Table of Contents

Table of Figures

1 Introduction

This simple prototype collision avoidance system is a sub-system of a self-driving car system, this report scope is to shows a simple system architecting sequence using UML diagrams for only version one which is only simulate the collision avoidance system with printing data on console.

2 System architecting sequence

2.1 Case study.

2.1.1 Assumptions:

- Physical installation of components will not be designed.
- Sensors, controller, and actuators maintenance will not be considered.
- The controller will not face power cut ever.
- Sensors and actuators will never fail.
- The system will read distance values from an external sensor (sensor driver will not be modeled).
- System will control an external DC motor (DC motor driver will not be modeled).

2.1.2 Versioning:

- Version one: will simulate logic by printing data on windows OS console.
- Version two: full version which will be burned into controller.

2.2 Method:

This system will be implemented using V-model SDLC.

2.3 Requirement:

This section shows a very high view about how this system works and if it really meets the customer needs or not.

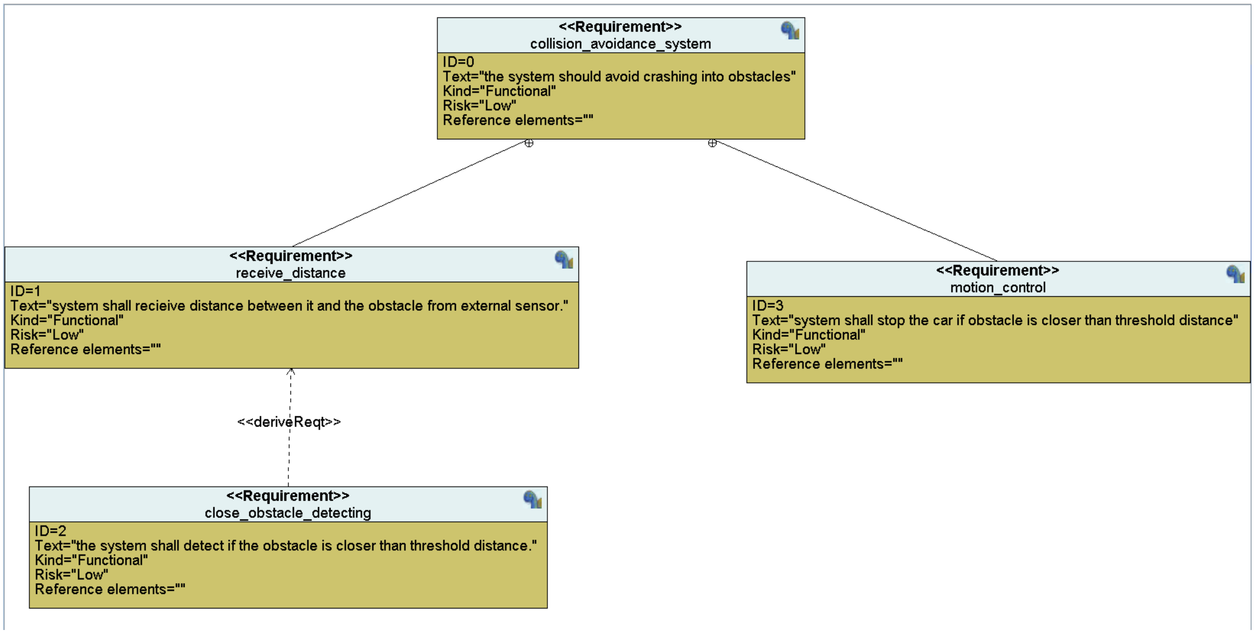

Figure 2.1 Requirement Diagram.

2.4 System analysis:

This section will give a deeper understanding of and how it works.

2.4.1 Use case diagram:

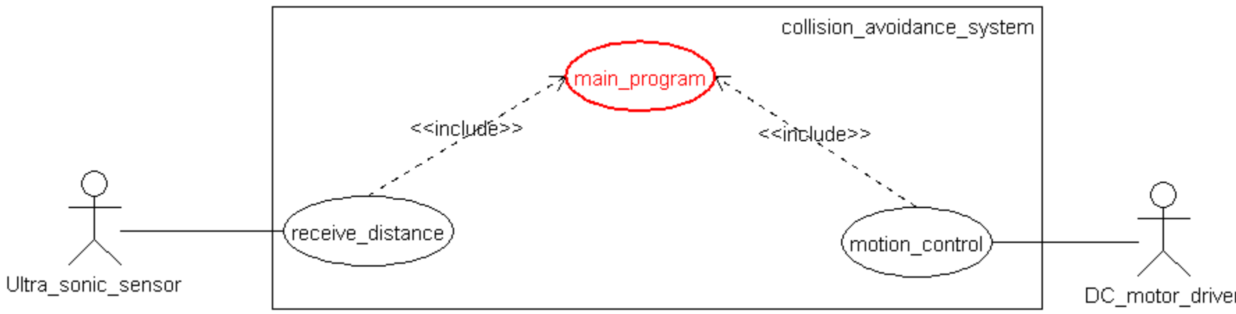

Figure 2.2 Use Case Diagram.

2.4.2 Activity diagram:

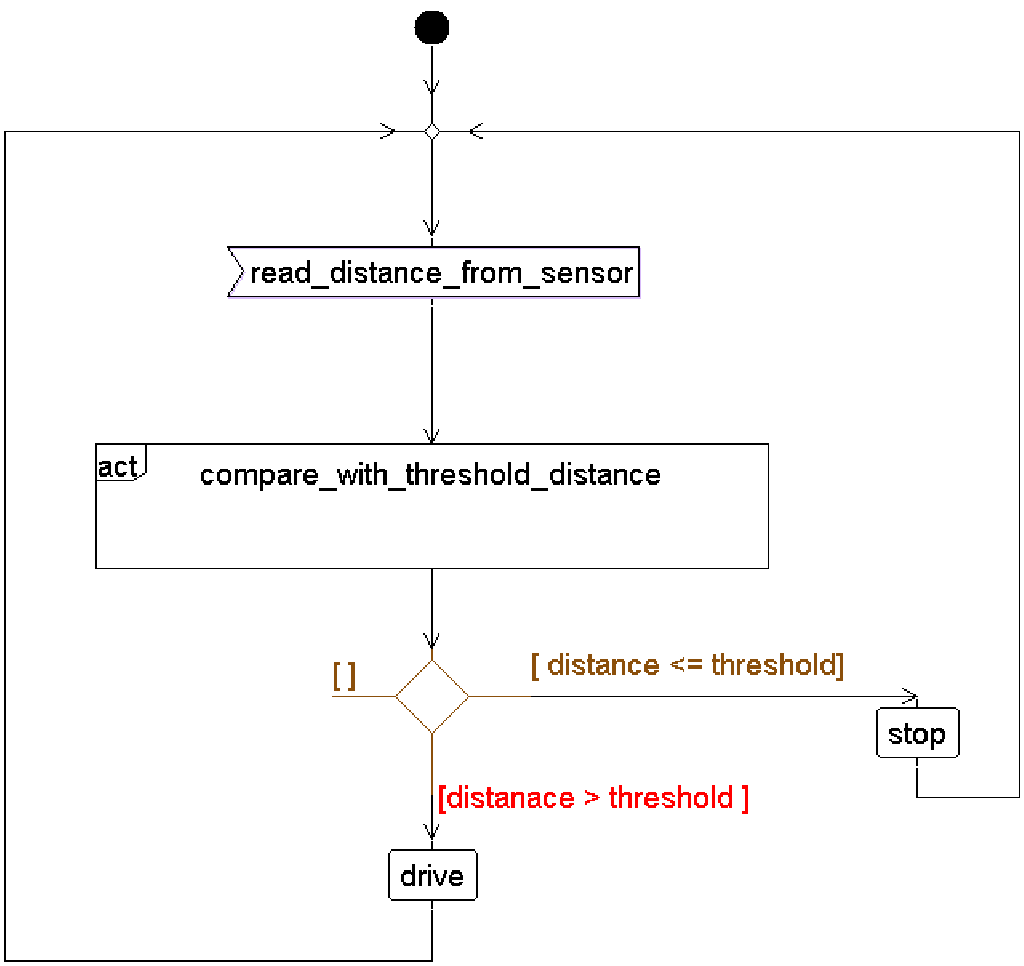

Figure 2.3 Activity Diagram.

2.4.3 Sequence diagram

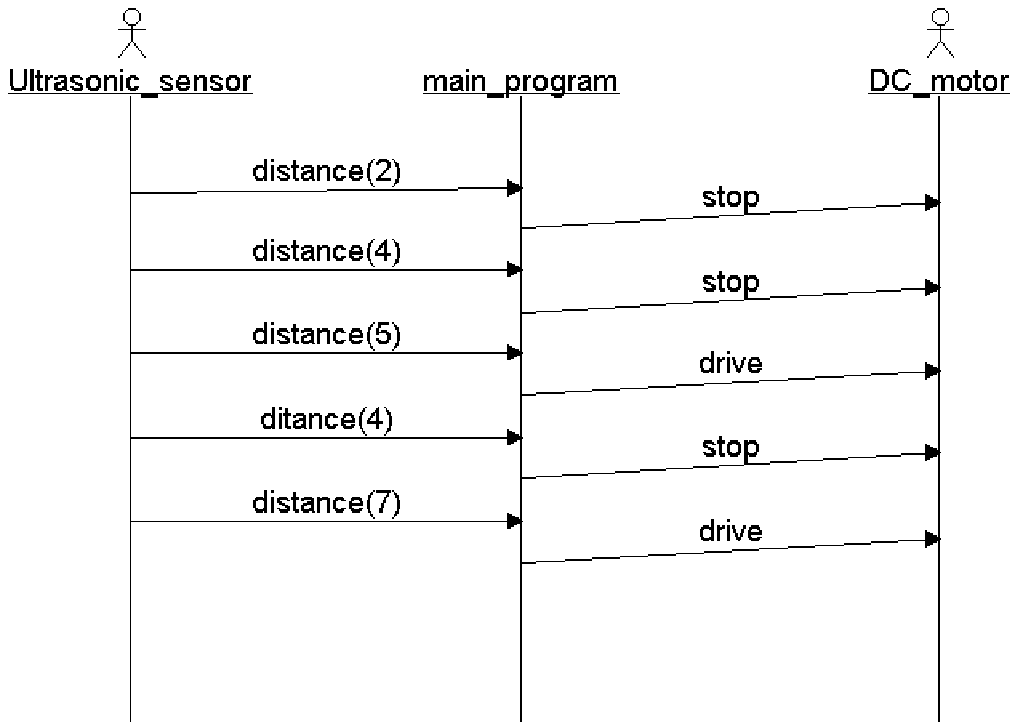

Figure 2.4 Sequence diagram.

2.5 System design:

This section will show how the program will be implemented by a low-level detail.

2.5.1 Overall system block diagram:

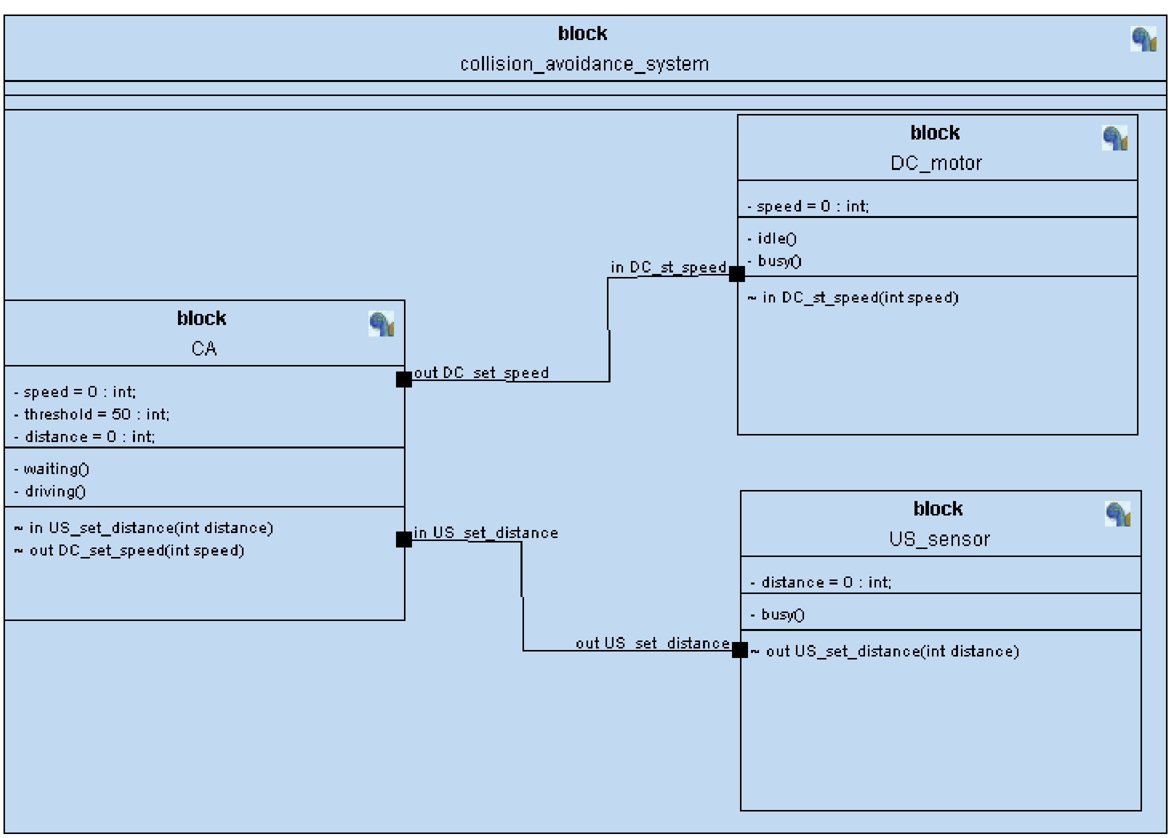

Figure 2.5 System Block Diagram

2.5.2 US logic:

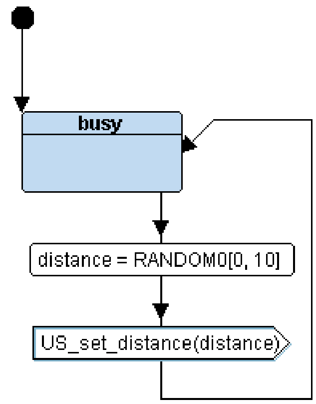

Figure 2.6 Virtual Ultrasonic Sensor Driver.

2.5.3 CA logic:

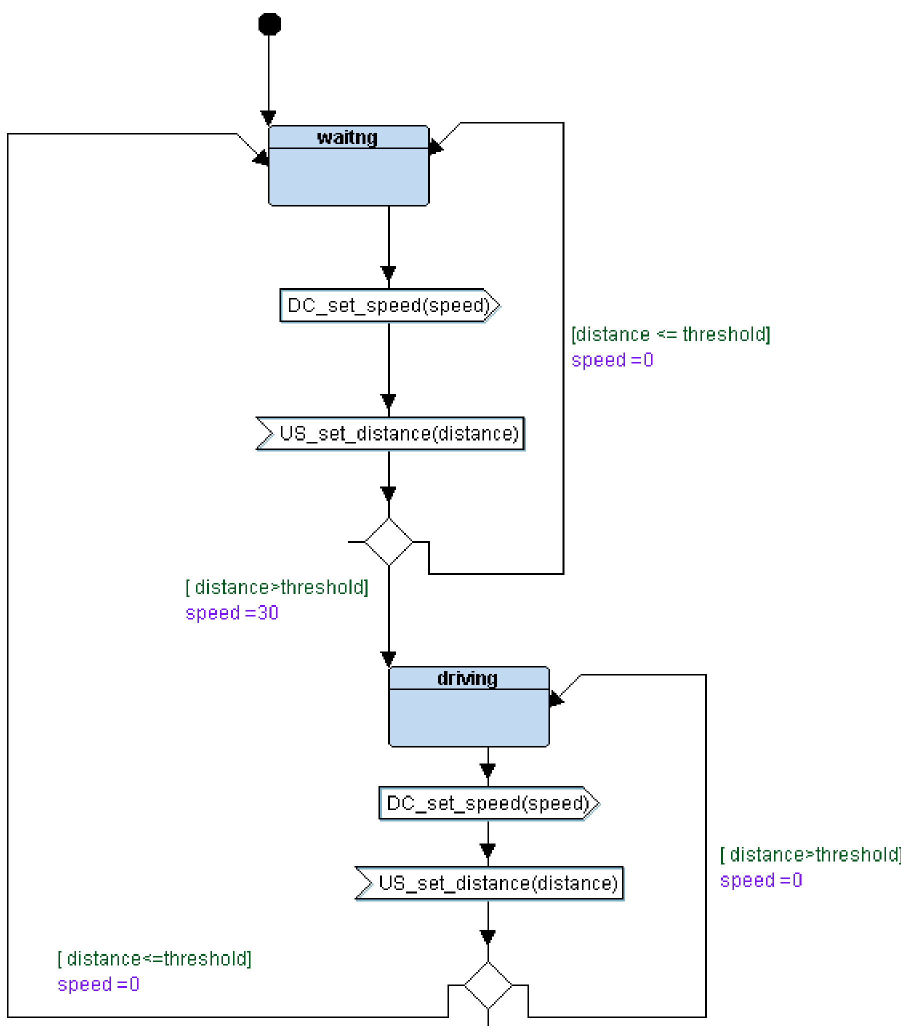

Figure 2.7 Collision Avoidance Main Algorithm.

2.5.4 DC logic:

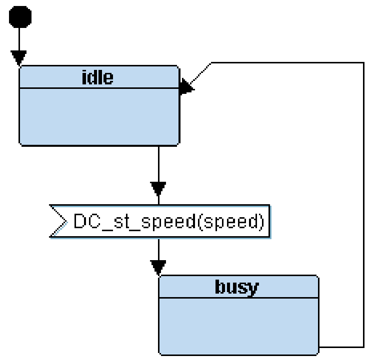

Figure 2.8 Virtual DC Motor Driver.

3 Implementing code:

3.1 Collision avoidance main algorithm

```c
#include "stdio.h"
#include "DC.h"

#define threshold_distance 5

typedef enum {
    _CA_waiting,
    _CA_driving
}CA_status_ID_t;

void(* P_CA_state)() ;
int CA_distance = 0, CA_speed =0;
CA_status_ID_t CA_ID;

void CA_waiting()
{
    CA_ID = _CA_waiting;
    DC_set_speed(0);
    printf("waiting\n");
}

void CA_driving()
{
    CA_ID = _CA_driving;
    DC_set_speed(30);
    printf("driving\n");
}

void US_set_dist(int distance)
{
    CA_distance = distance;
    (CA_distance <= threshold_distance)?(P_CA_state = CA_waiting):(P_CA_state = CA_driving);
}
```

Figure 3.1 Collison Avoidance Main Algoritm.

3.2 Virtual Ultrasonic sensor Driver:

This virtual driver is only used to generate a random value for distance to be used as a test cases.

```c
#include "stdio.h"
#include "CA.h"

typedef enum {
	_US_busy
}US_status_ID_t;

int distance;
US_status_ID_t US_ID;
void(* P_US_state)();

int random_value(int min, int max)
{
	return (rand() % (max-min+1)) + 1;
}

void US_busy()
{
	US_ID = _US_busy;

	distance = random_value(0,10);
	printf("US busy state, distance =%d.\n",distance);
	US_set_dist(distance);
	P_US_state = US_busy;
}

void US_init()
{
	printf("---------------US sensor is initialized---------------\n");
	 P_US_state = US_busy;
}
```

Figure 3.2 Virtual Ultrasonic Driver Code.

3.3 Virtual DC motor driver:

This virtual driver will be only used to take order of driving or not from main algorithm to be tested by different distance values.

```
#include "stdio.h"

typedef enum {
    _DC_idle,
    _DC_busy
}DC_status_ID;

DC_status_ID DC_ID;
int speed;
void(* P_DC_state)();

void DC_idle()
{
    DC_ID = _DC_idle;

    printf("DC idle state, speed= %d.\n\n",speed);

    P_DC_state = DC_idle;
}

void DC_busy()
{
    DC_ID = _DC_busy;

    printf("DC busy state, speed= %d.\n\n",speed);

    P_DC_state = DC_idle;
}
void DC_set_speed(int s)
{
    speed =s;
    P_DC_state = DC_busy;
}

void DC_init()
{
    printf("---------------DC motor is initialized---------------\n");
    P_DC_state = DC_idle;
}
```

Figure 3.3 Virtual DC Motor Driver Code.

3.4 main.c Code:

```c
#include "stdio.h"
#include "CA.h"
#include "DC.h"
#include "US.h"
void setup()
{
    DC_init();
    US_init();
    printf("\n");
}

int main()
{
    setup();
    while (1)
    {
        P_US_state();

        P_CA_state();

        P_DC_state();
    }
    return 0;
}
```

Figure 3.4 main.c Code.

4 Version one running on windows OS console:

```
---------------DC motor is initialized---------------
---------------US sensor is initialized---------------

US busy state, distance =9.
driving
DC busy state, speed= 30.

US busy state, distance =10.
driving
DC busy state, speed= 30.

US busy state, distance =10.
driving
DC busy state, speed= 30.

US busy state, distance =2.
waiting
DC busy state, speed= 0.

US busy state, distance =8.
driving
DC busy state, speed= 30.

US busy state, distance =6.
driving
DC busy state, speed= 30.

US busy state, distance =6.
driving
DC busy state, speed= 30.

US busy state, distance =11.
driving
DC busy state, speed= 30.

US busy state, distance =2.
waiting
DC busy state, speed= 0.
```

Figure 4.1 Version One Test Result On Windows OS Console.