UNIT 4

ASSIGNMENT LESSON 2

COLLISION AVOIDENCE PROJECT

- **Abstract**

The car robot takes the distance from it to an obstacle from an ultra sonic sensor and compares this distance to a threshold equals 50 cm.

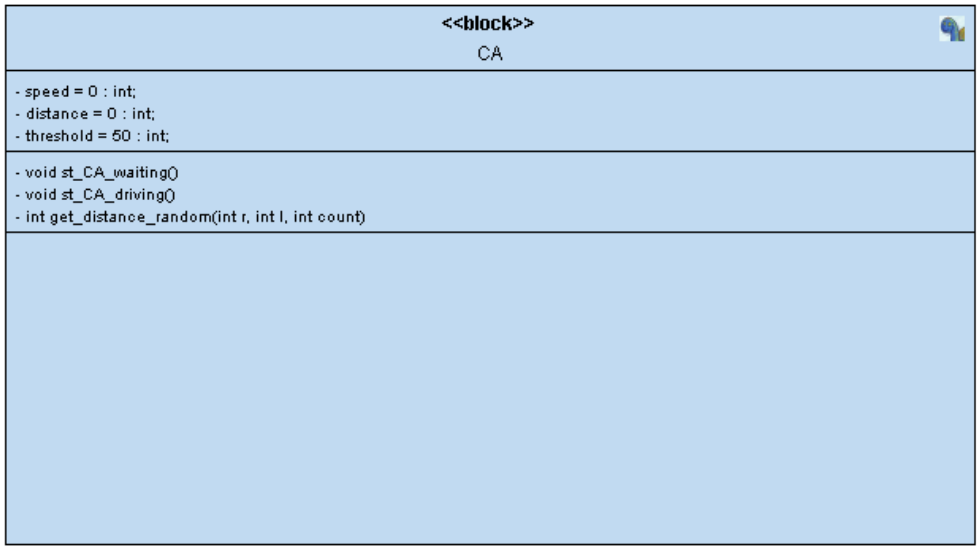
The car robot moves until the distance to the obstacle becomes equal or less than 50 cm then the robot stop moving.

- **STATE MACHINE**

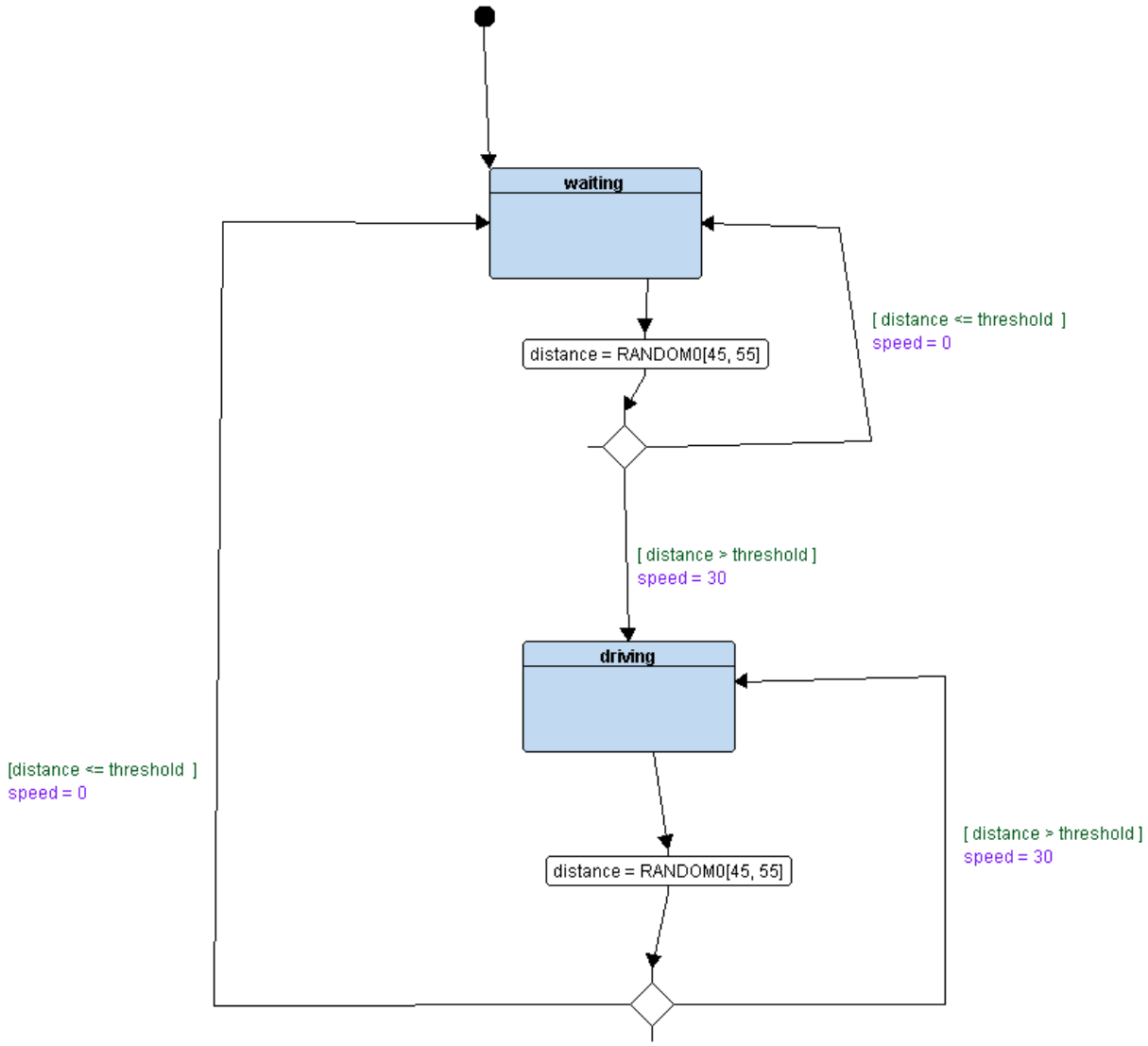
➢ **Using One Module**

Here I used one block to explain the system .

This block contains three global variables and two functions which describe the robot states.



- **State Diagram**

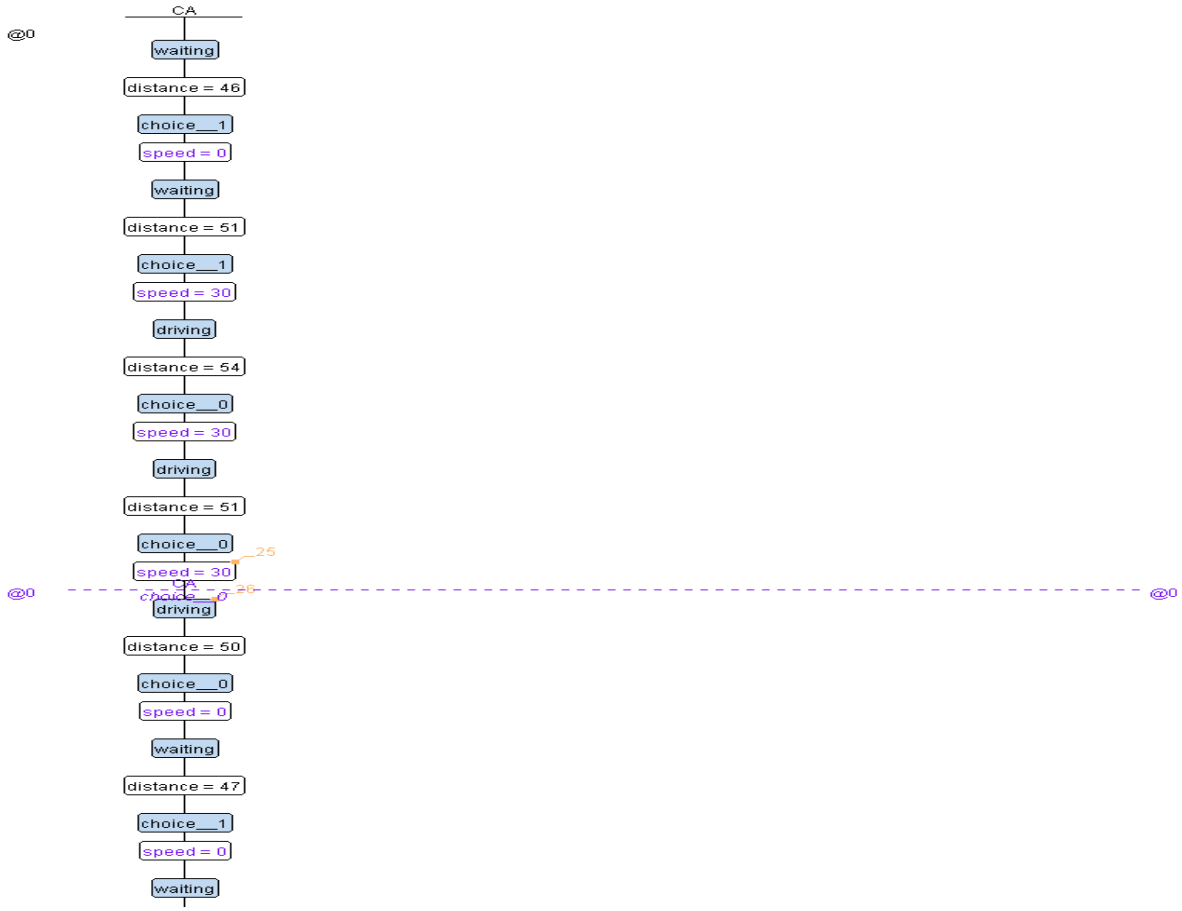


This diagram explains the two states of our system.

First state is waiting state in which the car robot stop moving until distance became greater than 50 cm , in this case the robot switchs from waiting state to driving state .

Second state is driving state in which the car robot moves with specific speed. when the distance became less than or equal 50 cm , the robot switchs from driving state to waiting state and stop moving.

- **Simulation**



- **C IMPLEMENTATION**
  - **State.h**

CA.c | CA.h | main.c | state.h

```
2  * state.h
7
8  #ifndef STATE_H_
9  #define STATE_H_
10
11
12 //automatic state function generate
13 #define state_define(_state_func_)      void st_##_state_func_()
14 #define state(_state_func_)             st_##_state_func_
15
16
17
18 #endif /* STATE_H_ */
19
```

  - **CA.h**

CA.c | CA.h | main.c | state.h

```
2  * CA.h
7
8  #ifndef CA_H_
9  #define CA_H_
10
11 #include "state.h"
12
13 //define states
14 enum
15 {
16     waiting,
17     driving
18 }state_id;
19
20
21 //functions prototypes
22 state_define(CA_waiting);
23 state_define(CA_driving);
24
25
26 //global pointer to function
27 extern void (*CA_state)();
28
29
30 #endif /* CA_H_ */
31
```

- **CA.c**

CA.c | CA.h | main.c | state.h

```c
/* CA.c...

#include <stdio.h>
#include <stdlib.h>
#include "CA.h"
#include "state.h"

int get_distance_random (int l, int r, int count);

//global variables
unsigned int speed = 0;
unsigned int distance = 0;
unsigned int threshold = 50;

//global pointer to function
void (*CA_state)();



state_define(CA_waiting)
{
    //state name
    state_id = waiting;
    //state action
    speed = 0;
    //generate random distance
    distance = get_distance_random (45, 55, 1);

    //check event
    (distance <= threshold) ? (CA_state = state(CA_waiting)) : (CA_state = state(CA_driving));
    printf("Waiting state : distance = %d speed = %d\n", distance, speed);
}


state_define(CA_driving)
{
    //state name
    state id = driving;
```

```c
state_define(CA_driving)
{
    //state name
    state_id = driving;
    //state action
    speed = 30;
    //generate random distance
    distance = get_distance_random (45, 55, 1);

    //check event
    (distance <= threshold) ? (CA_state = state(CA_waiting)) : (CA_state = state(CA_driving));
    printf("Driving state : distance = %d speed = %d\n", distance, speed);
}


int get_distance_random (int l, int r, int count)
{
    int i, rand_num;
    for(i=0; i<count; i++)
    {
        rand_num =( rand() % (r - l + 1) ) + l;
    }
    return rand_num;

}

```

- **Main.c**

CA.c | CA.h | main.c | state.h

```
/* main.c...

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


void setup ();


int main()
{
    volatile int d;

    setup();

    while(1)
    {
        //call state for each block
        CA_state();
        for(d=0; d<=1000; d++);
    }

    return 0;
}


void setup ()
{
    //initialize all drivers
    //initialize IRQ
    //initialize HAL US_DRIVER DC_DRIVER
    //set states pointers for each block
    CA_state = state(CA_waiting);
}

```

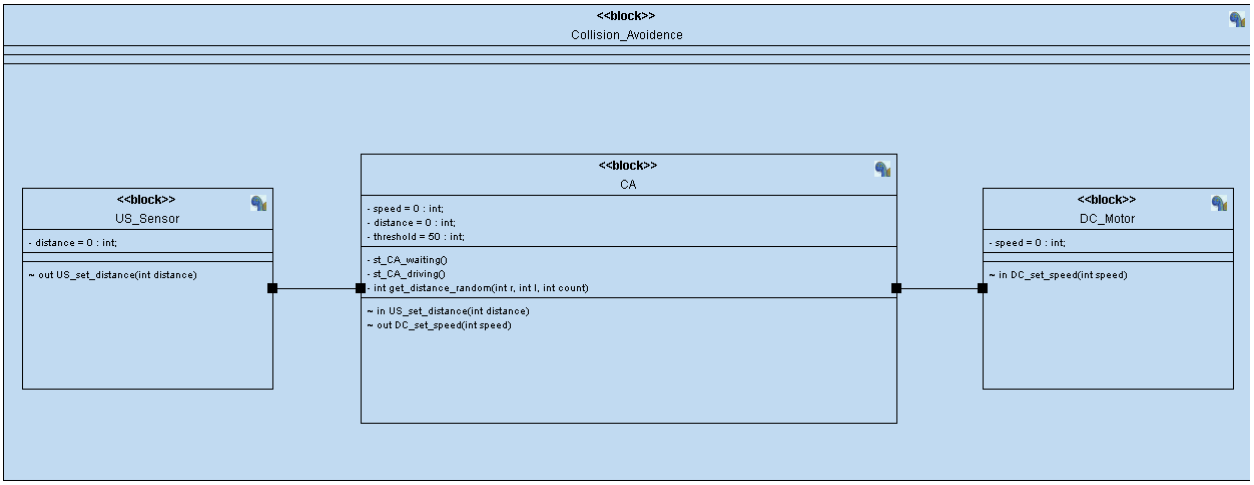
- **Code running**

```
Waiting state : distance = 48 speed = 0
Waiting state : distance = 47 speed = 0
Waiting state : distance = 48 speed = 0
Waiting state : distance = 50 speed = 0
Waiting state : distance = 46 speed = 0
Waiting state : distance = 47 speed = 0
Waiting state : distance = 47 speed = 0
Waiting state : distance = 47 speed = 0
Waiting state : distance = 54 speed = 0
Driving state : distance = 53 speed = 30
Driving state : distance = 55 speed = 30
Driving state : distance = 52 speed = 30
Driving state : distance = 53 speed = 30
Driving state : distance = 55 speed = 30
Driving state : distance = 55 speed = 30
Driving state : distance = 45 speed = 30
Waiting state : distance = 53 speed = 0
Driving state : distance = 53 speed = 30
Driving state : distance = 51 speed = 30
Driving state : distance = 54 speed = 30
Driving state : distance = 49 speed = 30
Waiting state : distance = 55 speed = 0
Driving state : distance = 54 speed = 30
Driving state : distance = 51 speed = 30
Driving state : distance = 55 speed = 30
Driving state : distance = 45 speed = 30
Waiting state : distance = 52 speed = 0
```

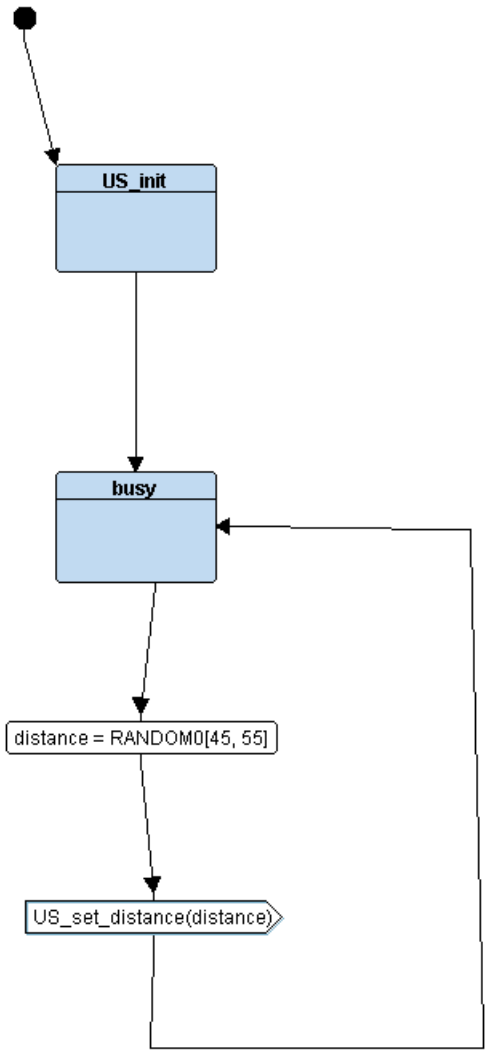
➢ Using Multiple Modules

Here I used three modules to describe the system.

One module for ultrasonic sensor, one for DC motor and one module for controlling and connecting these modules .



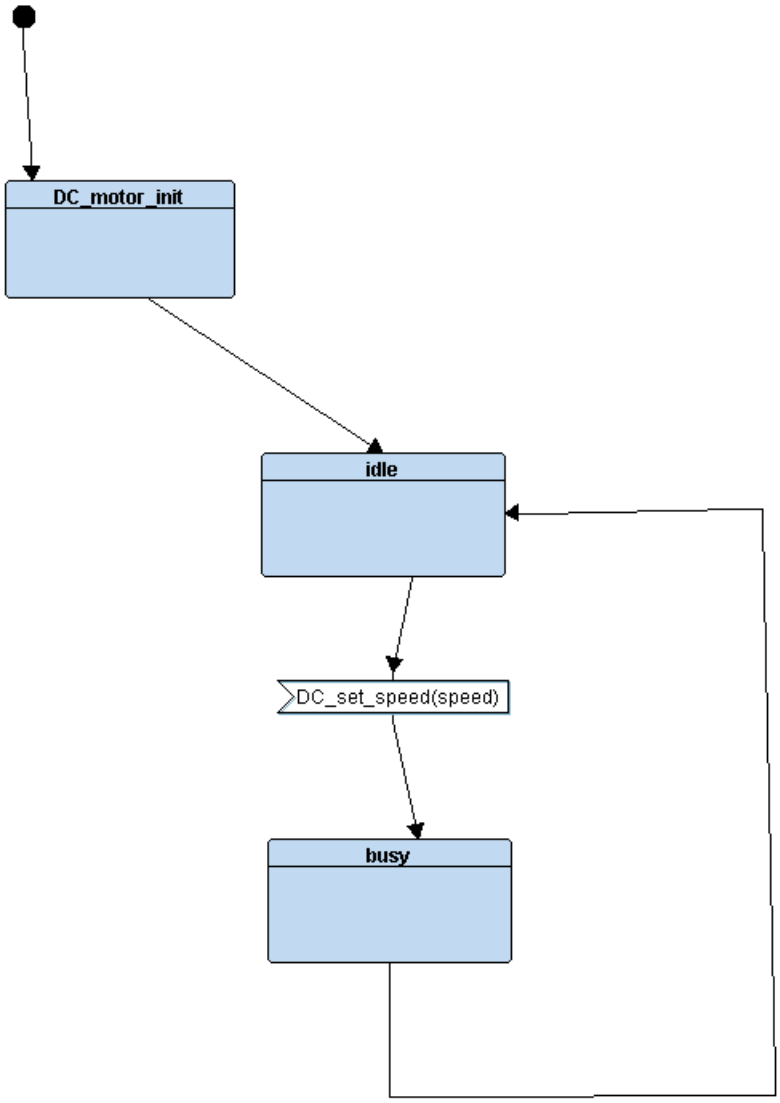
- **State Diagram**
  - **For ultrasonic sensor**



We first intialize the sensor then the sensor working in busy state .

In busy state the sensor reads distance and send it and return again to busy state.

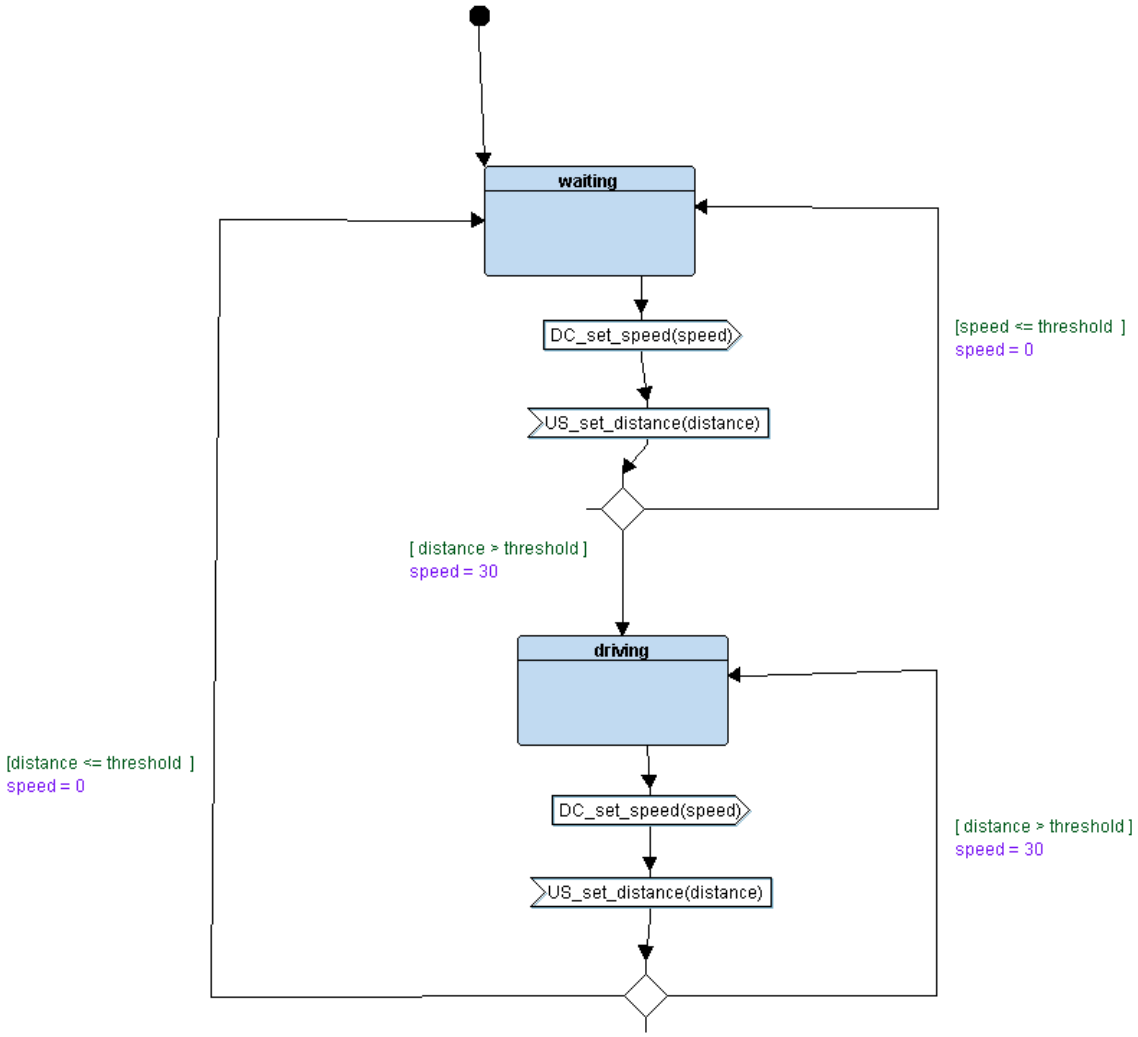
- **For DC Motor**



After intializing DC motor it goes to idle state in which motor doesnot move .

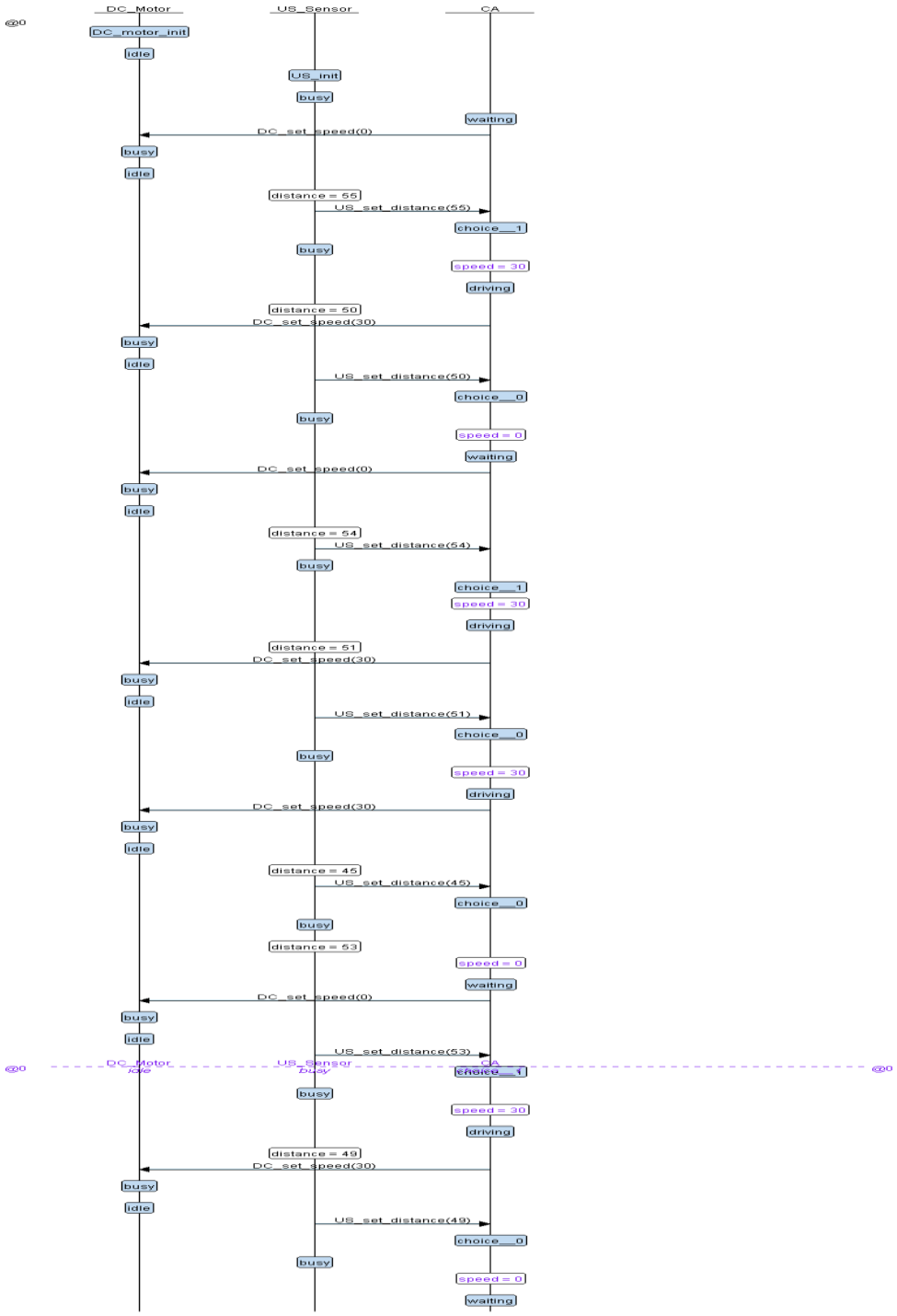
After comparing distance we send speed to DC motor and it goes to busy state and motor moves.

- **For collision avoidence module**



Here it is the same like one module diagram and the same two states driving and waiting, but here it takes distance from ultrasonic and send speed to dc motor after comparing distance .

- **Simulation**


DC_Motor
US_Sensor
CA
@0
DC_motor_init
idle
US_init
busy
waiting
DC_set_speed(0)
busy
idle
distance = 55
US_set_distance(55)
choice__1
busy
speed = 30
driving
distance = 50
DC_set_speed(30)
busy
idle
US_set_distance(50)
choice__0
busy
speed = 0
waiting
DC_set_speed(0)
busy
idle
distance = 54
US_set_distance(54)
busy
choice__1
speed = 30
driving
distance = 51
DC_set_speed(30)
busy
idle
US_set_distance(51)
choice__0
busy
speed = 30
driving
DC_set_speed(30)
busy
idle
distance = 45
US_set_distance(45)
choice__0
busy
distance = 53
speed = 0
waiting
DC_set_speed(0)
busy
idle
US_set_distance(53)
DC_Motor
idle
US_Sensor
busy
CA
choice__1
@0
@0
busy
speed = 30
driving
distance = 49
DC_set_speed(30)
busy
idle
US_set_distance(49)
choice__0
busy
speed = 0
waiting

- **C IMPLEMENTATION**

  - **State.h**

state.h

```
2  * state.h
7
8  #ifndef STATE_H_
9  #define STATE_H_
10
11
12 //automatic state function generate
13 #define STATE_DEFINE(_state_func_)      void st_##_state_func_()
14 #define STATE(_state_func_)             st_##_state_func_
15
16
17 //states connections
18 void US_set_distance(int d);
19 void DC_set_speed(int s);
20
21
22 #endif /* STATE_H_ */
23
```

  - **DC.h**

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

```
2  * DC.h
7
8  #ifndef DC_H_
9  #define DC_H_
10
11
12 #include "state.h"
13
14 //define states
15 enum
16 {
17     dc_idle,
18     dc_busy
19 }DC_state_id;
20
21
22 //functions prototypes
23 STATE_DEFINE(DC_idle);
24 STATE_DEFINE(DC_busy);
25
26 void DC_init();
27
28
29 //global pointer to function
30 extern void (*DC_state)();
31
32
33 #endif /* DC_H_ */
34
```

- **US.h**

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

```
 2  * US.h
 7
 8  #ifndef US_H_
 9  #define US_H_
10
11
12  #include "state.h"
13
14  //define states
15  enum
16  {
17      us_busy
18  }US_state_id;
19
20
21  //functions prototypes
22  STATE_DEFINE(US_busy);
23
24  void US_init();
25
26  //global pointer to function
27  extern void (*US_state)();
28
29
30  #endif /* US_H_ */
31
```

- **CA.h**

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

```
2  * CA.h
7
8  #ifndef CA_H_
9  #define CA_H_
10
11 #include "state.h"
12
13 //define states
14 enum
15 {
16     waiting,
17     driving
18 }state_id;
19
20
21 //functions prototypes
22 STATE_DEFINE(CA_waiting);
23 STATE_DEFINE(CA_driving);
24
25
26 //global pointer to function
27 extern void (*CA_state)();
28
29
30 #endif /* CA_H_ */
31
```

- **US.c**

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

```c
 * US.c

#include "stdio.h"
#include "stdlib.h"
#include "US.h"
#include "state.h"


int get_distance_random (int l, int r, int count);
void (*US_state)();


unsigned int US_distance = 0;


void US_init()
{
    printf("US init\n");
}



STATE_DEFINE(US_busy)
{
    //state name
    US_state_id = us_busy;

    //state action
    US_distance = get_distance_random (45, 55, 1);
    printf("US busy state : distance = %d \n", US_distance);
    US_set_distance(US_distance);
    US_state = STATE(US_busy);

}


int get_distance_random (int l, int r, int count)
{
```

```c

int get_distance_random (int l, int r, int count)
{
    int i, rand_num;
    for(i=0; i<count; i++)
    {
        rand_num =( rand() % (r - l + 1) ) + l;
    }
    return rand_num;

}

```

- **DC.c**

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

```c
#include "stdio.h"
#include "DC.h"
#include "state.h"

void (*DC_state)();
unsigned int DC_speed = 0;

void DC_init()
{
    printf("DC init\n");
}


void DC_set_speed(int s)
{
    DC_speed = s;
    DC_state = STATE(DC_busy);
    printf("CA----------speed = %d-------------->DC\n", DC_speed);

}


STATE_DEFINE(DC_idle)
{
    //state name
    DC_state_id = dc_idle;
    //call pwm
    printf("DC_idle state : speed = %d \n", DC_speed);

}


STATE_DEFINE(DC_busy)
{
    //state name
    DC_state_id = dc_busy;

    //state action
    DC_state = STATE(DC_idle);
    //call PWM to make speed = DC_speed
    printf("DC_busy state : speed = %d \n", DC_speed);

}
```

- **CA.c**

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

```c
/*
 * CA.c
 */

#include <stdio.h>
#include <stdlib.h>
#include "CA.h"
#include "state.h"


//global variables
unsigned int speed = 0;
unsigned int distance = 0;
unsigned int threshold = 50;

//global pointer to function
void (*CA_state)();



STATE_DEFINE(CA_waiting)
{
    //state name
    state_id = waiting;
    printf("CA Waiting state : distance = %d speed = %d\n", distance, speed);
    //state action
    speed = 0;
    DC_set_speed(speed);

}


STATE_DEFINE(CA_driving)
{
    //state name
    state_id = driving;
    printf("CA Driving state : distance = %d speed = %d\n", distance, speed);
    //state action
    speed = 30;
    DC_set_speed(speed);
}


void US_set_distance(int d)
{
    distance = d;
    //check event
    (distance <= threshold) ? (CA_state = STATE(CA_waiting)) : (CA_state = STATE(CA_driving));
    printf("US-----------distance = %d-------------->CA\n", distance);
}
```

- **Main.c**

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

```c
#include <stdio.h>
#include <stdlib.h>
#include "CA.h"
#include "US.h"
#include "DC.h"

void setup ()
{
    //initialize all drivers
    //initialize IRQ
    //initialize HAL US_DRIVER DC_DRIVER
    //initialize 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);

}


int main()
{
    volatile int d;

    setup();

    while(1)
    {
        //call state for each block
        US_state();
        CA_state();
        DC_state();
        for(d=0; d<=1000; d++);
    }

    return 0;
```

- **Code Running**

```
US init
DC init
US busy state : distance = 53
US----------distance = 53-------------->CA
CA Driving state : distance = 53 speed = 0
CA----------speed = 30-------------->DC
DC_busy state : speed = 30
US busy state : distance = 54
US----------distance = 54-------------->CA
CA Driving state : distance = 54 speed = 30
CA----------speed = 30-------------->DC
DC_busy state : speed = 30
US busy state : distance = 54
US----------distance = 54-------------->CA
CA Driving state : distance = 54 speed = 30
CA----------speed = 30-------------->DC
DC_busy state : speed = 30
US busy state : distance = 46
US----------distance = 46-------------->CA
CA Waiting state : distance = 46 speed = 30
CA----------speed = 0-------------->DC
DC_busy state : speed = 0
US busy state : distance = 52
```