Conclusion

So, software for web-based education system is developed by web-programming language PHP, MySQL database and web design languages HTML and CSS.

Access to this EIS can be granted on a local network or the Internet. EIS located on a web server that provides protection against unauthorized access.

References

1. Robert I.V. Modern Information Technologies in Education / Robert IV-M.School Press, 2009. - 204 c.
2. Tim Converse, Joyce Park and Clark Morgan.PHP 5 and MySQL.The User's Bible / Tim Converse, Joyce Park and Clark Morgan, 2014. - 1219 c.
3. Horton U. E-learning: tools and technology. / Horton U., Horton K. Trans. With the English. - Moscow: KUDITS-IMAGE,2009. - 640 p.

UDC 681.52

MAIN PRINCIPLES OF BUILDING SYSTEM FOR MONITORING OF RECURRENT LARYNGEAL NERVE BY SINGLE-BOARD COMPUTER RASPBERRY PI MODEL B

Mykola Dyvak[1)], Volodymir Tymets[2)]
Ternopil National Economic University
[1)] DsC., professor; [2)]PhD student

I. Problem

Nowadays, the number of surgeries for neck organs constantly increases. The greatest danger during this surgery is injury of the recurrent laryngeal nerve (RLN). Therefore, engineering software complex tools are used to reduce the risk of its damage. The basic principle of functioning of such devices is as follows: stimulation of surgical wound tissues by alternating current; registration and software process of stimulation results to identify an informative characteristic of tissues type [1].

II. The goal of the work

A scheme of application of new method and hardware for development of RLN monitoring system is described in the paper [2]. This approach based on stimulating the tissues of surgical wound by an alternating current. Then, result of simulation registering by sound sensor which is implemented into end tracheal tube above vocal cords. The main purpose of this paper is description of principles of building RLN monitoring system by single-board computer Raspberry Pi.

III. Specifics of implementation Raspberry Pi

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries[3]. Computer Raspberry Pi can work under control of Linux operation system. Via this operating system, node js and npm ecosystem can be used. Node js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Npm - node.js package ecosystem, is the largest ecosystem of open source libraries in the world [1].

IV Principles of realizations

The basic principle of implementation consists of two parts. The first part is responsible for the recognition of sound and its transformation. Transformation of sound signal executes in sound encrypt module by fast fourier transform method[1].

The second part of the program generates sequential square wave signal. The signal must be generated continuously with constant current strength and with operation loop factor α.

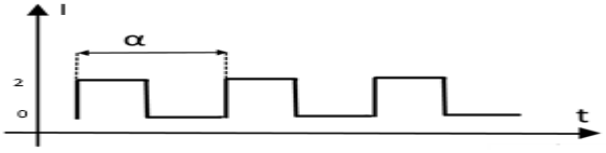

Figure 1. The visual display of sequential a meander of signal.

This coefficient of duty must be variable and specified by a software. Current strength of a signal must be 2 mA and the coefficient of duty must vary from 1/500 of seconds to 1/20 of seconds for generating correct signal.

Conclusion

Two main parts of the program for monitoring of recurrent laryngeal nerve by single-board computer Raspberry Pi model bare shown in the paper.

References

1. M. Dyvak, V. Tymets. "Emulation of programming environment for single-board computer Raspberry Pi at monitoring the recurrent laryngeal nerve", MEMSTECH, Lviv, pp.35-37, 2017
2. M. Dyvak et al., "Spectral analysis the information signal in the task of identification the recurrent laryngeal nerve in thyroid surgery" Przegląd Elektrotechniczny, vol. 89, no. 6, pp.275- 277, 2013.
3. Bush, Steve (25 May 2011). "Dongle computer lets kids discover programming on a TV". Electronics Weekly. Retrieved 11 July 2011.

UDC 004.4

NON-CLASSICAL PROPERTIES OF PROGRAM-ORIENTED LOGICS

Mykola Nikitchenko

Taras Shevchenko National University of Kyiv, DsC., professor

Software development is a grateful area of logic application. Logics can be used at every stage of development cycle, in particular, during requirement analysis, specification, design, verification, and testing.

To be successful, such logics should adequately represent essential features of the development stages. Among various logics, oriented on software development, the central place belongs to logics describing main features of computer programs. In particular, we identify the following features:

- *partiality*: programs may be undefined on some input data;
- *usage of complex system of data types*: arrays, hash-tables, linked lists, semi-structured data, etc.;
- *non-determinism*: programs can evaluate to different results on the same input data;
- *possibility of transformation* of abstract specifications (programs) to more concrete ones.

It is naturally to expect that implementation of these features in program logics will lead to substantial changes in reasoning rules. Even more, in some cases we will obtain non-classical reasoning rules.

In this paper we investigate non-classical properties of program-oriented logics. We restrict ourselves to considering the following program properties: 1) *partiality*, 2) *unfixed and unrestricted arity*, 3) *unessential variables*, 4) *sensitivity to unassigned variables*, and 5) *non-determinism*.

Let us discuss the impact of such properties on program-oriented logics [1, 2].

1. *Partiality* of program, functions, and predicates requires to change the consequence relation for the logic. Instead of classical relation which leads to totally-true predicates, we have to consider a dual consequence relation – the irrefutability relation – which leads to irrefutable predicates. As a result, we cannot rely on such traditional reasoning rules as modus ponens, cut, resolution etc.

2. *Unfixed and unrestricted arity*. In classical predicate logic we know precisely the number of variables upon which a predicate depends. This property permits to construct new predicates by substituting into initial predicate (into a formula) a fixed number of functions (terms). But if a predicate does not have fixed arity, we are obliged to change the definition of substitution which is formalized as superposition [1]. Such a superposition has different parameters indicating on function substituted into a predicate. This leads to new extended language of a logic [1, 2], and, as a result, to new language properties. If an arity is unrestricted we cannot say what variables are essential to a predicate and what are not. Therefore, those classical properties based on a known number of essential variables such as, say, coincidence lemma, properties of closed sentences, are not valid any more.

3. *Unessential variables*. In classical predicate logic it possible to identify free variables in a formula (which are essential variables), while other variables are unessential and can be used, for example, to substitute a quantified variables to obtain a formula variant. Such variants are necessary for different formula