Table of Contents

1. Introduction to Problem Statement

The automotive subsystem that has been chosen in this report is 'Electronic Brake Force Distribution' or EBD in simpler terms. EBD is used along with Anti-lock Brake System (ABS) and Electronic Stability Control (ESC). Since EBD in itself is a complex system, this report and all the firmware will be focusing only on EBD.

An attempt has been made to provide a simple prototype of EBD and to demonstrate its working. The operating procedure to be followed can be summarized as follows:

1. The requirement specifications of an EBD system have been written. A few hours have been spent on brainstorming about how and what functions does the EBD have. These are the high level requirements.
2. UML Diagrams have then been made to demonstrate the different features of the EBD system. Further, UML diagrams have been provided for each feature as well. Lastly, UML diagrams for the software architecture of each file have been provided.
3. The basic algorithm has been presented with the help of a flowchart.
4. After completing the design process, the '.c' and '.h' files have been written. An attempt has been made to keep the program as modular as possible.
5. Static code analysis report using Cpp Check has been added along with this report.
6. A model has been made on SimulIDE, in order to demonstrate the working of EBD in the best possible way.
7. Code Optimization has been done on the .c files to reduce the size of the map file.

---

2. EBD Requirements:

Feature	Requirement Specification
Brake Force Distribution	The system should recognize brake pedal being pressed and should regulate the brake pressure at all the four wheels depending upon the forces at each wheel
Reading Values from sensors for input	Should take correct values from wheel load sensor, yaw sensor and should recognize when the pedal is pressed.
Should not work at high values of slip ratio	Should read the value of slip ratio from the ABS algo. If during braking, the wheel slip ratio is too high, the system should transfer control to ABS. If value lower than threshold, then distribute brake forces appropriately.
Correct operation of valves	Depending upon the desired distribution of the brake force, the ABS modulator should distribute brake fluid
3 stages of modulator	The modulator should be able to distribute the brake force in 3 ways. Either the modulator should be open, or it should hold the brake fluid, or it should release the fluid.

Recognize when brake pedal is pressed	The EBD should only operate when the brake pedal is pressed.
Should operate based on yaw sensor values, if braking under turning	If the vehicle is turning, then the brake force distribution should be done in accordance to the values of the yaw sensor. Yaw sensor values indicate if the vehicle is turning right or left.
If no yaw, brake force distribution only for the front wheels	If there is no yaw, that means there is no steering input. The brake force should only be distributed towards the front of the vehicle and not on the outer wheels.

3. UML Diagrams

3.1 Diagrams for the entire system

3.1.1 Structural Diagram

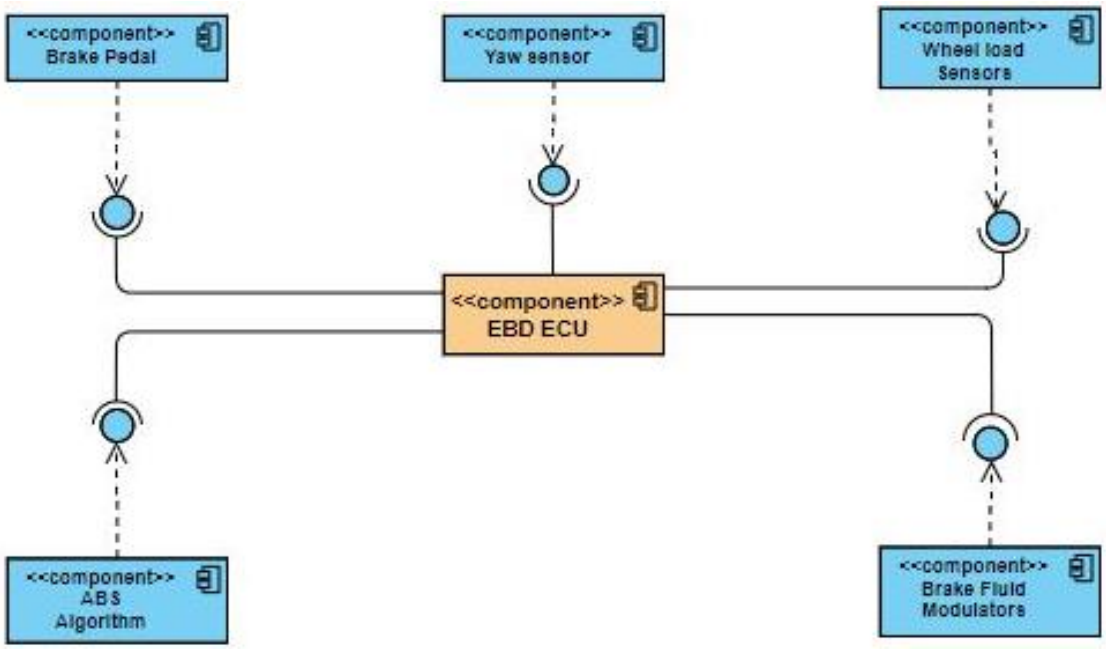

Figure 1: Component Diagram

3.1.2 Behavioural Diagram

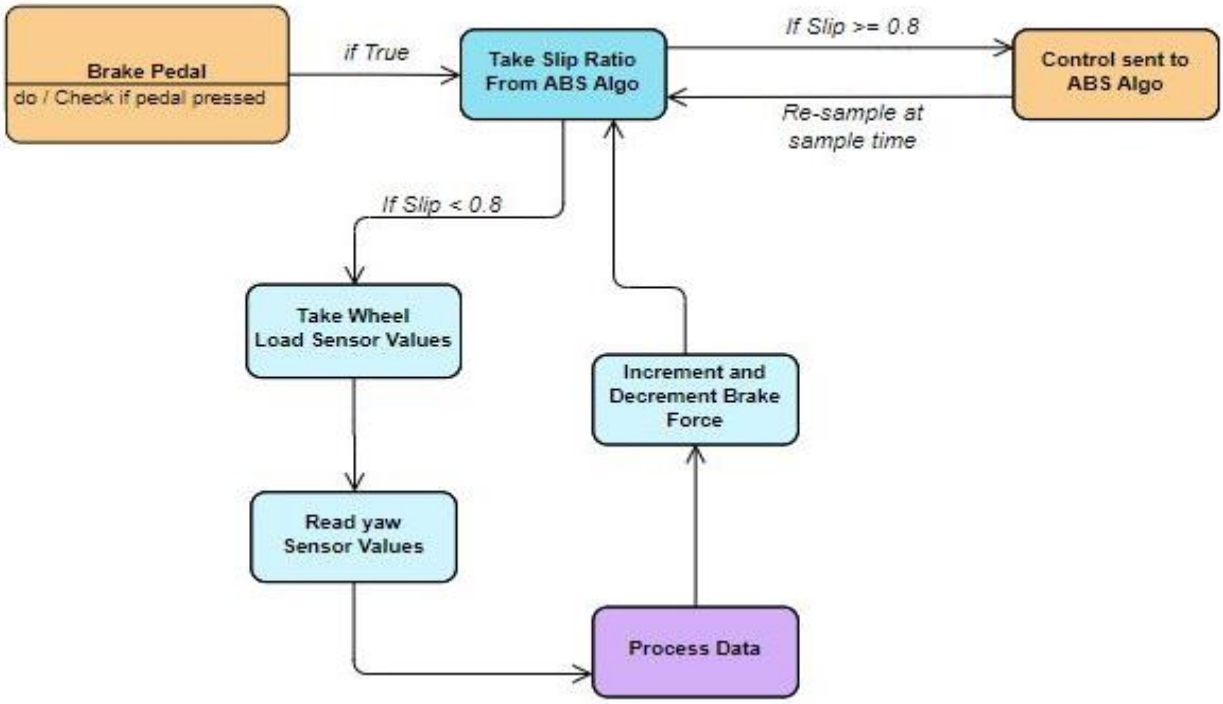

Figure 2: State Machine Diagram

3.2 Diagrams for the each feature

3.2.1 Reading Sensor Values:

Take data from the sensors and feed it to the ADC

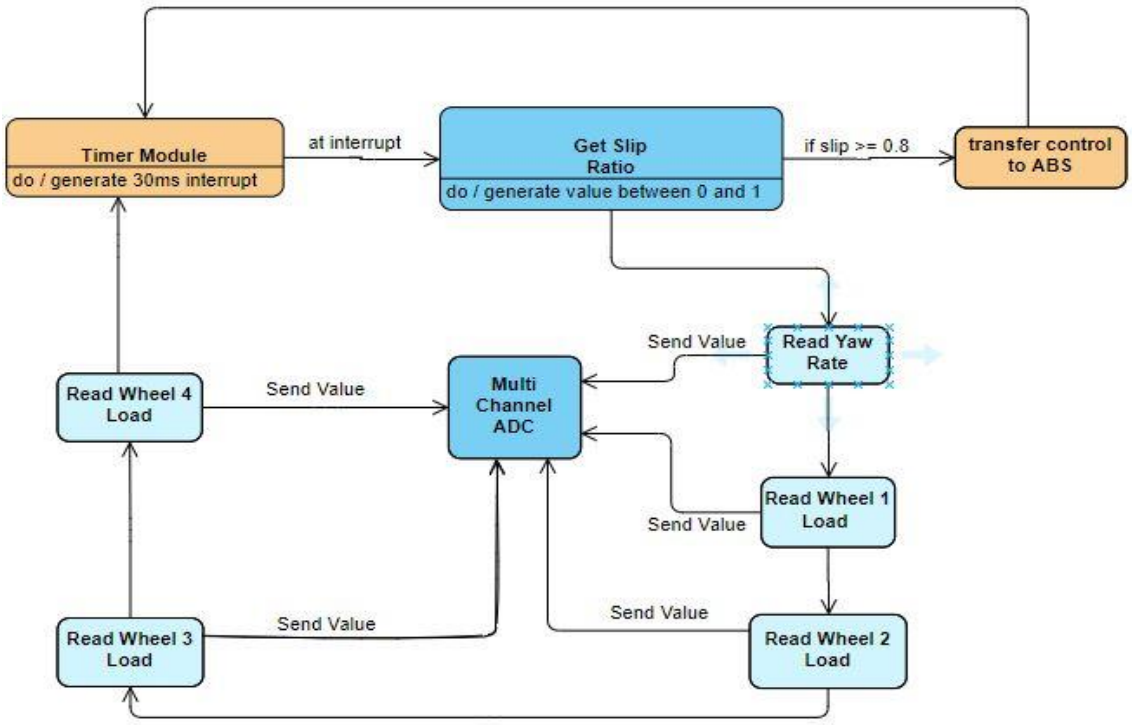

Figure 3: Read Sensor Values

3.2.2 Process Data

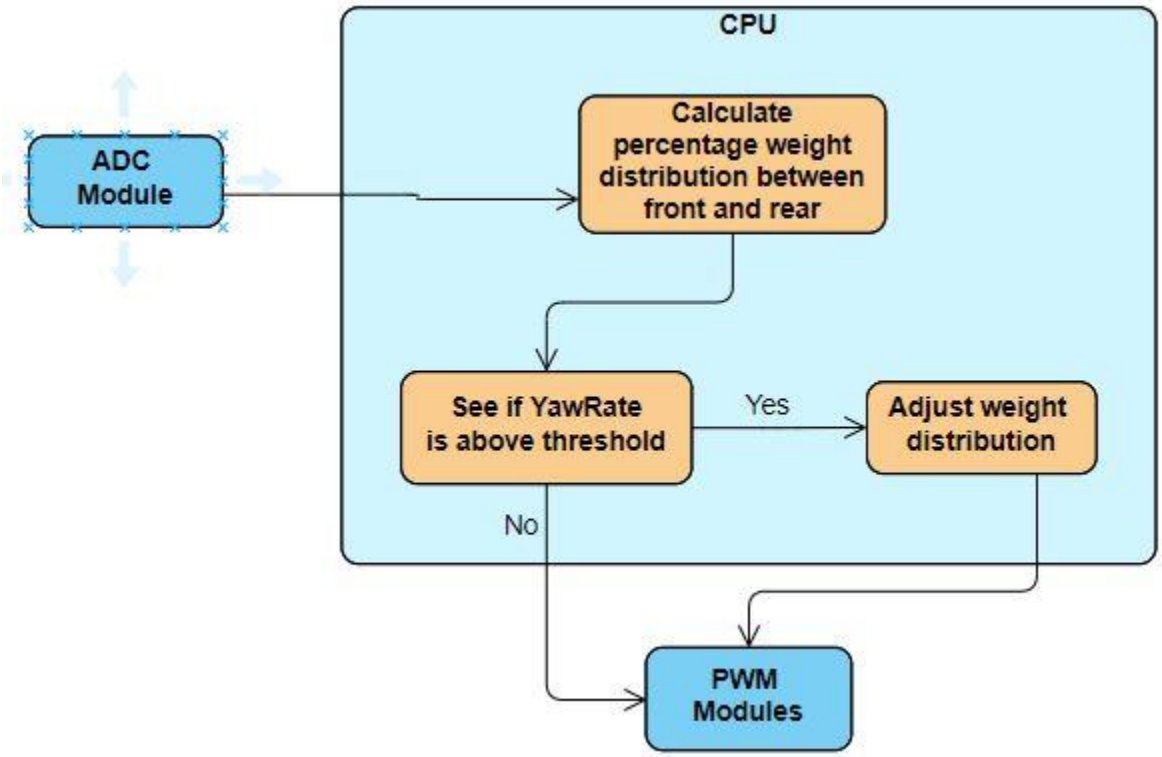

Figure 4: State Machine explaining data flow

3.2.3 Controlling the Modulators using PWM

Out of the 4 modulators, only one modulator is shown. The other 3 modulators have the same implementation.

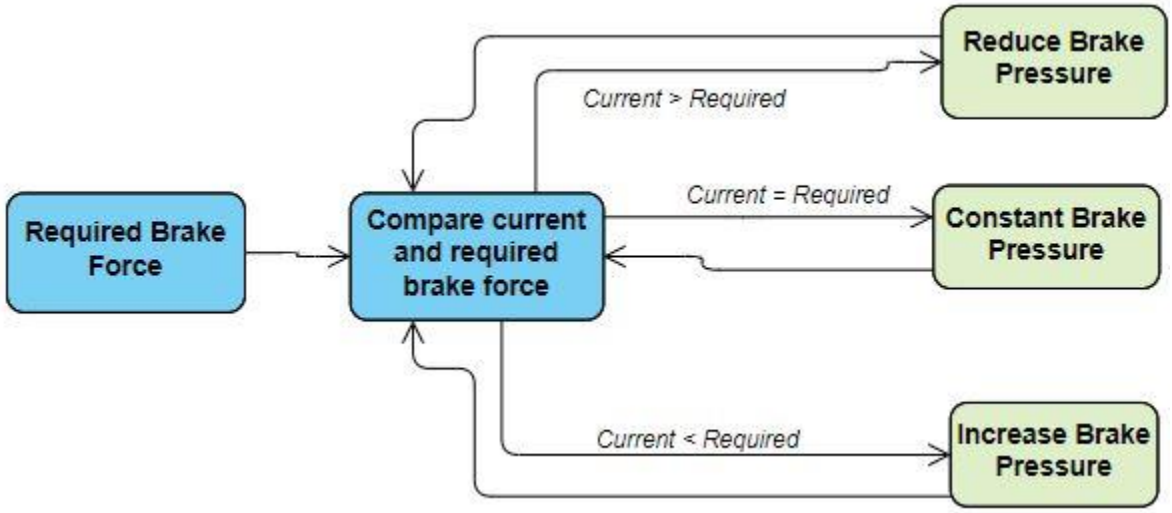

Figure 5: Modulator Algorithm for brake force distribution

4. Software Architecture for Source Codes

This section contains the software architecture of the driver codes and the source codes. All the functions used, and which function calls which is depicted in the form of UML diagrams.

4.1 Diagrams for file 'system.c'

This diagram explains the relationship between the functions written for 'system.c'.

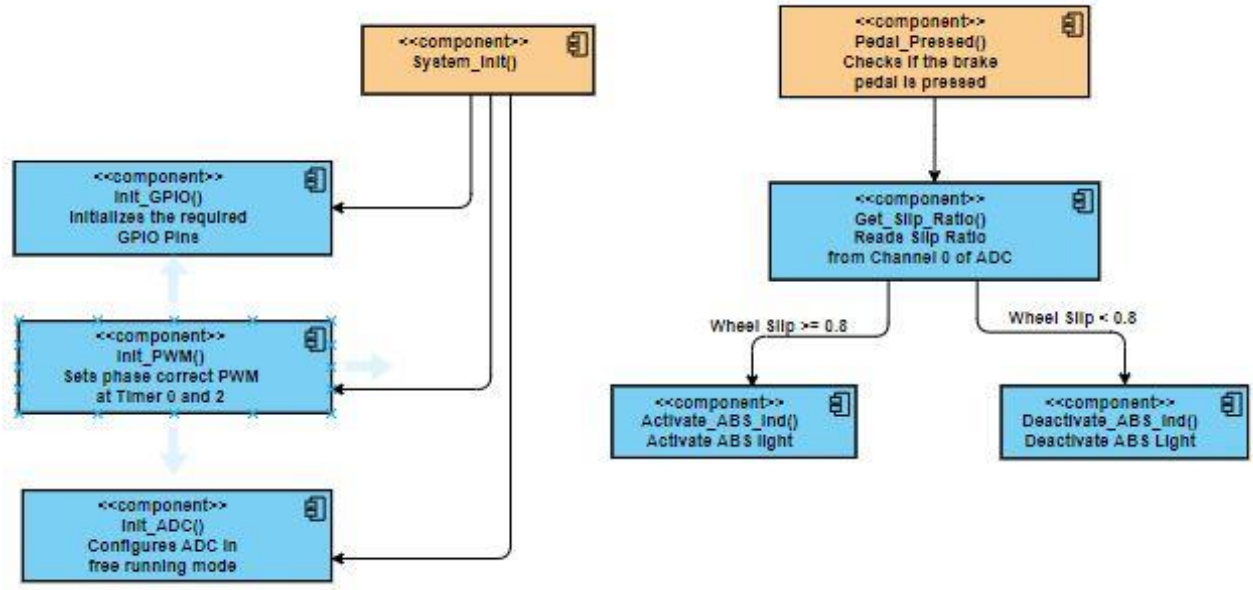

Figure 6: System.c software architecture

4.2 Diagrams for file 'read_wheel_loads.c'

This diagram explains the relationship between the functions written for 'read_wheel_loads.c'.

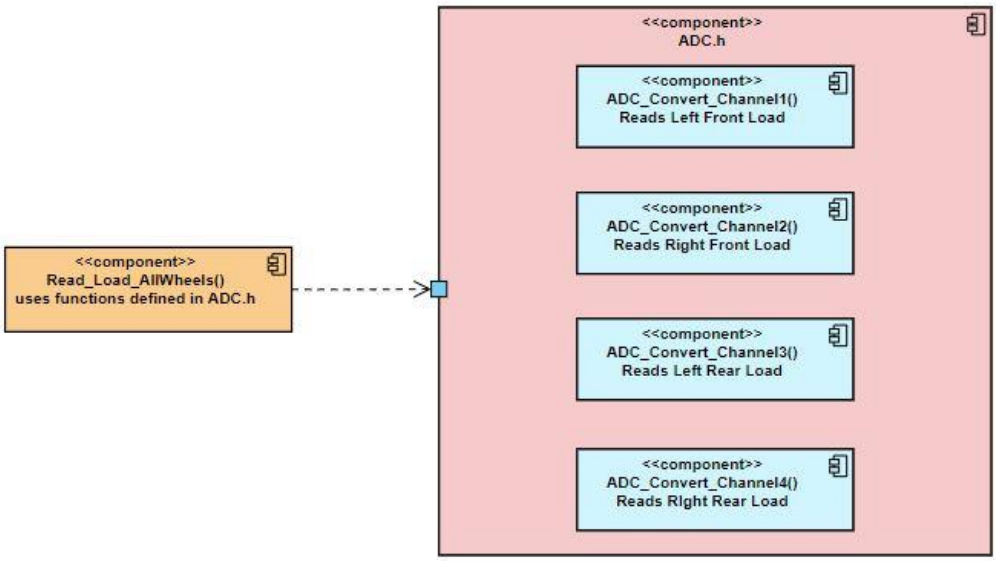

Figure 7: Read Wheel Loads

4.3 Diagrams for file 'main.c'

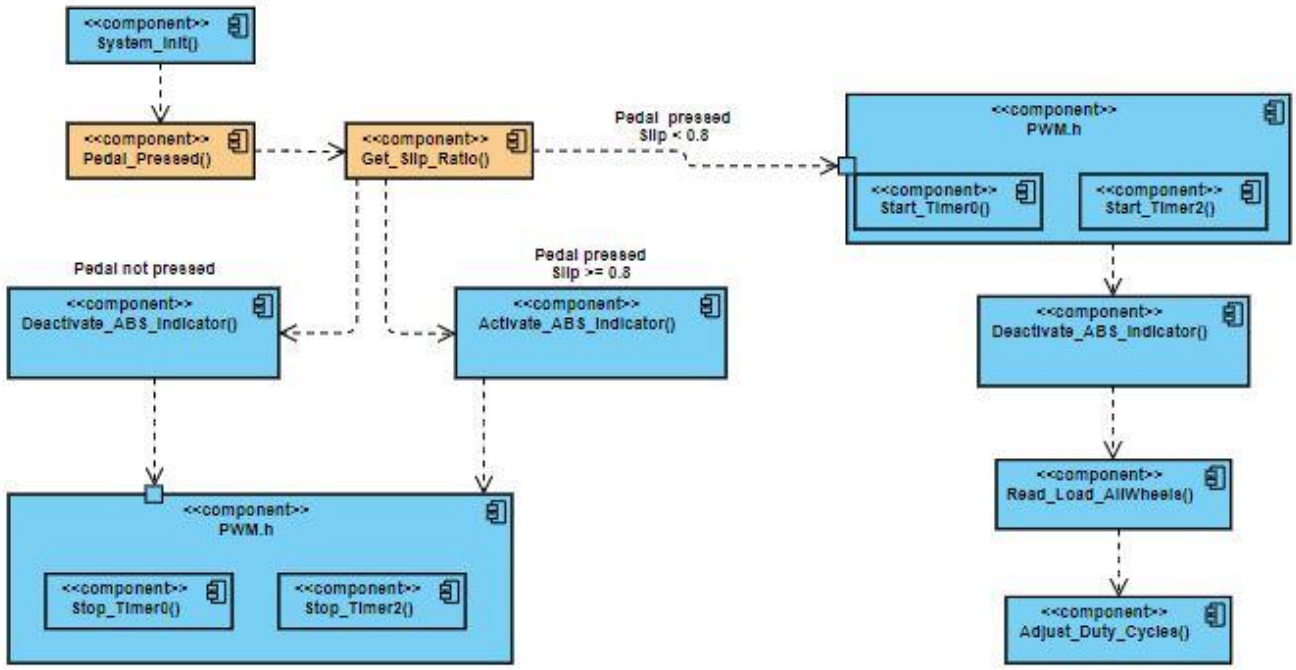

Figure 8: main code - software architecture

5. Steps to run the simulation

The following steps need to be taken in order to effectively run the simulation:

- Press the brake switch to turn the system on. Opening the brake switch, turns the system off
- The right most potentiometer connected to ADC0 is for the slip. When the slip is too high, then the EBD system turns off and the control is handed over to ABS system. This can be seen as the PWM outputs become 0, and the 'ABS On' LED lights up
- Reduce the Slip to give control back to EBD
- Potentiometers connected from ADC1 to ADC4 control the loading on the wheels. Each wheel is controlled by its own pot. Upon changing the loading, we can see that the duty cycle of the PWM output on the wheels change. More the duty cycle, more brake pressure on that particular wheel. The pot can be adjusted individually for all the wheels and the change in the duty cycle can be monitored.
- The potentiometer connected to ADC5, is for the yaw rate. Upon turning the wheel towards the right from the centre position, indicates a right turn. This would result in dynamic load transfer and the left wheels getting loaded. We see that as we turn right, the duty cycle of the left front and the left rear wheels change. Indicating an increasing brake pressure.

- The same is true for the right wheels.
- Turn the system off using the brake pedal switch.