Embedded Systems and Vehicle Networks

Bassam Mahmoud Abdelwahab
Military Technical College, Egypt, Bassam11459@gmail.com

Supervisor: Mostafa Ismail Yacoub, Associate Professor
Automotive Engineering Department, Military Technical College, Egypt, mostafa.yacoub@mtc.edu.eg

***Abstract*– The embedded systems are widely used nowadays. Like other mechanical systems, automotive systems have been developed to include a considerable amount of embedded systems. The present article provides an overview of the embedded systems and their role in industry. Starting from definition, usage, requirements, types and components, the embedded systems are explained in a nutshell. Also, in this paper, some tools that are used in embedded systems testing and evaluation are explored, like Arduino and Raspberry Pi, while stating the differences between microcontrollers and microprocessors. In addition, suitable methods for using the embedded systems along with the required procedures to develop an embedded system are discussed. The embedded systems in automotive applications are highlighted referring to the currently used protocols and networks. Classification of communication protocols used in automotive systems is discussed. As a case study, a suitable network/protocol is illustrated explaining its capabilities and usage. The present article is a simplified guide for mechanical engineers interested in automotive research.**

***Keywords*-- embedded systems, microcontrollers, protocols, automotive networks, vehicle control.**

I. Introduction

We are in the age of technology, which is developed continuously. It's clear that embedded systems have an effective role in developing and updating the technology. Since the 1970s [1], there is a noticeable increasing in the usage of electronic systems instead of mechanical, pneumatic or hydraulic systems [1]. As embedded systems are used for manufacturing many modern equipment like: cars, mobile phones, avionics, smart homes, firefighting systems and many other systems, they seem flexible to develop. They depend on coding creatively in addition to circuit design to achieve the required functionality. So, before using them you should learn how to write codes (coding) and the circuit design. Embedded systems started in the 20th century then the dependency on it has been increased at 21st century as they become the kernel for many modern equipment [1]. During the studying of embedded systems, you must integrate two things: hardware and software, to be able to get the required product. Secondly, we will focus on the usage of embedded systems in vehicles. Then, overviewing the networks and protocols used in vehicles.

The remainder of the paper is organized as follows: section 2 provides an overview on the embedded systems. Section 3 demonstrates the classification of embedded systems. Section 4 describes two simplified practical applications for the embedded systems. Section 5 illustrates the embedded systems in automotive applications and finally, section 6 concludes the paper.

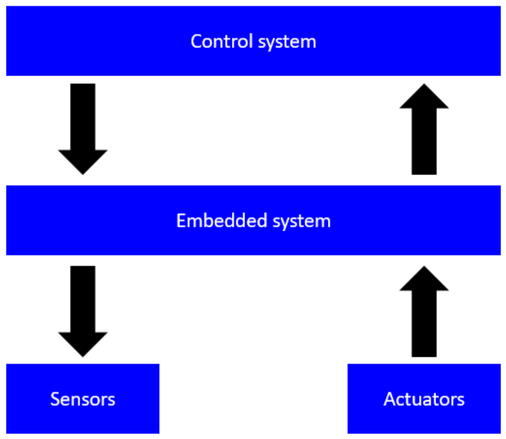


Fig. 1 A schematic diagram for the embedded systems working sequence.

II. Embedded Systems Overview

The embedded system is considered a computer in a chip. It contains computer's hardware and software. It may be dependent, a part of a large system or independent.

There are some popular definitions for embedded systems:

- It is any device which has a programmable computer without being a general purpose computer.[2]
- The embedded system includes a microcomputer having electrical, chemical, mechanical characteristic. It is programmed for a specified job and considered as one system [3].
- The embedded system is electronic system having a microprocessor or microcontroller without considering that system as a computer. The computer is embedded in that system [4].

Components of embedded systems are describe in the following subsections.

A. *Hardware*

The embedded hardware contains the following:

1) Power Management: It contains the power supply and additional control to for enabling many different power modes [5].

2) Embedded Processor: It is the heart of any microcontroller based embedded system [5].

3) Embedded Memory: It should have low access time and high density. As the embedded system programs are small, we don't need a virtual storage [5].

4) Peripherals and I/O: They are the input and output devices connected to the serial or parallel ports of the embedded system [5].

5) Timers and Watchdog: To regulate time events, a microcontroller should have different timers [5].

6) Sensors and Analog: should have many sensors like temperature sensor and analog modules like Analog to Digital converter (ADC), Digital to Analog converter (DAC), and Operational Amplifiers for signal conditioning and improved sensing [5].

7) Interrupt Controller: It tries to produce low latency response [5].

8) Clocking and Reset: It should have a number of clock options including external crystals and internal oscillators, to allow the system to run at low power/during quick start up [5].

9) Application Specific: It should have application specific logic as part of microcontroller or microprocessor [5].

B. Application Software

1) Main Function: It must be provided to manage the system to do the required job. It is produced by coding using a suitable programming language like C/C++. Program codes are located in embedded Flash or ROM [5].

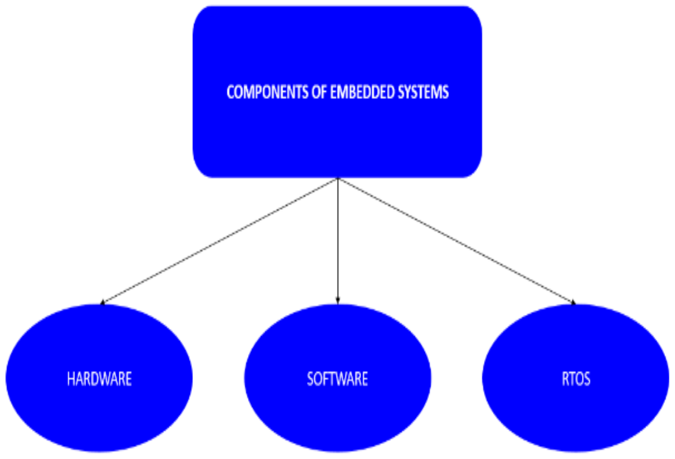


Fig. 2 Components of the embedded systems.

2) Real Time Operating System (RTOS): It contains a scheduler that manages many tasks in the embedded systems. An RTOS should arrange tasks and interrupt services such that they are completed within their deadlines [5].

The following table shows a comparison between the microprocessor and the microcontroller [5].

TABLE I
THE MAIN DIFFERENCES BETWEEN MICROCONTROLLERS AND MICROPROCESSORS.

Microcontrollers	Microprocessors
It is considered as a computer with all components.	It needs other components to produce a complete computer.
It can manage and do a specified job by itself.	It must integrate in a large system to do a specified job.

The embedded systems are commonly used as they are considered practical and simple alternatives to computers but with low abilities. They are characterized by their low consumption for electricity. They are invisible for the user. Briefly we can say that they are simple computers used for limited purposes with high accuracy.

The advantages of embedded systems are: they have small size, cheap, have various types so they can be used for many purposes, and have many input and output pins so they can be connected with many components.

The disadvantages of the embedded systems are: they have small memories, almost doing only one job, weak so it can be easily damaged, needs great efforts to be manufactured or developed.

The following are the characteristics of embedded systems:

– Reliability: it's the ability of the system to do its function job during the mission [6].
– Fault-tolerance: it's the ability of the system to correct faults rapidly [7].
– Real-time: it's the ability of the system to its function job at the correct time [7].
– Flexibility: the ability of the system to be repaired easily [7].
– Portability: the ability of the system to work in more than one environment without affectingly changes [7].
– Security: the transmitted data must be secured to avoid catastrophes occurrence [6].
– Safety: the final product which is made using the embedded systems must not be harmful for humans [6].

There are many other characteristics like: scalability, upgradeability and others.

It's developed by writing a code which is designed for a specified job then connecting the components including the used embedded system correctly. We can summarize the development process through the following steps:

1) Define the function that you want to make.

2) Choose the suitable microcontroller (PIC or MCP) for that purpose according its storage space, data transmission speed and other specifications.

3) Write the code that accomplishes the required function.

4) Connect the electronic circuitry correctly.

5) Secure the power source.

6) Run and test the system safely.

III. CLASSIFICATION OF EMBEDDED SYSTEMS

The following figure shows the types of embedded systems. The embedded systems are classified into two main types: based on the functional requirements and based on the microcontroller performance.

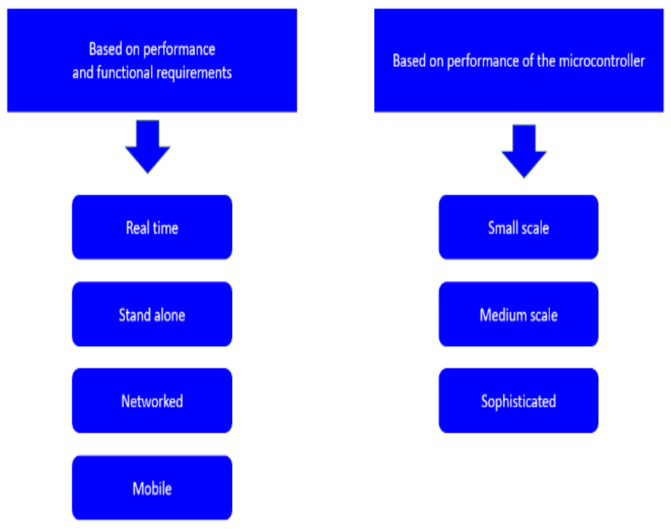

Based on performance and functional requirements
Real time
Stand alone
Networked
Mobile
Based on performance of the microcontroller
Small scale
Medium scale
Sophisticated

Fig. 3 Classification of embedded systems.

A. *Based on Performance and Functional Requirements*

The embedded systems could be classified, based on the functional requirements, into four types.

1) Real-Time Embedded System: It's designed to do specific work in pre-specified time. It enhances the energy consumption and predictability [8]. It has two types:

- Soft real time embedded system [5].
- Hard real time embedded system [5].

2) Stand-Alone Embedded System: It can work alone without a host system. It receives the inputs from sensors and produces decisions and sends them to actuators [9].

3) Networked Embedded System: It consists of multiple individual embedded systems that are integrated (connected) together using a single network to control the whole system. Although each of those multiple embedded systems may perform a specified task they share some data to help them accomplish their specific tasks. Collecting them in a single network is for the purpose of avoiding using redundant sensors or actuators that serve the same whole system [9].

4) Mobile Embedded System: They are systems with small size which are used in small devices. They are used in mobile phones and digital sensors. They may have memory, constraints and good user interface [10].

B. *Based on Performance of the Microcontroller*

The embedded systems could be classified, based on the performance, into three types.

1) Small-Scale Embedded System: It's created using an 8-bit microcontroller. Those microcontrollers can be battery activated [11].

2) Medium-Scale Embedded System: It's created using a 16-bit or 32-bit microcontroller. That system has a number of hardware and complex software. So, it's not preferred by many users [12].

3) Sophisticated Embedded System: It produces many functions on multiple algorithms. Those are composed of complex software and hardware. They need configurable processor and logic array which can be programmed [13].

For using embedded systems, the developer must have many skills and one of the most important skills is coding. We can write a code using programing languages like: C, C++, Python, MikroC and others [14].

IV. Examples on Embedded Systems

Since the most common used embedded systems by the hobbyists are Arduino and Raspberry, we will demonstrate a description for each.

A. *Arduino*

The language that you will choose for writing the code for Arduino is similar to the languages which is used for computer languages which depend on C/C++ [15]. It has (I/O) digital pins and analog pins which are used for input or output signals [15].

The components of Arduino UNO are:

- Microcontroller: ATmega328
- Operating Voltage: 5 V
- Input Voltage (recommended): 7–12 V
- Input Voltage (limits): 6–20 V
- Digital I/O Pins: 14 (of which 6 provide PWM outputs)
- Analog Input Pins: 6
- DC Current per I/O Pin: 40 mA
- DC Current for 3.3 V Pin: 50 mA
- Flash Memory: 32 KB of which 0.5 KB is used by the bootloader
- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB
- Clock Speed: 16 MHz

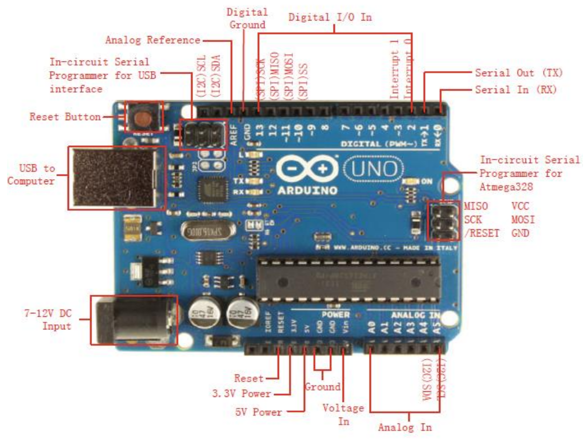


Fig. 4 Components of the Arduino UNO board.

In this example, the function of the embedded system is to measure the distance between an ultrasound sensor and an object in front of it. If the distance is less than 5 m, the LED will lit. Otherwise the LED will turn off.

1) Hardware and Connections: Connecting the pins as follows:

- (Vcc) of ultrasonic sensor to (5V) pin of Arduino.
- (GND) of ultrasonic sensor to (GND) pin of Arduino.
- (TRIG) of ultrasonic sensor to pin (12) of Arduino.

– (ECHO) of ultrasonic sensor to pin (11) of Arduino.
– (-) of the LED to (GND) pin of Arduino.
– (+) of the LED to (5V) pin of Arduino.

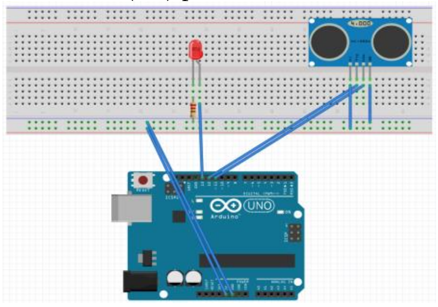
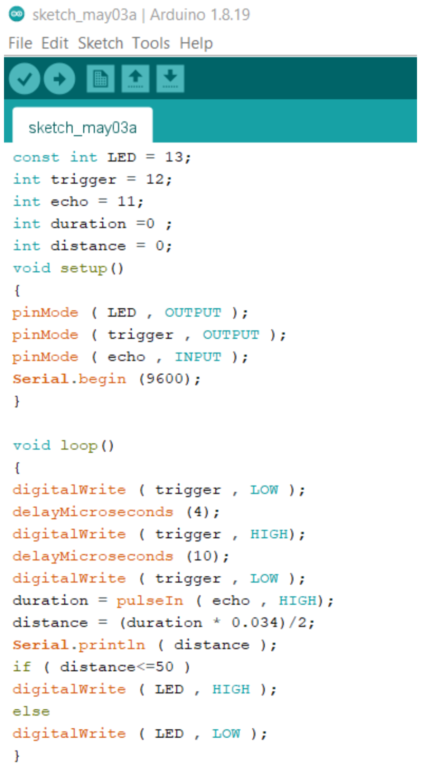


Fig. 5 Connections of the ultrasonic sensor and the LED with the Arduino board.

2) Coding: The following figure shows the details of the Arduino code to achieve the prescribed function.



```
const int LED = 13;
int trigger = 12;
int echo = 11;
int duration =0 ;
int distance = 0;
void setup()
{
pinMode ( LED , OUTPUT );
pinMode ( trigger , OUTPUT );
pinMode ( echo , INPUT );
Serial.begin (9600);
}

void loop()
{
digitalWrite ( trigger , LOW );
delayMicroseconds (4);
digitalWrite ( trigger , HIGH);
delayMicroseconds (10);
digitalWrite ( trigger , LOW );
duration = pulseIn ( echo , HIGH);
distance = (duration * 0.034)/2;
Serial.println ( distance );
if ( distance<=50 )
digitalWrite ( LED , HIGH );
else
digitalWrite ( LED , LOW );
}
```

Fig. 6 Arduino code of ultrasonic sensor and LED example.

B. *Raspberry Pi*

It is a computer that has the size of a credit card. The language for writing the code for Raspberry Pi is Python. The components of Raspberry Pi are [16]:

– 2 USB ports (1 USB port available in model A)
– Ethernet port (available only in models B and B+)
– RCA output
– HDMI output
– Audio output
– Low level peripherals, which include:
  - GPIO (General Purpose Input Output)
  - UART/Serial Port (Universal Asynchronous Receiver Transmitter)
  - I2C (two wire interface)
  - SPI (Serial Peripheral Interface) [16]

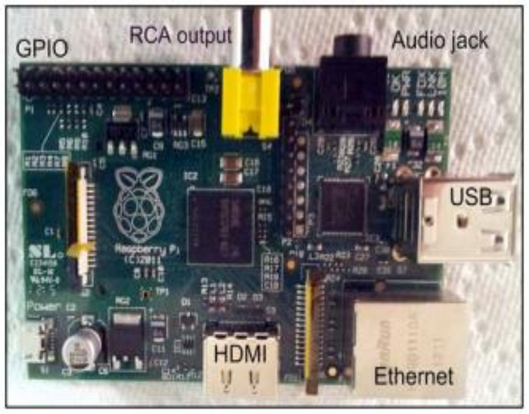


Fig. 7 Components of Raspberry Pi board.

V. EMBEDDED SYSTEMS IN AUTOMOTIVE APPLICATIONS

Since the 1970s, the increasing use of embedded systems in vehicles instead of hydraulic, pneumatic, mechanical parts led to complexity in automotive electric harness. The modern systems provide more comfort and safety for the driver and passengers. Those systems include steering, traction, braking, stability control and using entertainment equipment (Radio and navigation systems). So the need for communication protocols was originated from the desire to connect that large number of ECUs together for controlling the vehicle [1] without the need to complicate the wiring harness.

A. *Priority Buses (Vehicle Networks)*

Until the beginning of 1990s the data was transmitted through point-to-point links between ECUs [1], but there are problems found like: cost, weight, complexity, and reliability. So, many communication networks were found as follows:

1) Local interconnect network (LINs): in 1998 the four doors for BMW used (LINs) which caused reducing of the vehicle weight by 15 Kg [1].

2) Controller area network (CAN): in 1990s Mercedes used (CAN) which became the most commonly used protocol in automotive systems [1].

3) Vehicle area network (VAN): it has many same features as (CAN) but it's varies by the following: no need for bit-stuffing and in frame response (that is a node being asked for data answers in the same frame). It had been used by Peugeot and Citroen. As it is not adopted by the market and it was abandoned [1].

4) J1850 network: it is found in United States. It has two versions: 10.4 kbps single-wire version and 41.6 kbps two-wire version. It was replaced by LIN or CAN because of its high cost [1].

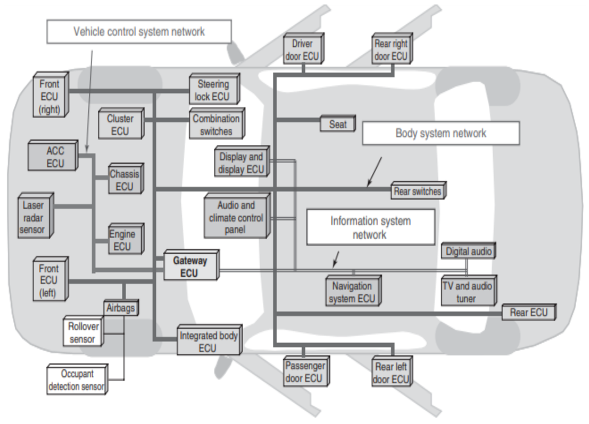


Fig. 8 ECUs in a typical vehicle.

B. Types of Communication Networks

The types of communication networks are: Time-Triggered (TT) networks and event-triggered networks [1].

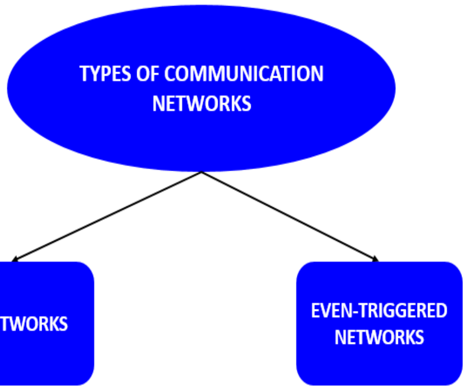


Fig. 9 Types of communication networks.

1) TT Network: It's characterized by activities driven by the progress of time. As it's more dependable it has been chosen for use for X-by-wire applications [1].

2) Event-Triggered Network: It's characterized by activities driven by the occurrence of events [1].

C. Types of Network Protocols

There are three types of network protocols of interest:

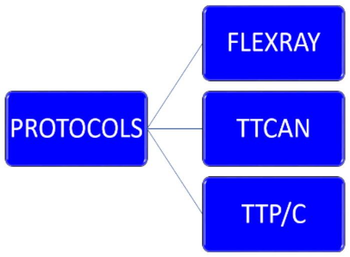


Fig. 10 Network protocols.

1) FlexRay: It's released in 2004 and developed by many companies like: BMW, Bosch, Daimler, General Motors, NXP Semiconductors, Freescale Semiconductor, and Volkswagen. It may be a bus, a star, or a multi-master. The FlexRay frame contains three parts: the header, the payload segment having up to 254 bytes of data, and the CRC having 24 bits [1].

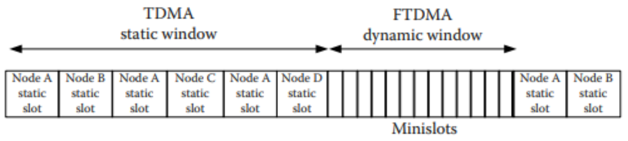


Fig. 11 Example of a FlexRay communication cycle with four nodes A, B, C, and D.

2) TTCAN: It's developed by Robert Bosch GmbH. Its data rate is up to 1 Mbps. It's the most common used protocol for vehicles because of its low cost compared with the produced performance [1].

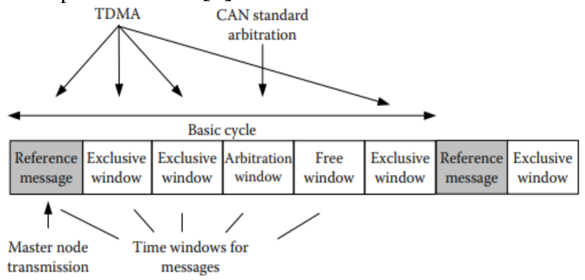


Fig. 12 Example of a TTCAN basic cycle.

3) TTC/P: It's not used for vehicles currently but it is used in aircraft electronic systems. We have gained experience from the years with TTP/C usage. Which is beneficial to FlexRay [1].

D. Requirements of Automotive Networking

1) Deterministic behavior: It's achieved when the measured performance of its services is predictable under a specified condition. It's characterized by specific nonrandom equations. For in-vehicle networks, the most important measured performance is message latency which is defined as the time interval starting when a transmitter node sends a message till that message is read by the receiver [1].

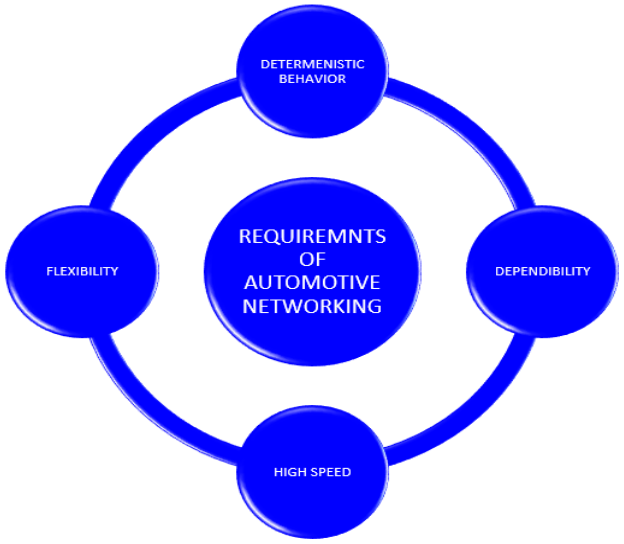


Fig. 13 Requirements of automotive networking.

2) High speed: For supporting multimedia, Internet-related applications and inter subnetwork communications. It's predictable that in the future the speed will be up to 100 Mbps for the backbone, for supporting global system integration with transferring large amounts of information, and one Mbps for subnetworks [1].

3) Flexibility: It's the most important requirement to be achieved by the network. It's required to produce a flexible vehicle architecture [1].

4) Dependability: It includes reliability, availability, maintainability and safety, which have been related to in-vehicle systems. It also includes integrity and confidentiality which became more important since vehicles have been more networked with the external world like: car-to-car, car-to-road communications, internet connection, integration in the car architecture, wireless vehicle access control and remote/wireless vehicle diagnostics [1].

VI. Conclusion

In the present article, an overview of the embedded systems was presented. The classification, types and requirements of the embedded systems were explored. The developer's skills were identified. Two simplified embedded systems examples were demonstrated. In addition, the application for the embedded systems in automotive systems were illustrated. The discussion addressed the protocols, systems requirements and performance of the embedded systems in automobiles. The article provides an adequate overview in a nutshell on the embedded systems for the reader.

Acknowledgment

I would like to acknowledge my supervisor Associate Professor Mostafa Yacoub for his support to my research. I would also like to acknowledge the Automotive Engineering Department/The Military Technical College for facilitating the resources to support my research.

References

[1] Nicolas Navit and Francoise Simonont-Lion, *Automotive Embedded Systems Handbook,* Roca Raton, FL: CRC Press, 2009.

[2] W. Wolf, *Computers as Components: Principles of Embedded Computing Systems Design*, 4th ed., Cambridge, MA: Elsevier, 2000.

[3] J. W. Valvano, *Introduction to Embedded Microcomputer Systems, Motorola 6811 and 6812 Simulation*, Pacific Grove, CA: Brooks/Cole-Thomson Learning, 2003.

[4] T. D. Morton, *Embedded Microcontrollers*, Hoboken, NJ: Prentice Hall, 2000.

[5] Mohit Arora, *Embedded System Design*, Austin, TX: Learning Bytes, 2016.

[6] Alexander Romanovsky, Elena Troubitsyna, Ilir Gashi, Erwin Schoitsch, Friedemann Bitsch, Computer safety, reliability and security: *SAFECOMP Workshops, ASSURE, DECSoS, SASSUR, STRIVE, and WAISE Proceedings*, Turku, Finland, 2019.

[7] Qamar Jabeen, Fazlullah Khan, Muhammed Tahir, Shahzad Khan, Syed Roohullah Jan, Farman Ullah, "A survey: Embedded systems supporting by different operating systems", *International Journal of Scientific Research in Science, Engineering and Technology(IJSRSET),* vol. 2, no. 2, pp 664-673, April, 2016.

[8] Mostafa Bazzaz, Ali Hoseinghorban and Alireza Ejlali, "Fast and Predictable Non-Volatile Data Memory for Real-Time Embedded Systems," in *IEEE Transactions on Computers*, vol. 70, no. 3, pp. 359-371, March, 2021.

[9] KCS Murti, "Networked Embedded Systems (NES)", in *Design Principles for Embedded Systems Transactions on Computer Systems and Networks*. Springer, Singapore, 2022.

[10] Mark penhaker, Martin stankus, Janki jonka, Petr Grygarck, "Design and Application of Mobile Embedded Systems for Home Care Applications", *Second International Conference on Computer Engineering and Applications*, pp. 412-416, Bali Island, Indonesia, March, 2010.

[11] Tim Wilmshurst. *An introduction to the design of small-scale embedded systems*, London, UK: Palgrave, 2001.

[12] Christos Profentzas, Mirac Gunes, Yiannis Nikolakopoulus, Olaf Landsieder, Maguus Almgren, "Performance of Secure Boot in Embedded Systems", *15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pp. 198-204, Santorini, Greece, May, 2019.

[13] Yuuki Furukarva, Toshihiro Yamauchi, Hideo Taniguchi, "Implementation and evaluation for sophisticated periodic execution control in embedded systems", *International Journal of Control and Automation*, vol. 3, no. 3, pp 87-106, September, 2011.

[14] Michael Barr, *Programming Embedded Systems in C and C++*, Sebastopol, CA: O'Reilly Media, 1999.

[15] T. Pan and Y. Zhu, *Designing Embedded Systems with Arduino: A Fundamental Technology for Makers*, Singapore, Singapore, Springer, 2018.

[16] Sai Yamanoor and Srihari Yamanoor, *Raspberry Pi Mechatronics Projects HOTSHOT*, Birmingham, UK: Packt, 2015.