Vehicle Simulator: Virtual Trip Runner (VTR)/ Real Time Recorder (RTR)

Aatray Kumar Singh
Department of Electronics & Communication
University of Pune
Pune, India
aatraysingh85@gmail.com

Ashish Nanaware
Department of Electronics & Communication
University of Pune
Pune, India
ashishnanaware310@rediffmail.com

Satyajit Pangaonkar
Department of Electronics & Communication
University of Pune
Pune, India
sapangaonkar@entc.maepune.ac.in

ABSTRACT-This topic is an attempt to develop an open source vehicle simulator for use by anyone needing realistic vehicle data delivered as it would be in a real vehicle. As you probably know, most vehicles nowadays have an On-Board Diagnostic (OBD) connector which is wired up to the car's internal computer. It is used in many garages where the mechanic can probe the car through the OBD connector and read out parameters onto a display. In an Internet of Things (IoT) world where the car can be easily connected to the Internet (perhaps via Bluetooth to a smart phone), it could automatically search an online knowledge base and not only report a fault code but also let you know the most likely cause either based on the car's personal history or on the environment (an expert system could conclude "it is minus 15 degrees Celsius outside, and there is a water leak, and it is likely to be a cracked hose due to the cold temperature – and there was a manufacturer recall notice concerning this hose"). There will be whole sectors of applications such as, safer driver operation of the vehicle, fewer distractions, more automation, safer mechanical performance, better & more timely maintenance warnings, lower cost operation and lower maintenance costs.

Keywords- On board diagnostics, Raspberry Pi Kit, CAN Protocol, Real trip recorder, Virtual trip runner.

1. INTRODUCTION

This topic is to help explain a new project which could be extremely useful for the community. If you've ever been stuck in a traffic jam, or broken down, or wanted to know the health of your car, and wanted to do something about it, then this open source project will resonate with you. When thinking about how the internet-of-things will evolve, it is easy to predict vehicles will be a significant focal point for consumer applications, since vehicles currently represent by far the biggest technology expenditure/investment by the average consumer. Vehicles are very complex machines incorporating multiple Microcontrollers communicating via on-board busses. The level of electronic technology in vehicles has exploded and there is a large amount of vehicle data readily available on the diagnostic bus which is used locally, but not yet used for the vast number of potential external applications.

There will be whole sectors of applications such as safer driver operation of the vehicle, fewer distractions, more automation, safer mechanical performance, better & more timely maintenance warnings, lower cost operation and lower maintenance costs, better and more timely driver information, better advanced traffic warnings and route planning. The paper is organized as follows: section 2 provides reader with a initial simulator concept of the project. Then section 3 is dedicated to the objectives of this project. Then section 4 is dedicated to the architecture of the vehicle simulator. Then in section 5 performance of this project is explain in details. Then section 6 provides the results of this project. . Eventually in section 7 conclusions is drawn and also future research are discussed.

2. INITIAL SIMULATOR CONCEPT

Initially the system will need to be able to provide OBD2 data just as a real vehicle would in response to queries on a CAN bus or K-Line interface. To ensure realistic data, trip data will be collected from real vehicles the project will include a data collection system (Real Trip Recorder - RTR) as well as a playback / simulation system (Virtual Trip Runner - VTR). During a simulated trip, previously recorded data can be interpolated to provide appropriate data to any bus queries.

Fault code simulation will be initially implemented on a best guess basis as we will not be able to create and monitor real faults in real vehicles. An attempt will be made to keep the software modular enough that it won't be difficult to replace fault code simulations with better simulations when more is understood about their behavior and also to replace recorded trip scenarios with computed simulations, if and when they get developed.

In order to make a compact open system that can be easily setup for testing or demonstration at any location the Raspberry Pi has been chosen as the platform. This will allow a data collection system (RTR) to collect trip data from a real vehicle, then a simulator system (VTR) can simulate this trip for the same data collection unit to see if it collects the same data from the simulation as it did from the real trip. This dual system provides an easy way to validate the system. A single system will be able to function as either a trip recorder or a simulator.

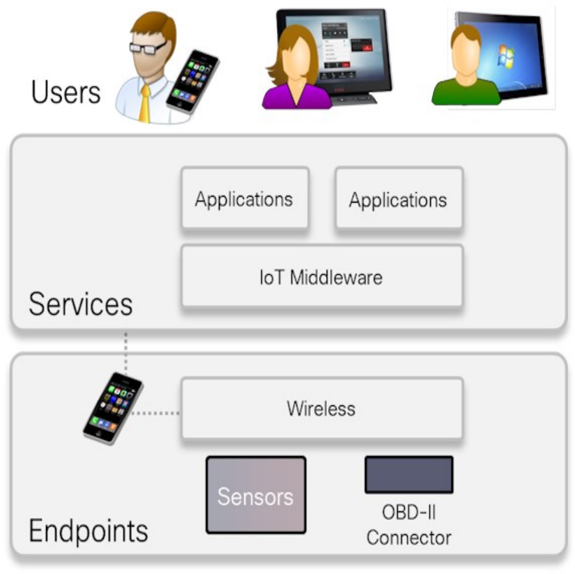

Fig1. Block diagram of Initial simulator concept

3. OBJECTIVES

There are mainly four objectives that we are concluding for this project which are as follows:

- The primary objective for this project is to provide a vehicle simulator system that provides realistic enough data, primarily in electronic form, to allow developers to test their OBD2 reader prototypes & data applications in a lab instead of needing to test in a vehicle. The VTR system should provide some ability to generate fault codes.
- To perform real-time vehicle status surveillance, i.e. to monitor engine rpm, vehicle speed, coolant temperature, fault codes, and other vehicle dynamics information
- A secondary objective is to make the vehicle simulator such that it can be used to demonstrate new products and applications indoors at trade shows or sales presentations.
- A third objective is to make the vehicle simulation system a modular platform that can be extended and upgraded to become a more accurate simulation for more vehicle.
- To transmit vehicle information to the user GSM wireless network for fault analysis as well as to alert user.

- To decode OBD system installed on the vehicles

4. ARCHITECTURE

The architecture of our system is discussed below in a more physical level.

4.1 System Design

OBD2 data provides a lot of information about what is occurring inside vehicle systems, but it does not necessarily provide everything desired in a simulation, such as location and orientation of the vehicle, distance travelled, and time stamps on all data.
The Raspberry Pi has a CAN bus capability, but a custom cape will be needed to translate this to OBD2 signals and add in a K-Line interface. This cape will also include a GPS module and a 10channel sensor suite - 3 orthogonal linear accelerometers, 3 orthogonal angular rate sensors, 3 orthogonal magnetometers and an absolute pressure sensor. These sensors are not needed in the simulator system (VTR), but will allow more sophisticated sensor data to be collected when the device is used for recording real trips (RTR) and this data can later be presented by the simulator (VTR). The VTR/RTR cape will also include a Bluetooth module - when in VTR mode it will allow generation of fault codes and control of the VTR from a smart phone, when in RTR mode it will allow connection to a Bluetooth OBD2 interface or possibly a smart phone.

Since the RTR will have a full OBD2 interface, either wired or wireless, there is no reason it couldn't be used to reset fault codes, however this feature will likely not be included in the first software release.

4.2 Real Trip Recorder (RTR)

Real trip recorder is a data collection system which is used to collect the trip data from real vehicles. It is an automotive engineering device which is used to record the status of the vehicles running speed, distance travelled. It is also indicates the mileage covered during a particular journey either mechanically or electronically, it can be reset to zero by turning or pushing the button.

Fig.2 shows the architecture of the system that we are going to implement in this project. The vehicle is directly connected to the OBD 2 Interface, the OBD2 connector is directly connected to OBD2 CAN connector, and then the CAN connector is connected with Microcontroller via CAN protocol bus or K-line interface. Then a GSM-Module and a display is connected with the Raspberry Pi kit. The display shows the results of the faults and a mobile is connected with the GSM module which receives the alert of that fault.

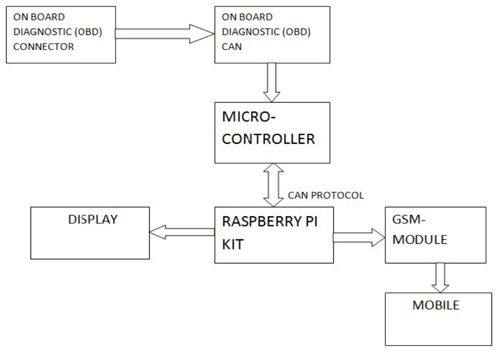

Fig.2 System architecture including Raspberry Pi Kit

4.3 Virtual Trip Runner (VTR)

Virtual trip runner (VTR) is a data collection system which is used to store trip data as well as playback and simulation system. During a simulated trip, previously recorded data can be interpolated to provide appropriate data to any bus queries. The data that is collected by the real trip recorder (RTR) system is then by using a simulator system virtual trip runner (VTR) can simulate this trip for the same data collection unit to see if it collects the same data from the simulation as it did from the real trip.

This dual system provides an easy way to validate the system. A single system will be function as either real trip recorder (RTR) or virtual trip runner (VTR) means either a trip recorder or a simulator.

4.4 Trip Video

Initially it should be easy to record video of a trip while electronic data is being collected using a separate video camcorder. Having the simulator play it back in proper sync is a bit of an unknown right now, but obviously it could be played on a separate player while the simulation is running. Ultimately it may be possible to record and playback video on the same Raspberry Pi platform, but this functionality is not a target for the first release.

4.5 Raspberry Pi Kit

Fig. 3. Raspberry Pi Kit (Source: Internet)

In this topic we are going to use Broadcom BCM2835 SoC Multimedia processor in Raspberry Pi Kit.

- CPU
  1. ARM 1176JZF-S (armv6k) 700Mhz.
  2. RISC Architecture and low power draw.
  3. Not compatible with traditional PC software.
- GPU
  1. Broadcom Video IV
  2. Specialized graphical instruction sets.
- RAM
  1. 512MB (Model B rev.2)
  2. 256MB (Model A rev.1)
- HDMI
  1. Digital signal
  2. Video and audio signal
  3. DVI cannot carry audio signal
  4. Up to 1900x1200 resolution
- IEEE 802.11 Wi-Fi, frequency band of 2.4Ghz and 5Ghz and Ethernet (IEEE 802.3)
- General purpose Input/output (GPIO), micro-USB power connector, powered USB hub.

Configuration of Raspberry Pi includes RPi doesn't have a BIOS menu. It relies on text files containing configuration strings that are loaded by the chip when powers on.

- Hardware settings: config.txt
- Memory partitioning :start.elf
- Software settings: smdline.txt

5. PROJECT PLAN

- Study how OBD works and get the OBD connector.
- Analyze output of the OBD via CAN analyzer.
- Interface CAN base micro controller to OBD.
- Send decoded data to Raspberry Pi for further processing & monitoring
- Interface GSM modem to the Raspberry Pi.
- Write the RTR data collection firmware.
- Write VTR simulation firmware.
- Test and validate system.

To achieve this project plan both vehicle simulator and vehicle recorder systems have to build and they communicate with each other fine over the CAN bus. The car cape that we are displaying in the block diagram of both the trip recorder system and simulator system is need to running on Raspberry Pi. In order to implement this project plan we have to make sure that the CAN based Microcontroller and Raspberry Pi Kit communicate with each other. They can communicate fine with the USB-CAN module above. The commercial reader can also communicate fine with the USB-CAN module. In this project we are using two types of OBD2 connectors which are OBD2 (male) and OBD2 (female). The OBD2 male is connected with the vehicle and OBD2 female is connected with OBD2 reader, and these two connecters are directly connected with a 5v power supply.

6. RESULTS

This simulator looks like an operating vehicle to any instrumentation or application that can interface to a normal vehicle via its OBD2 connector. The data supplied by the simulator is real trip data collected by the Trip Recorder function of the system. There is additional sensor data available that is collected directly from the Car Cape sensor suite during the same reference trips, such as GPS position, compass heading, accelerations, angular rates, barometric pressure, temperature and ambient light.

The system is built around a Raspberry Pi Kit with a "Car Cape" mounted on it to provide an OBD2 interface, a sensor suite, a user interface and a wireless communications interface. The system may be used as either a simulator or as a stand-alone trip recorder.

Since the introductory blog, we have been busy designing hardware, ordering parts and building hardware. We have also been setting up a Raspberry Pi Board development environment and configuring an operating system build and experimenting with CAN bus communications.

We have built 3 capes with CAN bus capability and a suite of functionality appropriate for our vehicle applications. The various features are outlined in bullets around the Car Cape image below. The software and hardware are currently functional enough for Raspberry Pi Kit, with installed capes, to communicate over the CAN bus. Not shown are a separate keypad card and a separate OBDII connector card which also has a DC-DC converter to supply 5 volts to the Raspberry Pi. These cards are built, but not fully tested yet. All those peripherals required most of the available pins on the Raspberry Pi, although I refrained from using the pins allocated to the Raspberry Pi internal HDMI cape - so HDMI should still be functional.

7. CONCLUSION AND FUTURE RESEARCH

This topic is to help explain a new project which could be extremely useful for the community. In future there will whole sectors of application such as accident response application, ways to save fuel, vehicles better designed to address owner needs and habits. The list goes far beyond the few obvious sectors listed here, but clearly there will be an enormous number of vehicle-related internet-of-things applications. The RTR and VTR are two devices that we are used for collecting real data from the vehicles and simulate them using a platform that is Raspberry Pi board. In this project we are using two types of OBD2 connectors which are OBD2 (male) and OBD2 (female). The OBD2 male is connected with the vehicle and OBD2 female is connected with OBD2 reader, and these two connecters are directly connected with a 5v power supply. After doing these things correctly we should able to get the results such as version of the CAR-CAPE 2, Temperature of the engine and pressure of tires.

8. REFERENCES

[1] D.Nandhini, G.Nandhini, M.Nandhini, R.Vidhya, “ON-BOARD DIAGNOSTIC SYSTEM FOR VEHICLES”, IEEE March 2014.

[2] LiShiwu, Coll. of Transp,JilinUniv.,Changchun “Research on method for real-time monitoring vehicle based on multi_sensors” china, 2011.

[3] UmitOzguner, Christoph Stiller, Keith Redmill,” System for safety and autonomous behavior in cars” IEEE Feb 2007.

[4] Website. http://www.element14.com/

[5] James William Topliss, Victor Zappi, Andrew Mcpherson, “Latency performance for real time audio on Beaglebone black, England 2013.

[6] Jorge Zaldivar, Carlos T.Calafate, Juan Carlos Cano, Pietro Manzoni,” Providing accident detection in vehicular networks through OBD2 devices android based smart phones. IEEE 2011.

[7] International Organization for Standardization, “ISO 15765: Road vehicles, Diagnostics on Controller Area Networks (CAN),” 2004