

Data Acquisition and Storage in Engineering Systems

Cezar Liviu CERVINSCHI

Academy of Economic Studies, Bucharest, Romania
cezar@symbolsoftware.ro

Data storage represents a sensitive subject when it comes to a software system or a software application. Referring to engineering systems, data storage becomes even more challenging, since multiple functions must be accomplished and various tasks need real time response and high accuracy. While gathering information through data acquisition is relatively simple, with hardware and software equipment providing wide and complex documentation, data storage raises a set of issues, mainly due to communication channels, storage devices or software algorithms. In this context, the paper will focus on presenting some of the mostly used hardware devices and data transmission protocols in engineering applications, creating an overview over data manipulation and providing an implementation method, together with proposed solutions and a model of implementation regarding the topic.

Keywords: data storage, engineering, data acquisition, database, system, microchip

1 Introduction

The term ‘*engineering*’, taken into consideration as it is, is void from the very beginning: the American Engineers’ Council for Professional Development (ECPD) defines the term as ‘*the creative application of scientific principles to design or develop structures, machines, apparatus, or manufacturing processes, or works utilizing them singly or in combination, or to construct or operate the same with full cognizance of their design; or to forecast their behaviour under specific operating conditions; all as respects an intended function, economics of operation and safety to life and property*’ [1, 4]. Considering this rather wide definition, it is obvious that the area of research must be clearly specified.

The paper will focus on the electrical engineering branch, studying data acquisition and storage methods for electronic devices, hardware components and will provide a functional example, together with other possible implementation scenarios regarding both acquisition and storage of information.

Since most engineering-related software concentrate on resolving only one specific task (the task the system is designed to do), each engineering system needs and uses its own software platform, requiring special resources for both the development and operating processes. However, with the massive growth of nano-technology and electronic devices, lots of issues have been removed, since high performance microchips have been created and powerful development boards incorporate all the needed functionality in terms of connectivity, communication interfaces, memory types, signal converters or graphic display [5]. In addition to these hardware break through-s, software development environments have adapted to these changes and provide specialized classes and objects that manipulate data transmitted through these devices (e.g. Microsoft Visual Studio has specialized classes for manipulating threads, message queuing or different types of networks).

Regarding the data storage support, it can be of various types, regarding the system put into discussion. Since most of the microcontrollers have embedded memory, it comes natural for the engineer to use the

built-in memory available onboard. It is a good strategy, but is mostly used into loading and running the software the microchip uses, leaving the gathered data in a third state logic, the 'unknown / undefined' state. This is where the developer, through his code, must decide what to do with the gathered data and one of the safest ways of avoiding both data loss and memory overloading is to redirect information to a database system.

Arguments over using database technology in engineering systems are the mechanisms they provide in managing the consistency and availability of data, standardization of input data or high storage capabilities. Reuse of data can be done at any time, a simple dump or query interrogation retrieving information that afterwards may be listed on external support (e.g. .xls files), used in statistic processing or in automatic creating of log files.

2. Engineering systems

As previously mentioned, an engineering system is void without clearly specifying its functional area. Process automation has always been a challenge for engineers and researchers, since new and more sophisticated electric and electronic devices appear, and automated processing lines tend to replace manual labour. Terms like microchip, multi-agent system, PLC (Programmable Logic Controller), development board, I/O (input - output) have emerged and human society must adapt to them.

2.1. Microcontrollers

Microcontrollers are integrated circuits based on different types of architectures and which, depending on the system's functionalities, incorporates different components. Since the majority of microprocessors require a lot of

discrete components and integrated circuits to execute relatively simple functions, large and complex hardware structure results [2]. Therefore, new controllers have been developed and they incorporate the hardware structure as a 'black box' with input and output signals and the microchip as the box (containing Analog/Digital converter, comparator, Pulse Width Modulation, serial communication, internal clock, RAM / EPROM / EEPROM / Flash memory, programmable I/O).

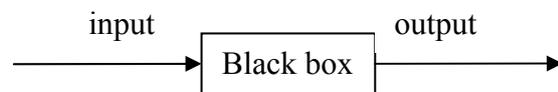


Figure 1. Black box structure

Loading the software on the microchip requires only a connection to a PC.

2.2. Microcontrollers' architectures

One of the most important elements that must be taken into consideration when analyzing a system is the Central Process Unit's (CPU) architecture [3].

a) Von Neumann architecture

Microcontrollers based on this architecture have a CPU with only one memory device used for both memorizing code and instructions and also the processed data. There is only one internal bus used for evaluating code functions (e.g. FETCH_OP()) and data. The two operations are executed sequentially and derive into a delay on operations response.

b) Harvard architecture

This type of architecture has separate memory devices for program code and for data, therefore separate buses exist, making possible the parallel execution of both processes. Digital Signal Processing (DSPs) use this type of architecture and, because of its high costs, it is not widely used.

c) CISC (Complex Instruction Set Computer)

Almost all microchips' CPUs rely on the CISC concept, meaning that over 80 instruction functions are built-in the

microchip's CPU, operating with addresses, registers or only allowing certain operations. A big disadvantage would be the usage of small and memory consumption functions rather than a single macro-instruction.

d) *RISC (Reduced Instruction Set Computer)*

As its name states, this type of architecture uses a small set of instructions, making the CPU efficient and rapid. Its rapidness is increased by including a separate instruction pipeline, thus accelerating data transmission.

2.3. Communication

a) *RS232*

RS232 is a series of standards for serial binary single-ended data and control signals, connecting between a Data Terminal Equipment (DTE) and a Data Circuit-terminating Equipment (DCE). It is commonly used in computer serial ports, defining electrical characteristics, signals meanings and physical size and pinout of connectors.

A serial port is a serial communication physical interface through which information transfers from one device to another as a bit stream. A limitation of the standard is the maximum length of the data cable, which cannot be bigger than 10 meters.

The RS232 connector was originally developed to use 25 pins. In this **DB25** connector pinout, provisions were made for a secondary serial RS232 communication channel. On personal computers, the smaller **DB9** version is more commonly used (Figure 1) [4].

b) *RS485*

Similar to RS232, RS485 sends data streams through serial ports using a pair of wires to send differential signal over distances up to 1200 meters. The differential signals make RS485 very robust and more appropriate to industrial applications, while its noise immunity

and long-distance capability adapt perfectly to the industrial environment [6].

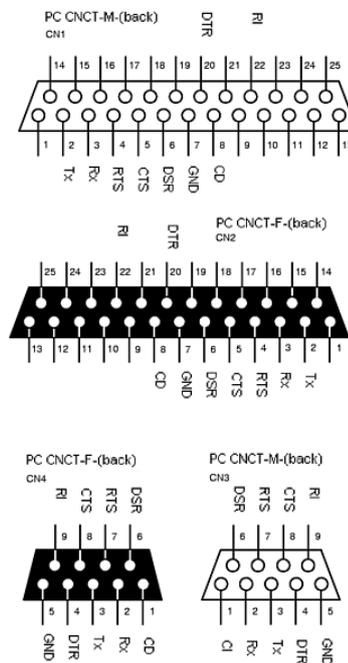


Figure 2. RS232 pinouts

3. Data acquisition and storage

3.1. Overview

The problem of data acquisition and storage is a complex one in engineering systems. While, as previously mentioned, microchips do own a small memory of their own (mostly used for storing the software application and functions the system performs), this memory, be it volatile or not, is not capable of storing all the data streams as they are registered and measured by the system.

Considering a high sampling rate for read data, the amount of information the microchip would have to memorize would increase within seconds, causing memory overflow and system crashes. To prevent such things, the software developer must decide what to do with the data.

First scenario states that the developer wants to discard the data, meaning that in this case he would just read the data, process it, if necessary, by holding

it into a memory buffer and, while new data flows through the system, the old data would disappear. The maximum output of this kind of approach would be receiving a graphical result of the data (e.g. measured parameters).

A second scenario would imply that the developer wants to have a hardcopy of the data flow somewhere on an external device. In this case, using a database management system represents a good approach, since no data would be lost, due to the DBMS's automatic backup tools. Therefore, as data flows through the system, the developer can set a sample rate (for example, the sample rate the system gathers data with) and insert into the database the desired information (data content, parameters, errors). This way, after stopping the device and analyzing data, engineers would know the parameters of the machine or the problems that occurred. This database-aided approach can be extremely useful in debugging an engineering system, since the user can track both system problems and productivity issues.

Depending on the user and on the type of the application he develops, the

Preamble (7 bytes)	Starting Delimiter (1 byte)	Destination Address (2 or 6 bytes)	Source Address (2 or 6 bytes)	Type or Length (2 bytes)	LLC Header and Information Field (0 – 1500 bytes)	Pad (0 – 46 bytes)	Frame Check Sequence (4 bytes)
-----------------------	-----------------------------------	---	--	--------------------------------------	--	--------------------------	---

Table 1. Ethernet frame format

The Preamble is used by the receiver to allow establishing the connection. The Starting delimiter is a simple flag that indicates the start of a frame. The type or Length field is the field that sets both the Information field and the Pad field (which is actually a compensation field for the Information field – if it has less than 46 bytes, the Pad field compensates it). Finally, the Frame Check uses a CRC-32 (Cyclic Redundancy Check) polynomial code to check the frame for

database management system must carefully be chosen. Whereas with CPU applications developed in C# technology, it would be best to use MS SQL Server, in applications that need lots of disk space (e.g. large databases or data warehouses) Oracle can be used, while with small or medium size engineering applications even the open-source MySQL DBMS can be used.

3.2. Transmission protocols

Having to choose which transmission protocol to use could prove to be a challenging task since the market provides several such protocols, each one of them meeting different demands, according to the engineering system they have to support.

a) The Ethernet protocol

Developed in the mid 70's as a part of the Ph. D. dissertation of Robert Metcalfe and standardized in the early 80's [7], when IEEE (Institute of Electrical and Electronics Engineering) started project 802 to standardize local area networks – LAN. Ethernet stations communicate through data frames, which have the following structure:

errors. In order to send a frame over a network through the Ethernet protocol, a station listens to the Ether. If it is busy, the station will enter a waiting queue and, after the thread gains access, the frame starts the transmission.

b) The GSM protocol

Similar to the Ethernet protocol, the GSM (Global System for Mobile Communications) protocol can be used in data transmission or acquisition systems.

The standard has been developed by the European Telecommunications Standards Institute (ETSI) in order to describe second generation (or 2G) digital cellular networks' technologies. Clearly defined for mobile communications, GSM is a cellular network (data is sent through the network from one cell to another) that can easily be implemented on engineering systems that need data acquisition. It can be implemented using a GSM modem as a data dispatcher and connect to several other devices. Nowadays, with 3G networks expanding daily, this protocol should really be taken into consideration in any type of system that needs data transmission, although it has its limitations.

c) The Wi-Fi protocol

IEEE 802.11 is a set of standards for implementing wireless local area network (WLAN) computer communication in the 2.4, 3.6 and 5 GHz frequency bands [8]. Empirically, a WLAN network could be considered a wireless version of Ethernet. Over time, the protocol has improved, giving rise to different variations of it: 802.11a is a version that operates at 54 Mbps and is considered the favorite wireless LAN protocol for IP telephony, even though it has a major disadvantage – the network coverage only goes up to 30 m. A second version of the protocol, 802.11b, operates at 11 Mbps but has a major improvement in the network coverage – almost 100 m – making it suitable for public hotspots or small environments. An even newer version, 802.11g, works at 54 Mbps and is considered a high speed replacement for 802.11b.

d) The ZigBee protocol

ZigBee is a specification for a suite of high level communication protocols using small, low-power digital radios based on an IEEE 802 standard for personal area networks [8]. Since this protocol is extremely versatile, it has

been mostly implemented in engineering applications concerning fields like home automation, smart energy consumption systems, telecommunications, medical appliance or smart remote control. Thanks to its low-cost and low-power usage, the protocol allows longer life with smaller batteries, high reliability and extensive range of products. Standard ZigBee microchips have a built-in flash memory of 60 to 256 KB and they operate in all the major radio bands: industrial, scientific and medical (ISM) with data transmission rates varying from 20 to 250 kbps.

After analyzing this short and generic overview over some of the mostly used data transfer protocols, a conclusion can be made, and that is that electronic communication systems are evolving with an astonishing rhythm, from old-fashioned network cables and adapters, to cutting edge wireless technologies, high speed data transfer rates and glamorous GSM technologies for next generation gadgets.

In the same context, engineering systems become more and more sophisticated, integrates high-end technology and deliver real-time responses. Software support must keep up with the hardware devices, must create frameworks that would deliver adaptable software for these devices and integrate in the scientific process of technology development.

For an engineer, having to choose between a data acquisition protocol over another could become really difficult. Every decision must be fully compatible with the kind of task the system is projected to perform. Thus, choosing a DBMS to store the collected data would prove itself an even more challenging task, considering the compatibility issues that might occur.

4. Case study

For experimental purpose, I have developed a small robot whose task was to resolve a maze using proximity sensors and

the functions loaded inside the CPU. The maze had vertical walls and the robot would recognize black lines on the floor of it. Guided by these lines and by the sensors' received signals, the robot executes 180 degrees spins, left or right turns and stops, in order to get out of the maze.

4.1. Solutions

The solution for resolving the maze proposes loading a Visual Studio C# program inside the microcontroller. When the robot starts running, the proximity sensors will search a wall on the right side, guiding the robot's moves. If the sensors detect no wall on the right side, the robot would turn right. If the sensors detect a wall on the right side and in front of the robot, he would turn left. The movement is set by the engine driver, which sends a signal to the two wheels. While moving, the communication is done through radio waves and real time information is sent to the external display device (a PC monitor). The emitter (placed on the robot) and receiver (connected to the RS232 PC interface) work on the same frequency with a similar baud rate, being fully compatible.

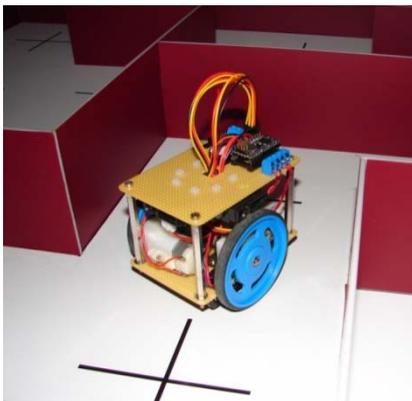


Figure 3. The robot inside the maze

4.2. The sensorial system

In order to navigate through the maze, the robot uses three Sharp GP2D120 distance sensors for wall detection. They are highly precise and very commonly used in robotic systems. The detection distance keeps between 40 and 300 mm, allowing the robot to both keep its trajectory and to detect the maze's structure. The sensor is immune to the interference of ambient light and offers excellent indifference to the detected object's colour, therefore making the detection of a black wall in full sun light possible.

4.3. The ATmega168 Microcontroller

The autonomous robot is built around a CPU Module who supports an AVR ATmega168 microcontroller (MC):

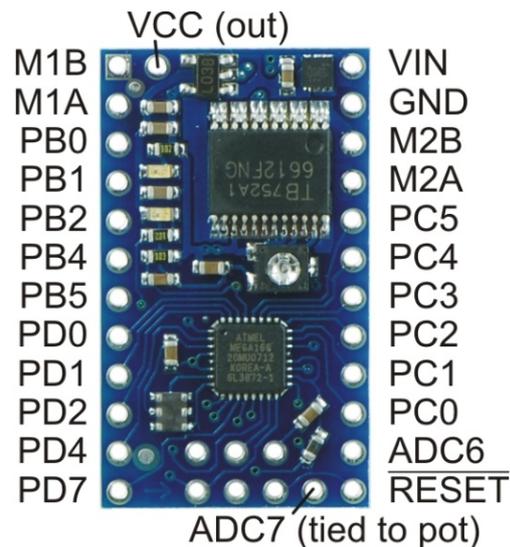


Figure 4. The CPU Module (pin's configuration)

The MC is based on the RISC architectural model. It has a 131 set of instructions and 32 x 8 general usage registers which are directly addressable by the Arithmetic and Logic Unit (ALU), allowing the access of two independent registers with one instruction, therefore the MC being up to 10 times faster than CISC-based conventional MCs.

The AVR architecture has two main memory spaces: Data Memory and Program Memory plus a Data Storage Memory. The

left, the robot must find a wall in front of him whereas, to turn 180 degrees, the system must find walls in both his left and right sides.

4.7. Acquisition and storage

Regarding the data acquisition and storage issues, they remained at the developer's choice, meaning that he had the alternative to save or discard the received data. For the presented solution, since the purpose was creating a functional autonomous robotic system rather than a collection of data, data would not be saved on an external storage device.

While the robotic system is running through the maze, the alphanumeric display would indicate its state through the transmitted data. The robot moves inside the maze using the procedures described in section 4.6 and the SerialPort Terminal application displays the state messages on the computer.

A completely different scenario would have been if the data needed on-line or offline processing. Since the data acquisition software program has been developed using Microsoft's Visual Studio C#, a compatible and convenient DBMS for data storage would have been MS SQL Server.

Through an SQL Connection, when transmitting data, the program would automatically send the log messages and the parameters to a database table. Using a convenient baud rate, the software system would insert database recordings inside a specific table. Such a table structure is presented below:

Table 3. Proposed database table

Field Name	Data Type	Functionality
Acq_id	int	Recording identifier
Port_name	varchar	The opened port for the recording

Data_bit	varchar	Transmission data (bits)
Parity	int	Transmission parity
Baud_rate	varchar	Transmission baud rate
Stop_bit	bit	Transmission end bit (flag)

Using an administration module (for example, a web interface), any user could be able to manipulate and use the database records. Since every recording gives details over all the communication parameters, exporting the data set would give important information over the engineering system's functionality, over the status of the mechanisms, transmission parameters and system errors, if they occur. This data can easily be processed and the functionality of the engineering system significantly increased.

Extrapolating this idea, it is obvious that any engineering system can be improved by adding database storage support that can save all the parameters' statistics and functional log messages. Dealing with a large amount of information makes it almost mandatory that acquisition systems must have a backup support, mostly due to microchips' limited memory space. A database management system would most definitely assure disk space, backup utilities, data extraction tools and a completely functional work flow for the system: Data acquisition --> Database storage --> Data processing. This would also imply that system engineers would be aware at any time about the system's parameters and be able to take important decisions and might even change the system's structure, if the procedure calls it.

5. Conclusions

Building an engineering system is not necessarily a matter of quantity but a matter of quality. Such a system requires solid grounds in different areas of research,

but different tasks can be split between more specialists in order to achieve full potential from each engineering module. The paper has presented some of the most used microchips architectures and communication protocols. Automating a process proves, in this context, that understanding the system's requirements is essential into choosing a project development work flow.

After analyzing the functional needs and choosing the pieces that must be put together in order to develop the system, engineers must work together with software developers and system administrators at calibrating and tuning the whole system. Because the project to be delivered is not only a solution to a specific task, but might even be a worldwide innovative system, everything must go hand in hand and work perfectly. Examples of such automated systems can be found in medicine, home automation, smart energy consumption methods, remote control devices, building automation or retail services. All these research areas need data acquisition systems and storage space.

An important role in the development part comes to the system administrator, which has to decide which hardware platform to use, which operating system to install on the server and to decide the database management system to use for data storage. The DBMS is an important piece of the puzzle since a fraction of a second delay could compromise an entire period of data acquisition and research work or, worse, a company's productivity, therefore it must be carefully chosen in full compatibility with both hardware and software equipments.

In this context, the paper has presented possible ways of implementation for database management systems in data acquisition applications, relating the complexity of the engineering system to the DBMS to choose. In addition, a case study has been analyzed

from a functional point of view to data flow and data storage management. The research results state that a good engineering system needs a solid database backup support, together with high storage capacity and speed of response from the database server.

The presented case study represents an example of such a system, considering that autonomous robotic systems are studied intensely in matters of developing microcontrollers which, afterwards, can become the starting point of more complex systems such as robotic arms and other anatomical parts or intelligent devices for different areas of life. In the same time, the presented software application represents a good starting point for a more complex and highly developed data acquisition software system, together with the transmitter / receiver wireless connection (which, of course, could have been implemented using one of the listed protocols, not necessarily the radio wave one).

In conclusion, a good engineering system must work hand in hand with a database system since data must not be lost but processed. Important break-throughs have been made in the domains of nanotechnology and electronics resulting in high speed data flow devices, and this speed must now be transferred to databases in order to capture and store all data streams that flows inside these devices. Having implemented these features would make engineering, software development and system administration go hand in hand in all domains of life.

References

- [1] "The Engineers Council for Professional Development", Science, Vol. 93, No. 2426, Jun. 1941, pp. 611-612.
- [2] F. M. A. da Silva, P. H. I. A. de Oliveira, "Development of a data acquisition system for light airplanes flight tests", 18th International Congress of Mechanical Engineering, 2005

- [3] G. Dudek and M. Jenkin, Computational Principles of Mobile Robotics, Cambridge University Press, Cambridge, 2010
- [4] Wikipedia, <http://en.wikipedia.org/>
- [5] MicroElectronika <http://www.mikroe.com/>
- [6] B&B Electronics <http://www.bb-elec.com/>
- [7] J. F. Shoch, Y. K. Dalal, D. D. Redell, R. C. Crane, Evolution of the Ethernet Local Computer Network, IEEE Computer, Vol. 2, No. 18, August 1982, pp. 14 – 26.
- [8] Wikipedia, <http://www.wikipedia.org/>



Cezar Liviu CERVINSCHI graduated from the Faculty of Automatics and Applied Informatics of the Transilvania University of Brasov in 2009. With a masters degree in Information Technology, his interest in web technologies, web development, integrated software systems and database administration brought the motivation for advanced research in this field of science, since his main target is building robust, reliable and complex software solutions. At present, he is a PhD candidate at the Academy of Economic Studies in Bucharest at the Economic Informatics department studying web technologies, databases, online security, e-business and e-marketing.