# Project Specifications for CAN Bus Communication Simulator with ABS Module

**TBA: While the project can serve as a useful educational tool for understanding CAN communication and basic ABS concepts, it would require substantial enhancements and rigorous testing to be considered credible and reliable for real-world vehicle use.**

## Project Overview

This project simulates a CAN Bus communication system specifically designed for an Anti-lock Braking System (ABS) in vehicles. It demonstrates how different electronic control units (ECUs) can communicate over a CAN bus to monitor and control vehicle functions.

## Project Components

The project consists of the following main components:  
1. CAN Bus Simulator: Manages sending and receiving messages.  
2. ABS Module: Simulates the ABS functionality by sending wheel speed data and determining when to send a stop command.  
3. Main Program: Integrates the CAN Bus and ABS modules, controlling the simulation flow.

## Code Structure

The project is structured into the following source files:  
1. can.h: Contains shared data structures and function declarations for CAN messages.  
2. can\_bus.c: Implements the CAN Bus functionality, including message validation and communication.  
3. abs.c: Implements the ABS module logic, including dynamic speed simulation and stop command logic.  
4. main.c: Ties all components together, initiating the simulation and handling interactions.

## Functional Specifications

1. CAN Bus Functionality:  
- Validate CAN messages to ensure proper format and data integrity.  
- Simulate sending and receiving CAN messages between the ABS module and the main program.  
2. ABS Module Functionality:  
- Simulate dynamic wheel speed generation, ranging from 0 to a defined maximum speed.  
- Send wheel speed data to the main program through the CAN Bus.  
- Receive a stop command when the wheel speed exceeds a specified threshold (120 km/h).  
3. Main Program Functionality:  
- Initialize the simulation environment and manage the interaction between components.  
- Continuously monitor the wheel speed and respond accordingly, including sending stop commands.

## Technical Specifications

1. Language: C programming language.  
2. Compilation: Uses GCC for compiling the code files into an executable.  
3. Operating Environment: Simulated environment without real hardware integration.  
4. Random Speed Generation: Utilizes a random number generator to simulate wheel speeds.

## Limitations and Considerations

1. Safety and Reliability: The system requires rigorous testing and validation to meet automotive safety standards.  
   2. Hardware Integration: The project does not include actual sensor or actuator integration, limiting real-world application.  
   3. Compliance: Must comply with automotive standards (e.g., ISO 26262) for safety-critical systems.  
   4. Real-Time Processing: Lacks real-time capabilities necessary for effective control in an actual vehicle environment.

## Outputs examples



