ADAPTIVE CRUISE CONTROLLER MODULE DEVELOPMENT FOR OSKAR ELECTRIC VEHICLE

by

151220102061

Mustafa Özçelikörs

An Engineering Synthesis and Design Project Report

Electrical Engineering Department

June 2015

ADAPTIVE CRUISE CONTROLLER MODULE DEVELOPMENT FOR OSKAR ELECTRIC VEHICLE

by

151220102061

Mustafa Özçelikörs

A Report Presented in Partial Fulfillment of
the Requirements for the Degree
Bachelor of Science in Electrical Engineering

ESKISEHIR OSMANGAZI UNIVERSITY

June 2015

ADAPTIVE CRUISE CONTROLLER MODULE DEVELOPMENT FOR OSKAR ELECTRIC VEHICLE

by

151220102061

Mustafa Özçelikörs

has been approved.

June 2015

APPROVED:

_____

(Full Name of the Chair of the Committee), Chairperson

_____

(Full Name of the other member)

Supervisory Committee

ABSTRACT

In order to prevent traffic accidents and ensure the security and comfort in the traffic, there are several intelligent vehicle technologies, one of which having crucial importance is "Adaptive Cruise Control" (ACC). In this technology, the adjustment of the speed of a vehicle is done by perceiving the environment and controlling through considering obstacles and other vehicle's positioning and speed. This technology involves the usage of high-tech sensor fusion and control techniques. The limitations of the environment are considered and if there is no danger, vehicle keeps its setpoint speed. If there is a vehicle in front, the vehicle reduces its speed by using dynamically determined equations that involves velocity and acceleration constraints. In this project, an Adaptive Cruise Controller module for the electric vehicle OSKAR has been designed.

ÖZET

Trafik kazalarının önlenmesinde, trafikte güven ve konforun temininde, akıllı araç teknolojilerinden olan "Uyarlamalı Hız Kontrolü" (UHK) önemli bir yere sahiptir. Bu teknolojide, üstün sensör işleme ve kontrol teknikleri kullanılarak aracın çevresi algılanır ve aracın, önündeki diğer araç ve engellere bağlı olarak hızını ayarlaması, eğer bir tehdit durumu yok ise, çevredeki kısıtlar da dikkate alınarak aracın sabit hızını koruması sağlanır. Bu projede, OSKAR isimli elektrikli araç için Uyarlamalı Hız Kontrolünü sağlayacak modül tasarlanmıştır.

ACKNOWLEDGEMENTS

This work has been supported the Ministry of Science and Industry under the SANTEZ project of 0508.STZ.2013-2.

This work has been supervised by Assoc. Dr. Rifat Edizkan, Asst. Dr. Ahmet Yazıcı and Elect. Eng. İslam Kılıç.

Table of Contents

1.Introduction

Dynamics and control subject of electrical vehicles is an area which is convenient to development and also studies on this subject are increasing with every passing day. Dependence of our country abroad about the subject is desired to be eliminated while emphasizing the importance of work made in this area. The aim of this project is preventing driver-induced traffic accidents occuring in our country and therefore to reduce loss of life and property.

There are various intelligent systems that are used in vehicles to maximize safety. These systems most of the time communicating with each other to minimize the threats that vehicle may face on the road. Examples to these safety systems can be given as, ABS (Anti-lock Braking System) that prevents the wheels from locking up and uncontrolled skiding while braking, ESP (Electronic Stabilty Program) that improves a vehicle's stability by detecting and reducing loss of traction, and RVM (Rear Vehicle Monitoring) that warns the driver if there is a vehicle in the blind spot which can not be seen with mirrors.

There is also available inteligent systems that enhance driving comfort. Cruise Control is the most important system that can be given as an example to these type of systems. Cruise Control maintains vehicle speed as set by the driver and provides more comfortable driving especially on long journeys. Mentally, driving a vehicle is an extremely challenging activity. Therefore, the driver concentrations are required to keep a high level for long periods. The driver also must be ready to react o changing instant situations. In particular, drivers need to evaluate the distance and relative speed of the vehicle in front continuously and accordingly need to set their speed. Although Cruise Control system reduces difficulties that are mentioned above and provides a

comfortable driving experience, it is insufficient for safe driving and doesn't have a mechanism to prevent accidents while maintaining the speed of the vehicle.

Adaptive Cruise Control (ACC) is a prominent system among the inteligent vehicle technologies which avoids the trafic accidents. Driver origin trafic accidents can be reduced by widespread usage of Adaptive Cruise Control system in vehicles. Adaptive Cruise Control system makes the Cruise Control system safer and also it is designed to prevent dangerous driving conditions. ACC is a semi-autonomous system that ensures pasenger safety and reduces the risk of accidents independently from the driver. The ACC system is responsible for detecting obstacles which could create a situation of collision by various methods and sensors such as RADAR, GPS, LIDAR etc. and adjusting the sped of the vehicle within an advanced controller system. The system shows its benefits especialy on long journeys by maximizing the comfort of the driver. With this technology, even though there is such reasons; thoughtfulnes and loss of reflexes that driver may face, deaths and injuries are occured in the driver-induced accidents can be avoided.

In this project, we are planning to design Adaptive Cruise Control system which decreases driver-induced accidents while maximizing the passenger and vehicle safety and also driver comfort. The project has two main parts such as Low-Level design and High-Level design. In the Low-Level design, mathematical model of the electrical motor vehicle, which the ACC system is implemented on the vehicle is created by using PID controller. Mathematical model of the vehicle will be implemented by selecting one of the methods Root-Locus or Cohen-Coon. Simulation and analysis of Low-Level design with its algorithms are implemented on MATLAB environment. In the High-Level design, control algorithm of the Adaptive Cruise Control is investigated with

literature researches. We are planning to choose one of the control algorithms of Model Predictive Control Method (MPC) or Proportional Integral Derivative (PID) to be used in High-Level. These two control algorithms are eliminated from several control methods considering usabilty, popularity and ease of implementation. We are planning to develop High-Level controller that will produce acceleration output for the Low-Level controller that apply this value to the vehicle.

At the end of the project, the module that will implement the ACC system to be applied on vehicle will be developed. The module wil consist of Low-Level and High-Level control units, sensors (GPS, Encoder, RADAR etc.) and user interface unit (KAB) that shows information about the system such as the destination from the vehicle in front, speed, acceleration, etc.

2.Motivation

The number of vehicles in the world, including our country is increasing every day. Because of the number of vehicles increases, traffic density increases, that leads to vehicle accidents which are resulting in injury and death events and also financial losses. Acording to Emniyet Genel Müdürlüğü Trafik Hizmetleri Başkanlığı web page, 1.207.354 traffic accidents was occured in 2013. In these accidents, 3.685 citizens have lost their lives and 274.829 citizens have ben injured [1]. The most striking statistic about cause of the accidents is observed in defect rate of driver, pedestrian, road, vehicle and pasenger. Drivers are found defective in 135.605 of 154.11 trafic accidents with 88 percentage [2]. This statistic clearly shows the burden on the driver is needed to

aleviate in driving. In other words, vehicles need to be “smart” to detect the driver’s error and prevent the accidents.

In recent years, the idea of an all-domestic car is seriously putting into words in our country. Since the vehicle technology has been varying in the interdisciplinary manner, only mechanical advance is not enough to make this idea real. State-of-the-art technology about the electric-powered/eco-friendly vehicle, intelligent vehicle and safety systems are the other essential areas to focus on. In this regard, studies about these subjects are very important to gain experience, train specialized engineers about vehicle systems area and attract the domestic industry’s attention on this subject. In this respect, studies that are constituted by university-industry collaboration have a special importance in order to achieve mentioned goals.

3. Literature Review

After the Cruise Control (CC) system had announced in the beginning of 1990s, worldwide studies has begun about this subject both in academic and commercial areas. Advanced Cruise Control (ACC) system has announced soon afterwards the CC system’s announcement. Today, a lot of studies about ACC system can be found with various methods and for various types of vehicles.

After the first studies which explained idiosyncrasy of ACC system [3,4], the further studies has became varied. There are some studies in academic literature which is aimed to improve the ACC algorithm in general [5,6,7,8]. It should be noted that in a large majority of the studies about ACC systems, PID Control [5, 9,10,11,12] and MPC (Model Predictive Control) [13,14,15,16,17] is used. But there are also some studies with PI Control [7, 18], Fractional Order Control [19] and Time Delay Control [3].

There is also one study that is about comprison of PID and MPC in ACC systems [20]. It is stated that PID is easy to implement and it allows high performance in microcontrollers. Negative sides of PID Control are: limited functional structure and complications in cases that include more than one variable. According to this paper, MPC's most important supremacy is the ability of using operational limitations (such as collision avoidance) incorporated with the control algoritm.

In literature, there are some studies about implementation and algorithmic development of ACC for low speed values (Low-Sped ACC, Stop-and-Go ACC) [1, 19,21] and improvement for fuel consumption [6]. Also there are some studies about vision-based ACC by pattern-matching [22], the improvement of sensors/detectors that are used in ACC systems [23, 18] and the ACC systems specialized for 2-wheel vehicles[24]. Some examples of university-company cooperated studies can also be found in academic literature [25,26].

There are numerous vehicles in market that use CC system. At the beginning of 1990's CC was a luxury for cars and it caused a big difference in price [27]. Nowadays, CC system is used in many cars, furthermore, some elite car manifacturers is using ACC system with their own commercial names and different features. Lexus, BMW, Acura, Mercedes Benz, Volvo, Volkswagen, Audi and Ford can be given as examples for these brands [28, 29, 30,31,32,33,34,35].

Since it is more useful in bad weather conditions and more affordable in price, RADAR and laser are the most common sensor/detectors that are used in commercial ACC systems. Apart from this, commercial ACC systems assorted by: numbers and features of RADAR or laser that are used [36,37, 38, 39,40,41,42], specialized speed ranges for

high performance work [38, 39, 40, 41, 42] and having extra features such as strip tracking system[42].

4.Methodology

The project is planned to progress under three work packages.

The first work package, “Determination of the System Requirements”, includes the determination of motor’s characteristics and peripheral units in order to accomplish the design goals of the project. In this part, following tasks have been created: Determination of low-level and high-level mathematical models; designing an ARM-based control card including the microcontroller communication modules and peripheral units such as converter, amplifier and regulator. This work package also includes the determination of software needs of the system and operating platforms of detectors and sensors. Almost all of the tasks mentioned above are completed except the implementation of communication interfaces. This part is planned to be done after the end of the second work package.

The second work package, “Design and Simulation of the Adaptive Cruise Control System” includes following tasks: For both low-level and high-level controllers, designing acceleration and velocity control algorithms for the motor that is used in OSKAR (with the algorithms that are accepted in the academic field such as PID and MPC); applying these algortihms with Root-Locus or Cohen-Coon methods; simulation and analysis of these algortihms on MATLAB environment. Until now, low-level velocity control algortihm with PID controller is designed with Root-Locus method and successfully simulated in MATLAB. Currently, low-level team is working on modifying the velocity control algorithmto obtain acceleration as output. Due to the

diffuclty of obtaining motor's characteristic constants, using different methods (such as obtaining a model by determining mechanical time constant) is still an alternative way that put aside. High-level control algorithm part is still on the literature research level. In High Level control algorithm, almost all methods that are used in literature and also market are investigated with researches. Seven different high-level control algorithms are eliminated with their usability, popularity and ease of implementation when the electrical car environment is considered. Under this conditions which are mentioned, although we conclude the High Level control algorithm on Model Predictive Control Method (MPC) or Proportional Integral Derivative (PID), there is no exact control algorithm we choose to use in High level in the strict sense.

"Design of the System Electronics and Implementation of Software Libraries" is the last work package of the project. In this package, it is aimed to design the electronic parts of OSKAR (low-level, high-level boards and peripheral units mentioned in the first work package) to satisfy both low and high level specifications that are determined in first and second work packages. In this electronic board, it is planned to have; communication units that are used for reading data from sensor/detector units such as encoder and RADAR (CANBUS); interfaces to read data from throttle (ADC) and to send voltage values to motor driver (DAC and PWM). Communication between high-level and low-level controllers is planned to apply with protocols such as TCP/IP and CANBUS. In this package, software libaries for mentioned units are planned to realize with a modular structure. After these, a Graphical User Interface (GUI) which will show velocity, acceleration and distance values is desired to designfor OSKAR. For this task, QT or Android based software is planned to use. Until now; ADC, DAC and PWM modules are implemented in low-level controller. Also, it has been achieved to read data

from RADAR with CANBUS and the CANBUS library for RADAR is almost completed. Designed system block diagram can be seen in Figure 1b.

OSKAR vehicle is shown in Fig 1a.

Fig 1a. OSKAR Vehicle

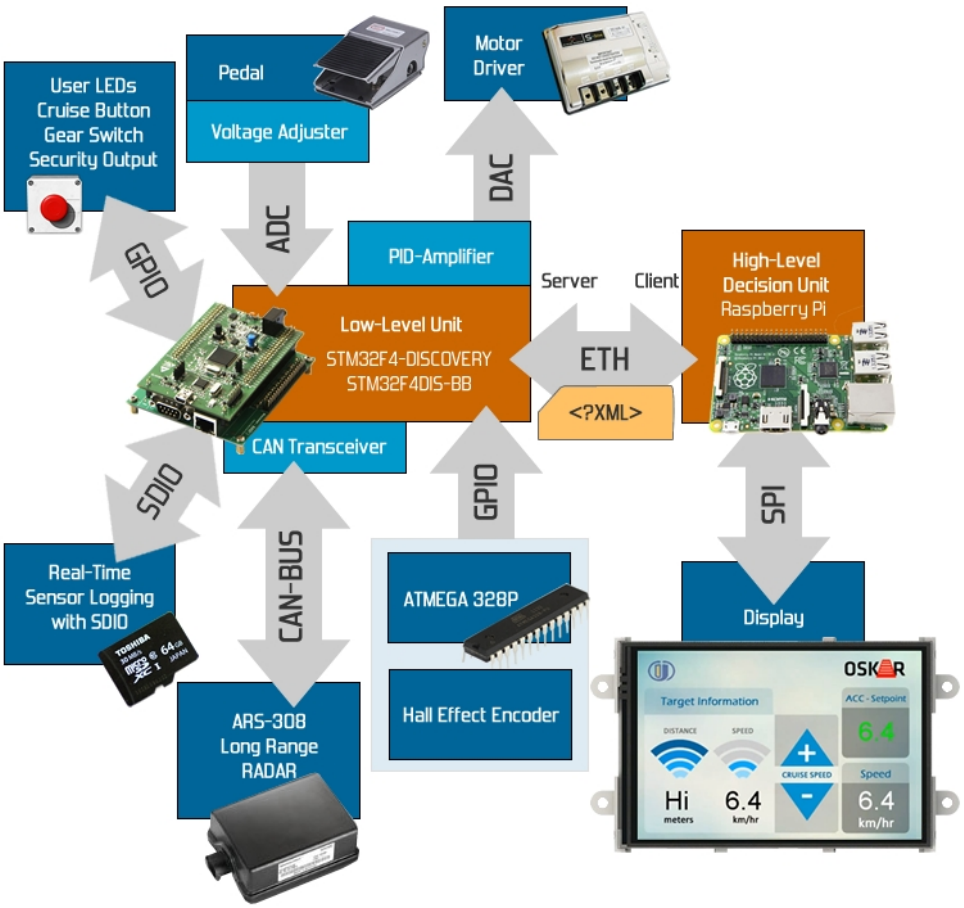

Fig 1b. System Block Diagram

5. Work Packages

The work breakdown structure regarding the project is given below:

WP	WORK PACKAGE NAME	Septemb	Octobe	Novemb	Decemb	January	Febru	March	April	May	June
1	Literature Review										
1.1	Low Level Literature Review										
1.2	High Level Literature Review										
2	CC and ACC Design and Simulation										
2.1	Conventional Cruise Design										
2.2	Adaptive Cruise Controller Design										
3	System Electronics Design										
3.1	Low Level Electronics Unit Design										
3.2	High Level Electronics Unit Design										
4	System Software Implementation										
4.1.	Low Level Software Implementation										
4.2.	High Level Software Implementation										
5	System Tests and Final Design										

Work Package 1: Literature Review: In this work package, the speed control of a DC motor (as low level) and the adaptive cruise controllers (as high level) are researched.

Work Package 2: CC and ACC Design and Simulation: This work package involves a conventional cruise controller design on the low level unit; as well as the Adaptive cruise controller design on both high and low level units. This work page tends to work simultaneously with the "System Electronics Design" and "System Software Implementation" work packages since the design and implementation process is going simultaneously, for the troubleshooting purposes.

Work Package 3: System Electronics Design: In this work package, two electronic systems will be designed one of which is low level and the other is the high level controller. Low level controller (based on STM32F4-DISCOVERY) is responsible of obtaining RADAR, Encoder, Throttle sensor information using various peripherals such

as USART, CAN-BUS, ETH, GPIO, ADC; while driving the motor using the DAC unit. The security measures and system power amplifiers will also be included in the low level electronics design. Moreover, the low level controller is responsible for communicating with the high level controller unit. The high level controller unit is based on Raspberry Pi which is going to be the main display and input unit for the system and will also be the decision mechanism for the Adaptive Cruise Controller design.

Work Package 4: System Software Implementation: In this work package, the software developement in the environments of EWARM (for low-level sensor and motor controlling purposes), PythonQT (for high-level ACC control mechanism and display unit control) will be used. In the low level software, there will be a library written that will communicate with ARS-308 long range RADAR via Controller Area Network- BUS (CAN-BUS). Moreover, the throttle sensor, the DAC unit control, the encoder sensor reading, indicator LEDs, and USART communication for troubleshooting will also constitute the low level software. As for the high level software, the Raspberry Pi device TCP/IP socket communication will be carried out to communicate with the low level unit. Also, a display will be programmed in PythonQT in order to act as the display unit for the system. Moreover, ACC algorithm will also be implemented in the high level unit.

Work Package 5: System Tests and Final Design: In this work package, the overall system will be tested as a whole with different types of inputs, in different circumstances in order to test the robustness and responsiveness of the system.

5.1. Work Organisation

Work organisation is given below:

Low-Level Design and Implementation:

İbrahim Yıldırım, Deniz Can Dayan

High-Level Design and Implementation:

Atakan Ondoğan

Hardware Design:

All team members

Software (LL-HL Software Infrastructure , Radar Sensor Driver, CAN-BUS, TCP, RTOS, Display):

Mustafa Özçelikörs

Encoder Design and Software, PCB Realization:

Metin Kundakçığlu

5.2. Achievements

The all work packages except WP5 has been completed.

6. System Design

6. 1. Low Level

6.1.1. Low Level Design

As mentioned before, OSKAR electric vehicle has the Advanced Cruise Control system which uses RADAR. This ACC system works under two control systems: Low-level control and high-level control. Low-level control system's inputs are desired velocity that is periodically taken from high-level controller and the current velocity that is obtained from the encoder. With a control algorithm, low-level controller gives out the voltage value that corresponds to the velocity that is applied to the motor driver.

As the low-level controller, STM32F4-DISCOVERY microcontroller board has been selected. As the next step, mathematical model of the brushed 24V DC motor needed to be obtained. After a comprehensive literature research, two main paths were defined to obtain the mathematical model and to design a low-level controller system to electric vehicle OSKAR. First one was to use a common mathematical model for motor and to design a proportional-integral-derivative (PID) controller by Root-Locus method and also by PID Tuner tool of MATLAB. Second path was to obtain the mathematical model and designing PID controller by Cohen-Coon method.

(i) Design of Low-Level Control System by Root-Locus Method

In order to design a PID controller, the model of the system needs to be obtained. After the literature research, a model for the DC motor that is commonly used (shown in Fig 2, Fig 3) in literature was selected [*]. First step in modeling the DC motor was to

derive its equations. This model has the voltage as the input, and the velocity as the output.

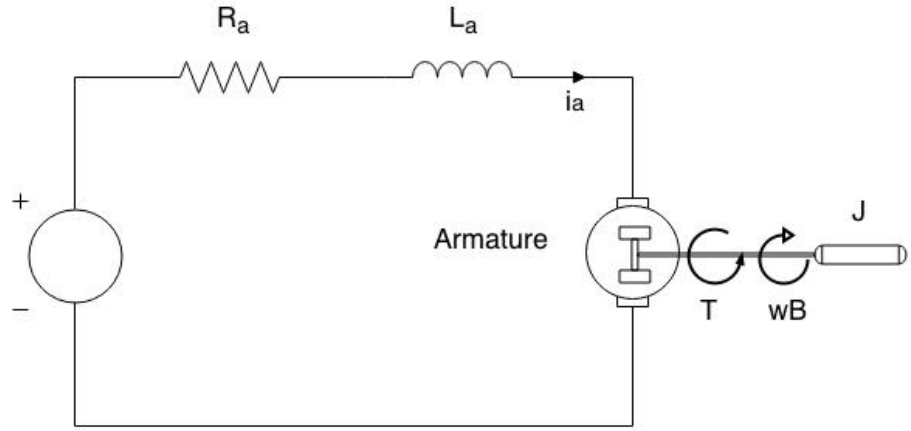

Fig 2. Electromechanical model of DC motor

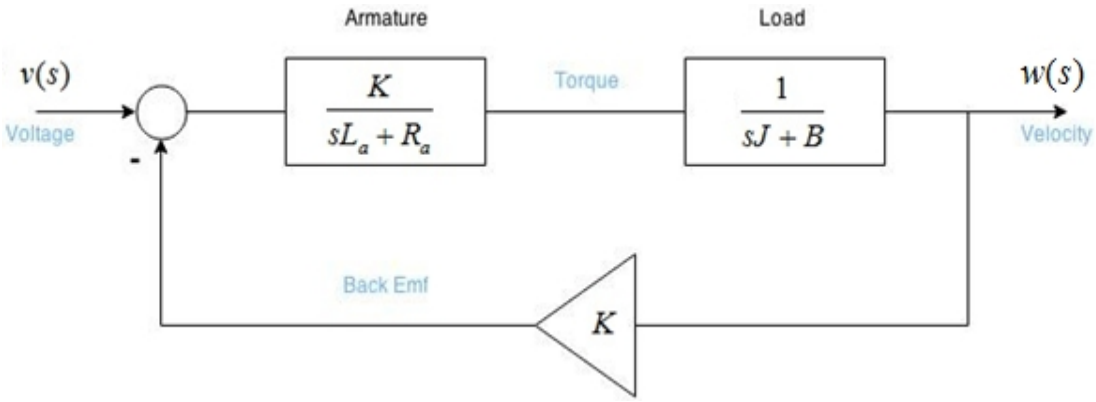

Fig 3. Block diagram for the mathematical model of DC motor

Using this block diagram, the transfer function of the model has been obtained.

$$G(s) = \frac{w(s)}{V(s)} = \frac{K}{(L_a J)s^2 + (L_a b + R_a J)s + (R_a b + K^2)}$$

As can be seen from the figure above, the model includes some characteristic motor constants. For the motor that is used in OSKAR, these constants were used as shown below.

Table 1. DC Motor Constants

Constant Name	***Symbol***	***Obtained Value***	***Unit***
EMF Constant	K	1.4882	Nm/A
Moment of Inertia	J	0.27	$Kg.m^2$
Friction Constant	B	0.1044	Nm.s/rad
Armature Inductance	L_a	0.082	Henry
Armature Resistance	R_a	0.2957	Ohm

The system above was uncompensated. It needed to be compensated in order to have acceptable transient and steady-state response characteristics. PID control algorithm was decided to be used for this purpose. PID control was applied by first designing a PD (to satisfy desired transient response characteristics), then a PI controller (to satisfy desired steady-state characteristics) to the system in order to modify the system to desired transient and steady-state characteristics. Since it is advantageous concerning the ease of implementation and it is more applicable in low-level controllers -it allows higher performance on restricted hardware-, PID control method has been selected among the others such as Model Predictive Control and Sliding-Mode Control.

PID algorithm can be applied with various methods. Root-Locus was the first method selected. The design by the root-locus method is based on reshaping the root locus of

the system by adding poles and zeros to the system's open-loop transfer function (Fig 4) and forcing the root loci to pass through desired closed-loop poles in the s plane. MATLAB is used for the design of PID with root-locus method.

Design criterions for the PID system was a settling time of 0.4 seconds, 8% overshoot and zero steady-state error. MATLAB code and the outputs are shown in Table 2.

Table 2. MATLAB code of the PID design with root-locus method

```
armature = tf([Kt],[La Ra]);

load = tf([1],[J B]);

geribesleme = Kt;

sys1 = series(armatur,load);

sys2 = series(sys1,geribesleme);


%obtaining the system's poles and zeros

[num,den] = tfdata(sys2,'v');

[z,p,k] = tf2zp(num,den)      %p(1)=-3.6061 p(2)=-0.3867


%calculation of damping ratio

Mp = overshoot/100;

zetnum = (log(Mp)/pi) * (log(Mp)/pi);

zet = sqrt(zetnum/(1+zetnum));

theta = acos(zet);      %the angle that zeta line makes


%dominant pole location


zetawn = 4/ts;     %zeta.wn calculation for desired settling time

wd = zetawn*tan(theta);
```

```
%PD design part:


%applying angle criterion


ang = 180+(atan2(wd,zetawn+p(1))+atan2(wd,zetawn+p(2)))*180/pi;
zer = zetawn-(wd/tan(ang*pi/180));      %PD zero=15.82


numpd = [1 zer];    %PD compensator


%PI design part:


numpi = [1 0.01];    %zero of PI is selected close to the origin
denpi = [1 0];
sys3 = tf(numpi,denpi);
[num2,den2] = tfdata(sys3,'v');


% obtaining the transfer function of the PID controller and plotting
the step response of the closed-loop compensated system:
numpid = conv(num,(conv(numpi,numpd)));
denpid = conv(den,denpi);
syspid = tf(numpid,denpid);


wn = 0:1:30;
rlocus(syspid);
sgrid(zet,wn);
pause;
[kpid,ppid] = rlocfind(syspid)
```

```
numcl = numpid*kpid;

dencl = poly(ppid);

syscl = tf(numcl,dencl);

step(syscl)
```

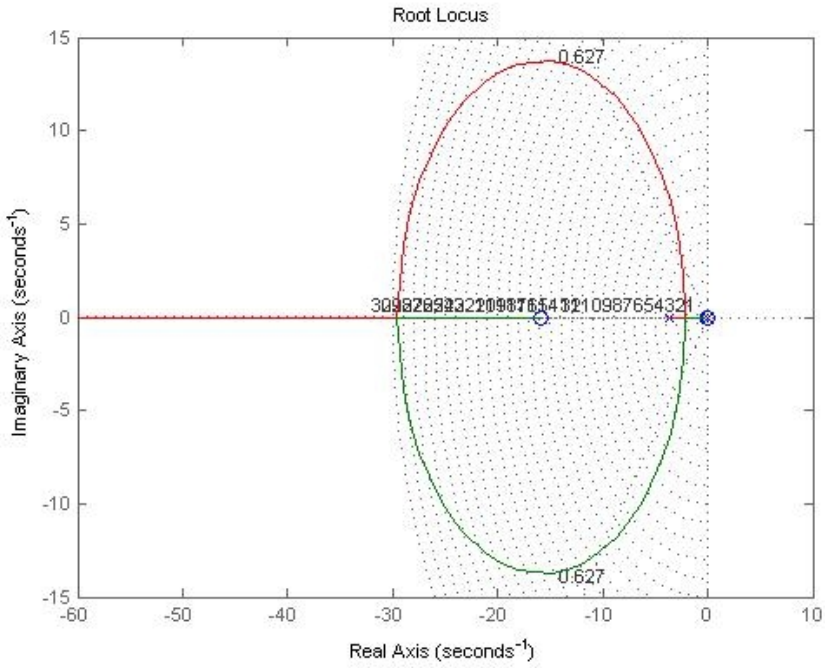

Fig 4. Selecting the intersection of the zeta line and the root-locus

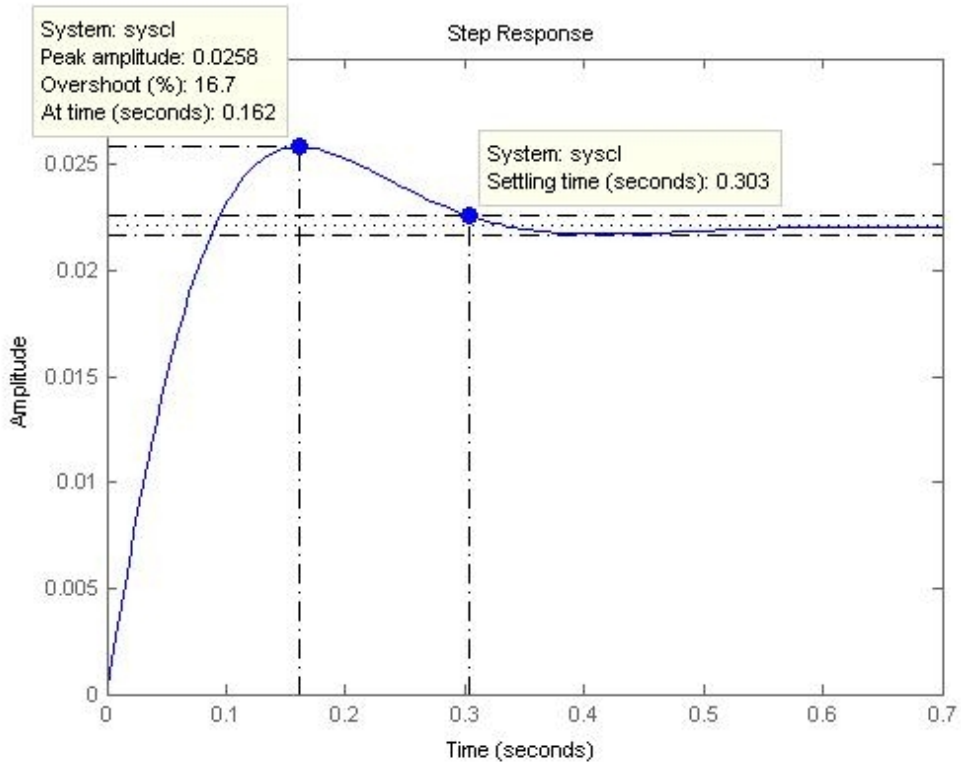

Fig 5. Step response of the PID compensated system obtained by root-locus method

By root-locus method, transfer function of the PID controller for the system was found as below:

$$G_{PID}(s) = \frac{0.1596s^2 + 2.528s + 0.02526}{s}$$

At this stage, it was possible to obtain proportional, integral and the derivative gains of the controller. These gain values were going to be used for implementing the PID controller in low-level controller.

$$K_p = 0.0253 \quad , \quad K_i = 2.5275 \quad , \quad K_d = 0.1596$$

Another PID controller design method is to use MATLAB's "PID Tuner" tool. This readymade special interface of MATLAB allows user to design controller with various

design modes and controller types. User can simultaneously see the response of the compensated system as the compensator changes.

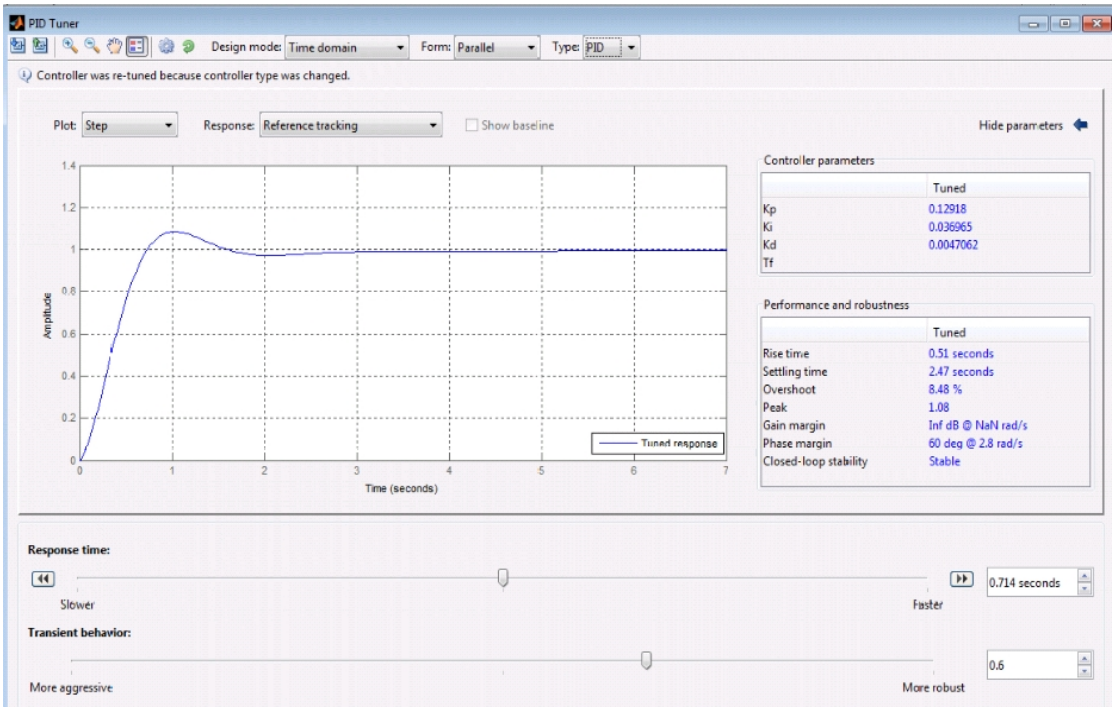

Fig 6. Controller design with PID Tuner tool of MATLAB

As seen in Fig 6, the following gain values were obtained from using the PID Tuner for the motor's uncompensated transfer function:

$$K_p = 0.12918 \quad , \quad K_i = 0.036965 \quad , \quad K_d = 0.0047062$$

(ii) **Design of Low-Level Control System by Cohen-Coon Method**

There are several ways to determine what values to use for the proportional, integral, and differential parameters in the controller, and used the Cohen-Coon method is one of the method . The Cohen-Coon method is classified as an 'offline' method for tuning, meaning that a step change can be introduced to the input once it is at steady-state. Then the output can be measured based on the time constant and the time delay and this response can be used to evaluate the initial control parameters [2]. By looking at the

system's response to manual step changes without the controller operating, initial values for the PID parameters are determined and then tune them manually [1].

There are several advantages for using Cohen-Coon method on modeling and control design. First of all, for the systems that are hard to obtain mathematical model and find constant parameters, Cohen-Coon method is advantageous since the model is obtained by applying step inputs and observing the responses. Also, this method is used for systems with time delay and it has quicker closed loop response time. On the other hand, using Cohen-Coon may cause unstable closed-loop systems and it can only be used for first-order models including large process delays. Also, approximations for the K, τ, and τ_{del} values might not be entirely accurate for different systems.

In this section, the process in Cohen-Coon tuning method was mentioned. For this system, output is velocity and its values were obtained through encoder. First of all, the step input was applied to the system until the process reached steady state.

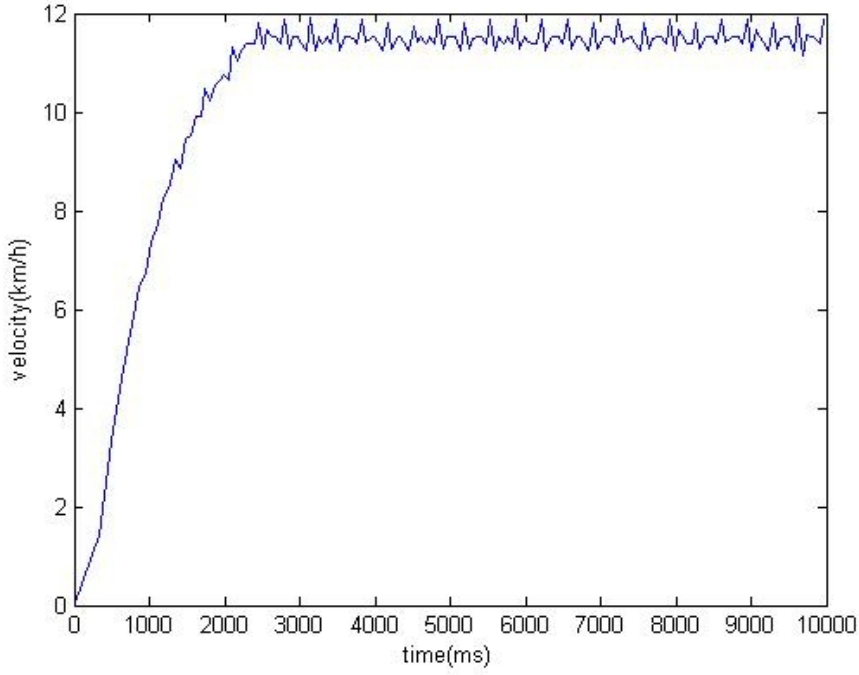

Fig 7. Step Test for Cohen-Coon Tuning

After obtaining the step graphics of the system, 5 parameters were determined by the direct use of Fig 8.

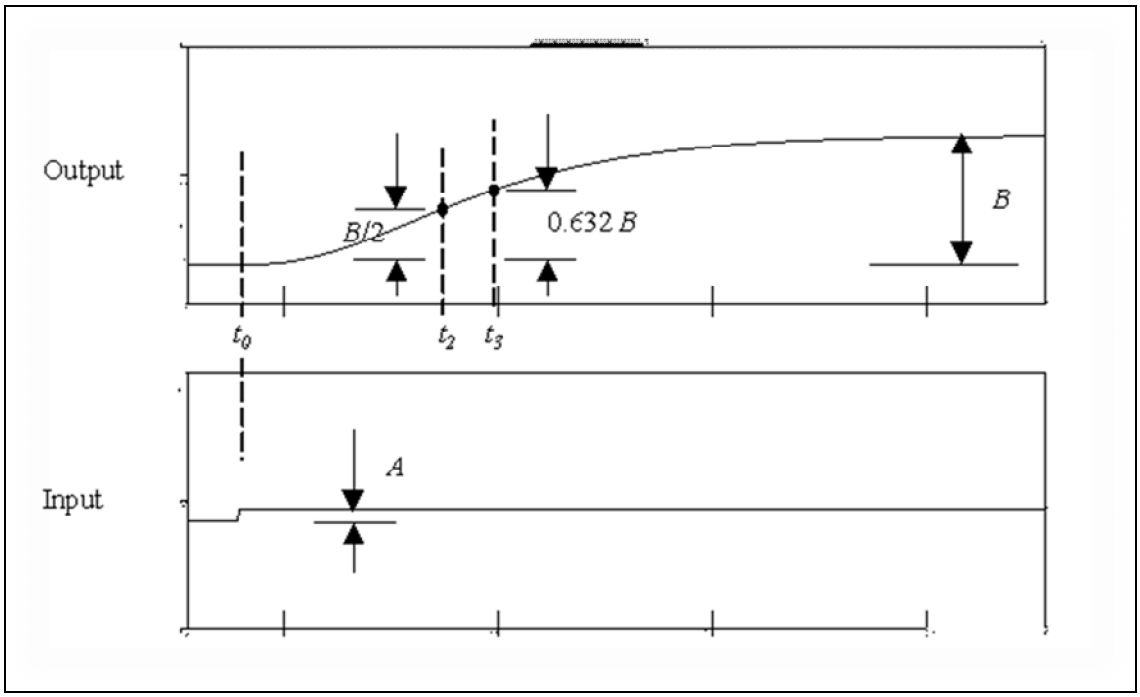

Fig 8. Cohen-Coon Representative System Output and Input

Table 3. The main parameters of system input and output

t_0 : Time at input step start point
t_2 : Time when the system reaches half of steady state point
t_3 : Time when the system reaches 63.2% of steady state point
A : The magnitude of the step input
B : The magnitude of the system output in steady state

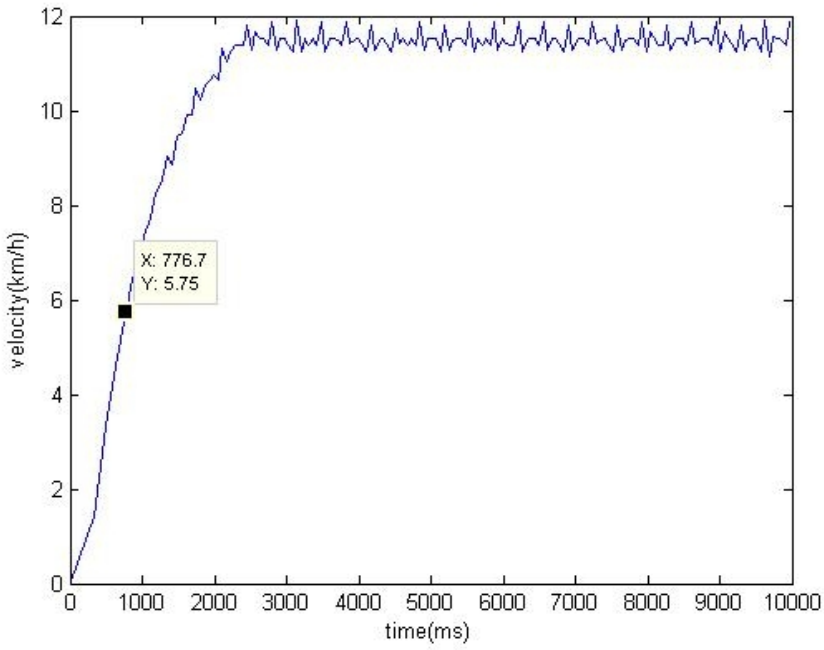

Fig 9. Determination of t2

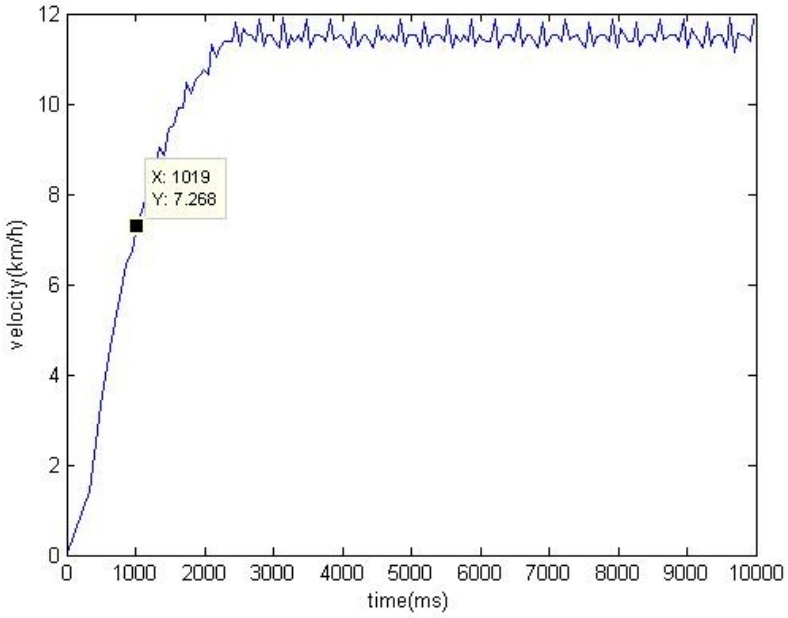

Fig 10. Determination of t3

Then using the measurements at t0, t2, t3, A and B, the process parameters t1, τ, τdel, K and r were calculated by using MATLAB as shown in Table 4.

Table 4. Obtaining the process parameters

```
% The main parameters


B = 11.5;  %velocity magnitude at steady state

A = 3.3; %step input magnitude

t0 = 0; %time at step input start point

t2 = 0.7767;%time when the system reaches half of ss point (B/2)

t3 = 1.019;%time when the system reaches 63.2% of ss point (0.632*B)


% Calculation of process parameters


t1 = (t2-(t3*log(2)))/(1-log(2));

t = t3-t1;

tdel = t1-t0;

k = B/A;

r = tdel/t;
```

The results are;

$t_1 = 0.2294 \sec$, $r = 0.2905$, $K = 3.4848$

$\tau = 0.7896 \sec$, $\tau_{del} = 0.2294 \sec$

After that, the controller parameters were obtained based on Table 5.

Table 5. Cohen-coon tuning rules

Controller Type	K_c	τ_{int}	τ_{der}
PID	$\frac{1}{rK}(\frac{4}{3}+\frac{r}{4})$	$\tau_{del}\frac{32+6r}{13+8r}$	$\tau_{del}\frac{4}{11+2r}$

The implementation of specified formulas and obtaining the PID function are shown as follows:

```
%Controller parameters for PID


Kc = (1/(r*K))*((4/3)+(r/4))

tint = tdel*((32+(6*r))/(13+(8*r)))

tder = tdel*(4/(11+(2*r)))


%Obtaining of the PID function


%tint/s

num = tint;

den = [1 0];

sys = tf(num,den)


%tder*s

num1 = [tder 0];

den1 = 1;
```

```
sys1 = tf(num1,den1)


 %PID transfer function


Gpid = Kc*(1+sys+sys1)
```

The obtained results and the gain constants of PID controller (K_p, K_i and K_d) are :

$$K_p = 0.0253 \quad , \quad \tau_{\text{int}} = 0.5051 \quad , \quad \tau_{der} = 0.0792$$

$$G_{PID} = \frac{0.11s^2 + 1.389s + 0.7015}{s}$$

$$K_i = 0.7015 \quad , \quad K_p = 1.389 \quad , \quad K_d = 0.11$$

After obtaining the PID gain constant, speed tests were made in laboratory environment. But these coefficients caused non-negligible oscillations. Although the gain coefficients of PID were tuned for solving this problem, desired results were not obtained. The following MATLAB figures that based on some coefficients and 5 km/h set speed value was occurred during the tests.

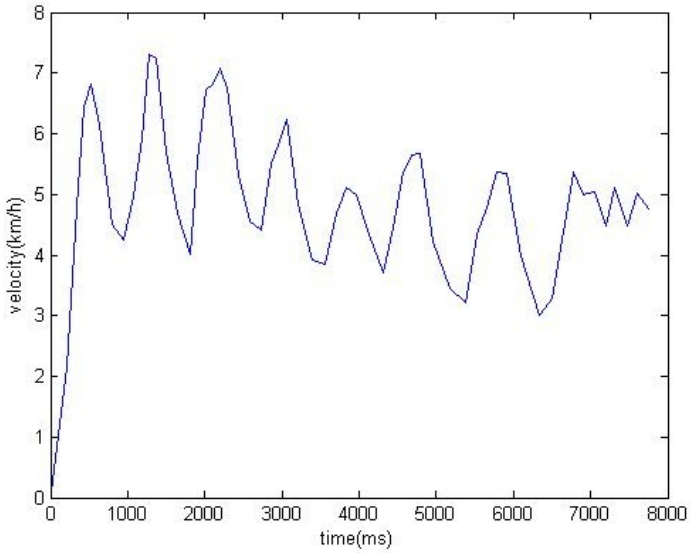

Fig 11. Kp=0.8, Ki=0.4, Kd=1, Setpoint=5 km/h

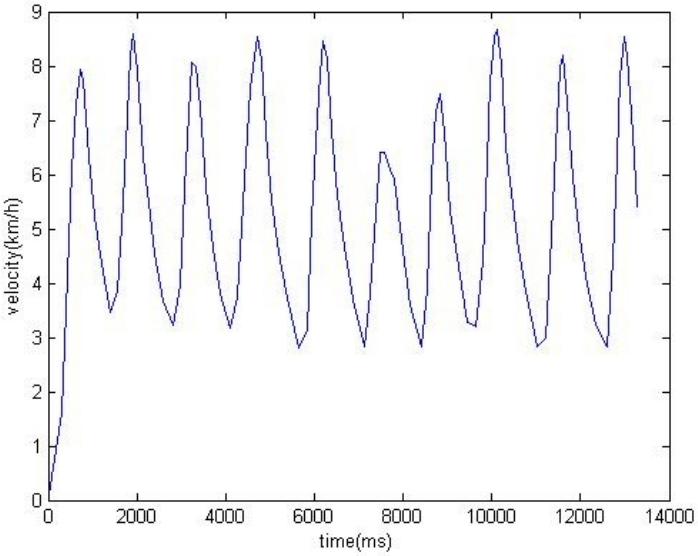

Fig 12. Kp=2, Ki=0.7015, Kd=0.11, Setpoint=5 km/h

These three methods that were used to obtain the low-level PID control algorithm were tested on OSKAR.

When the controller gains were used in case of model is obtained by using the mathematical model and PID was designed by root-locus method, it was observed that

extreme overshoots occurred and it could not be able to settle in steady-state with a constant velocity. When the gains obtained by Cohen-Coon method was used, nearly same negative situation occurred. The vehicle could not settle the velocity, oscillated between the maximum and minimum voltage values that the restrictions (2.24V-4.8V, forward voltage for the motor driver) allowed.

As the gains were used when using mathematical model of the DC motor and designing PID by MATLAB's PID Tuner, more optimized values were observed. For smaller velocity set points, higher overshoot values were obtained and settling time was generally more than the expected.

Table 5. PID controller gains for different modeling and control methods

Motor Model Obtaining Method	**PID Controller Design Method**	**K_i**	**K_p**	**K_d**
Conventional Model	Root-Locus	0.0253	2.5275	0.1596
Cohen-Coon	Cohen-Coon	0.7015	1.389	0.11
Conventional Model	PID Tuner Tool	0.036965	0.12918	0.0047062

After obtaining the controller gain values as shown on the table above, indoor tests were taken. Procedure of indoor test was to prepare a contrivance in order not to let any contact between the wheels of OSKAR and ground. Thus, there was no load (driver and assistant) on the car which causes zero load torque. With this setup, step velocity set points were applied to the vehicle by assigning desired constant values to the corresponding constants in low-level software code. Then, resulting responses of the vehicle were logged and plotted on MATLAB.

After the indoor tests for all three methods, last method path was decided to use in PID controller design. It was reached a consensus that since the calculations of model and PID controller were obtained for the ideal conditions, resulting gain values did not completely hold in practice. Thus, the gain values were slightly changed until the optimum results are observed. Final gain values obtained are given below:

$$K_p = 0.1 \quad , \quad K_i = 0.2 \quad , \quad K_d = 0.0047$$

6.1.2. Low Level Implementation and Tests

Using final PID controller gains, a PID controller code was written. As mentioned before, low-level controller takes the desired velocity value from high-level controller and current velocity from the encoder system. With these velocity values and the controller gains, PID controller has built in STM32F4-DISCOVERY microcontroller board, on IAR Embedded Workbench environment.

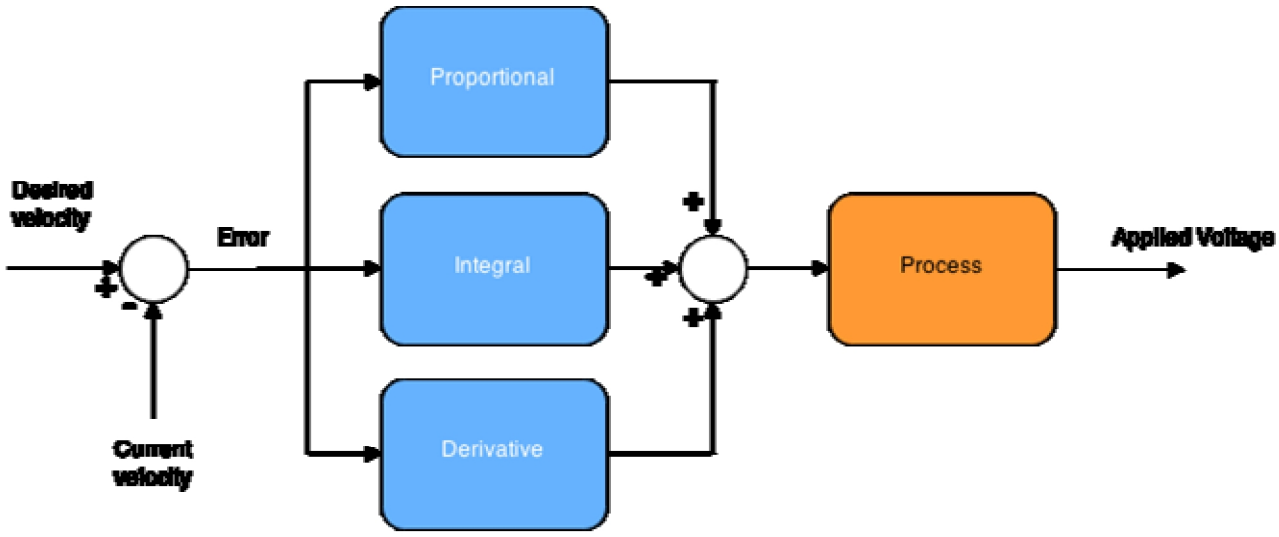

Fig 13. Block diagram of the PID controller that is used in low-level controller

Contents of the PID controller blocks which is represented in Fig 13 are shown on Table 6. Since the PID controller system was desired to implement on microcontroller board, continuous integral and derivative terms were needed to be discretized. Discrete values for PID controller terms are also shown in Table 6.

Table 6. Formulas for the elements of PID controller

		Formula	**Discretized Formula**
Proportional	P	$K_p e(t)$	$K_p e(n)$
Integral	I	$K_i \int_0^t e(\tau)d\tau$	$K_i T_s \sum_{i=0}^{n} e(i)$
Derivative	D	$K_d \frac{de(t)}{dt}$	$K_d \frac{e(n)-e(n-1)}{T_s}$

PID controller was decided to operate at the frequency of 10 Hertz. This feature was set by using Real-Time Operating System's task assignment attributions. Pseudo code of the PID controller is shown in Fig 14.

```
void function PID
    current_velocity = read_Velocity_From_Encdoer()
    error = velocity_setpoint - current_velocity

    P = prop_gain * error
    accumError += error * sampling_time
    I = integral_gain * accumError;
    differential_Error = (error - last_Error) / sampling_time
    last_Error = error
    D = diff_gain * differential_Error

return P + I + D;
```

Fig 14. Pseudo code of PID controller

Also, since the algorithm may give out extremely large numbers on both positive and negative sides, output of the algorithm had to be restricted in respect to the motor driver's operating voltage area. Motor driver's forward operating area is between 2.24 V and 4.8 V. Pseudo code of this arrangement is shown below.

```
if calculated_voltage > 4.80
      calculated_voltage = 4.80
else if calculated_voltage < 2.24
      calculated_voltage = 2.24

applied_voltage = (calculated_voltage *(2/3.33)*(4095/3));
```

Fig 15. Pseudo code of output voltage restriction for PID controller

It should be noted that STM32F4-DISCOVERY board can give maximum 3V output. So, as mentioned before, a 0-3V to 0-5V amplifier had designed to operate between the low-level controller and the motor driver. On the other hand, required voltage was calculated as if it reaches directly to the motor driver. Last line of the code in Fig 15 represents this correction when the amplifier circuitry is operating between low-level controller and motor driver.

After modeling, PID designing, coding and indoor test process were completed, outdoor tests began. For outdoor testing, a driver and an assistant rode the vehicle. For different constant velocity set points; response characteristics and settling velocity values were observed and the velocity data which were read from the encoder were logged. Results obtained from outdoor tests are shown below.

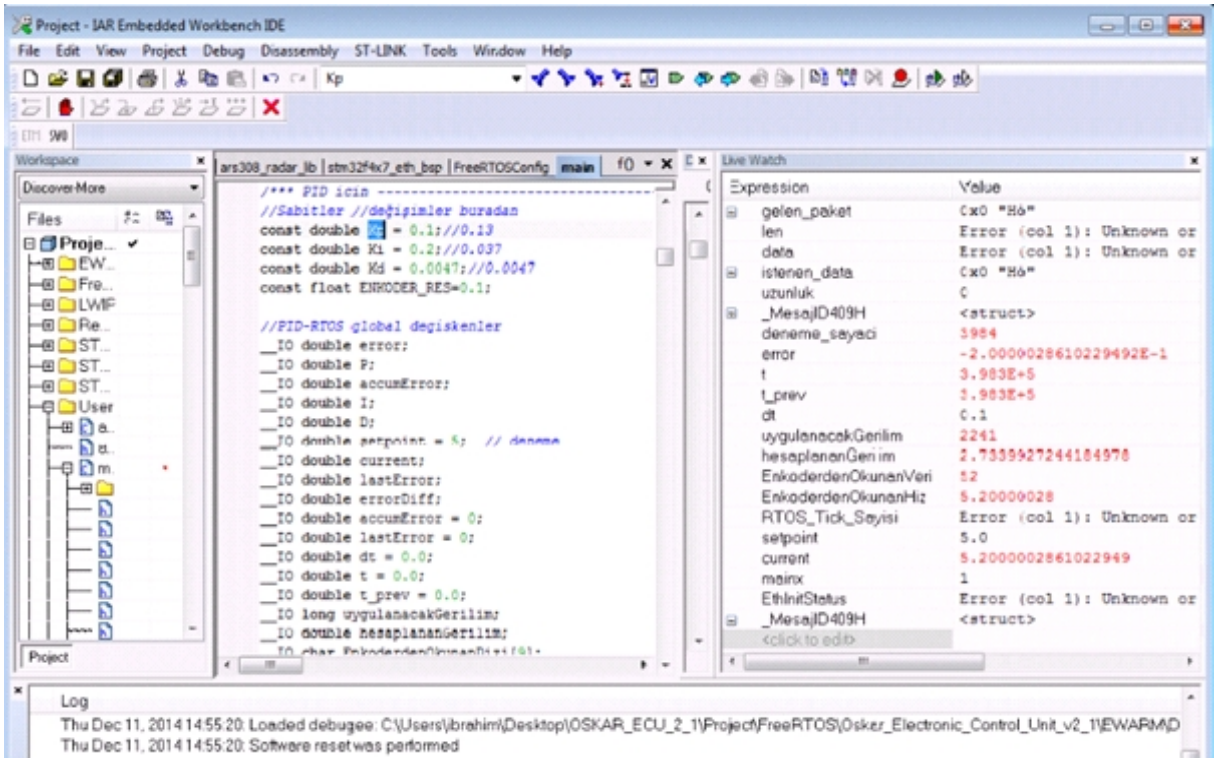

Fig 16. A screenshot of IAR Workbench during an outdoor PID test. Velocity set point is 5 km/h

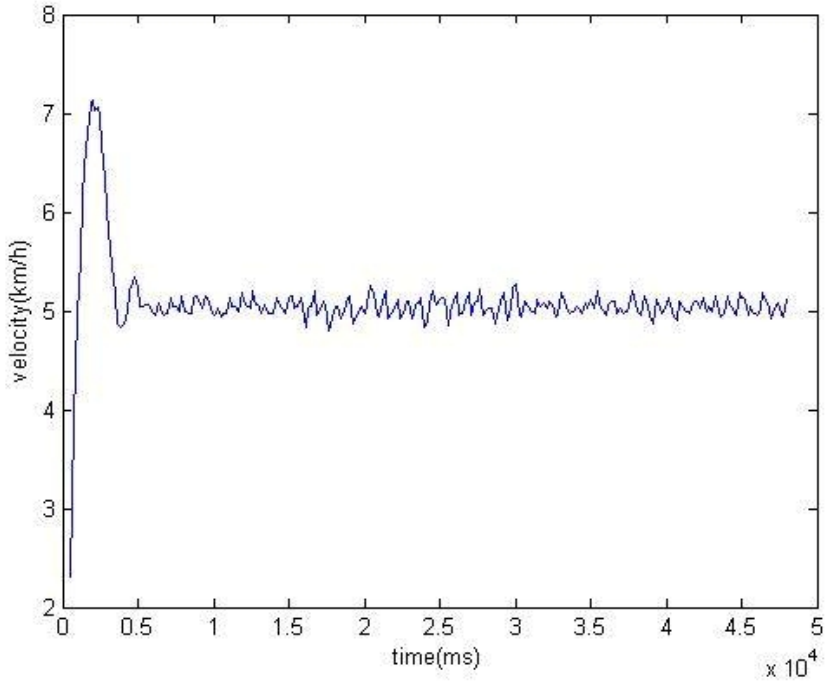

Fig 17. PID outdoor test result when velocity set point is 5 km/h

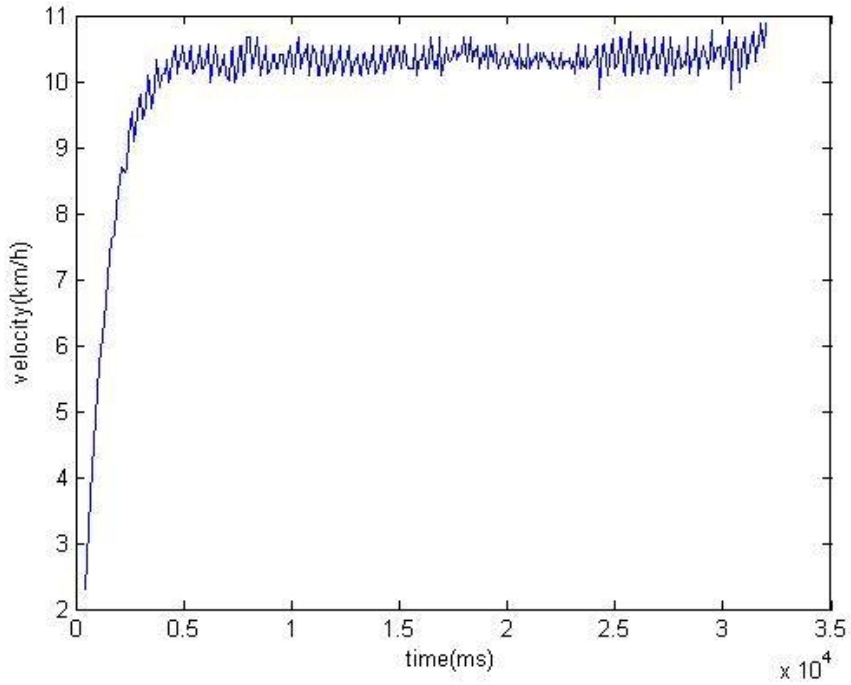

Fig 18. PID outdoor test result when velocity set point is 10 km/h

6.1.2.1. Hardware Implementation

6.1.2.1.1. Information on Motor Driver and External Circuitry of the Electric Vehicle

The OSKAR electric vehicle uses a PG S-Drive D51448.01 motor driver (seen in Figure 19) for actuation. For the implementation of the low level design, the electric vehicle OSKAR motor driver and motor is subjected to tests in order to create an engine control unit (ECU) for the motor. During these tests the voltage and the current characteristics of both motor and motor driver has been analyzed and observed by using MATLAB. It is seen that the idle voltage for the motor to be steady was 2.24Volts. Figure 20 shows that motor controller drives motor forwards and increases its forward speed from 2.24V to 5V, whereas the controller drives motor backwards and increases its backward speed

from 2.24V to 0V. The shaded area in the Figure 20 is the deadzone where motor does not actuate if the voltage input to the motor controller is between 1.7 and 2.24 Volts.

Fig 19. PG S-Drive D51448.01 motor driver

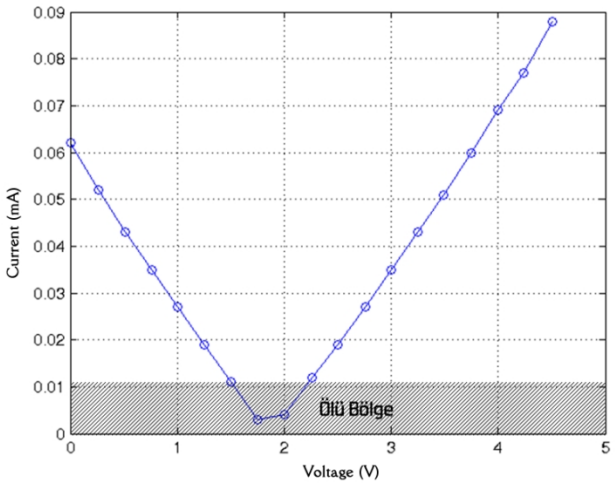

Fig 20. Test Results for the Motor Driver

By observing the OSKAR circuitry the following informations are gathered: Motor driver input line was connected to a commutator that works as an electronic shifter by adjusting the input current. The motor driver is supplied with 2x12Ampere batteries through the motor driver B+ and B- lines. There is also an electronic ignition switch connected to B1 and B3 lines of the motor driver. The ignition switch is connected in series with the battery indicator. The overall OSKAR circuitry before the design implementation is seen in Figure 21. The information gathered is used for proceeding to design the engine control unit.

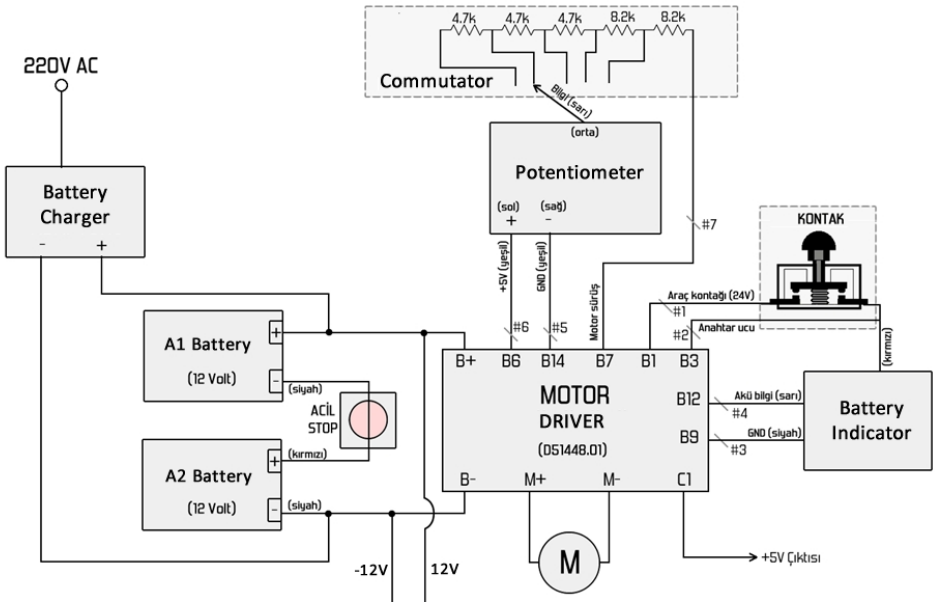

Fig 21. OSKAR Circuitry before ECU Design

6.1.2.1.2. Engine Control Unit Hardware Design and Implementation

This section introduces and explains the engine control unit design shown in Fig 22 in details.

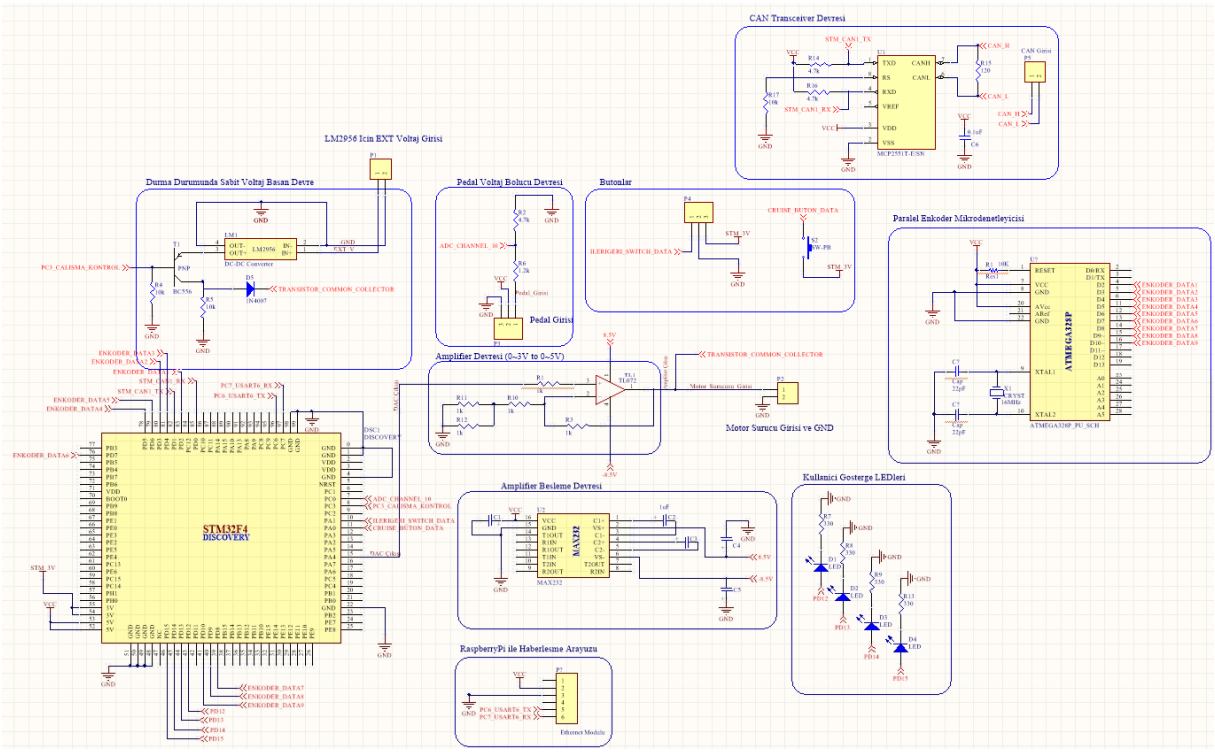

Fig 22. OSKAR ECU Hardware Design

6.1.2.1.2.1. Amplifier Unit

As for the engine control unit, STM32F4-DISCOVERY Board is selected. ADC and DAC units of the STM32 board is planned to act as input and output for the motor controller. Datasheet and Reference Manual of STM32 board states that the board working voltage is maximum 3 Volts. Since the motor driver needs maximum of 5Volts, our team has designed an amplifier that takes the input voltage in the range of 0~3V and converts it to the range of 0~5V. The simulations of the amplifier is done in the Proteus software as seen in Figure 23. Moreover, the hardware circuit schematics and PCB layout design is done by using the Altium Designer software.

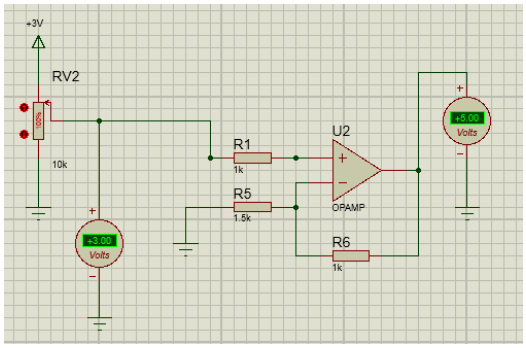

Fig 23. Engine Control Unit Amplifier Simulation in Proteus

It is selected that Texas Instruments TL072 to be used for the amplification purpose. For the bias circuitry of the TL072 device, we have used a MAXIM Semiconductor MAX232 device to provide negative voltage alongside the positive voltage. MAX232 device provides 8.5V and -8.5V which is sufficient to be used for the bias circuit of the TL072 amplifier. The output of the DAC port of STM32F4 microcontroller is connected to the input of the amplifier so that the data coming from DAC is amplified to have a voltage between range 0V~5V. Figure 24 shows the circuit of the final design of amplifier circuitry.

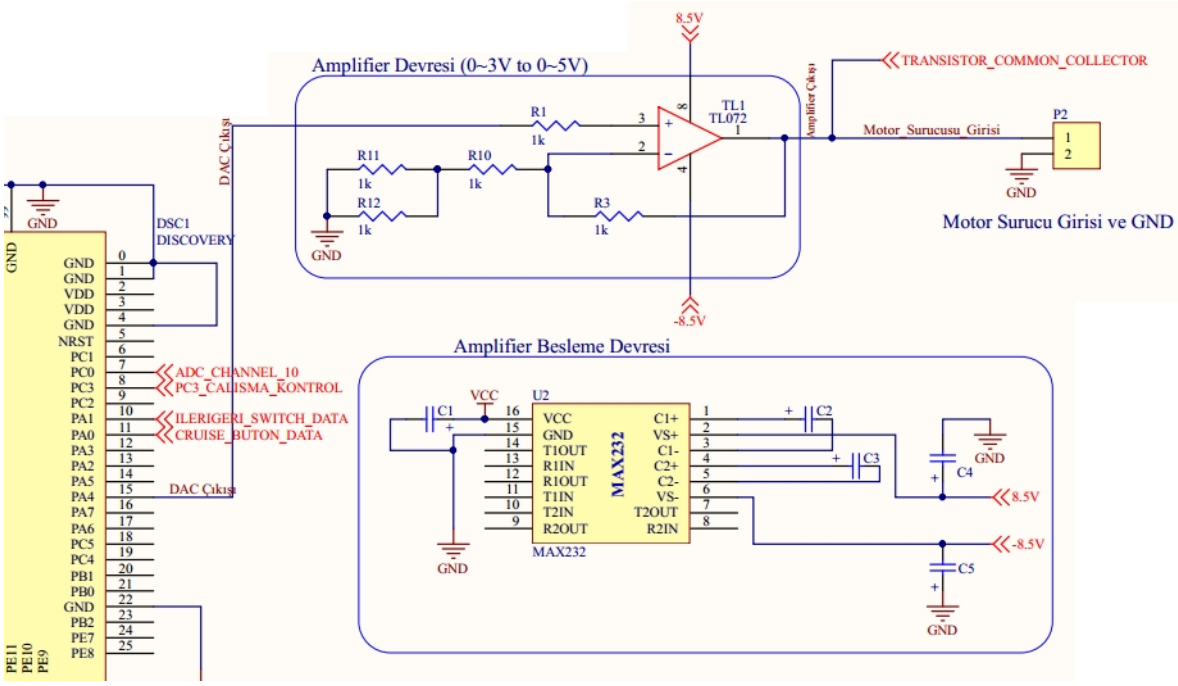

Fig 24. Amplifier Circuitry in Altium Designer Software

6.1.2.1.2.2. External Supply Unit

It is known from the section *"6.1.2.1.1. Information on Motor Driver and External Circuitry of the Electric Vehicle"* that the motor is steady when a voltage of 2.24V is applied to the motor driver input. Since electric vehicle needs to be steady at all cost, a security measure must be taken to prevent power run out on the motor driver input terminal. For that purpose, a constant 2.24V must be supplied to the motor driver if the STM32 microcontroller stops working. If the microcontroller is working, the DAC output must be significant. To supply the 2.24V constant voltage, LM2956 device has been selected. A BC556 PNP transistor is used in the switching mode in order to control whether the STM32 device is on or off. The PC3 port of the STM32 device is allocated for this purpose which supplies 3V when device is ON and 0V when the device is OFF.

The transistor seemed not to work correctly at this stage since a backward current is coming from DAC output to transistor common collector. In order to prevent this backward current, we have used a 1N4007 diode. Figure 25 shows the final design of the external supply unit.

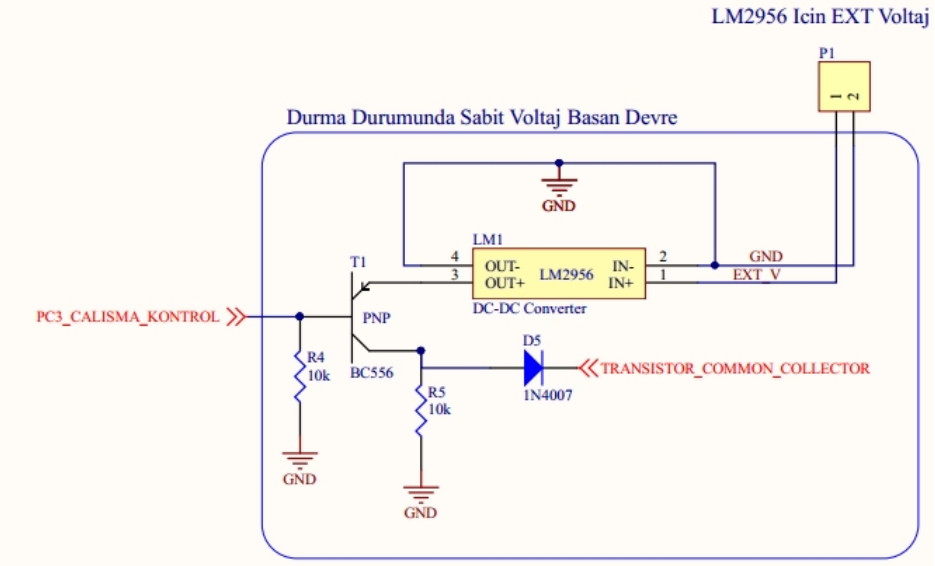

Fig 25. Final Design of the External Supply Unit in Altium Designer Software

6.1.2.1.2.3. Throttle Circuitry

In order to implement the conventional cruise controller, the throttle is essential to get the vehicle moving. An electronic throttle that works as a potentiometer is installed to the vehicle for that purpose. The throttle input is planned to be read from the ADC port of the STM32 device. The electrical tests of the throttle showed that it operates between 0.8V ~ 4.2V depending on the potentiometer level. By voltage division, this voltage is being reduced to 0.57 ~3V range at the ADC input. The final design of the throttle circuitry is shown in Fig 26.

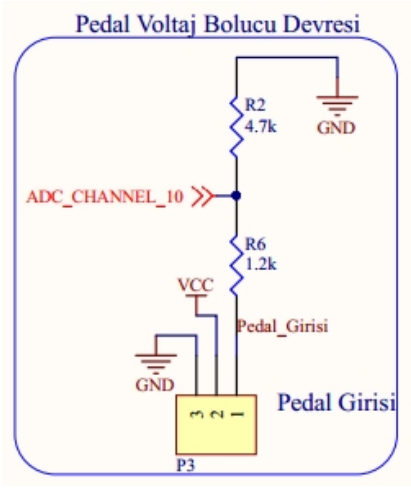

Fig 26. Voltage Divider Circuit for the ADC Input

6.1.2.1.2.4. Indicator Leds and Information Switches

As forward/backward shifting or cruise state input a button and a switch have been added to the system which are shown in Fig 27. To show the whether the forward or backward shifting is turned on, whether the cruise state is on or of, or if there is an error in the STM32 device or not, indicator Leds have also been to the system shown in Fig 28. These peripherals are controlled with the STM32 device software.

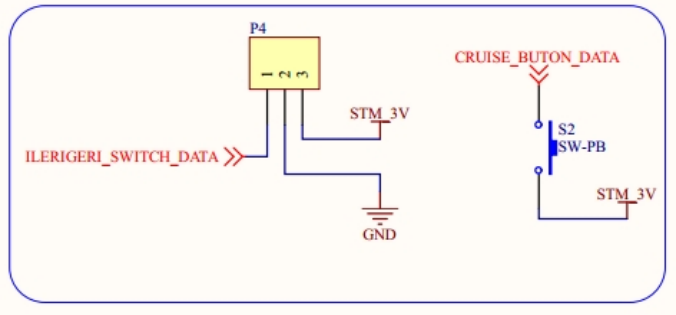

Fig 27. Information Switch and Button for ECU

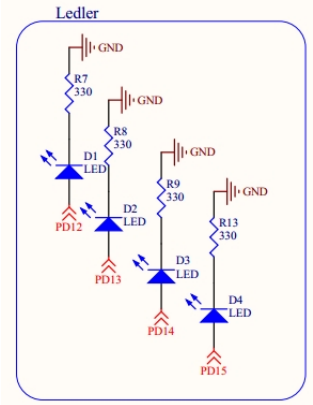

Fig 28. Indicator Leds for ECU

6.1.2.1.2.5. CAN Transceiver Circuitry

To communicate with the RADAR in order to get absolute and relative information on the vehicles, the CAN-bus node must be read. In order to read the CAN-bus node, the ECU is included with a CAN-bus transceiver that consists of a MCP2551 high-speed CAN-bus transceiver device. A 120 ohm termination resistor is also adeed between the node terminals. Figure 29 shows the CAN transceiver circuitry in detail.

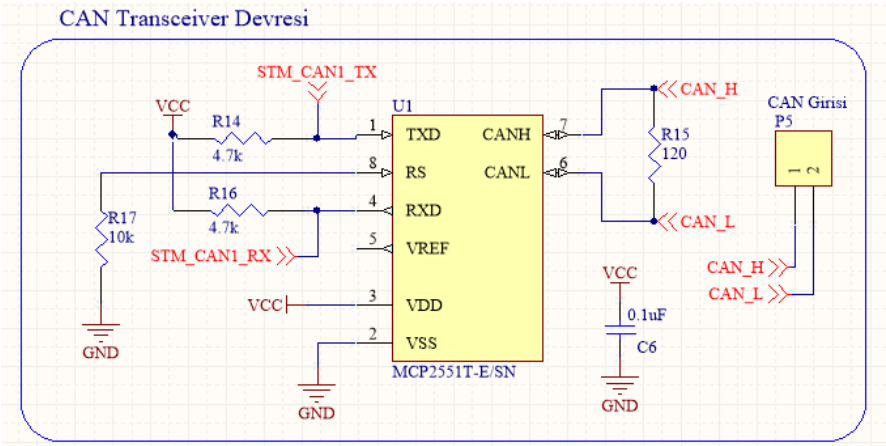

Fig 29. CAN Transceiver Circuitry

6.1.2.1.2.6. Parallel Encoder Microcontroller

Since STM32 device is responsible for too many periodical tasks such as PID control, RADAR CAN-bus communication, high-level USART/Ethernet communication, a parallel encoder microcontroller (ATMEGA 328P) is used as the data processor for the Encoder unit. This unit provides 9-bit velocity information with a resolution of 0.1 km/h to the STM32 device by reading a Hall Effect Sensor that is connected to it. The Figure 30 shows the parallel encoder microcontroller circuitry.

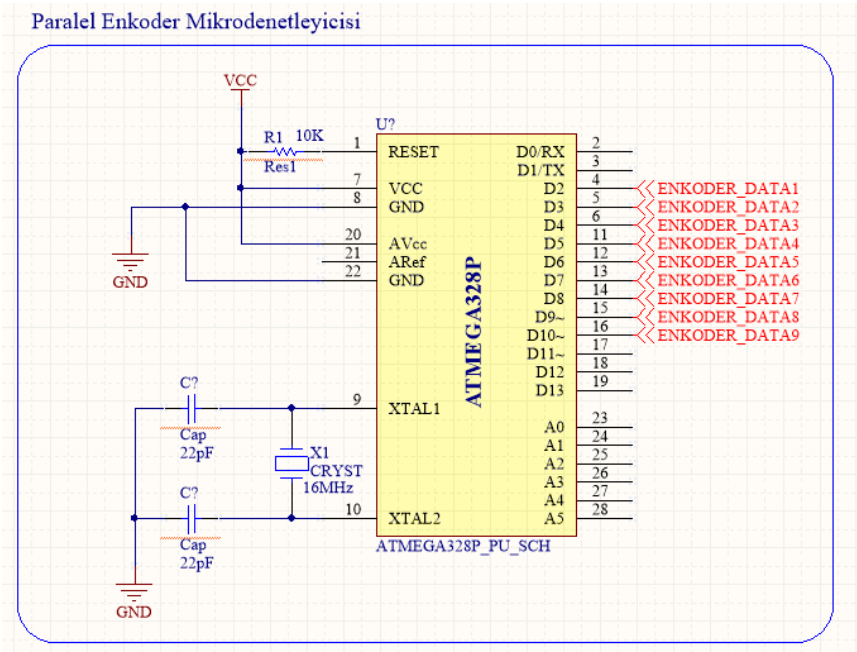

Fig 30. Parallel Encoder Microcontroller

6.1.2.2. Software Implementation

6.1.2.2.1. Low-Level Embedded Development Environment

6.1.2.2.1.1. STM32F4-DISCOVERY Kit

For the low level unit implementation, STM32F4 DISCOVERY kit (seen in Fig 31) is used. This kit is an embedded developement board that is based on the STM32F407VGT6 microcontroller. It includes an ST-LINK/V2 embedded debug tool, two ST MEMS, digital accelerometer and digital microphone, one audio DAC with integrated class D speaker driver, LEDs and push buttons and an USB OTG micro-AB connector. Since the kit offers a 32-bit ARM processor with 1MB Flash and 192KB RAM, it is considered to be useful for many applications regarding low level motor control and data communication. It provides the peripherals of ADC, DAC, DMA, ETH, USART, CAN, SPI etc. which suits the design requirements of the project.

Moreover, it is an ARM device with CMSIS support, which makes coding flexible and compatible with all other ARM microcontroller devices.

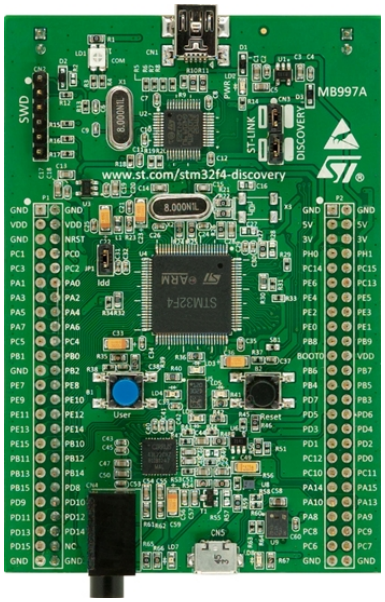

Fig 31. STM32F4-DISCOVERY Kit

Key features of the device are given below: [43]

- STM32F407VGT6 microcontroller featuring 32-bit ARM Cortex-M4F core, 1 MB Flash, 192 KB RAM in an LQFP100 package
- On-board ST-LINK/V2 with selection mode switch to use the kit as a standalone ST-LINK/V2 (with SWD connector for programming and debugging)
- Board power supply: through USB bus or from an external 5 V supply voltage
- External application power supply: 3 V and 5 V
- LIS302DL or LIS3DSH ST MEMS 3-axis accelerometer
- MP45DT02, ST MEMS audio sensor, omni-directional digital microphone
- CS43L22, audio DAC with integrated class D speaker driver
- Eight LEDs:
  - LD1 (red/green) for USB communication

- LD2 (red) for 3.3 V power on
- Four user LEDs, LD3 (orange), LD4 (green), LD5 (red) and LD6 (blue)
- 2 USB OTG LEDs LD7 (green) VBus and LD8 (red) over-current
- Two push buttons (user and reset)
- USB OTG FS with micro-AB connector
- Extension header for all LQFP100 I/Os for quick connection to prototyping board and easy probing

6.1.2.2.1.2. IAR Systems Embedded Workbench for ARM

IAR Embedded Workbench for ARM is an embedded software development environment for ARM-based processors and kits. It uses the programming languages of C/C++ and Assembly languages. The environment supports many kits and processors that are ARM products, that includes the STM32F4-DISCOVERY kit. With IAR EWARM, one can set a workspace for the embedded project and configure all the linking and debugging options. It is also useful when constructing a project with various libraries, since its workspace has library organization capabilities. An instance of the software IAR Embedded Workbench is shown in Fig 32.

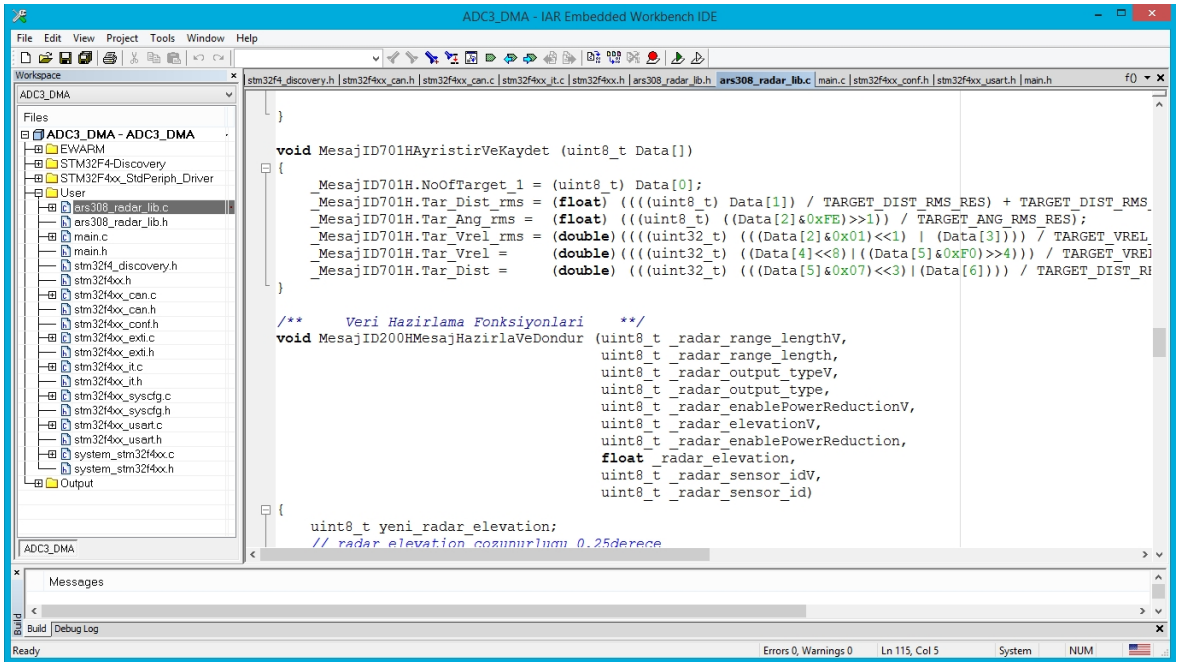

Fig 32. IAR Embedded Workbench IDE

In the EWARM project workspace, the project options are configured to meet the requirements for the whole project. Full D-Library configuration for C/C++ environment has been set for the ability to use all the C libraries. Moreover, the ST-Link, which is the debugger and programmer for many STM devices, has also been configured to work on the SWD (Serial-Wire Debug) port. The device core clock has been set to 168 MHz, which is the clock of the device STM32F407VG6. Moreover, the standart 1Mb Flash memory configuration must be checked before building projects.

EWARM offers a capability to live watch variables which is used frequently in the project software development. To watch variables, they are stored in RAM memory by using CMSIS keyword __IO. EWARM also offers a moduler structure of files organized in the Workspace window. There are 4 basic components in an EWARM workspace (for a STM32F4-DISCOVERY project) which are: (1)- The standart CMSIS library with MCU initiation file, (2)- STM32F4xx Microcontroller Library, (3)-

STM32F4 DISCOVERY board library, (4)- EWARM functions. By using the "User" folder, the standart and created libraries has been added to the project.

6.1.2.2.2. Real Time Operating Systems and FreeRTOS

Most operating systems appear to allow multiple programs to execute at the same time. This is called multi-tasking. In reality, each processor core can only be running a single thread of execution at any given point in time. A part of the operating system called the scheduler is responsible for deciding which program to run when, and provides the illusion of simultaneous execution by rapidly switching between each program [44].

The type of an operating system is defined by how the scheduler decides which program to run when. For example, the scheduler used in a multi user operating system (such as Unix) will ensure each user gets a fair amount of the processing time. As another example, the scheduler in a desk top operating system (such as Windows) will try and ensure the computer remains responsive to its user [44].

The scheduler in a Real Time Operating System (RTOS) is designed to provide a predictable (normally described asdeterministic) execution pattern. This is particularly of interest to embedded systems as embedded systems often have real time requirements. A real time requirements is one that specifies that the embedded system must respond to a certain event within a strictly defined time (the deadline). A guarantee to meet real time requirements can only be made if the behaviour of the operating system's scheduler can be predicted (and is therefore deterministic) [44].

Traditional real time schedulers, such as the scheduler used in FreeRTOS, achieve determinism by allowing the user to assign a priority to each thread of execution. The

scheduler then uses the priority to know which thread of execution to run next. In FreeRTOS, a thread of execution is called a task [44].

6.1.2.2.3. Implementation and Algorithms

6.1.2.2.3.1. Motor Drive via Throttle Input

In the conventional cruise control implementation, system algorithm is designed to work within RTOS. The information coming from ADC is resolved and the calculated output is given from the DAC unit that works between 0V~3V. System also has another inputs that are forward/backward shift switch, cruise control enabling button which are also considered within the algorithm to manipulate the data that is send from the DAC unit. The figure 33 shows the implementation of the algorithm.

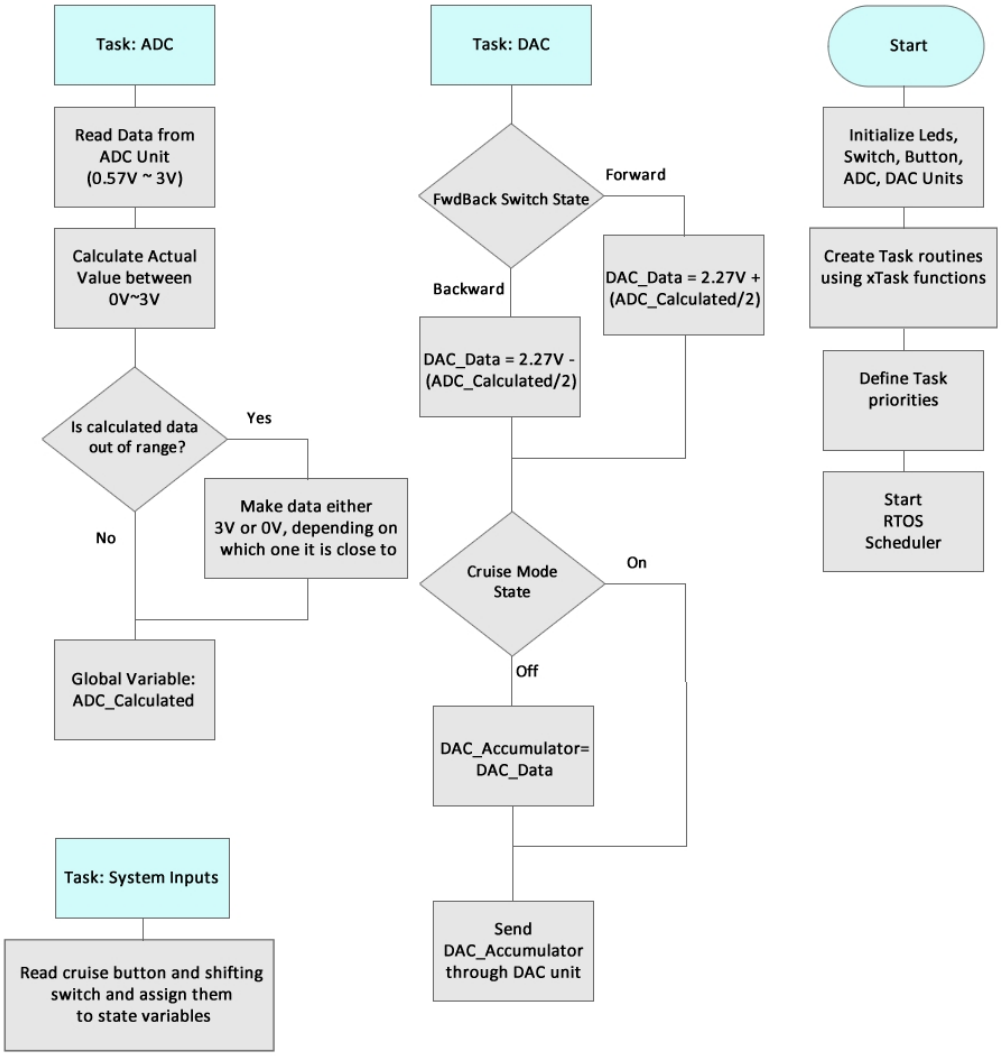

Fig 33. Motor Drive via Throttle Input RTOS Algorithm Flowchart

6.1.2.2.3.2. Controller Area Network Bus (CAN-BUS) Implementation

CAN is an International Standardization Organization (ISO) defined serial communications bus originally developed for the automotive industry to replace the complex wiring harness with a two-wire bus. The specification calls for high immunity to electrical interference and the ability to self-diagnose and repair data errors. These features have led to CAN's popularity in a variety of industries including building automation, medical, and manufacturing [45].

The CAN communications protocol, ISO-11898: 2003, describes how information is passed between devices on a network and conforms to the Open Systems Interconnection (OSI) model that is defined in terms of layers. Actual communication between devices connected by the physical medium is defined by the physical layer of the mode [45]. The standart and extended ID frames for the CAN-BUS communication is given in Fig 34 and Fig 35, respectively.

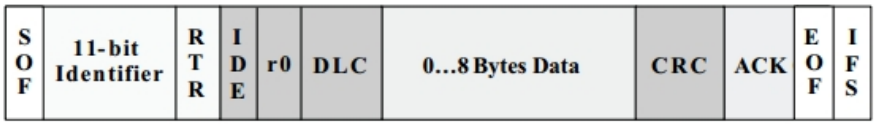

Fig 34. Standart ID CAN Frame

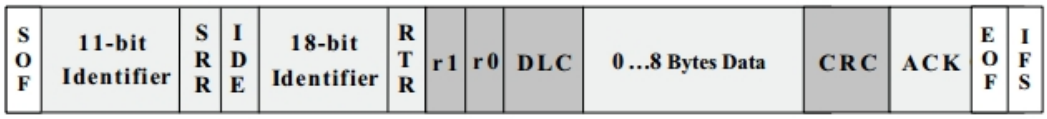

Fig 35. Extended ID CAN Frame

In STM32F4 software, the Identifier, RTR, IDE, DLC, and DATA fields have to be configured in order to send a CAN frame.

• Identifier-The Standard CAN 11-bit identifier establishes the priority of the message. The lower the binary value, the higher its priority. If 29-bit identifier is used IDE field must be set to 0x1 whereas, if 11-bit identifier is used IDE field must be set to 0x0.

• RTR–The single remote transmission request (RTR) bit is dominant when information is required from another node. All nodes receive the request, but the identifier determines the specified node. The responding data is also received by all nodes and used by any node interested. In this way, all data being used in a system is uniform.

• IDE–A dominant single identifier extension (IDE) bit means that a standard CAN identifier with no extension is being transmitted.

• DLC–The 4-bit data length code (DLC) contains the number of bytes of data being transmitted.

• Data–Up to 64 bits of application data may be transmitted.

For the CAN-BUS implementation the Standart Peripherial Library (stm32f4xx_can.h) has been used to develop functions for CAN-BUS communication. In the data sending, the clock configuration has become an issue, since it has to be 500KBps for the ARS308 radar. CAN Library presents time quanta registers to configure CAN-BUS baudrate. The further research lead to the following equation:

$$CANBaudrate = \frac{ClockFrequency}{(SJW + BS1 + BS2) * \text{Prescaler}}$$

In the equation above, SJW, BS1, and BS2 indicates time quanta registers. According to this equation, a design has been done where SJW is selected as 1, BS1 is selected as 6, and BS2 is selected as 7. Since the CAN1 clock frequency of the STM32F4 is 168MHz divided by internal prescaler (4), which lead to the result 42MHz, we have selected the Prescaler to be 6 in order to achive 500 KBps baudrate. However, later work showed that the system clock definitions of the peripheral library was flawed. The adjustments in the peripheral library configurations have been done in order to correctly adjust the CAN-bus baudrate. Since the external clock frequency in the DISCOVERY kit is 8MHZ, the corresponding PLL register values have been investigated and set correctly. In addition, the library core clock frequency configuration has been adjusted to be 168MHz instead of 144MHz. With the corrected configurations, CAN-bus was able to send and receive data from the I7565 USB to CAN Analyzer Device via 500KBps

without problems. The flowcharts in Fig 36 shows the data transmission and reception algorithms developed in STM32F4 DISCOVERY kit.

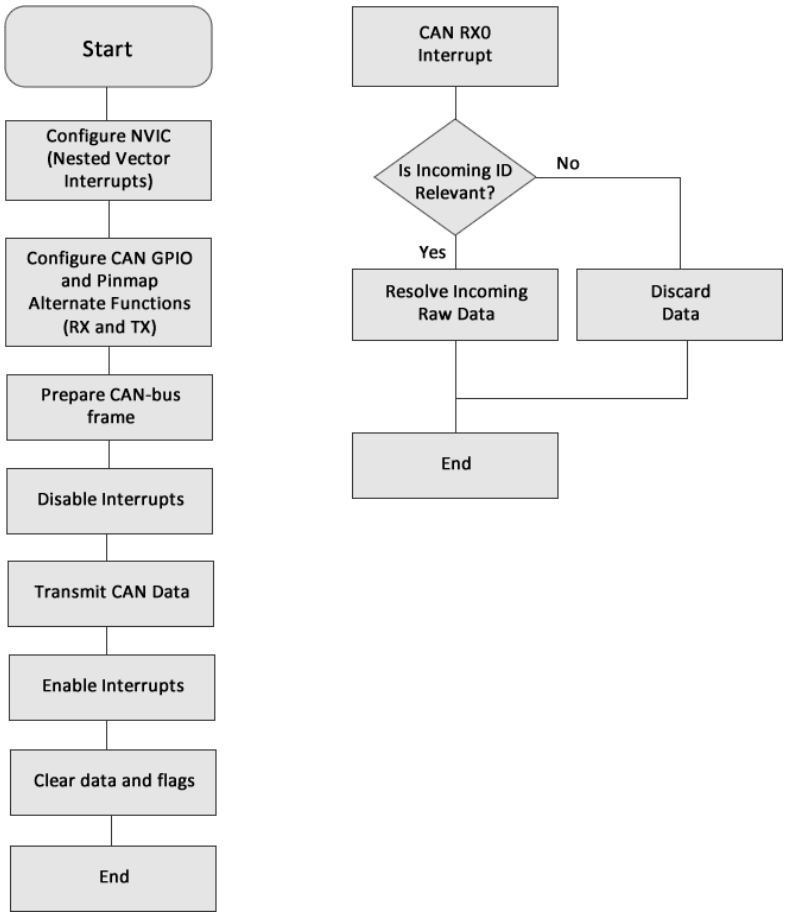

Fig 36. Developed CAN Transmission and Reception Algorithm

6.1.2.2.3.3. ARS-308 High-Range RADAR Driver Implementation for STM32F4 Discovery

6.1.2.2.3.3.1. RADAR Driver Implementation

The ARS 308 is a Radar Sensor System developed by Continental for the Automotive Industry to realize advanced driver assistance functions. The usual interface of this system is mainly based on deceleration requests to the vehicle network [46].

The software of the sensor was adapted to use it also for general purposes. With the simple software interface it is possible to connect the sensor to a CAN network and to provide radar based environmental information to one or several evaluation units. The sensor can also be configured via CAN [46].

The ARS sensor uses radar radiation to analyze its surroundings. The reflected signals are processed and after multiple steps they are available in form of targets and objects. One target consists of multiple reflections which have a similar position and movement and therefore can be combined. The information about a target like size, relative velocity and position is then transmitted on CAN1. The position is calculated in an angular coordinate system, i.e. distance and angle relative to the sensor. The targets are newly evaluated every cycle. In contrast to this, objects have a history and consist of tracked targets. The position of the object is calculated relative to an assumed vehicle course. The course is determined by using the speed and yaw rate information. Figure 37 shows the illustration and the conventional directions of the RADAR and the Figure 38 shows the ARS-308 Long Range RADAR by Continental Corp.

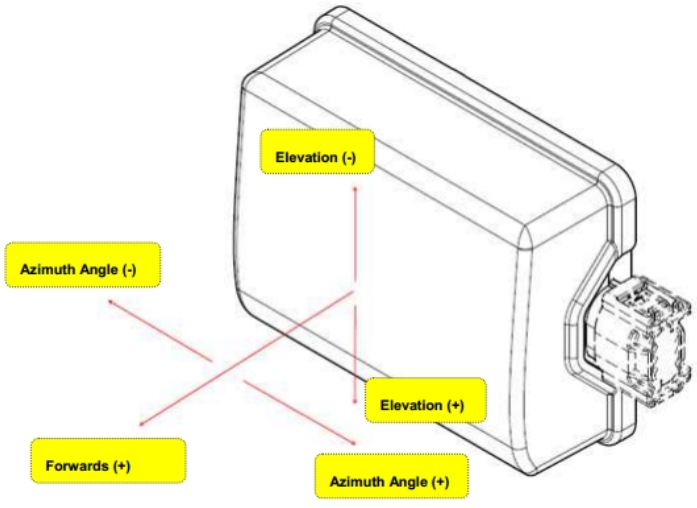

Fig 37. Illustration and Sign conventions of the ARS308 RADAR

Fig 38. The Continental ARS-308 RADAR

In order to understand the data coming from the RADAR, first we have to resolve the data being received from CAN-bus. According to the datasheet or ARS308 RADAR, the data received is in form of Motorola Byte order and is bigendian. The frames for the bytes are resolved by using bitwise operations. If the data has resolution and offset, those factors have to be considered. An illustrative example of a frame resolved will be useful to tell the work done at this point:

In ARS308 RADAR, message ID 0x200 has been reserved for radar configurations message structure. This message has the form seen in Fig 39.

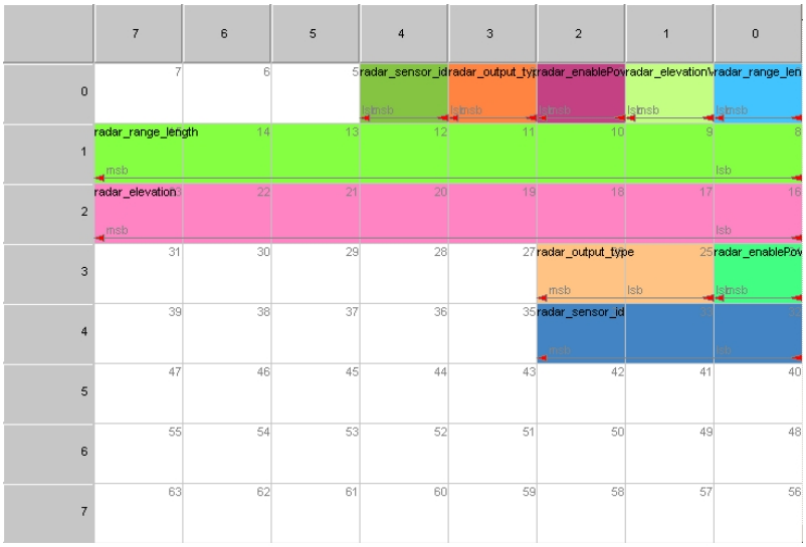

Fig 39. Radar Configuration Structure of ARS308

The Table 7 involves the pseudocode that is written to resolve the data for this frame:

Table 7. PseudoCode for the 0x200 Message Resolution

```
void MesajID201HAyristirVeKaydet (uint8_t Data[])
{
  _MesajID201H.CurrentRadarPower =     (uint8_t)((Data[0] & 0x02) >> 1);
  _MesajID201H.SensTempErr =               (uint8_t)((Data[0] & 0x08) >> 3);
```

```
    _MesajID201H.SensDef =                  (uint8_t)((Data[0] & 0x20) >> 5);
    _MesajID201H.SupVolt_L =                (uint8_t)((Data[0] & 0x40) >> 6);
    _MesajID201H.RadarPowerReduction =  (uint8_t)((Data[0] & 0x80) >> 7);
    _MesajID201H.SensorID =                 (uint8_t)(Data[1] & 0x0F);
    _MesajID201H.RxInvalid =                (uint8_t)(Data[2] & 0x03);
    _MesajID201H.NVMreadStatus =            (uint8_t)((Data[2] & 0x0C) >> 2);
    _MesajID201H.NVMwriteStatus =           (uint8_t)((Data[2] & 0x10) >> 4);
    _MesajID201H.currElevationCal =  (float)(((uint8_t) Data[3]) * CURRELEVATIONCAL_RES);
    _MesajID201H.currRangeLengthCal =     (uint8_t) Data[4];
}
```

The ARS308 RADAR driver has been written in this fashion where the message ID's are represented as structs having uint8_t, uint16_t, uint32_t, float, and double types depending on the data structure, offset and resolution. For the Message ID 0x409, an Object struct has been constructed since the radar returns distance, acceleration, and velocity informations for 64 objects that has been detected. For dynamic selection of the correct object, the message data having 0x409 is divided according to object ID's and stored in the created Object struct. Moreover, the radar driver have modular structure where the resolution and offset information are stored as preprocessor definitions shown in Table 8.

Table 8. Preprocessor definitions for radar driver

```
#define MESAJ408H_STATEMSG_TYPE_WARN_INFO (uint8_t)0x00
#define MESAJ408H_STATEMSG_TYPE_REGION_INFO (uint8_t)0x01


#define OBJECT_COUNT 64
```

```
/** Transmission Resolutions  **/
#define RADARELEVATION_RES            ((float)0.25)
#define RADARDEVICESPEED_RES          ((float)0.02)
#define RADARDEVICEYAWRATE_RES        ((float)0.01)
#define CFGOBJMINWIDTH_RES            ((float)0.1)
#define CFGOBJMINDETECTIONTIME_RES    ((float)0.1)
#define CFGREGIONPOINT1X_RES          ((float)0.2)
#define CFGREGIONPOINT1Y_RES          ((float)0.2)
#define CFGREGIONPOINT2X_RES          ((float)0.2)
#define CFGREGIONPOINT2Y_RES          ((float)0.2)

/** Reception Resolutions **/
#define CURRELEVATIONCAL_RES          ((float)0.25)
#define TARGET_DIST_RMS_RES           ((float)0.1)
#define TARGET_ANG_RMS_RES            ((float)0.1)
#define TARGET_VREL_RMS_RES           ((float)0.03)
#define TARGET_VREL_RES               ((float)0.03)
#define TARGET_DIST_RES               ((float)0.1)
#define LATDISTTOBORDERLEFT_RES       ((float)0.1)
#define LATDISTTOBORDERRIGHT_RES      ((float)0.1)
#define OBJ_LONGDISPL_RES             ((float)0.1)
#define OBJ_VRELLONG_RES              ((float)0.0625)
#define OBJ_ACCELLONG_RES             ((float)0.0625)
#define OBJ_LATDISPL_RES              ((float)0.1)
#define OBJ_RCSVALUE_RES              ((float)0.5)
#define OBJ_LATSPEED_RES              ((float)0.25)

/** Offsets **/
```

```
#define RADARDEVICEYAWRATE_OFFSET       ((float)-327.68)
#define CFGREGIONPOINT1Y_OFFSET         ((float)-50.0)
#define CFGREGIONPOINT2Y_OFFSET         ((float)-50.0)
#define TARGET_DIST_RMS_OFFSET          ((float)-10.0)
#define TARGET_VREL_RMS_OFFSET          ((float)-5.0)
#define TARGET_VREL_OFFSET              ((float)-25.0)
#define OBJ_VRELLONG_OFFSET             ((float)-128.0)
#define OBJ_ACCELLONG_OFFSET            ((float)-16.0)
#define OBJ_LATDISPL_OFFSET             ((float)-52.0)
#define OBJ_RCSVALUE_OFFSET             ((float)-64.0)
#define OBJ_LATSPEED_OFFSET             ((float)-32.0)
```

The Figure 40 shows the diagram of the structs created in the entire radar driver that is written for STM32F4 DISCOVERY kit.

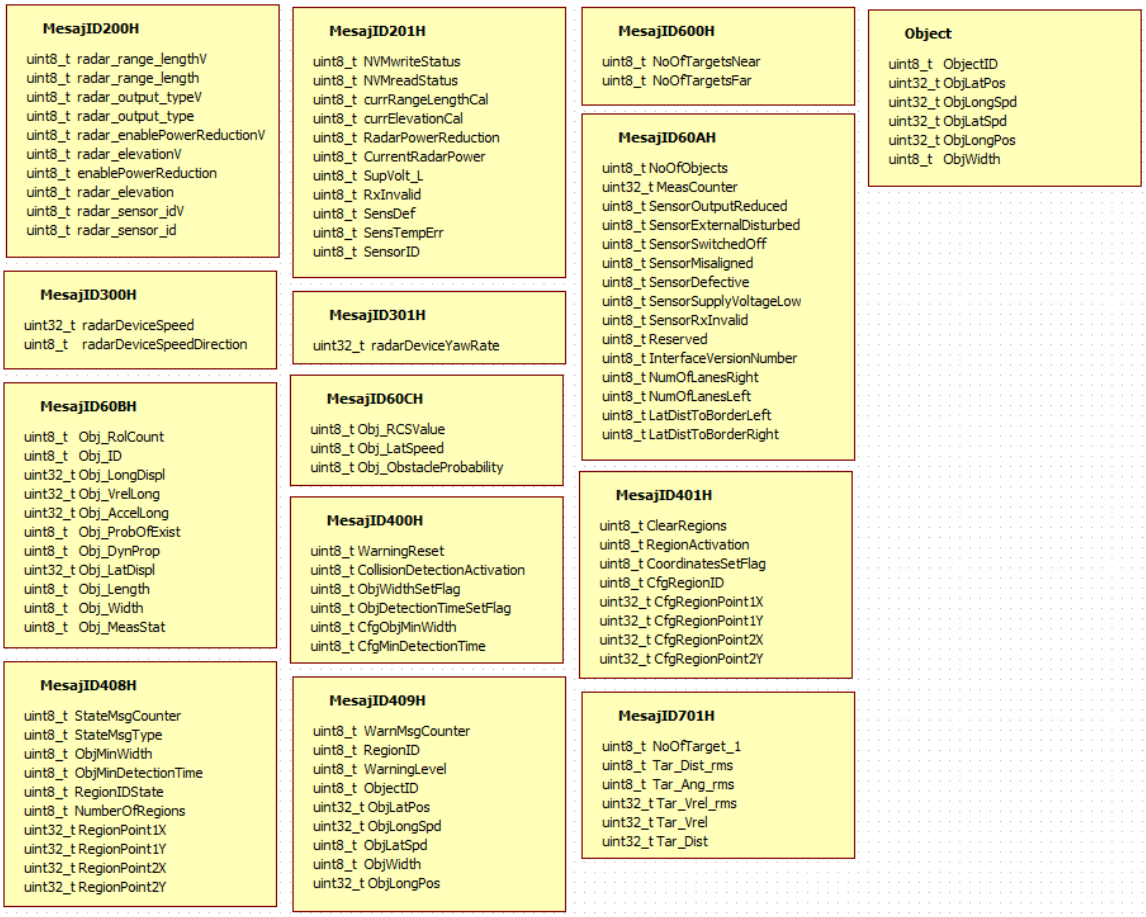

Fig 40. Structs in the RADAR Driver Created for STM32F4 DISCOVERY

6.1.2.2.3.3.2. Object Data Processing and Proposed Perception Algorithm

For the reception of the correct data for the high level algorithm, first we have found out the Message frame that sends relevant object data. With the correct configuration through the 0x200 configuration message, we have found that the message with ID 0x409 gives the crucial information that is needed by the high-level ACC algorithm. The frame 0x409 provides the data: ObjectID, ObjLatPos (Lateral position of the object), ObjLongPos (Longtitudinal position of the object), ObjLatSpd (Lateral speed of

the object), ObjLongSpd (Longtitudinal speed of the object), ObjWidth (Object width). These data are illustrated in the Figure 41.

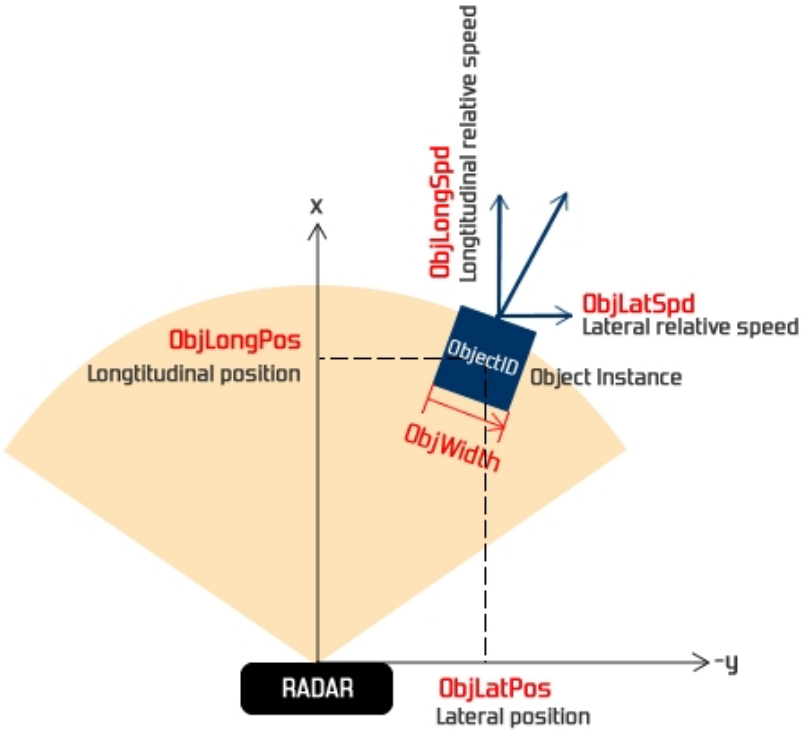

Fig 41. Illustration of RADAR Message of ID 409

Experimental tests also resulted that the right hand side of the RADAR device is (-y) whereas the left hand side is (+y). The radar provides longtitudinal speed values relatively, so that if an object is slower than the RADAR device, it gives negative values. As it is at the most importance to understand the 0x409 message data, it is necessary to correctly construct processed data from raw data. The Table 9 and Fig 42 is provided in the datasheet which is used in the driver software for processing raw data.

Fig 42. Message of ID 409

Table 9. Message of ID 409 Information

Signal	Start	Len	Byte Order	Value Type	Res	Offset	Value Range
WarnMsgCounter	0	5	Motorola	Unsigned	1	0	0 ... 31
RegionID	5	3	Motorola	Unsigned	1	0	0 ... 7
WarningLevel	8	2	Motorola	Unsigned	1	0	0 -> No warning 1 -> Object warning 2 -> Reserved 3 -> Warning deactivated
ObjectID	10	6	Motorola	Unsigned	1	0	0 ... 63
ObjLatPos	30	10	Motorola	Unsigned	0.1	-52 m	-51.9 ... 52 m
ObjLongSpd	34	12	Motorola	Unsigned	0.0625	-128 m/s	-127.9375 ... 128 m/s
ObjLatSpd	42	8	Motorola	Unsigned	0.25	-32 m/s	-32 ... 31.75 m/s
ObjWidth	56	7	Motorola	Unsigned	0.1	0	0 ... 12.7 m
ObjLongPos	63	11	Motorola	Unsigned	0.1	0	0 ... 204 m

The resolution and offset values must be considered when constructing the data from raw form. In order to that, format in Table 10 has been followed.

Table 10. Message of ID 409 Information

```
_ProcessedObject [_MesajID409H.ObjectID].ObjectID = _MesajID409H.ObjectID;
_ ProcessedObject [_MesajID409H.ObjectID].ObjLatPos =
(_MesajID409H.ObjLatPos)*res1 - off1;
_ ProcessedObject [_MesajID409H.ObjectID].ObjLatSpd =
(_MesajID409H.ObjLatSpd)*res2 - off2;
_ ProcessedObject [_MesajID409H.ObjectID].ObjLongPos =
(_MesajID409H.ObjLongPos)*res3;
_ ProcessedObject [_MesajID409H.ObjectID].ObjLongSpd =
(_MesajID409H.ObjLongSpd)*res4 -off4;
_ ProcessedObject [_MesajID409H.ObjectID].ObjWidth =
(_MesajID409H.ObjWidth)*res5;
```

In the pseudocode above, the resolution and offset values are taken from datasheet and the _MesajID409H has been constructed from 8byte data frame (given in Figx0) on CAN-bus line using Motorola byte order, using the bit operations like masking, shifting in a fashion that has been given in Table 11.

Table 11. Getting Data Using Byte Operations

```
  _MesajID409H.WarnMsgCounter =  (uint8_t)   (Data[0]&0x1F);
  _MesajID409H.RegionID =        (uint8_t)        ((Data[0]&0xE0)>>5);
  _MesajID409H.WarningLevel =  (uint8_t)        (Data[1]&0x03);
  _MesajID409H.ObjectID =        (uint8_t)        ((Data[1]&0xFC)>>2);
  _MesajID409H.ObjLatPos =       (uint32_t)
((Data[2]<<2)|((Data[3]&0xC0)>>6));
```

```
   _MesajID409H.ObjLongSpd =      (uint32_t)
(((Data[3]&0x3F)<<6)|((Data[4]&0xFC)>>2));
   _MesajID409H.ObjLatSpd =       (uint32_t)
(((Data[4]&0x03)<<6)|((Data[5]&0xFC)>>2));

   //3 portions
   uint32_t OLP_HighByte =       (uint32_t)    (Data[5]&0x03);
   uint32_t OLP_MidByte =        (uint32_t)    Data[6];
   uint32_t OLP_LByt =           (uint32_t)    ((Data[7]&0x80)>>7);

   _MesajID409H.ObjLongPos =     (uint32_t)
((OLP_HighByte<<9)|(OLP_MidByte<<1)|(OLP_LByt));
   _MesajID409H.ObjWidth =       (uint8_t)  (Data[7]&0x7F);
```

The struct indexing approach in Table 10 gives us the opportunity to use struct array indexes to hold the ObjectID's of objects. The array _ProcessedObject is of size OBJECT_COUNT=64 as it is the maximum number of objects that can be detected, given by datasheet.

The information of (maximum of 64) objects is needed to be reduced to a form that can be used in the high-level algorithm. In the construction of the filter that selects the relevant object, the algorithm in Fig 43 has been developed. With the help of this filter, it is intended to filter out the roadside objects such as trees and poles while using ACC system. In the complete algorithm, we search every object having lateral position that does not exceed the width of the vehicle and having nonzero longtitudunal distance

value to find the object having minimum longtitudunal distance. This filter is illustrated in Fig 44. This processed data is later transmitted to the High-Level unit for usage using Ethernet (TCP Socket) communication.

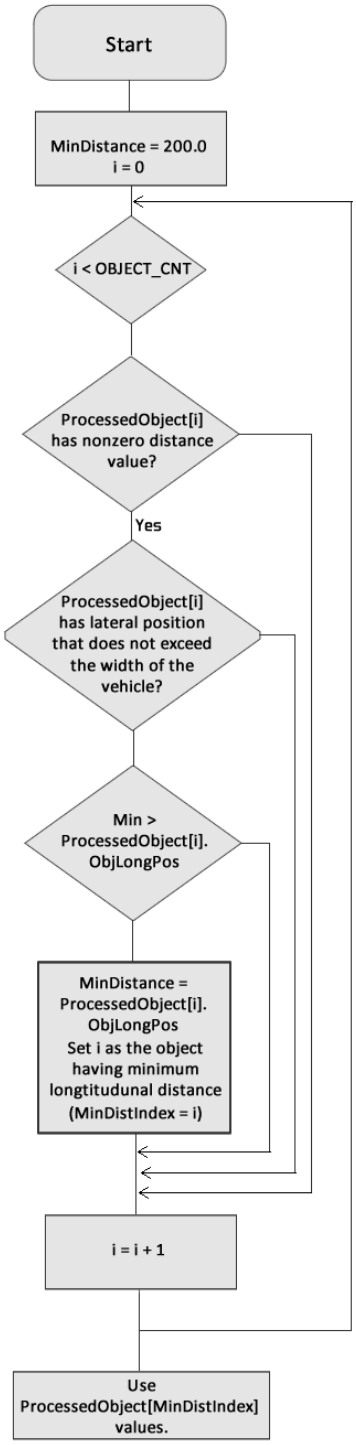

Fig 43. Selecting Relevant Object Index

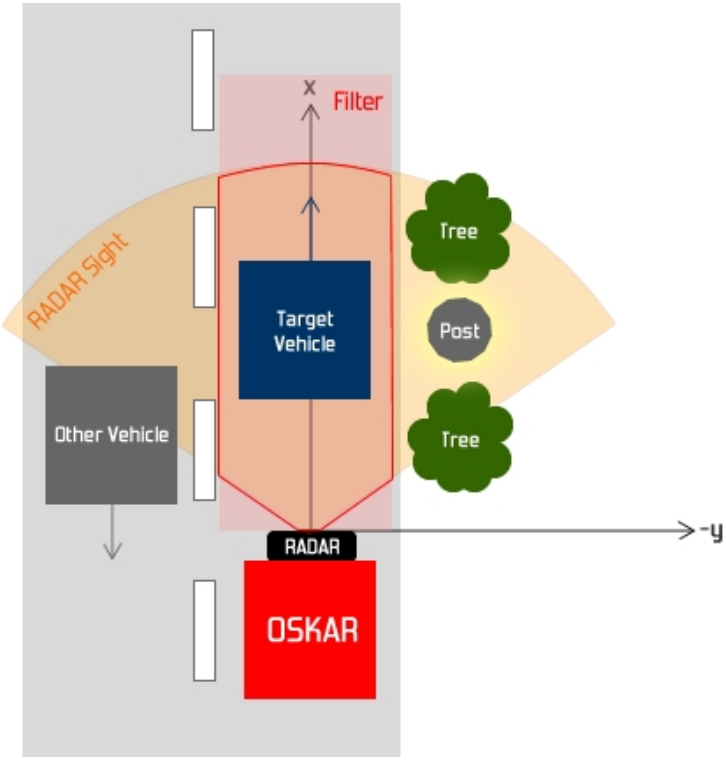

Fig 44. Filter Illustration

6.1.2.2.3.3.3. RADAR Data Logging for Correction

For the further stages of the project, a FatFs-based SD Card I/O program has also been written to log the RADAR raw data and processed data information to visualize the data and see whether the driver that has been written in embedded system works. Table 12 and Fig 44 demonstrate the raw data log format and example, respectively.

Table 12. Radar Data Log Format

No	Mode	Mesaj ID(Hex)	RTR	DLC	D1	D2	D3	D4	D5	D6	D7	D8	Time stamp
1	0		00	00	00	00	00	00	00	00	00		

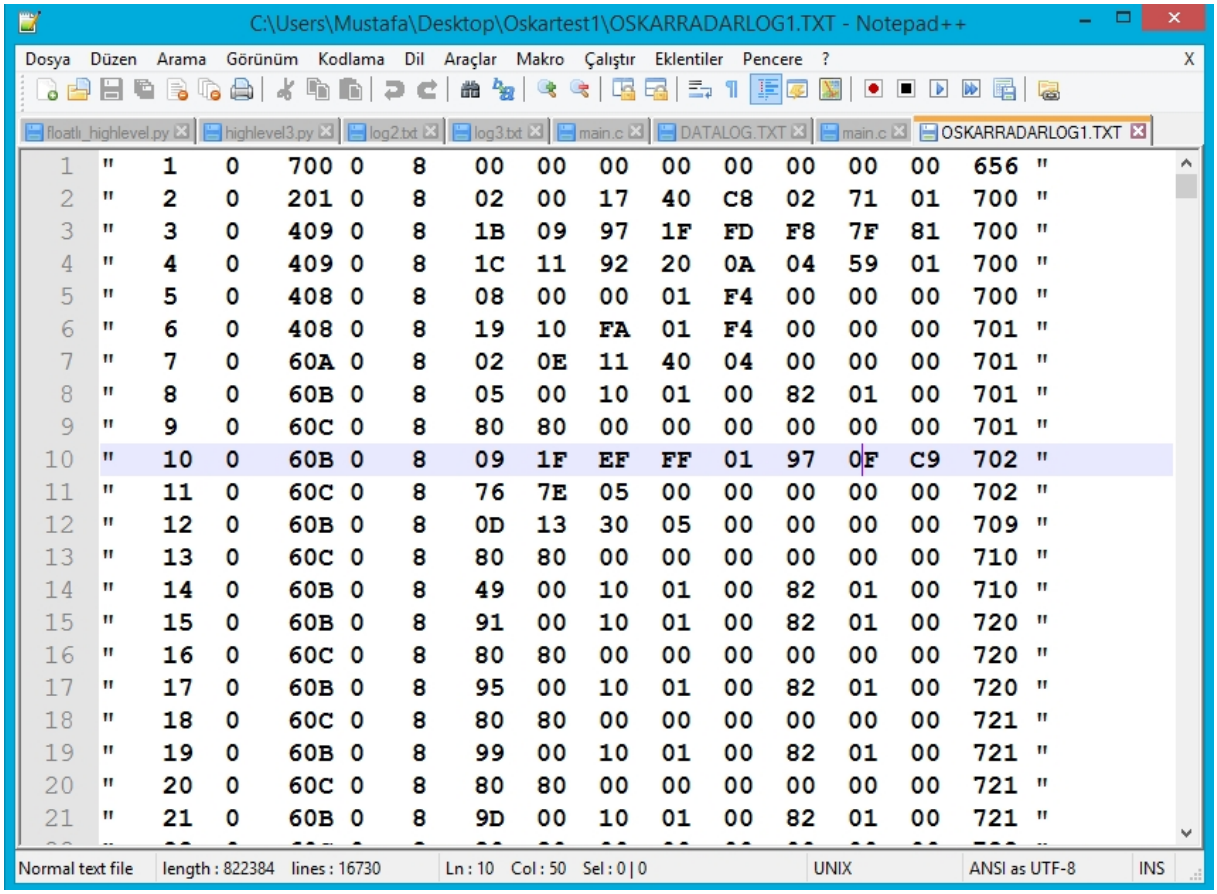

Fig 44. Radar Raw Data Log Example

It is also needed to log the processed data for comparison. For that purpose, we have created a program that logs the data from 0x409 and 0x60B messages in the form of Table 13. In this table, it is seen that the 0x60B provides displacement values for the objects in addition to the standart 0x409 message data.

Table 13. Data Log format for Processed Data

Count	ReleventMsgID	ObjectID	ObjLatPos	ObjLongSpd	ObjLatSpd	ObjWidth	ObjLongPos	ObjLongDisp	ObjLatDisp	ObjVrelLong	ObjAccelLong	TimeStamp
	409							0	0	0	0	
	60B		0	0	0	0	0					

The created algorithm to log data uses the SDIO port in the STM32F4-DISBB expansion kit. "a+" append permission has been used in order to create two text files: RAW.TXT and PROC.TXT.

The string library that has been compiled in the Ethernet work is actively used in SDIO algorithm. A STRING_BUF[33] buffer is needed to be constructed in order to hold the data. The Table 14 illustrates the pseudo code of one row of data log.

Table 14. Pseudo Code of Logging one row of data

```
//Determine Object ID
iz = _MesajID60BH.Obj_ID;


//Write ObjLongPos
FloatToStr(_ProcessedObject[iz].ObjLongPos);
f_write(&file, STRING_BUF, strlen(STRING_BUF), &bw);
f_write(&file, "\t", strlen("\t"), &bw);
```

6.1.2.2.3.4. Low Level TCP Socket Data Transmission & Reception

In order to send the RADAR, encoder and low-level cruise state information to high-level unit for control and display purposes, it has been selected to use Ethernet communication. Since Transmission Control Protocol (TCP) communication is interfaced with RJ45 connector, which is easy to be accessed by the high-level unit, and the communication is based on server-client principle, the ethernet communication has been implemented and considered to be more reliable for the ACC module design.

The TCP communication requires the link layer MAC to be implemented. For the implementation of the Link layer (MAC) and the transport layer (TCP), a highly known lightweight TCP/IP Stack library "LwIP" has been used [48]. The LwIP Stack library provides socket functions in order to achieve ethernet communication. Since it is a lightweight library, it is widely used in embedded system ethernet-based applications.

In order to achieve ethernet communication with STM32F4-DISCOVERY board, the expansion kit STM32F4DIS-BB is needed which has a LAN8720 based ethernet communications interface. The expansion kit is shown in Fig 45.

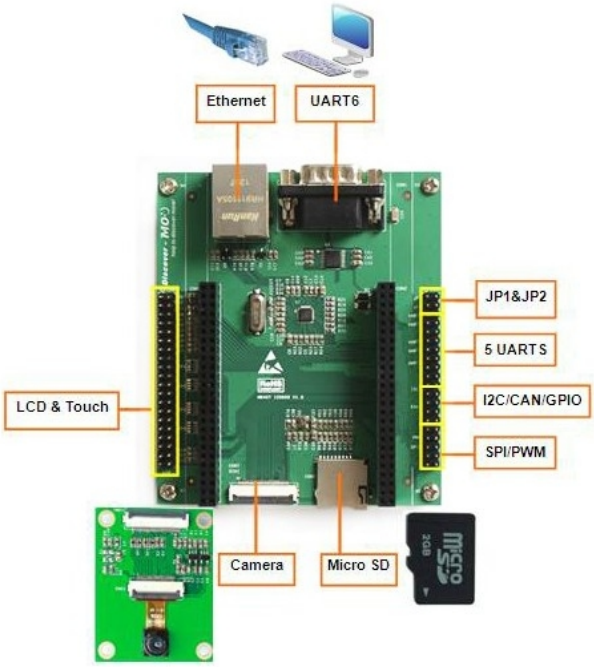

Fig 45. STM32F4-DISBB Expansion Kit

By using the LwIP sources and simple communication examples, the Netconn API information has been gathered in order to continue implementing the TCP software. In Netconn API, following functions has importance:

`netconn_new();` Creates a new communication variable.

`netconn_bind();` Bind the communication variable with a desired port.

`netconn_accept();` Accepts incoming communication request.

`netconn_recv();` Loads incoming data package into buffer.

`netconn_close();` Closes a connection.

`netconn_delete();` Delets a communication variable.

Further research lead that Netconn API has a struct buffer that can be manipulated.

```
struct netbuf {
  struct pbuf *p, *ptr;
  struct ip_addr *addr;
  u16_t port;
#if LWIP_NETBUF_RECVINFO
  struct ip_addr *toaddr;
  u16_t toport;
#endif /* LWIP_NETBUF_RECVINFO */
```

In this struct, pbuf is traced and found to be the struct type variable that holds the payload and length information. The payload and length information can be used to achieve either sending or reception of data. The pbuf variable can be loaded by either using netbuf_data(netbuf, &data, &len) function of by manipulating netbuf->p->payload and netbuf->p->len variables.

The basic TCP Server implementation using Netconn API is as follows: (1)- Create a new connection identifier. (2)- Bind it with the desired port. (3) - Start listening incoming connection requests. (4) - Accept connection if any exist. (4)- Receive a data and investigate the incoming data to decide if it is desired data or not. (5)- Send a reply. (6)- Delete buffer and close connection.

By using the TCP server implementation that has been described above, a skeleton code has been created by using an RTOS scheduler. The data that is going to be sent is constructed of XML (eXtensible Markup Language) which is widely used in World Wide Web and represents a structure in data transmission. The XML format design in Table 15 has been achieved:

Table 15. Designed XML Format for Low-level TCP Communication

```
<data>
<items>
<item V_Host=""></item>
<item V_CC_Setpoint=""></item>
<item Cruise_Button_State=""></item>
<item Vehicle_Accel=""></item>
<item X=""></item>
<item V_Relative=""></item>
</items>
</data>
```

In order to construct a string format like the XML given in Table 15, many string functions are required that are not fully supported in embedded systems, which brings an issue. For this purpose, a research has led to manually written float to string, integer to string, string to float functions. Moreover, the sprintf and strcat functions has been used in order to copy a string and concatanate another strings to it. Once the data of the struct in Table 16 is constructed and sent as a string in Table 15, the High-Level unit which acts as a client responses back the desired velocity of the vehicle, which is then set as the setpoint velocity for the PID controller, if the Cruise state is ON.

Table 16. Struct to hold High Level Data in Low-Level Software

```
struct HighLevelData
{
    float V_Host;
    float V_Relative;
    float V_CC_Setpoint;
    float X;
    int Cruise_Button_State;
    float Vehicle_Accel;
};
```

The illustration of the data transmission between Low-Level and High-Level unit is given in Fig 46.

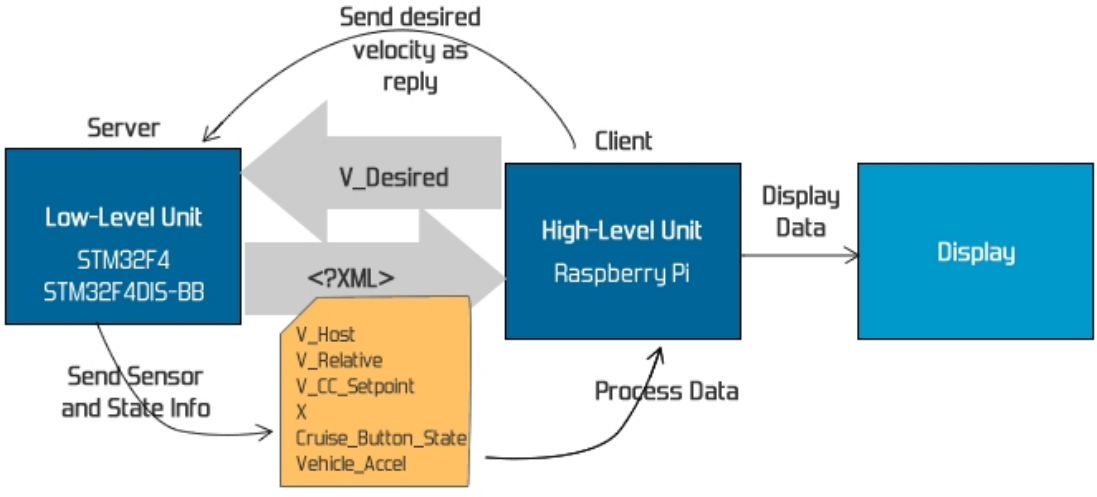

Fig 46. Data Transmission using TCP between Server and Client

6.1.2.2.3.5. Encoder Implementation

In our first version of encoder , stm32 device directly counted pulses which come from hall effect sensor, without any filters. But there was lots of noises because of vehicle vibration and this noise made lots of ripple in data line. After that we dediced to add some filters . But STM32 device is responsible for too many periodical tasks and encoder made lots of interrupts so applying filters in every interrupt can crash other periodical tasks. To prevent this we decided to add atmega 328p microcontroller to our circuit which is only responsible for encoder.

In atmega 328p device , using interrupt function we catch data pulses from hall effect sensor. First filter responsible from ignoring ripples like in the Figure 47.

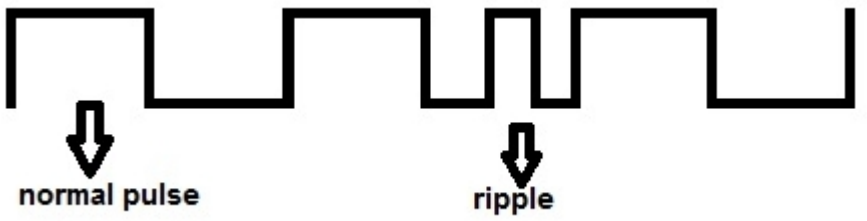

Fig 47. Encoder pulse illustration

For ignoring ripples this filter need maximum speed of vehicle. Oscars maksimum speed is 30 km/h, the filter calculates the minimum time between two pulses according to maximum speed. If an interrupt occurs during this time the filter detects the ripple and ignores it.

We calculate the speed at every interrupts ,from the time between last 2 pulses. After this calculation second filter checks last two speed. If this two speed values more than %10 different from each other , the filter don't send the speed value to parallel lines. Other case speed value is sending paralel line for STM32 device.

6.2. High Level

6.2.1. High Level Design

In ACC Unit, lots of methods such as MPC, PID, PD, PFM and Fuzzy Logic etc., which are used in literature, are investigated. Seven different control algorithms for ACC Unit were eliminated when the electric car environment was considered. As a result of investigations, two control algorithms were concluded. Although these control algorithms were Model Predictive Control (MPC) and Proportional Derivative (PD) Control, the data obtained from researches are limited for choosing exact control algorithm, and for this reason the ACC Unit is implemented with Potential Fields Method (PFM). This method is based on the Fuzzy Logic Controller but simple version. Potential Field Method involves modeling a vehicle as a particle in space, acted on by some combination of attractive and repulsive fields [49]. Obstacles are modeled as charged surfaces and the net potential generates a force on the vehicle, and these forces push the vehicle away from obstacles while pulling it towards the goal. The vehicle tends to moving in the direction of greatest negative gradient in the potential which means a hard braking to avoid collision [50]. Potential fields can be applied on path determination, trajectory planning in real time. The idea of obstacles exerting virtual repelling forces towards a vehicle, while the target generates a virtual attractive force uses a similar concept that takes into consideration the vehicle's velocity in the vicinity of obstacles. PFM is suitable for real-time motion planning of vehicle since the algorithm is simple and computationally easier than other methods. While Potential Fields Method is widely used in obstacle avoidance applications for mobile robots and

manipulators, this method is used in the project for ACC implementation which contains in addition to collision avoidance, light braking from braking distance to safe following distance, following the front vehicle, hard braking (to avoid collision) from safe following distance to critical distance and standstill.

In the project, ACC Unit has two main tasks. The first task is, as conventional Cruise Control (CC) systems, maintaining a steady speed that is set by the driver. The second and crucial task is carrying out Adaptive Cruise Control (ACC) which provides with two modes of control, velocity and distance control. Unlike the conventional Cruise Control, ACC can automatically adjust velocity in order to maintain a proper distance between obstacle (target car) and the vehicle (host car equipped with ACC). When the obstacle is far away from the safe distance, ACC Unit performs first task which is adjusting the desired speed set by the driver. In order to perform these tasks, RADAR sensor is used to measure the relative distance and relative speed between the host vehicle and a vehicle in front.

In order to implement ACC Unit, the relative speed and distance data measured by RADAR sensor is processed with the PFM algorithm. Output of the PFM algorithm is the desired speed for the vehicle which is sent to Low Level as input and according to Low Level design the desired speed is converted to voltage that is applied to motor driver.

6.2.2. High Level Implementation

6.2.2.1. Hardware Implementation

High-level system is designed to be based on the Raspberry Pi, which is explained in detail in the section of "6.2.2.2.1. Development Environment". As the high- level hardware, a 3.5" primary screen display unit manufactured by 4DPi Systems (shown in Fig 48) is used in extension to the Raspberry Pi. This is a display unit that is controlled via SPI (Serial Peripheral Interface) by installing the kernel specifically designed for the Raspberry Pi.

Fig 48. 4D Systems 3.5" Primary Screen Display Unit

6.2.2.2. Software Implementation

6.2.2.2.1. Development Environment

Raspberry Pi Model B+ development kit is used for the ACC Unit and User Interface Unit implementation. The Raspberry Pi is a credit card sized single board computer based on the Broadcom BCM2835 system on a chip (SoC) which includes an ARM1176JZFS 700 MHz processor. The kit uses Dual Core VideoCore-IV Multimedia Co-Processor as GPU that provides Open GL ES 2.0. It includes 512MB SDRAM (shared with GPU) and Micro SD Card socket (supports up to 8GB) for booting the operating system, media and persistent storage. It runs some versions of the Linux operating system such as Debian, Arch Linux ARM, Raspbian, Pidora and also non-Linux distributions as RISC OS. It has 4 USB 2.0 ports, 40 GPIO pins, HDMI connector as a video output, 3.5mm jack for audio output. Onboard network is provided by 10/100 Mbit/s Ethernet or Wi-Fi adapter. For low level peripherals it provides UART, PWM, I2C, SPI, I2S, and Ethernet. It operates at 5V power supply via Micro USB or GPIO header. 5V supply has polarity protection, 2A fuse and hotswap protection. It also has dual step-down (buck converter) power supply for 3.3V and 1.8V. Raspberry Pi supports C, C++, Java, Perl and Ruby programming languages but Python [51] is widely used as the main programming language with support for BBC BASIC (via RISC OS image or Brandy Basic clone for Linux). The device is shown in Fig 49.

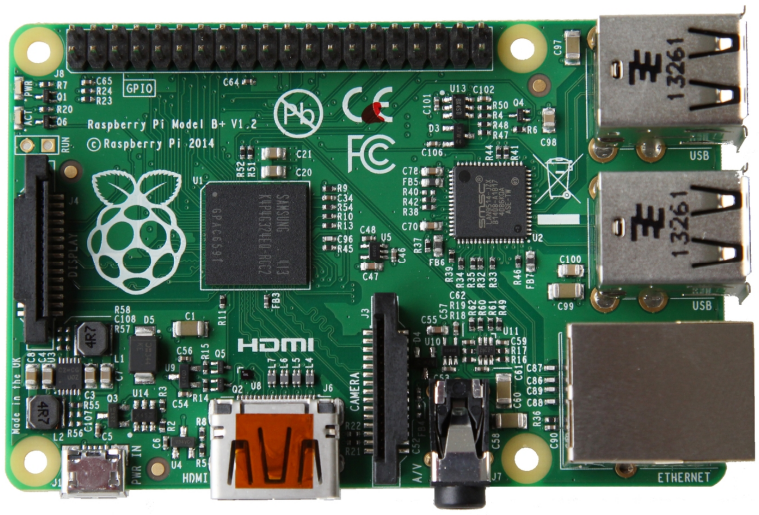

Fig 49. Raspberry Pi Model B+

Raspberry Pi Model B+ features are given below:

Chip: Broadcom BCM2835 SoC

Core architecture: ARM11

CPU: 700 MHz Low Power ARM1176JZFS Applications Processor

GPU: Dual Core VideoCore IV Multimedia Co-Processor that provides Open GL ES 2.0, hardware-accelerated Open VG, and 1080p30 H.264 high-profile decode, Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure

Memory: 512MB SDRAM

Operating System: Boots from Micro SD card, running a version of the Linux Operating System

Power: Micro USB socket 5V, 2A

Ethernet: 10/100 BaseT Ethernet socket

Video Output: HDMI (rev 1.3 & 1.4)

Audio Output: 3.5 mm jack, HDMI

USB: 4 × USB 2.0 Connector

GPIO Connector: 40 pin 2.54 mm (100mil) expansion header: 2 × 20 strip

Providing 27 GPIO pins as well as +3.3V, +5V and GND supply lines

Camera Connector: 15-pin MIPI Camera Serial Interface (CSI-2)

Display Connector: Display Serial Interface (DSI) 15 way flat flex cable connector

Memory Card Slot: SDIO

For the High Level implementation, Python programming language is used which is executed in the Raspbian (Linux based Operating System) of the Raspberry Pi development kit. Python is a widely used general-purpose, high-level programming language. Design philosophy of the language emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than other languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale. It supports multiple programming paradigms, including object-oriented, imperative and functional programming [52]. For Python programming, Python shell is used for execution which is available in both Windows and Linux systems. An instance of Python shell is shown in Fig 50.

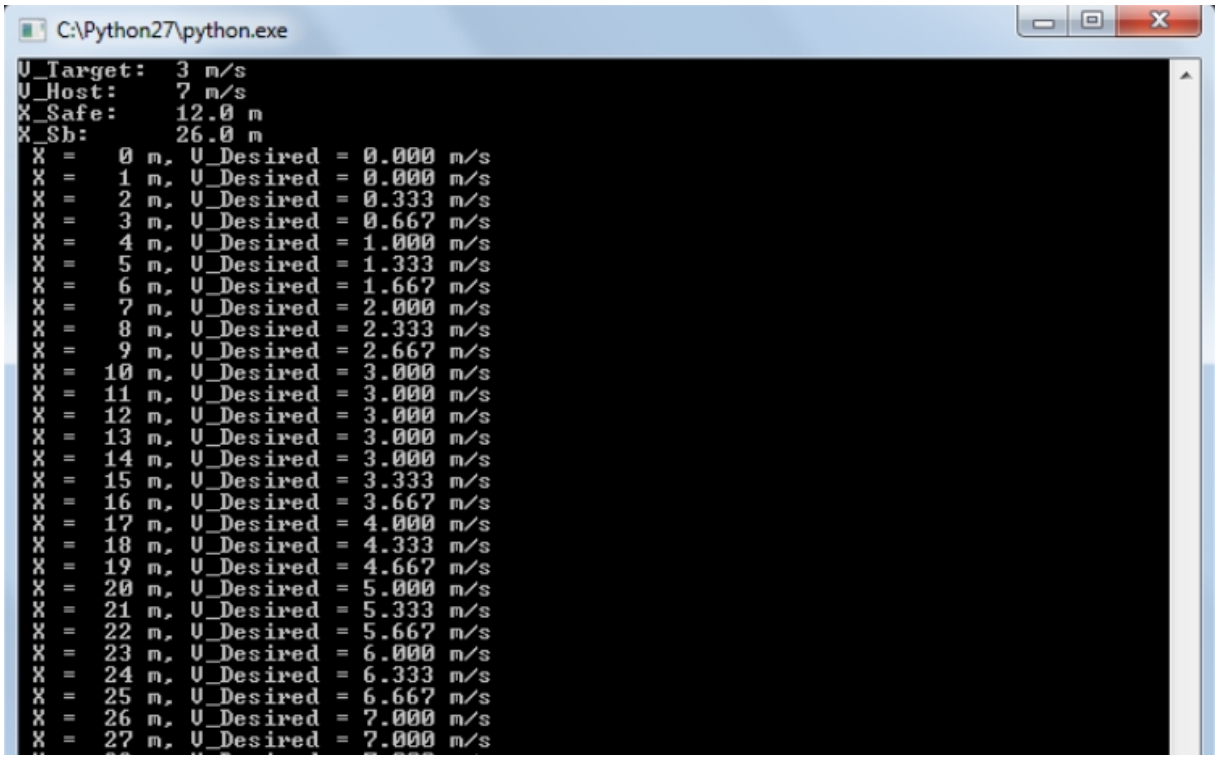

Fig 50. Python Shell instance

6.2.2.2.2. Adaptive Cruise Controller (High Level Decision) Implementation

The data coming from the RADAR sensor, and information related to follower (host) car and leading (target) car are given in Table 17 with their symbol and explanation.

Table 17. Variables used in the Adaptive Cruise Control Unit

X_Safe	Minimum safe distance between host and target car	Metre (m)
Th	Constant Time Headway	Second (s)
V_Host	Velocity of the host (follower) car	Velocity (m/s)
d0	Minimum safe interdistance used in X_Safe calculation	Metre (m)
X_Safe_Tolerance	Tolerance distance for X_Safe required to track the vehicle in front	Metre (m)
X	Actual distance between host and target car (Measured from Radar)	Metre (m)
X_Stop	Critical distance to avoid collision	Metre (m)
Acceleration	Deceleration of the host car (Required for X_Sb calculation)	Acceleration (m/s^2)
X_Sb	Distance the host car start braking	Metre (m)
V_Target	Velocity of the leading (target) car	Velocity (m/s)
V_Relative	Host car speed according to target car (Relative speed measured from Radar)	Velocity (m/s)
V_CC_Setpoint	The velocity that is adjusted when the ACC is opened (Cruise Control Setpoint velocity)	Velocity (m/s)
V_Desired	Output of the ACC Unit (Desired velocity applied to the host car)	Velocity (m/s)

In order to implement ACC Unit, the relative speed and distance data measured by RADAR sensor is processed with the PFM algorithm. Adaptive Cruise Control model is created using the variables that are given in the Table 17. To calculate these variables dynamically, the PFM algorithm model's control design must be determined which is given below.

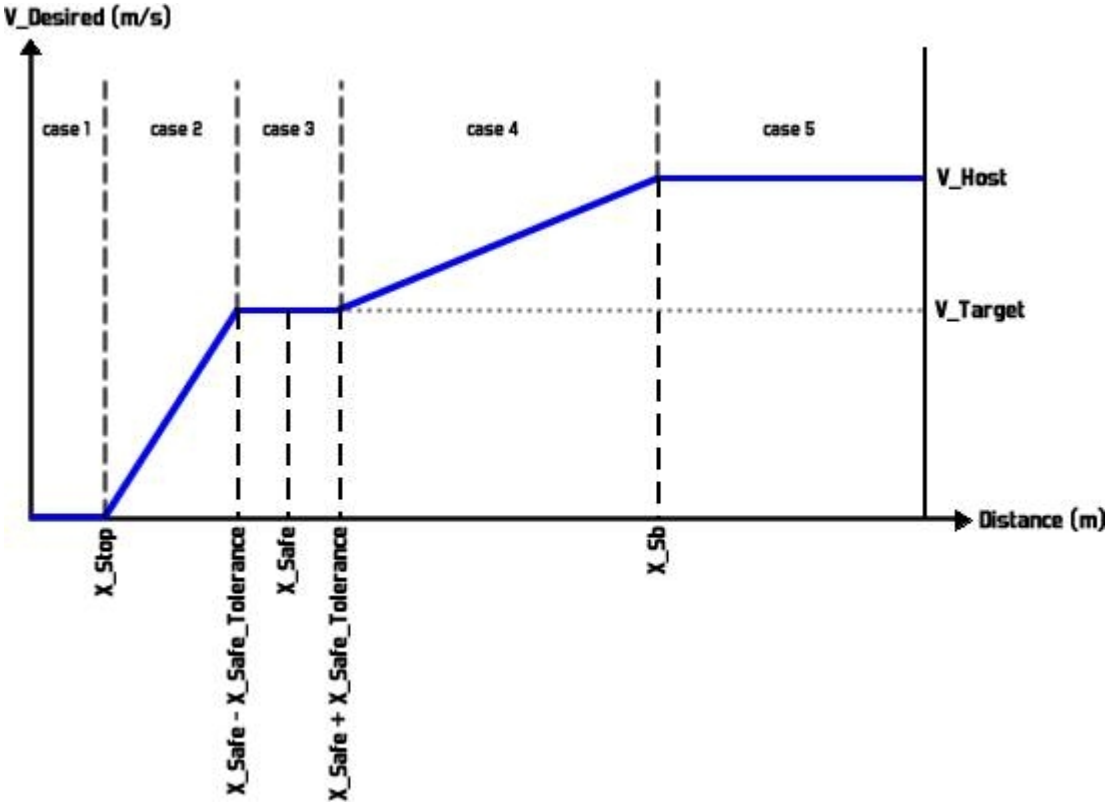

Fig 51. PFM Control Design of the Adaptive Cruise Control Unit

As it seen in the Figure 51, in ACC Unit design, the velocity of the follower (host) car must be greater than velocity of lead (target) car in order to control the distance. According to control design of the ACC Unit, some constants and dynamically performed calculations are given below:

Th: 1 second	$X\_Safe = (Th * V\_Host) + d0$
d0: 5 metres	$Acceleration = 2\ m/s^2$
X_Stop: 1 metre	$V\_Target = V\_Relative + V\_Host$
X_Safe_Tolerans: ± 2 metres	$X\_Sb = dc + X\_Safe + \frac{(V\_Host - V\_Target)^2}{(Acceleration)^2}$
$dc = d0 + vehicle\ length = 10$ metres	

6.2.2.2.2.1. Adaptive Cruise Controller (High Level Decision) Implementation

Desired speed (V_Desired) is determined by using the case analysis and calculations made above. The figure 52 shows the implementation of the algorithm.

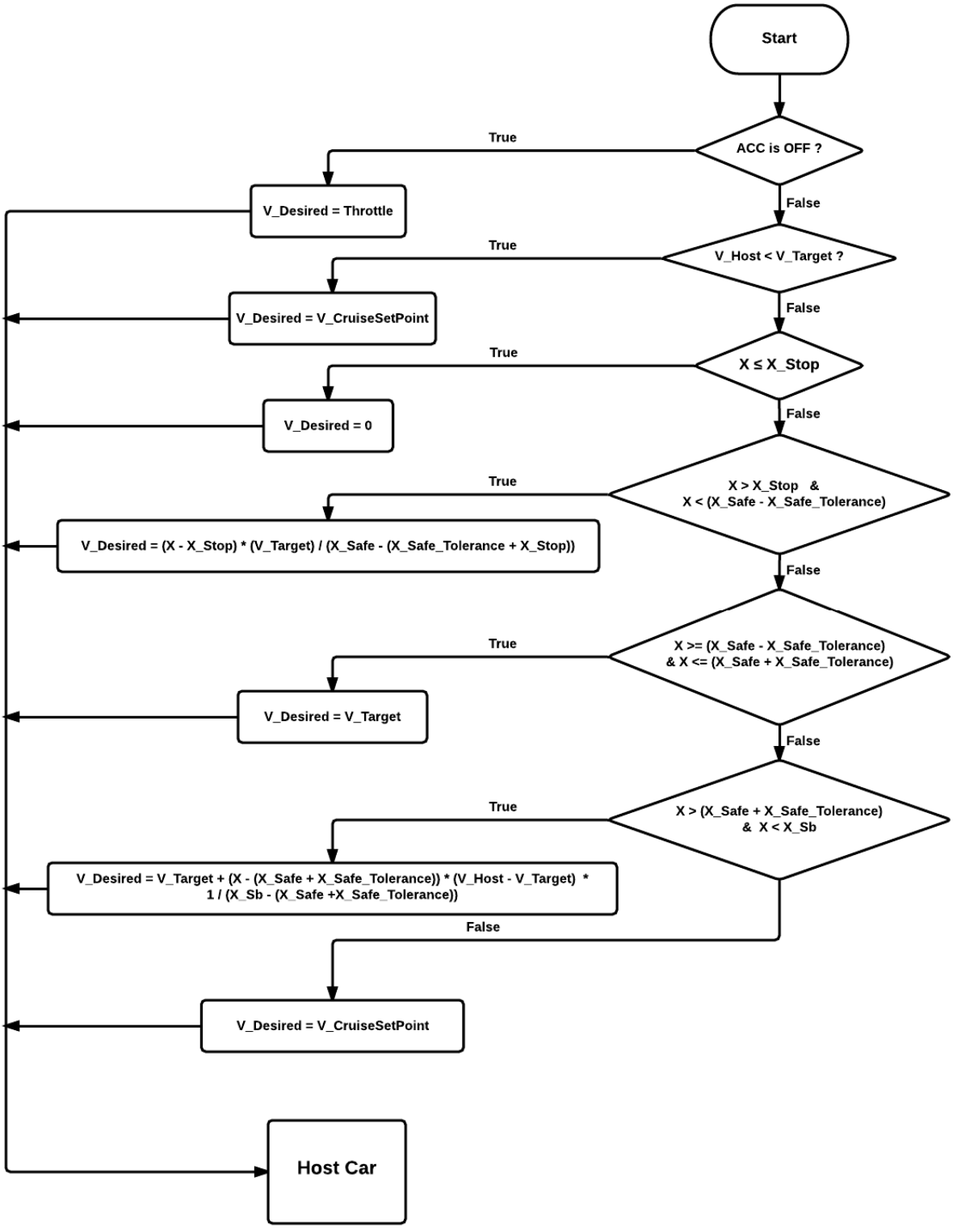

Fig 52. Adaptive Cruise Control Unit Algorithm Flowchart

Case 1: X ≤ X_Stop

If the distance between host and target car is equal or smaller than critical distance to avoid collision, host car must be stopped.

V_Desired = 0 m/s

Case 2: X > X_Stop and X < (X_Safe - X_Safe_Tolerans)

If the distance between host and target car is greater than critical distance to avoid collision and also smaller than minimum safe distance (-Tolerance distance is considered), host car must be slow down with negatively high acceleration and stop if necessary.

$$V\_Desired = (X - X\_Stop) * \left(\frac{V\_Target}{X\_Safe - (X\_Safe\_Tolerans + X\_Stop)}\right)$$

Case 3: X >= (X_Safe - X_Safe_Tolerans) and X <= (X_Safe + X_Safe_Tolerans)

If the distance between host and target car is equal to minimum safe distance (±Tolerance distance is considered), host car must follow the lead (target) car with lead car's velocity.

V_Desired = V_Target

Case 4: X > (X_Safe + X_Safe_Tolerans) ve X < X_Sb

If the distance between host and target car is greater than minimum safe distance (+Tolerance distance is considered) and smaller than host car start braking distance, host car must be slow down with negatively low acceleration.

$$V\_Desired = V\_Target + \frac{(X - (X\_Safe + X\_Safe\_Tolerans)) * (V\_Host - V\_Target)}{X\_Sb - (X\_Safe + X\_Safe\_Tolerans)}$$

Case 5: X ≥ X_Sb

If the distance between host and target car is greater than host car start braking distance, host car's velocity must equal to Cruise Control Setpoint velocity.

V_Desired = V_CC_Setpoint

6.2.2.2.2.2. Simulation and Results

The accuracy of the code, that is written in python programming language, is tested in software environment Matlab simulations before starting hardware and field tests. In the Matlab (Fig 53), responses that host vehicle will behave is analyzed according to various scenarios. These scenarios were converted to graphs that show the desired speed of the host vehicle by changing the distance between the host and target vehicles.

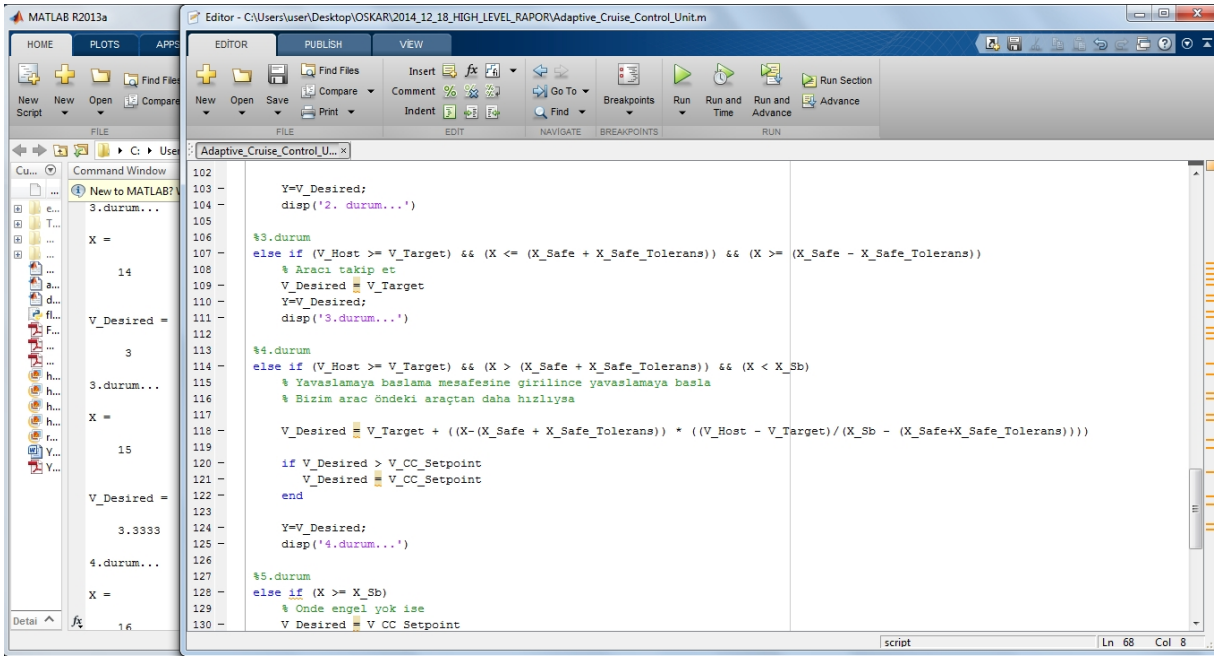

Fig 53. MATLAB Environment

Scenario 1:

V_Host = 7m/s, V_Target = 3m/s, V_CC_Setpoint = 7m/s, X_Safe = 12m, X_Sb = 26m

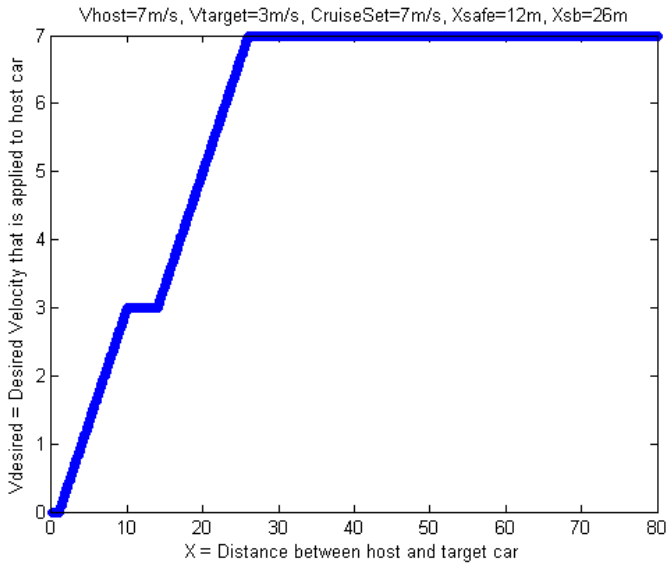

Fig 54. Scenerio 1

The ACC unit is opened when the host car's velocity is 7 m/s and the target car's velocity is 3 m/s. When the distance between host car and target car is lower than 26 metres, which is equals to start braking distance, the host car slow down with negatively low acceleration until the distance is 14 metres. When the distance between host car and target car is between 10 and 14 metres, which is equals to minimum safe distance (±Tolerance distance is considered), host car follows the lead car with lead car's velocity. When the distance between host and target car is lower than 10 metres, which means the distance is smaller than minimum safe distance (-Tolerance distance is considered), host car slow down with negatively high acceleration. When the distance between host and target car is equal or smaller than critical distance to avoid collision (1 metre), host car is at rest.

Scenario 2:

V_Host = 5.5m/s, V_Target = 1m/s, V_CC_Setpoint = 5.5m/s, X_Safe =10.5m, X_Sb = 25.5625m

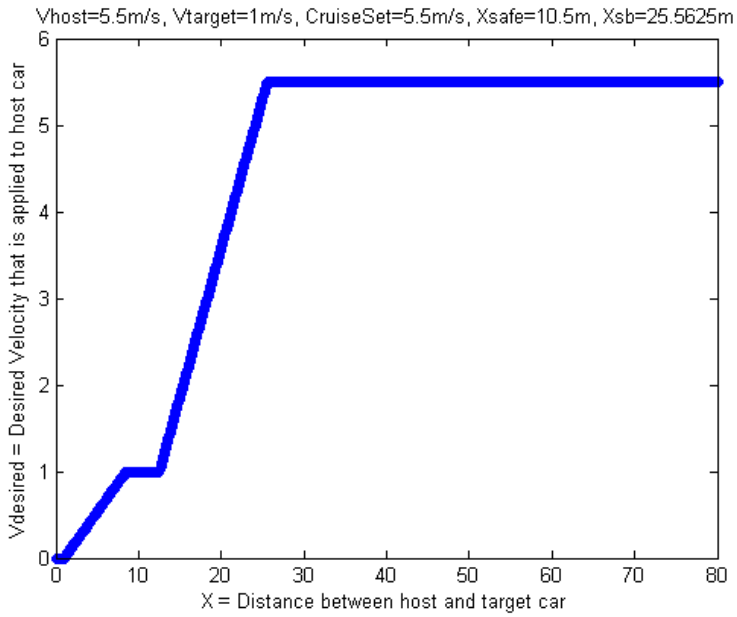

Fig 55. Scenerio 2

The ACC unit is opened when the host car's velocity is 5.5 m/s and the target car's velocity is 1 m/s. When the distance between host car and target car is lower than 25.5625 metres, which is equals to start braking distance, the host car slow down with negatively low acceleration until the distance is 12.5metres. When the distance between host car and target car is between 8.5 and 12.5 metres, which is equals to minimum safe distance (±Tolerance distance is considered), host car follows the lead car with lead car's velocity. When the distance between host and target car is lower than 8.5 metres, which means the distance is smaller than minimum safe distance (-Tolerance distance is considered), host car slow down with negatively high acceleration. When the distance between host and target car is equal or smaller than critical distance to avoid collision (1 metre), host car is at rest.

Scenario 3:

V_Host = 2m/s, V_Target = 4.5m/s, V_CC_Setpoint = 2m/s

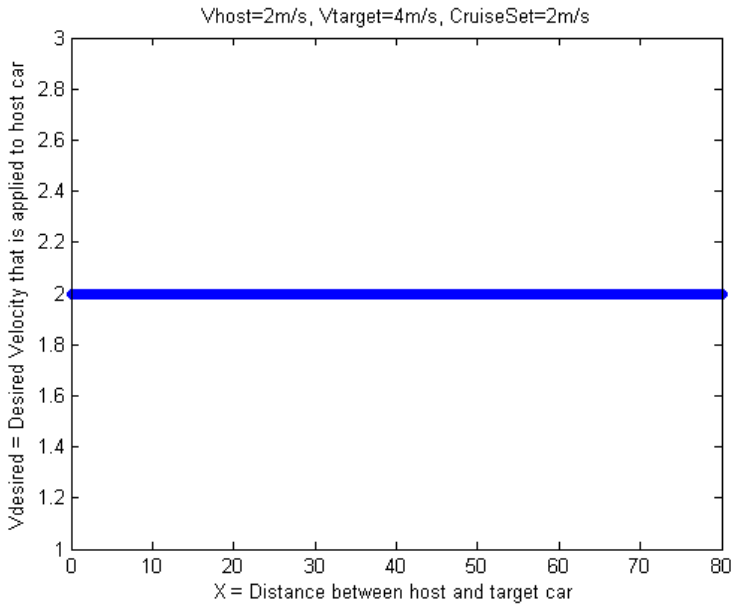

Fig 56. Scenerio 3

The ACC unit is opened when the host car's velocity is 2 m/s and the target car's velocity is 4 m/s. Because of the host car velocity is smaller than target car velocity, the host car adjust its velocity at 2m/s.

6.2.2.2.3. User Interface Software

In the display unit implementation, it has been decided to use Pygame library, which is an open-source Python library for making games and interfaces [53]. The gathered data from low-level unit (by using TCP socket) is visualized in the interface that has been designed. After the library has been set up, a layout is created in Adobe Fireworks

software which is a web graphics design software. The layout that is designed is shown in Fig 57. Fig 58 shows the level indicators that has been designed to illustrate target information and acceleration. The software is also capable of manually manipulating the cruise setpoint speed by pressing + and - buttons on the touchscreen. Moreover, the cruise setpoint is also set to the host car speed when the cruise button is pressed.

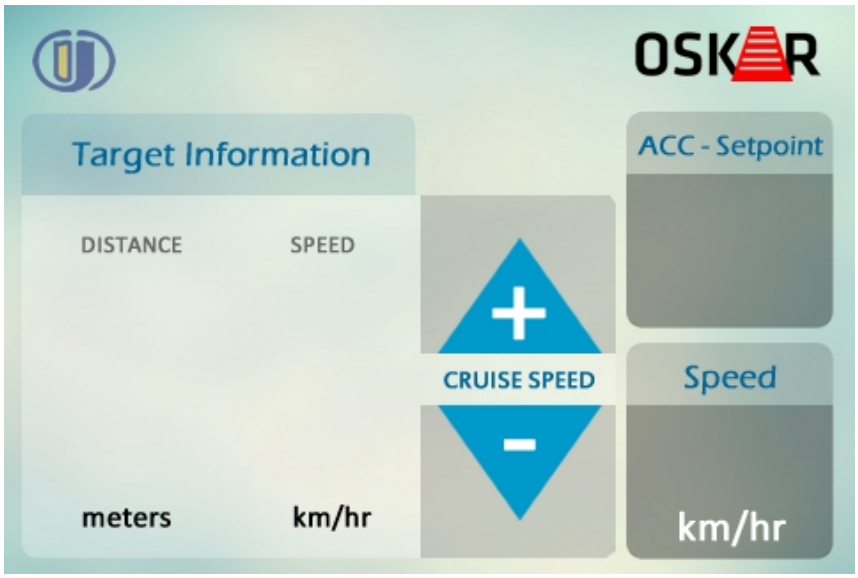

Fig 57. OSKAR Display graphics

Fig 58. Level graphics

This visual is imported into the pygame workspace in order to be improved via coding. The flowchart of the high-level display unit is provided in the Fig 59.

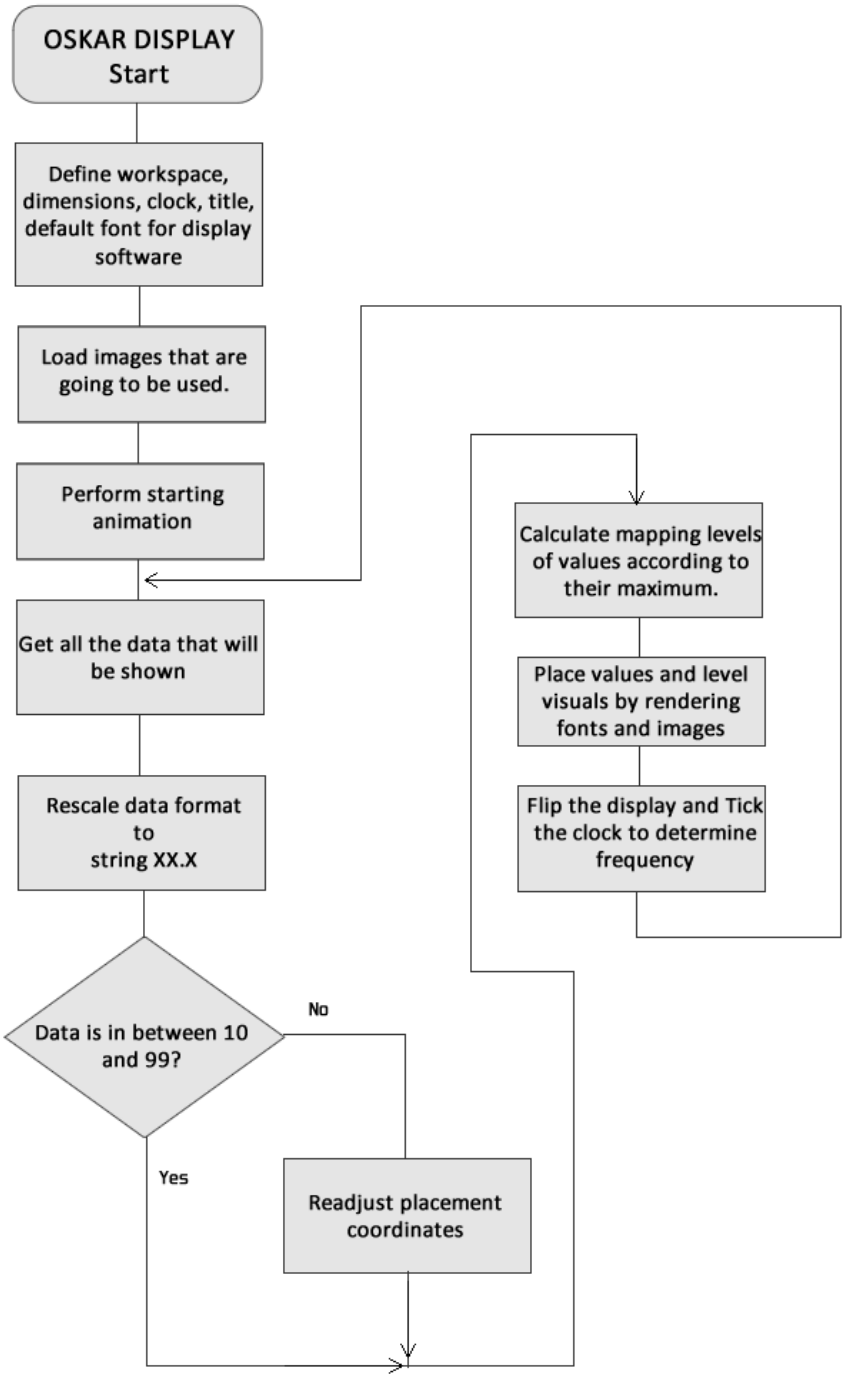

Fig 59. Display algorithm flowchart

Fig 60. Final layout of the display

Fig 60 shows the final layout that has been designed in the Ubuntu 12.04 OS using Python Shell.

6.3. Overall Design Block Diagram

Overall system block diagram has given in Fig 61, using some of the variables that are used in the design of the system. It is important to note that the high level algorithm is not explicitly included in the block diagram given, so one must refer to the section of "High Level" in order to completely understand how the system works.

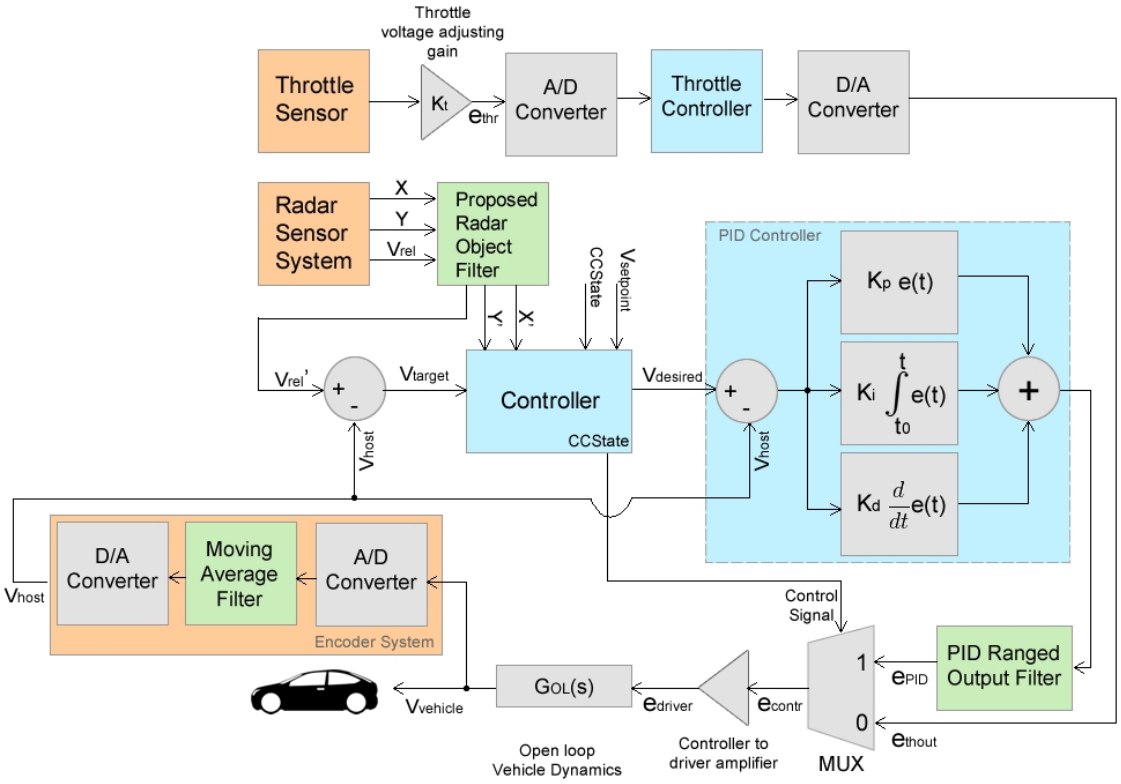

Fig 61. Overall design block diagram

7. Recommendations and Risk Analysis

In low level design, because of not having an accelerometer, measurement could not be made, and therefore, parameters and constant values that are used (indicated 6.1.1. Low Level Design) for DC motor was found from literature review. Also, cohen-coon tuning method for finding these values are still being investigated.A decision has not been made yet for the method which will be used in high level control algorithm. Risk analysis of the system is given in Table 18.

Table 18-19. Risk Analysis of the System

SEVERITY	***LEVEL***
NEGLIGIBLE	1
MARGINAL	2
CRITICAL	3
CATASTROPHIC	4

No	UNIT	POSSIBLE CAUSES	RISK	SEVERITY
1	WHEELS	Tires can be explode	Accidents and injuries	4
2	CONTROL BOARDS	Control boards can be crash	Car can't run	2
3	ENCODER SYSTEM	Encoder system can send wrong speed data	Uncontrolled speed and maybe accident	3
4	RADAR	Radar can send wrong data	Accidents and injures	4
5	BRAKE SYSTEM	The car don't have physical brake system	Can't stop immediately	2
6	CRUISE BUTTON	Button can be stuck	Uncontrolled speed and crash	3
7	IN VEHICLE COMMUNICATION	Communication problem can be occur	Car can't run	2
8	ACCUMULATOR	Running out of battery	Car can't run	1
9	CHARGE SYSTEM	Chargesystem can be burn	Fire exits	4

8. Conclusion

Today, increasing number of vehicles with the increase of traffic intensity, brings (causes) traffic accidents.In this project, as a result of development of Adaptive Cruise Control Module, driver-induced traffic accidents occuring in our country and based on these accidents, loss of life and property are intented to reduce.

In our country,on the grounds that the number of commercial studies on electronic systems used in vehicle safety within the special vehicle dynamics and control to electric vehicles are limited, be made of these studies abroad does not contribute to our country in terms of commercial and academic.In order to eliminate the dependence of our country to abroad on this point, and contribute to the domestic automobile production process, development of Adaptive Cruise Control Module is planned.

Predecessors of the product which will be project output, as well as less in the international market and be available in some cars, the predecessors of the system has not been found in the commercial market of our country. This system which will be applied to electric vehicles, has PF control algorithm which are used in the automotive industry. It is indicative of a study that will be an element of innovation.The sensors which are used, electronics cards for generating low and high level control unit, communication interfaces, motor driving techniques, technological factors such as user interface, can be applied to a lot of electric vehicles and because of these, commercial value of the Adaptive Cruise Control Module is increased and it provides widespread to product in the market.

The output of the project will contribute to the our country's automotive industry with the aspect of the development of the electronics systems which are used in vehicle

security. Project output which has increasing quality of necessary knowledge for our country in terms of industrial and academic knowledge, will prepare the environment to perform Research-Development activities, train enginners who can specialize in this area and create employment with university-industry cooperation about intelligent vehicle technologies.

The one of the main part of the system design is low level control unit and in the low level design, mathematical model of DC motor and it's constant parameters were obtained from literature research. Mathematical model of the vehicle was implemented on MATLAB environment by Root-Locus, Cohen- Coon and PID Tuner tool of MATLAB. As a result, PID Tuner tool is used because of giving better results in tests sequence. Also, desired performance specifications were mostly satisfied in outdoor test. In high level design which is the other main part of the system design, control algorithm of the Adaptive Cruise Control such as MPC, Fuzzy Logic, PFM, PID, PD, CTG, TDC were investigated with literature researches and as a result, Potential Fields Method (PFM) has been used when the high level control unit is implemented. This control algorithm make differences in usabilty, popularity and ease of implementation to other control methods.

The library of the ARS 308 radar driver has been created for STM32 microcontroller and tests of the ARS 308 radar were achieved by getting the raw object information and transforming it to a useful form. In the encoder design, after first prototype was created with using magnet and hall effect sensor with no filters, Atmega 328p microprocessor was used for capturing better the pulse occuring. Because the first protype caused lots of ripples. Also in this context, additional filters in software have been added for preventing the noises.

References

[1] htp:/www.trafik.gov.tr/Sayfalar/Istatistikler/Genel-Kaza.aspx

[2] htp:/www.trafik.gov.tr/SiteAsets/istatistik/2013_BULTEN_OZET.xls

[3] Youcef-Toumi, K.; Sasage, Y.; Ardini, J.; Huang, S. Y., "The Aplication of Time Delay Control to an Inteligent Cruise Control System," American Control Conference, 192 , vol., no., p.1743,1747, 24-26 June 192

[4] Bjornberg, A, "Autonomous inteligent cruise control," Vehicular Technology Conference, 194 IEEE 4th , vol., no., p.429,43 vol.1, 8-10 Jun 194 doi: 10.109/VETEC.194.345091

[5] Ganji, Behnam; Kouzani, Abas Z.; Kho, Sui Yang; Nasir, Mojdeh, "A sliding-mode-control-based adaptive cruise controler," Control & Automation (ICCA), 1th IEEE International Conference on , vol., no., p.394,397, 18-20 June 2014

[6] Meadows, AD.; Lingxi Li; Yaobin Chen; Widman, G., "Plant eror analysis and compensation in adaptive cruise control systems," Inteligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on , vol., no., p.438,43, 16-19 Sept. 2012

[7] Mayr, R.; Bauer, O., "Safety isues in inteligent cruise control," Inteligent Transportation Systems, 199. Procedings. 199 IEEE/IEEJ/JSAI International Conference on , vol., no., p.970,975,199 doi: 10.109/ITSC.199.82195

[8] Mayr, R., "Robust performance for autonomous inteligent cruise control systems," Decision and Control, 198. Procedings of the 37th IEEE Conference on , vol.1, no., p.487,492 vol.1, 198

[9] Pananurak, W.; Thanok, S.; Parnichkun, M., "Adaptive cruise control for an inteligent vehicle," Robotics and Biomimetics, 208. ROBIO 208. IEEE

[10] Zhang, B. S.; Leigh, I; Leigh, J. R., "Learning control based on patern recogniton applied to vehicle cruise control systems," American Control Conference, Procedings of the 195 ,vol.5, no., p.3101,3105 vol.5, 21-23 Jun 195

[11] Ioanou, P.; Xu, Z.; Eckert, S.; Clemons, D.; Sieja, T., "Inteligent cruise control: theory and experiment," Decision and Control, 193., Procedings of the 32nd IEEE Conference on , vol., no., p.185,1890 vol.2, 15-17 Dec 193

[12] Osman, K.; Rahmat, M.F.; Ahmad, M.A, "Modeling and controler design for a cruise control system," Signal Procesing & Its Aplications, 209. CSPA 209. 5th International Coloquium on ,vol., no., p.254,258, 6-8 March 209

[13] Shengbo Li; Jianqiang Wang; Keqiang Li; Dezhao Zhang, "Study on robustnes and feasibilty of MPC based vehicular Adaptive Cruise Control system," Inteligent Vehicles Symposium, 209 IEEE ,vol., no., p.1297,1301, 3-5 June 209

[14] Bageshwar, V.L.; Garard, Wiliam L.; Rajamani, R., "Model predictive control of transitonal maneuvers for adaptive cruise control vehicles," Vehicular Technology, IEEE Transactions on ,vol.53, no.5, p.1573,1585, Sept. 204

[15] Naus, G.; Ploeg, J.; van de Molengraft, R.; Steinbuch, M., "Explicit MPC design and performance-based tuning of an Adaptive Cruise Control Stop-&-Go," Inteligent Vehicles Symposium, 208 IEEE ,vol., no., p.434,439, 4-6 June 208

[16] Wei Xi; Baras, J.S., "MPC based motion control of car-like vehicle swarms," Control & Automation, 207. MED '07. Mediteranean Conference on , vol., no., p.1,6, 27-29 June 207

[17] Naus, G.; van den Blek, R.; Ploeg, J.; Schepers, B.; van de Molengraft, R.; Steinbuch, M., "Explicit MPC design and performance evaluation of an ACC Stop-&-Go," American Control Conference, 208 ,vol., no., p.24,29, 1-13 June 208

[18] Gholamhosein, M.; Khalozadeh, H., "Automotive radar data filtering aproach for Adaptive Cruise Control systems," Sensing Technology, 208. ICST 208. 3rd International Conference on , vol., no., p.10,14, Nov. 30 208-Dec. 3 208 doi: 10.109/ICSENST.208.4757064

[19] Hoseinia, S.H.; Tejado, I; Milanes, V.; Vilagra, J.; Vinagre, B.M., "Experimental Application of Hybrid Fractional-Order Adaptive Cruise Control at Low Sped," Control Systems Technology, IEEE Transactions on , vol.P, no.9, p.1,1

[20] Pathan, Dur Muhammad; Zeshan, Ali Memon; Husain, Tanwer, “Analysis of the Controlers for the Transitonal Manoeuvres of Adaptive Cruise Control Systems”, Mehran University Research Journal of Enginering & Technology, Volume 31, No. 3, July, 2012

[21] Person, M.; Botling, F.; Heslow, E.; Johanson, R., "Stop and go controler for adaptive cruise control,"Control Aplications, 199. Procedings of the 199 IEEE International Conference on , vol.2, no., p.1692,1697 vol. 2, 199

[22] Kanje, R.; Bacho, AK.; Carol, J., "Vision-based Adaptive Cruise Control using patern matching,"Robotics and Mechatronics Conference (RobMech), 2013 6th , vol., no., p.93,98, 30-31 Oct. 2013 doi: 10.109/RoboMech.2013.685498

[23] Robinson, J.; Paul, D.K.; Bird, J.; Dawson, D.; Brown, T.; Spencer, D.; Prime, B., "A milimetric car radar front end for automotive cruise control," Automotive Radar and Navigation Techniques (Ref. No. 198/230), IEE Coloquium on , vol., no., p.9/1,9/8, 9 Feb 198 doi: 10.1049/ic:1980195

[24] Hiremath, S.I; Sampagoan, S.M.; Ojanahali, S.D.; Bhajantri, S.; Kaushik, M., "Adaptive Cruise Control System for two whelers to avoid and reduce accidents," Confluence 2013: The Next Generation Information Technology Summit (4th International Conference) , vol., no., p.92,94, 26-27 Sept. 2013

[25] Richardson, M.J.; Smith, D., "Design of the driver interface for autonomous inteligent cruise control,"Design of the Driver Interface, IEE Coloquium on , vol., no., p.7/1,7/4, 19 Jan 195 doi: 10.1049/ic:195029

[26] Person, M.; Botling, F.; Heslow, E.; Johanson, R., "Stop and go controler for adaptive cruise control,"Control Aplications, 199. Procedings of the 199 IEEE International Conference on ,vol.2, no., p.1692,1697 vol. 2, 199

[27] htp:/www.businesinsider.com/audi-a4-using-adaptive-cruise-control-2012-7

[28] htp:/www.lexus.com/models/IS/safety

[29] htp:/www.bmw.com.tr/tr/en/

[30] htp:/www.acura.com/

[31] htp:/www.daimler.com/

[32] htp:/www.volvocars.com/

[33] htp:/www.vw.com.tr/

[34] htp:/www.audi.com.tr/tr/brand/tr.html

[35] htp:/www.ford.com/

[36] htp:/www.lexus.com/models/RX/safety

[37] htp:/www.bmw.com/com/en/insights/technology/technology_guide/articles/active_cruise_control.html

[38] htp:/m.acura.com/Owners/2014/RLX/HowTo/Adaptive-Cruise-Control-with-Low-Sped-Folow

[39] htp:/www.daimler.com/dcom/0-5-1210218-1-1210321-1-0-0-121028-0-0-135-0-0-0- 0-0-0-0-0.html

[40] htp:/www.volvocars.com/tr/al-cars/volvo-s80/specifications/pages/features.aspx

[41] htp:/binekarac.vw.com.tr/guvenlik.aspx

[42] htp:/www.volkspage.net/echnik/sp/sp/SP_289.PDF

[43] http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/PF252419/

[44] http://www.freertos.org

[45] http://www.ti.com/lit/an/sloa101a/sloa101a.pdf

[46] ARS Datasheet

[47] http://controls.engin.umich.edu/wiki/index.php/PIDTuningClassical/

[*] Sarin, S.; Hindersah, H.; Prihatmanto, A.S., "Fuzzy PID controllers using 8-Bit microcontroller for U-Board speed control," System Engineering and Technology (ICSET), 2012 International Conference on , vol., no., pp.1,6, 11-12 Sept. 2012

[48] http://savannah.nongnu.org/projects/lwip/

[49] Y. Koren , J . Borenstein.(1991) Potential field methods and their inherent limitations for mobile robot navigation. In Proceedings of the IEEE International Conference on Robotics and Automation.

[50] K.S . Al-Sultan and M. D. S. Aliyu.(1996) A new potential field-based algorithm for path planning. Journal of Intelligent and Robotics Systems.

[51] https://www.adafruit.com/datasheets/pi-specs.pdf

[52] https://www.python.org/

[53] http://pygame.org