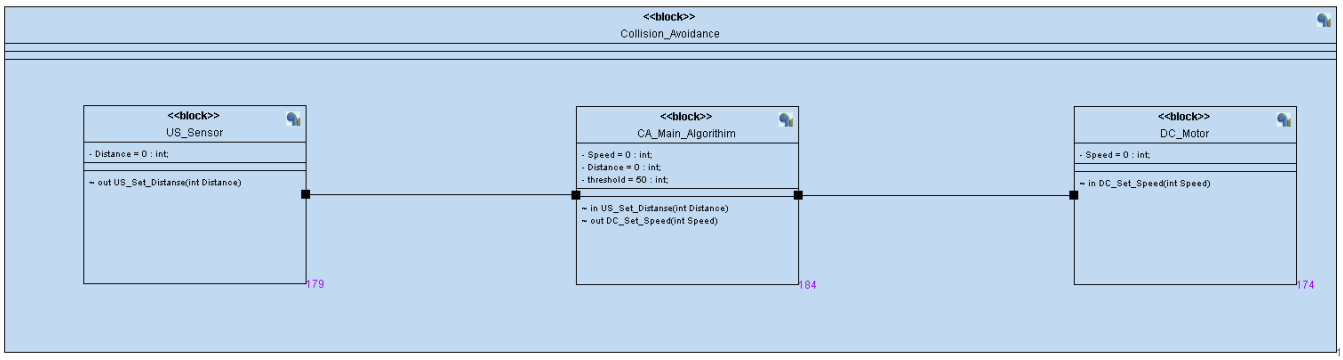
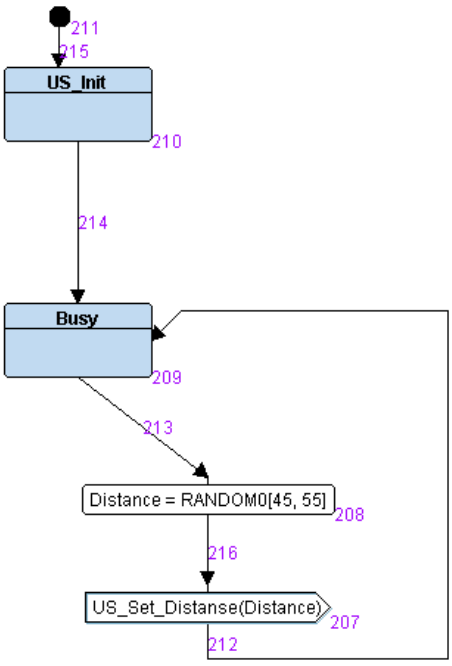
Simple state machine Implementation of ULTRASONIC OBSTACLE AVOIDING robot in C using multiple modules

Modules level:

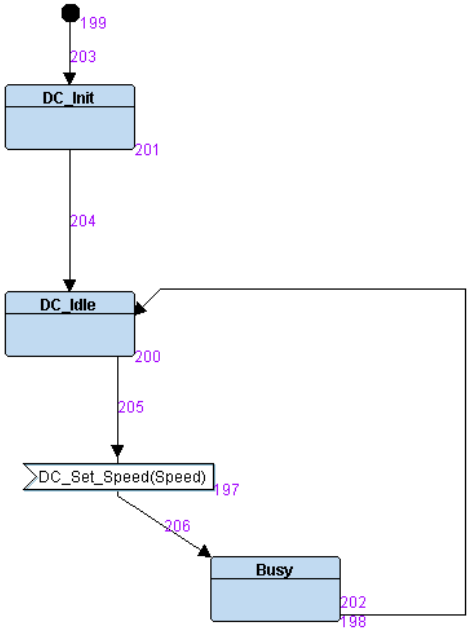


Logical design:

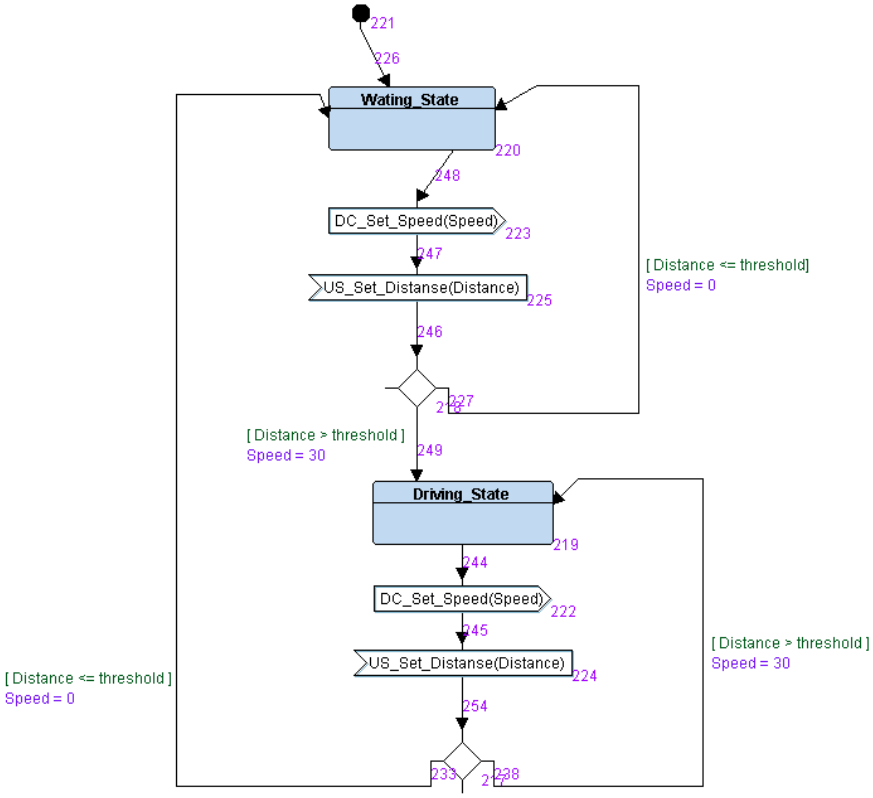
1.Ultrasonic sensor module:



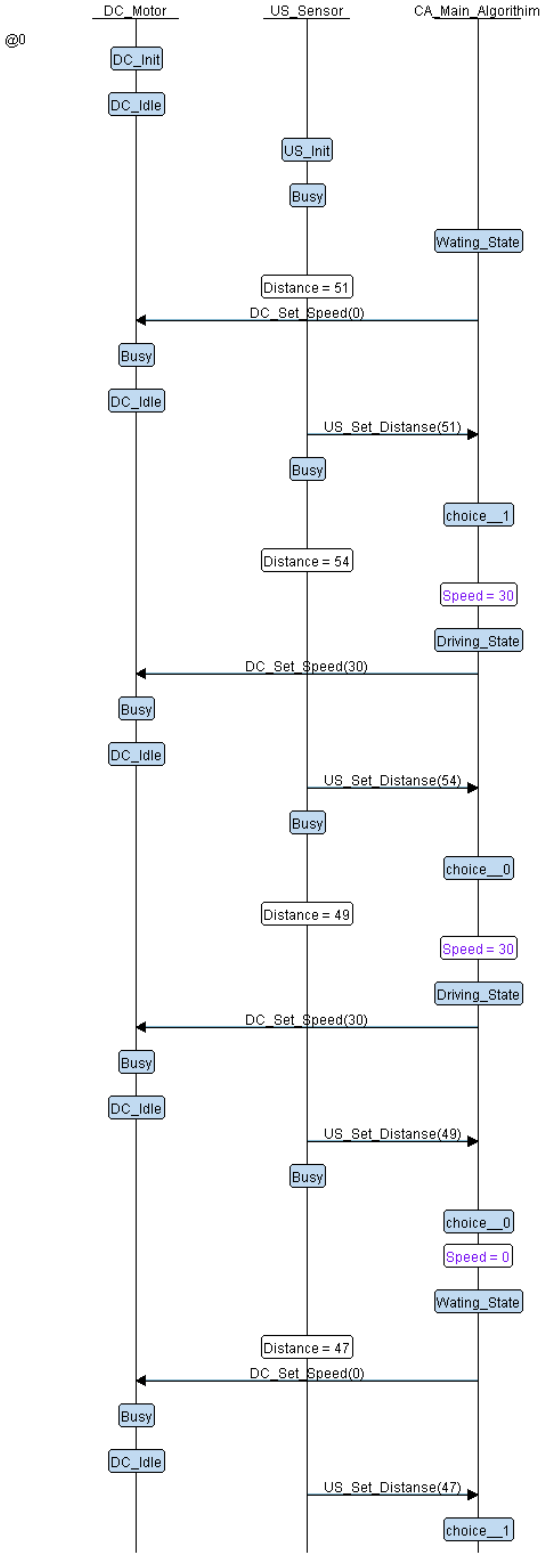
2.DC motor module:



3.Collision avoidance module:

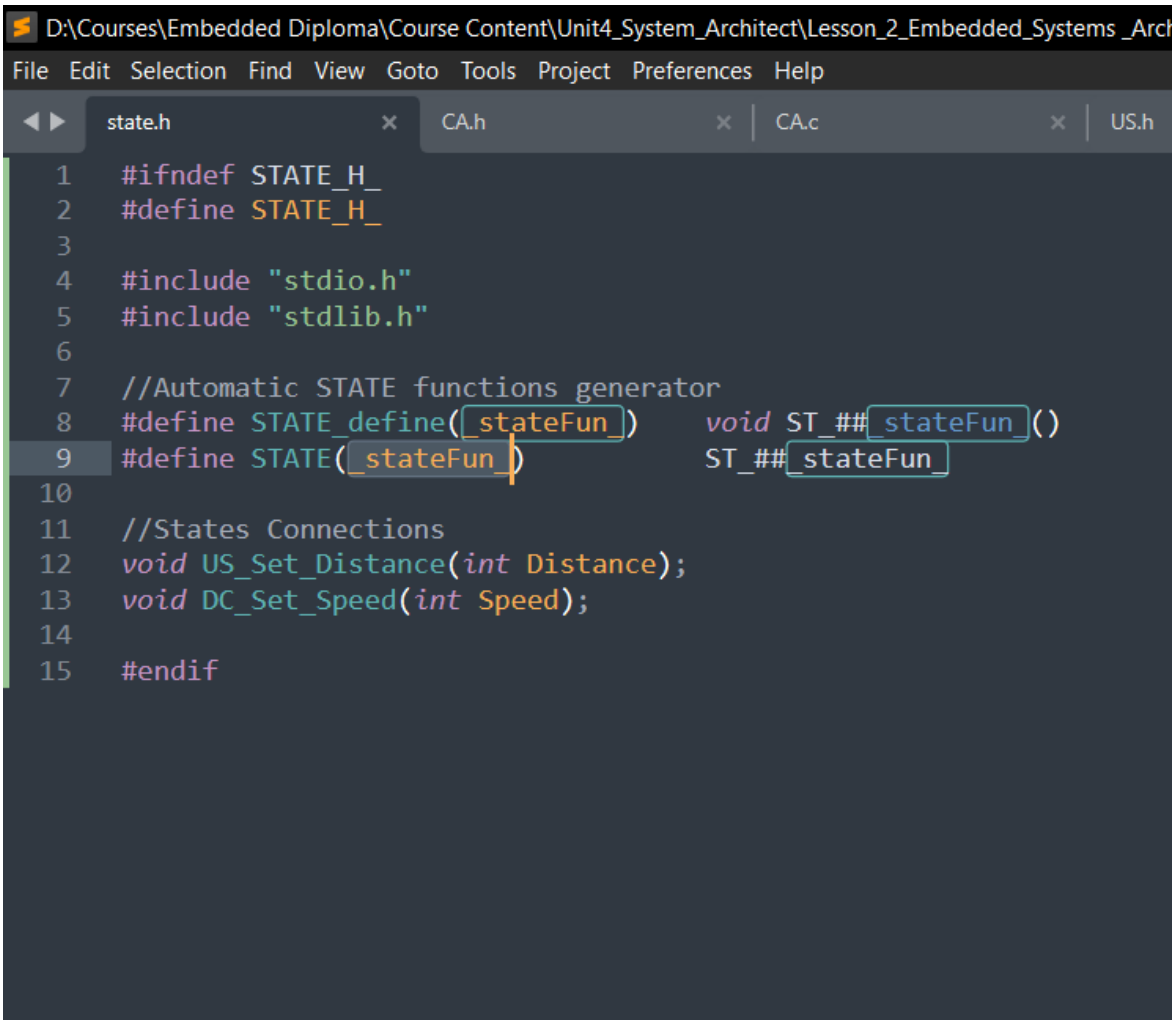


Simulation:

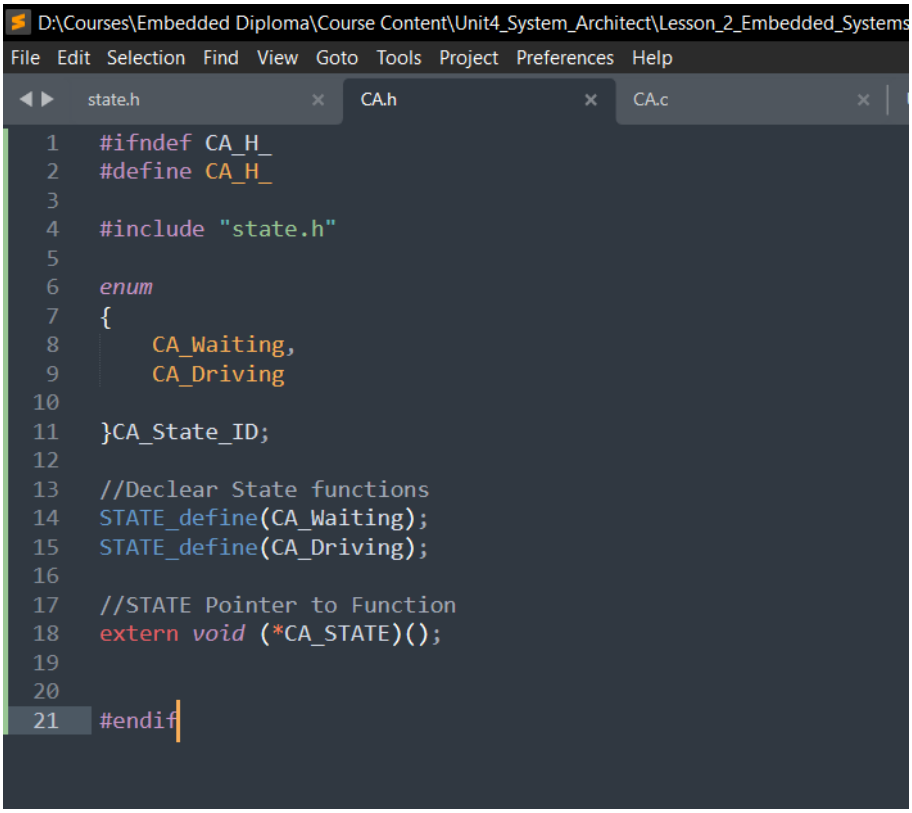

DC_Motor
US_Sensor
CA_Main_Algorithim
@0
DC_Init
DC_Idle
US_Init
Busy
Wating_State
Distance = 51
DC_Set_Speed(0)
Busy
DC_Idle
US_Set_Distanse(51)
Busy
choice__1
Distance = 54
Speed = 30
Driving_State
DC_Set_Speed(30)
Busy
DC_Idle
US_Set_Distanse(54)
Busy
choice__0
Distance = 49
Speed = 30
Driving_State
DC_Set_Speed(30)
Busy
DC_Idle
US_Set_Distanse(49)
Busy
choice__0
Speed = 0
Wating_State
Distance = 47
DC_Set_Speed(0)
Busy
DC_Idle
US_Set_Distanse(47)
choice__1

C implementation:

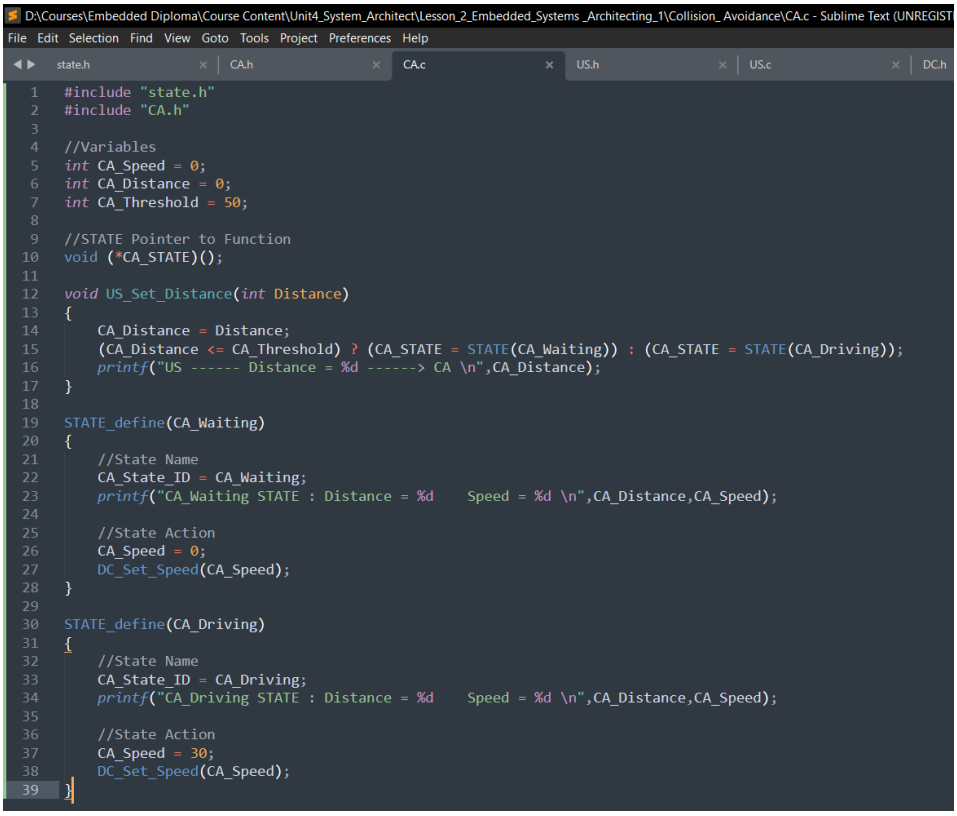
- State.h



- CA.h


D:\Courses\Embedded Diploma\Course Content\Unit4_System_Architect\Lesson_2_Embedded_Systems
File Edit Selection Find View Goto Tools Project Preferences Help
state.h
CA.h
CA.c
#ifndef CA_H_
#define CA_H_
#include "state.h"
enum
{
CA_Waiting,
CA_Driving
}CA_State_ID;
//Declear State functions
STATE_define(CA_Waiting);
STATE_define(CA_Driving);
//STATE Pointer to Function
extern void (*CA_STATE)();
#endif

- CA.c


D:\Courses\Embedded Diploma\Course Content\Unit4_System_Architect\Lesson_2_Embedded_Systems _Architecting_1\Collision_ Avoidance\CA.c - Sublime Text (UNREGIST
File Edit Selection Find View Goto Tools Project Preferences Help
state.h
CA.h
CA.c
US.h
US.c
DC.h
#include "state.h"
#include "CA.h"
//Variables
int CA_Speed = 0;
int CA_Distance = 0;
int CA_Threshold = 50;
//STATE Pointer to Function
void (*CA_STATE)();
void US_Set_Distance(int Distance)
{
CA_Distance = Distance;
(CA_Distance <= CA_Threshold) ? (CA_STATE = STATE(CA_Waiting)) : (CA_STATE = STATE(CA_Driving));
printf("US ------ Distance = %d ------> CA \n",CA_Distance);
}
STATE_define(CA_Waiting)
{
//State Name
CA_State_ID = CA_Waiting;
printf("CA_Waiting STATE : Distance = %d Speed = %d \n",CA_Distance,CA_Speed);
//State Action
CA_Speed = 0;
DC_Set_Speed(CA_Speed);
}
STATE_define(CA_Driving)
{
//State Name
CA_State_ID = CA_Driving;
printf("CA_Driving STATE : Distance = %d Speed = %d \n",CA_Distance,CA_Speed);
//State Action
CA_Speed = 30;
DC_Set_Speed(CA_Speed);
}

- US.h

D:\Courses\Embedded Diploma\Course Content\Unit4_System_Architect\Lesson_2_Embedded_Systems _Ar

File Edit Selection Find View Goto Tools Project Preferences Help

state.h | CA.h | CA.c | US.h

```h
#ifndef US_H_
#define US_H_

#include "state.h"

enum
{
    US_Busy,

}US_State_ID;

//Declear State functions
STATE_define(US_Busy);

void US_Init();
int US_Get_Random_Distance(int start , int end , int count);

//STATE Pointer to Function
extern void (*US_STATE)();

#endif
```

- US.c

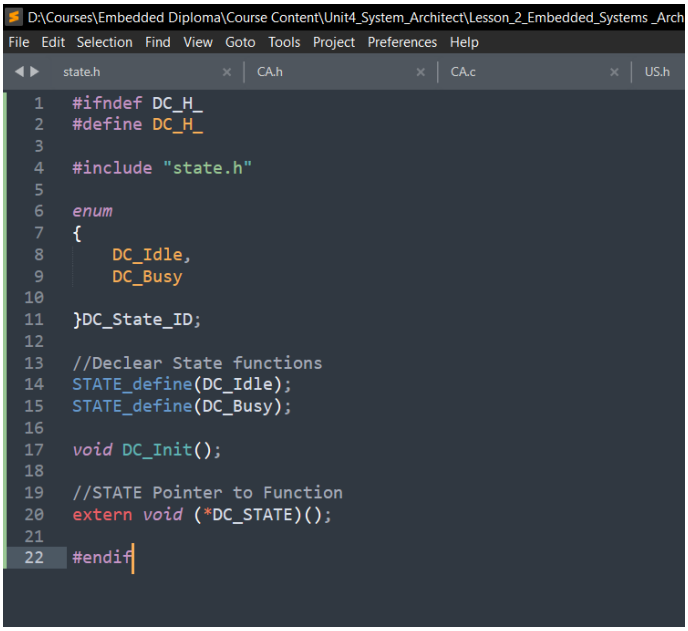
D:\Courses\Embedded Diploma\Course Content\Unit4_System_Architect\Lesson_2_Embedded_Systems _Architecting_1\Collision_ Avoidance

File Edit Selection Find View Goto Tools Project Preferences Help

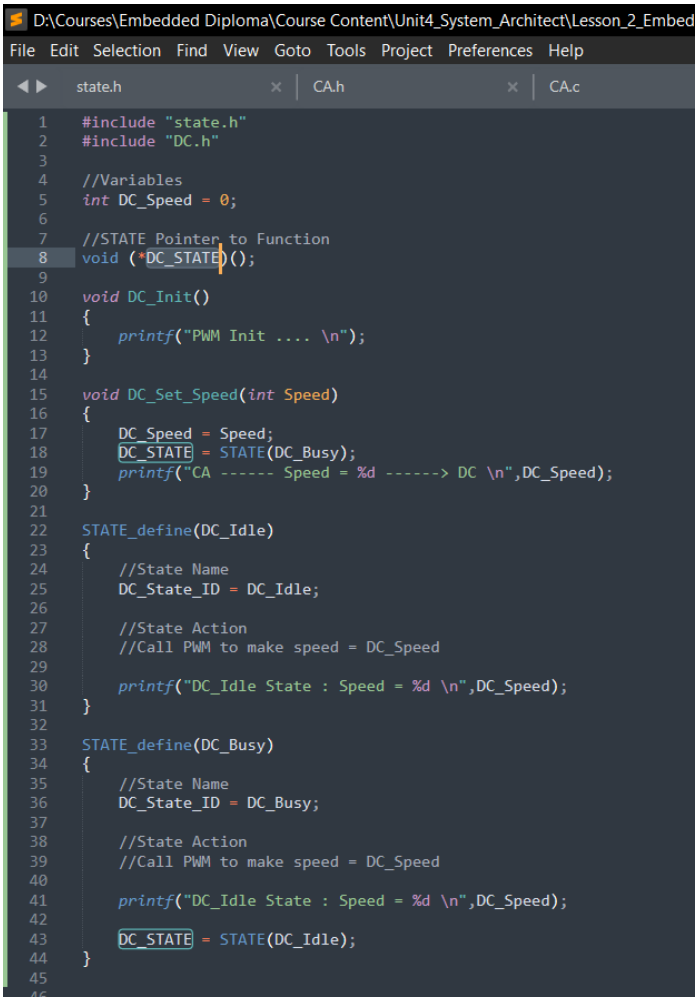
state.h | CA.h | CA.c | US.h | US.c

```c
#include "state.h"
#include "US.h"


//Variables
int US_Distance = 0;


//STATE Pointer to Function
void (*US_STATE)();

void US_Init()
{
    printf("UltraSonic Sensor Init ... \n");
}

STATE_define(US_Busy)
{
    //State Name
    US_State_ID = US_Busy;

    //State Action
    US_Distance = US_Get_Random_Distance(45 , 55 , 1);
    printf("US_Busy STATE : Distance = %d\n",US_Distance);

    //Send the Signal (Distance)
    US_Set_Distance(US_Distance);

    //Next STATE
    US_STATE = STATE(US_Busy);


}

int US_Get_Random_Distance(int start , int end , int count)
{
    int i = 0;

    for(;i < count ; i++)
    {
        int Rand_Num = (rand() % (end - start + 1)) + start;
        return Rand_Num;
    }

    return 0;
}
```

- DC.h


D:\Courses\Embedded Diploma\Course Content\Unit4_System_Architect\Lesson_2_Embedded_Systems _Arch
File Edit Selection Find View Goto Tools Project Preferences Help
state.h
CA.h
CA.c
US.h
#ifndef DC_H_
#define DC_H_
#include "state.h"
enum
{
DC_Idle,
DC_Busy
}DC_State_ID;
//Declear State functions
STATE_define(DC_Idle);
STATE_define(DC_Busy);
void DC_Init();
//STATE Pointer to Function
extern void (*DC_STATE)();
#endif

- DC.c


D:\Courses\Embedded Diploma\Course Content\Unit4_System_Architect\Lesson_2_Embed
File Edit Selection Find View Goto Tools Project Preferences Help
state.h
CA.h
CA.c
#include "state.h"
#include "DC.h"
//Variables
int DC_Speed = 0;
//STATE Pointer to Function
void (*DC_STATE)();
void DC_Init()
{
printf("PWM Init .... \n");
}
void DC_Set_Speed(int Speed)
{
DC_Speed = Speed;
DC_STATE = STATE(DC_Busy);
printf("CA ------ Speed = %d ------> DC \n",DC_Speed);
}
STATE_define(DC_Idle)
{
//State Name
DC_State_ID = DC_Idle;
//State Action
//Call PWM to make speed = DC_Speed
printf("DC_Idle State : Speed = %d \n",DC_Speed);
}
STATE_define(DC_Busy)
{
//State Name
DC_State_ID = DC_Busy;
//State Action
//Call PWM to make speed = DC_Speed
printf("DC_Idle State : Speed = %d \n",DC_Speed);
DC_STATE = STATE(DC_Idle);
}

- main.c

D:\Courses\Embedded Diploma\Course Content\Unit4_System_Architect\Lesson_2_Embe

File Edit Selection Find View Goto Tools Project Preferences Help

state.h | CA.h | CA.c

```c
#include "state.h"
#include "CA.h"
#include "US.h"
#include "DC.h"


void setup()
{
    //Init all Blocks
    US_Init();
    DC_Init();

    //Set STATES Pointers for each Block
    CA_STATE = STATE(CA_Waiting);
    US_STATE = STATE(US_Busy);
    DC_STATE = STATE(DC_Idle);
}

void main()
{
    volatile int d ;
    setup();

    while(1)
    {
        //Call state for each Block
        US_STATE();
        CA_STATE();
        DC_STATE();

        //Delay
        for(d = 0 ; d < 1000 ; d++);

    }
}
```

Results:

log.txt - Notepad

File Edit Format View Help

```
UltraSonic Sensor Init ...
PWM Init ....
US_Busy STATE : Distance = 53
US ------ Distance = 53 ------> CA
CA_Driving STATE : Distance = 53        Speed = 0
CA ------ Speed = 30 ------> DC
DC_Idle State : Speed = 30
US_Busy STATE : Distance = 54
US ------ Distance = 54 ------> CA
CA_Driving STATE : Distance = 54        Speed = 30
CA ------ Speed = 30 ------> DC
DC_Idle State : Speed = 30
US_Busy STATE : Distance = 54
US ------ Distance = 54 ------> CA
CA_Driving STATE : Distance = 54        Speed = 30
CA ------ Speed = 30 ------> DC
DC_Idle State : Speed = 30
US_Busy STATE : Distance = 46
US ------ Distance = 46 ------> CA
CA_Waiting STATE : Distance = 46        Speed = 30
CA ------ Speed = 0 ------> DC
DC_Idle State : Speed = 0
US_Busy STATE : Distance = 52
US ------ Distance = 52 ------> CA
CA_Driving STATE : Distance = 52        Speed = 0
CA ------ Speed = 30 ------> DC
DC_Idle State : Speed = 30
US_Busy STATE : Distance = 50
US ------ Distance = 50 ------> CA
CA_Waiting STATE : Distance = 50        Speed = 30
CA ------ Speed = 0 ------> DC
DC_Idle State : Speed = 0
US_Busy STATE : Distance = 50
US ------ Distance = 50 ------> CA
CA_Waiting STATE : Distance = 50        Speed = 0
CA ------ Speed = 0 ------> DC
DC_Idle State : Speed = 0
US_Busy STATE : Distance = 55
US ------ Distance = 55 ------> CA
CA_Driving STATE : Distance = 55        Speed = 0
CA ------ Speed = 30 ------> DC
DC_Idle State : Speed = 30
```