Unit (4) _Lesson (2)

Case Study:

Specification (from the client):

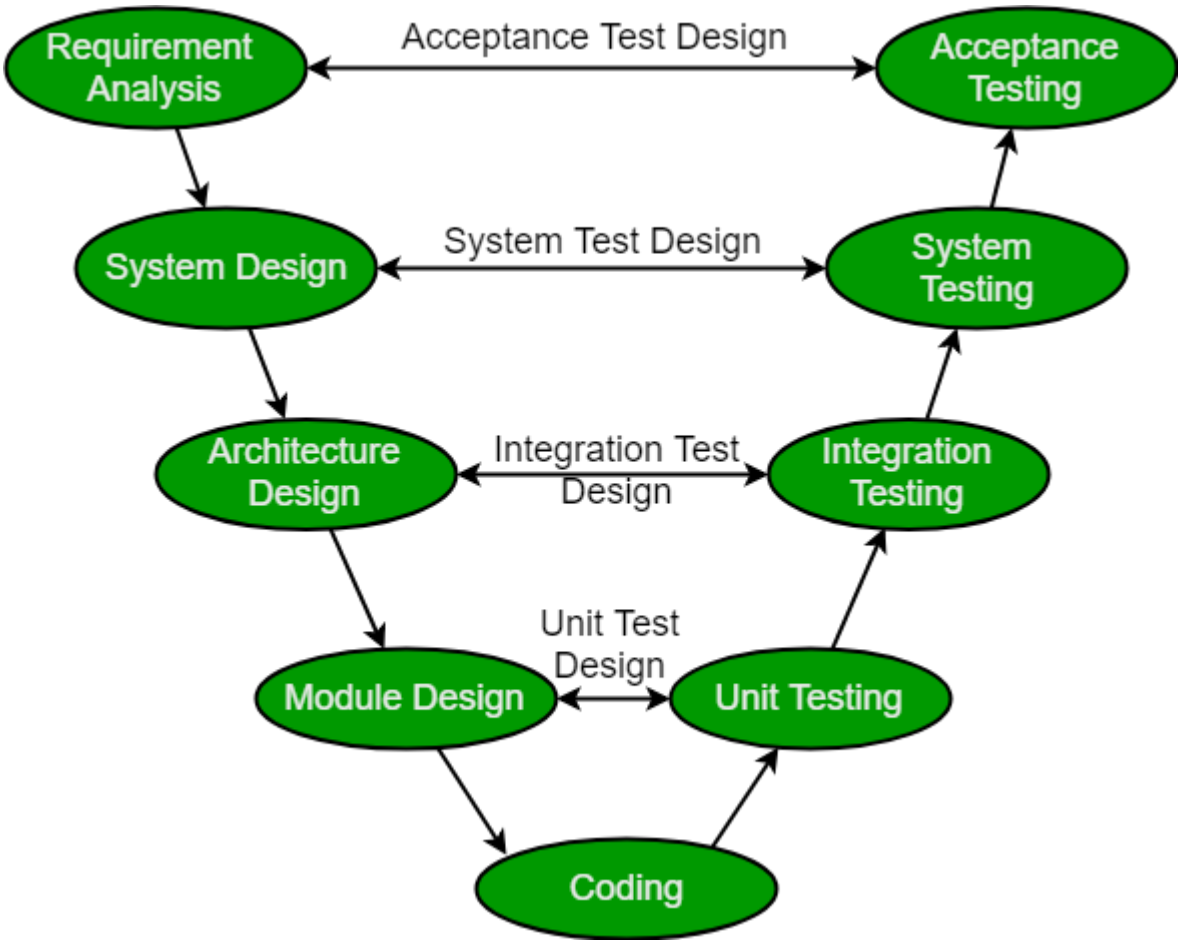
A system protects mobile robot from hitting.

System assumptions:

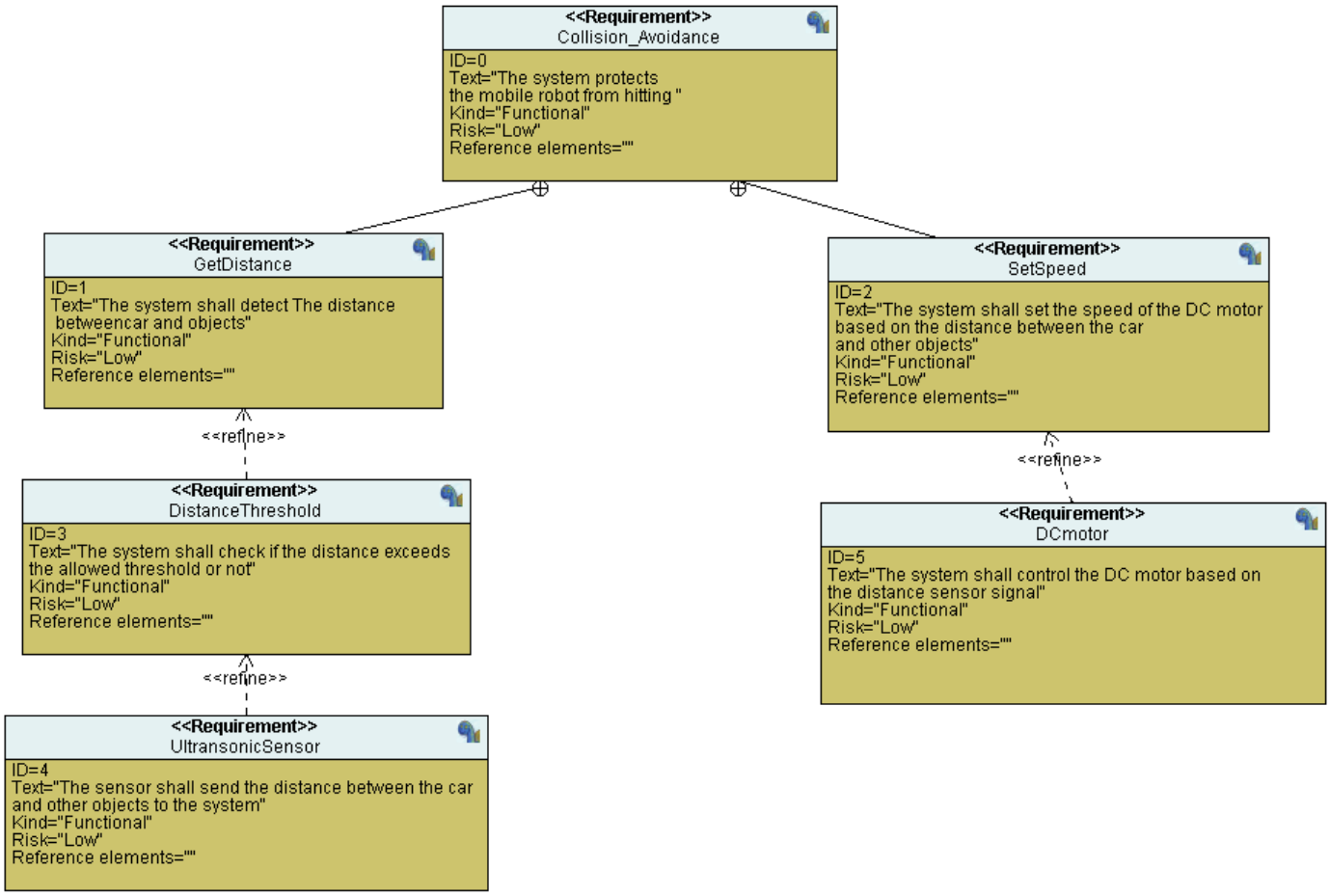
- The Ultrasonic sensor never fails.
- The DC motor never faces power cut.
- The controller maintenance is not modelled.
- The controller never faces power cut.

The method:

The V-model is a type of SDLC model where process executes in a sequential manner in V-shape. It is also known as Verification and Validation model. It is based on the association of a testing phase for each corresponding development stage. Development of each step is directly associated with the testing phase. The next phase starts only after completion of the previous phase for each development activity, there is a testing activity corresponding to it.



Requirements:

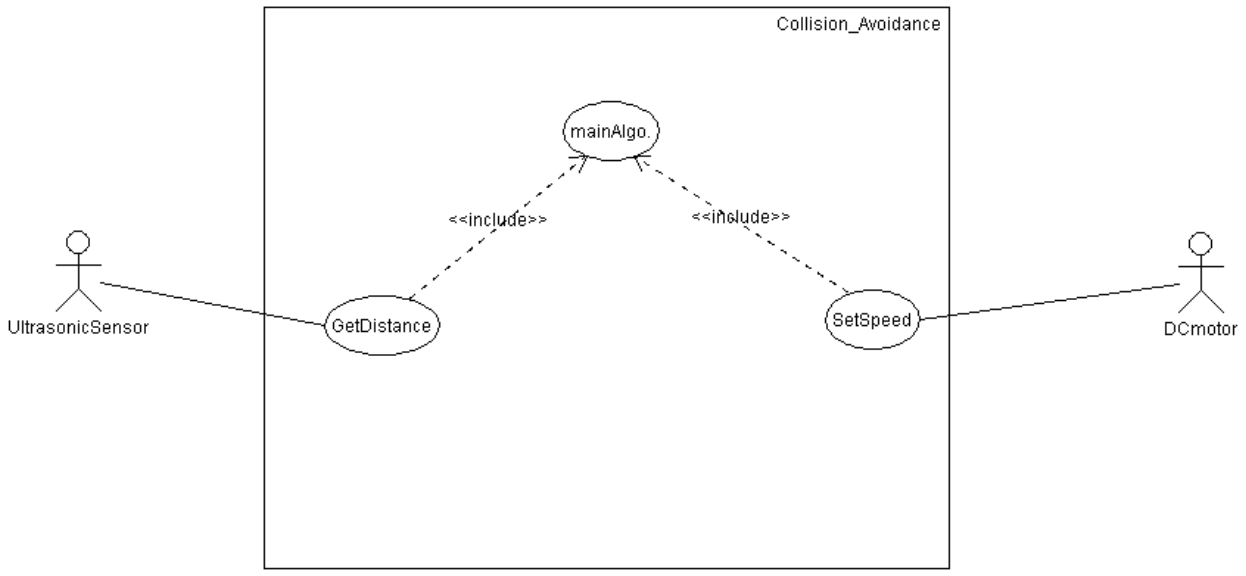


Design Space Exploration:

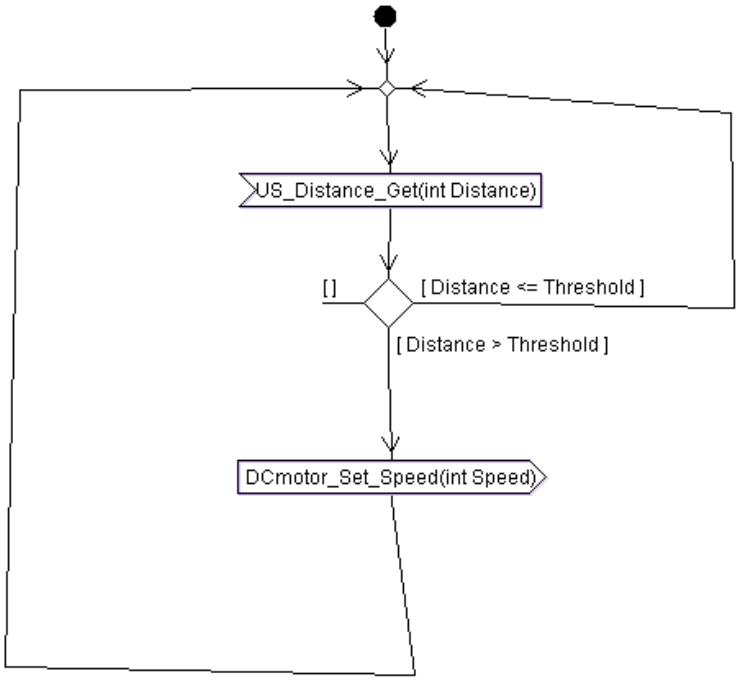
In this project the most optimal way is to use Stm32 which based on cortex m4 processor.

System Analysis:

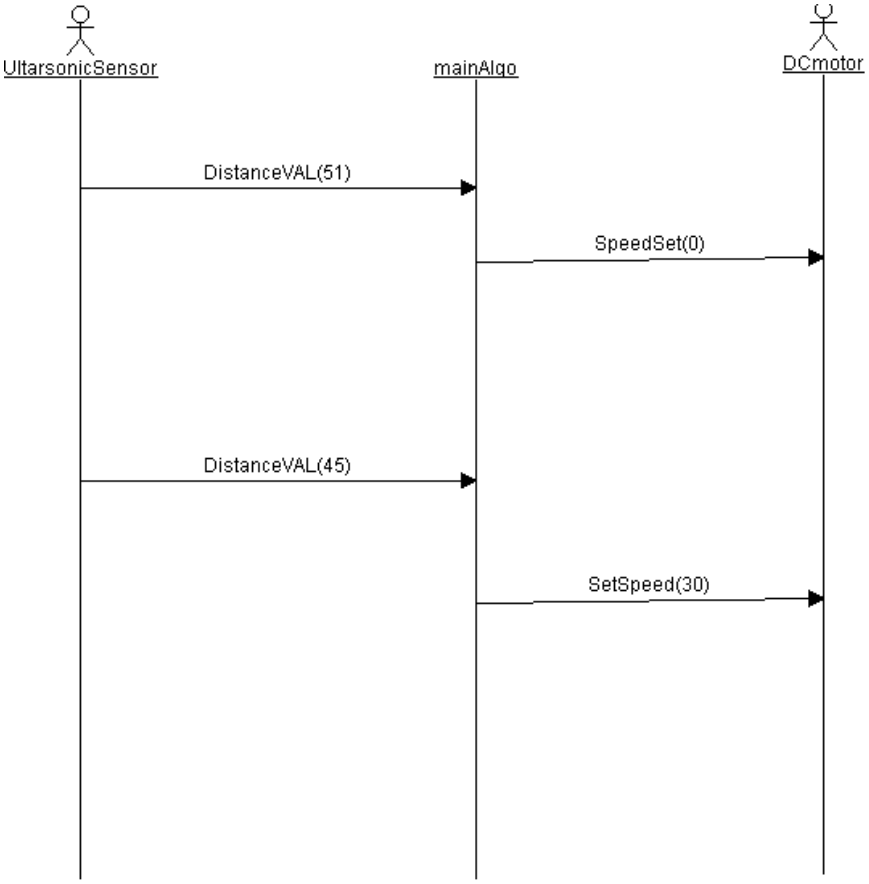
Use Case Diagram:



Activity Diagram:



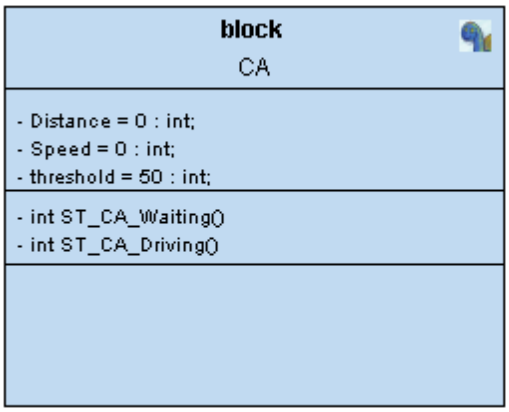
Sequence Diagram:



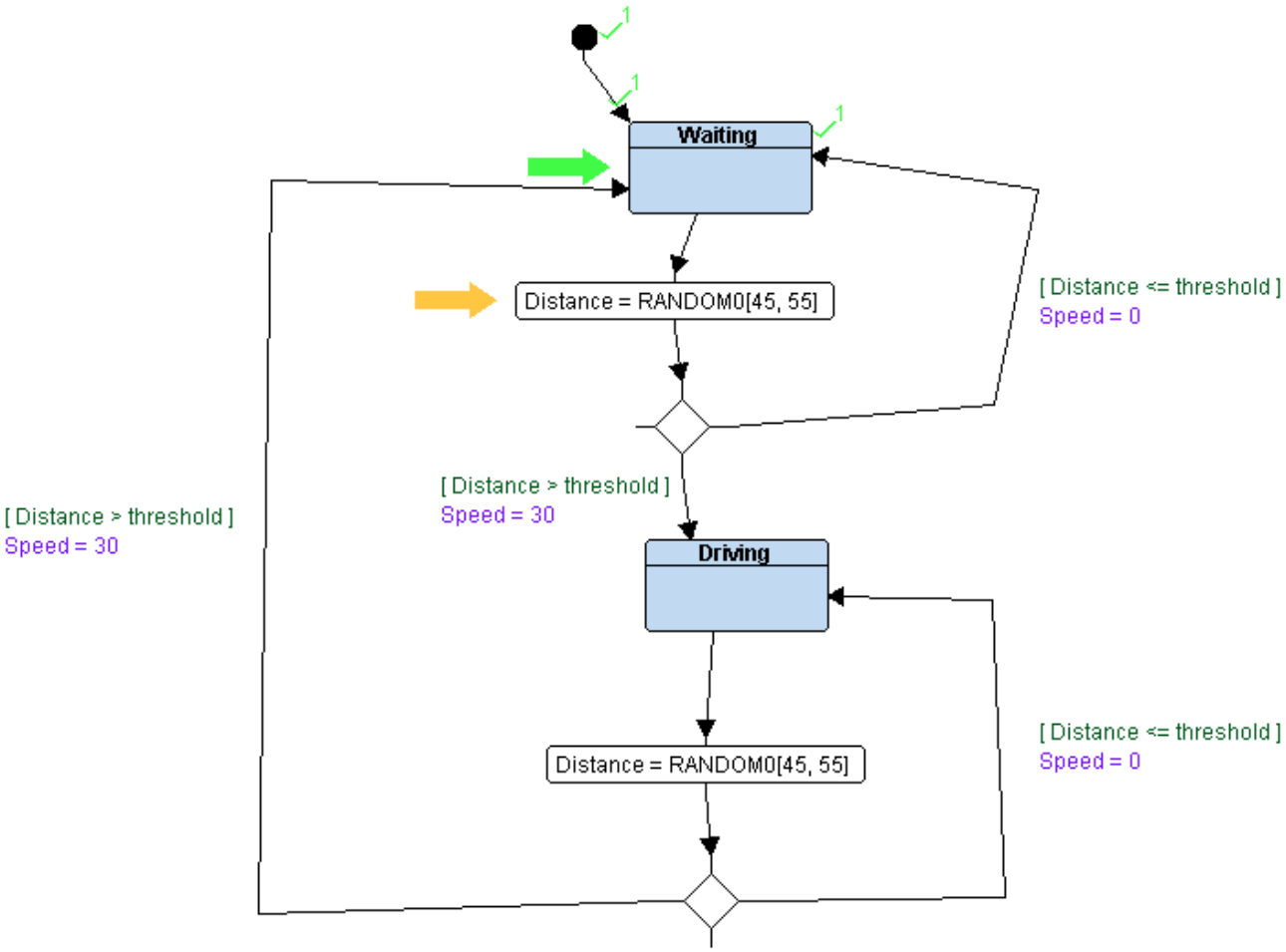
Block Diagram:

Using one module

Here I used one block to explain the system. This block contains three global variables and two functions which describe the machine states.



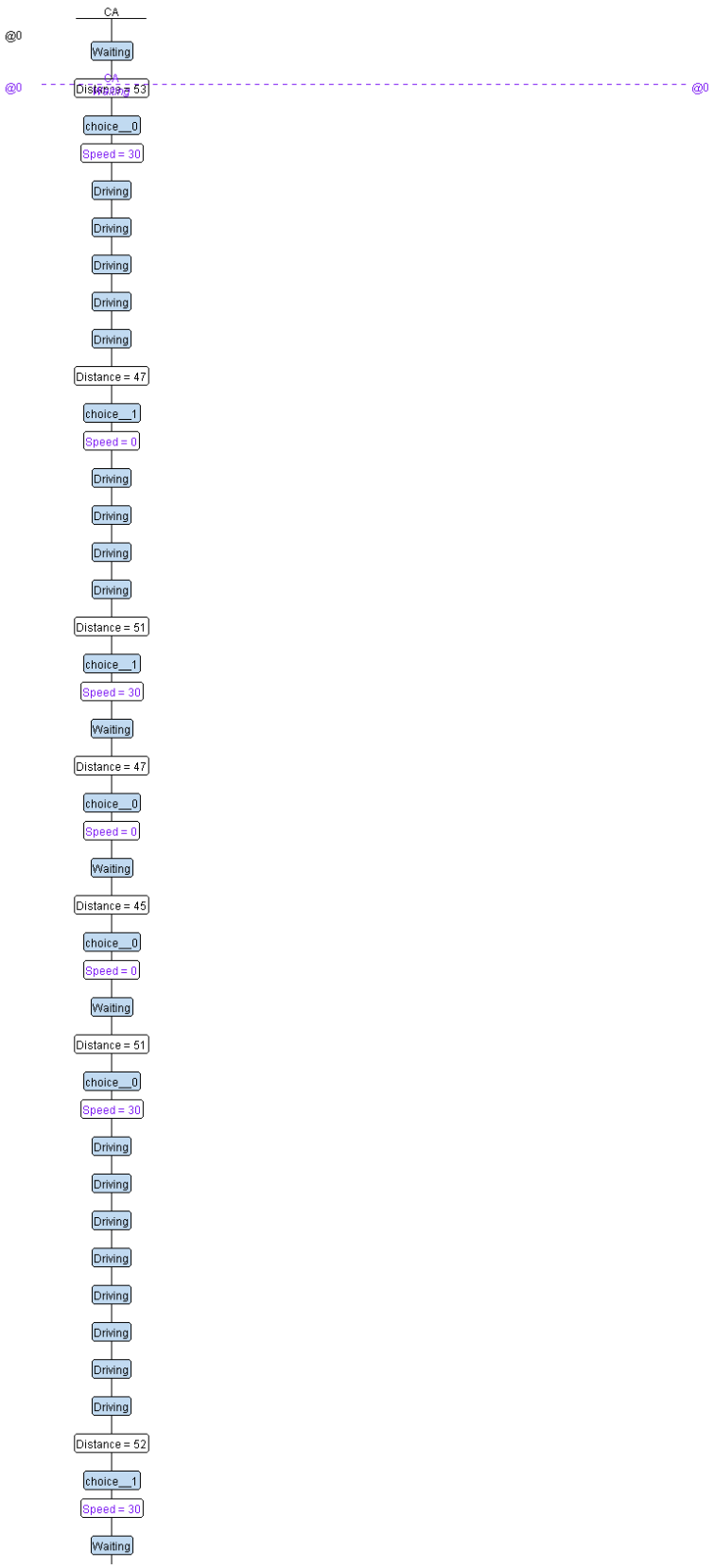
State Diagram:



This diagram explains the two states of our system. First state is waiting state in which the mobile robot stop moving until distance became greater than 50 cm, in this case the robot switches from waiting state to driving state.

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

Simulation:



C code:

State.h:

```
 2⊕ * state.h..
 7
 8 #ifndef STATE_H_
 9 #define STATE_H_
10
11 #define state_define(_stateFunc_)  void ST_##_stateFunc_()
12 #define state(_stateFunc_)    ST_##_stateFunc_
13
14
15
16 #endif /* STATE_H_ */
17
```

oneModule.h:

```
 2⊕ * OneModuleCA.h..
 7
 8 #ifndef ONEMODULECA_H_
 9 #define ONEMODULECA_H_
10
11⊖#define DPRINTF(...)    {fflush(stdout); \
12                          fflush(stdout); \
13                          printf(__VA_ARGS__); \
14                          fflush(stdout); \
15                          fflush(stdout);}
16
17⊖enum{
18      CA_Waiting,
19      CA_Driving
20 }CA_State;
21
22 void (*State)();
23
24 state_define(CA_Driving);
25 state_define(CA_Waiting);
26
27
28
29
30
31 #endif /* ONEMODULECA_H_ */
32
```

oneModule.c:

```c
#include <stdio.h>
#include <stdlib.h>
#include "state.h"
#include "OneModuleCA.h"

int Speed = 0;
int Distance = 0;
int threshold = 50;

extern void (*State)();


void setUp(){

    State = state(CA_Waiting);

}


state_define(CA_Waiting){
    // state name

    CA_State = CA_Waiting ;

    // action
    Speed = 0;
    Distance = generateRandom( 45 , 55 , 1);

    // check
    (Distance <= threshold) ? (State = state(CA_Waiting)) : (State = state(CA_Driving));
    DPRINTF("Waiting state: Speed: %d   Distance= %d\n",Speed,Distance)
}
state_define(CA_Driving){
    // state name

    CA_State = CA_Waiting ;

    // action
    Speed = 30;
    Distance = generateRandom( 45 , 55 , 1);

    // check
    (Distance > threshold) ? (State = state(CA_Driving)) : (State = state(CA_Waiting)
    DPRINTF("Driving state: Speed: %d   Distance= %d\n",Speed,Distance)
}


int generateRandom(int min, int max) {

  return min + rand() % (max - min + 1);
}
```

main.c:

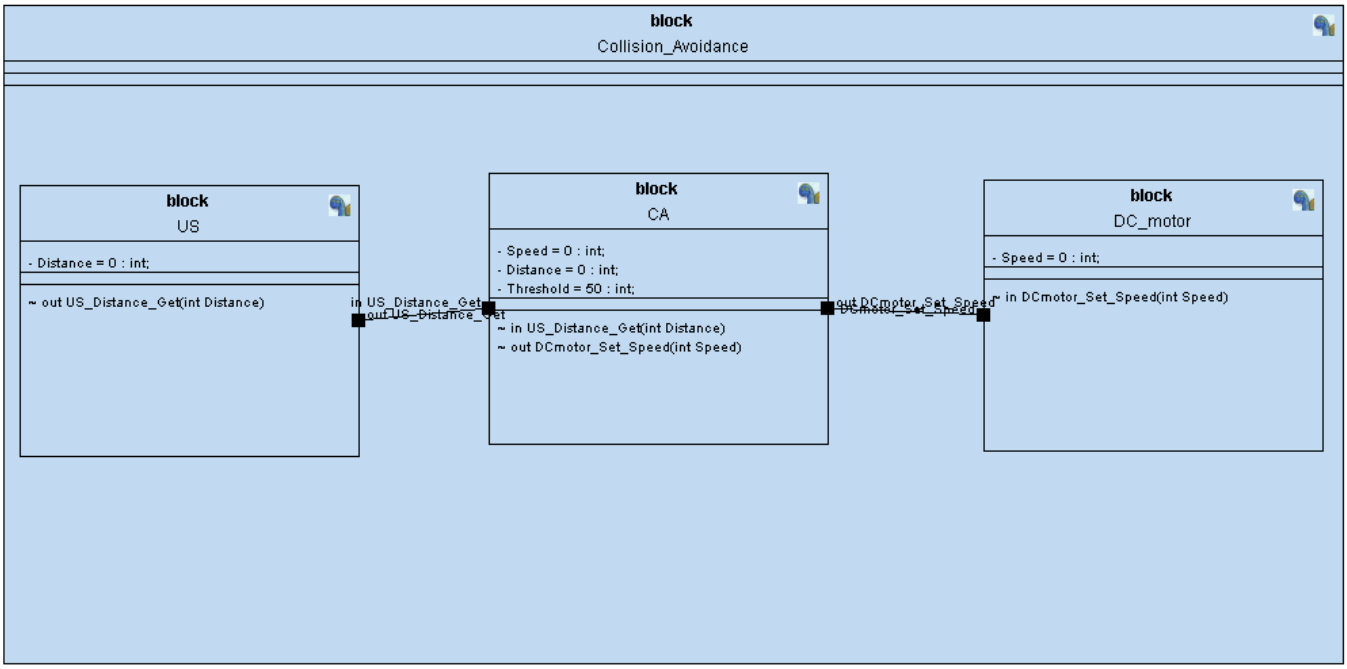
```c

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


int main(void) {
    setUp();
    int i;
    while(1){
        State();
        for(i=0 ; i<20000 ; i++);
    }


    return EXIT_SUCCESS;
}

```

Output:

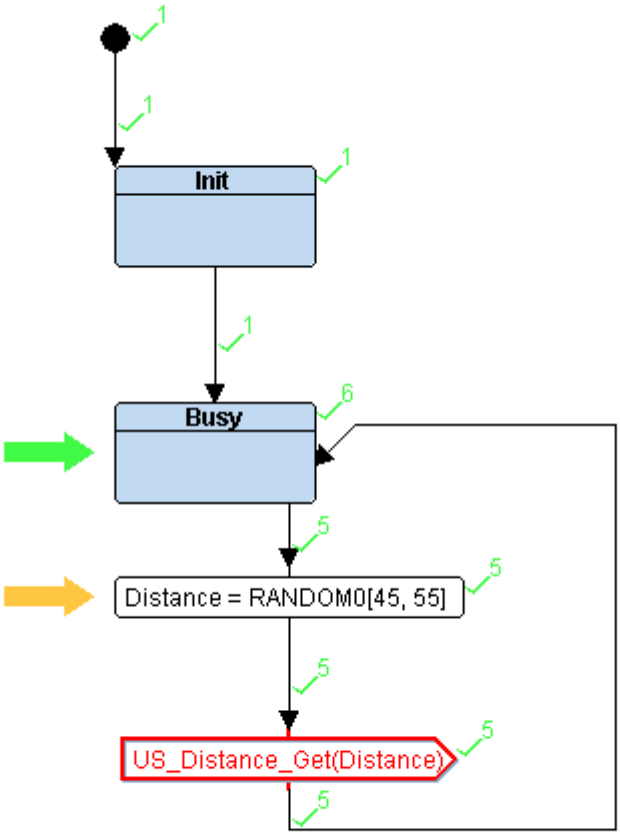
```
OneModuleCA.exe [C/C++ Application] F:\Handsa\Embedded courses\Learn in depth\I
Waiting state: Speed: 0   Distance= 53
Driving state: Speed: 30   Distance= 53
Driving state: Speed: 30   Distance= 48
Waiting state: Speed: 0   Distance= 48
Waiting state: Speed: 0   Distance= 45
Waiting state: Speed: 0   Distance= 49
Waiting state: Speed: 0   Distance= 53
Driving state: Speed: 30   Distance= 55
Driving state: Speed: 30   Distance= 47
Waiting state: Speed: 0   Distance= 52
Driving state: Speed: 30   Distance= 47
Waiting state: Speed: 0   Distance= 48
Waiting state: Speed: 0   Distance= 45
Waiting state: Speed: 0   Distance= 53
Driving state: Speed: 30   Distance= 47
Waiting state: Speed: 0   Distance= 49
Waiting state: Speed: 0   Distance= 54
Driving state: Speed: 30   Distance= 54
Driving state: Speed: 30   Distance= 53
Driving state: Speed: 30   Distance= 54
Driving state: Speed: 30   Distance= 55
Driving state: Speed: 30   Distance= 54
Driving state: Speed: 30   Distance= 47
Waiting state: Speed: 0   Distance= 52
Driving state: Speed: 30   Distance= 48
```

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.

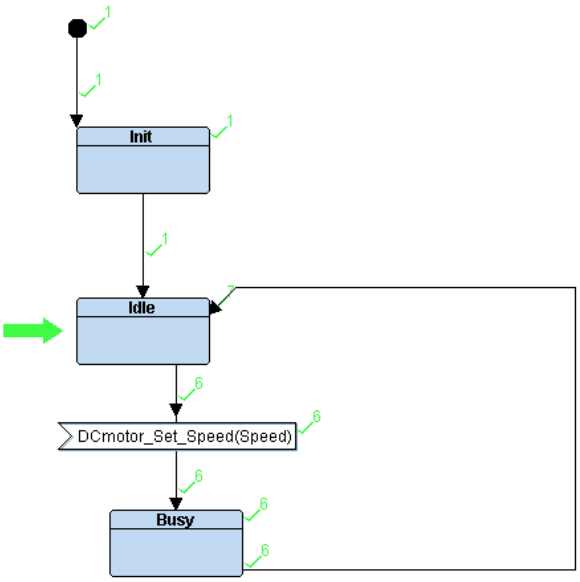


State Diagram: for ultrasonic sensor



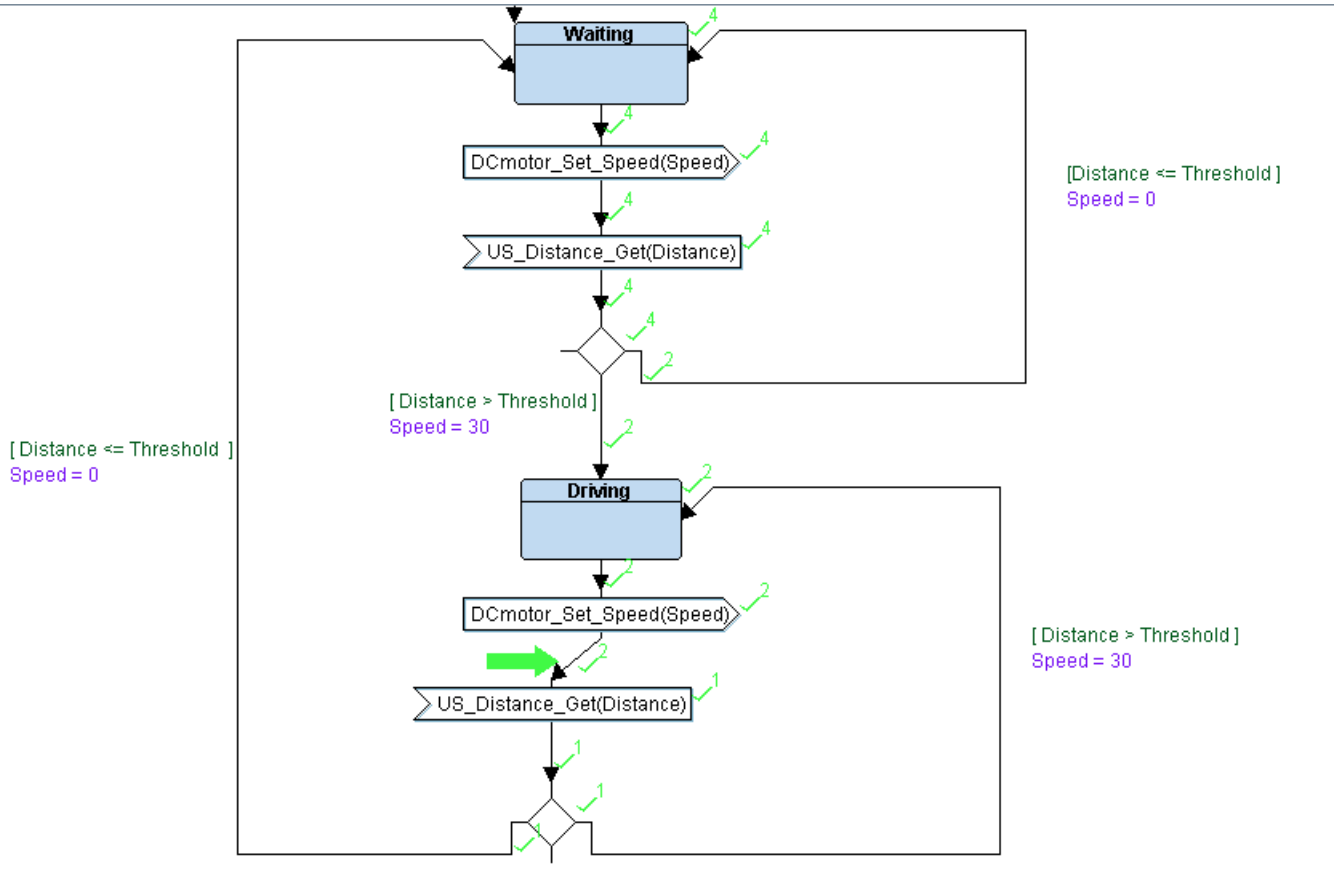
We first initialize the sensor then the sensor working in busy state. In busy state the sensor reads distance and send it and return to busy state.

For DC Motor:



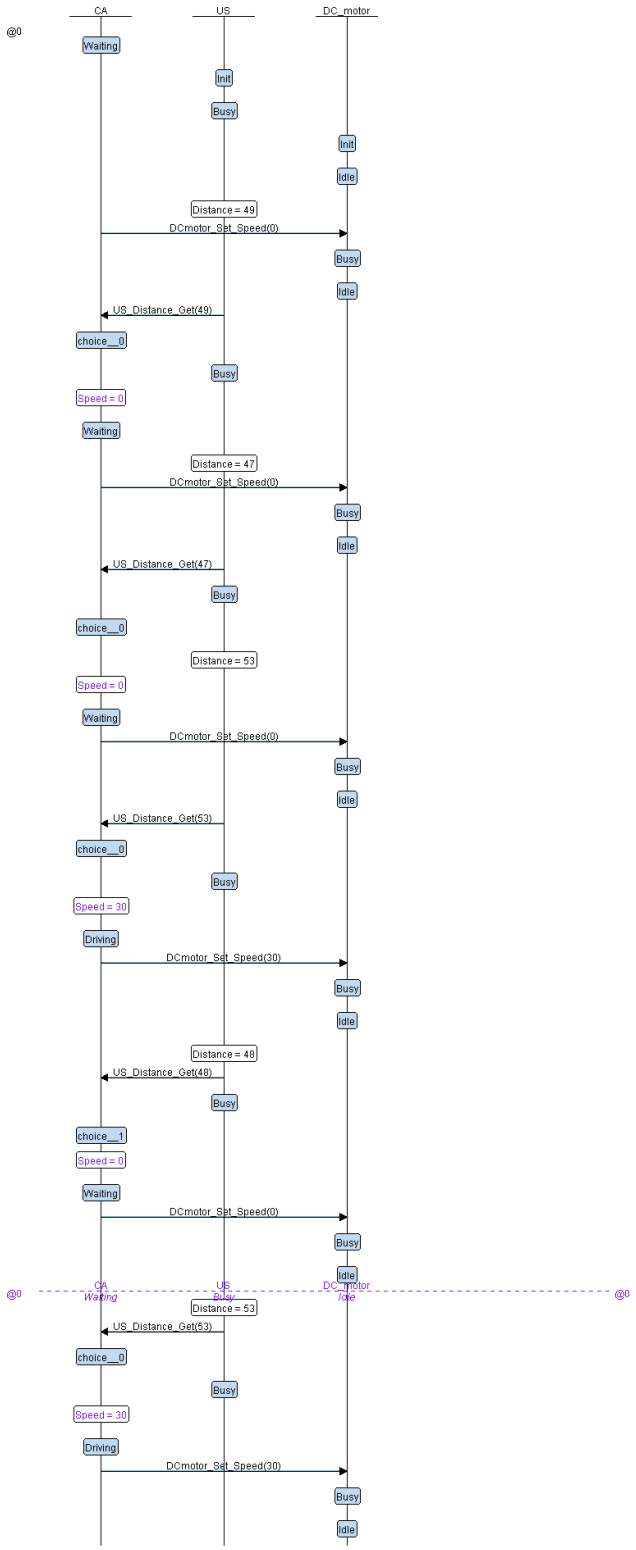
After initializing DC motor it goes to idle state in which motor does not move . After comparing distance, we send speed to DC motor and it goes to busy state and motor moves.

For collision avoidance module:



Here it is the same as 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:



C Code:

State.h:

```
 2⊕ * state.h
 7
 8 #ifndef STATE_H_
 9 #define STATE_H_
10
11 #define state_define(_stateFunc_)  void ST_##_stateFunc_()
12 #define state(_stateFunc_)    ST_##_stateFunc_
13
14
15
16
17 #endif /* STATE_H_ */
18
```

US.h:

```
 7
 8 #ifndef US_H_
 9 #define US_H_
10
11 #include "state.h"
12
13
14 enum{
15     US_Busy,
16 }US_State_id;
17
18 void (*US_State)();
19
20 void US_init();
21 state_define(US_busy);
22
23
24 #endif
25
```

US.c:

```c

#include "US.h"

unsigned int Distance = 0;
extern void (*US_State)();

int generateRandom(int min, int max) {

  return min + rand() % (max - min + 1);
}


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

state_define(US_busy){
    // state name
    US_State_id = US_Busy;

    // action
    Distance = generateRandom(45,55);
    printf("US Busy state :  Distance = %d\n",Distance);

    US_Distance_Get(Distance);
    US_State = state(US_busy);

}
```

DC_Motor.h:

```c
 * DC_Motor.h

#ifndef DC_MOTOR_H_
#define DC_MOTOR_H_

#include "state.h"

enum{
    DC_idle,
    DC_busy
}DC_State_id;

void (*DC_State)();

void DC_init();
state_define(DC_Idle);
state_define(DC_Busy);


#endif /* DC_MOTOR_H_ */

```

DC_Motor.c:

```c
/* DC_Motor.c */

#include "DC_Motor.h"

unsigned int Speed = 0;
extern void (*DC_State)();
void DC_init(){
    printf("DC_Init\n");
}
DCmotor_Set_Speed(int s){
    Speed = s;
    DC_State = state(DC_Busy);
    printf("CA------->DC\n");

}
state_define(DC_Idle){

    DC_State = DC_idle;
    printf("DC_IdleState : Speed= %d\n",Speed);

}
state_define(DC_Busy){
    DC_State = DC_idle;

    DC_State = state(DC_Idle);

    printf("DC_BusyState : Speed= %d\n",Speed);
}

```

ThreeModulesCA.h:

```c

#ifndef THREEMODULESCA_H_
#define THREEMODULESCA_H_

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

enum{
    CA_waiting,
    CA_driving
}CA_State;

void (*State)();

state_define(CA_Driving);
state_define(CA_Waiting);

US_Distance_Get(int d);
DCmotor_Set_Speed(int s);




#endif /* THREEMODULESCA_H_ */

```

ThreeModulesCA.c:

```c

#include <stdio.h>
#include <stdlib.h>
#include "state.h"
#include "ThreeModulesCA.h"
#include "US.h"
#include "DC_Motor.h"


int CA_Speed = 0;
int CA_Distance = 0;
int Threshold = 50;

extern void (*State)();


void setUp(){

    US_init();
    DC_init();

    State = state(CA_Waiting);
    US_State = state(US_busy);
    DC_State = state(DC_Idle);

}

US_Distance_Get(int d){
    CA_Distance = d;
    (CA_Distance <= Threshold) ? (State = state(CA_Waiting)) : (State = state(CA_Driving));
    printf("US------Distance----->CA %d\n",CA_Distance);
}

state_define(CA_Waiting){
    // state name

    CA_State = CA_waiting ;

    // action
    CA_Speed = 0;
    DCmotor_Set_Speed(CA_Speed);

    printf("Driving state: Speed: %d   Distance= %d\n",CA_Speed,CA_Distance);
}
state_define(CA_Driving){
    // state name
    CA_State = CA_driving ;
    // action
    CA_Speed = 30;
    DCmotor_Set_Speed(CA_Speed);


    printf("Driving state: Speed: %d   Distance= %d\n",CA_Speed,CA_Distance);
}
```

main.c:

```c
#include <stdio.h>
#include <stdlib.h>
#include "ThreeModulesCA.h"
#include "US.h"
#include "DC_Motor.h"
#include "state.h"
int main(void) {
    volatile unsigned int i;
    setUp();
    while(1){
        State();
        US_State();
        DC_State();

        for(i=0 ; i<200000 ; i++);
    }


}
```

Output:

```
ThreeModulesCA.exe [C/C++ Application] F:\Handsa\Embedded courses\Learn in
CA------->DC
Driving state: Speed: 30   Distance= 55
US_Busy state :  Distance = 53
US------Distance----->CA 53
DC_BusyState : Speed= 30
CA------->DC
Driving state: Speed: 30   Distance= 53
US_Busy state :  Distance = 45
US------Distance----->CA 45
DC_BusyState : Speed= 30
CA------->DC
Driving state: Speed: 0   Distance= 45
US_Busy state :  Distance = 46
US------Distance----->CA 46
DC_BusyState : Speed= 0
CA------->DC
Driving state: Speed: 0   Distance= 46
US_Busy state :  Distance = 55
US------Distance----->CA 55
DC_BusyState : Speed= 0
CA------->DC
Driving state: Speed: 30   Distance= 55
US_Busy state :  Distance = 46
US------Distance----->CA 46
DC_BusyState : Speed= 30
CA------->DC
Driving state: Speed: 0   Distance= 46
US_Busy state :  Distance = 53
```