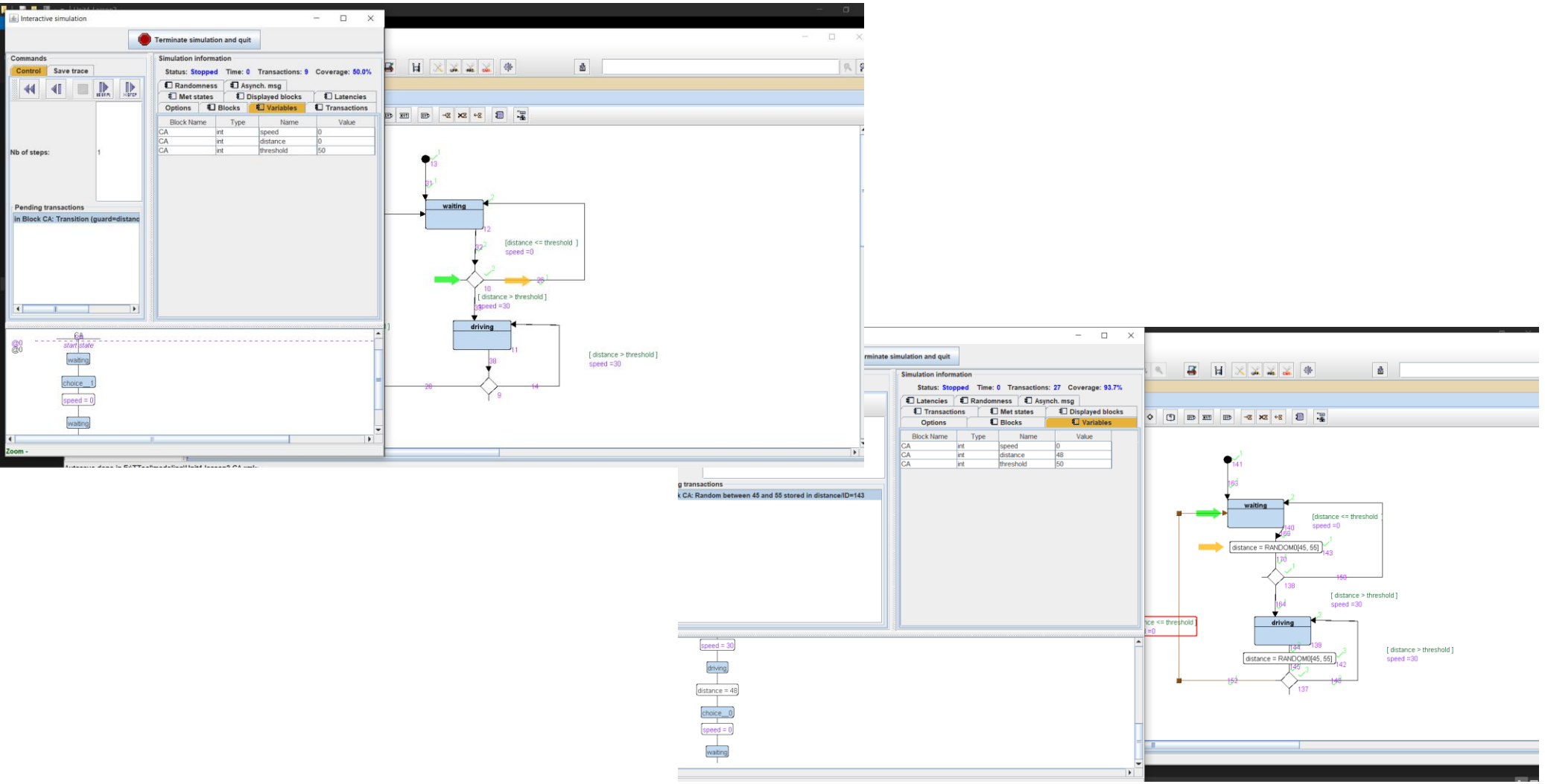
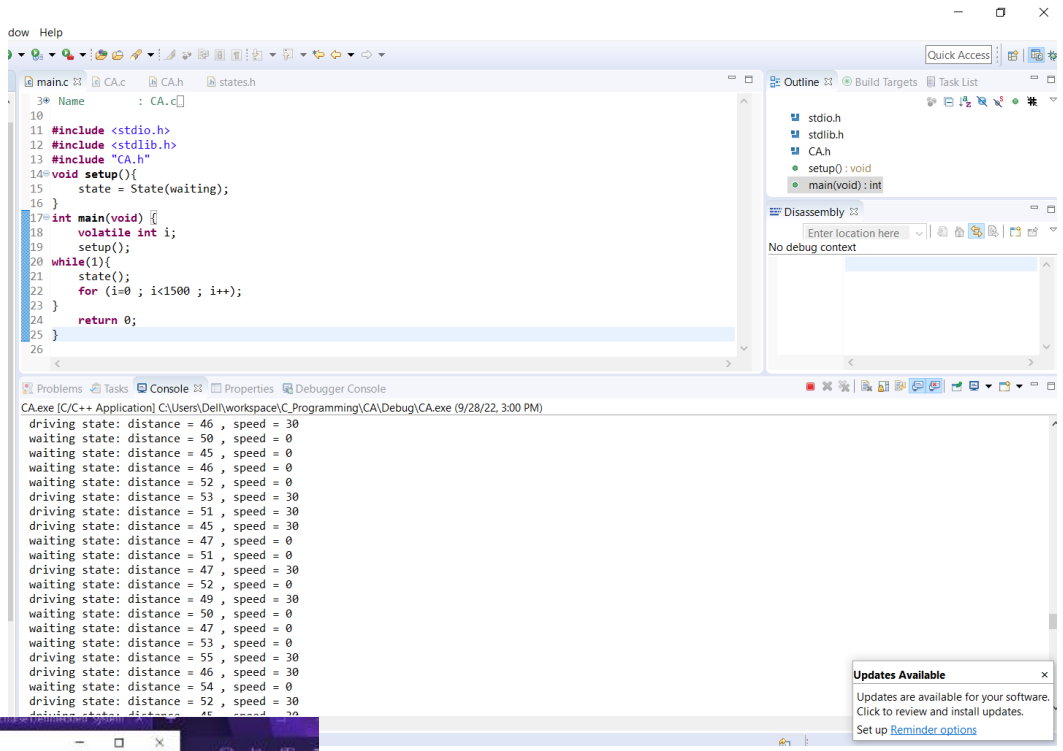
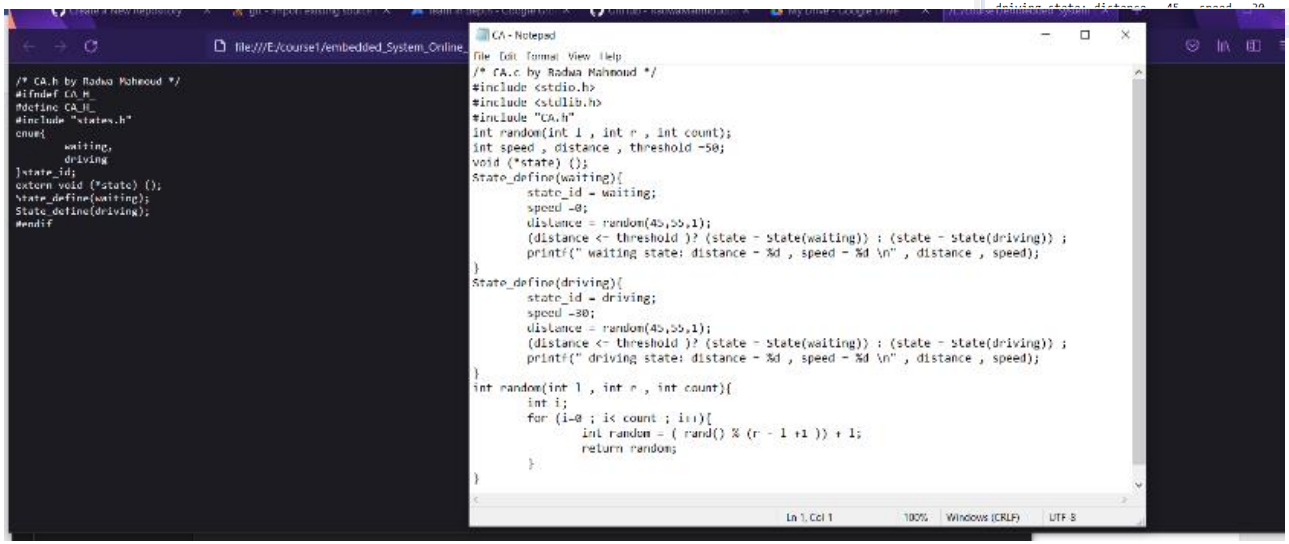
REPORT UNIT 4 LESSON 2 By Radwa Mahmoud:

STATE MACHINE DIAGRAM:

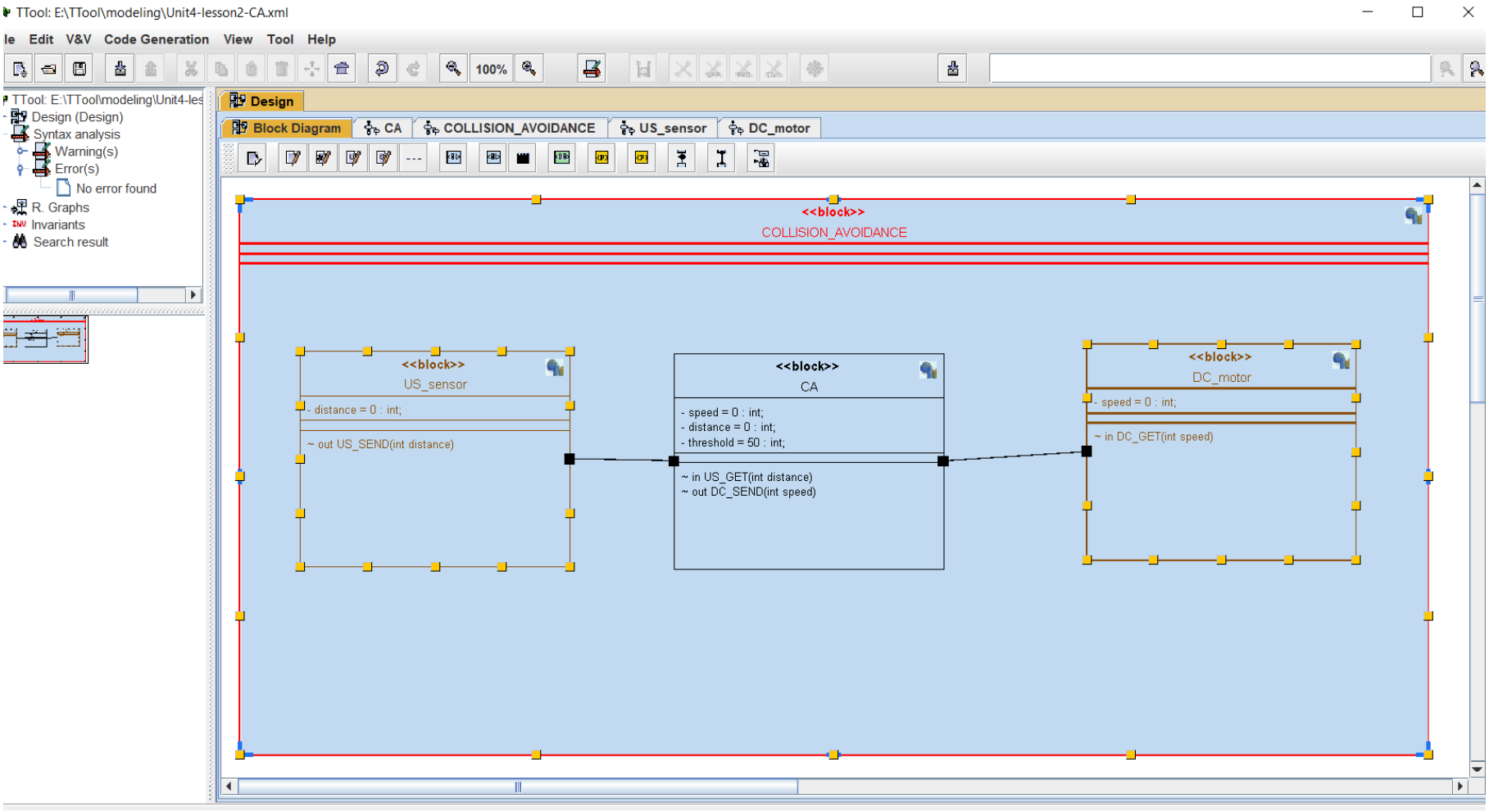


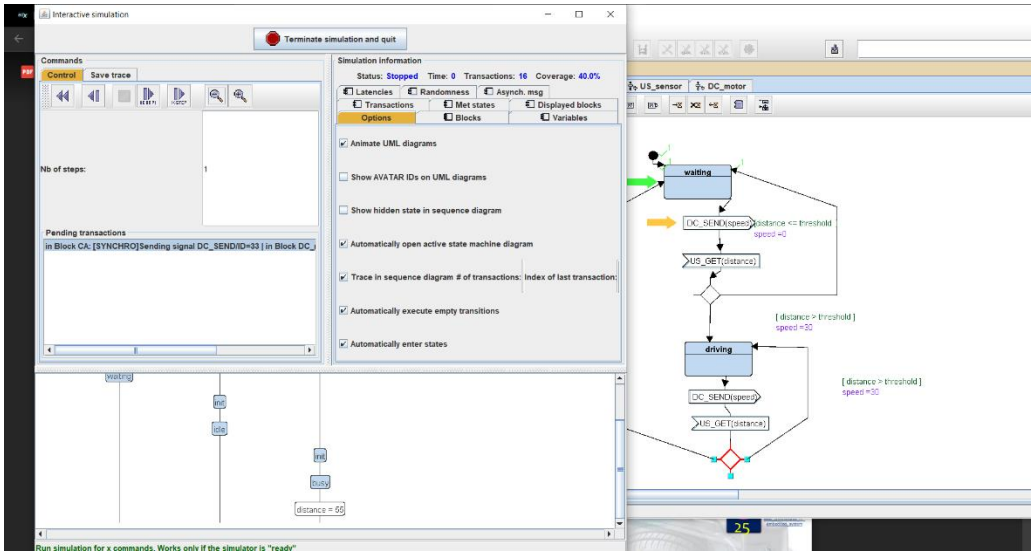
C-codes:

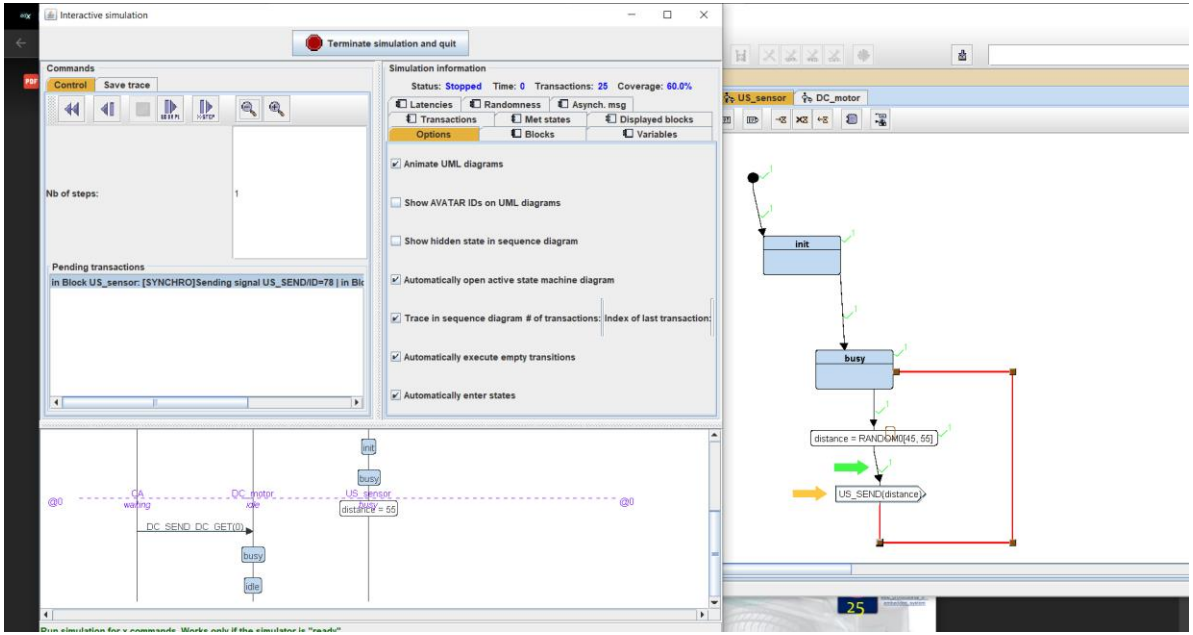

dow Help
Quick Access
main.c
CA.c
CA.h
states.h
Name : CA.c
#include <stdio.h>
#include <stdlib.h>
#include "CA.h"
void setup(){
state = State(waiting);
}
int main(void) {
volatile int i;
setup();
while(1){
state();
for (i=0 ; i<1500 ; i++);
}
return 0;
}
Outline
Build Targets
Task List
stdio.h
stdlib.h
CA.h
setup() : void
main(void) : int
Disassembly
Enter location here
No debug context
Problems
Tasks
Console
Properties
Debugger Console
CA.exe [C/C++ Application] C:\Users\Dell\workspace\C_Programming\CA\Debug\CA.exe (9/28/22, 3:00 PM)
driving state: distance = 46 , speed = 30
waiting state: distance = 50 , speed = 0
waiting state: distance = 45 , speed = 0
waiting state: distance = 46 , speed = 0
waiting state: distance = 52 , speed = 0
driving state: distance = 53 , speed = 30
driving state: distance = 51 , speed = 30
driving state: distance = 45 , speed = 30
waiting state: distance = 47 , speed = 0
waiting state: distance = 51 , speed = 0
driving state: distance = 47 , speed = 30
waiting state: distance = 52 , speed = 0
driving state: distance = 49 , speed = 30
waiting state: distance = 50 , speed = 0
waiting state: distance = 47 , speed = 0
waiting state: distance = 53 , speed = 0
driving state: distance = 55 , speed = 30
driving state: distance = 46 , speed = 30
waiting state: distance = 54 , speed = 0
driving state: distance = 52 , speed = 30
Updates Available
Updates are available for your software.
Click to review and install updates.
Set up Reminder options

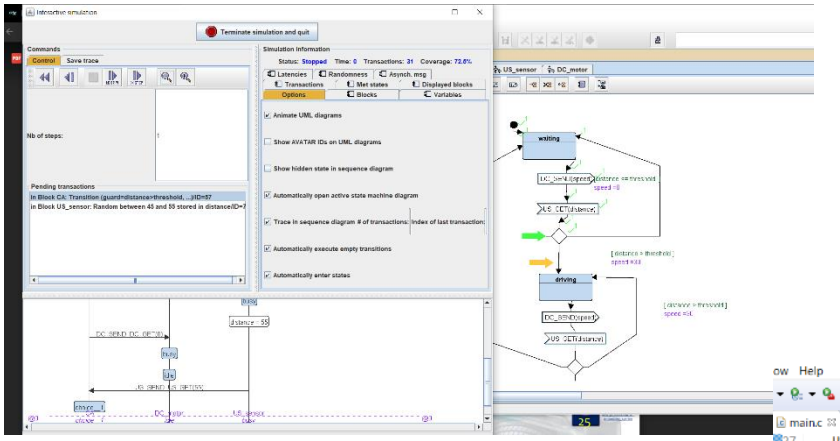

file:///E:/course1/embedded_System_Online_
/* CA.h by Radwa Mahmoud */
#ifndef CA_H
#define CA_H
#include "states.h"
enum{
waiting,
driving
}state_id;
extern void (*state) ();
State_define(waiting);
State_define(driving);
#endif
CA - Notepad
File Edit Format View Help
/* CA.c by Radwa Mahmoud */
#include <stdio.h>
#include <stdlib.h>
#include "CA.h"
int random(int l , int r , int count);
int speed , distance , threshold =50;
void (*state) ();
State_define(waiting){
state_id = waiting;
speed =0;
distance = random(45,55,1);
(distance <= threshold)? (state = State(waiting)) : (state = State(driving)) ;
printf(" waiting state: distance = %d , speed = %d \n" , distance , speed);
}
State_define(driving){
state_id = driving;
speed =30;
distance = random(45,55,1);
(distance <= threshold)? (state = State(waiting)) : (state = State(driving)) ;
printf(" driving state: distance = %d , speed = %d \n" , distance , speed);
}
int random(int l , int r , int count){
int i;
for (i=0 ; i< count ; i++){
int random = (rand() % (r - l +1)) + l;
return random;
}
}
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8

STATE MACHINE DIAGRAMS FOR:




Interactive simulation
Terminate simulation and quit
Commands
Control
Save trace
Nb of steps:
Pending transactions
Simulation information
Status: Stopped Time: 0 Transactions: 16 Coverage: 40.0%
Latencies
Randomness
Asynch. msg
Transactions
Met states
Displayed blocks
Options
Blocks
Variables
Animate UML diagrams
Show AVATAR IDs on UML diagrams
Show hidden state in sequence diagram
Automatically open active state machine diagram
Trace in sequence diagram # of transactions:
Index of last transaction:
Automatically execute empty transitions
Automatically enter states
Run simulation for x commands. Works only if the simulator is "ready"
US_sensor
DC_motor
waiting
DC_SEND(speed)
distance <= threshold
speed =0
US_GET(distance)
[distance > threshold]
speed =30
driving
[distance > threshold]
speed =30
init
idle
busy
distance = 60


Interactive simulation
Terminate simulation and quit
Commands
Control
Save trace
Nb of steps:
Pending transactions
in Block US_sensor: [SYNCHRO]Sending signal US_SEND(ID=78 | in Blo
Simulation information
Status: Stopped Time: 0 Transactions: 25 Coverage: 60.0%
Latencies
Randomness
Asynch. msg
Transactions
Met states
Displayed blocks
Options
Blocks
Variables
Animate UML diagrams
Show AVATAR IDs on UML diagrams
Show hidden state in sequence diagram
Automatically open active state machine diagram
Trace in sequence diagram # of transactions:
Index of last transaction:
Automatically execute empty transitions
Automatically enter states
US_sensor
DC_motor
init
busy
distance = RANDOM[45, 55]
US_SEND(distance)
@0
CA
waiting
DC_motor
idle
US_sensor
busy
distance = 55
@0
DC_SEND, DC_GET(0)
busy
idle



C _CODES:

ow Help

Quick Access

main.c | CA.c | CA.h | states.h | US_sensor.c | US_sensor.h | DC_motor.c | DC_motor.h

```
27      US_state();
28      state();
29      DC_state();
30      for (i=0 ; i<1500 ; i++);
31  }
```

Outline | Build Targets | Task List

DC_motor.h
US_sensor.h
setup() : void
main(void) : int

Problems | Tasks | Console | Properties | Debugger Console

<terminated> (exit value: 0) CA.exe [C/C++ Application] C:\Users\Dell\workspace\C_Programming\CA\Debug\CA.exe (9/30/22, 4:22 AM)

```
US INIT
DC INIT
 US_busy state: distance = 53
 US_sensor -> -> ->  CA : distance = 53
 driving state: distance = 53 , speed = 30
CA -> -> ->  DC_motor : speed = 30
 DC_busy state: speed = 30
 US_busy state: distance = 54
 US_sensor -> -> ->  CA : distance = 54
 driving state: distance = 54 , speed = 30
CA -> -> ->  DC_motor : speed = 30
 DC_busy state: speed = 30
 US_busy state: distance = 54
 US_sensor -> -> ->  CA : distance = 54
 driving state: distance = 54 , speed = 30
CA -> -> ->  DC_motor : speed = 30
 DC_busy state: speed = 30
 US_busy state: distance = 46
 US_sensor -> -> ->  CA : distance = 46
 waiting state: distance = 46 , speed = 0
CA -> -> ->  DC_motor : speed = 0
 DC_busy state: speed = 0
 US_busy state: distance = 52
 US_sensor -> -> ->  CA : distance = 52
 driving state: distance = 52 , speed = 30
CA -> -> ->  DC_motor : speed = 30
 DC_busy state: speed = 30
 US_busy state: distance = 50
 US_sensor -> -> ->  CA : distance = 50
 waiting state: distance = 50 , speed = 0
CA -> -> ->  DC_motor : speed = 0
 DC_busy state: speed = 0
 US_busy state: distance = 50
```

MAIN.C

main.c | CA.c | CA.h | states.h | US_sensor.c | US_sensor.h | DC_motor.c | DC_motc

```
3  Name        : CA.c
10
11 #include <stdio.h>
12 #include <stdlib.h>
13 #include "CA.h"
14 #include "DC_motor.h"
15 #include "US_sensor.h"
16 void setup(){
17     US_INIT();
18     DC_INIT();
19     US_state = State(US_busy);
20     state = State(waiting);
21     DC_state =State(idle);
22 }
23 int main(void) {
24     volatile int i ;
25     setup();
26     while(1){
27     US_state();
28     state();
29     DC_state();
30     for (i=0 ; i<1500 ; i++);
31 }
32     return 0;
33 }
```

CA.C:

main.c | CA.c | CA.h | states.h | US_sensor.c | US_sensor.h | DC_motor.c | DC_motor.h

```
1  /* CA.c by Radwa Mahmoud */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include "CA.h"
5  int speed , CA_distance , threshold =50;
6  void (*state) ();
7
8  void US_SEND_DISTANCE(int distance){
9      CA_distance = distance;
0      printf(" US_sensor -> -> ->  CA : distance = %d \n" , CA_distance);
1      (CA_distance <= threshold )? (state = State(waiting)) : (state = State(driving)) ;
2  }
3  State_define(waiting){
4      state_id = waiting;
5      speed =0;
6      printf(" waiting state: distance = %d , speed = %d \n" , CA_distance , speed);
7      DC_GET_SPEED(speed);
8  }
9  State_define(driving){
0      state_id = driving;
1      speed =30;
2      printf(" driving state: distance = %d , speed = %d \n" , CA_distance , speed);
3      DC_GET_SPEED(speed);
4  }
```

CA.h:

main.c | CA.c | CA.h | states.h | US_sensor.c | US_sensor.h

```c
/* CA.h by Radwa Mahmoud */
#ifndef CA_H_
#define CA_H_
#include "states.h"
enum{
    waiting,
    driving
}state_id;
extern void (*state) ();
State_define(waiting);
State_define(driving);
#endif
```

States.h:

```c
#ifndef STATES_H_
#define STATES_H_
#define State_define(state_name) void St_##state_name()
#define State(state_name) St_##state_name
//connections
void DC_GET_SPEED(int speed);
void US_SEND_DISTANCE(int distance);
#endif
```

US_sensor.c:

main.c | CA.c | CA.h | states.h | US_sensor.c | US_sensor.h | DC_motor.c | DC_motor.

```c
/* US_sensor.c by Radwa Mahmoud */
#include <stdio.h>
#include <stdlib.h>
#include "US_sensor.h"
int random(int l , int r , int count);
int US_distance ;
void (*US_state) ();
void US_INIT(){
    printf("US INIT \n");
}
State_define(US_busy){
    US_state_id = US_busy;
    US_distance = random(45,55,1);
    printf(" US_busy state: distance = %d \n" , US_distance);

    US_SEND_DISTANCE(US_distance );
    US_state = State(US_busy);
}

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

US_sensor.h:

main.c | CA.c | CA.h | states.h | US_sensor.c | US_sensor.h | DC_motor.c | DC_motor.h

```c
/* US_sensor.h by Radwa Mahmoud */
#ifndef US_SENSOR_
#define US_SENSOR_
#include "states.h"
enum{
    US_busy
}US_state_id;
void US_INIT();
extern void (*US_state) ();
State_define(US_busy);
#endif
```

DC_motor.c:

main.c | CA.c | CA.h | states.h | US_sensor.c | US_sensor.h | DC_motor.c | DC_motor.h

```c
/* DC_motor.c by Radwa Mahmoud */
#include <stdio.h>
#include <stdlib.h>
#include "DC_motor.h"
int DC_speed ;
void (*DC_state) ();
void DC_INIT(){
    printf("DC INIT \n");
}
void DC_GET_SPEED(int speed){
    DC_speed = speed;
    DC_state =State(DC_busy);
    printf("CA -> -> ->  DC_motor : speed = %d \n" , DC_speed );
}

State_define(idle){
    DC_state_id = idle;
    printf(" idle state: speed = %d \n" , DC_speed);
}
State_define(DC_busy){
    DC_state_id = DC_busy;
    printf(" DC_busy state: speed = %d \n" , DC_speed);
    DC_state =State(idle);
}
```

DC_motor.h:

main.c | CA.c | CA.h | states.h | US_sensor.c | US_sensor.h | DC_motor.c | DC_motor.h

```c
/* DC_motor.h by Radwa Mahmoud */
#ifndef DC_MOTOR_
#define DC_MOTOR_
#include "states.h"
enum{
    idle,
    DC_busy
}DC_state_id;
void DC_INIT();
extern void (*DC_state) ();
State_define(idle);
State_define(DC_busy);
#endif
```