2024

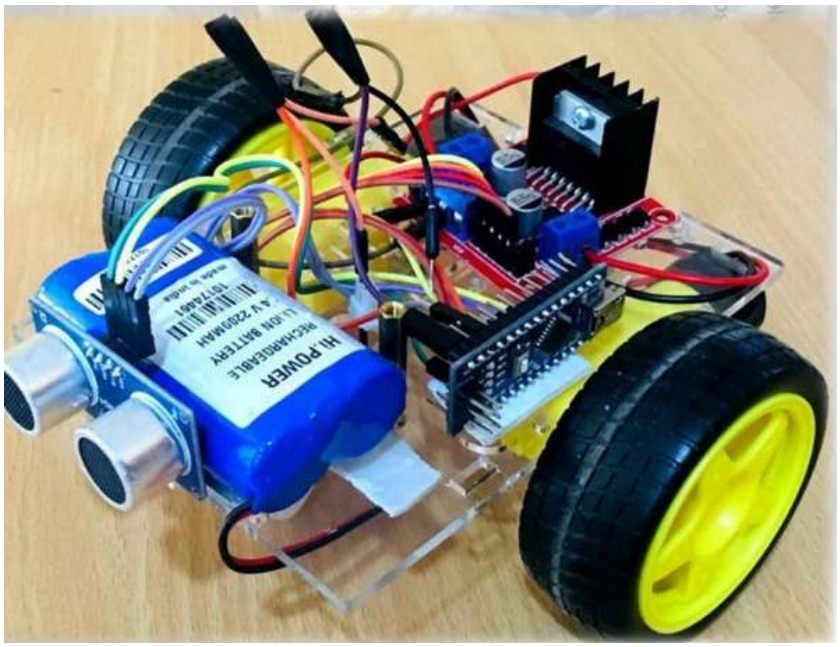
Collision Avoidance

Mohamed Hakeem

Master Embedded System

8/17/2024

1 Case Study:



- Ultrasonic Obstacle-avoiding Robot:

This case study involves building an Ultrasonic Obstacle-Avoiding Robot designed to detect and avoid obstacles. To simplify the process, an ultrasonic sensor will be used to measure the distance between the robot and any obstacles in its path. A DC motor will be used to control the robot's movement, allowing it to either stop or continue moving based on the detected distance.

- Sequence for building software:
  1. Design a state diagram with any tool according to the standard between teams and each other.
  2. Verify this design using the same tool.
  3. Once it is achieved successfully, implement the code of each task that is built on the state diagram with the same template in any IDE.
  4. mapping between these tasks by standard function.

- Design a state diagram with any tool according to the standard between teams and each other.

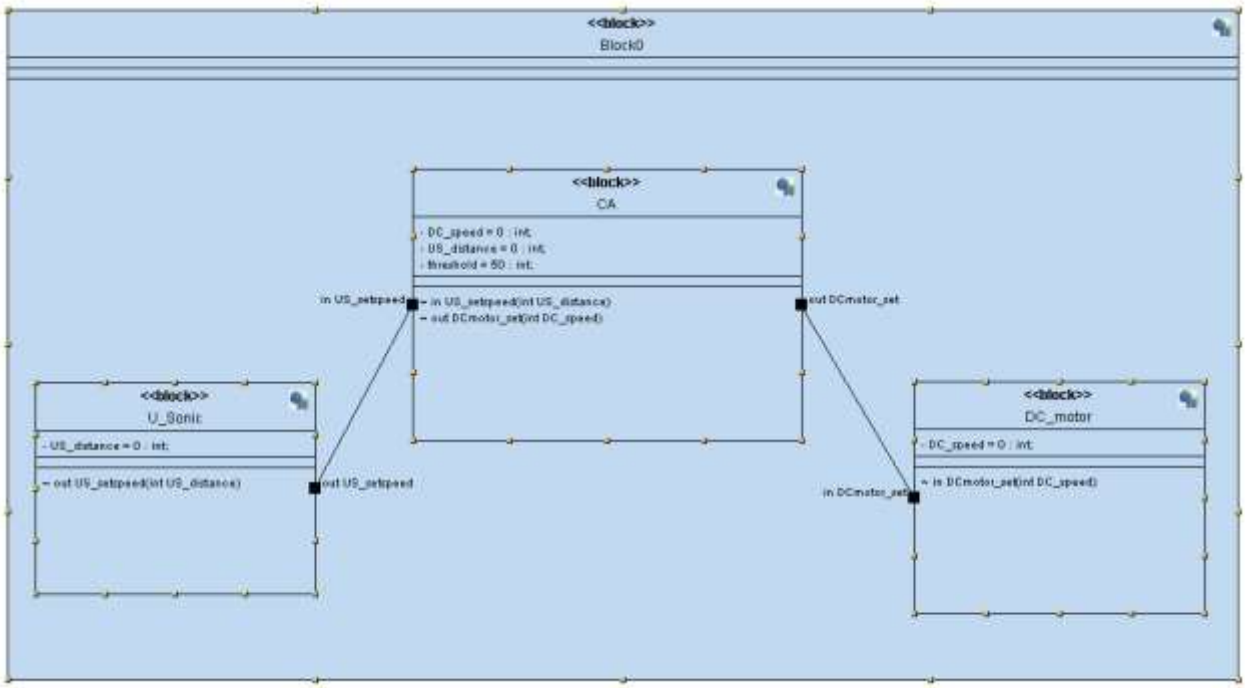


Figure 1:Collision Avoidance Block

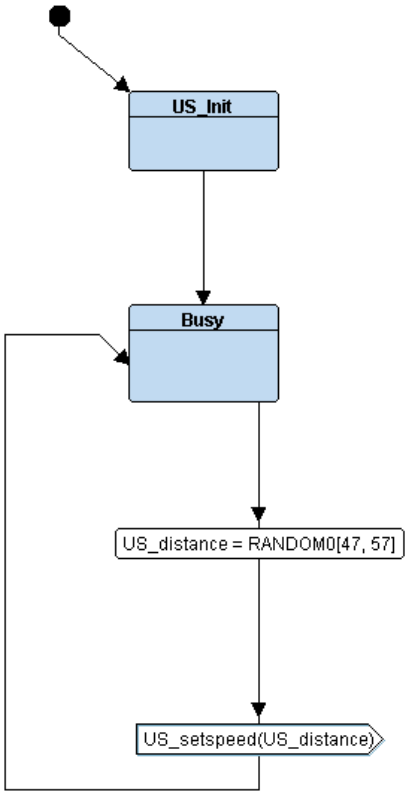


Figure 2:Ultrasonic_States

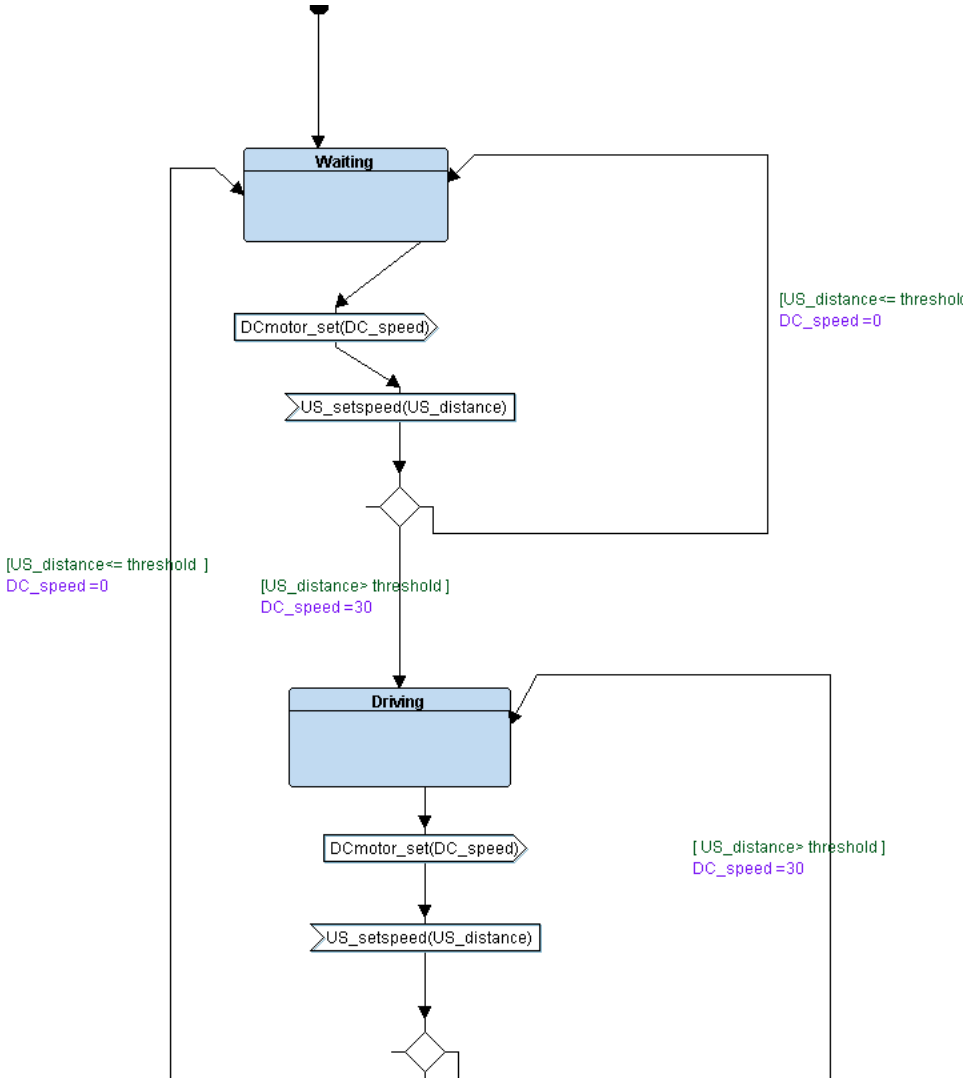


Figure 3:Collision Avoidance Algorithm States

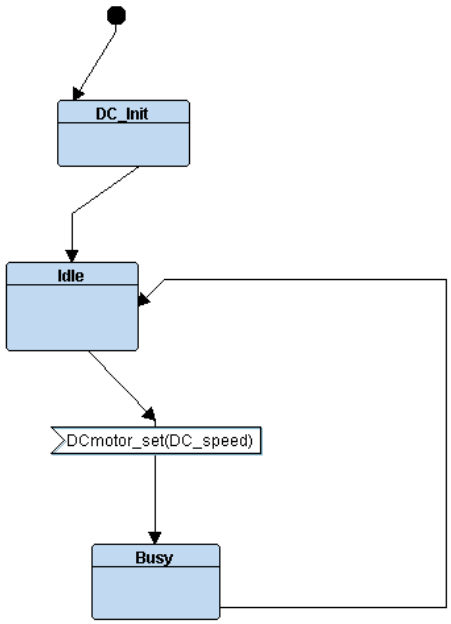
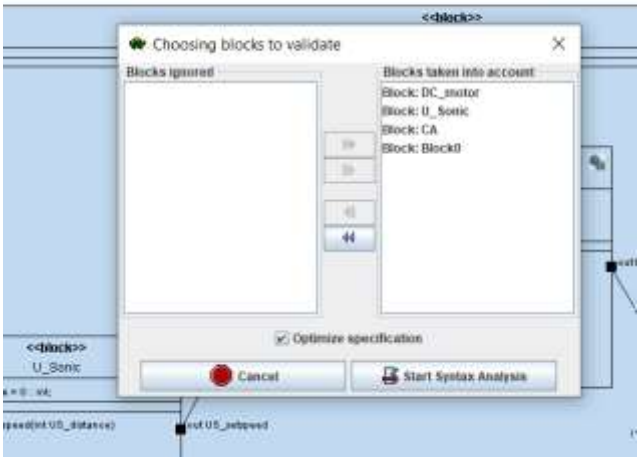


Figure 4: Dc_Motor_States

- Verify this design using the same tool.



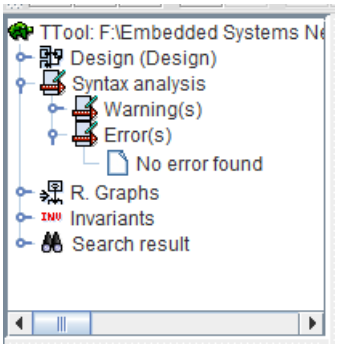


Figure 5:Syntax Analysis

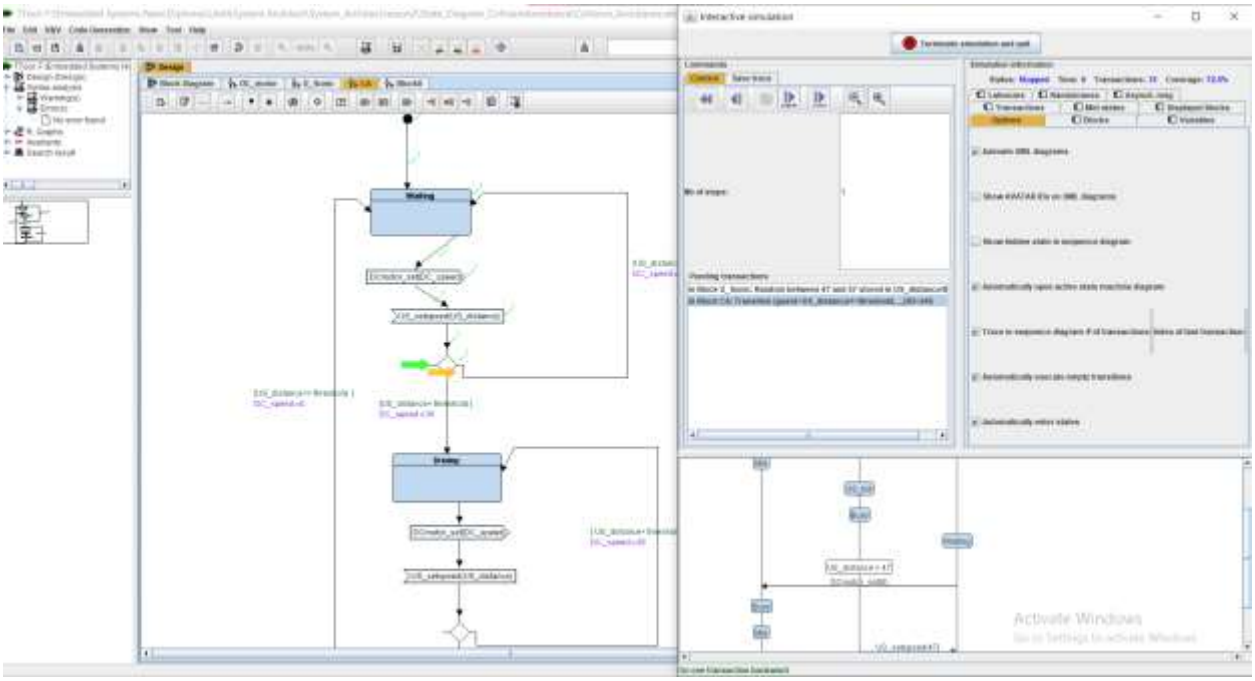
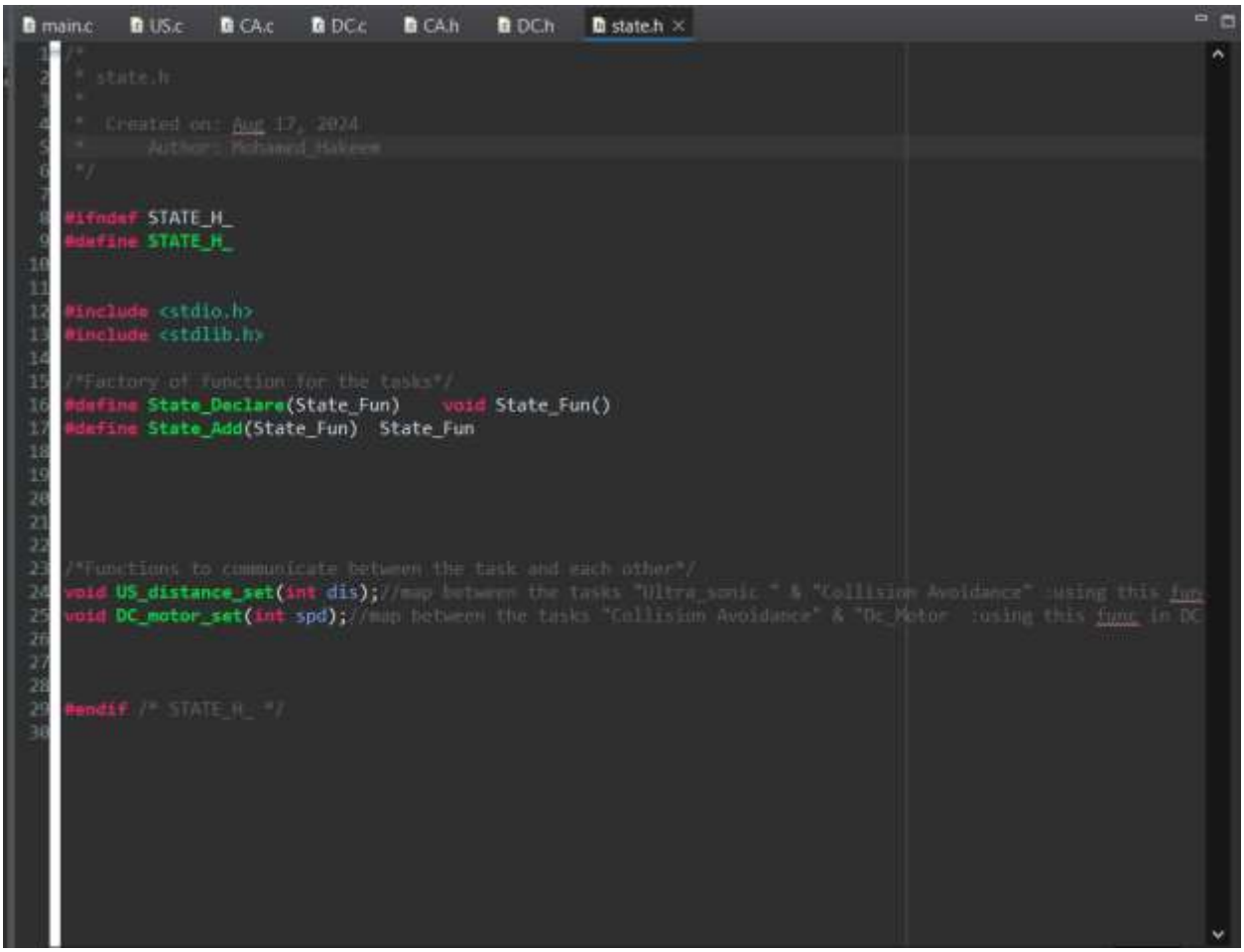


Figure 6: Verification

- Once it is achieved successfully, implement the code of each task that is built on the state diagram with the same template in any IDE.



```
/*
 * state.h
 *
 *  Created on: Aug 17, 2024
 *      Author: Mohamed_Hakeem
 */

#ifndef STATE_H_
#define STATE_H_


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

/*Factory of function for the tasks*/
#define State_Declare(State_Fun)    void State_Fun()
#define State_Add(State_Fun)  State_Fun




/*Functions to communicate between the task and each other*/
void US_distance_set(int dis);//map between the tasks "Ultra_sonic " & "Collision Avoidance" :using this fun
void DC_motor_set(int spd);//map between the tasks "Collision Avoidance" & "Dc_Motor  :using this func in DC



#endif /* STATE_H_ */
```

Figure 7: States_code

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

```c
/*
 * US.h
 *
 *  Created on: Aug 17, 2024
 *      Author:  Mohamed Hakeem
 */

#ifndef US_H_
#define US_H_

#include"state.h"

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


/*states that you made in the tool of state diagram*/
enum
{
    US_Init,
    US_Busy

}US_State_ID;


State_Declare(US_init);
State_Declare(US_busy);


//global pointer to func.
void (*US_State)();


#endif /* US_H_ */
```

Figure 8: Ultrasonic_header

```c
/*
 * US.c
 *
 *  Created on: Aug 17, 2024
 *      Author: Mohamed_Hakeem
 *  US.c[Ultrasonic_Sensor] is task or file that responsible for  reading the distance and then sending the
 *  to the task is called CA.c [Collision Avoidance] that is task that has "algorithm" for achieving the req
 */


#include "US.h"
unsigned int distance=0;
extern void (*US_State)();




State_Declare(US_init)
{


    //Initialization of US Device
    //Timer initlization
    //....
    US_State_ID=US_Init;
    printf("========The US Init ....... is done=========\n");



}

State_Declare(US_busy)
{

    US_State_ID=US_Busy;
    distance=generate_random(45,55,1);
    printf("US_busy:-------distance=%d\n",distance);
```

Figure 9:Ultrasonic_source

```
main.c   US.c ×   CA.c   DC.c   CA.h   DC.h   state.h   US.h

    //Initialization of US Device
    //Timer Initilization
    //....
    US_State_ID=US_Init;
    printf("=======The US Init ....... is done=========\n");



}

State_Declare(US_busy)
{

    US_State_ID=US_Busy;
    distance=generate_random(45,55,1);
    printf("US_busy:-------distance=%d\n",distance);

    US_distance_set(distance);
    US_State=State_Add(US_busy);



}




int generate_random(int l,int r,int count)
{
    /* Func generate the random number in range of ( r , l ) */
    int i;
    for(i=0;i<count;i++)
    {
        int rand_num=(rand() % (r-l+1) ) +l;
        return rand_num;
    }

}

Console ×   Problems   Executables   Debugger Console
```

Figure 10:Ultrasonic _source

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

```c
/*
 * CA.h
 *
 *  Created on: Aug 17, 2024
 *      Author: Mohamed Nakeem
 *      Collision Avoidance task is the algorithm that control between the another tasks
 */

#ifndef CA_H_
#define CA_H_

#include"state.h"


/*states that you made in the tool of state diagram*/
enum
{
    CA_Waiting,
    CA_Driving

}CA_State_ID;


State_Declare(CA_waiting);
State_Declare(CA_driving);



//global pointer to func;
void (*CA_State)();



#endif /* CA_H_ */
```

Figure 11:Collision_Avoidance_Header

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

```c
/*
 * CA.c
 *
 *  Created on: Aug 17, 2024
 *      Author: Mohamed_Hakeem
 *
 *  CA.c[Collision Avoidance] is THE Algorithm
 */

#include "CA.h"

unsigned int distance,speed,threshold=50;
extern void (*CA_State)();



void US_distance_set(int dis)
{

    distance=dis;
    (distance<=threshold)?(CA_State=State_Add(CA_waiting)):(CA_State=State_Add(CA_driving));
    printf("==========the US distance............... = %d ===============\n",distance);

}

State_Declare(CA_waiting)
{
    CA_State_ID= CA_Waiting ;
    printf("Waiting State--------------------: Speed=%d, Distance=%d\n",speed,distance);

    speed=0;
    DC_motor_set(speed);
}




State_Declare(CA_driving)
{
```

Figure 12::Collision_Avoidance_Source

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

```c
#include "CA.h"

unsigned int distance,speed,threshold=50;
extern void (*CA_State)();



void US_distance_set(int dis)
{

    distance=dis;
    (distance<=threshold)?(CA_State=State_Add(CA_waiting)):(CA_State=State_Add(CA_driving));
    printf("==========the US distance............. = %d ==============\n",distance);

}

State_Declare(CA_waiting)
{
    CA_State_ID= CA_Waiting ;
    printf("Waiting State-------------------: Speed=%d, Distance=%d\n",speed,distance);

    speed=0;
    DC_motor_set(speed);
}




State_Declare(CA_driving)
{
    CA_State_ID= CA_Driving ;
    printf("Driving State-------------------: Speed=%d, Distance=%d\n",speed,distance);

    speed=30;
    DC_motor_set(speed);
}
```

Figure 13::Collision_Avoidance_Source

```
main.c   US.c   CA.c   DC.c   CA.h   DC.h   state.h   US.h
/*
 * DC.h
 *
 *  Created on: Aug 17, 2024
 *      Author: Mohamed_Hakeem
 */

#ifndef DC_H_
#define DC_H_



#include"state.h"


/*states that you made in the tool of state diagram*/
enum
{
    DC_Init,
    DC_Idle,
    DC_Busy

}DC_State_ID;


State_Declare(DC_init);
State_Declare(DC_idle);
State_Declare(DC_busy);


//global pointer to func
void (*DC_State)();




#endif /* DC_H_ */
```

Figure 14:Dc_Motor_Header

```c
/*
 * DC.c
 *
 *  Created on: Aug 17, 2024
 *      Author: Mohamed Hakeem
 *
 *
 *      DC.c[DC_Motor] is the task that receives actor(speed value) from the algorithm CA.c
 */

#include "DC.h"
unsigned int speed=0;

extern void (*DC_State)();



void DC_motor_set(int spd)
{


    speed=spd;
    DC_State=State_Add(DC_busy);
    printf("DC..........Speed= %d\n",speed);



}


State_Declare(DC_init)
{


    //DC_Motor Initialization
    //PWM module init........
    DC_State_ID=DC_Init;
    printf("==============The DC Init is .......Done===============\n");
```

Figure 15:Dc_Motor_Source

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

```c
    speed=spd;
    DC_State=State_Add(DC_busy);
    printf("DC..........Speed= %d\n",speed);



}


State_Declare(DC_init)
{


    //DC_Motor Initialization
    //PWM module init .......
    DC_State_ID=DC_Init;
    printf("===============The DC Init is .......Done===============\n");

}

State_Declare(DC_idle)
{
    //it don't  anything ,it leaves the motor on the previous speed
    DC_State_ID=DC_Idle;
    printf("Idle State -----------:Speed=%d\n",speed);



}

State_Declare(DC_busy)
{
    DC_State_ID=DC_Busy;
    //send PWM to DC motor
    printf("Busy_State------------: Speed= %d\n",speed);
    DC_State=State_Add(DC_idle);

}
```

Figure 16::Dc_Motor_Source

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

```c
/*
 * main.c
 *
 *  Created on: Aug 17, 2024
 *      Author: Mohamed Hakeem
 *
 *      Main.c is considered as the "APP" it is executed for the all system
 */

#include "CA.h"
#include "US.h"
#include "DC.h"


void setup_app()
{


    //init all drivers
    //init IRQn
    //init HAL US_DRIVER  DC DRIVER
    //init Blocks
    State_Declare(DC_init);
    State_Declare(US_init);


    //set states pointers for each first task'states
    US_State=State_Add(US_busy);
    CA_State=State_Add(CA_waiting);
    DC_State=State_Add(DC_idle);

}




int main()
{
```

Figure 17:Main

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

```c
    State_Declare(DC_init);
    State_Declare(US_init);


    //set states pointers for each first task'states
    US_State=State_Add(US_busy);
    CA_State=State_Add(CA_waiting);
    DC_State=State_Add(DC_idle);

}




int main()
{

    volatile int i=0;
    setup_app();


    while(1)
    {
        //call the pointer to function respectively according to your state diagram
        US_State();
        CA_State();
        DC_State();

        for(i=0;i<1000;i++);//delay

    }



    return 0;
}
```

Figure 18:Main

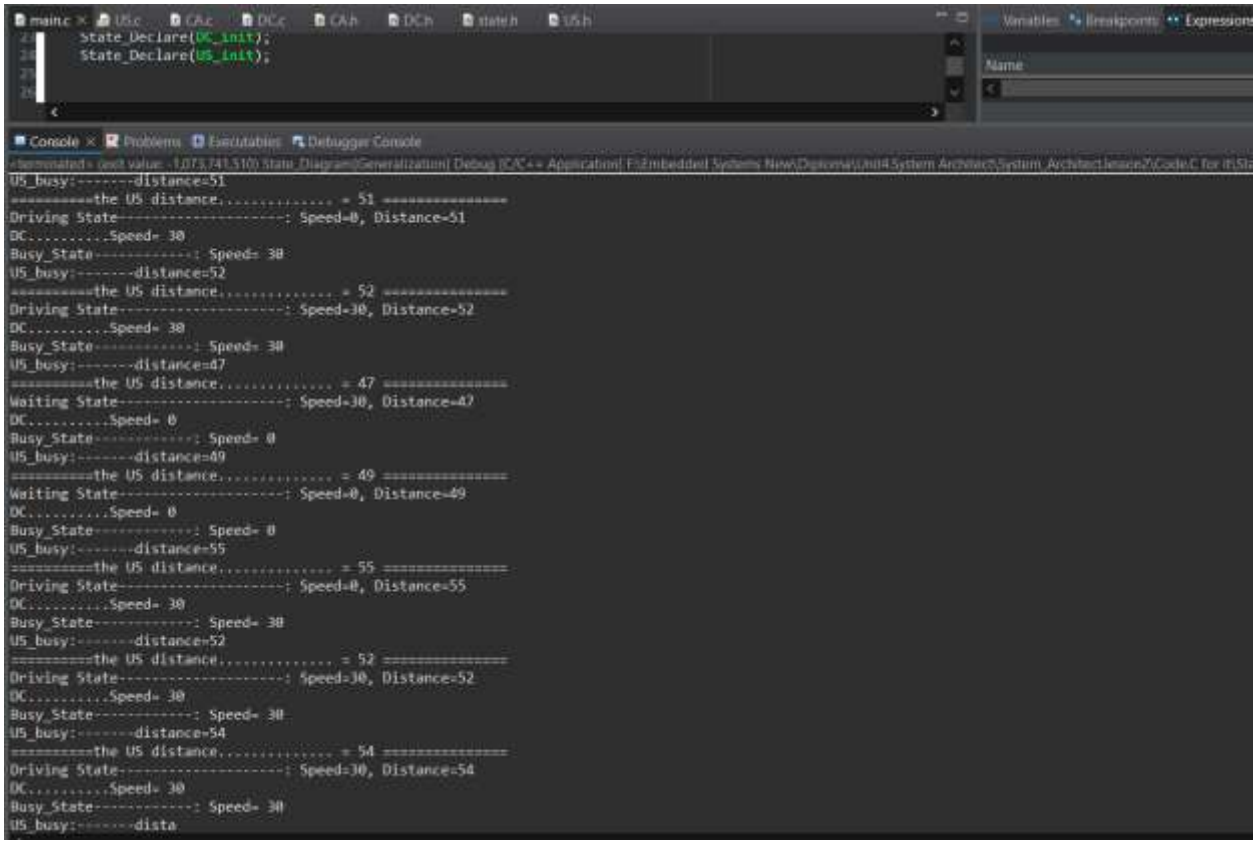


Figure 19:Run_System