Design And Implementation of CAN bus Protocol for Monitoring Vehicle Parameters

A. A. Salunkhe, Pravin P Kamble, Rohit Jadhav

***Abstract*— CAN bus is a serial bus protocol used in most automotive industries. Recently, more attention on the CAN bus network application and wireless transmission bus carrying technology. CAN bus technology is adopted in modern vehicles to reduce the wiring harness and controls the vehicle using electronics system. In this paper, CAN bus protocol is implemented using a single board computer (SBC) and accessed the vehicle parameters remotely. Raspberry Pi is used as SBC which has a high performance processor used to increase speed ability. The presented work concentrate on the communication between three nodes using a CAN bus and how to access modern vehicle ECU parameters using OBD II connector. The paper is described further set up of CAN device using Linux software.**

Keywords*— *CAN Bus Protocol, Embedded System, Remote monitoring system

I. Introduction

The Controller Area Network (CAN) is a serial field bus system with a message based protocol. The protocol is multi-master based. CAN was developed by the Robert Bosch GmbH and released in 1986. Developed to reduce wiring cost in automobile industry. Currently, computer technology has been widely used in vehicle monitoring and control systems, and the proportion of car electronics represented by micro-controller continues to increase at the whole vehicle electronic systems.

The system describes communication between three nodes using a CAN bus. The two nodes are designed for getting the information of vehicle parameters such as speed, Engine temperature and brake status measurement. The two nodes transmit their sensed data over a can bus. The third node receive the data from the CAN bus and display it on GUI (Graphical User Interface) as well as the third node act as server so that a user or owner can get the information of such parameters remotely[1]. The second part of this paper is that third node is connected to OBD-II (On Board Diagnostic-II) connector which is present in every modern vehicle below or near the steering. The third node can access all the ECU (electronic control unit) parameters and control the vehicle by writing the messages through the third node. The system will be used to hack the vehicle remotely.

A. A. Salunkhe, Asst. Professor, Department Of Electronics Engineering, Walchand College Of Engg., Sangli, Maharashtra, India (aas.coepian@gmail.com)

Pravin P Kamble, Department Of Electronics Engineering, Walchand College Of Engg., Sangli, Maharashtra, India (kambleppk@gmail.com)

Rohit Jadhav, Department Of Electronics Engineering, Walchand College Of Engineering., Sangli, Maharashtra, India (rohitjadhav1@gmail.com)

II. Overall System Design

The figure1 shows the network structure of the interior body of the vehicle. All the electrical system nodes inside the bus vehicle are attached to the CAN bus through their own respective CAN transceiver, CAN controller and SBC. The wireless module is connected to the single board computer which is used as internal and external communication data interface. The single board computer gathered up information from all the interior electrical system nodes send to the remote Internet monitoring service center computer via wireless module.

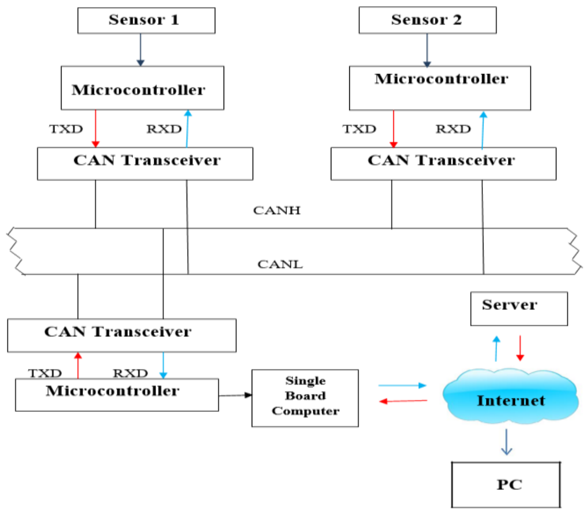


Fig. 1: Network structure of CAN bus system

III. Node Of Than Can Bus

The network uses two cables as a transmission medium. The two cables are CAN High and CAN Low. All the system nodes connected to the CAN bus through the corresponding hardware interface. All nodes share the same data channel. Each node consists a CAN Transceiver, CAN Controller and SBC has shown in figure 2.

A. Raspberry Pi as a single board computer

The Raspberry Pi 2 model B, shown in figure 3 is a single board minicomputer developed in the UK by the Raspberry Pi foundation. It is based on the Broadcom BCM2836 system-on-chip which includes a quad core ARMv7 processor. For this work important are the Serial Peripheral Interface (SPI) and the General Purpose Input Output Pins (GPIO) of the BCM2836. The Raspberry Pi needs a power supply of 5 V and at least 1200 mA which is connected via micro USB. SD cards are used for booting an operating system and store memory.

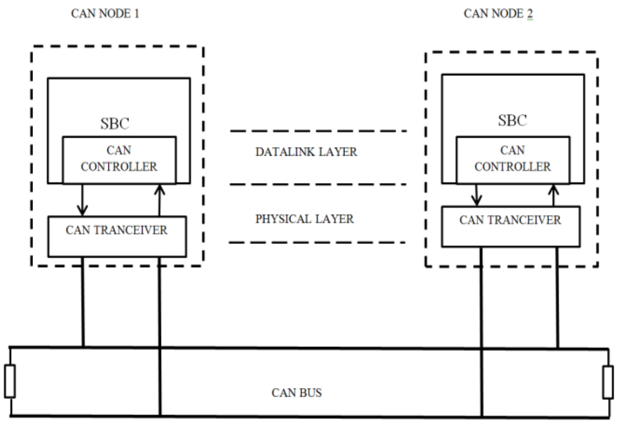


Fig. 2: Node of the CAN bus system

The Raspberry Pi contains 40 low-level peripheral pins, called GPIO Interface, which are connected directly to the BCM2836 chip's GPIO pins. With the installed secure shell (SSH) client, it is possible to access the Raspberry Pi over Ethernet as well as it can be used as a server. The Raspberry Pi uses Linux-kernel based operating systems. For this work, Linux distribution Raspbian (based on Debian) is used. The Raspberry Pi does not have CAN bus support. It uses SPI pins to connect with the CAN bus using CAN Controller and CAN Transceiver. For connecting the CAN controller to the SPI pin of the Raspberry Pi, the MCP251x driver must be installed in the kernel of the Raspberry Pi. The latest raspbian (raspbian wheezy 2015-05-05 onwards) version has now a built-in support for Microchip CAN Controller MCP251x. We don't need to add or install modules for support CAN bus protocol to the Raspberry Pi.

Fig. 3: Raspberry Pi 2 Model B as a Single Board Computer

B. *MCP2515 as a CAN Controller*

The MCP2515 controller Microchip implements CAN protocol up to 1 MBit/s transmission speed [3]. It handles the transmission of normal (11-bit ID) and extended (29-bit ID) CAN frame and works both with high speed (1 Mbit/s) and low speed (125 kbit/s) CAN transceivers. This CAN controller includes two reception buffer and three transmission buffer. The device consists of three blocks CAN module, control logic block and SPI protocol block. The CAN module handles the transmission of messages. The control logic block controls the setup and operation of the other blocks. The SPI protocol block is the interface to the Raspberry Pi. It uses usual SPI read and write commands to access the controller's registers as well as special commands like reset or request-to-send. As compare with the OSI layers of the communication system MCP2515 work as a data link layer. MCP2515 generates the message frame, does the arbitration/prioritization processes and monitors the bus state (idle/busy). The MCP2515 performs bit timing and synchronization, message validation, acknowledgment and error detection. It also decides which message should be transmitted or rejected.

C. *MCP2551 as a CAN Transceiver*

The MCP2551 is a CAN transceiver which is a high speed CAN device that serves as the interface between a CAN controller and the physical bus [3]. The MCP2551 device provides differential signaling transmit and receive capability for the CAN protocol controller, and is fully compatible with the ISO-11898 standard. It will operate at speeds of up to 1 MB/s. Typically, each node must have a device to convert the digital signals generated by a CAN controller to signals suitable for transmission over the bus (differential output). It also provides a buffer between the CAN controller and the high-voltage spikes generated on the CAN bus by outside sources. As compare to OSI layers of the communication system MCP2551 work as a physical layer. It defines a bit representation and the electrical aspects (currents, voltage levels). The CAN bus has two states shown in figure 4. The two states are Dominant and Recessive. A Dominant state occurs when the differential voltage between CANH and CANL is 2V. A Recessive state occurs when the differential voltage is less than a defined voltage i.e. 0v. The Dominant and Recessive states correspond to the Low and High state of the TXD input pin, respectively. However, a Dominant state initiated by another CAN node will override a Recessive state on the CAN bus.

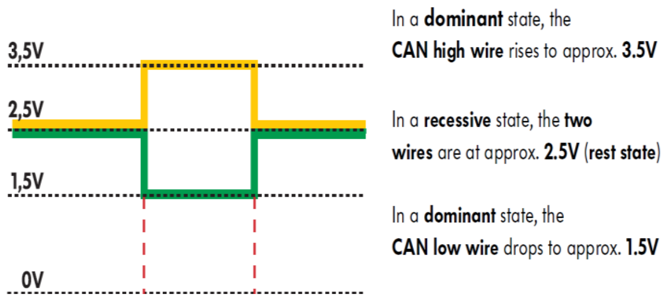


Fig. 4: Signal level of CAN bus

IV. SOFTWARE DESIGN

SPI is used for transmitting CAN frames from user space applications to the CAN controller. The modules needed to set up SPI communication with a CAN controller on Raspberry Pi is already loaded in the latest Linux kernel [4]. The modules are CAN controller driver mcp251x.ko, SPI master driver spi-bcm2708.ko for the SPI provided by the chip BCM2836 on the RasPi, SPI driver spidev.ko, which allows user space access to the SPI devices, SPI configuration tool spi-config.ko, can.ko and can-raw.ko. can-utils software package contains the original and additional drivers and the configuration tool spi-config, as well as the SocketCAN programs. SPI master driver spi-bcm2708 has to be loaded into the kernel at run time. This module enables SPI access. The tool spi-config allows registering a connected SPI device and initializing it at run time. The kernel module can-dev.ko is a general driver for all CAN devices. It provides functions which all CAN controllers have in common. The driver mcp251x is a specific driver for the CAN controller MCP2515. The driver activates the device by initializing the registers and handles the reception (RX) and transmission (TX) of messages over the SPI. The mcp251x only supports the interrupt method to check if a message was transferred or received. The software library SocketCAN, which is a Low Level CAN Framework (LLCF) by Volkswagen Research [5], is already integrated in the Raspian kernel and allows transmission of CAN frames from user space via sockets to the CAN controller. The controller is then handled as a network device. SocketCAN implements a new protocol family in the network protocol layer to address CAN device. SocketCAN provides a wide range of programs for monitoring CAN traffic and generation of CAN frames for test purpose.

V. HARDWARE DESIGN

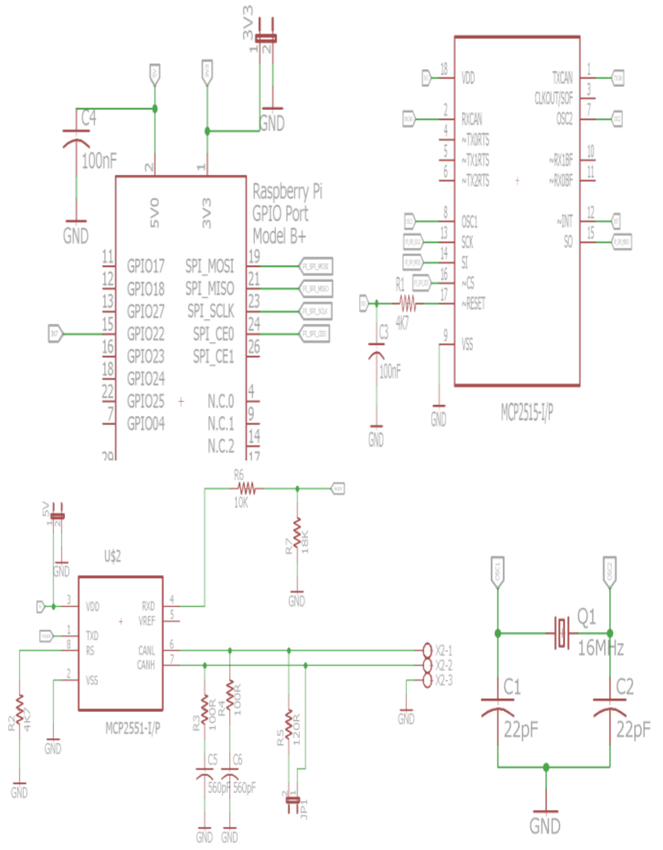


Fig. 5: RaspberryPi interface with MCP2515 and MCP2551

The figure 5 shows the interfacing diagram of Raspberry Pi with the CAN controller and CAN transceiver. First the Raspberry Pi writes a message over SPI then the message is loaded into the buffer of CAN controller via SPI drivers. If the buffer is not empty, then it will wait. Using the send command from Raspberry Pi, the controller transmits the message via transceiver over the CAN bus. After successful transmission, the controller generates an interrupt. The driver checks the register via SPI, to identify what triggered the interrupt. Another SPI command is sent to clear the interrupt register, whereupon the controller stops generating the interrupt signal. In this way the message gets transferred over CAN bus using CAN controller and CAN transceiver. On receiving the message from CAN bus to Raspberry Pi, the voltage divider circuitry is used for voltage level compatibility between MCP2515 and MCP2551.

VI. RESULTS

Communication between the three nodes using CAN bus is shown in figure 6. The two nodes sense the parameters using the particular sensor. In this system speed measurement and brake status of vehicle are measured. For speed measurement Hall Effect sensor is used. For detecting the brake status of the vehicle, a simple switch is placed below the brake pedal. The switch gives high output when force is applied on the brake pedal and vice versa. These two measured parameters are transferred to the third node via CAN bus. Each node has its particular identification field which will be used for prioritization of the node. Detection brake status has higher priority than speed measured.

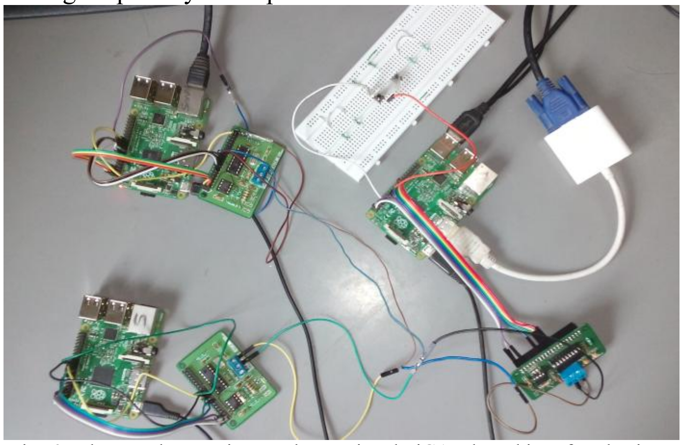
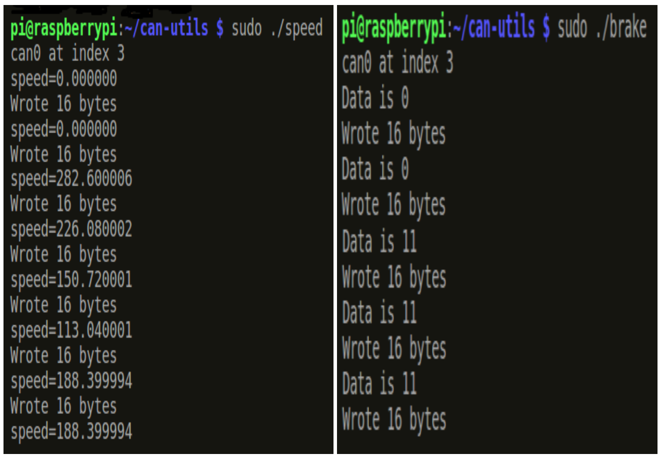


Fig. 6: Three nodes consist Raspberry Pi and PiCAN board interfaced using CAN bus

The PiCAN board is designed using Eagle CAD software. This board consist MCP2515 and MCP2551 which act as an interpreter between Raspberry Pi and CAN Bus. The two nodes transmit status of speed and brake of vehicle to the third node which is displayed on the terminal of third Raspberry Pi shown in figure 7. In figure 7(a), the status of brake from node1 is transmitted and displayed on terminal window of Raspberry Pi of node1. If brake is applied it shows data is 1 and transmitted one byte data i.e. 11 along with node identifier on CAN bus otherwise it will shows 0 if brake is not applied and transmitted one byte data i.e. 00 on CAN bus. Figure 7(b) shows the speed of the vehicle in meter per second which measured from node 2. For speed measurement of vehicle Hall Effect Sensor is used. It detects the magnetic field. The magnet is placed on the wheel of vehicle and the sensor is fixed in parallel with the magnet. When wheel complete its one rotation, the sensor detects the field for ones. Revolution of wheel is measured for one second and multiplies with circumference of wheel (radius in meter) we will get speed in meter per second.

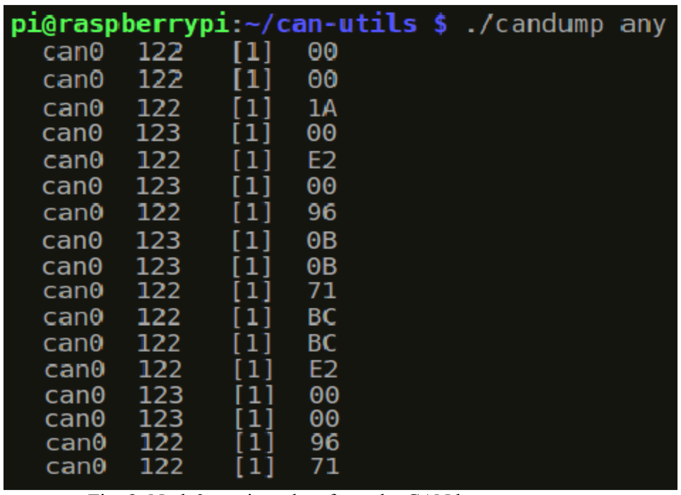


```
pi@raspberrypi:~/can-utils $ sudo ./speed
can0 at index 3
speed=0.000000
Wrote 16 bytes
speed=0.000000
Wrote 16 bytes
speed=282.600006
Wrote 16 bytes
speed=226.080002
Wrote 16 bytes
speed=150.720001
Wrote 16 bytes
speed=113.040001
Wrote 16 bytes
speed=188.399994
Wrote 16 bytes
speed=188.399994
```

```
pi@raspberrypi:~/can-utils $ sudo ./brake
can0 at index 3
Data is 0
Wrote 16 bytes
Data is 0
Wrote 16 bytes
Data is 11
Wrote 16 bytes
Data is 11
Wrote 16 bytes
Data is 11
Wrote 16 bytes
```

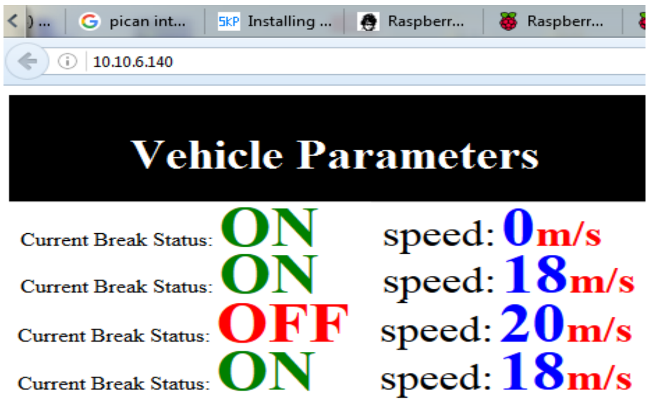
Fig. 7(a): Node1 and 7(b) Node2 send data over CAN Bus

The third node receives the data from CAN bus. The data on CAN bus is brake status of node 1 and speed of node 2 along with its unique identifier. The figure 8 shows received data with identifier and one byte data. As well as the third node act as a server. On the third node the database is created which stores identifier and one byte data of node 1 and node 2. This stored can be accessed remotely using a web page shown in figure 9.



```
pi@raspberrypi:~/can-utils $ ./candump any
  can0  122   [1]  00
  can0  122   [1]  00
  can0  122   [1]  1A
  can0  123   [1]  00
  can0  122   [1]  E2
  can0  123   [1]  00
  can0  122   [1]  96
  can0  123   [1]  0B
  can0  123   [1]  0B
  can0  122   [1]  71
  can0  122   [1]  BC
  can0  122   [1]  BC
  can0  122   [1]  E2
  can0  123   [1]  00
  can0  123   [1]  00
  can0  122   [1]  96
  can0  122   [1]  71
```

Fig. 8: Node3 receives data from the CAN bus



Vehicle Parameters

Current Break Status: ON speed: 0m/s
Current Break Status: ON speed: 18m/s
Current Break Status: OFF speed: 20m/s
Current Break Status: ON speed: 18m/s

Fig. 9: Vehicle parameters display on Web Page

VII. Conclusion

Using Raspberry Pi to implement a CAN bus system works. The system can be used for accessing vehicle internal ECU parameters. It controls/hack the vehicle from remote place. Connecting two or more nodes in the system does not cause any problem. The system works at a high speed up to 1Mbps, So that it will cause latency problem with Raspberry Pi when transmitting frames. One can develop its own network application using the above system such as home automation using CAN bus; light control network using CAN bus etc.

References

[1] HaiPing Sun, Hong Zeng and JiaLi Guo, “Bus Data Acquisition and Remote Monitoring System Based on CAN bus and GPRS,” international conference on CECNet 2011, Pages: 1094 – 1097

[2] Dai Qiang Wang, ShiYou Gao, Yu Qing Chen and Yi Wang, “Intelligent control system based on CAN bus for car doors and windows”, 3rd International Conference on ASID Year: 2009

[3] Microchip Technology's MCP2515 and MCP2551 developed by microchip on Oct, 2007, www.microchip.com/downloads/pdf

[4] Software installation on raspberry pi by Sk Pang: skpang.co.uk/blog/archives/1165

[5] Controller Area Network protocol family: Socket CAN implemented by Volkswagen Research, https://www.kernel.org/doc/ Documentation/networking/can.txt