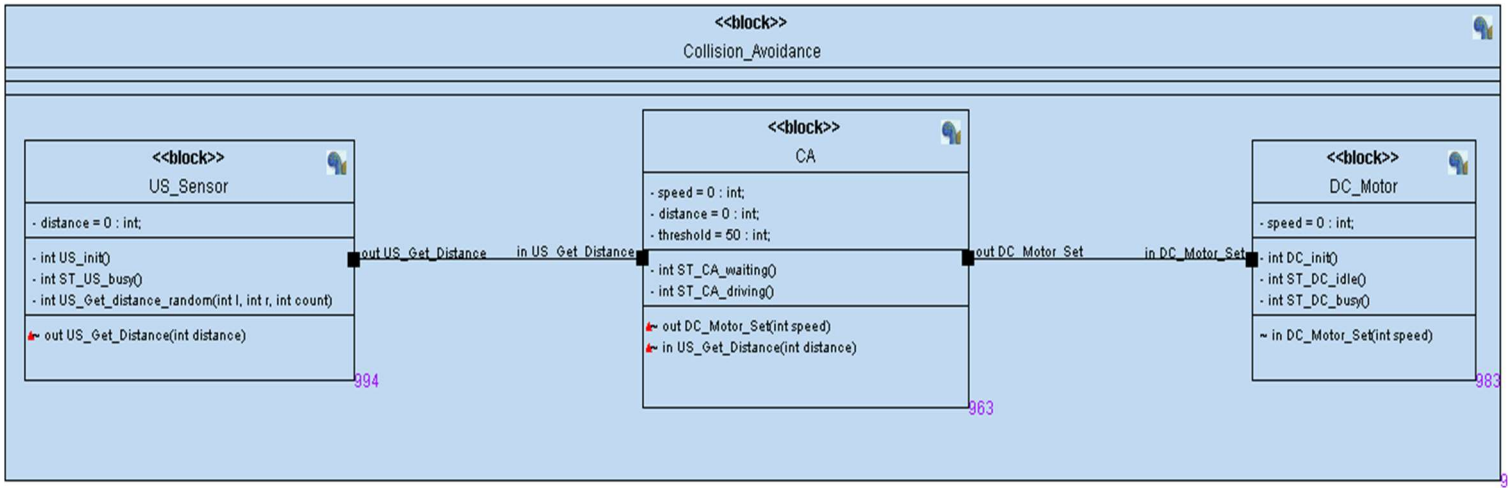
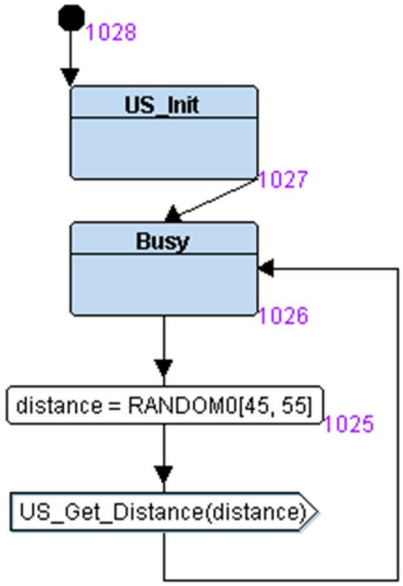
Simple state machine Implementation of ULTRASONIC OBSTACLE AVOIDING robot in C using multiple modules

Modules level:

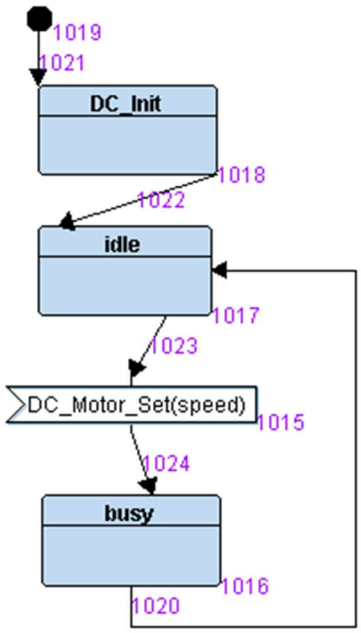


Logical design:

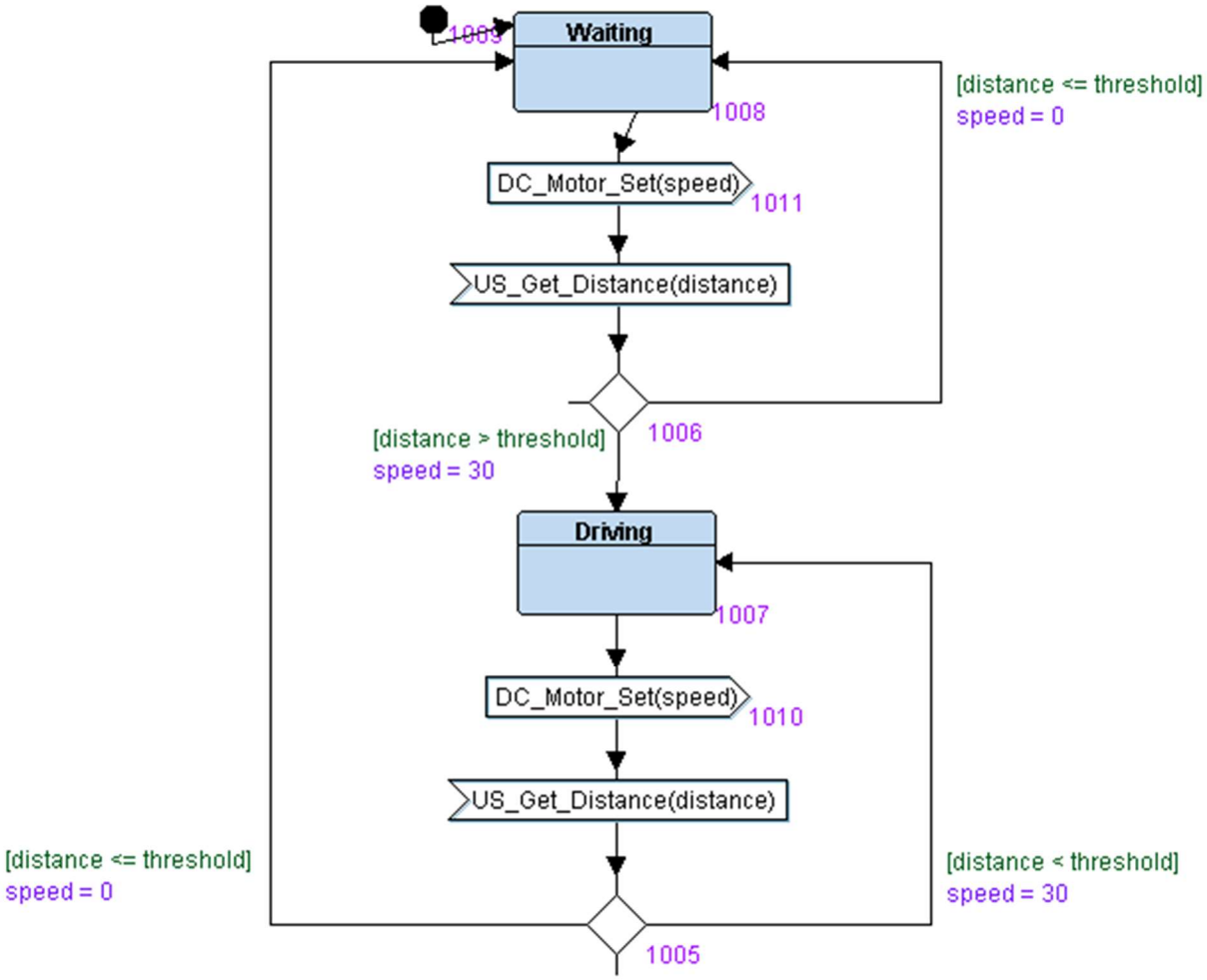
Ultrasonic sensor module:



DC motor module:



Collision avoidance module:



C implementation:

- main.c

```c
/* main.c */
#include "CA.h"
#include "DC.h"
#include "US.h"
#include "state.h"

void setup ()
{
    /* Init BLOCKs */
    DC_init();
    US_init();
    /* Set states pointers for each block */
    CA_state = STATE (CA_waiting);
    DC_state = STATE (DC_idle);
    US_state = STATE (US_busy);
}

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

    while(1)
    {
        /* Call state for each block */
        US_state(); CA_state(); DC_state();
        /* Delay */
        for (d = 0; d <= 1000; d++);
    }
    return 0;
}
```

- state.h

```c
/* state.h */
#ifndef STATE_H_
#define STATE_H_

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

/* Automatic state function generated */
#define STATE_define(_statefun_)  void ST_##_statefun_()
#define STATE(_statefun_) ST_##_statefun_


/* States connection */
void US_Get_Distance (int d); /* Argument: distance */
void DC_Motor_Set(int s);     /* Argument: speed */


#endif
```

- US_Sensor.c

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

/* Module variables */
int US_distance = 0;

/* State Pointer (pointer to function) */
void (*US_state)();

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

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

STATE_define(US_busy)
{
    US_state_id = US_busy;
    US_distance = US_Get_distance_random(45, 55, 1);
    printf("US_busy state: distance = %d\n", US_distance);
    US_Get_Distance (US_distance);
    US_state = STATE(US_busy);
}

int US_Get_distance_random(int l, int r, int count)
{
    /* this code will generate random number in range l and r */
    int i;
    int rand_num;
    for (i = 0; i < count; i++) {   rand_num = (rand() % (r - l + 1)) + l;  }
    return rand_num;
}
```

- US_Sensor.h

```c
/* US.h */
#ifndef US_H_
#define US_H_

#include "state.h"

/* Define states */
enum
{
    US_busy
} US_state_id;

/* Declare states functions US */
void US_init();
STATE_define(US_busy);

/* State Pointer (pointer to function) */
extern void (*US_state)();

#endif
```

- CA.c

```c
/* CA.c */
#include "CA.h"
#include "state.h"

/* Module variables */
int CA_speed = 0;
int CA_distance = 0;
int CA_threshold = 50;

/* State Pointer (pointer to function) */
void (*CA_state)();

void US_Get_Distance (int d)
{
    CA_distance = d;
    (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)
{
    CA_state_id = CA_waiting;
    printf("CA_waiting state: distance = %d  speed = %d\n", CA_distance, CA_speed);
    CA_speed = 0;
    DC_Motor_Set(CA_speed);
}

STATE_define(CA_driving)
{
    CA_state_id = CA_driving;
    printf("CA_driving state: distance = %d  speed = %d\n", CA_distance, CA_speed);
    CA_speed = 30;
    DC_Motor_Set(CA_speed);
}
```

- CA.h

```c
/* CA.h */
#ifndef CA_H_
#define CA_H_

#include "state.h"

/* Define states */
enum
{
    CA_waiting,
    CA_driving
} CA_state_id;

/* Declare states functions CA */
STATE_define(CA_waiting);
STATE_define(CA_driving);

/* State Pointer (pointer to function) */
extern void (*CA_state)();

#endif
```

- DC_Motor.c

```c
/* DC.c */
#include "DC.h"
#include "state.h"

/* Module variables */
int DC_speed = 0;

/* State Pointer (pointer to function) */
void (*DC_state)();

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

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

STATE_define(DC_idle)
{
    DC_state_id = DC_idle;
    printf("DC_idle state: speed = %d\n", DC_speed);
}

STATE_define(DC_busy)
{
    DC_state_id = DC_busy;
    DC_state = STATE(DC_idle);
    printf("DC_busy state: speed = %d\n", DC_speed);
}
```

- DC_Motor.h

```c
/* DC.h */
#ifndef DC_H_
#define DC_H_

#include "state.h"

/* Define states */
enum
{
    DC_idle,
    DC_busy
} DC_state_id;

/* Declare states functions DC */
void DC_init ();
STATE_define(DC_idle);
STATE_define(DC_busy);

/* State Pointer (pointer to function) */
extern void (*DC_state)();

#endif
```