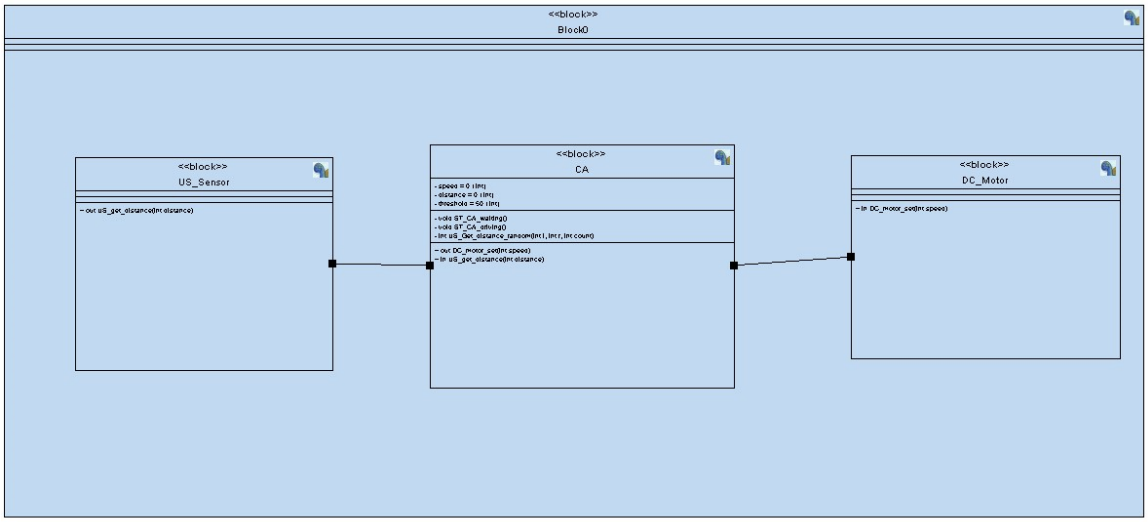
Simple
Collision Avoidance System

Author: Hassan Attia

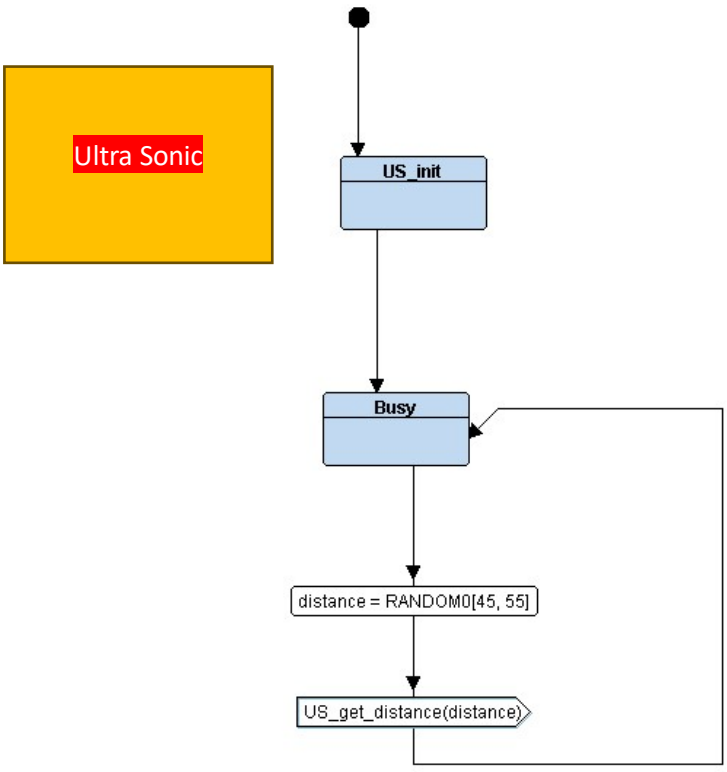
Brief:

- This report illustrates design modules & basic implementation of collision avoidance.
- Figures that will be shown are:
    - Modules Level.
    - Logical Design (State Machine).
    - Design Verification and application output.
    - C code implementation for each module.

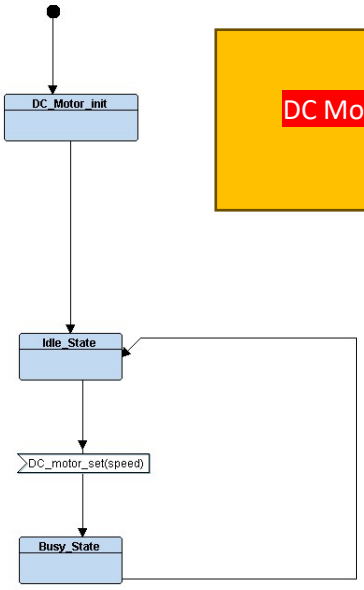
1.Modules Level:



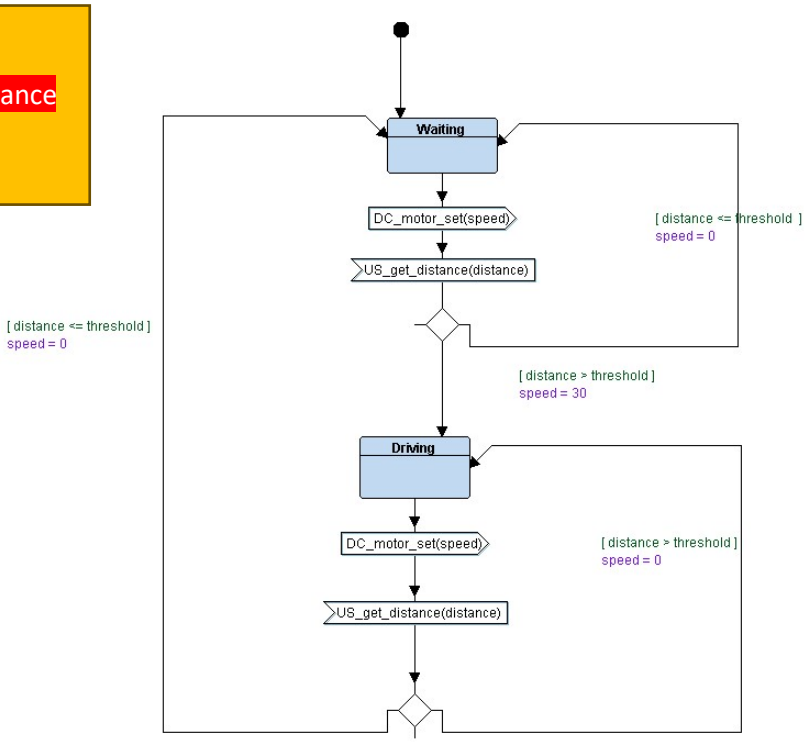
2.Logical Design:



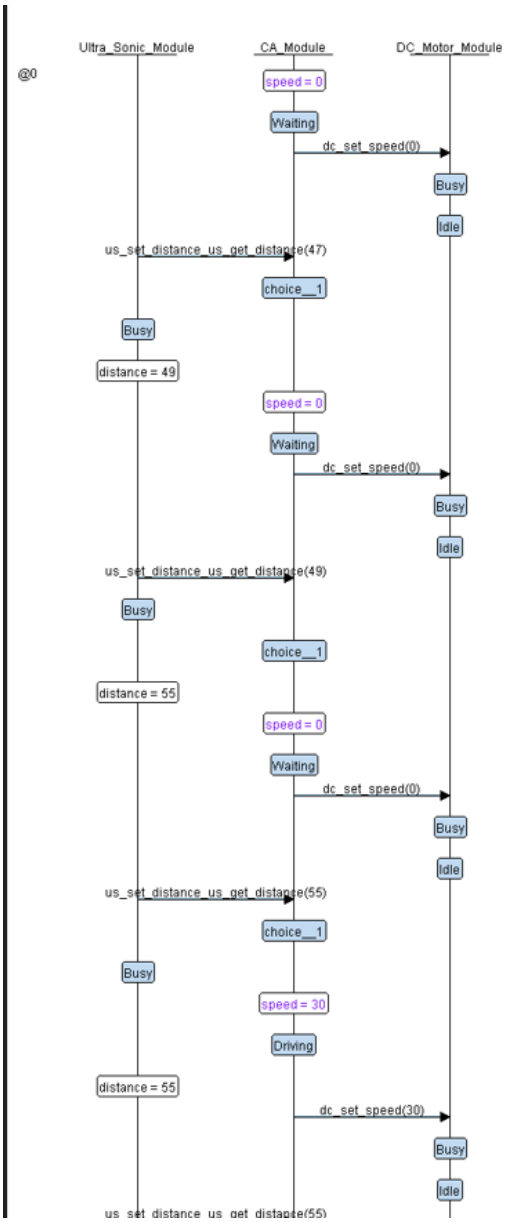
DC Motor



Collision Avoidance



3.Design Verification and app output:




```
US----------------distance = 55----------------->CA

CA_Driving State: distance = 55   Speed = 0

CA----------------speed = 30----------------->DC

DC_busy state:  speed = 30

US----------------distance = 46----------------->CA

CA_Waiting State: distance = 46   Speed = 30

CA----------------speed = 0----------------->DC

DC_busy state:  speed = 0

US----------------distance = 45----------------->CA

CA_Waiting State: distance = 45   Speed = 0

CA----------------speed = 0----------------->DC

DC_busy state:  speed = 0

US----------------distance = 52----------------->CA

CA_Driving State: distance = 52   Speed = 0

CA----------------speed = 30----------------->DC

DC_busy state:  speed = 30

US----------------distance = 52----------------->CA

CA_Driving State: distance = 52   Speed = 30

CA----------------speed = 30----------------->DC

DC_busy state:  speed = 30

US----------------distance = 50----------------->CA

CA_Waiting State: distance = 50   Speed = 30

CA----------------speed = 0----------------->DC

DC_busy state:  speed = 0
```

4.Code implementation:

Ultra Sonic Source File

```c
#include "US.h"


 void(*P_US_STATE)();

//Variables
int US_distance = 0;



//function definition

void US_init(){

//init US Driver

printf("\n US_init...\n");


}


STATE_define(US_Busy){

//STATE NAME
US_state = US_busy;
//STATE action
//Ultra sonic readings
US_distance = US_Get_distance_random(45,55,1);

Us_set_distance(US_distance);
P_US_STATE = STATE(US_Busy);



}




int US_Get_distance_random (int l, int r, int count){

//this will generate numbers from l to r range


int i;

for (i  = 0; i< count; i++){

    int rand_num = ( rand() % (r - l + 1) ) + l;
    return rand_num ;


}
return 0;
}
```

Ultra Sonic Header File

```c
#ifndef US_H_
#define US_H_

//Libraries
#include "state.h"


//Ultra Sonic States enumeration

enum{

US_busy


}US_state;


//Ultra Sonic init prototype
void US_init();

//Ultra sonic State Functions prototype

STATE_define(US_Busy);

//Pointer to function
extern void(*P_US_STATE)();


#endif // US_H_
```

DC Motor Source File

```c
#include "DC_Motor.h"

void(*P_DC_Motor_STATE)();

 //variables
 int DC_speed =0;



  //DC motor init

void DC_MOTOR__init(){

//init US Driver

printf("\n DC Motor_init...\n");


}


//Interface between CA & DC motor
void DC_get_speed(int speed){
    DC_speed = speed;
    P_DC_Motor_STATE = STATE(DC_MOTOR_Busy);
printf("\nCA----------------speed = %d---------------->DC\n",DC_speed);
}



STATE_define(DC_MOTOR_Idle){


//STATE NAME
DC_MOTOR_state = DC_idle;


//STATE Action
//Action : change the speed via PWM

printf("\nDC_idle state: speed = %d\n",DC_speed);

}


STATE_define(DC_MOTOR_Busy){


//STATE NAME
DC_MOTOR_state = DC_busy;


//STATE Action
//Action : change the speed via PWM

P_DC_Motor_STATE = STATE(DC_MOTOR_Idle);

printf("\nDC_busy state:  speed = %d \n", DC_speed);
```

DC Motor Header File

```c
#ifndef DC_MOTOR_
#define DC_MOTOR_


//Libraries
#include "state.h"




//DC Motor States enumeration

enum{

DC_idle,
DC_busy


}DC_MOTOR_state;

//Pointer to function
 extern void(*P_DC_Motor_STATE)();

//DC Motor initialization
void DC_init();

//DC Motor State Functions
STATE_define(DC_MOTOR_Idle);
STATE_define(DC_MOTOR_Busy);


#endif // DC_MOTOR_
```

```c
#include "CA.h"

//Pointer to function
  void(*P_CA_STATE)();


//variable
int CA_distance = 0;
int CA_speed = 0;
int CA_threshold = 50;


void Us_set_distance(int distance){

CA_distance = distance;

(CA_distance <= CA_threshold)?  (P_CA_STATE = STATE(CA_Waiting) ):(P_CA_STATE = STATE(CA_Driving));
printf("\nUS----------------distance = %d---------------->CA\n",CA_distance);


}




STATE_define(CA_Waiting){

//STATE NAME
CA_state = CA_waiting;
printf("\nCA_Waiting State: distance = %d   Speed = %d\n",CA_distance,CA_speed);



//STATE Action
 CA_speed = 0;
 DC_get_speed(CA_speed);


}


STATE_define(CA_Driving){

//STATE NAME
CA_state = CA_driving;
printf("\nCA_Driving State: distance = %d   Speed = %d\n",CA_distance,CA_speed);
//STATE Action
CA_speed = 30;
DC_get_speed(CA_speed);


}
```

Collision Avoidance Source File

```c
#ifndef CA_H_
#define CA_H_

//Libraries
#include "state.h"



//collision avoidance States enumeration

enum{

CA_waiting,
CA_driving


}CA_state;

//Collision Avoidance init prototype
void CA_init();


//Collision Avoidance STATE function
STATE_define(CA_Waiting);
STATE_define(CA_Driving);

//Pointer to function
 extern void(*P_CA_STATE)();

#endif // CA_H_
```

Collision Avoidance Header File

```c
#ifndef STATE_H_
#define STATE_H_


 //Libraries
#include "state.h"
#include "stdio.h"
#include "stdlib.h"



//State Function Declaration Macros
#define STATE_define(_FUNCTION_) void STATE_##_FUNCTION_()
#define STATE(_FUNCTION_) STATE_##_FUNCTION_



//useful functions
int US_Get_distance_random (int l, int r, int count);

//State Connections wires(interface)
//these connections are the interface between each module and the other
//Definitions are in destination i.e CA module & DC module
void Us_set_distance(int distance);// OUT: ultra sonic  ... IN: collision avoidance
void DC_get_speed(int speed);//OUT: collision avoidance ... IN: DC motor

#endif // STATE_H_
```

A header file contains some useful macros, functions interface between each module.