GORTERIA ISSN: 0017-2294
2024 64(1 )

# Driving the Future: Design and Implementation of an Advanced Smart Car Network Simulation Platform for Education

Abdulmalik Humayed [1,*], Abdoh Jabbari [1*], Faheem Ahmad Reegu [1], Mueen Uddin [2], Faheem Syeed Masoodi [3]

[1] Department of Computer and Network Engineering, College of Computer Science and Information Technology, Jazan University, Jazan 45142, Saudi Arabia
[2] College of Computing and Information Technology, University of Doha for Science and Technology, Doha 24449, Qatar
[3] Department of Computer Science, University of Kashmir, India
*Corresponding author
Abdoh Jabbari: ajabbari@jazanu.edu.sa

*Abstract*— **Incorporating smart car network education into an academic setting often faces a substantial cost barrier. Traditionally, creating a realistic learning environment involved significant investments, such as procuring an actual smart car or specialized automotive-grade testing equipment. However, these approaches pose challenges, especially in resource-constrained educational contexts. This project addresses these challenges by introducing an innovative solution—a smart car education platform. Our objective is to design and implement this platform using a limited number of microcontrollers and components extracted from modern vehicles, enabling the creation of a lifelike simulation of genuine smart car networks. By adopting this approach, we enhance accessibility for students interested in gaining practical insights into automotive networks, offering a more pragmatic and cost-effective educational pathway.**

***Keywords— Automotive communication system, CAN bus system, Controller area network (CAN), CAN transceiver, CAN frame***

## I. INTRODUCTION

Rapid advancements in smart car technology have revolutionized the automotive industry in recent years, promising safer and more efficient transportation systems. The development of smart cars relies heavily on intricate networks of interconnected electronic systems that facilitate communication and coordination between various vehicle components. As the demand for professional expertise in automotive net- works grows, the need for comprehensive and accessible educational platforms becomes increasingly evident[1].

Over the course of time, automobiles have undergone a remarkable transformation. Initially designed solely for transportation, they have evolved into complex technological marvels, akin to sophisticated computer systems replete with numerous digital nodes. These nodes, referred to as Electronic Control Units (ECUs), play a pivotal role in governing an array of operations, spanning from engine optimization to navigation systems. They are intricately interconnected through a structured bus network. The seamless coordination and communication among these vehicular ECUs are indispensable for the proper functioning of an automobile. This seamless interaction is made feasible through a communication protocol, one of which is the well-established Controller Area Network (CAN) bus. CAN operates as a standardized, message-based protocol that fosters dependable and priority-driven exchanges among ECUs. Within this context, "priority-driven" signifies that communication between the ECUs on the CAN bus is orchestrated based on a predefined order of precedence[2].

Originally, the CAN bus system was conceived with a primary focus on ensuring reliable communication, with security considerations being secondary. However, the ever-increasing complexity and functionality of modern automobiles have led to a continuous proliferation of nodes within the network, consequently rendering the entire system more susceptible to potential security breaches. Thus, in pursuit of cultivating a robust system, it becomes imperative to conduct an in-depth analysis of the security threats associated with this protocol[3].
Vehicles generate a wealth of data, encompassing diverse factors, including location, engine condition, speed, and more. The bulk of this data emanates from ECUs, the CAN network, and even infotainment systems. This data serves as a foundational resource for investigating, scrutinizing, and resolving a spectrum of automotive issues. Given that this data is perpetually transmitted and received in real-time, unauthorized access to this system could offer a treasure trove of information to entities interested in scrutinizing the data exchanged within a vehicle[4].
In light of these challenges and with the goal of enhancing accessibility to smart car education, our project embarks on designing and implementing an innovative, cost-effective Smart Car Education Platform (SCEP). This platform leverages microcontrollers and components extracted from contemporary vehicles to craft a lifelike simulation of real smart car networks. Through this endeavor, we aspire to equip students with a hands-on and economically viable approach to delve into the intricacies of automotive networks. Our aim is to empower learners to explore the nuances of smart car technology without the financial constraints typically associated with such educational pursuits.

In this paper, we present the development and deployment of the SCEP as an instrumental solution for narrowing the gap in smart car education. We outline the platform’s components, architecture, and function laities, emphasizing its potential to foster a deeper understanding of automotive networks among students and researchers. Furthermore, we discuss the opportunities and implications of utilizing SCEP to inspire in novation’s, enhance research capabilities, and cultivatea skilled workforce ready to embrace the challenges of the rapidly evolving automotive industry.
The remainder of this paper is organized as follows:Section 2 provides a comprehensive review of relatedwork in smart car education, simulation platforms, andaffordable learning tools. Section 3 delves into the de- tailed design and implementation of the Smart Car Ed ucation Platform, emphasizing the key features that differentiate it from existing approaches. In Section 4,we evaluate the effectiveness and practicality of SCEP through a series of educational case studies and user feedback. Finally, Section 5 presents our concluding remarks, discussing the broader implications of the platform and outlining potential avenues for future re-search and development[5].

In conclusion, the creation of an accessible andcost-effective Smart Car Education Platform represents a crucial step toward cultivating a skilled and di-verse workforce proficient in smart car technology. Bybreaking down the barriers to learning about automotive networks, we aim to inspire a new generation of innovators and researchers, driving the progress of smartcar technology toward a safer, more sustainable, and more connected future[6]. However, in general, smart cars with CAN tend to offer cost advantages in terms of manufacturing, maintenance, and long-term efficiency compared to those without CAN as shown in Table 1.

Table 1. Shows comparison of smart cars with CAN and without CAN

| **Aspect** | **Smart Cars without CAN** | **Smart Cars with CAN** |
|---|---|---|
| Wiring Complexity | High complexity due to numerous wires | Simplified wiring with a single CAN bus |

| | | |
|---|---|---|
| Labor-Intensive Wiring | Labor-intensive wiring process | Reduced labor and material costs |
| Integration Challenges | Challenging integration of numerous lines | Streamlined integration, quicker development |
| Increased Weight | Additional wires and connectors add weight | Reduced weight, potentially improved fuel efficiency |
| Maintenance and Repairs | Diagnostics and repairs can be time-consuming and costly | Easier diagnostics, faster repairs, and lower maintenance costs |
| Simplified Diagnosis | Complex wiring makes issue identification difficult | Standardized communication simplifies diagnosis |
| Scalability | Adding new features may be costly and complicated | Easier addition of new ECUs and features |
| Regulatory Compliance | May face challenges meeting regulatory standards | Compliance with industry standards may ease certification |
| Long-Term Cost Efficiency | Higher long-term costs due to complexity | Potential for cost savings over time due to simplified design and maintenance |

## A. *Background*

- ***Introduction to Controller Area Network (CAN) Protocol***

The Controller Area Network (CAN) protocol has significantly transformed the landscape of communication among electronic devices and systems in distributed networks. Initially developed by Robert Bosch GmbH during the 1980s, CAN was initially intended for the automotive sector, facilitating communication between electronic control units (ECUs) within vehicles. However, its robustness, efficiency, and adapt- ability led to its broad adoption across various industrial and embedded systems beyond the automotive domain[7].
CAN's fundamental concept enables multiple nodes, each equipped with a microcontroller (MCU), to ex- change data in a peer-to-peer manner, eliminating the need for a central controller. This decentralized architecture not only optimizes communication but also ensures a reliable, fault-tolerant network, making CAN suitable for real-time applications requiring data integrity and rapid response.
The protocol's success stems from its unique features, such as the differential bus for noise immunity, message prioritization through an arbitration mechanism, and self-contained error detection and recovery capabilities. These traits, combined with its straight- forward design, have positioned CAN as a preferred choice for industries seeking reliable and efficient communication protocols[1].
Over time, CAN has evolved, resulting in various versions and specifications tailored to diverse application requirements. The original 11-bit CAN (CAN 2.0A) expanded to support 29-bit identifiers (CAN 2.0B), accommodating an extensive range of unique message identifiers and enabling integration into morecomplex systems.
In this overview, we delve into the core principles and features underpinning the CAN protocol. We explore its historical development, key attributes, and ad-vantages that have solidified CAN's position as a dominant communication protocol across automotive, industrial, and embedded systems. Additionally, we dis-cuss CAN's versatility in adapting to a wide array of applications, making it an indispensable communication standard in our interconnected world[8].

- ***Differential Bus and Noise Immunity***

CAN employs a differential bus, transmitting signals as complementary voltage levels (CAN_H and CAN_L). This differential signaling ensures robust noise immunity, allowing CAN to function reliably inelectrically noisy environments like vehicles, industrial machinery, and factory automation systems.

- ***Arbitration Mechanism for MessagePrioritization***

CAN employs bitwise arbitration to determine message priority on the bus. Nodes compete for bus access by comparing transmitted bits with observed ones. Messages with lower identifier values hold higher priority, ensuring swift transmission of critical messages,such as safety commands or sensor data, ideal for real-time control applications.

- ***Error Detection and Recovery***

CAN excels in autonomous error detection and recovery. Nodes continually monitor the bus for trans- mission errors and fault conditions. Upon error detection, nodes initiate automatic error recovery mechanisms, including retransmission and error frame generation. This capability maintains network integrity, even during transient faults.

- ***In-Vehicle Systems***

Modern vehicles are equipped with numerous Electronic Control Units (ECUs) that manage various functions such as engine control, brakes, airbags, lights, and more. This shift from mechanical to electronic components, driven by emissions regulations and efficiency goals, necessitated interconnections among de-vices using network protocols. ECUs include the Engine Control Module (ECM) for managing fuel and ignition timing, the Brake Control Module (BCM) ensuring controlled braking and preventing skidding, the Transmission Control Module (TCM) optimizing smooth shifts, the Telematics Control Module (TCU) overseeing in-car services like navigation and connectivity, and the Suspension Control Module (SCM) adjusting ride height and suspension settings. In modernvehicles, ECUs are integrated through various networkprotocols such as Controller Area Network (CAN), Local Interconnect Network (LIN), Media Oriented Systems Transport (MOST), and Flex Ray, with CANemerging as the most crucial due to its robustness andefficiency. These networks facilitate improved information exchange and diagnostics by connecting different subsets of ECUs[9].

## II. RELATED WORK

Existing platforms are either expensive, too technical, or not suitable. One platform costs about $28000. Another is built only for security testing. There is a need for platform that is not expensive and suitable for education. We surveyed the related research and found that ex siting platforms are either expensive, too technical, ornot suitable For example, one platform costs about28000, and Another is built only for security testing,which is not suitable for people who do not know a lotabout automotive networks There is a need for a plat-form that is not expensive and suitable for education[10]. In Table 1, we classified the platforms we found in the literature into 3 types: low, medium, and high, Weconsider any platform that costs less than 500 to bea low cost, and medium if it costs between $500 and $1000, and high if it costs more than $1000 And also,classified the platforms in terms of the needed technical level to build or use one as you can see in the table 1. Blockchain and machine learning are integral to the advancement of smart cars. Moreover, Technologies like Blockchain technology plays a crucial role in securing data, ensuring vehicle identity, and facilitating secure software updates[11][12][13]. It also helps establish tamper-proof records for vehicle history and ownership. On the other hand, machine learning powers autonomous driving capabilities, predictive maintenance, traffic prediction, and personalized user experiences. These combined technologies enhance safety, efficiency, and the overall intelligence of smart cars, shaping the future of automotive transportation[14][15][16].

### A. *Comparison*

We compared four platforms in terms of their purpose, cost, and needed technical level We see the in the top that the purpose is to reduce the barrier of entry into automotive security research The cost and technical level are considered medium, in The second plat- form is built for conducting security testing safely Butits cost and needed technical level are considered veryhigh, in The third platform is built to inject malicious traffic to assess the impact on the network The cost and technical level are considered medium, in the fourth example performed testing on two real cars This means it is very expensive and a high level of technicalknowledge is needed[17]. The comparison between existing platforms is summarized inTable. 2.

Table 2: Comparison between the existing platforms

| Purpose | Estimated Cost | Technical Knowledge Needed | Ref. |
|---|---|---|---|
| Reduce the barrier of en- try into automotive security research | Medium( $500) | Medium | [1] |
| Safe environment for auto-motive security testing | High ($28000) | High | [5] |
| Inject malicious traffic to assess the impact on the network | Medium($1000) | Medium | [3] |
| Performed testing on tworeal cars | High ( $40000) | High | [4] |

## III. DESIGN

Fig. 1 is a simplified CAN network that has 3 ECUsand each of which consists of 3 parts: a microcontroller, a CAN controller, and a CAN transceiver. Theneach ECU is connected to a bus terminated by 120 ohm resistors through two wires. In an actual car, eachECU performs certain tasks. For example, ECU1 could be connected to the (Anti-lock braking) ABS system, whereas ECU2 runs the radio and has Bluetooth interface to connect with mobile devices. ECU3 could be running the Instrument Panel Cluster (IPC) to show the speed and other information to the driver.

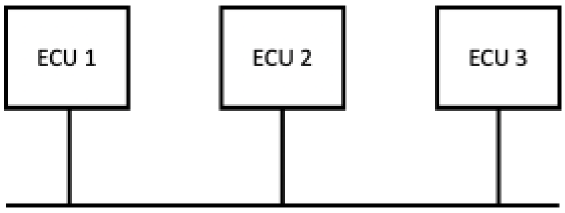


**Figure 1**. CAN network with three ECU's

Fig. 2 shows more details where it demonstrates what each ECU consists of and how to connect with the CAN bus. As shown in Fig. 2, each CAN have three parts: the First is a microcontroller: it responsible for the trans- missions and processing of the computer CAN frame and from the CAN controller and supervising the operation of second one is the CAN controller: it synchronizes with the CAN signal, sends and receives logic data to and from the CAN transceiver, adds object bitsautomatically, and deals with errors. Notably, in our attack we take advantage of this error-handling mechanism. The third part is the CAN transceiver. It actsas an interface between the CAN controller and the physical bus by translating the logic signals coming from the CAN controller into the electrical levels of the bus. CAN buses do not have clock synchronous signals, and use Non Return to Zero coding. The CAN standard dictates that one region must dominate the other. If zero "dominant" and one "recessive" bit are transmitted simultaneously, the bus state - and thus thelogic signal received by all CAN nodes - is "dominant". Most CAN bus implementations have a wired-AND configuration, hence the dominant bit is logical 0 while the necessary bit is logical 1. The state read bythe nodes is determined by the voltage measured be- tween the CAN-H and CAN-L lines; When it exceeds a certain threshold (usually 0.9 volts), the dominant state(otherwise recessive) is encoded for the data link layer,the CAN standard describes four types of frames: dataframe, far frame, error, high frame, and data The frameconsists of the start frame, and the field Judgment field, control field, data field, CRC field, ACK field, and terminated by frame end. The judging field contains the tire ID, which defines the meaning of the message con-tent, and prioritizes the tire when two or more are competing for the bus. The arbitration field is either 11 or 29 bits long, depending on the specification (CAN 2.0A or 2.0B), and the error frame consists of an error marker and an error selector.

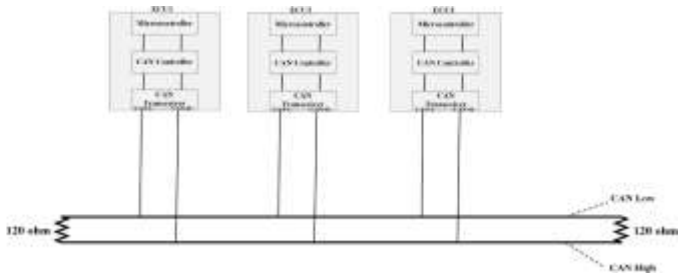
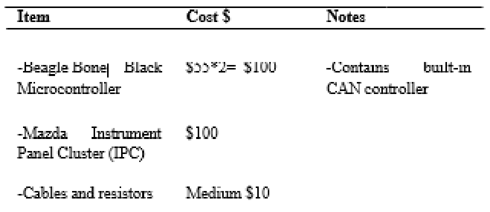


**Figure. 2**. ECUs, their components, and the CAN Bus

## IV. IMPLEMENTATION

The platform was assembled using a specific set of components, including two Beagle Bone Black microcontrollers, a Linux OS-powered laptop, CAN transceivers (specifically, MSP2551), and 120Ω resistors. These components were interconnected using a breadboard and jumper wires to facilitate communication. For cost estimation, please refer to Table 3 for a breakdown of expenses associated with our platform. It's worth noting that Beagle Bone Black microcontrollers were chosen due to their built-in CAN controller, eliminating the need for a separate CAN controller. Consequently, the total expenditure for this setup remains budget-friendly, coming in at less than $210.

**Table 3**.Estimated Cost



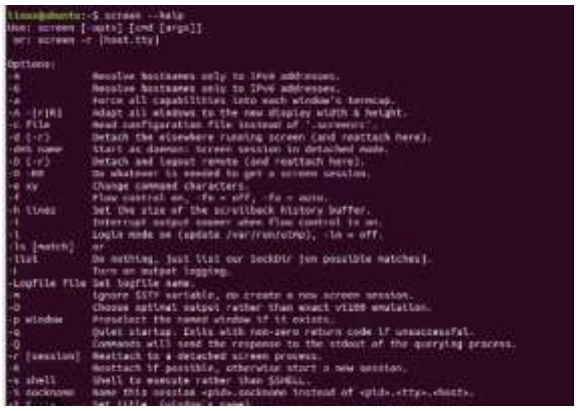
| Item | Cost $ | Notes |
|---|---|---|
| -Beagle Bone Black Microcontroller | $55*2= $100 | -Contains built-in CAN controller |
| -Mazda Instrument Panel Cluster (IPC) | $100 | |
| -Cables and resistors | Medium $10 | |

### A. *Beaglebone Black Microcontroller (BBB)*

BBB is a specially designed 32-bit ARM controller for embedded applications. It features an AM335X processor with 1GHz ARM Cortex core. This mini dashboard has 4 GB flash and 512 MB of fast DDR3 RAM. Processing capabilities are further enhanced with NEON floating point accelerator, 3D graphics accelerator, and 2x PRU 32-bit microcontrollers. This board supports a wide range of software including De bian, Angstrom, Ubuntu and Android operating systems. It run Linux in less than 10 seconds and start developing the Sitara ™ AM335x ARM Cortex-A8 processor in less than 5 minutes with just a single USB cable.

### B. *Connecting to BBB*

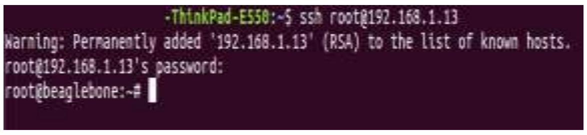
There are two ways to connect to a BeabgleBone Black. One way is through a Linux command called "Screen" as shown in Fig. 3, 4 and another method to establish a connection with a Beagle Bone Black is through Secure Shell (SSH), a secure network protocol that allows for remote access, command execution, and file transfer over a network.



```
linux@ubuntu:~$ screen --help
Use: screen [-opts] [cmd [args]]
 or: screen -r [host.tty]

Options:
-4            Resolve hostnames only to IPv4 addresses.
-6            Resolve hostnames only to IPv6 addresses.
-a            Force all capabilities into each window's termcap.
-A -[r|R]     Adapt all windows to the new display width & height.
-c file       Read configuration file instead of '.screenrc'.
-d (-r)       Detach the elsewhere running screen (and reattach here).
-dmS name     Start as daemon: Screen session in detached mode.
-D (-r)       Detach and logout remote (and reattach here).
-D -RR        Do whatever is needed to get a screen session.
-e xy         Change command characters.
-f            Flow control on, -fn = off, -fa = auto.
-h lines      Set the size of the scrollback history buffer.
-i            Interrupt output sooner when flow control is on.
-l            Login mode on (update /var/run/utmp), -ln = off.
-ls [match]   or
-list         Do nothing, just list our SockDir [on possible matches].
-L            Turn on output logging.
-Logfile file Set logfile name.
-m            ignore $STY variable, do create a new screen session.
-O            Choose optimal output rather than exact vt100 emulation.
-p window     Preselect the named window if it exists.
-q            Quiet startup. Exits with non-zero return code if unsuccessful.
-Q            Commands will send the response to the stdout of the querying process.
-r [session]  Reattach to a detached screen process.
-R            Reattach if possible, otherwise start a new session.
-s shell      Shell to execute rather than $SHELL.
-S sockname   Name this session <pid>.sockname instead of <pid>.<tty>.<host>.
```

**Fig 3**.Screen command in Linux



```
-ThinkPad-E550:~$ ssh root@192.168.1.13
Warning: Permanently added '192.168.1.13' (RSA) to the list of known hosts.
root@192.168.1.13's password:
root@beaglebone:~# 
```

**Figure 4**. SSH secure shell

## C. *SocketCAN*

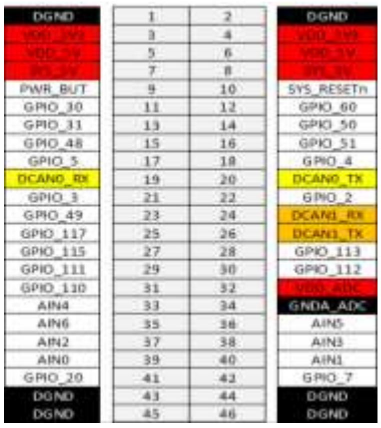
The SocketCAN package is an implementation of CAN protocols (Controller Area Network) for Linux. CAN is a networking technology which has widespread use in automation, embedded devices, andautomotive fields. While there have been other CAN implementations for Linux based on character devices,SocketCAN uses the Berkeley socket API, the Linux network stack and implements the CAN device driversas network interfaces. The SocketCAN concept on theother hand uses the model of network devices, which allows multiple applications to access one CAN de- vice simultaneously. Also, a single application is ableto access multiple CAN networks in parallel.
SocketCAN is a set of open-source CAN drivers and a networking stack that allows CAN (Controller Area Network) communication in a Linux-based environment. It provides a standardized and user-friendly interface to work with CAN devices on a Linux system. SocketCAN enables developers to communicate with CAN bus networks through the Linux kernel's networking stack, making it easier to develop CAN applications. SocketCAN simplifies CAN communication on Linux systems by providing a standardized and easy-to-use interface. It is widely used in embedded systems, automotive applications, and industrial control systems where CAN bus communication is prevalent.

## D. *CAN Transceiver*

A CAN transceiver is the interface between the CAN protocol controller and the physical wires of the CAN bus lines. Our high-speed and low-speed controller area network transceivers offer, integrated isolation, high ESD and high fault protection with value- added features specified by the ISO 11898 standard. It drives and detects data to and from the bus. It converts the single-ended logic used by the controller to the differential signal transmitted over the bus. CAN transceiver plays a crucial role in ensuring the proper electrical interface between the CAN controller and the physical CAN bus. It helps maintain signal integrity, fault tolerance, and reliable communication in CAN-based systems. The choice of the right CAN transceiver depends on factors such as the desired communication speed, environmental conditions, and specific application requirements.

## E. *Complete Wiring and Setup*

Beagle Bone Black is a low-cost, open source, community-supported development platform for ARM® Cortex™-A8 processor developers and hobbyists. Boot Linux in under 10-seconds and get started on Sitara™ AM335x ARM Cortex-A8 processor development in less than 5 minutes with just a single USB cable.
Beagle Bone Black ships with the Debian GNU/Linux™ in onboard FLASH to start evaluation and development. Many other Linux distributions and operating systems are also supported on Beagle Bone Black including: Ubuntu, Android, and Fedora. Its capabilities can be extended using plug-in boards called "capes" that can be plugged into BeagleBone Black's two 46- pin dual-row expansion headers. Capes are available for, VGA, LCD, motor control, prototyping, battery power and other functionality. Fig 5. Shows simplified pin diagram for the BeagleBone Black.



| | | | |
|---|---|---|---|
| DGND | 1 | 2 | DGND |
| VDD_3V3 | 3 | 4 | VDD_3V3 |
| VDD_5V | 5 | 6 | VDD_5V |
| SYS_5V | 7 | 8 | SYS_5V |
| PWR_BUT | 9 | 10 | SYS_RESETn |
| GPIO_30 | 11 | 12 | GPIO_60 |
| GPIO_31 | 13 | 14 | GPIO_50 |
| GPIO_48 | 15 | 16 | GPIO_51 |
| GPIO_5 | 17 | 18 | GPIO_4 |
| DCAN0_RX | 19 | 20 | DCAN0_TX |
| GPIO_3 | 21 | 22 | GPIO_2 |
| GPIO_49 | 23 | 24 | DCAN1_RX |
| GPIO_117 | 25 | 26 | DCAN1_TX |
| GPIO_115 | 27 | 28 | GPIO_113 |
| GPIO_111 | 29 | 30 | GPIO_112 |
| GPIO_110 | 31 | 32 | VDD_ADC |
| AIN4 | 33 | 34 | GNDA_ADC |
| AIN6 | 35 | 36 | AIN5 |
| AIN2 | 37 | 38 | AIN3 |
| AIN0 | 39 | 40 | AIN1 |
| GPIO_20 | 41 | 42 | GPIO_7 |
| DGND | 43 | 44 | DGND |
| DGND | 45 | 46 | DGND |

**Fig. 5**. *BeagleBone Black's Pins*

Each pin has a specific function and can be used for various purposes, such as GPIO (General-Purpose Input/Output), AIN, DGND and more. The pins are usually labeled with their functions, and you can configure them according to your project's requirements.

We used the BeagleBone Black's manual to deter- mine the pins used for the built-in CAN controller [2]. BAsed on that, we used the following pins on P9 Header Expansion:
Pin number 7 or 8 for SYS 5V
Pin number 24 for DCAN1 RX
Pin number 26 for DCAN1 TX
Pins number 1,2,43,44,45,46 for DGND
A CAN transceiver circuit is a circuit that can communicate with the CAN bus and extract data from it. Multiple electrical subsystems can connect to the CAN bus and each can communicate with the microcontroller or CAN chip connected, though not at the same time. This allows the microcontroller to analyze data from all of these units and respond accordingly to the data and which subsystems are priority. It's a message- based communication system between the microcontroller and all the electrical subsystems on the CAN bus.
The Rx pin, pin 4, is the receiver pin. It receives the data from the CAN bus. It reflects the differential bus voltage between CANH and CANL. If the differential voltage is LOW, this corresponds to a dominant state. If the differential voltage is HIGH, this corresponds to a recessive state. So the Rx pin is able to receive data

from the CAN bus so that interpretation can be made. The CANL, CAN LOW, pin, pin 6, is the pin that represents the low side of the CAN differential bus. It is connected to one of the inputs of the internal comparator inside the chip. Fig. 6. CAN Transceiver Breadboard's Connections

The CANH, CAN HIGH, pin, pin 7, is the pin that represents the high side of the CAN differential bus. It is connected to the other input of the same internal comparator chip as the CANL pin. By both the CANH and CANL pins being connected to the comparator chip, the differential voltage between the 2 pins can be measured. If the differential voltage is 0, this represents a dominant state by the particular node the chip is reading from. If the differential voltage is 1, this represents a recessive state.
The RS pin, pin 8, is the slope resistor input. This pins is used to select high-speed, slope-control, or standby mode based on the external biasing resistor. When in high-speed mode, the transistor output drivers have fast output rise and fall times to support high-speed CAN bus rates. High-speed mode is selected by connecting the RS pin to V SS.
Slope-control mode further reduces electromagnetic interference (EMI) by limiting the rise and fall times of CANH and CANL. The slope, or slew rate, is con- trolled by connecting an external resistor between RS and ground. The slope is proportional to the current output at the RS pin.
The last mode is the standby, or sleep, mode. In sleep mode, the transmitter is switched off and the receive pin operates at a lower current. The receive pin still is functional, only operating at a slower speed.
VREF, the reference voltage, pin, pin 5, is equal to VDD/2. We don't use it in our circuit, so we leave it unconnected.

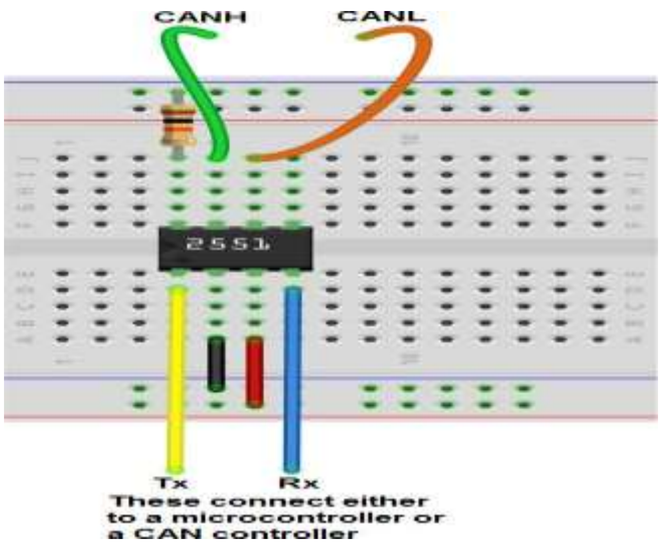

CANH
CANL
2551
Tx
Rx
These connect either to a microcontroller or a CAN controller

**Fig. 6**. *CAN Transceiver Breadboard's Connections*

To power on the MCP2551 CAN transceiver chip, we connect VDD, pin 3, to 5V and we connect VSS, pin 2, to ground. The maximum voltage that the chip can receive is 7V, so 7V should not be exceeded. 5 volts supplies sufficient power for the MCP2551 to beable to operate.
The CAN-compatible device that you want to add tothe CAN bus connects to the CANH and CANL pins. A CAN-compatible device has 2 terminals, a CANH and a CANL. You connect the CANH terminal of the device to the CANH terminal of the MCP2551 and theCANL terminal of the device to the CANL terminal of the chip. All devices that you want to connect to the CAN bus are connected to the CANH and CANL lines. Up to 112 nodes can be connected to the CAN bus.
How it works is that the CANH and CANL linesare connected to the inputs of an internal comparatorof the chip. These comparators measure the inputs andare able to give us the differential voltage output of the2

signals. If the differential output is 0V, then the sig-nal is said to be dominant. If the differential voltage is greater than 1V, then the signal is said to be recessive.The output of this signal, the differential signal, is output through the Rx pin, pin 4. This differential volt-age is then transmitted either to the CAN controller or microcontroller for analysis.
After the signal is analyzed by the microcontroller/CAN controller, it is then sent out from the Tx pin to the MCP2551 transceiver chip. So the Rx pinoutputs the differential voltage to the microcontrollerand the microcontroller outputs the analyzed signalthrough the Tx pin back to the MCP2551. This is whythe chip is a transceiver. It sends out and receives data.To reduce EMI emission in the circuit, we connect the RS pin, pin 8, to a resistor connected to ground.A resistor of about 10KΩ suffices well. This preventsfurther EMI.
And this is all that is required for connecting the hardware of a CAN transceiver chip. Programming the chip, on the other hand, is a completely different story and is much harder. These is because there are many components of the code to under-stand. There are identifier bits, error bits, and understanding of priority, different formats, and more to deal with.Programming the chip requires a lot of in-depth knowledge about what exactly occurs during the entirereceive/transmit process. You must also be able to de-code the entire message for data transmission, which requires a good deal of knowledge.
Again, CAN bus communication is a standard to- day. If you know the schematic diagram of the electronics housed in a car, you can access the data on thisCAN bus system using a CAN transceiver chip and connect it to a microcontroller to get data from electricalsubunits of the car to a microcontroller for processing.This can allow you to tap into the data and perform whatever actions you need based on it.Fig. 7 shows how we connected the parts in details.

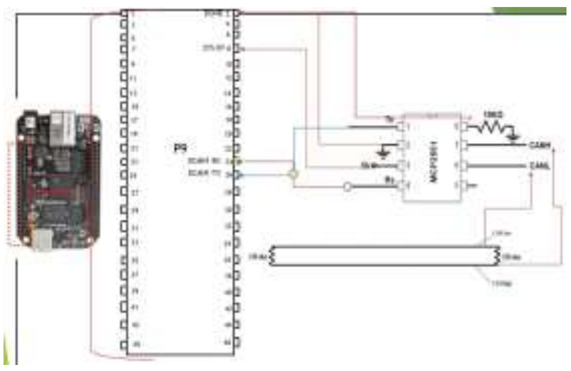


**Fig. 7**. *Detailed Overview Wiring*

On Ubuntu, the required kernel modules are in-cluded, however you will need to install can-utils.
sudo apt-get install can-utils Enabling SocketCAN
You may need to enable the relevant kernel modules.To do so, run:

- sudo modprobe can - sudo modprobe vcan -sudo modprobe slcan

**SocketCAN with CANtact**

- To use CANtact with SocketCAN, you will need torun slcand. For example:
- sudo slcand -o -c -s6 /dev/ttyACM0 can0

This command creates a new device called can0 thatis connected to the CANtact at /dev/ttyACM0. It will open the device when starting (-o), close the device when finished (-c), and set the speed mode to 6 (-s6).
After running this command, you will need to en- able the interface:

- sudo ifconfig can0 up

There are two ways to connect the laptop to a BBB:
1. Command screen 2. Ethernet cable and we have used screen command:
$ Screen /dev/ttyACM0 115200

## V. ECUs' Communication

### A. *Getting SocketCAN*

To get SocketCAN on your Linux system, you typically don't need to install it separately because it's already included in the Linux kernel. SocketCAN is a set of CAN drivers and a networking stack that's integrated into the Linux kernel. However, you might need to load the relevant kernel modules and configure them to use SocketCAN. SocketCAN simplifies CAN communication on Linux-based systems by providing a standardized interface and tools for working with CAN networks. If you need to develop CAN applications or perform diagnostics, you can use the SocketCAN API and related utilities to interact with your CAN network.
This number and determine the speed of communication in the wire and the enter user with or without password if you want too When you do this things you should be inside the BeagleBone Black CAN bus after that use.
# canconfig can0 bitrate 500000
# canconfig can0 start to start the ECU.
And the use command
#cangen can0 /To Generates messages. #candump can0 /to display messages

### B. *Experimenting with an Instrument Panel Cluster (IPC)*

To experiment with a realistic in-vehicle network, we connected the CAN bus we have built with an IPC that we have purchased. We located the pins from the manual showing in fig.8.

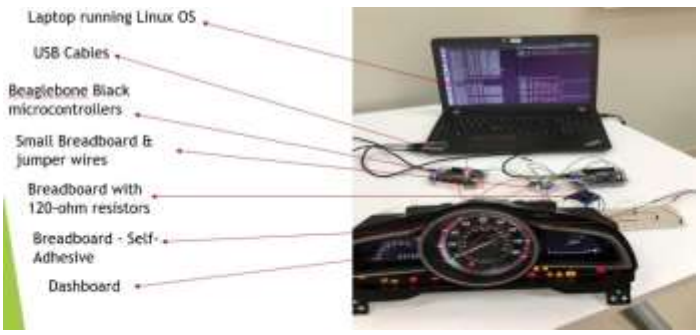

Laptop running Linux OS
USB Cables
Beaglebone Black microcontrollers
Small Breadboard & jumper wires
Breadboard with 120-ohm resistors
Breadboard - Self-Adhesive
Dashboard

Fig. 8. IPC & the Platform

To use CAN (Controller Area Network) bus and CAN transceivers on a BeagleBone Black (BBB):

1. Enable the CAN bus by loading kernel modules.
2. Connect a CAN transceiver to the BBB, specifying pins for CANH and CANL.
3. Configure the CAN interface using the ip command or device tree overlay.
4. Send and receive CAN messages using SocketCAN utilities like cansend and candump.
5. Develop custom CAN applications in languages like C/C++ or Python.

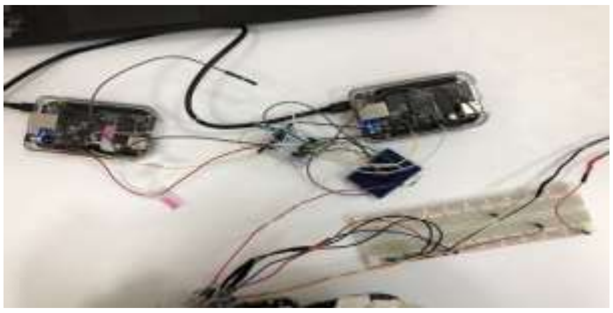


**Fig. 8**. BBBs, CAN Bus, and Transceivers

- ***Signals***

```
root@beaglebone:~# cat flasher.sh #!/bin/bash
While :  do
cansend can0 09a#0000ff0000000000 sleep 0.5
cansend can0 09a#0000000000000000 sleep 0.5
Done
```

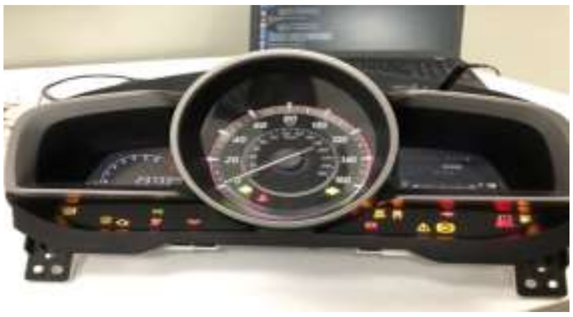


Fig 10. Result of Sending Flasher’s CAN Frames

```
root@beaglebone:~# cat right.sh #!/bin/bash
While : do
cansend can0 09a#00000a0000000000 sleep 0.5
cansend can0 09a#0000000000000000 sleep 0.5
Done
```

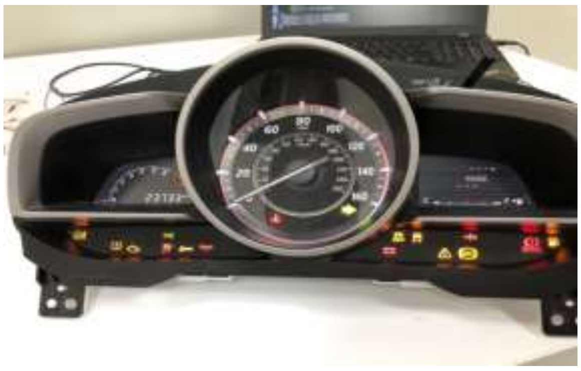


**Fig.11**.Result of sending right signal CAN frame

```
root@beaglebone:~# cat left.sh #!/bin/bash
While : do
```

```
cansend can0 09a#0000040000000000 sleep 0.5
cansend can0 09a#0000000000000000 sleep 0.5
done
```

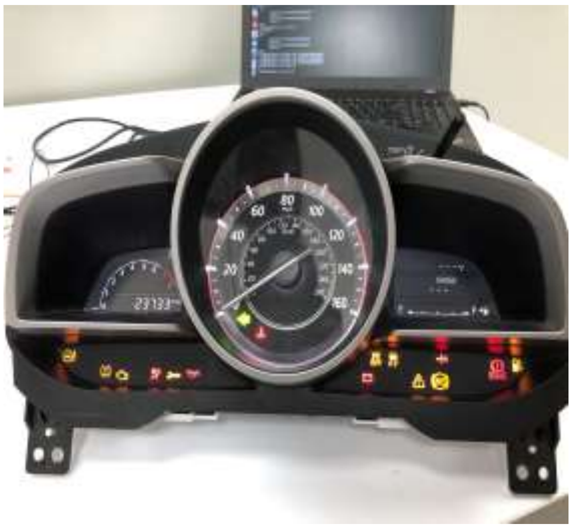


**Fig.12**.Result of sending right signal CAN frame

## VI. CONCLUSION

"Driving the Future: Design and Implementation of an Advanced Smart Car Network Simulation Platform for Education" has achieved significant milestones in the realm of automotive education and simulation. This innovative platform, built around the BeagleBone Black, has successfully provided students and enthusiasts with a practical and affordable means to delve into the complexities of smart car networks.
The project's accomplishments include the integration of CAN bus communication, the use of CAN transceivers, and the development of educational content that empowers learners to understand the intricacies of automotive networks. By leveraging the power of the BeagleBone Black and the flexibility of Linux-based SocketCAN, this platform has become a valuable resource for fostering a deeper understanding of modern automotive technology.

## VII. FUTURE WORK

As we look to the future, there are several exciting avenues for further development and enhancement of this smart car network simulation platform:

1. Continue refining the simulation to closely mimic real-world smart car networks, incorporating additional sensors, actuators, and automotive subsystems.
2. Expand the platform's educational content to cover a broader range of topics, including advanced driver-assistance systems (ADAS), autonomous driving, and cybersecurity in smart vehicles.
3. Explore the integration of artificial intelligence (AI) and machine learning (ML) components to create more dynamic and adaptive simulations for educational purposes.
4. Foster a collaborative learning community where users can share their projects, insights, and experiences, creating a vibrant ecosystem of automotive enthusiasts and educators.
5. Develop remote access capabilities, allowing users to interact with the platform from anywhere, enabling distance education and research collaboration.
6. Forge partnerships with automotive industry players to ensure the platform remains aligned with current industry trends and technologies.
7. Consider offering certification programs or badges for users who complete specific modules or demonstrate proficiency in automotive network simulations.

By pursuing these avenues, the "Driving the Future" platform can continue to evolve and remain at the forefront of automotive education, preparing the next generation of engineers and innovators to drive the future of smart car technology.

## VIII. ACKNOWLEDGEMENT

The author extends their appreciation to the Deanship of Scientific Research, Jazan University, for supporting this research work through Research Units Support Program, Support Number RUP2-01.

## REFERENCE:

[1] T. Indiketiya, “Smart and Interactive Laboratory Setup to Teach Controller Area Network Protocol,” *11th Annu. IEEE Inf. Technol. Electron. Mob. Commun. Conf. IEMCON 2020*, pp. 797–803, 2020, doi: 10.1109/IEMCON51383.2020.9284831.

[2] B. S. Bari, K. Yelamarthi, and S. Ghafoor, “Intrusion Detection in Vehicle Controller Area Network (CAN) Bus Using Machine Learning: A Comparative Performance Study,” *Sensors*, vol. 23, no. 7, 2023, doi: 10.3390/s23073610.

[3] R. S. Rathore, C. Hewage, O. Kaiwartya, and J. Lloret, “In-Vehicle Communication Cyber Security: Challenges and Solutions,” *Sensors*, vol. 22, no. 17, 2022, doi: 10.3390/s22176679.

[4] S. F. Lokman, A. T. Othman, and M. H. Abu-Bakar, “Intrusion detection system for automotive Controller Area Network (CAN) bus system: a review,” *Eurasip J. Wirel. Commun. Netw.*, vol. 2019, no. 1, 2019, doi: 10.1186/s13638-019-1484-3.

[5] A. Buscemi, I. Turcanu, G. Castignani, A. Panchenko, T. Engel, and K. G. Shin, “A Survey on Controller Area Network Reverse Engineering,” *IEEE Commun. Surv. Tutorials*, vol. 25, no. 3, pp. 1445–1481, 2023, doi: 10.1109/COMST.2023.3264928.

[6] E. Aliwa, O. Rana, C. Perera, and P. Burnap, “Cyberattacks and Countermeasures for In-Vehicle Networks,” *ACM Comput. Surv.*, vol. 54, no. 1, pp. 1–37, 2021, doi: 10.1145/3431233.

[7] T. P. Presi, “Design and development of PIC microcontroller based vehicle monitoring system using Controller Area Network (CAN) protocol,” *2013 Int. Conf. Inf. Commun. Embed. Syst. ICICES 2013*, pp. 1070–1076, 2013, doi: 10.1109/ICICES.2013.6508232.

[8] D. Oladimeji, R. Amar, S. Narasimha, and C. Varol, “A Testbed for a Controller Area Network Communication Protocol in Automobiles,” *Proc. - IEEE Consum. Commun. Netw. Conf. CCNC*, vol. 2023-Janua, pp. 1–6, 2023, doi: 10.1109/CCNC51644.2023.10059608.

[9] H. Olufowobi, C. Young, J. Zambreno, and G. Bloom, “SAIDuCANT: Specification-Based Automotive Intrusion Detection Using Controller Area Network (CAN) Timing,” *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 1484–1494, 2020, doi: 10.1109/TVT.2019.2961344.

[10] S. N. Narayanan, K. Khanna, B. K. Panigrahi, and A. Joshi, *Security in smart cyber-physical systems: A case study on smart grids and smart cars*. Elsevier Inc., 2018.

[11] F. A. Reegu *et al.*, “Interoperability Requirements for Blockchain-Enabled Electronic Health Records in Healthcare: A Systematic Review and Open Research Challenges,” *Secur. Commun. Networks*, vol. 2022, 2022, doi: 10.1155/2022/9227343.

[12] F. A. Reegu *et al.*, “Blockchain-Based Framework for Interoperable Electronic Health Records for an Improved Healthcare System,” *Sustain.*, vol. 15, no. 8, 2023, doi: 10.3390/su15086337.

[13] F. Reegu, W. Khan, … S. D.-… C. on R., and undefined 2020, “A Reliable Public Safety Framework for Industrial Internet of Things (IIoT),” *ieeexplore.ieee.org*.

[14] S. Ayoub, Y. Gulzar, J. Rustamov, A. Jabbari, F. A. Reegu, and S. Turaev, “Adversarial Approaches to Tackle Imbalanced Data in Machine Learning,” *Sustain.*, vol. 15, no. 9, 2023, doi: 10.3390/su15097097.

[15] S. Ayoub, Y. Gulzar, F. A. Reegu, and S. Turaev, “Generating Image Captions Using Bahdanau Attention Mechanism and Transfer Learning,” *Symmetry (Basel).*, vol. 14, no. 12, 2022, doi: 10.3390/sym14122681.
[16] shadab alam faheem reegu, salwani daud, zaid hakami, Kaiser kareem reegu, “Towards Trustworthiness of Electronic Health Record system using Blockchain,” *Ann. RSCB*, vol. 25, no. 6, 2021.
[17] A. De Faveri Tron, S. Longari, M. Carminati, M. Polino, and S. Zanero, “CANflict: Exploiting Peripheral Conflicts for Data-Link Layer Attacks on Automotive Networks,” *Proc. ACM Conf. Comput. Commun. Secur.*, pp. 711–723, 2022, doi: 10.1145/3548606.3560618.