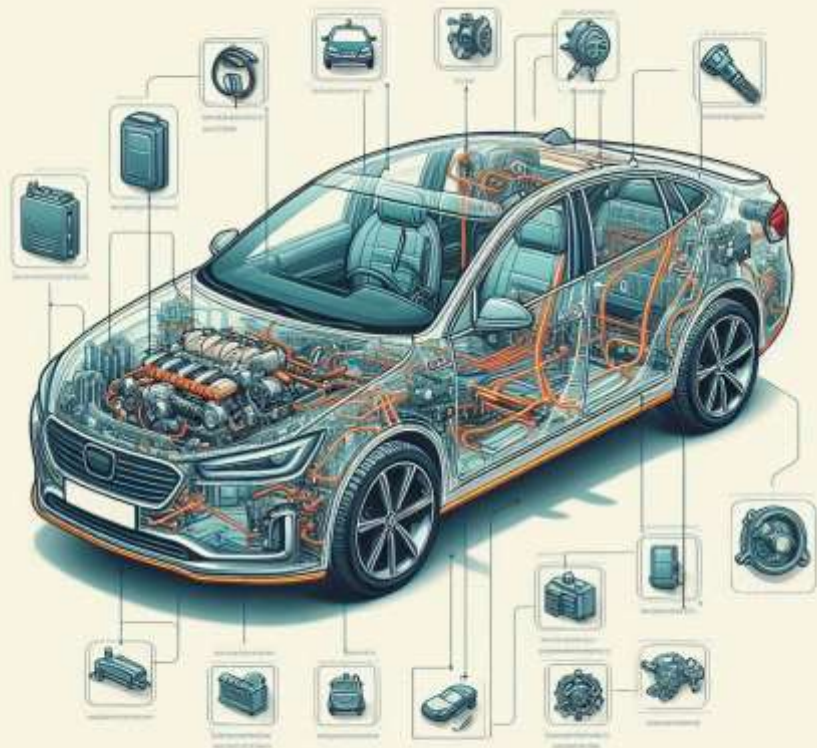


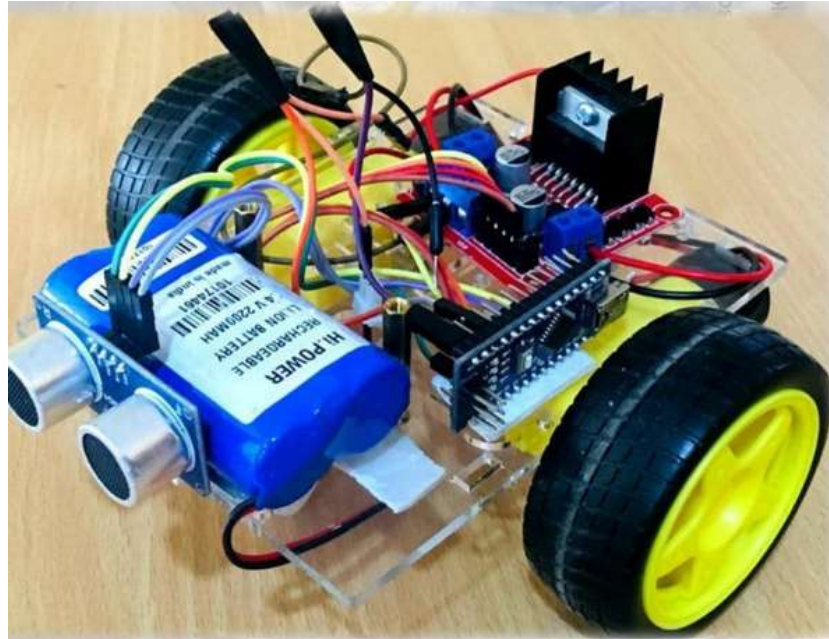
2024

Collision Avoidance



Mohamed Hakeem
Master Embedded System
8/17/2024

1 Case Study:



- Ultrasonic Obstacle-avoiding Robot:

This case study involves building an Ultrasonic Obstacle-Avoiding Robot designed to detect and avoid obstacles. To simplify the process, an ultrasonic sensor will be used to measure the distance between the robot and any obstacles in its path. A DC motor will be used to control the robot's movement, allowing it to either stop or continue moving based on the detected distance.

- Sequence for building software:

1. Design a state diagram with any tool according to the standard between teams and each other.
2. Verify this design using the same tool.
3. Once it is achieved successfully, implement the code of each task that is built on the state diagram with the same template in any IDE.
4. mapping between these tasks by standard function.

- Design a state diagram with any tool according to the standard between teams and each other.

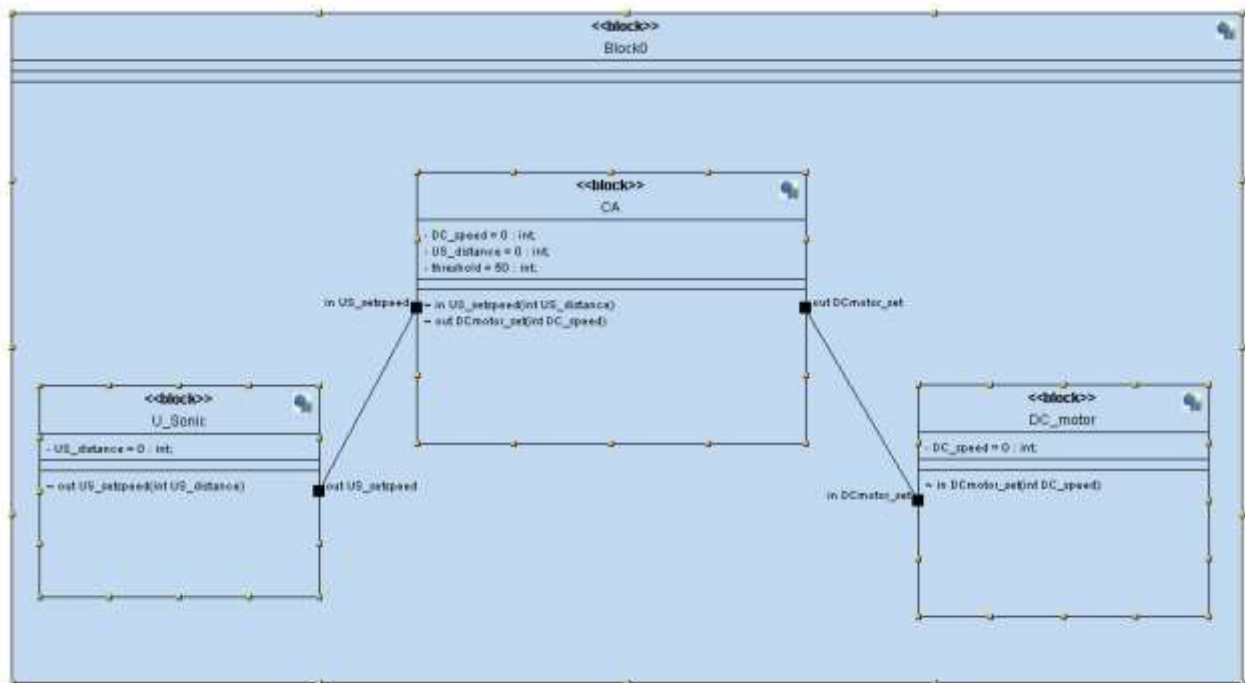


Figure 1: Collision Avoidance Block

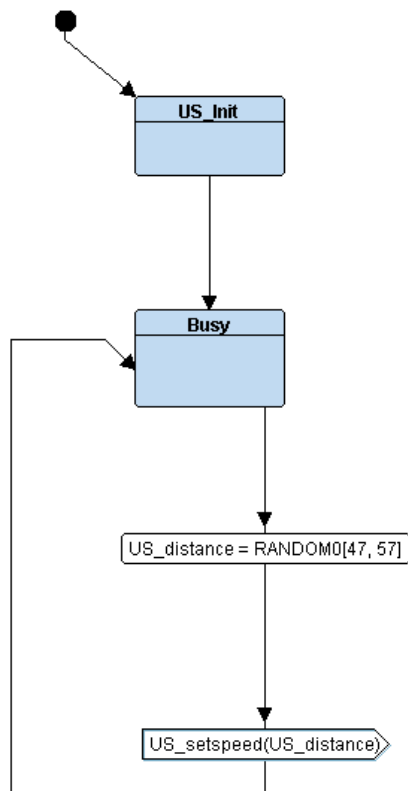
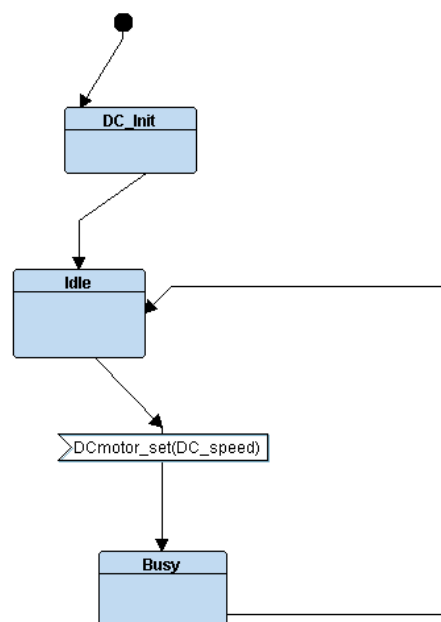
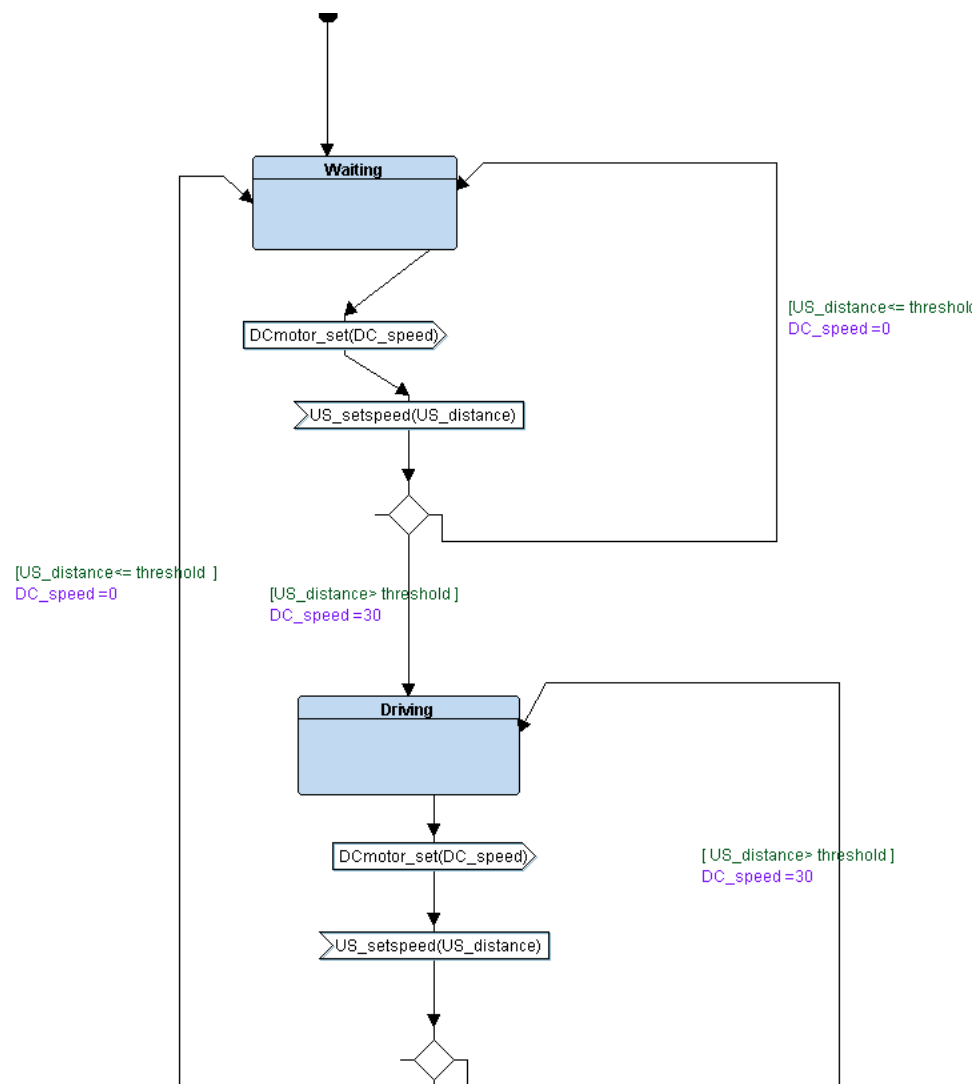


Figure 2: Ultrasonic_States



- Verify this design using the same tool.

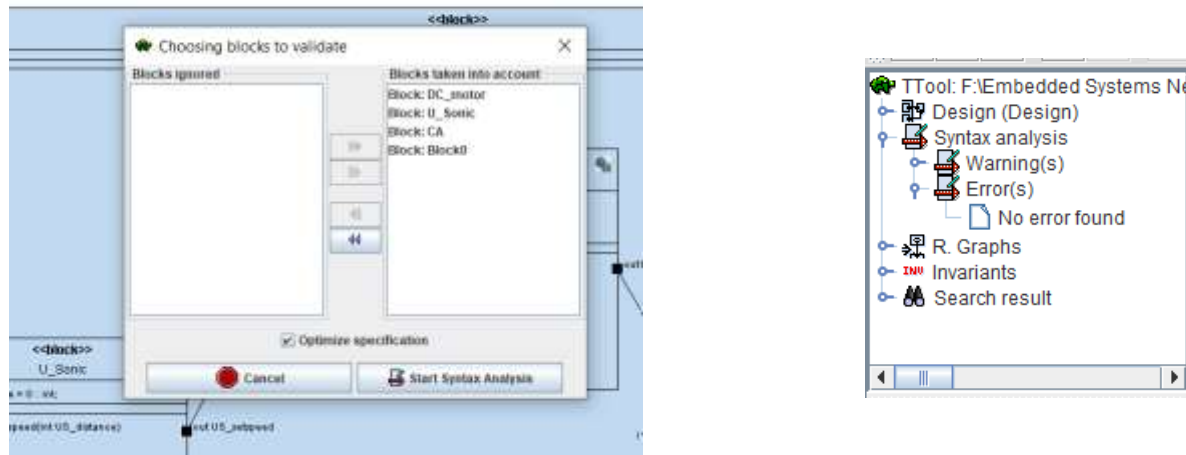


Figure 5: Syntax Analysis

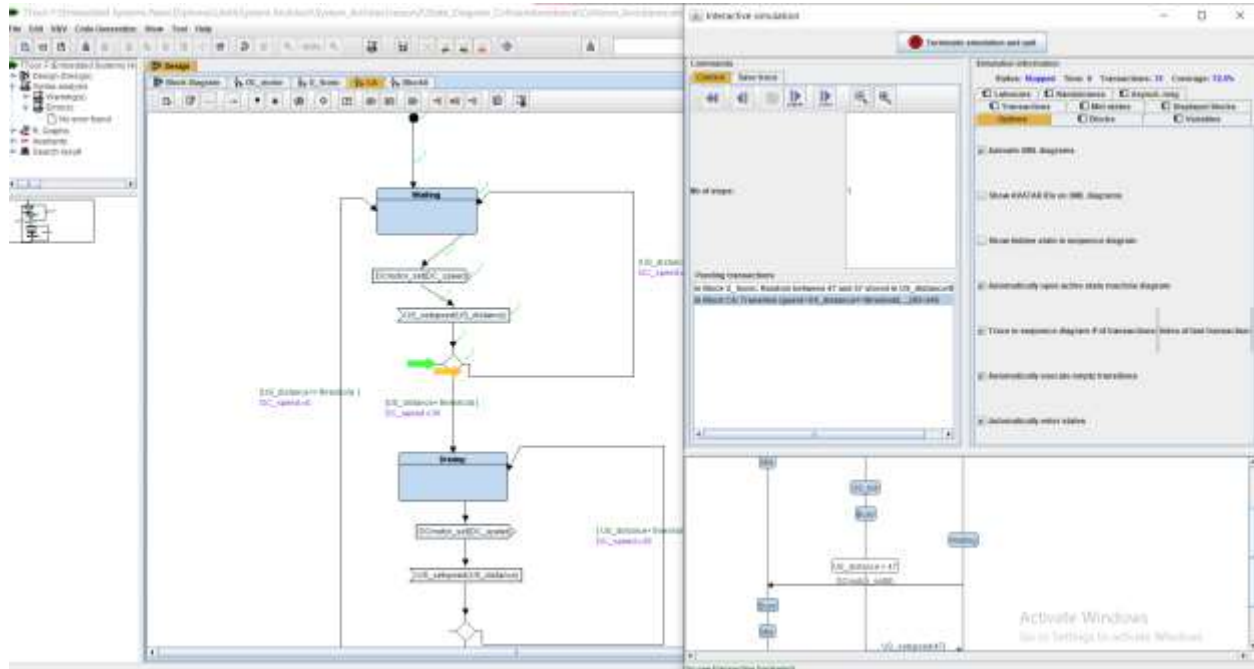
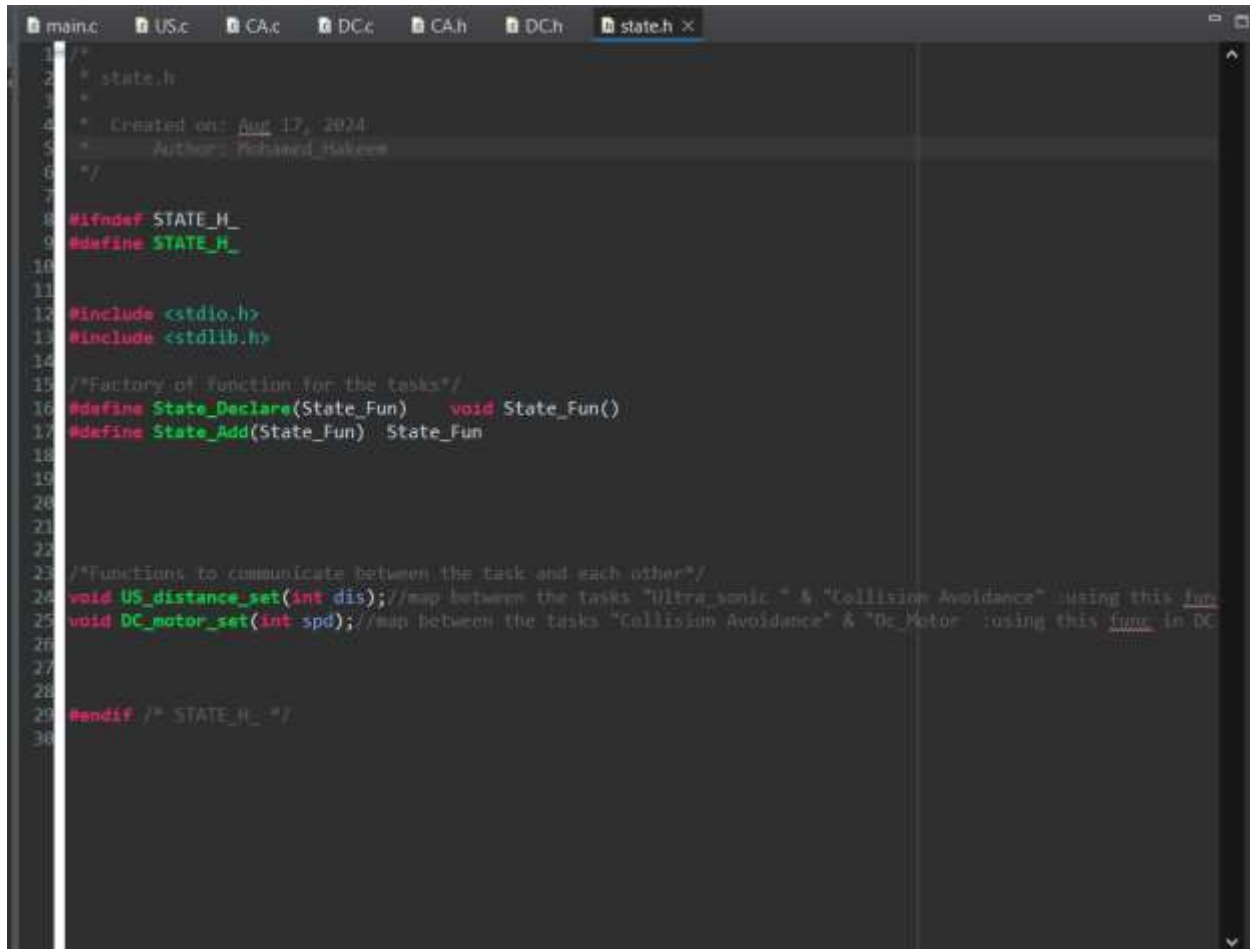


Figure 6: Verification

- Once it is achieved successfully, implement the code of each task that is built on the state diagram with the same template in any IDE.



```
1 /*
2  * state.h
3  *
4  * Created on: Aug 17, 2024
5  * Author: Mohamed_Hakem
6  */
7
8 #ifndef STATE_H_
9 #define STATE_H_
10
11
12 #include <stdio.h>
13 #include <stdlib.h>
14
15 /*Factory of function for the tasks*/
16 #define State_Declare(State_Fun) void State_Fun()
17 #define State_Add(State_Fun) State_Fun
18
19
20
21
22
23 /*Functions to communicate between the task and each other*/
24 void US_distance_set(int dis); //map between the tasks "Ultra_sonic" & "Collision Avoidance" using this func
25 void DC_motor_set(int spd); //map between the tasks "Collision Avoidance" & "Dc_Motor" using this func in DC
26
27
28
29 #endif /* STATE_H_ */
30
```

Figure 7: States_code

```
1 /*
2  * US.h
3  *
4  * Created on: Aug 17, 2024
5  * Author: Mohamed Bakr
6  */
7
8 #ifndef US_H_
9 #define US_H_
10
11 #include "state.h"
12
13 int generate_random(int l, int r, int count);
14
15
16 /*states that you made in the tool of state diagram*/
17 enum
18 {
19     US_Init,
20     US_Busy
21 }US_State_ID;
22
23
24
25 State_Declare(US_init);
26 State_Declare(US_busy);
27
28
29 //global pointer to func.
30 void (*US_State());
31
32
33 #endif /* US_H_ */
34
```

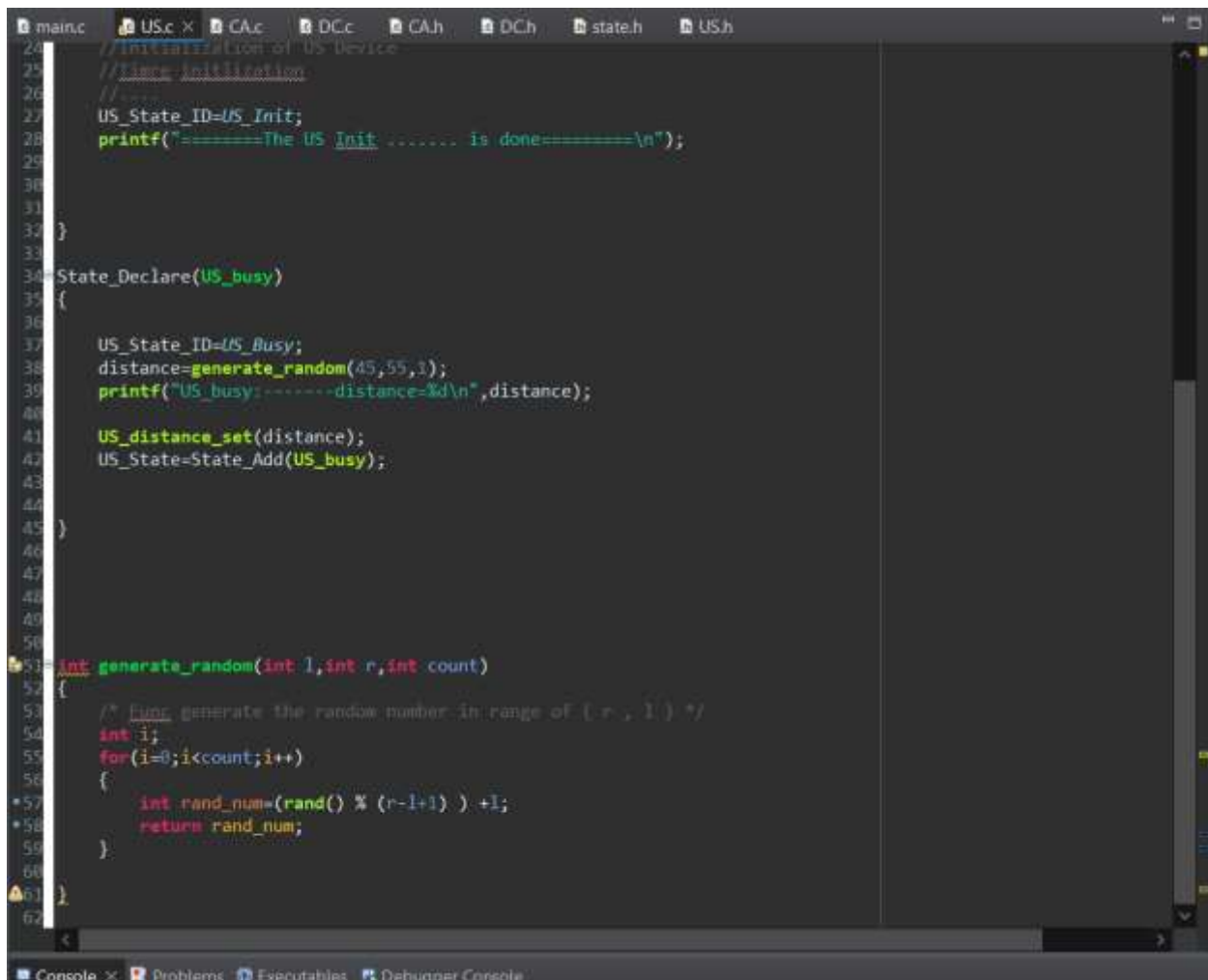
Figure 8: Ultrasonic_header

```

1 //
2 * US.c
3 *
4 * Created on: Aug 17, 2024
5 * Author: Mohamed Hakeem
6 * US.c[Ultrasonic_Sensor] is task or file that responsible for reading the distance and then sending the
7 * to the task is called CA.c [Collision Avoidance] that is task that has "algorithm" for achieving the req
8 */
9
10
11 #include "US.h"
12 unsigned int distance=0;
13 extern void (*US_State)();
14
15
16
17
18
19
20 State_Declare(US_init)
21 {
22
23
24     //Initialization of US Device
25     //Time initialization
26     //....
27     US_State_ID=US_Init;
28     printf("=====The US Init ..... is done=====\n");
29
30
31
32 }
33
34 State_Declare(US_busy)
35 {
36
37     US_State_ID=US_Busy;
38     distance=generate_random(45,55,1);
39     printf("US busy:-----distance=%d\n", distance);
40

```

Figure 9:Ultrasonic_source



```
main.c  US.c x  CA.c  DC.c  CA.h  DCh  state.h  US.h
24 //Initialization of US Device
25 //Time initialization
26 //....
27 US_State_ID=US_Init;
28 printf("=====The US Init ..... is done=====\n");
29
30
31
32 }
33
34 State_Declare(US_busy)
35 {
36
37     US_State_ID=US_Busy;
38     distance=generate_random(45,55,1);
39     printf("US_busy:-----distance=%d\n",distance);
40
41     US_distance_set(distance);
42     US_State=State_Add(US_busy);
43
44 }
45
46
47
48
49
50
51 int generate_random(int l,int r,int count)
52 {
53     /* Func generate the random number in range of ( r , l ) */
54     int i;
55     for(i=0;i<count;i++)
56     {
57         int rand_num=(rand() % (r-l+1) ) +l;
58         return rand_num;
59     }
60 }
61
62
```

Figure 10:Ultrasonic_source

```
main.c  US.c  CA.c  DCc  CA.h x  DCh  state.h  US.h
1  /*
2   * CA.h
3   *
4   * Created on: Aug 17, 2024
5   * Author: Mohamed Elakeem
6   * Collision Avoidance task is the algorithm that control between the another tasks
7   */
8
9  #ifndef CA_H_
10 #define CA_H_
11
12 #include "state.h"
13
14
15 /*states that you made in the tool of state diagram*/
16 enum
17 {
18     CA_Waiting,
19     CA_Driving
20 }CA_State_ID;
21
22
23
24 State_Declare(CA_waiting);
25 State_Declare(CA_driving);
26
27
28
29 //global pointer to func;
30 void (*CA_State)();
31
32
33
34 #endif /* CA_H_ */
35
```

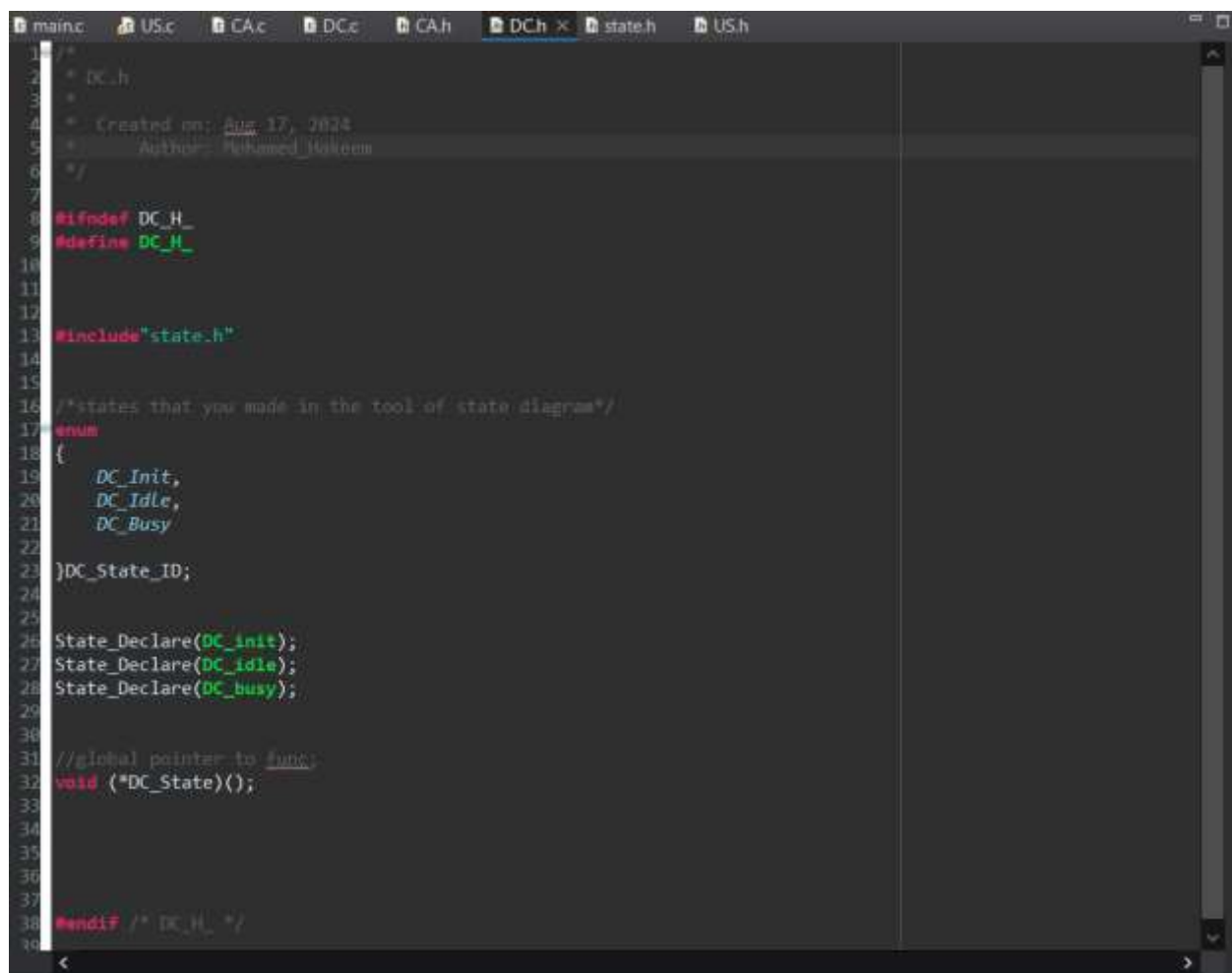
Figure 11:Collision_Avoidance_Header

```
main.c  US.c  CA.c x  DC.c  CA.h  DC.h  state.h  US.h
1  /*
2   * CA.c
3   *
4   * Created on: Aug 17, 2024
5   * Author: Mohamed_Hakeem
6   *
7   * CA.c[Collision Avoidance] is THE Algorithm
8   */
9
10 #include "CA.h"
11
12 unsigned int distance,speed,threshold=50;
13 extern void (*CA_State)();
14
15
16
17 void US_distance_set(int dis)
18 {
19
20     distance=dis;
21     (distance<=threshold)?(CA_State=State_Add(CA_waiting)):(CA_State=State_Add(CA_driving));
22     printf("-----the US distance..... = %d -----\\n",distance);
23 }
24
25
26 State_Declare(CA_waiting)
27 {
28     CA_State_ID= CA_waiting ;
29     printf("Waiting State-----: Speed=%d, Distance=%d\\n",speed,distance);
30
31     speed=0;
32     DC_motor_set(speed);
33 }
34
35
36
37
38 State_Declare(CA_driving)
39 {
40 }
```

Figure 12::Collision_Avoidance_Source

```
main.c US.c CA.c x DC.c CA.h DC.h state.h US.h
10 #include "CA.h"
11
12 unsigned int distance,speed,threshold=50;
13 extern void (*CA_State)();
14
15
16
17 void US_distance_set(int dis)
18 {
19
20     distance=dis;
21     (distance<threshold)?(CA_State=State_Add(CA_waiting)):(CA_State=State_Add(CA_driving));
22     printf("=====the US distance..... = %d =====\n",distance);
23 }
24
25
26 State_Declare(CA_waiting)
27 {
28     CA_State_ID= CA_Waiting ;
29     printf("Waiting State-----: Speed=%d, Distance=%d\n",speed,distance);
30
31     speed=0;
32     DC_motor_set(speed);
33 }
34
35
36
37
38 State_Declare(CA_driving)
39 {
40     CA_State_ID= CA_Driving ;
41     printf("Driving State-----: Speed=%d, Distance=%d\n",speed,distance);
42
43     speed=30;
44     DC_motor_set(speed);
45 }
46
47
```

Figure 13::Collision_Avoidance_Source

A screenshot of a code editor window with multiple tabs at the top: 'main.c', 'US.c', 'CA.c', 'DC.c', 'CA.h', 'DCh.x', 'state.h', and 'US.h'. The 'DCh.x' tab is active. The code is a C header file for a DC motor, starting with a multi-line comment block containing the file name 'DC.h', creation date 'Aug 17, 2024', and author 'Mohamed_Makrem'. It uses preprocessor directives to define 'DC_H_'. It includes 'state.h' and defines an enumeration 'DC_State_ID' with values 'DC_Init', 'DC_Idle', and 'DC_Busy'. It then declares state variables 'DC_init', 'DC_idle', and 'DC_busy' using 'State_Declare'. A global pointer to a function 'DC_State' is declared. The file ends with a conditional compilation directive '#endif /* DC_H_ */'.

```
1 /*  
2  * DC.h  
3  *  
4  * Created on: Aug 17, 2024  
5  * Author: Mohamed_Makrem  
6  */  
7  
8 #ifndef DC_H_  
9 #define DC_H_  
10  
11  
12  
13 #include "state.h"  
14  
15  
16 /*states that you made in the tool of state diagram*/  
17 enum  
18 {  
19     DC_Init,  
20     DC_Idle,  
21     DC_Busy  
22 }DC_State_ID;  
23  
24  
25  
26 State_Declare(DC_init);  
27 State_Declare(DC_idle);  
28 State_Declare(DC_busy);  
29  
30  
31 //global pointer to func;  
32 void (*DC_State)();  
33  
34  
35  
36  
37  
38 #endif /* DC_H_ */  
39
```

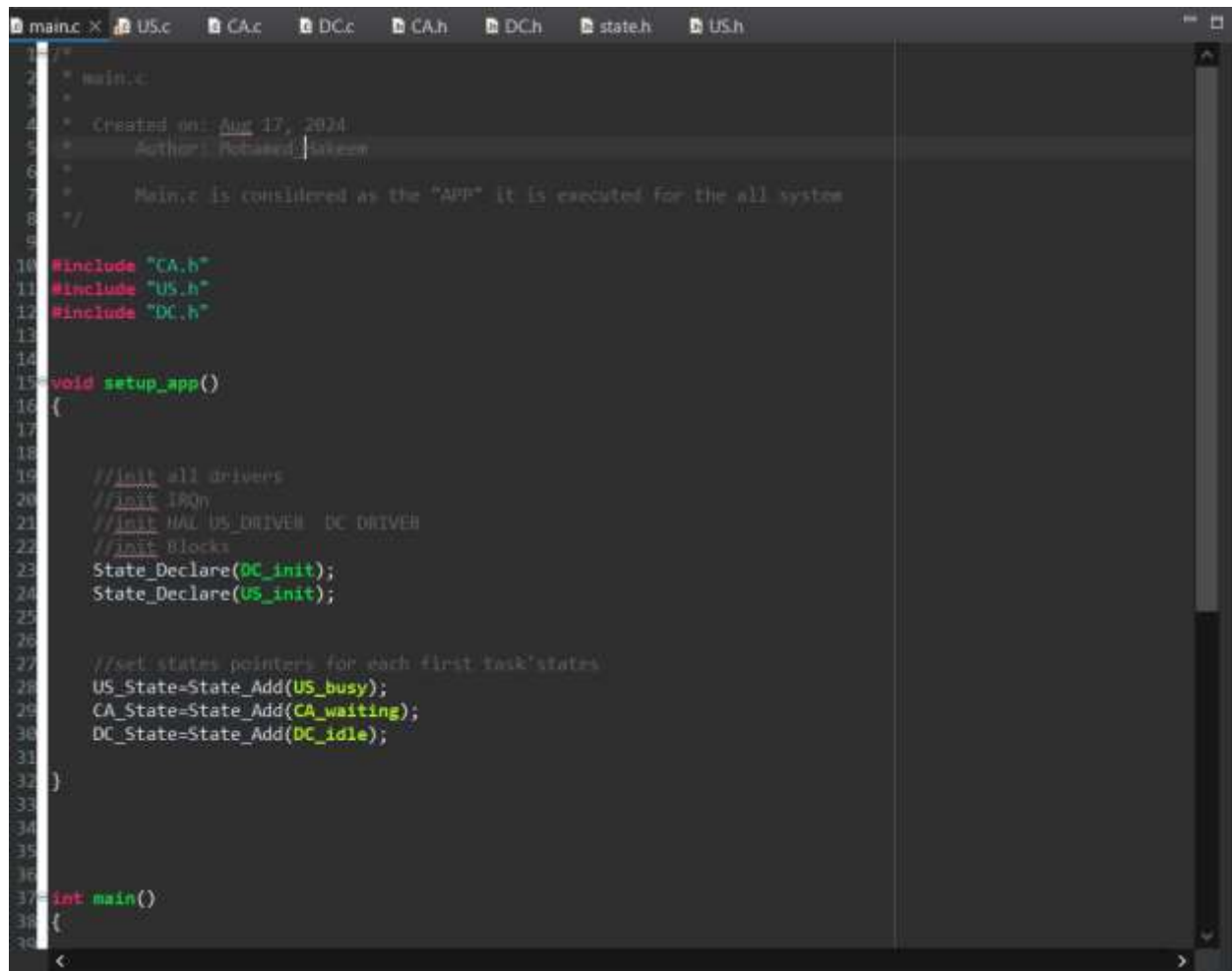
Figure 14:Dc_Motor_Header

```
main.c  US.c  CA.c  DC.c x  CA.h  DCh  state.h  US.h
1  /*
2   * DC.c
3   *
4   * Created on: Aug 17, 2024
5   * Author: Mohamed_Hakem
6   *
7   *
8   * DC.c[DC_Motor] is the task that receives actor(speed value) from the algorithm CA.c
9   */
10
11 #include "DC.h"
12 unsigned int speed=0;
13
14 extern void (*DC_State)();
15
16
17
18 void DC_motor_set(int spd)
19 {
20
21
22     speed=spd;
23     DC_State=State_Add(DC_busy);
24     printf("DC.....Speed= %d\n", speed);
25
26
27 }
28
29
30
31 State_Declare(DC_init)
32 {
33
34
35     //DC_Motor Initialization
36     //PM module init.....
37     DC_State_ID=DC_Init;
38     printf("=====The DC Init is .....Done=====\\n");
39
40 }
```

Figure 15:Dc_Motor_Source

```
main.c  US.c  CA.c  DCc.X  CA.h  DCh  state.h  US.h
22 speed=spd;
23 DC_State=State_Add(DC_busy);
24 printf("DC.....Speed= %d\n",speed);
25
26
27
28 }
29
30
31 State_Declare(DC_init)
32 {
33
34
35 //DC_Motor Initialization
36 //PWM module init.....
37 DC_State_ID=DC_Init;
38 printf("=====The DC Init is .....Done=====\\n");
39
40 }
41
42 State_Declare(DC_idle)
43 {
44 //it don't anything ,it leaves the motor on the previous speed
45 DC_State_ID=DC_Idle;
46 printf("Idle State -----:Speed=%d\\n",speed);
47
48
49
50 }
51
52 State_Declare(DC_busy)
53 {
54 DC_State_ID=DC_Busy;
55 //send PWM to DC motor
56 printf("Busy_State-----: Speed= %d\\n",speed);
57 DC_State=State_Add(DC_idle);
58
59 }
60
```

Figure 16::Dc_Motor_Source



```
1  /*
2   * main.c
3   *
4   * Created on: Aug 17, 2024
5   * Author: Mohamed Saleem
6   *
7   * Main.c is considered as the "APP" it is executed for the all system
8   */
9
10 #include "CA.h"
11 #include "US.h"
12 #include "DC.h"
13
14
15 void setup_app()
16 {
17
18     //init all drivers
19     //init IRON
20     //init HAL US_DRIVER DC DRIVER
21     //init Blocks
22     State_Declare(DC_init);
23     State_Declare(US_init);
24
25
26
27     //set states pointers for each first task's states
28     US_State=State_Add(US_busy);
29     CA_State=State_Add(CA_waiting);
30     DC_State=State_Add(DC_idle);
31
32 }
33
34
35
36
37 int main()
38 {
39
```

Figure 17:Main


```
main.c x US.c CA.c DC.c CA.h DCh state.h US.h
23 State_Declare(DC_init);
24 State_Declare(US_init);
25
26
27 //set states pointers for each first task's states
28 US_State=State_Add(US_busy);
29 CA_State=State_Add(CA_waiting);
30 DC_State=State_Add(DC_idle);
31
32 }
33
34
35
36
37 int main()
38 {
39
40     volatile int i=0;
41     setup_app();
42
43
44     while(1)
45     {
46         //call the pointer to function respectively according to your state diagram
47         US_State();
48         CA_State();
49         DC_State();
50
51         for(i=0;i<1000;i++); //delay
52
53     }
54
55
56
57     return 0;
58 }
59
60
61
```

Figure 18:Main

