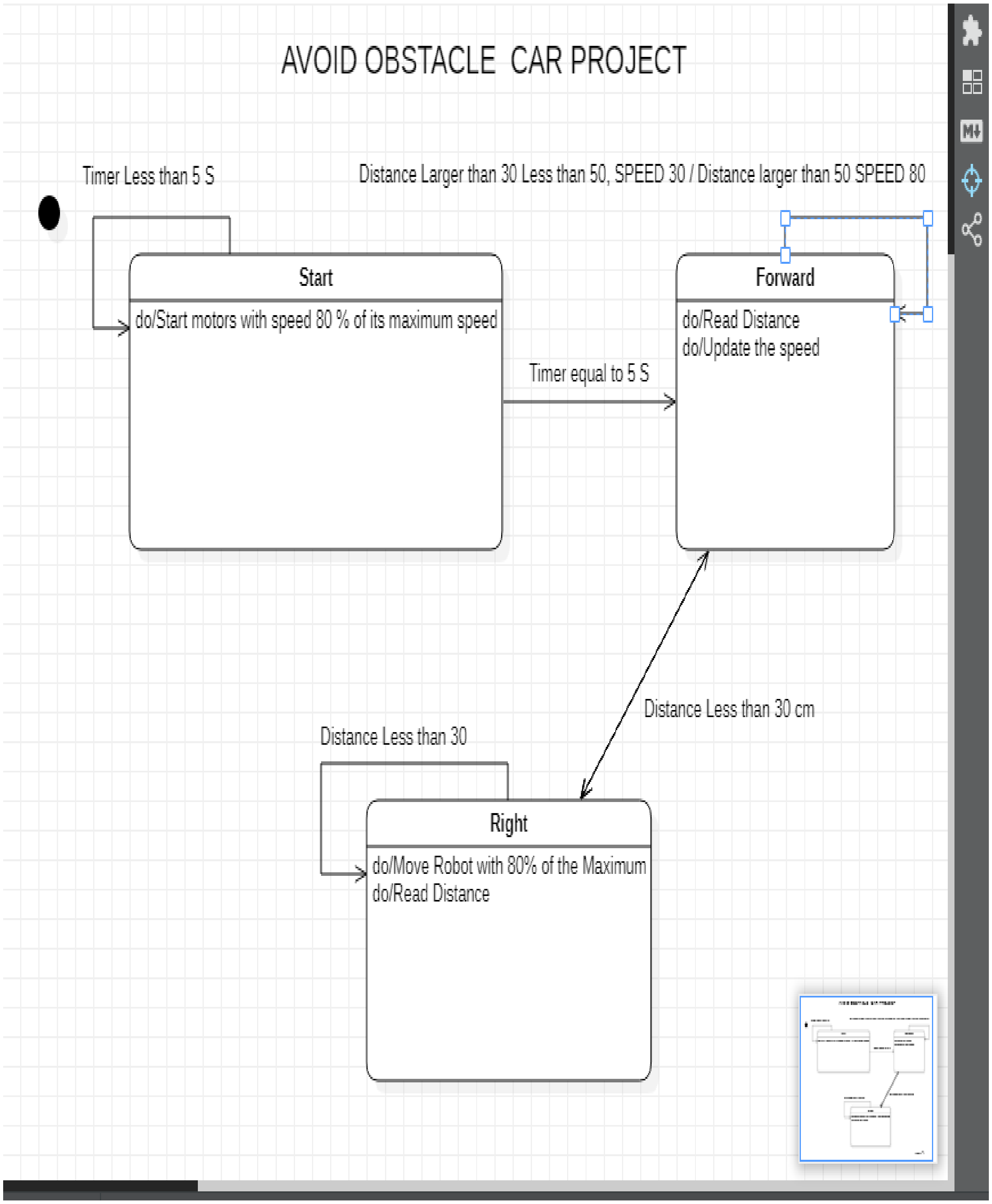
typedef enum {DIR_LEFT , DIR_RIGHT , DIR_FORWARD , DIR_BACKWARD , DID_STOP }Motor_DIR_t;

C- Application:

1- App :

void APP_Init(void);
void APP_UPdate(void);

- Flow Chart:


AVOID OBSTACLE CAR PROJECT
Timer Less than 5 S
Distance Larger than 30 Less than 50, SPEED 30 / Distance larger than 50 SPEED 80
Start
do/Start motors with speed 80 % of its maximum speed
Forward
do/Read Distance
do/Update the speed
Timer equal to 5 S
Distance Less than 30 cm
Distance Less than 30
Right
do/Move Robot with 80% of the Maximum
do/Read Distance