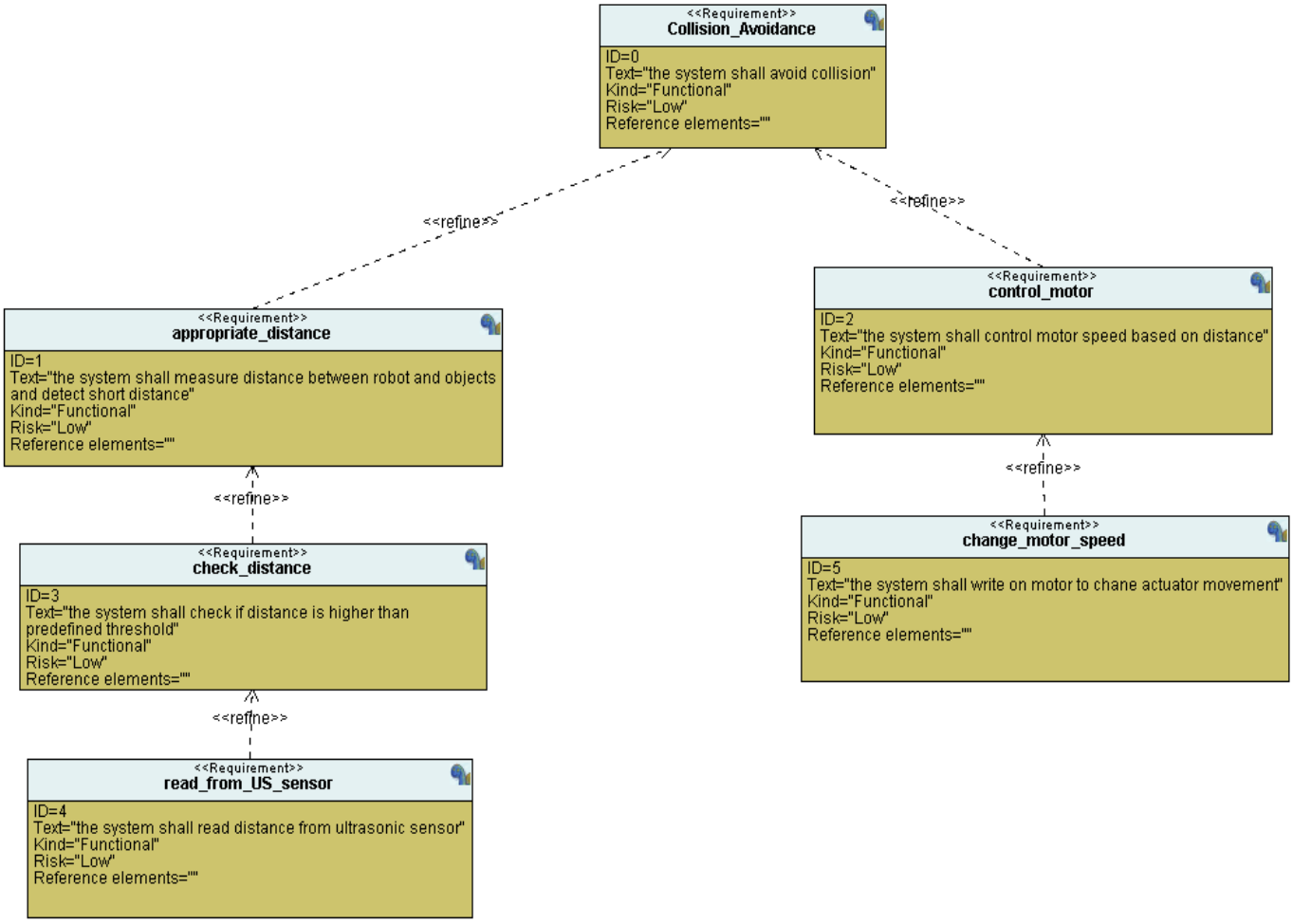
Collision Avoidance Robot

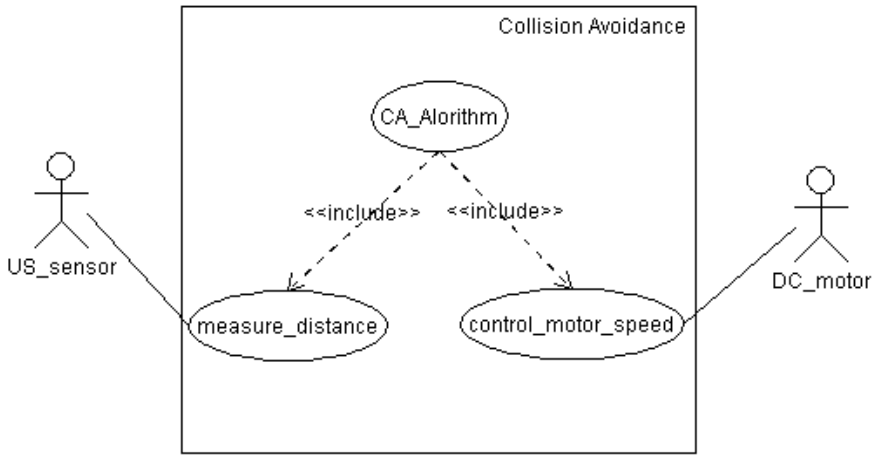
- **Design sequence:**
  - **Case study:**

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).

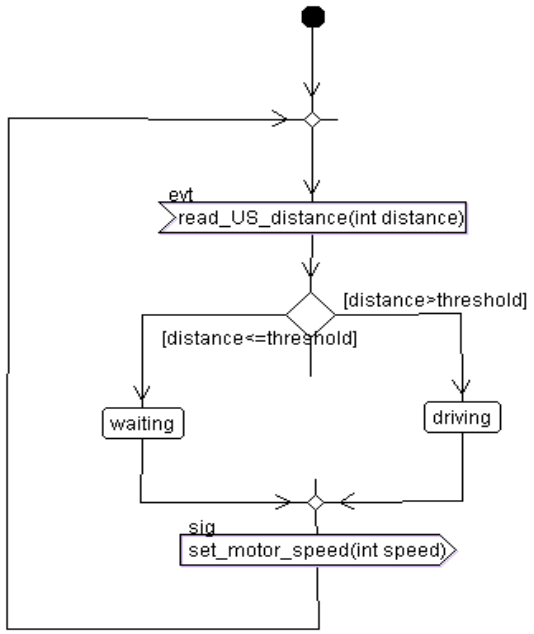
  - **Requirement diagram:**



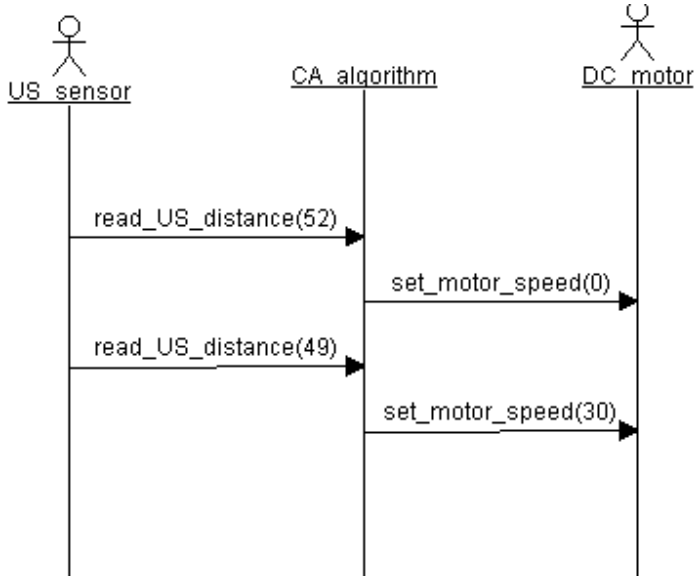
- **System Analysis:**
  - **Use case diagram:**



  - **Activity diagram:**

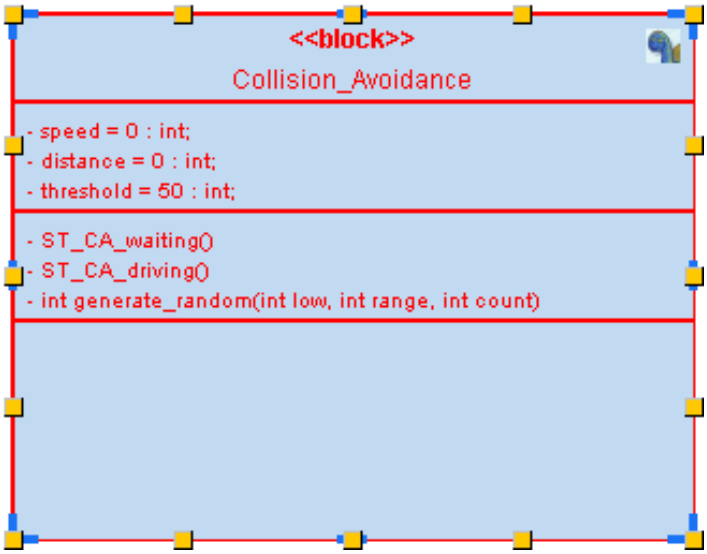


  - **Sequence Diagram:**

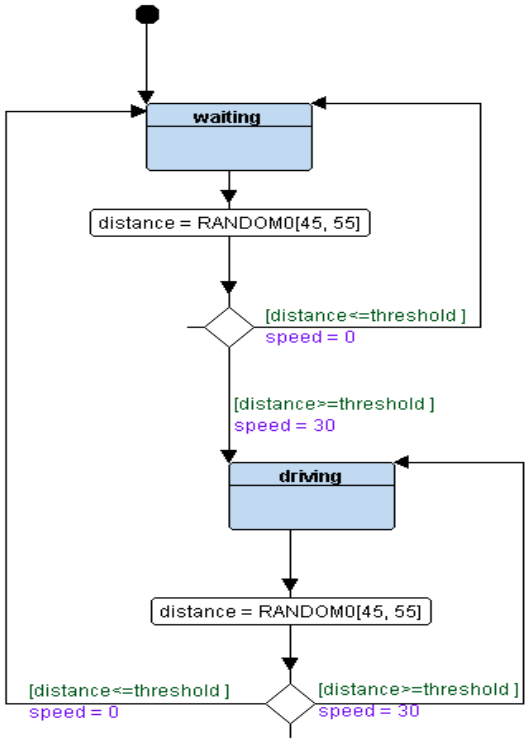


- **System design using Single Module:**

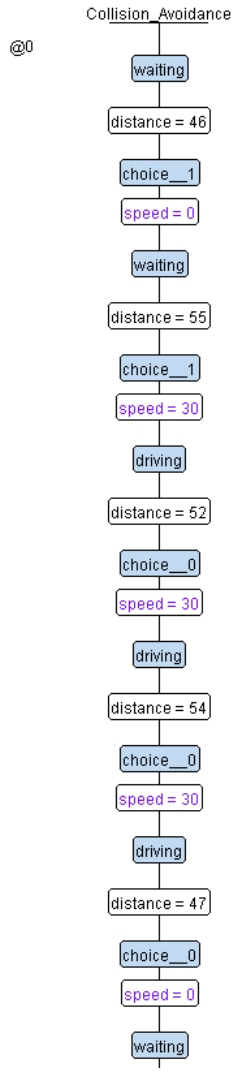
➢ **Block diagram:**



➢ **State diagram**



> **Interactive simulation:**



- **Implementation:**
  - **CA.h**

CA.h | main.c | CA.c

```c
#ifndef _CA_H_
#define _CA_H_

#include <stdio.h>
#include <stdlib.h>


#define STATE_define(_stateFunc_) void ST_##_stateFunc_()
#define STATE(_stateFunc_) ST_##_stateFunc_

//define states
enum {
    waiting,
    driving
}state_id;
//global pointer to function
void (*state)();

//APIs
STATE_define(waiting);
STATE_define(driving);

#endif
```

  - **main.c**

CA.h | main.c | CA.c

```c
#include "CA.h"

void setup()
{
    //init state of CA robot motor
    //init modules of SOC
    state = STATE(waiting);
}

void main()
{
    volatile int d;
    setup();
    while (1)
    {
        //detect state of motor
        state();
        //delay
        for(d = 0; d <=1000;d++);
    }
}
```

- CA.c

CA.h | main.c | CA.c

```c
#include "CA.h"
//define system variables
unsigned int distance, speed, threshold = 50;

//generate random number function
int generate_random(int low , int range,int count){

    int i ;
    int rand_num;
    for(i = 0; i < count; i++){
        rand_num = (rand()%(range-low+1)) + low;
    }
    return rand_num ;
}
//define states of robot motor speed
STATE_define(waiting){
    //state actions
    state_id = waiting ;
    speed = 0;
    distance = generate_random(40,60,1);
    //check reading from sensor
    (distance <= threshold)? (state=STATE(waiting)) : (state=STATE(driving)) ;
    printf("waiting state : distance = %d , speed = %d \n",distance,speed );
}
STATE_define(driving){
    //state actions
    state_id = driving ;
    speed = 30;
    distance = generate_random(40,60,1);
    //check reading from sensor
    (distance <= threshold)? (state=STATE(waiting)) : (state=STATE(driving)) ;
    printf("driving state : distance = %d , speed = %d \n",distance,speed );
}
```

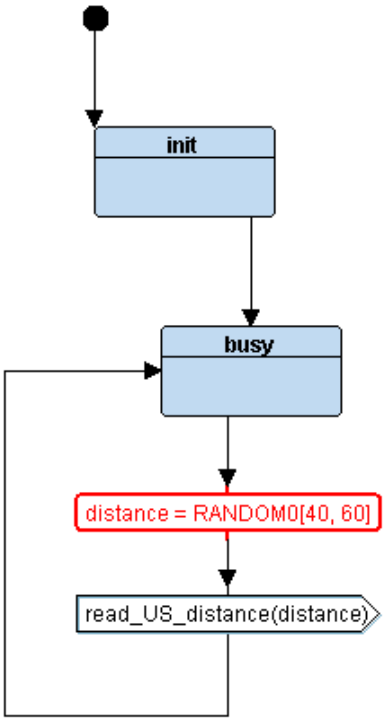
➢ Code running:

Problems | Tasks | Console | Properties

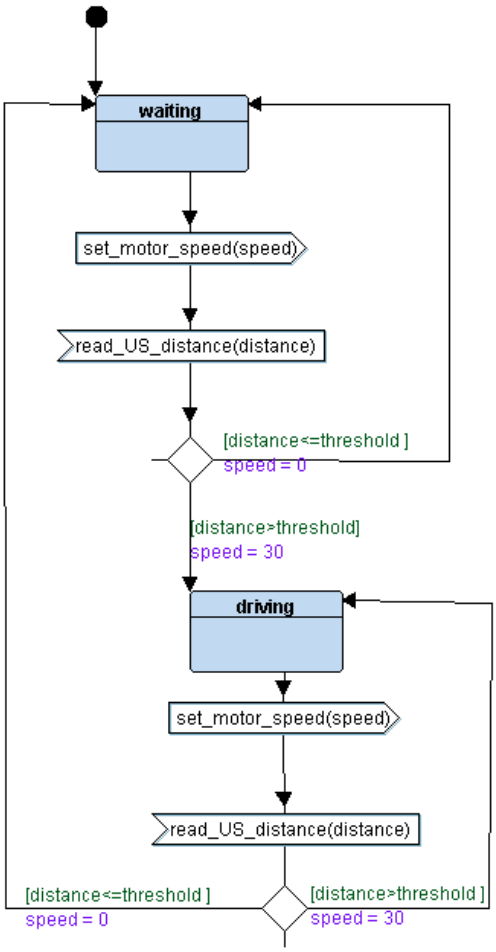
```
<terminated> (exit value: -1) CA_single_Module Debug [C/C
waiting state : distance = 52 , speed = 0
driving state : distance = 46 , speed = 30
waiting state : distance = 40 , speed = 0
waiting state : distance = 54 , speed = 0
driving state : distance = 58 , speed = 30
driving state : distance = 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:**

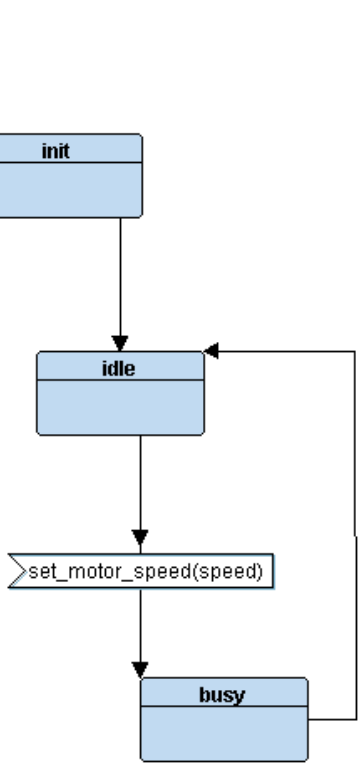
- UltraSonic sensor:



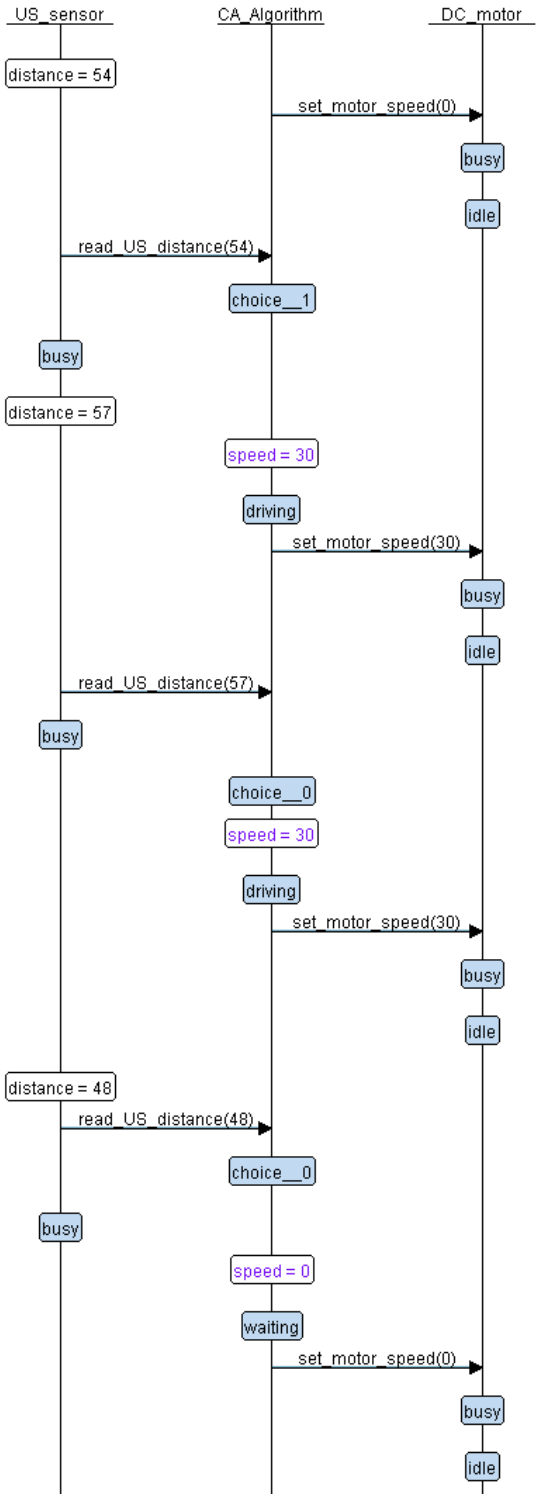
- CA Algorithm:



- DC motor:



➢ Interactive simulation:



- **Implementation:**

  - CA.h

CA.h | DC.c | DC.h | US.h | US.c | CA.c

```c
#ifndef _CA_H_
#define _CA_H_

#include <stdio.h>
#include <stdlib.h>

#define STATE_define(_stateFunc_) void ST_##_stateFunc_()
#define STATE(_stateFunc_) ST_##_stateFunc_

//signals connections
void read_US_distance(int d);
void set_motor_speed(int s);

//define states
enum {
    waiting,
    driving
}state_id;
//global pointer to function
void (*state)();

//APIs
STATE_define(waiting);
STATE_define(driving);

#endif
```

  - CA.c

CA.h | CA.c | DC.c | DC.h | US.h | US.c | main.c

```c

//define states of robot motor speed
STATE_define(waiting){
    //state actions
    state_id = waiting ;
    //check reading from sensor
    printf("waiting state : distance = %d , speed = %d \n\n",distance,speed );
    speed = 0;
    set_motor_speed(speed);
}

STATE_define(driving){
    //state actions
    state_id = driving ;

    //check reading from sensor
    (distance <= threshold)? (state=STATE(waiting)) : (state=STATE(driving)) ;
    printf("driving state : distance = %d , speed = %d \n\n",distance,speed );
    //set motor speed to 30 (Turn on)
    speed = 30;
    set_motor_speed(speed);
}

void read_US_distance(int d){
    distance = d;
    //check reading from sensor
    (distance <= threshold)? (state=STATE(waiting)) : (state=STATE(driving)) ;
    printf("US---------distance = %d------->CA\n",distance);
}
```

- **US.h**

US.h | CA.h | CA.c | DC.

```c
#ifndef _US_H_
#define _US_H_
#include "CA.h"

//define states
enum {
    US_busy
}US_state_id;
//global pointer to function
void (*US_state)();

//APIs
void US_init();
STATE_define(US_busy);

#endif

```

- **US.c**

CA.h | DC.c | DC.h | US.h | US.c | CA.c

```c
//define system variables
unsigned int distance ;

//generate random number of distances
int generate_random(int low , int range,int count){

    int i ;
    int rand_num;
    for(i = 0; i < count; i++){
        rand_num = (rand()%(range-low+1)) + low;
    }
    return rand_num ;
}

void US_init(){
    //init US sensor by calling its driver
    printf("US init \n");
}
//define states of robot motor speed
STATE_define(US_busy){
    //state actions
    state_id = US_busy ;
    //read distance from US
    distance = generate_random(40,60,1);
    printf("US_busy state : distance = %d  \n",distance );
    read_US_distance(distance);
    US_state = STATE(US_busy);
```

- DC.h

DC.h | US.h | CA.h

```c
#ifndef _DC_H_
#define _DC_H_
#include "CA.h"
//define states
enum {
    DC_idle,
    DC_busy
}DC_state_id;
//global pointer to function
void (*DC_state)();

//APIs
void DC_init();
STATE_define(DC_idle);
STATE_define(DC_busy);

#endif
```

- DC.c

DC.h | DC.c | US.h | CA.h | CA.c | US.c

```c
//define system variables
unsigned int speed;


void DC_init(){
    //init DC motor by calling its driver
    printf("DC init \n");
}
//define states of robot motor speed
STATE_define(DC_idle){
    //state actions
    DC_state_id = DC_idle ;

    printf("DC idle state : speed = %d \n",speed );
}

STATE_define(DC_busy){
    //state actions
    DC_state_id = DC_busy ;
    DC_state = STATE(DC_idle);
    printf("DC busy state : speed = %d \n",speed );
}

void set_motor_speed(int s){
    speed = s;
    //set dc state to busy
    DC_state = STATE(DC_busy);
    printf("DC---------speed = %d------->CA\n",speed);
}
```

➢ Simulation code running:

Problems Tasks Console Properties Debugg

```
<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
```