

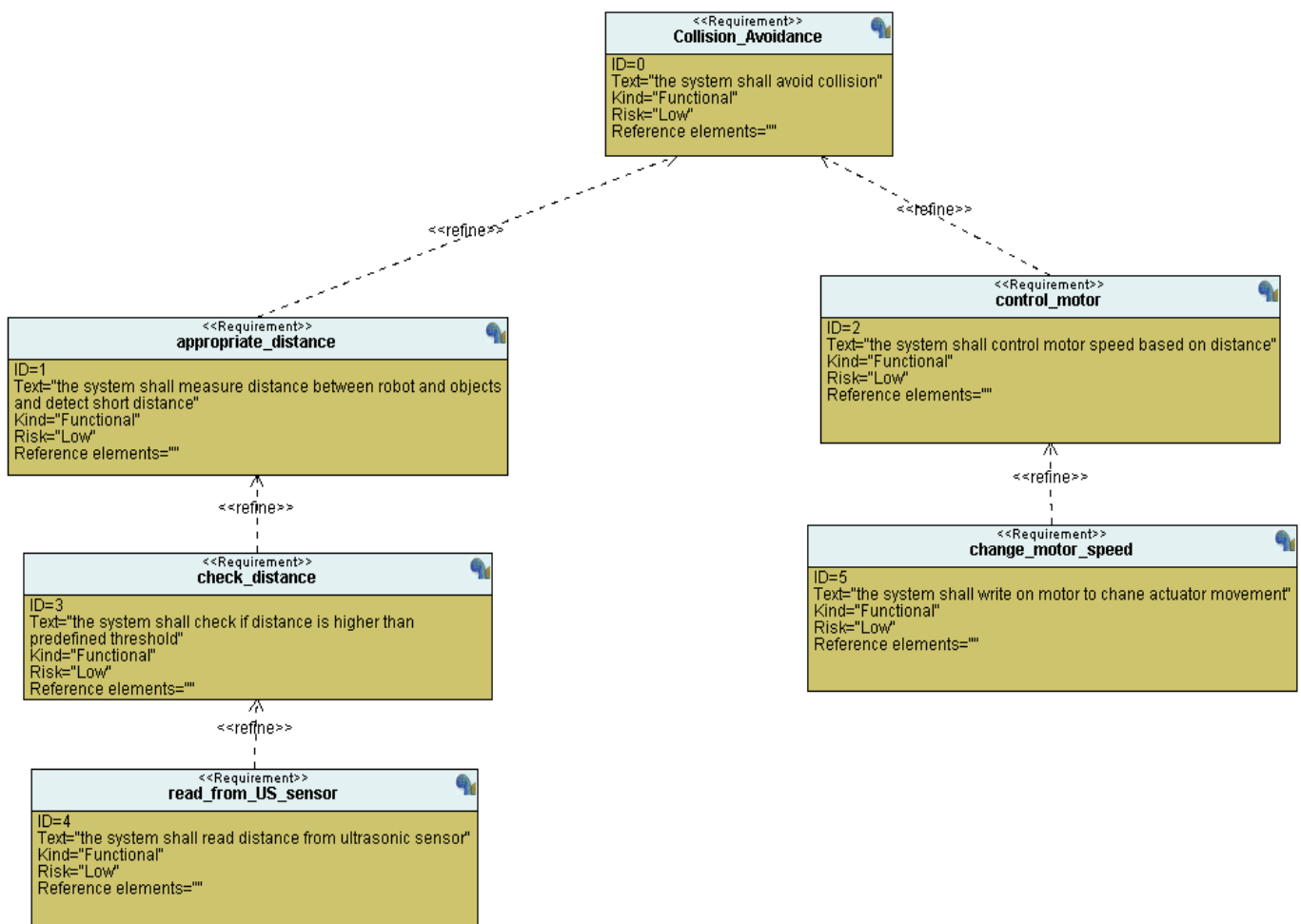
# Collision Avoidance Robot

- Design sequence:

- **Case study:**

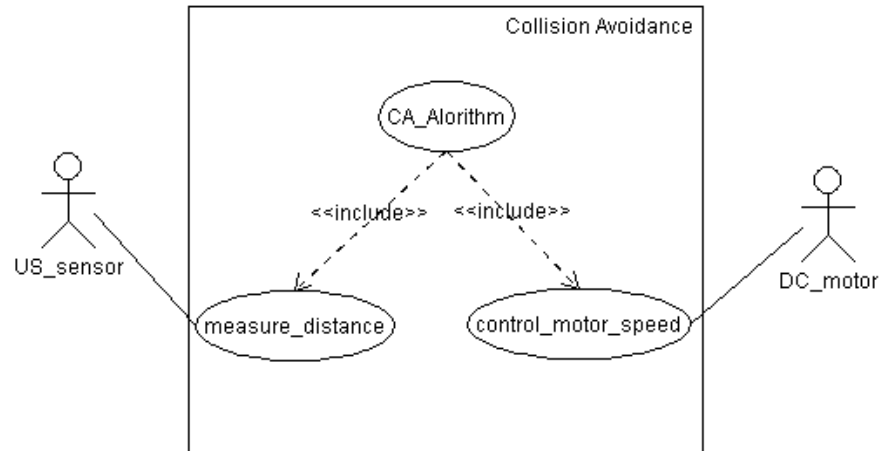
A robot could measure distance between it and objects around by reading it from ultrasonic sensor to avoid collision with this object if the distance is lower than threshold value (50 cm) then it will stop (speed = 0), if greater than 50 cm it will go driving (speed = 30).

- **Requirement diagram:**

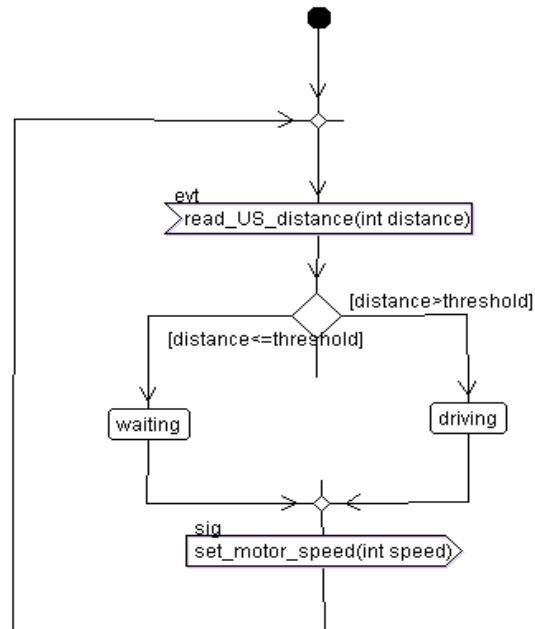


- **System Analysis:**

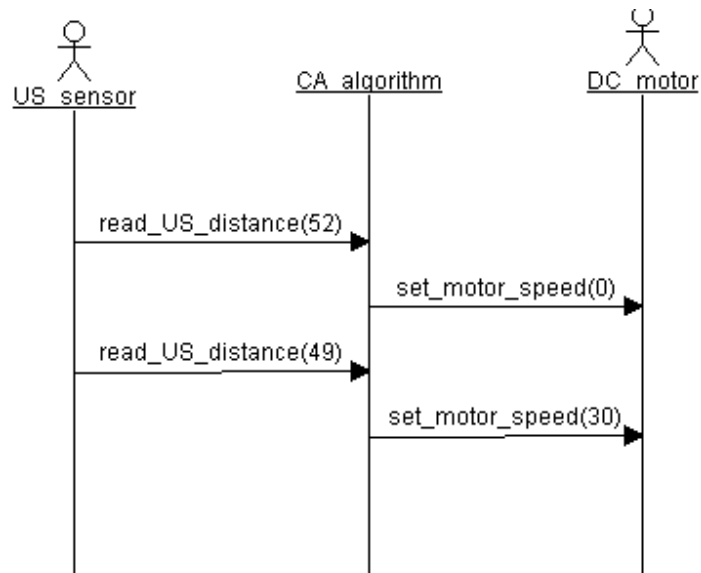
- **Use case diagram:**



- **Activity diagram:**

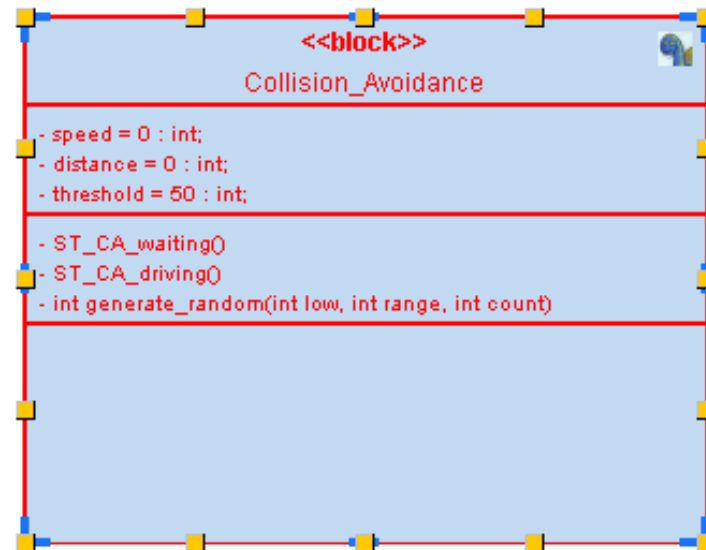


- **Sequence Diagram:**

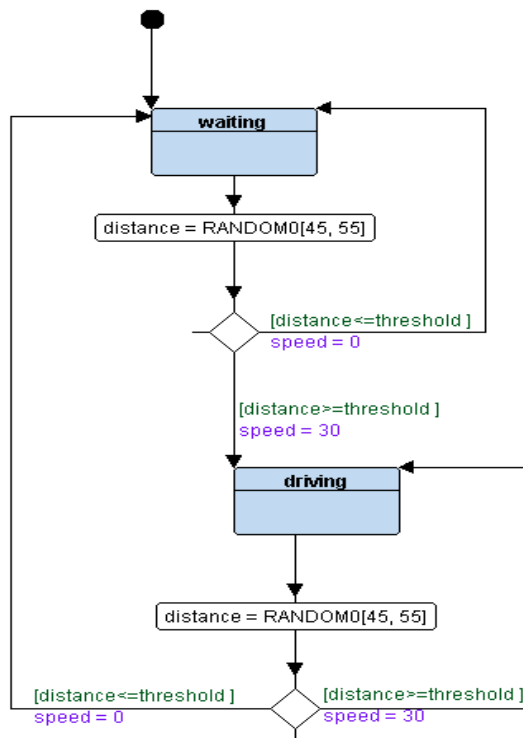


- System design using Single Module:

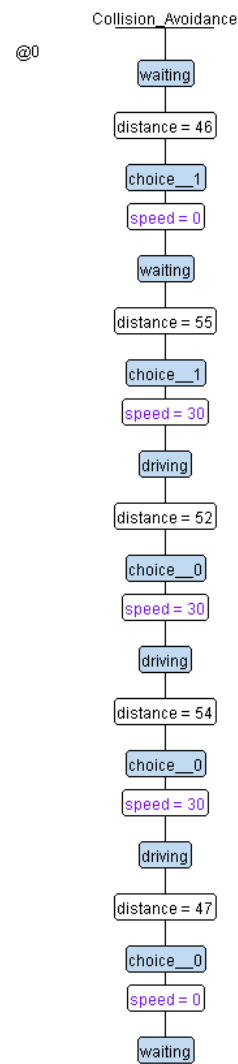
- **Block diagram:**



- **State diagram**



- > **Interactive simulation:**



➤ **Implementation:**  
**- CA.h**

```
CA.h  main.c  CA.c
1  #ifndef _CA_H_
2  #define _CA_H_
3
4  #include <stdio.h>
5  #include <stdlib.h>
6
7
8  #define STATE_define(_stateFunc_) void ST_##_stateFunc_()
9  #define STATE(_stateFunc_) ST_##_stateFunc_
10
11 //define states
12 enum {
13     waiting,
14     driving
15 }state_id;
16 //global pointer to function
17 void (*state)();
18
19 //APIs
20 STATE_define(waiting);
21 STATE_define(driving);
22
23 #endif
```

**- main.c**

```
CA.h  main.c  CA.c
1  #include "CA.h"
2
3 void setup()
4 {
5     //init state of CA robot motor
6     //init modules of SOC
7     state = STATE(waiting);
8 }
9
10 void main()
11 {
12     volatile int d;
13     setup();
14     while (1)
15     {
16         //detect state of motor
17         state();
18         //delay
19         for(d = 0; d <=1000;d++);
20     }
21 }
```

## - CA.c

```
CA.h  main.c  CA.c
1  #include "CA.h"
2  //define system variables
3  unsigned int distance, speed, threshold = 50;
4
5  //generate random number function
6  int generate_random(int low , int range,int count){
7
8      int i ;
9      int rand_num;
10     for(i = 0; i < count; i++){
11         rand_num = (rand()%(range-low+1)) + low;
12     }
13     return rand_num ;
14 }
15 //define states of robot motor speed
16 STATE_define(waiting){
17     //state actions
18     state_id = waiting ;
19     speed = 0;
20     distance = generate_random(40,60,1);
21     //check reading from sensor
22     (distance <= threshold)? (state=STATE(waiting)) : (state=STATE(driving)) ;
23     printf("waiting state : distance = %d , speed = %d \n",distance,speed );
24 }
25 STATE_define(driving){
26     //state actions
27     state_id = driving ;
28     speed = 30;
29     distance = generate_random(40,60,1);
30     //check reading from sensor
31     (distance <= threshold)? (state=STATE(waiting)) : (state=STATE(driving)) ;
32     printf("driving state : distance = %d , speed = %d \n",distance,speed );
33 }
```

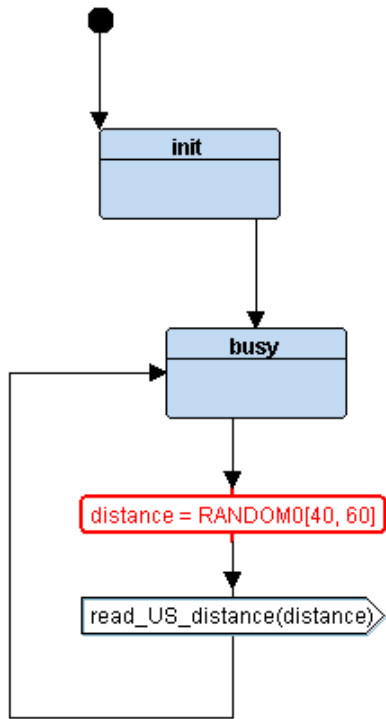
### ➤ Code running:

```
Problems  Tasks  Console  Properties
<terminated> (exit value: -1) CA_single_Module Debug [C/C
waiting state : distance = 52 , speed = 0
driving state : distance = 46 , speed = 30
waiting state : distance = 40 , speed = 0
waiting state : distance = 54 , speed = 0
driving state : distance = 58 , speed = 30
driving state : distance = 54 , speed = 30
driving state : distance = 50 , speed = 30
waiting state : distance = 50 , speed = 0
waiting state : distance = 59 , speed = 0
driving state : distance = 56 , speed = 30
driving state : distance = 47 , speed = 30
waiting state : distance = 46 , speed = 0
waiting state : distance = 42 , speed = 0
waiting state : distance = 51 , speed = 0
driving state : distance = 55 , speed = 30
driving state : distance = 42 , speed = 30
waiting state : distance = 46 , speed = 0
waiting state : distance = 42 , speed = 0
waiting state : distance = 49 , speed = 0
waiting state : distance = 57 , speed = 0
driving state : distance = 43 , speed = 30
```

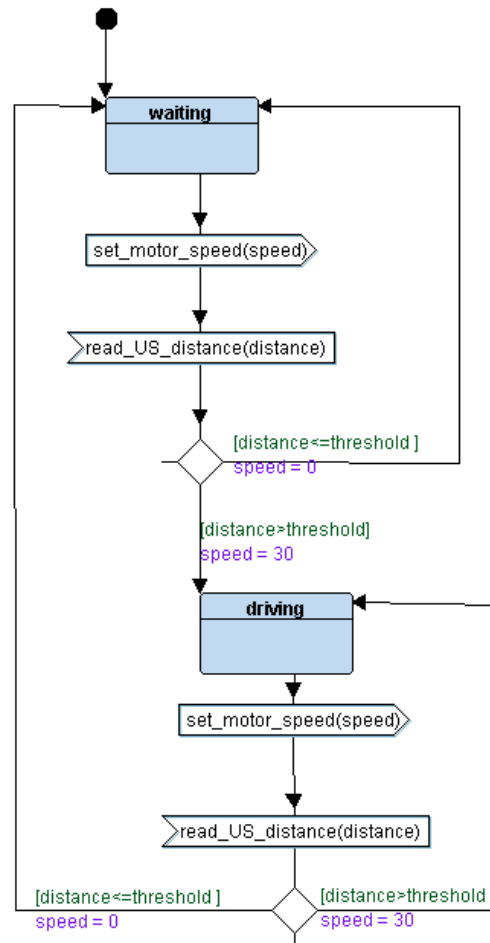
- System design using Multiple Modules:

- **Block diagram:**

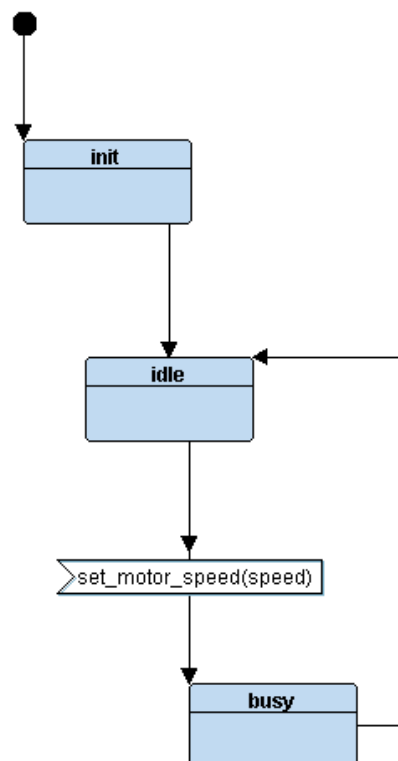
- UltraSonic sensor:



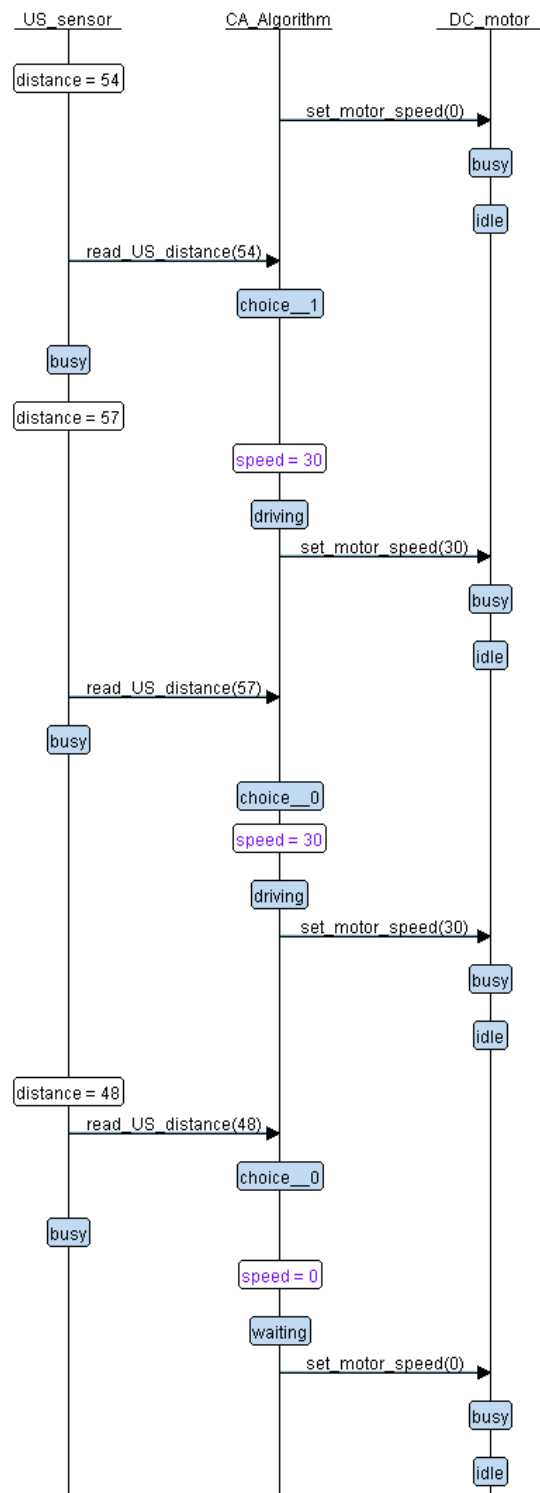
- CA Algorithm:



- DC motor:



➤ Interactive simulation:



➤ **Implementation:**  
**- CA.h**

```
CA.h  DC.c  DC.h  US.h  US.c  CA.c
1  #ifndef _CA_H_
2  #define _CA_H_
3
4  #include <stdio.h>
5  #include <stdlib.h>
6
7  #define STATE_define(_stateFunc_) void ST_##_stateFunc_()
8  #define STATE(_stateFunc_) ST_##_stateFunc_
9
10 //signals connections
11 void read_US_distance(int d);
12 void set_motor_speed(int s);
13
14 //define states
15 enum {
16     waiting,
17     driving
18 }state_id;
19 //global pointer to function
20 void (*state)();
21
22 //APIs
23 STATE_define(waiting);
24 STATE_define(driving);
25
26 #endif
```

**- CA.c**

```
CA.h  CA.c  DC.c  DC.h  US.h  US.c  main.c
4
5 //define states of robot motor speed
6 STATE_define(waiting){
7     //state actions
8     state_id = waiting ;
9     //check reading from sensor
10    printf("waiting state : distance = %d , speed = %d \n\n",distance,speed );
11    speed = 0;
12    set_motor_speed(speed);
13 }
14
15 STATE_define(driving){
16     //state actions
17     state_id = driving ;
18
19     //check reading from sensor
20     (distance <= threshold)? (state=STATE(waiting)) : (state=STATE(driving)) ;
21     printf("driving state : distance = %d , speed = %d \n\n",distance,speed );
22     //set motor speed to 30 (Turn on)
23     speed = 30;
24     set_motor_speed(speed);
25 }
26
27 void read_US_distance(int d){
28     distance = d;
29     //check reading from sensor
30     (distance <= threshold)? (state=STATE(waiting)) : (state=STATE(driving)) ;
31     printf("US-----distance = %d----->CA\n",distance);
32 }
```



## - US.h

```
US.h CA.h CA.c DC.  
1 #ifndef _US_H_  
2 #define _US_H_  
3 #include "CA.h"  
4  
5 //define states  
6 enum {  
7     US_busy  
8 }US_state_id;  
9 //global pointer to function  
10 void (*US_state)();  
11  
12 //APIs  
13 void US_init();  
14 STATE_define(US_busy);  
15  
16 #endif  
17
```

## - US.c

```
CA.h DC.c DC.h US.h US.c CA.c  
1  
2 //define system variables  
3 unsigned int distance ;  
4  
5  
6 //generate random number of distances  
7 int generate_random(int low , int range,int count){  
8  
9     int i ;  
10    int rand_num;  
11    for(i = 0; i < count; i++){  
12        rand_num = (rand()%(range-low+1)) + low;  
13    }  
14    return rand_num ;  
15 }  
16  
17 void US_init(){  
18     //init US sensor by calling its driver  
19     printf("US init \n");  
20 }  
21 //define states of robot motor speed  
22 STATE_define(US_busy){  
23     //state actions  
24     state_id = US_busy ;  
25     //read distance from US  
26     distance = generate_random(40,60,1);  
27     printf("US_busy state : distance = %d \n",distance );  
28     read_US_distance(distance);  
29     US_state = STATE(US_busy);  
30 }
```

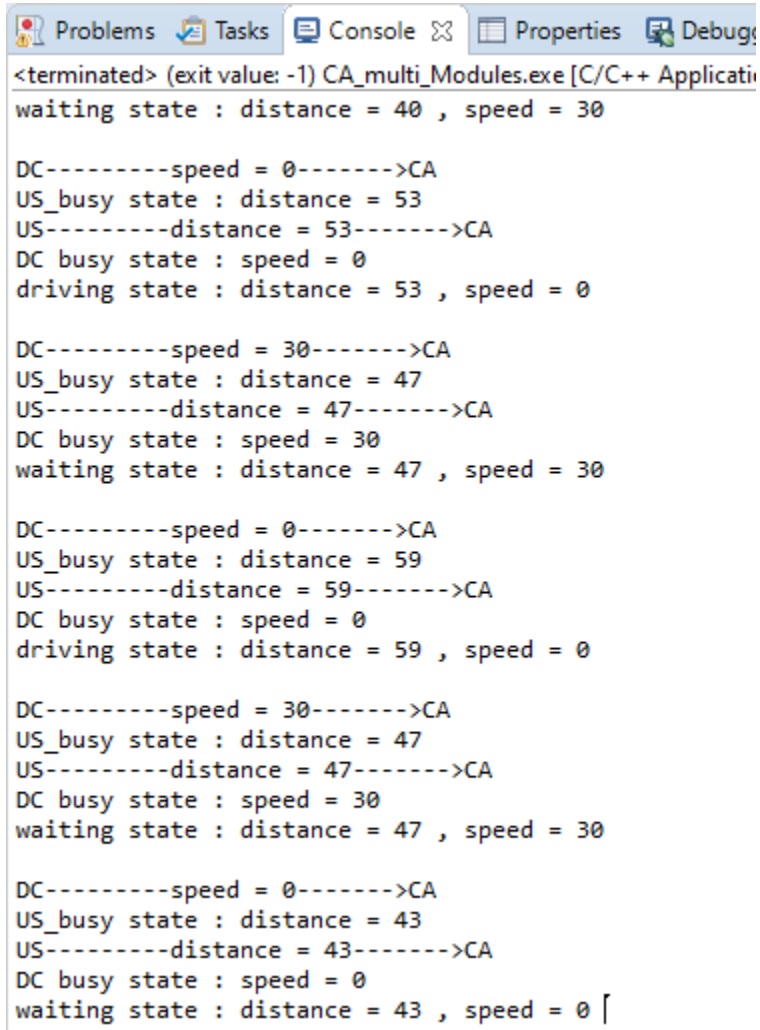
## - DC.h

```
DC.h  US.h  CA.h  C
1  #ifndef _DC_H_
2  #define _DC_H_
3  #include "CA.h"
4  //define states
5  enum {
6      DC_idle,
7      DC_busy
8  }DC_state_id;
9  //global pointer to function
10 void (*DC_state)();
11
12 //APIs
13 void DC_init();
14 STATE_define(DC_idle);
15 STATE_define(DC_busy);
16
17 #endif
```

## - DC.c

```
DC.h  DC.c  US.h  CA.h  CA.c  US.c
2  //define system variables
3  unsigned int speed;
4
5
6  void DC_init(){
7      //init DC motor by calling its driver
8      printf("DC init \n");
9  }
10 //define states of robot motor speed
11 STATE_define(DC_idle){
12     //state actions
13     DC_state_id = DC_idle ;
14     |
15     printf("DC idle state : speed = %d \n",speed );
16 }
17
18 STATE_define(DC_busy){
19     //state actions
20     DC_state_id = DC_busy ;
21     DC_state = STATE(DC_idle);
22     printf("DC busy state : speed = %d \n",speed );
23 }
24
25 void set_motor_speed(int s){
26     speed = s;
27     //set dc state to busy
28     DC_state = STATE(DC_busy);
29     printf("DC-----speed = %d----->CA\n",speed);
30 }
```

➤ Simulation code running:



```
<terminated> (exit value: -1) CA_multi_Modules.exe [C/C++ Applicati
waiting state : distance = 40 , speed = 30

DC-----speed = 0----->CA
US_busy state : distance = 53
US-----distance = 53----->CA
DC busy state : speed = 0
driving state : distance = 53 , speed = 0

DC-----speed = 30----->CA
US_busy state : distance = 47
US-----distance = 47----->CA
DC busy state : speed = 30
waiting state : distance = 47 , speed = 30

DC-----speed = 0----->CA
US_busy state : distance = 59
US-----distance = 59----->CA
DC busy state : speed = 0
driving state : distance = 59 , speed = 0

DC-----speed = 30----->CA
US_busy state : distance = 47
US-----distance = 47----->CA
DC busy state : speed = 30
waiting state : distance = 47 , speed = 30

DC-----speed = 0----->CA
US_busy state : distance = 43
US-----distance = 43----->CA
DC busy state : speed = 0
waiting state : distance = 43 , speed = 0 [
```