Instituto Tecnológico
y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

Especialidad en Sistemas Embebidos

Automotive cluster based on Raspberry Pi 3 B+

TRABAJO RECEPCIONAL que para obtener el **GRADO** de
ESPECIALISTA EN SISTEMAS EMBEBIDOS

Presentan: **MANUEL ALEJANDRO ALAYON SAUCEDA**
ADRIAN MOISES COLLAZO VELAZQUEZ

Asesor: **M.SC. HÉCTOR ANTONIO RIVAS SILVA**

Tlaquepaque, Jalisco. julio de 2019.

Acknowledgments

First, we want to thank to our families who supported us not only during this project, but also during all the program. In general, our families provided us their unconditional trust and gave us the opportunity to successfully accomplish the goal of finishing this program.

We also want to thank ITESO for the Specialization Program in Embedded Systems, which provided all the tools and knowledge we needed for the development of this project, including the infrastructure of its laboratories that were essential to conclude this work. We want to thank our teachers from the Specialization Program as well, since the knowledge they shared with us was so important for this success. In particular, we want to mention M.Sc. Hector Antonio Rivas Silva who gave us his guidance in support for the development of this project.

Finally, we sincerely thank CONACYT for the scholarship granted (930406), since that was very important to accomplish this work.

Abstract

The automotive cluster is one of the most important devices in a car and is the interface between the car and the driver. However, the development of this Electronic Control Unit (ECU) used in the automotive industry is complex and expensive. The Raspberry Pi 3 is a lower cost board that can be used as an alternative board due to its versatility and ease of use. Therefore, the aim of the project is to develop an automotive cluster using the Raspberry Pi 3 as the main board, and be able to connect it to a CAN network. Buildroot was the main tool used in this project allowing to generate a customized Linux-based Operating System (OS). Furthermore, the project included the CAN-utils package and SocketCAN Application Programming Interface (API) to receive CAN messages, through a PiCAN2 module, and display the animation according to the message. Also, the interface was created using Qt5 IDE, so it was essential to add those Qt5 packages to the Raspberry Pi 3 as well. Finally, this project was configured to work with a baudrate of 100 Kbps. In conclusion, the proposed project is a practical choice for low cost hardware and free software automotive cluster, also it is user-friendly and interactive.

Introduction

One of the most important devices in a car is the instrument cluster, also known as the dashboard. This electronic control unit (ECU) is the interface between the car and the driver, so the driver can witness relevant information regarding the vehicle's components and features like speed, tire pressure, and engine status, among others. At the beginning of the 20th century, the automotive cluster was robust and consisted of a group of gauges unified in a single case. Nowadays, clusters have evolved from analog gauges to digital displays, such as the Liquid Crystal Display (LCD), providing more accurate information and reducing its size. Since 1986, to provide control and communication for ECUs, the Controller Area Network (CAN) communication protocol created by Bosch, started to be used in the automotive industry; thus, clusters were upgraded to receive information from the other ECUs using this protocol.

Many companies, like Bosch and Continental, and foreign investigators invest on the design and development of clusters [1]. In particular, the automotive industry has a major impact in Mexico; however, it is more focused on manufacturing instead of research. Furthermore, the development of clusters requires infrastructure, an environment where this device can be tested, and materials, including an embedded system, a screen, connectors, and others that increase the cost of the ECU.

Consequently, the purpose of this project is to design and develop an automotive dashboard that uses a low-cost board – Raspberry Pi 3 B+ -- and a free Operating System (OS) based on Linux. The software was developed using Qt libraries for the user interface, which is displayed on an LCD. Also, since Raspberry Pi 3 does not have a CAN transceiver installed, it was necessary to add an external transceiver to be able to receive data frames from other ECUs. Finally, this project was developed using the network infrastructure provided by the automotive laboratory from the Instituto Tecnológico de Estudios Superiores de Occidente (ITESO), where the communication tests were performed..

1. Methodology

This project was developed according to the V-Model, a software development methodology that begins with design stages, then, the development stage, and finally, test stages that evaluate each design stage [2]. The defined stages for this project were: system requirements, software requirements, coding, software test, and system test.

Based on the previous development of an embedded graphic user interface (GUI) from the Department of Electronics, Systems, and Informatics at ITESO [3], the project started with a hardware upgrade consisting of a new version of the board -- Raspberry Pi 3 B+ -- that included the addition of the CAN module (PiCAN 2) and a 7-inch LCD screen as represented in Fig. 1-1.

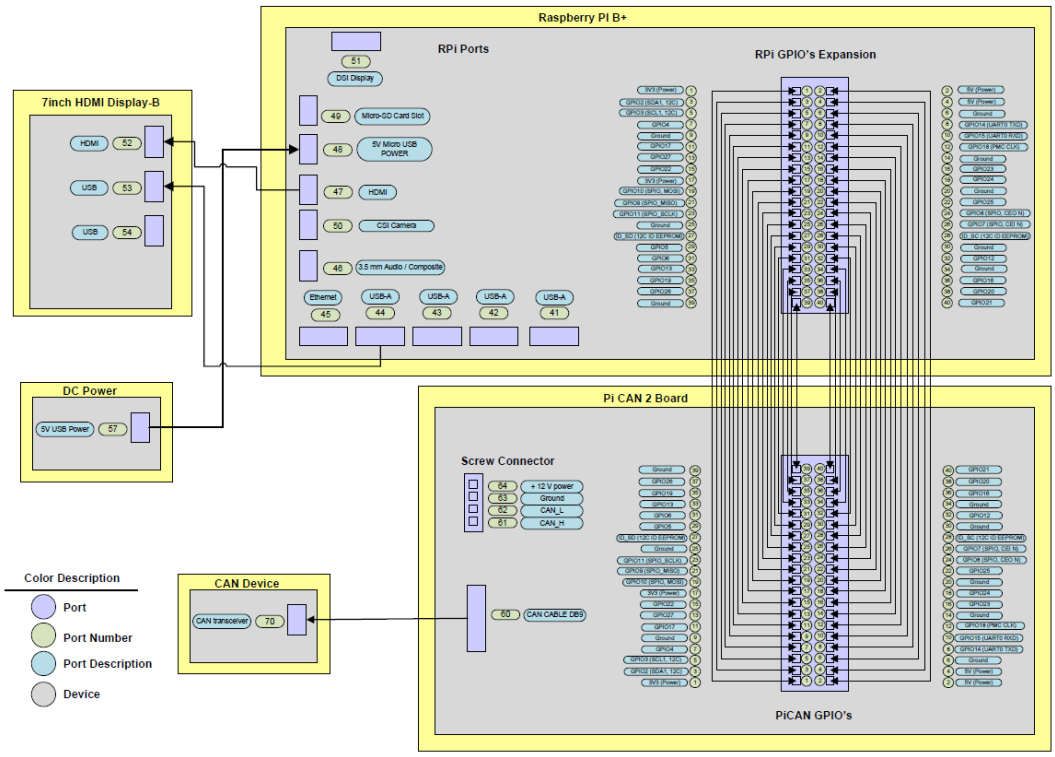

Fig. 1-1 System architecture of the automotive cluster.

The OS was developed using cross-compiling with Buildroot. a tool used to design Linux OS for different architectures. The main libraries installed in the file root system were Busybox, GCC 7.4, Eclipse plugin, Qt5, CAN-utils; the initialization system was performed using Busybox. Also, CAN-utils package was used to enable the CAN driver, through the PiCAN 2 transceiver.

The CAN driver was accessed using the SocketCAN library, an Application Programming Interface (API) used in Linux, which enables CAN communication using a socket port, emulates the functions of a Transmission Control Protocol/Internet Protocol (TCP/IP) [5]. The middleware application used API to receive CAN messages and update GUI within a CAN bus speed of 100 Kbps as shown in Fig. 1-2. Furthermore, the performance starts after booting, so the user does not need to perform any task to initialize the program.

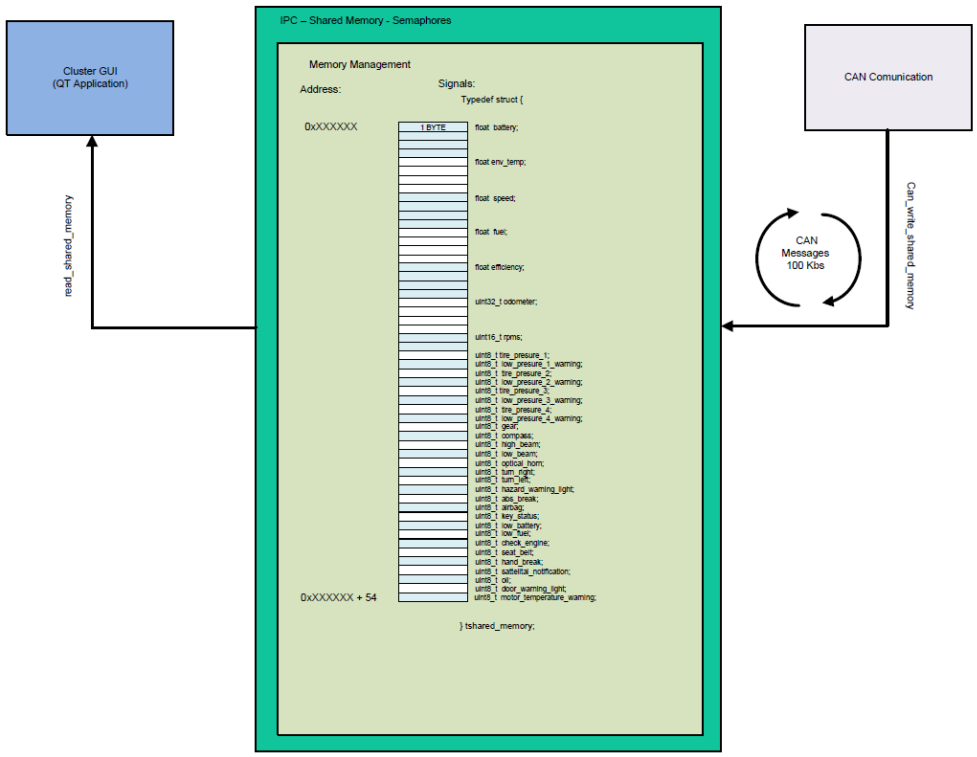

Fig. 1-2 Communication between the cluster GUI and the middleware application.

Furthermore, the cluster uses several messages to perform the specific animation of the indicators, like moving the gauge of the speedometer or turning on the door warning alert. There are three different IDs defined for this project that enable or disable the corresponding gauge or alert, as can be seen in TABLE I.

TABLE I

CAN COMMUNICATION MESSAGES.

		Bit							
CAN ID	**Byte**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0x010	**0**	**NA**	Fuel Level				High Beam	Lights ON	Low Beam
	1	Speedometer							
	2	Tachometer							
	3	**NA**	**NA**	Gear position			Battery voltage		
	4	Fuel Efficency							
	5	**NA**	**NA**	Hazard light	Left light	Right light	Compass		
	6	Check Engine	Enviroment Temperature						
	7	**NA**	Sate	Door	Seal	Motor	OIL	Air	ABS
0x020	**0**	Odometer							
	1								
	2								
	3	Tire 1 pressure							
	4	Tire 2 pressure							
	5	Tire 3 pressure							
	6	Tire 4 pressure							
0x271	**0**	**NA**	**NA**	**NA**	**NA**	**NA**	Key Engine		

2. Results

First, the cluster initializes when turned on, executing the Kernel, and the main libraries display on the GUI as shown in Fig. 2-1. Then, the program is executed with a dark screen until the user sends the engine signal to make the cluster visible, as shown in Fig. 2-2. Subsequently, the indicator cluster reacts with the corresponding signal when connected to an ECU CAN device that sends CAN messages.

Fig. 2-1 Kernel initialization.

Fig. 2-2 Automotive cluster engine on.

Fig. 2-3, shows the GUI interactive animation characteristics: 1/8 of fuel level gauge, 100 km on speedometer gauge, 2000 RPMS tachometer gauge, 500 km of odometer, current hour, hazard light on, high beam light on, and oil warning on, where these animation depend on corresponding CAN messages. The testing was performed using CANoe to visualize CAN messages. In Fig. 2-4, the messages received by the cluster are displayed using CANoe software.

Fig. 2-3 Automotive cluster motion animation.

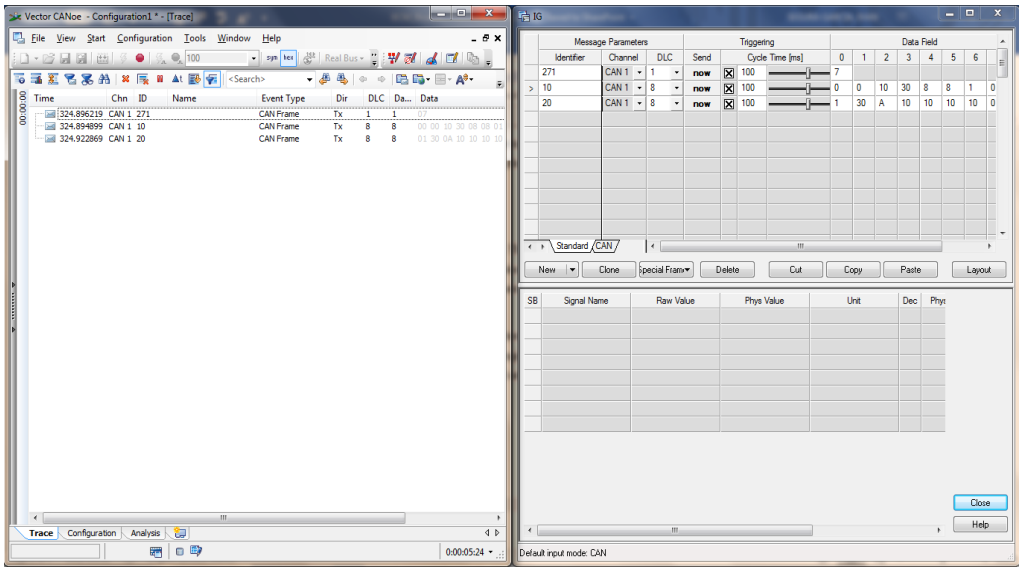

Fig. 2-4 CAN messages between the automotive cluster and CANoe.

Finally, the cluster is included in a hard case, which consists of the LCD, Raspberry Pi 3 with its respective CAN module (PiCAN 2), and several switches and keys that send CAN messages to the cluster, emulating a CAN network.

\

Conclusion

The embedded system for the automotive cluster proposed in this project included a Raspberry Pi 3 board, a PiCAN 2 module, and an LCD. These elements were successfully connected to receive CAN messages with the module and transfer the data to the board using SocketCAN API. Lastly, the received information was processed according to a defined frame and displayed on the GUI display. The cluster was designed to work with networks at 100 Kbps. Furthermore, the project fulfilled the V-Model structure as the tests complied with the defined requirements.

This project could be basis of future works to create different lists of CAN messages and interact with other ECUs with CAN communication. In addition, this communication system, can be commercialized to different final costumers using their own CAN messages; therefore, the price of the ECUs used this project was lower than current ECUs used in the automotive industry.

References

[1] J. Pešić, K. Omerović, I. Nikolić, M. Z. Bjelica, “Automotive Cluster Graphics: Current Approaches and Possibilities”, in International Conference on Consumer Electronics-Berlin, 2016, pp.1.

[2] K. Forsberg, H. Mooz, H. Cotterman, Visualizing Project Management: Models and Frameworks for Mastering Complex Systems, 3rd edition, New Jersey, 2005.

[3] R. Camacho, “Embedded Graphic User Interface for Automotive Cluster”, Trabajo de obtención de grado Especialidad en Sistemas Embebidos, Instituto Tecnológico y de Estudios Superiores de Occidente, Tlaquepaque, JAL, Mexico, 2017.

[4] Buildroot, "The Buildroot user manual", 2019. [Online]. Available: https://buildroot.org/downloads/manual/manual.html

[5] M. Kleine-Budde, “SocketCAN -The official CAN API of the Linux kernel”, in IEEE International Conference on Communications, 2012, pp. 05-17, 05-19.