Learn in Depth Diploma – System Architect Unit – Lesson 2 Assignment

Report: Creating Collision Avoidance for a Self-Driving Car

Name: Mohamed Hazem Yahya

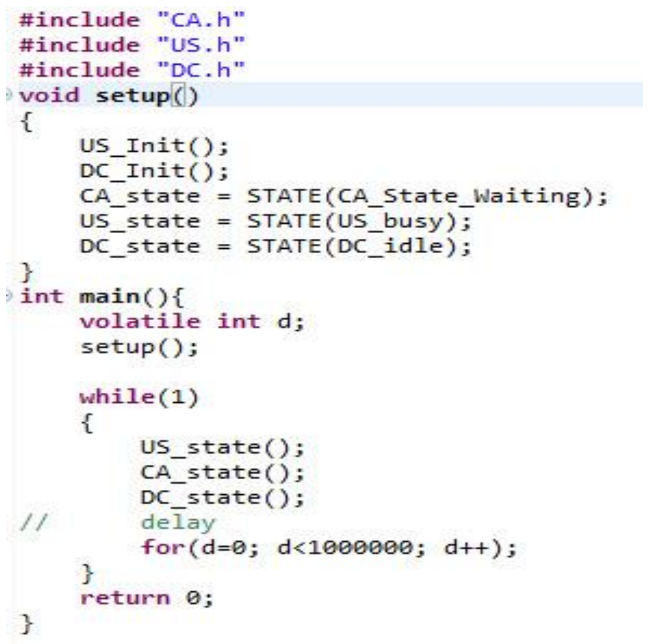
Introduction

The Collision Avoidance System is a project that aims to prevent collisions between a vehicle and obstacles by utilizing ultrasonic sensors and controlling a DC motor to adjust the vehicle's speed accordingly. This report provides an overview of the system's code structure, including the main control loop, the ultrasonic sensor module, the DC motor control module, and the collision avoidance module

Code Overview

1- Main Control Loop

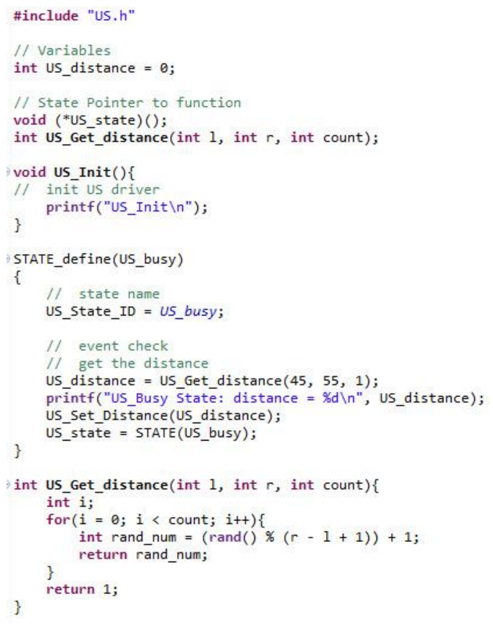
The main control loop is implemented in the **main()** function in the **main.c** file. It initializes the system components, including ultrasonic sensors and DC motors, and enters an infinite loop where it continuously monitors the distance from obstacles, the collision avoidance state, and the DC motor state. It also includes a delay to control the rate of execution.



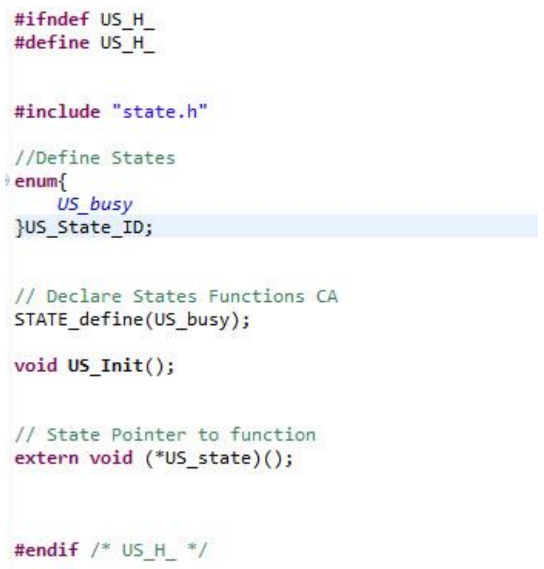
```
#include "CA.h"
#include "US.h"
#include "DC.h"
void setup()
{
    US_Init();
    DC_Init();
    CA_state = STATE(CA_State_Waiting);
    US_state = STATE(US_busy);
    DC_state = STATE(DC_idle);
}
int main(){
    volatile int d;
    setup();

    while(1)
    {
        US_state();
        CA_state();
        DC_state();
//      delay
        for(d=0; d<1000000; d++);
    }
    return 0;
}
```

2- Ultrasonic Sensor Module (US.c and US.h)

The Ultrasonic Sensor module is responsible for simulating the distance measurement from obstacles. It includes functions for initializing the sensor and generating random distance values within a specified range. The **US_busy** state simulates the process of obtaining the distance value from the sensor.



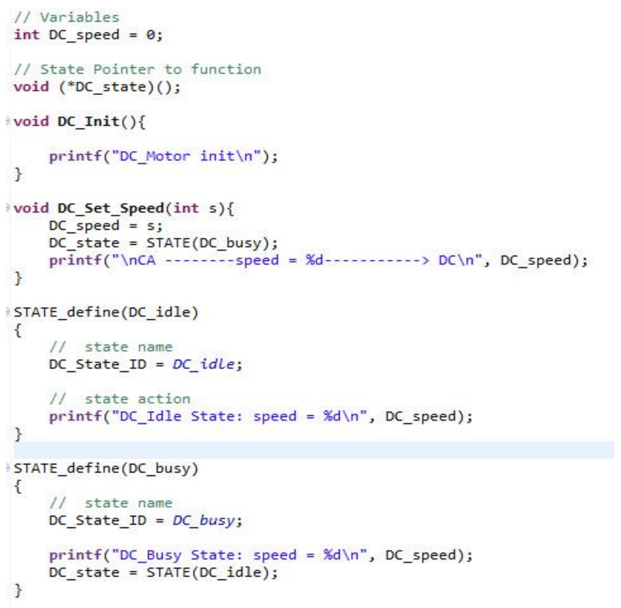
```c
#include "US.h"

// Variables
int US_distance = 0;

// State Pointer to function
void (*US_state)();
int US_Get_distance(int l, int r, int count);

void US_Init(){
//  init US driver
    printf("US_Init\n");
}

STATE_define(US_busy)
{
    //  state name
    US_State_ID = US_busy;

    //  event check
    //  get the distance
    US_distance = US_Get_distance(45, 55, 1);
    printf("US_Busy State: distance = %d\n", US_distance);
    US_Set_Distance(US_distance);
    US_state = STATE(US_busy);
}

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



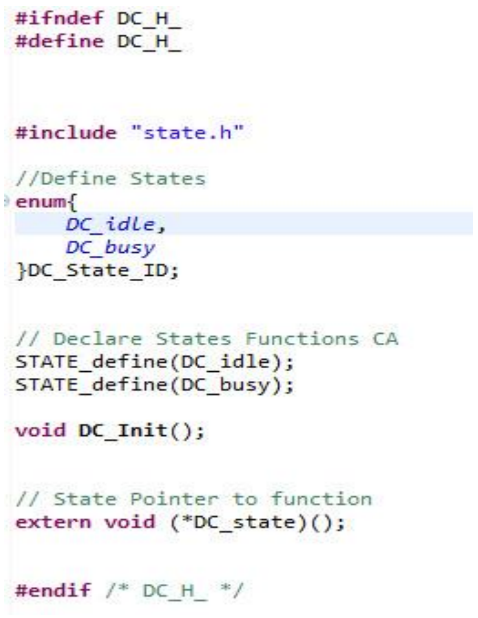
```c
#ifndef US_H_
#define US_H_


#include "state.h"

//Define States
enum{
    US_busy
}US_State_ID;


// Declare States Functions CA
STATE_define(US_busy);

void US_Init();


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



#endif /* US_H_ */
```

3- DC Motor Control Module (DC.c and DC.h)

The DC Motor Control module handles the control of the DC motor. It includes functions for initializing the motor and setting its speed. The module defines states for both the idle and busy states of the DC motor. The state transitions simulate the behavior of the motor when it changes from idle to busy and vice versa.



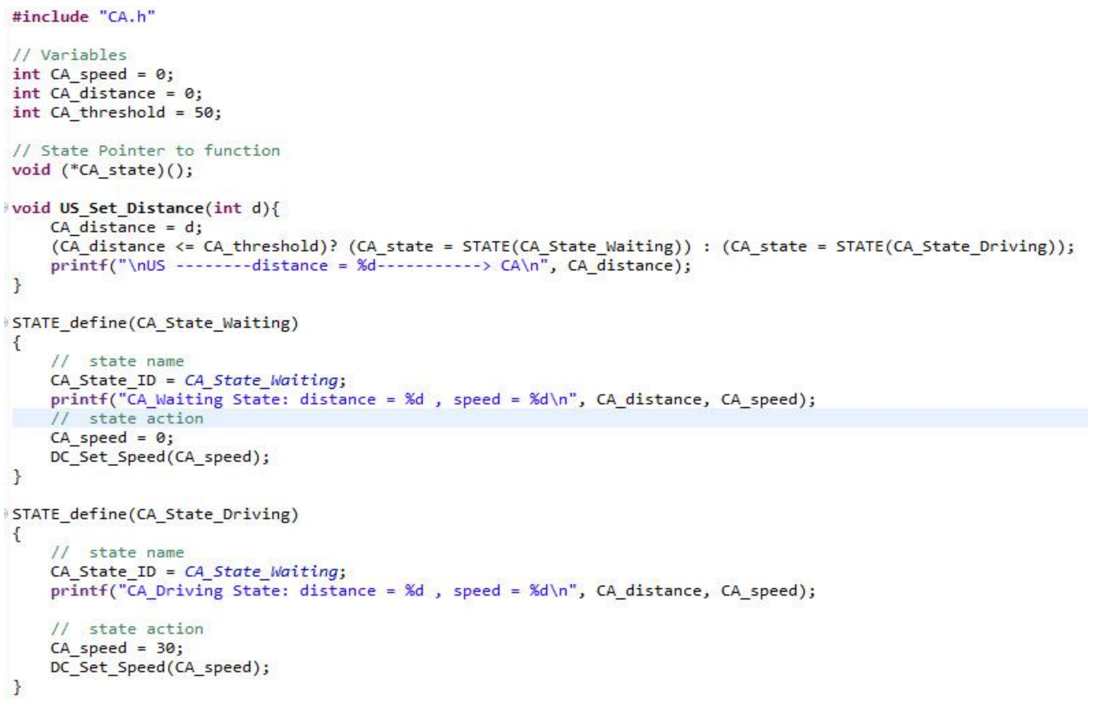
```c
// Variables
int DC_speed = 0;

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

void DC_Init(){

    printf("DC_Motor init\n");
}

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

STATE_define(DC_idle)
{
    //  state name
    DC_State_ID = DC_idle;

    //  state action
    printf("DC_Idle State: speed = %d\n", DC_speed);
}

STATE_define(DC_busy)
{
    //  state name
    DC_State_ID = DC_busy;

    printf("DC_Busy State: speed = %d\n", DC_speed);
    DC_state = STATE(DC_idle);
}
```



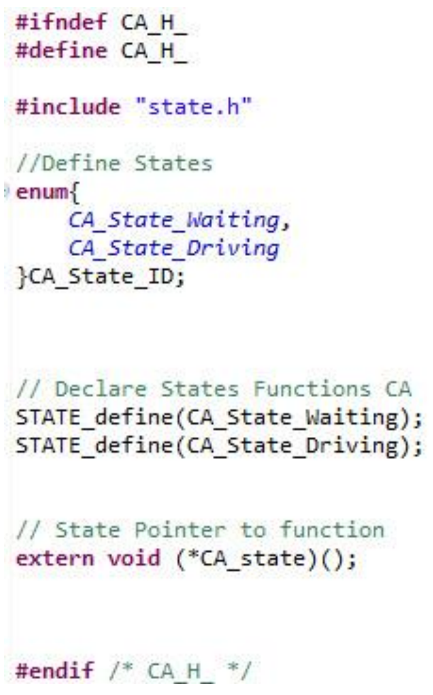
```c
#ifndef DC_H_
#define DC_H_


#include "state.h"

//Define States
enum{
    DC_idle,
    DC_busy
}DC_State_ID;


// Declare States Functions CA
STATE_define(DC_idle);
STATE_define(DC_busy);

void DC_Init();


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


#endif /* DC_H_ */
```

Collision Avoidance Module (CA.c and CA.h)

The Collision Avoidance module determines the appropriate action based on the detected distance from the ultrasonic sensor. It defines states for waiting and driving. When the distance is below a threshold, the system enters the waiting state and sets the DC motor speed to zero, simulating a stop. When the distance is above the threshold, the system enters the driving state and sets the DC motor speed to a non-zero value, simulating movement.



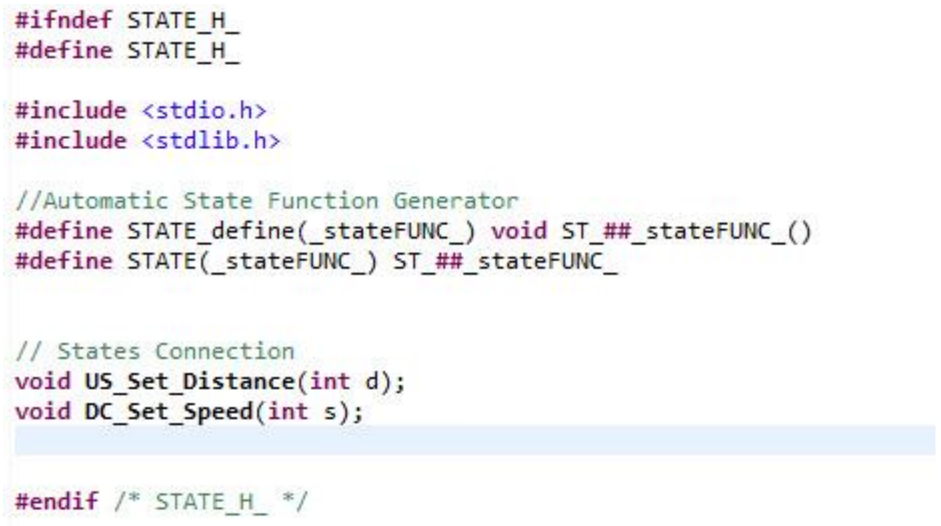
```
#include "CA.h"

// Variables
int CA_speed = 0;
int CA_distance = 0;
int CA_threshold = 50;

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

void US_Set_Distance(int d){
    CA_distance = d;
    (CA_distance <= CA_threshold)? (CA_state = STATE(CA_State_Waiting)) : (CA_state = STATE(CA_State_Driving));
    printf("\nUS --------distance = %d-----------> CA\n", CA_distance);
}

STATE_define(CA_State_Waiting)
{
    //  state name
    CA_State_ID = CA_State_Waiting;
    printf("CA_Waiting State: distance = %d , speed = %d\n", CA_distance, CA_speed);
    //  state action
    CA_speed = 0;
    DC_Set_Speed(CA_speed);
}

STATE_define(CA_State_Driving)
{
    //  state name
    CA_State_ID = CA_State_Waiting;
    printf("CA_Driving State: distance = %d , speed = %d\n", CA_distance, CA_speed);

    //  state action
    CA_speed = 30;
    DC_Set_Speed(CA_speed);
}
```



```
#ifndef CA_H_
#define CA_H_

#include "state.h"

//Define States
enum{
    CA_State_Waiting,
    CA_State_Driving
}CA_State_ID;



// Declare States Functions CA
STATE_define(CA_State_Waiting);
STATE_define(CA_State_Driving);


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



#endif /* CA_H_ */
```

State Generator (state.h)

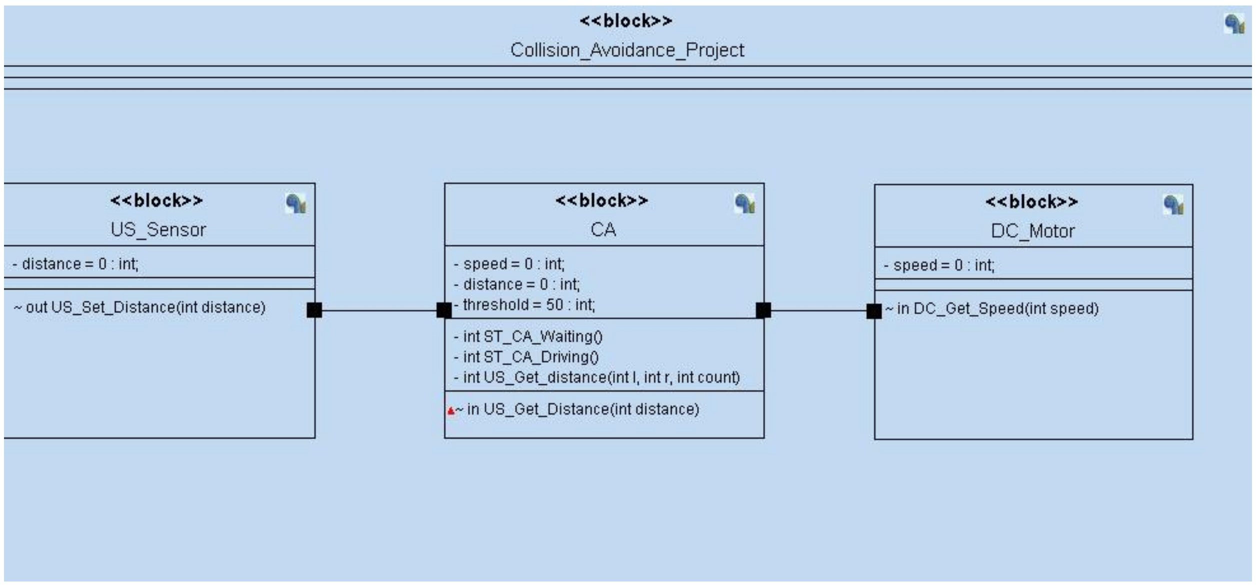
The **state.h** file contains a macro-based state function generator that simplifies the definition of state functions. It provides a convenient way to define states and their associated actions



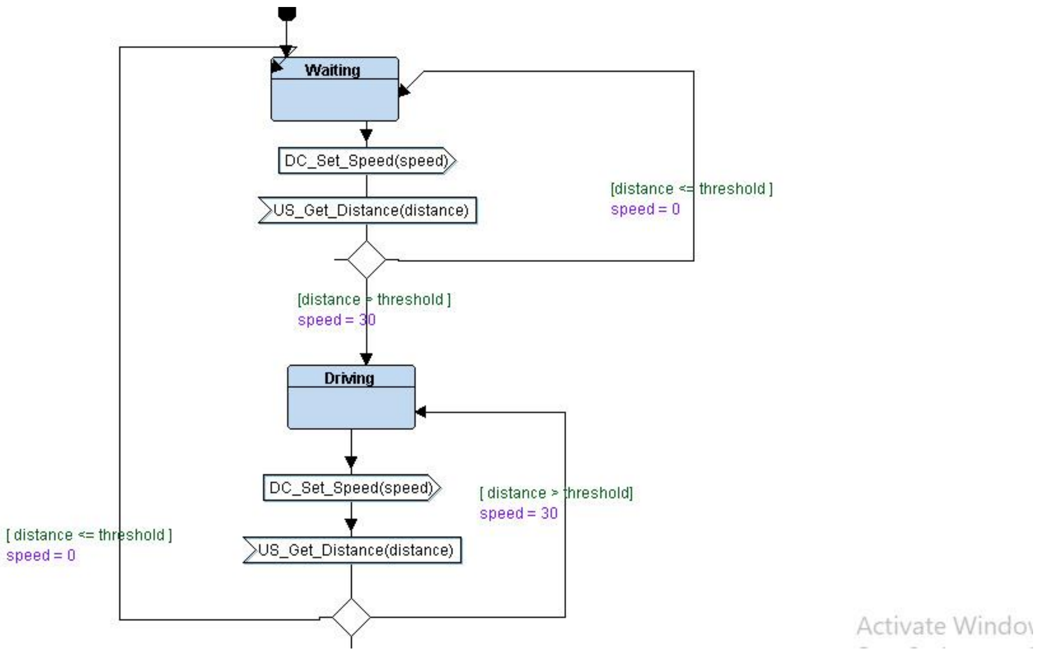
```c
#ifndef STATE_H_
#define STATE_H_

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

//Automatic State Function Generator
#define STATE_define(_stateFUNC_) void ST_##_stateFUNC_()
#define STATE(_stateFUNC_) ST_##_stateFUNC_


// States Connection
void US_Set_Distance(int d);
void DC_Set_Speed(int s);


#endif /* STATE_H_ */
```

Diagrams

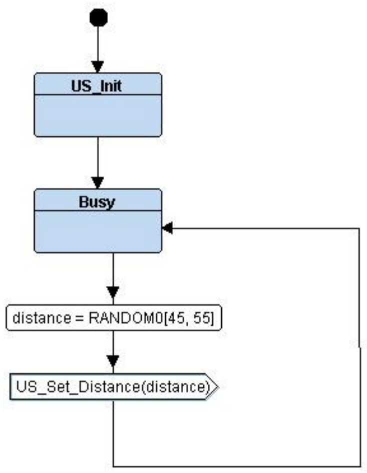
1- Main Block Diagram



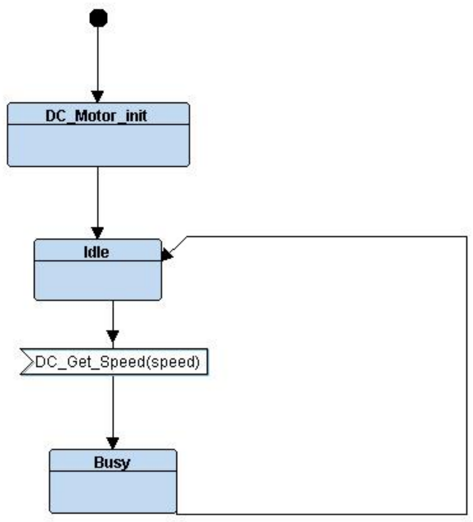
2- CA Diagram



3- US Diagram



4- DC Diagram


DC_Motor_init
Idle
DC_Get_Speed(speed)
Busy