REPUBLIC OF TURKEY

YILDIZ TECHNICAL UNIVERSITY

MECHATRONICS ENGINEERING

ADAPTIVE CRUISE CONTROL

Said Burak Güzel 16067043

Fateh Isgandarov 15067903

Samir Omarov 17067902

Taha Eksik 16067051

Lecturers:

Asst. Prof. Ahmet KIRLI, Assoc. Prof. Cenk ULU

May, 2020

ACKNOWLEDGEMENTS

We kindly would like to show our appreciations for Asst. Prof. Ahmet Kırlı and Assoc. Prof. Cenk Ulu for giving us the opportunity to implement our theoretical knowledge of Programmable Logic Controller's in a real-life project.

TEAM NAME

TABLE OF CONTENTS

LIST OF SYMBOLS

K_p	Proportinal Constant
K_i	Integral Constant
K_d	Derivative Constant
Ts	Sampling Time
u	Output
e	Input
z^{-n}	n Previous Sample

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ACC	Adaptive Cruise Control
BMS	Battery Management System
CAN	Controller Area Network
DBF	Digital Beam Forming
EV	Electric Vehicle
HOD	Hand Off Detection
HUD	Head-Up Display
MRR	Mid-Range Radar

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

ADAPTIVE CRUISE CONTROL

Advisors: Asst. Prof. Ahmet KIRLI, Assoc. Prof. Cenk ULU

There is perhaps no better symbol of the 21st century than the automobile. It is the dominant means of transport aspired to throughout the world; indeed, many a politician throughout this century has furthered his or her career by promising constituents a greater economic ability to own an automobile. Furthermore, there are more automobiles on the road than any other motorized vehicle. As of 1986, almost half a billion vehicles were on the road throughout the world; over three-quarters of these were cars. With this comes the responsibility of making safety a primary concern in vehicle technologies. New type of speed control, called adaptive cruise control, is being used on some new model vehicles. Adaptive Cruise Control (ACC) is an automotive feature that allows a vehicle's cruise control system to adapt the vehicle's speed to the traffic environment. It is basically an extension of conventional cruise control systems. These systems allow you to set a following distance, or time interval, between your vehicle and the vehicle ahead, as well as a maximum speed. In this paper, we present a case study on adaptive cruise control as modelled on the EV.

1
Introduction

1.1 Information

Cruise control system is developed for highway driving. This system is useful for driving in the roads which are big, straight, and the destination is farther apart. When traffic congestion is increasing, the conventional cruise control becomes less useful. The adaptive cruise control (ACC) system is developed to cope up with this situation. The conventional cruise control provides a vehicle with one mode of control, velocity control. On the other hand, ACC provides with two modes of control, velocity and distance control. ACC reduces the stress of driving in dense traffic by acting as a longitudinal control pilot. ACC can work like the conventional cruise control that it is used for maintaining the vehicle's preset velocity. Unlike the cruise control, however, ACC can automatically adjust velocity in order to maintain a proper distance between obstacle and the vehicle equipped with ACC. This is achieved by using multipurpose camera and radar measure the relative distance between the host vehicle and a vehicle in front and rear. As additionally combined with another driver assist features such as lane centering and blind spot detection lets more stable and safe drive.

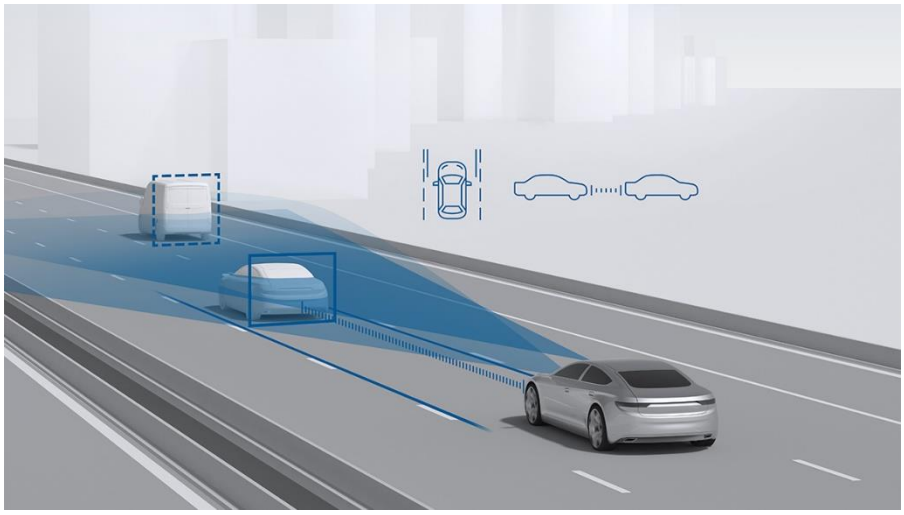


Figure 1 Simplified ACC presentation

The adaptive cruise control (ACC) can reduce stress for the driver by automatically controlling vehicle speed and maintaining a predefined minimum distance to the preceding vehicle. Therefore, the driver enjoys more comfort and can better concentrate on the traffic.

1.2 Benefits of system

The proposed system has following benefits for drivers:

- Comfortable and relaxed driving – even in heavy traffic and traffic jams
- Supports the driver in maintaining a safe distance
- Allows harmonious, fuel-efficient traffic flow on the roads
- Reduces the risk of rear-end collisions
- Driver is better able to concentrate on the current traffic situation
- Actively assists the driver to remain in the marked lane
- Enhanced safety through early correction of driving errors
- Gently, but noticeably counter steers in the event of unintentional straying from the marked lane

2
Design

2.1 Hardware

General Overview of System

The ACC system consists of a series of interconnecting components. The method of communication between the different modules is via a serial communication network knows as the Controller Area Network (CAN).

Figure 2 Overview

A radar sensor is usually at the core of the adaptive cruise control (ACC). Installed at the front of the vehicle, the system permanently monitors the road ahead. As long as the road ahead is clear, ACC maintains the speed set by the driver. If the system spots a slower vehicle within its detection range, it gently reduces speed by releasing the accelerator or

actively engaging the brake control system. If the vehicle ahead speeds up or changes lanes, the ACC automatically accelerates to the driver's desired speed.

Standard ACC can be activated from speeds of around 30 km/h (20 mph) upwards and supports the driver, primarily on cross-country journeys or on freeways. The ACC stop & go variant is also active at speeds below 30 km/h (20 mph). It can maintain the set distance to the preceding vehicle even at very low speeds and can decelerate to a complete standstill.

Lane departure warning uses a video camera to detect lane markings ahead of the vehicle and to monitor the vehicle's position in its lane. When the function detects that the vehicle is about to unintentionally move out of the lane, it warns the driver by means of a visual, audible and/or haptic signal, such as steering wheel vibration. These warnings signal the driver that the vehicle is drifting off course, allowing him/her to counter steer accordingly. The function does not issue a warning when the driver activates the turn signal to change.

The camera 's lane detection algorithm records and classifies all common lane markings up to a distance of approximately 60 meters ahead (or up to 100 meters in excellent visibility conditions), whether the road markings are continuous, dashed, white, yellow, red or blue. The camera can even detect Botts' dots (raised highway markers).

Components of System

Component	Data Type
Blind Spot Led Left	Digital Output
Blind Spot Led Right	Digital Output
Front Collision Warning Led on HUD & Sound	Digital Output
Lane Departure warning Led	Digital Output
Motor Control	Digital Output
Lane Departure Haptic Feedback	Analog Output
Steering Angle - Servo Motor	Analog Output
2 x Mid range radar sensor rear(MRR rear)	Digital Input
Mid-Range radar sensor (MRR)	Digital Input
Multi Purpose Camera	Digital Input
Push Button for ON / OFF	Digital Input
Breaking Pedal	Digital Input
Accelerator Pedal	Analog Input
ACC Speed Up	Digital Input
ACC Speed Down	Digital Input
Turn Signal	Digital Input
Lane Assist	Digital Input
Battery Voltage	Analog Input
Servo Position	Analog Input

Table 1 Inputs & Outputs

Blinds Spot Leds - The system informs the driver by illuminating a red warning signal in the glass of the exterior mirror. If the driver fails to see this warning and indicates to change lanes, a warning signal sounds as well. We chose *BA9s LED Bulb* as our component.

- Warning against impending collisions when changing lane
- Reduces the risk of accidents when changing lanes
- Avoids side collisions with vehicles in blind spot
- Prevents accidents due to misjudgment of the speed of approaching vehicles

Front Collison Warning - the system informs the driver by illuminating a red warning signal on the head-up display. If the driver fails to see this warning and indicates to accelerate, a warning signal sounds as well. We chose *Navdy NVD150-WEE Dashboard Head-Up Display* as HUD component.

- Electronic assistants are always vigilant, and, in emergencies, they respond more quickly than people can.
- The systems assist the driver in critical situation and support to avoid accidents

Lane Departure Warnings - Lane departure warning uses a video camera to detect lane markings ahead of the vehicle and to monitor the vehicle's position in its lane. When the function detects that the vehicle is about to unintentionally move out of the lane, it warns the driver by means of a visual HUD and haptic signal, such as steering wheel vibration. These warnings signal the driver that the vehicle is drifting off course, allowing him/her to counter steer accordingly.The function does not issue a warning when the driver activates the turn signal to change lanes or turn intentionally. Lane Departure Warnings are on the same HUD with Front Collison Warning. For haptic signal steering tactile device uses - Nidec Copal Corporation’s haptic device.

- Helps to avoid accidents which are caused by unintentional straying from the marked lane
- Allows to early correct driving mistakes
- Supports to stay in the lane when being inattentive

ACC control Buttons - To turn on/off Adaptive Cruise Control, press the cruise control button on the left side of the steering wheel. To increase / decrease speed of EV.

Accelerator Pedal Input: C To read the steering angle and throttle correctly, we have to use the precise potentiometers. We check the market and find this potentiometer manufactured by VISHAY.[7]

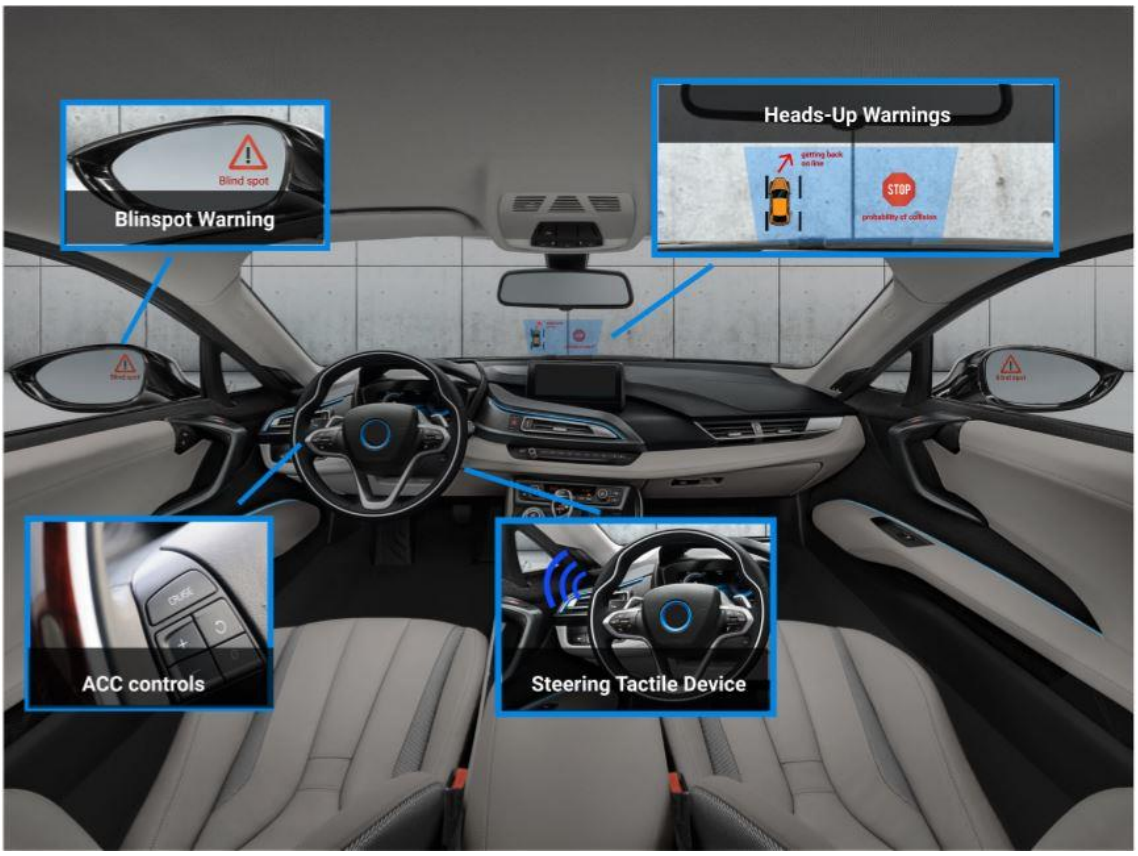


Figure 3 Interior View

Motor Controller - The Rinehart Motion Systems (RMS) AC Traction Controllers are designed for on and off road Electric (EV) or Hybrid Electric (HEV) applications. The PM-Series family is typically the lightest and smallest inverter for its power due to its high heat flux thermal design approach. Used in automotive, commercial vehicles, motorsports and military vehicle traction applications.

Steering Angle Servo Motor + Driver – This is robust industrial servomotor of the brand MiGE model 130ST and as a Motor Driver: AASD50A [6] To drive the steering train, we need a high torque motor. Also, we need a reliable close loop position-controlled motor. We might choose a stepper motor but most of them open loop and this type of motors can be dangerous for our application

Midrange Radar Sensors (Rear) - The lane change assist works by using two mid-range radar sensors that are concealed in the rear bumper – one on the left, one on the right. These two sensors monitor the area alongside and behind the car. These sensors are also gives us blind spot warnings if necessary.

Mid-Range radar sensor (MRR) - The MRR is a bi-static multimodal radar with four independents receive channels and digital beam forming (DBF). These technologies allow the MRR to be configured with independent antennae for different directions, which improves the angular measurement accuracy and means that the radar's field of view can be adjusted depending on the situation.By focusing the main antenna on a narrow main lobe with an opening angle of ±6 degrees, the system is capable of reacting to vehicles in front at long range (up to 160 meters) and performing exceptionally well at higher speeds while also minimizing interference from vehicles in adjacent lanes.

- Digital beam forming (DBF) for flexible antenna use and high accuracy throughout the angular range
- Independent mode for height measurement using an elevation antenna, enabling the system to reliably classify objects and brake safely, even when the object is stationary
- Cost-effective design means that the system can be installed as standard across all vehicle segments
- Self-calibration function reduces fitting costs
- Sensor data fusion in MRR possible without additional hardware

Multi Purpose Camera - During assisted and automated driving, the vehicle must know what is happening in its surroundings at all times. It must reliably detect objects and people, and be able to react to these appropriately. Here, the latest generation of the front video camera from Bosch plays a crucial part: The multi purpose camera for assisted and partially automated driving utilizes an innovative, high-performance system-on-chip (SoC) with a Bosch microprocessor for image-processing algorithms.

- Resilient scene interpretation thanks to artificial intelligence and algorithmic multipath approach
- Specially developed for high-performance driver assistance systems through innovative system-on-chip (SoC)
- Meets the future requirements of Euro NCAP extending up to automated driving (SAE level 2 and higher)

Breaking / Accelerator Pedals - Pressing pedals to stop using the system.

Turn Signal – Using turn signals override the lane assist system.

Battery Voltage Levels – Lets driver to turn on / off ACC according to battery levels.

Servo Position – Reading servo position for controlling lane assist system.

PLC - PLC selection is generally based on the controller's necessary inputs, outputs and features. The first choice is the controller sort; rack, mini, micro, or based on software.

Because we are using too many digital inputs and outputs, we need a to many input and out port. Also because of the to many components and provide more space to user, PLC has to be smaller. In this case, the Panasonic AFP0RF32CP is perfect solution for us. 16 digital input and 16 digital output with the compact 25mm width, 90mm height and 60 mm long body. But every good thing comes at a price, the PLC has not analog pins to use. But we can use 3 extension for this PLC to overcome the problem. Thus, we choose the Panasonic AFP0RA21 with 2 analog input and 1 analog output. Related to our needs, we will use 2 of them

1 x AFP0RF32CP:

Manufacturer	PANASONIC(AUTOMATION)
Mounting Type	DIN RAIL
Voltage - Supply	24VDC
Number of Inputs and Type	16 - Digital
Expandable	3 Modules
Communications OPEN	RS-232C
Memory Size	32K Words
Number of Outputs and Type	16 - Pulse (4), PWM (4), Solid State (8)

Figure 4 PLC-1

2 x AFP0RA21:

Manufacturer	PANASONIC(AUTOMATION)
Mounting Type	DIN RAIL
Number of Inputs and Type	2 - Analog
Memory Size	32K Words
Number of Outputs and Type	1 - Analog

Figure 5 PLC-Modules

PLC wiring scheme:

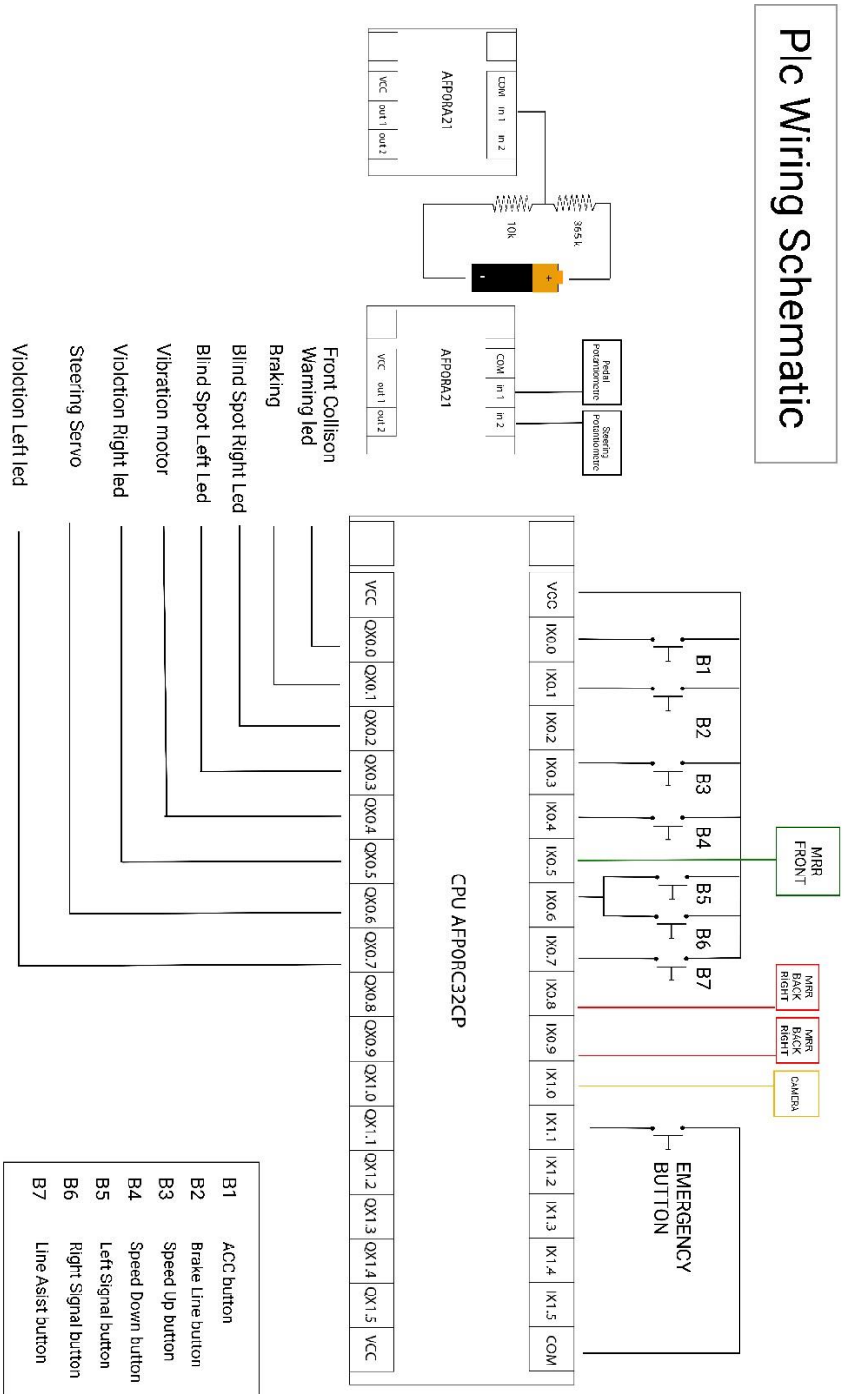


Figure 6 wiring scheme

2.2 Software

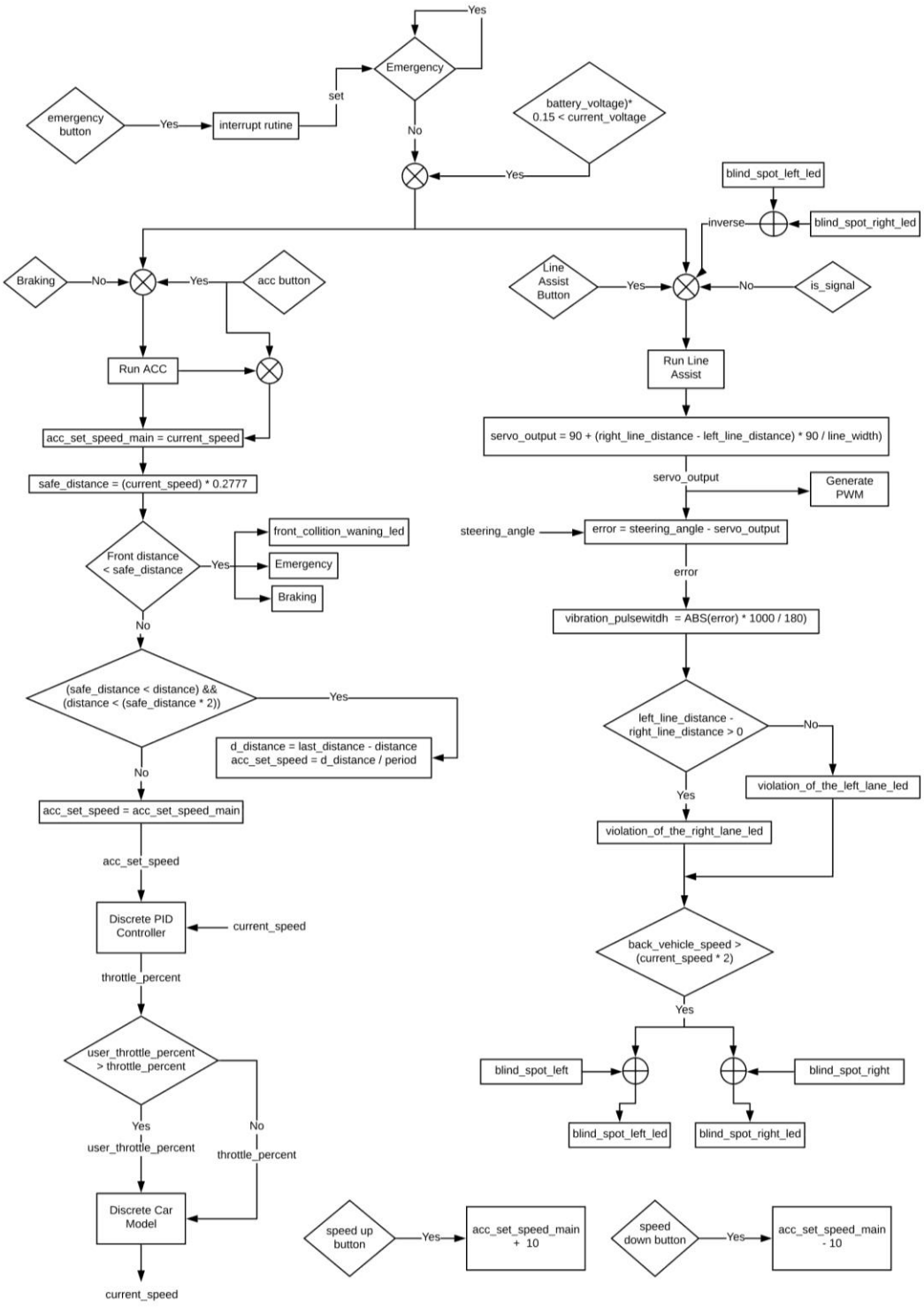


Figure 7 Flowchart

Car Model

Before the design controller, we need a plant that basic and as realistic as possible. Our first attempt is model our very own plant using basic newtons law and simplified shape. However, it takes more time than we assume and because of the simplifications the plant was not realistic enough. This failure pushes us to research accurate model for our application. Luckily in MATLAB's documentation we can be able to find an example about car cruise control [1]. Inside the SIMULINK schematic shown as Figure 4, the plant is located right down.

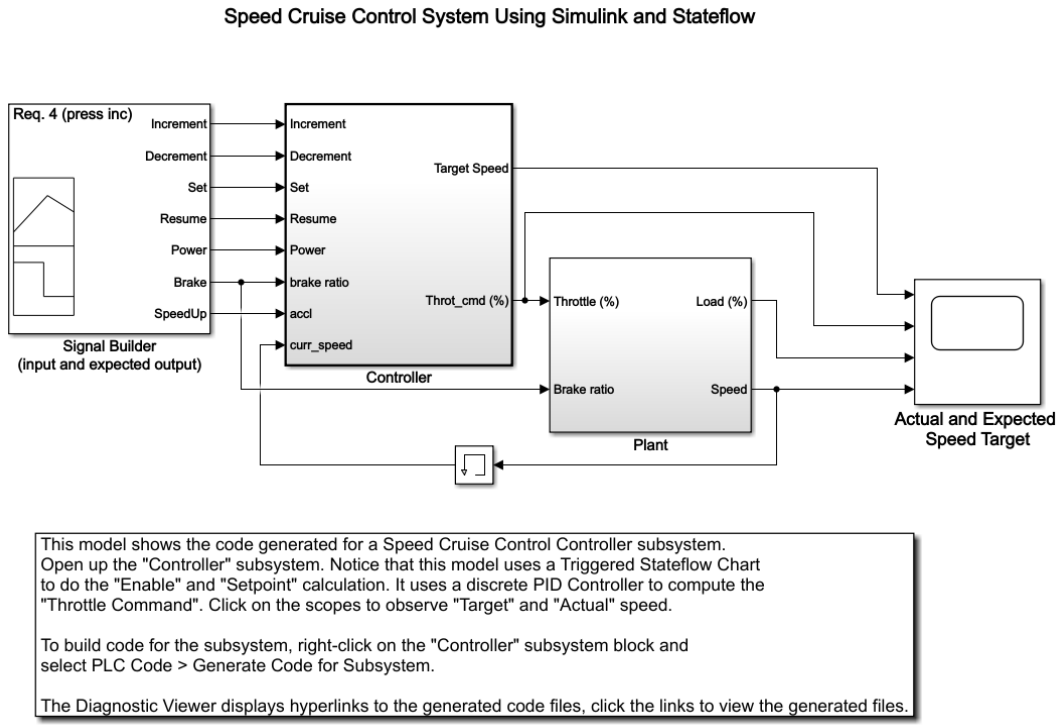


Figure 8 Speed Cruise Control System

The best part is plant takes the pedal pressing percentage, braking and road disturbances as input which is what we are looking for. Inside the plant model shown in Figure 5 there is another subsystem which called plant dynamics. Inside the plant dynamics shown in Figure 6, we can see the model itself. There is a sum block which sum up all the inputs as percent, which is great because we can feed the car model using only one controller output through one input. The output unit of the model is m/s.

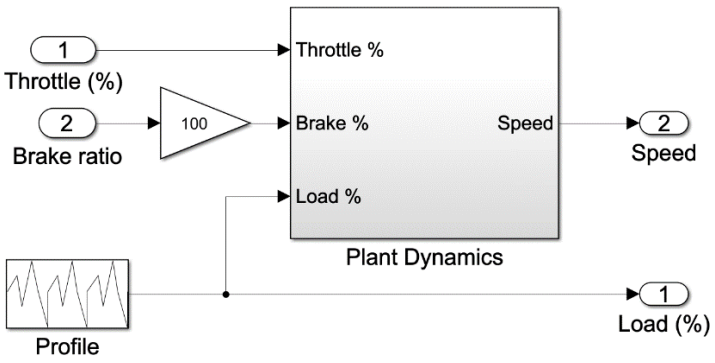


Figure 9 Plant Dynamics

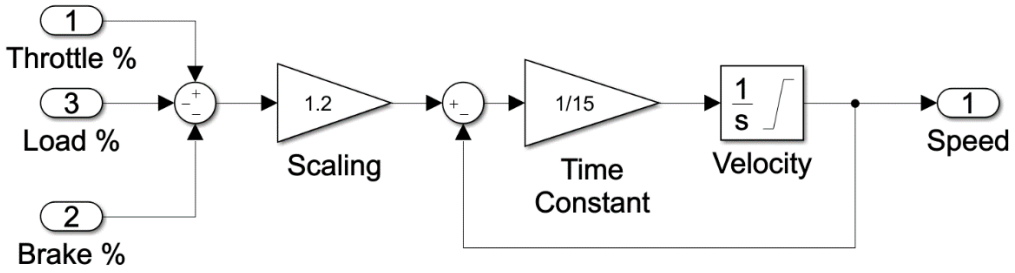


Figure 10 Model

Generating Plant Model Using MATLAB Car Model

We will use the MATAB model to calculate the car model as transfer function. We want to use plant model as transfer function because with that function we can use the MATLAB tools for estimate the PID constants and we can import our model directly to the PLC software to make a simulation.

First off all we have to simplify the system using block reduction method as bellow.

$$\frac{1}{15} \rightarrow \frac{1}{s} = \frac{0.0666}{s}$$

There is unit feedback loop,

$$\frac{G}{1+G} \rightarrow \frac{\frac{0.0666}{s}}{1+\frac{0.0666}{s}} = \frac{0.0666}{s} * \frac{s}{s+0.666} = \frac{0.0666}{s+0.666}$$

There is a scale factor as a gain

$$1.2 \rightarrow \frac{0.0666}{s+0.666} = \frac{0.07992}{s+0.666}$$

Now we will use the MATLAB command bellow to convert this continuous time function to discrete time function, another word, s plane to z plane.

```
num = [0.07992];
denum = [1 0.06666];
T = tf(num,denum);
T_Discrete=c2d(T,0.01)
```

As an output we have a z transform of the model as

$$T\_Discrete = \frac{0.0007989}{z - 0.9993}$$

To make it useable in the PLC software we can use the method bellow

$$\frac{output}{input} = \frac{u[t]}{e[t]} = \frac{0.0007989}{z - 0.9993} = \frac{0.0007989\, z^{-1}}{1 - 0.9993\, z^{-1}}$$

$$u[t] - u[t]\, 0.9993\, z^{-1} = e[t]\, 0.0007989\, z^{-1}$$

$$u[t] = u[t]\, 0.9993\, z^{-1} + e[t]\, 0.0007989\, z^{-1}$$

$$u[t] = u[t-1]\, 0.9993 + e[t-1]\, 0.0007989$$

Now we have a discrete time model of the plant.

Designing the Controller

Calculating PID Parameters

Because we have our plant as a transfer function, we can use the MATLAB auto tune tool for calculate the P, I and D constants. We will make a system as Figure 7. Inside the PID block there is a Tune button down which stands for tune the output. We have already tuned the system as need and the PID values are calculated as bellow.

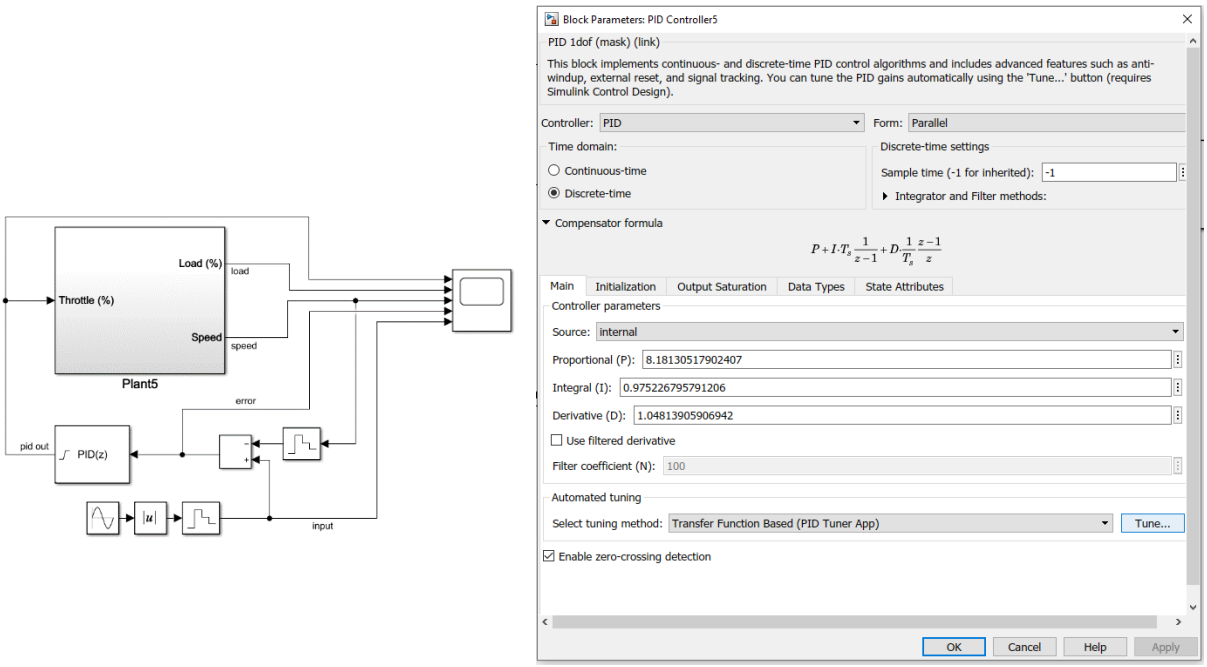


Figure 11 System Variables

Discrete Time PID Controller

We calculate the plant and controller parameters so far. Now we will create a discreate time controller for our PLC software. Our cycle frequency is 100 Hz, related to the frequency we can calculate the discrete time PID using Z transform as below [2].

$$\frac{u[z]}{e[z]} = \frac{\left(K_p + K_i\frac{Ts}{2} + \frac{K_d}{Ts}\right)z^2 + \left(-K_p + K_i\frac{Ts}{2} - \frac{2\,K_d}{Ts}\right)z + \frac{K_d}{Ts}}{z^2 - z}$$

$$\frac{u[z]}{e[z]} = \frac{\left(K_p + K_i\frac{Ts}{2} + \frac{K_d}{Ts}\right) + \left(-K_p + K_i\frac{Ts}{2} - \frac{2\,K_d}{Ts}\right)z^{-1} + \frac{K_d}{Ts}z^{-2}}{1 - z^{-1}}$$

$$u[z] = z^{-1}u[z] + a\,e[z] + b\,z^{-1}e[z] + c\,z^{-2}e[z]$$

$$u[t] = u[t-1] + a\,e[z] + b\,e[t-1] + c\,e[t-2]$$

$$a = \left(K_p + K_i\frac{Ts}{2} + \frac{K_d}{Ts}\right)$$

$$b = \left(-K_p + K_i\frac{Ts}{2} - \frac{2\,K_d}{Ts}\right)$$

$$c = \frac{K_d}{Ts}$$

Testing the Controller and Models in MATLAB

We will use the calculated controller functions to in MATLAB. First, we will use the MATLAB Function block as shown in below.

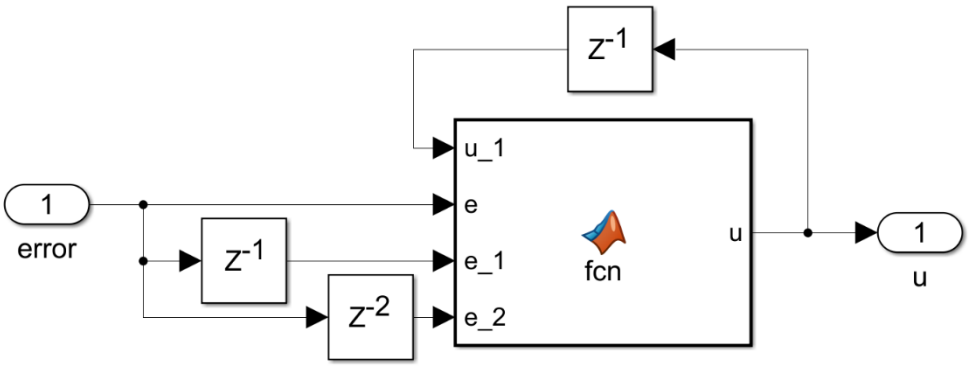


Figure 12 Function Block

Inside the function, the code will be settled as following.

```
function u = fcn(u_1,e,e_1,e_2)
%% variables
Ts = 0.01;
Kp = 8.18;
Ki = 2;
Kd = 1.04;

a = Kp + Ki*(Ts/2) + Kd*(1/Ts);
b = -Kp + Ki*(Ts/2) - Kd*(2/Ts);
c = Kd/Ts;

%% output
out = u_1 + a*e + b*e_1 + c*e_2;

if(out>100)
    out=100;
end

if(out<-100)
    out = -100;
end

u = out;
```

The output of the PID controller is limited to -100 to 100 because of the controller output is the throttle position as a percent. The negative values are representing the braking pedal position. We will use that technique for the simulate the system more easily in the MATLAB and PLC software. It is not the perfect way or may be the less unrealistic way of testing but because we do not have a change to test the system without the proper input and outputs, we will assume the system input and outputs like that. So far, the test results are looks acceptable good and realistic. Test diagrams and results are shown in Figure 9 and Figure 10, respectively.

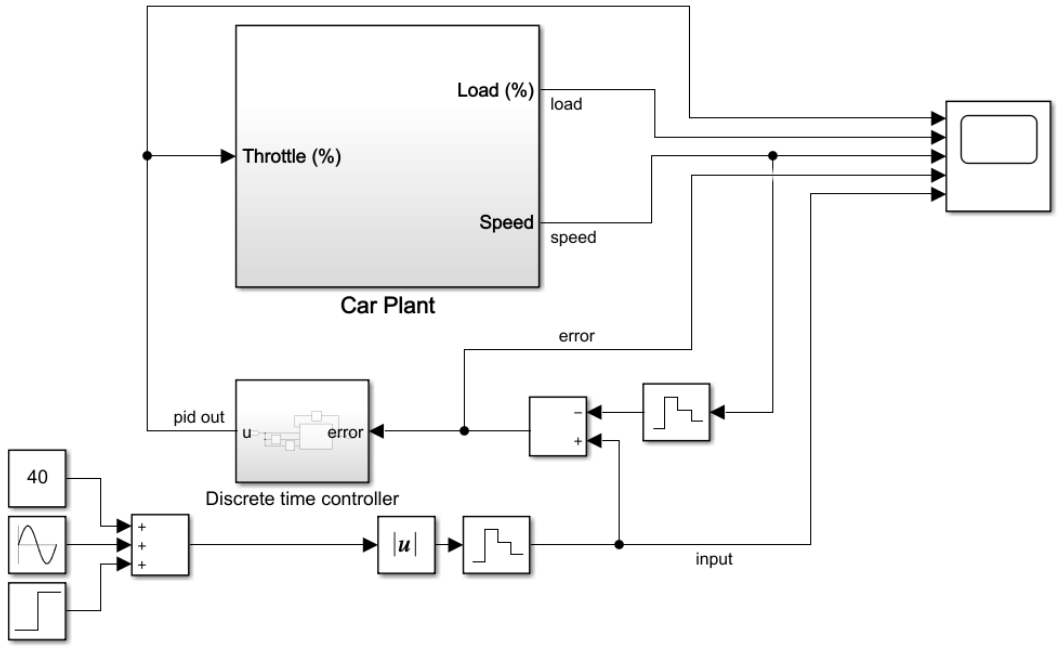


Figure 13 Test Diagrams

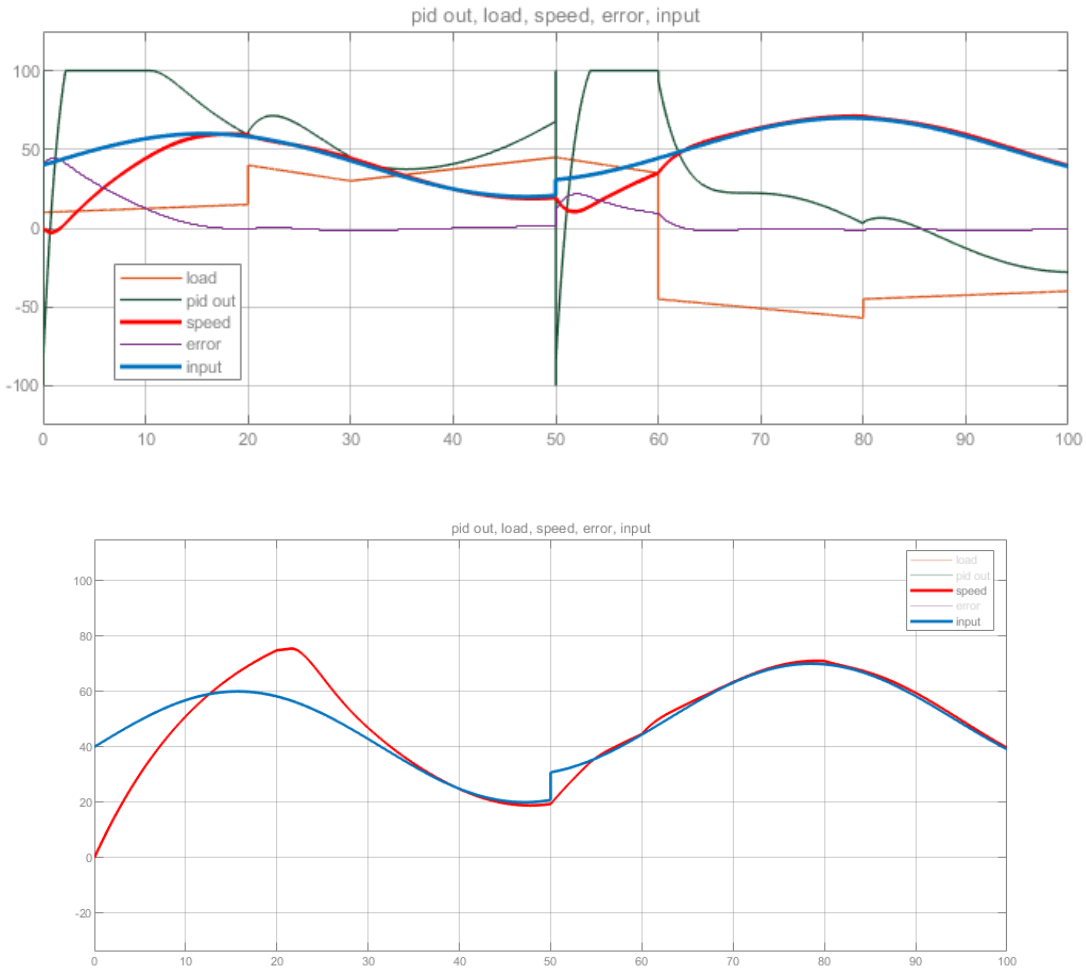


Figure 14 Results

Importing Discrete Time PID Controller and Plant to PLC

Our controller and plant are working with 100 Hz. So, we have to use the constant cycle time in PLC. To achieve this, we make a simple pulse generator using TON and rising edge trigger. TON pre-set time is defined as 10ms. And every time the output is rising the block is reset by reset_acc_timer variable and the cycle start again. With the help of the rising edge trigger we produce the pulse for acc_pid1 and acc_car_model1 blocks.

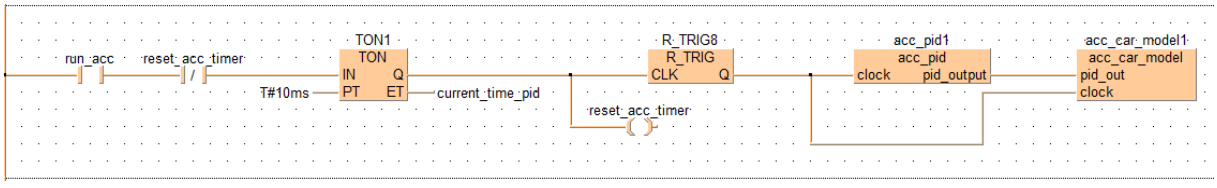


Figure 15 ACC reset

acc_pid_1 Block

We defined all the constants and initial values. All functions are directly transferred from MATLAB workspace to FPWINPRO. Same variable types are redefined related to the program.

	Class	Identifier	Type	Initial	Comment
0	VAR_INPUT	clock	BOOL	FALSE	
1	VAR_EXTERNAL	acc_error	ARRAY [0..2] OF REAL	[3(0)]	
2	VAR_EXTERNAL	acc_set_speed_main	INT	100	
3	VAR_OUTPUT	pid_output	REAL	0.0	
4	VAR	error	REAL	0.0	
5	VAR	Ts	REAL	0.01	
6	VAR	Kp	REAL	8.18	
7	VAR	Ki	REAL	0.97	
8	VAR	Kd	REAL	1.04	
9	VAR_EXTERNAL	acc_pid_out_last	REAL	0	
10	VAR	a	REAL	0	
11	VAR	b	REAL	0	
12	VAR	c	REAL	0	
13	VAR_EXTERNAL	pid_out	REAL	0	
14	VAR_EXTERNAL	current_speed_mps	REAL	27.7	

Figure 16 Variables

```
if (clock) then

          error := (INT_TO_REAL(acc_set_speed_main) * 0.2777) - current_speed_mps;

          acc_error[0] := error;

          a := Kp + Ki*(Ts/2) + Kd*(1/Ts);
          b := -Kp + Ki*(Ts/2) - Kd*(2/Ts);
          c := Kd/Ts;

          pid_output := acc_pid_out_last + a*acc_error[0] + b*acc_error[1] +
c*acc_error[2];

          acc_pid_out_last := pid_output;

          acc_error[2] := acc_error[1];
          acc_error[1] := acc_error[0];

          if (pid_output > 100) then
                    pid_output := 100;
          end_if;

          if (pid_output < -100) then
                    pid_output := -100;
          end_if;

          pid_out := pid_output;
end_if;
```

acc_car_mdel1 Block

This block not important for the software but we have to put the model into FPWINPRO to simulate the system. Again, this model is directly transferred from MATLAB workspace to FPWINPRO. Code behind and variables shown in below.

	Class	Identifier	Type	Initial	Comment
0	VAR_INPUT	pid_out	REAL	0	
1	VAR_EXTERNAL	acc_car_model_last_speed	REAL	27.7	
2	VAR_EXTERNAL	current_speed	INT	100	
3	VAR_INPUT	clock	BOOL	FALSE	
4	VAR_EXTERNAL	current_speed_mps	REAL	27.7	

Figure 17 car model variables

```
if (clock) then

          current_speed_mps := (0.9993 * acc_car_model_last_speed) + (pid_out *
0.0007989);
          acc_car_model_last_speed := current_speed_mps;

          current_speed := REAL_TO_INT(current_speed_mps * 3.601);
end_if;
```

3
Result

Simulation

Thanks to the simulation part, possible situations can be easily tested by simulating whether the system is working properly. No action is taken here. This section is only for controlling the input and output of data. It will be safer to deactivate this part while adapting to the real system. We use this section as follows:

1. Our current_voltage_read input is an analog input, since it is the values read from the sensors, it is entered manually for simulation, it takes a value between 0 and 4096. We can see the output value between 0 and 400V from the current_voltage output.
2. Our potentiometer_voltage_read input is an analog input, since it is actually the values read from the sensors, it is entered manually for simulation, it takes a value between 0 and 4096. We can see the output from the steering_angle output as a value between 0-180 degrees. If the angle is less than 90 degrees, the steering wheel is turned to the left and if it is greater than 90 degrees, the wheel is turned to the right.
3. Our pedal_voltage_read input is an analog input, since it is the values read from the sensors, it is entered manually for simulation, it takes a value between 0 and 4096. We can see the output value as a value between 0 and 100 from the user_throttle_percent output. With this value, we can see how much the user presses the gas pedal.
4. Current_speed is set once while the program starts, then it changes momentarily with speed_up_btn, speed_down_btn or when the user presses the accelerator pedal.
5. Our front_distance input allows us to manually enter the data to be received from the sensors for simulation, we can manually enter the distance of the next vehicle and see the reaction of the system.

6. Our righ_line_distance and left_line_distance inputs are for lane tracking system. With the difference of distance from the left lane and the right lane from the sensors while the system is activated, whichever side is near, the warning leds on that side (Y4 or Y5) and try to align the vehicle in the middle of the lane.
7. With our back_vechile _speed input, the speed data of the rear vehicle comes from the sensors. For simulation, this information is entered manually, and the system's response is observed according to the entered value. (For example, if the speed of the vehicle in the rear is more than 2 times the current speed of our vehicle, the Lane Tracking System (run_line_assistant) is closed.)
8. When we press the acc_btn entry, if the required values in the system's definition section are provided, our ACC system works, we can see that it works with our run_acc output. Our current_speed value is assigned to the value of set_speed_main, the vehicle goes at constant speed.
9. When we press line_assistant_btn, our voltage value is more than 15% and if there is no emergency, our Lane Tracking System works, we can see this with run_lise_assistant.
10. Our Is_braking login shows whether the user is braking or not, while not pressed. If we activate our interference, the system will detect it as braking, our Y1 output will be active and the ACC system will shut down.
11. Our blind_spot_left and blind_spot_right inputs check whether there are any vehicles in the blind spot of the vehicle, that is, the parts that the user cannot see. For simulation, these values are entered manually, with or without. If we activate the blind spot inputs in terms of the vehicle, we can see that the leds (Y2 or Y3) on their outputs are lit and the Lane Tracking System (run_line_assistant) is turned off. In case we deactivate blind spot inputs, our Lane Tracking System (run_line_assistant) is activated again.
12. When the speed_up_btn entry is pressed, our set_speed_main value increases by 10, the vehicle accelerates until the current_speed reaches the set_speed_main value.
13. When the speed_down_btn entry is pressed, our set_speed_main value decreases by 10, the vehicle slows down until the current_speed reaches the set_speed_main value.

14. Our is_signal input checks whether the vehicle is signaling left or right. If the user gives a signal, our Lane Tracking System (run_line_assistant) will shut down, when the user turns off the signal, our Lane Tracking System (run_line_assistant) will be active again.

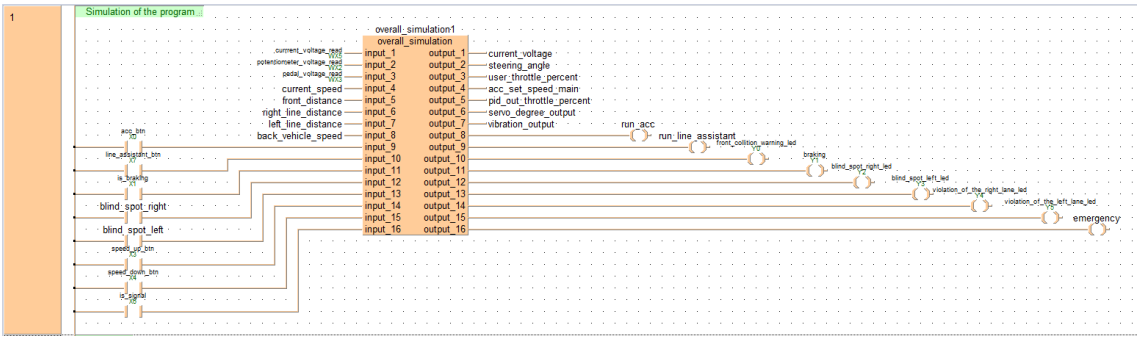


Figure 18 Simulation of Program

Emergency

In this section, an emergency button has been added to the PLC for use in emergencies. This button can be triggered by both external cutting. Thanks to the interruption, the program flowing from top to bottom will run at any stage of the emergency. At the same time, the emergency variable can activate the safety parameters that are continuously calculated within the program. For example, the front collision warning system can directly change the emergency variable.

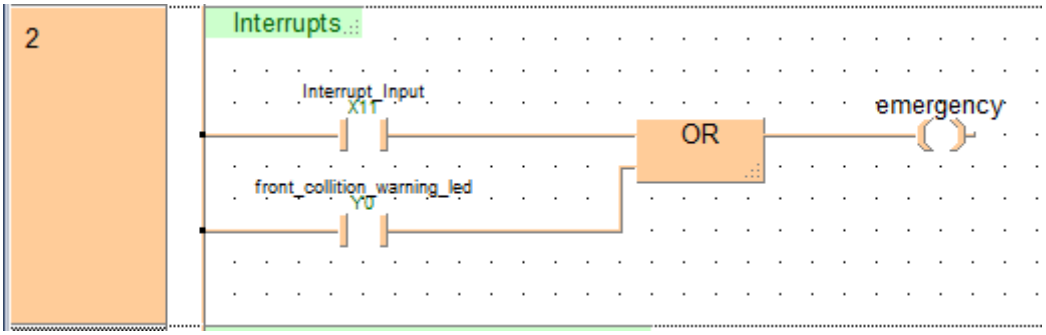


Figure 19 Interrupts

Initialize the ACC System

This section controls the conditions required for the system to operate. If the battery voltage is more than 15 percent and the user presses the start button while not braking, the system starts. It is enough for the user to press the brake to exit the adaptive speed controller. If the system is turned off by the user or software, the button must be pressed again to activate it.

In the algorithm specified in the figure, is_braking (X1) is a digital input for PLC and is connected to the brake pedal. When the user presses the brake, the braking (Y1) system is activated and the vehicle is braked. Acc_btn (X0) is connected to the digital input in the PLC and when the button is pressed, the adaptive speed control system is activated if the conditions are suitable. The run variable is the variable that indicates whether the system is running. The voltage input is read in the 2nd part of the PLC Ladder diagram and enters the start POU as integer. When this value is converted to real in the start POU, our battery_safe variable, which indicates the safety status of the battery, is set to true if the full battery voltage value is greater than 15 percent.

Start_acc POU

Code behind

```
if (INT_TO_REAL(battery_voltage)* 0.15 <
INT_TO_REAL(current_voltage)) then
battery_safe := true;
else
     battery_safe := false;
End_if;
output := braking AND acc_btn AND battery_safe;
```

Initialize the Line Assistant System

Here, it is ensured that the lane tracking system is ready for operation by providing safety parameters such as the battery voltage is more than 15 percent and the emergency variable is inactive.

Start_Line_assistant

Code behind

```
if (INT_TO_REAL(battery_voltage)* 0.15 <
INT_TO_REAL(current_voltage)) then
battery_safe := true;
else
     battery_safe := false;
end_if;

output := condition_1 AND btn AND battery_safe;
```

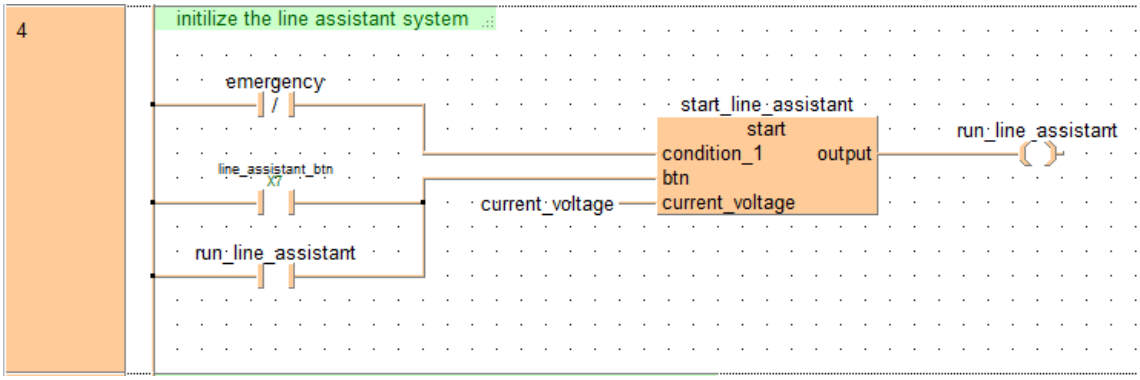


Figure 20 Line Assistant system

Analog Inputs

In this section, the program will calculate the needed variables related to the voltage read from the analog input ports. Program can read 3 analog inputs which are the voltage of the car battery, steering wheel position as angle and the throttle pedal position as percent. To achieve the reading values correctly, 3 POU makes calculations related to the input and output ranges as below.

POU's and Code Behind

Calculate_voltage1:

```
hesap := read_value * INT_TO_REAL(battery_voltage) / const;
output := REAL_TO_INT(hesap);
```

Calculate_steering_angle1:

```
steering_angle := REAL_TO_INT(input * 18);
```

Claculate_pedal_percent1:

```
throttle_percent := analog_read * 10;
```

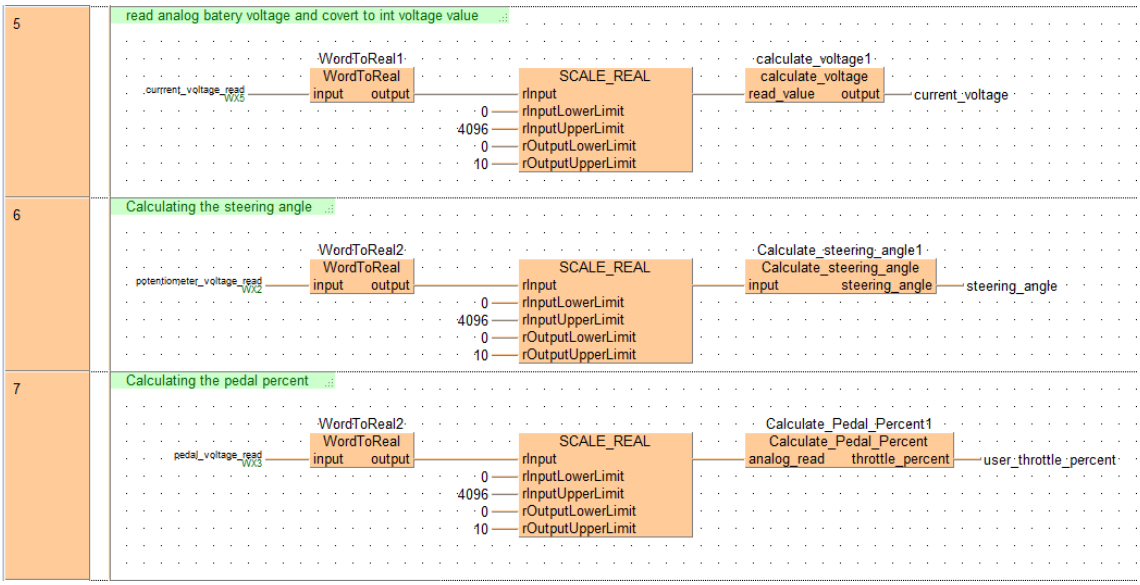


Figure 21 Analog Inputs

Reading the Digital Sensor Values via CAN BUS

We are using the two Mid Range MRR sensors for back , one Long Range MRR sensor and line assist camera sensor for front which are manufactured by BOSCH company. Related to the datasheet of the sensors, sensors give the data they read or calculate to the processor via serial communication. In serial communication data is transferred through master to slave using high and low states of the voltage. We are using the rising edge and falling edge detections to read corresponding variables related to the manufacturer datasheet. In our project because we could not reach the datasheets, we will keep the POU's empty and use the variables in the simulation block to simulate the values that we should have to read. The importing thing is we are actually calculating all the scenarios and all the parameters with our function block as if the data came, so the only thing left is reading the values via bus.

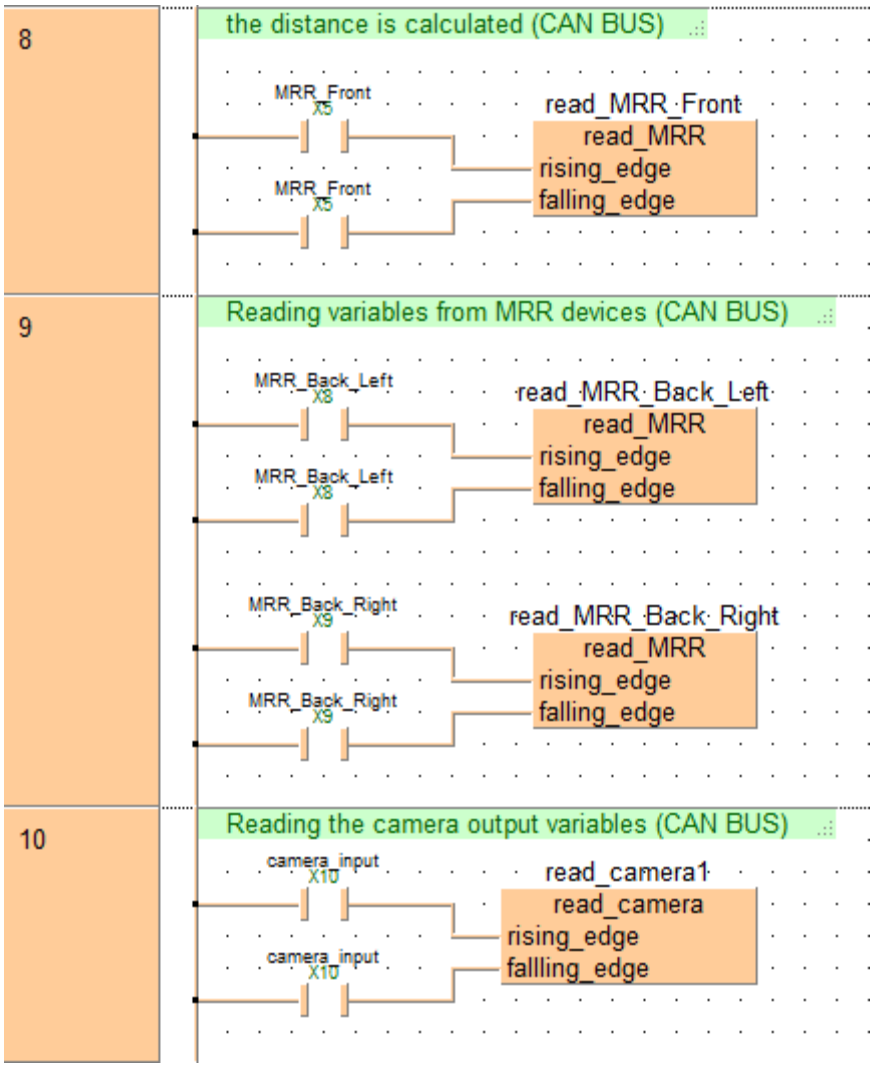


Figure 22 CAN BUS

Update Set Speed

By pressing the start button, the instantaneous vehicle speed is assigned to the system as a reference. At the same time, the reference speed value can be updated through the buttons on the steering wheel, without pressing the gas or the brake.

In this section, speed_up_btn (X3) and speed_down_button (X4) are connected to the digital inputs of the PLC. When these keys are pressed, the output of the rising edge function enters our update_set_speed function. Here, if the input_up signal is received, the fixed speed value increases by 10. In input_down input, same situation decreases fixed

speed by 10. For the sake of safety, in this case, if the speed is less than 20 kmh, the system turns itself off because the speed cannot get negative values.

Update Set Speed POU

```
if (input_up) then
acc_set_speed_main := acc_set_speed_main + 10;
end_if;

if (input_down) then
acc_set_speed_main := acc_set_speed_main - 10;
end_if;

if (acc_set_speed_main < 20) then
run = false;
end_if;
```

set_acc_speed POU

```
if (input) then
acc_set_speed_main := current_speed;
acc_set_speed := current_speed;
end_if;
```

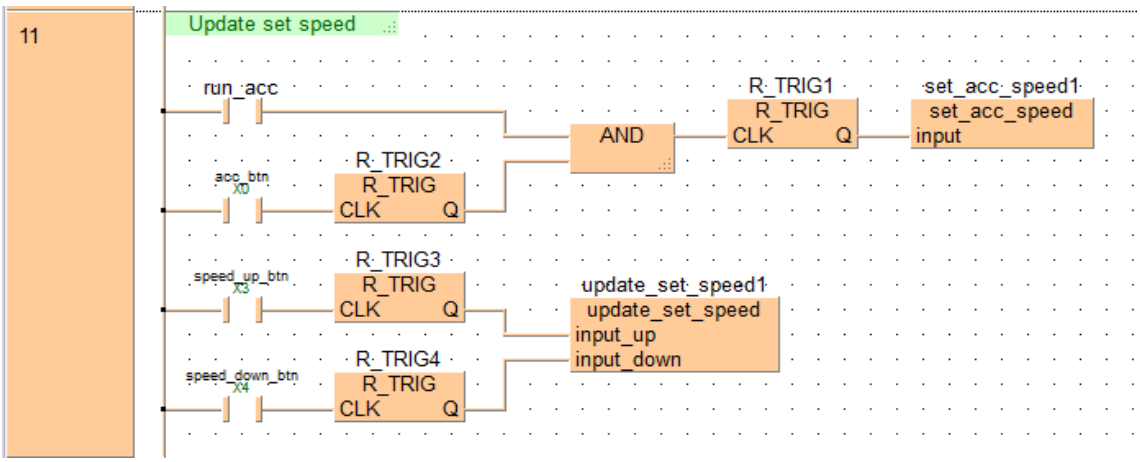


Figure 23 Update/Set Speed

Front Collition

This section calculates the speed of the vehicles in front by looking at the distance data from the sensor in front of the vehicle and synchronizes the speed of the vehicle to the speed of the vehicle in front or stops the vehicle by warning the driver in dangerous situations.

adaptive_control POU

If the vehicle in front is 2 seconds or more from us according to our speed, the system will not give any warning. If the distance between us and the vehicle in front is between 1-2 seconds, the system synchronizes the speed of the vehicle in front to our speed. In this way, the safe distance determined between us and the vehicle in front is protected. When the vehicle in front leaves the potentially dangerous distance, the system returns to the speed at which the reference was received.

```
 safe_distance := INT_TO_REAL(current_speed) * 0.2777; (* safe
distance is distance that the car

traveled distance(meter) after 1 second *)

if (distance < safe_distance) then
front_collition_warning_led := true;
emergency := true;
braking := true;

else
front_collition_warning_led := false;
emergency := false;
braking := false;

end_if;

if (safe_distance < distance AND distance < (safe_distance * 2))
then
d_distance := last_distance - distance;
acc_set_speed := REAL_TO_INT(d_distance / frequency);
```

```
else
      acc_set_speed := acc_set_speed_main;
end_if;

last_distance := distance;
```

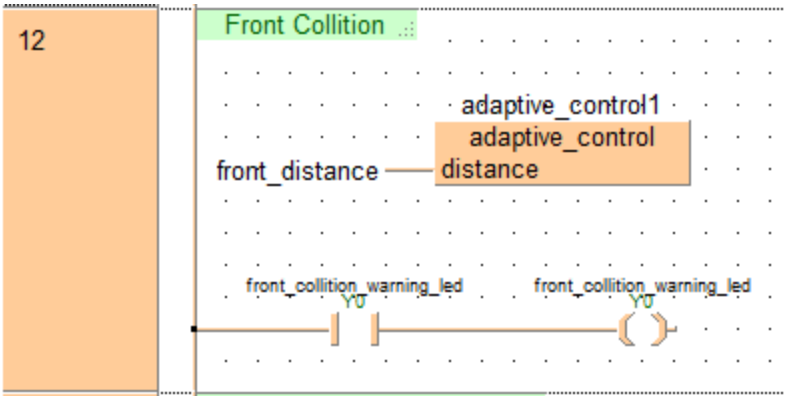


Figure 24 Front Collision

PID Control

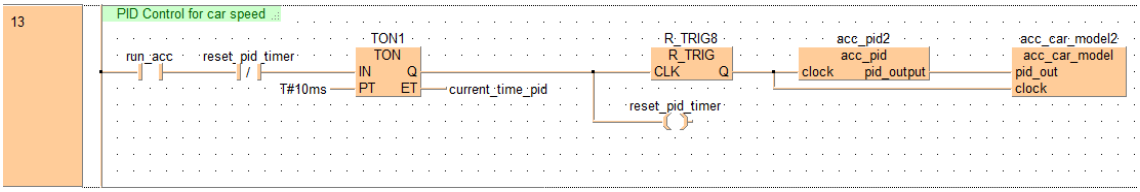


Figure 25 PID Control

Keeping Vechile on middle of the line

In this part, the main target is to keep the vehicle in the middle of the lane. For this, a POU named keep_in_line was created and necessary calculations and operations were done in it. The transactions performed are as follows:

1. First of all, the function block has 3 inputs: left_line_distance, right_line_distance and steering_angle, 4 outputs, servo_degree_output, vibration_output, violation_of_the_left_lane_led and violation_of_the_right_lane_led.
2. This block takes the difference of data from the left and right sensors.

3. According to this difference, whichever side is closer to the vehicle lights the warning leds belonging to that side.
4. It multiplies this difference by the specified number of times and adds the servo_output.
5. The difference of servo_output obtained with one of the inputs steering_angle is returned to us as an error.
6. This received error value is multiplied by a specified number of times and the vibration level to be given to the steering wheel is determined.

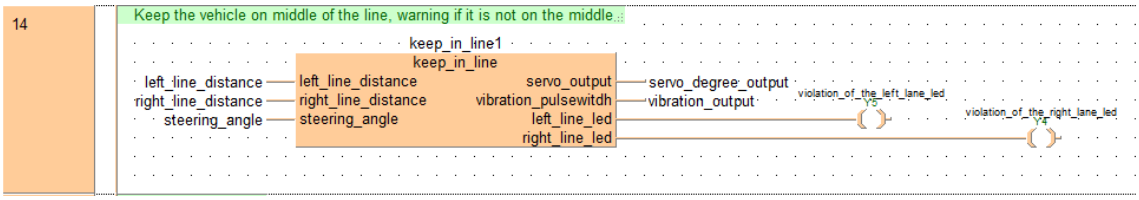


Figure 26 Middle Line Assist

Blind Spot Leds

Sensors are located in the blind spots of the vehicle to indicate whether there is any danger to the driver's invisible parts of the vehicle. If a warning comes from our blind_spot_left or blind_spot_right inputs that constitute a dangerous situation in the blind spots, our Lane Tracking System (run_line_assistant) will shut down and the system will be activated again when the warning is removed.

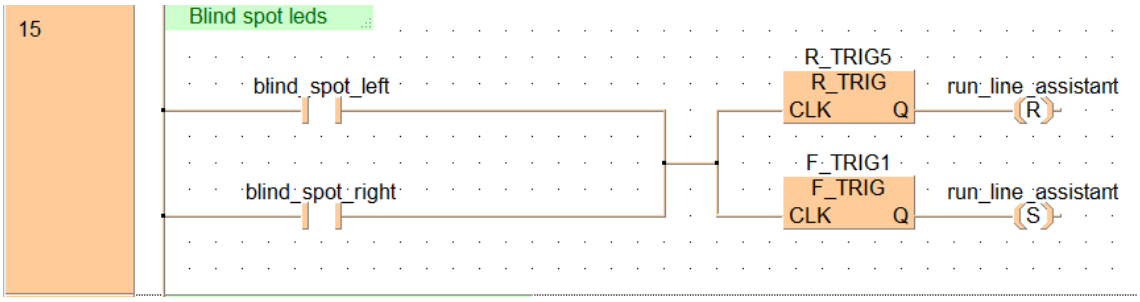


Figure 27 Blind Spots

Checking Signals

In this section, it is checked whether the user signals to turn left or right. If the user signaled, the is_signal (X6) input becomes active and run_line_assistant closes, when the user stops signaling, run_line_assistant is activated again.

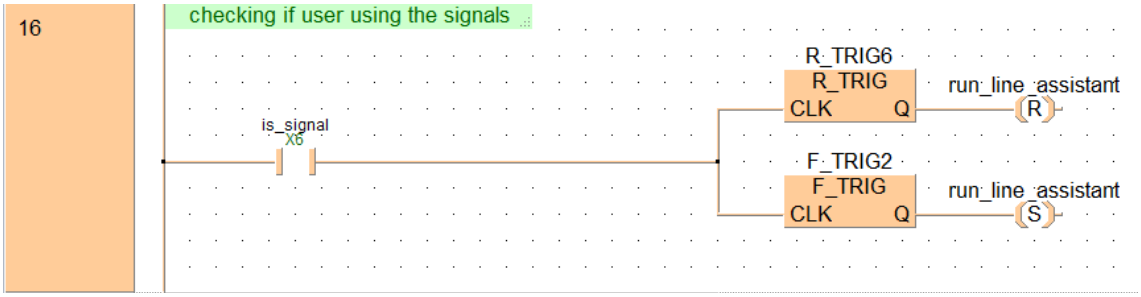


Figure 28 Turn Signals

Blind spot and Back Collision Warning Leds

Here, blind_spot_right_led (Y2) and blind_spot_left_led (Y3) are burned to show the user whether there is a car on the right or left blind spot according to the data received from the MRR sensor. At the same time, if a vehicle comes from behind very quickly, the driver is warned against this situation by burning the Y2 and Y3 outputs at the same time, despite the possibility that this vehicle passes to the right or left of the vehicle that the user is driving. In addition to the leds being burned, while the leds are burning, which is a dangerous situation, the lane tracking assistant is closed to give the driver the ability to maneuver. The assistant is automatically reactivated when the dangerous situation is over.

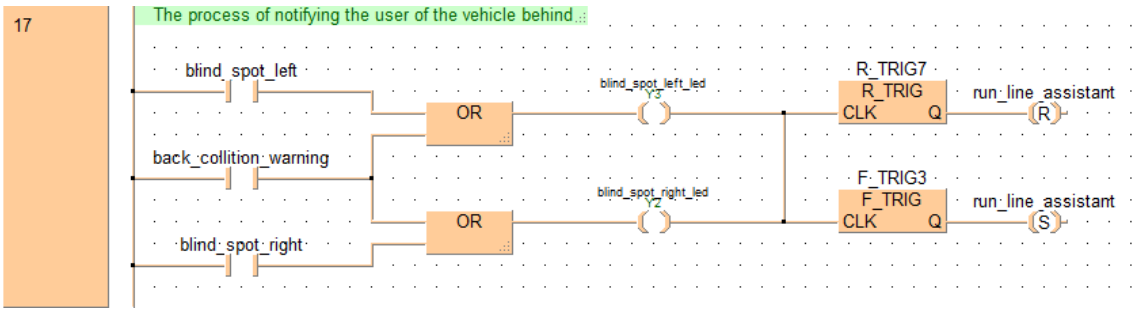


Figure 29 Back Collision and Blind Spots

Back Collision Detect

In this section, with the help of the radar sensor on the back of our vehicle, we take the speed data of the vehicle behind our vehicle and process it in our back_collition_detect1 block. If the speed of the vehicle behind us is more than twice the current speed of our vehicle, our Lane Tracking System will shut down.

Back_collition_detect POU

```
if (vehicle_speed > (current_speed * 2)) then
back_collition_warning := true;
else
     back_collition_warning := false;
end_if;
```

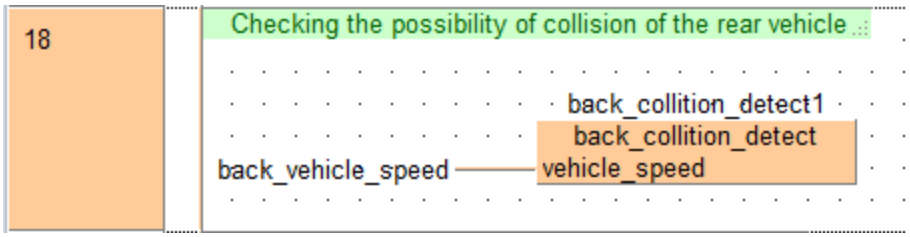


Figure 30 Back Collision Detection

PWM output

To drive the steering servo, we are using the pulse width modulated signals. This signals most generally run with 400 Hz which means every 20ms the driving command is sended to the servo driver. PWM signals are controlled by the pulse width from 1ms to 2ms. If 1ms high signal and then 19 ms low signal applied, servo driver drive servo to 0 degree, if 2ms high signal and then 18 ms low signal applied, servo driver drive servo to 180 degree. In our application the servo is directly connected to the steering wheel which means the 90 degree is the straight steering wheel and lower than 90 degree for left turn, higher than 90 degree for right turn.

Also, to drive the vibration motor on the steering wheel to warn the driver, we will use the pwm method. More the pulse width, more vibration is applied to the steering wheel. This vibration power is calculated by keep_in_line1 POU.

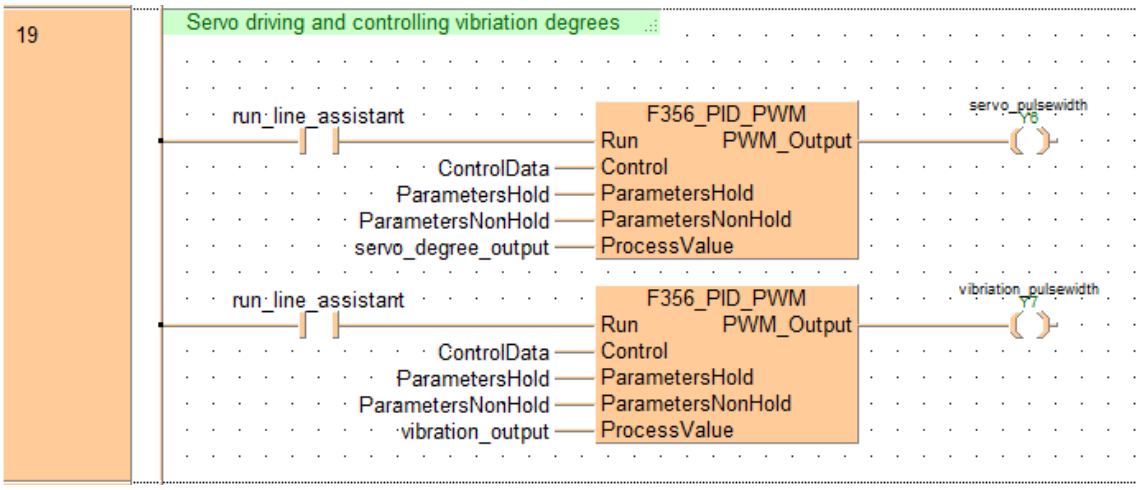
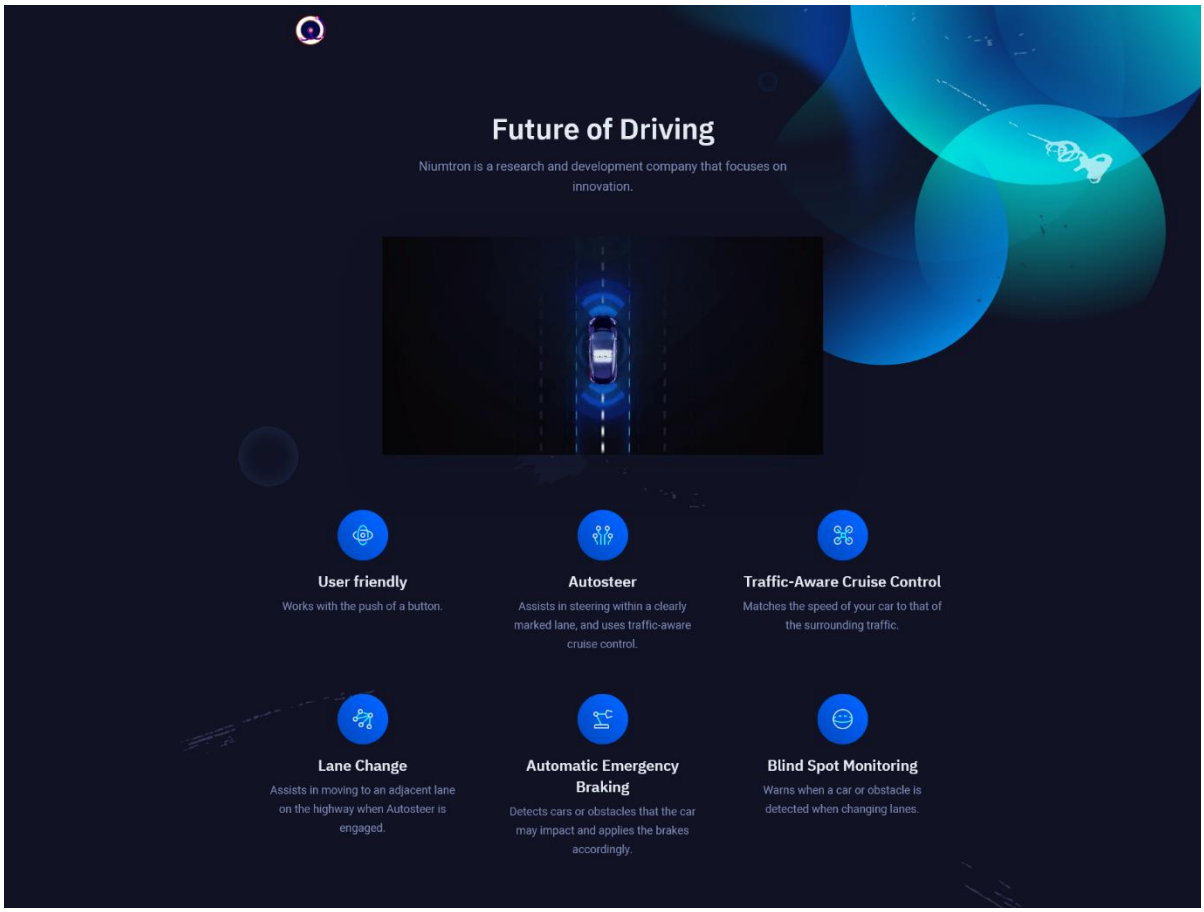


Figure 31 Servo Driving

Conclusion

Thus, we have seen how adaptive cruise control system differs from conventional cruise control system. We have also seen the latest modifications and additional functions for an ACC system. It shows how safety and ease of driving can be achieved using ACC.

For more information you can visit our website at **niumtron.com**



4
References

[1] https://nl.mathworks.com/help/plccoder/examples/speed-cruise-control-system-using-simulink-and-stateflow.html

[2] http://portal.ku.edu.tr/~cbasdogan/Courses/Robotics/projects/Discrete_PID.pdf

[3] Adaptive Cruise Control - Towards a Safer Driving Experience by Rohan Kumar, Rajan Pathak

[4] *"Adaptive Cruise Control System Overview"*- 5th Meeting of the U.S. Software System Safety

[5] Marsden, G. & Mcdonald, M. & Brackstone, Mark. (2001). Towards an understanding of adaptive cruise control. Transportation Research Part C: Emerging Technologies

[6] AC Motor Matched Servo Motor Driver

[7] https://www.vishay.com/docs/51034/p13.pdf

[8] Bosch Mobility Solutions